Imperial College of Science, Technology and Medicine
University of London
Department of Computing

PROOF THEORY AND COMPUTER PROGRAMMING

The Logical Foundations of Computation.

by

Ruy José Guerra Barretto de Queiroz

January 1990

A thesis presented to the University of London
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy (Ph.D.)
and the
Diploma of Imperial College (D.I.C.)
in
Computing Science

Nel mezzo del cammin di nostra vita
mi ritrovai per una selva oscura
chè la diritta via era smarrita.

Ah quanto a dir qual era è cosa dura
esta selva selvaggia e aspra e forte
che nel pensier rinova la paura!

Tant'è amara che poco è più morte;
ma per trattar del ben ch'io vi trovai,
dirò dell'altre cose ch'i' v'ho scorte.

DANTE

To my parents

Geraldo & Joana d'Arc,

and

to my 'Beatrice'

Pascale

who probably think they were of no help

Table of Contents

Acknowledgements xiv

Abstract xvi

Preface xviii

1 *Introduction* **1**

1.1. Motivation 1

1.1.1. Semantics and Programming 3

1.1.2. Frege's notions of definition: constructive and analytic 6

1.2. Setting the Aims 7

1.3. General Overview 8

1.4. General Remark 10

2 *On Frege's Notions of Definition* **11**

2.1. Preamble 11

2.2. Introduction 13

2.3. Frege's View 16

2.4. Definitions in Recursion Theory 18

2.4.1. The existence theorem and the logical analysis 19
2.5. Conclusions and Final Remarks 26

3 *A Critique of Proof-Theoretic Semantics* **29**

3.1. Preamble 29
3.2. Introduction 30
3.2.1. Reduction Rules and the Inversion Principle 32
3.2.2. Prof. Prawitz's 'elimination from introduction' general procedure 36
3.3. Prof. Prawitz's Development 49
3.3.1. Prof. Dummett's Aspects of Use 50
3.3.2. Insufficiency of Introduction Rules 51
3.3.3. The Rôle of Reduction Rules 52
3.3.4. Unconvincing Proof-theoretic Semantics 60
3.4. Verification and Meaning 62
3.4.1. Wittgenstein and Verification 62
3.4.2. A Refinement: Canonical (Direct) Proofs 67
3.4.3. Wittgenstein's Early Intuition Prevails 68

4 *Wittgenstein and the Paradigm 'Meaning-Is-Use'* **81**

4.1. Introduction 83
4.2. Wittgenstein's Operational View of Meaning 86
4.2.1. Against a 'static' account of representing meaning 87
4.2.2. On the close interplay between rules and meaning 90
4.2.3. On the rôle of logical reduction 95
4.2.4. The nature of use in natural language 96
4.3. Meaning-is-Use and Consequences 98
4.4. The Evolution of Wittgenstein's Philosophy 100
4.4.1. Change in Spirit but not in Early Intuition 102
4.4.2. 'Disturbingly' Original Intuition 108
4.4.3. The Objection to the View on 'Meaning-is-an-Object' 113
4.4.4. The Aspect of Consequences and the Notion of Language-Games 117

4.5. Games and Commitments: The Rôle of Consequences 119
4.6. Ordinary Language 122
4.7. Early Intuition and Modern Mathematical Languages 123
4.8. Meaning, Use and Consequences 125
4.8.1. Meaning and Use 128
4.8.2. Meaning and Consequences 129
4.8.3. Use and Consequences 130
4.9. Language-Games, Context of Utterances and the Consequences 133
4.10. Concluding Remarks 138

5 *Formalising the Notion of Use/Utility* **141**

5.1. Preamble 142
5.2. Introduction 146
5.3. Formalising Constructive Mathematics 157
5.3.1. The Crucial Distinction: Propositions and Judgements 159
5.3.2. Forms of Judgement 162
5.3.3. Canonical (Direct) Proofs 172
5.3.4. Propositions and Types 180
5.3.4.1. Equality in Intuitionistic Type Theory 187
5.3.4.2. General Rules of Equality 193
5.3.4.3. Hypothetical Judgements and Families of Types 194
5.3.4.4. Finite Types 195
5.3.4.5. Cartesian Product of a Family of Types 196
5.3.4.6. Disjoint Union of a Family of Types 198
5.3.4.7. Disjoint Union of Two Types 200
5.3.4.8. Natural Numbers 201
5.3.4.9. Lists 202
5.3.4.10. Wellorderings 204
5.3.4.11. Universes 207
5.3.5. Meaning and Utility 212
5.3.6. Neither (Strict) Atomism nor (Unmanageable) Holism 214

5.4. Type Theory and The Meaning of Logical Constants 218

5.4.1. Wittgenstein's Early Awareness 219

5.4.2. Proof Theory, Meaning and Language-Games 222

5.4.3. Game-Theoretical Semantics and Semantical Games 227

5.4.4. Normalisation and Semantical Games 228

5.4.5. A Reflection on Public vs. Private Mathematics 232

5.5. Incoherences of Intuitionistic Type Theory Due to Canonical Value-Based Semantics 232

5.5.1. Definition of Arithmetical Functions 237

5.5.2. Functions and Operations 244

5.6. Final Remarks 252

6 *'Meaning-As-USE' Type Theory and Computer Programming* **255**

6.1. Introduction 255

6.1.1. Equality Rules of ITT 257

6.1.2. Elimination on the Constructors 260

6.2. Derivability of Elimination Rules in Type Theory 264

6.3. Non-Inductive Types 270

6.3.1. Induction on the Π*-type and on the* Σ*-type* 272

6.3.2. Π*-types and* λ*-theories* 276

6.4. Inductively Defined Types 276

6.5. Adding a Fixpoint Operator to MAUSE 279

6.6. Natural Numbers 286

6.7. Parameterised Types 293

6.8. Final Remarks 309

7 *'Meaning-As-USE' Type Theory and Abstract Data Types* **313**

7.1. Preamble 313

7.2. Introduction 314

7.2.1. Frege's notions of definition: constructive (abbreviatory) and analytic (presentatory) 315
7.3. Algebraic Specifications 318
7.3.1. Type Definitions/Presentations 319
7.3.2. Hidden Functions 323
7.3.3. Completeness of Axioms 325
7.3.4. Parameterisation and Error-Propagation 331
7.3.5. Implementation 333
7.3.5.1. Implementation and **Lists** *vs.* **Stacks** 339
7.4. Theory Presentations and Type Presentations 345
7.4.1. The λ-theory of Prof. Barendregt and the Π-type of λ-terms 345
7.4.2. Logic as Theories 350
7.5. Concluding Remarks 351

8 *Epilogue* **357**

8.1. Evaluation of the Results 357
8.1.1. Definitions as Presentations 359
8.1.2. Weaknesses of Gentzen-style Semantics 359
8.1.3. Verification and Consequences 360
8.1.4. Wittgenstein, Consequences and Meaning 360
8.1.5. Semantics with no Reference to (Numerical) Values 361
8.1.6. The Rôle of Proof-Rules in Type Theory 361
8.1.7. Computer Programming and Infinity 362
8.1.8. Abstract Data Types 362
8.1.9. The Resulting Formal System 364
8.2. Perspectives on Further Work 365

Appendix **367**

'Meaning-As-USE' Type Theory 368

Bibliography **379**

Index of Names 399

Index of References 401

Acknowledgements

To my family, my father Geraldo and my mother Joana d'Arc, my brothers Flávio (who showed his presence in the right time), Marcos and José Romero, and my sisters Simone and Sybele, who I believe cannot imagine how much they have helped.

Thomas S. E. Maibaum. What can I say about someone who could be so wise and supportive even in the peaks of less enthusiastic phases ? I thank you deeply because you are special, as is your lovely family, Jan, Shula and Michael for whom I have learnt to feel special affection.

Many people in the Department of Computing have given significant help and support, either personal or intellectual, and in some cases both. I would like to mention a few names that come to my mind right now: Dr Michael Smyth (very especially), Mr Martin Sadler (especially), Dr Samson Abramsky, Dr Axel Poigné, Professor Dov Gabbay.

Professors Per Martin-Löf (with my deep admiration and respect), Peter Aczel, Giovanni Sambin, Georg Henrik von Wright, Peter Hancock, Michael Levy, Newton da Costa, Paulo Veloso, Dale Miller, Georg Kreisel, Donald Gillies, Henk Barendregt, Gershon Weiler, Elisabeth Leinfellner, Jeffrey Zucker, Sergio Moravia, Paul Gochet, Raymond Turner, to whom and from whom and I have sent as well as received very useful letters, comments and/or papers, shall be omnipresent in my whole intellectual mythology, with very special places as admirable human-myths. So will Professors Paulo Cunha,

Clylton Fernandes and Manoel Agamemnon Lopes for what they represent and stand for in the back of my mind.

To colleagues I would like to say thanks for all we have shared (alphabetical order): Mario Benevides, Marcos Costa, João Cunha, Hugo Fuks, João Gondim (thanks for commenting on a draft chapter), Paul Jeremaes, Samit Khosla, Celso Niskier, Marcelino Pequeno, Jeremy Pitt, Mark Ryan, C. C. Tam.

It was a most exciting experience to participate in a project such as GENESIS, ESPRIT 1222(1041), CEC, where people like Will Harwood, Mark Dawson, Leen Helmink and René Ahn, were inspiring and stimulating, each in their own way with style.

Administrative help from the Department would not have been as good without the energy of people like Linden Rice, Kim Harrison, Fiona Cantell, Rosalind Smith, Caroline Taylor, Ann Halford and Ruth Bradshaw, to whom I feel very grateful.

I am sure many will agree that it is a special privilege to have Mrs. Ellen Haigh as librarian of the Department. I have many reasons to praise her great sense of kindness, finesse and admirable competence.

Financial support has come from 'CNPq' (*Conselho Nacional de Desenvolvimento Científico e Tecnológico*), Brazilian national council for scientific and technological development, grant 20.2724/84-CC, with a notorious service, and from GENESIS, ESPRIT Project 1222(1041), CEC. I feel immensely grateful to such respectful institutions.

Last, but not least, Pascale who has provided me with the taste of my own fantasy of Dante's spirit, being more than a mere *Beatrice* by showing her presence all the way through our everyday excursions through the magic universe of *Inferno*, *Purgatorio* and *Paradiso*.

Abstract

Frege's Realism seems to have prescribed a particular way of thinking about language and meaning, traces of which are found rather sharply even in the most 'radical' attempts at an 'anti-realist' theory of meaning for the language of mathematics. The model of 'object and designation', 'expression and (numerical) value', seems to have remained an essential ingredient of most of the theories of meaning, be it a *truth* or a *proof*-based theory.

The profound and most remarkable teachings on an utterly distinct way of thinking about meaning offered by the great Austrian thinker Ludwig Wittgenstein appear to provide an immense new world of ideas regarding the nature and the structure of the mathematical activity which does not seem to have been properly explored so far. By attempting to 'cure us from the temptations to look about us for some object which we might called "the meaning" ' (*Blue Book*, p. 1), Wittgenstein appeared to have demonstrated that meaning can be much better understood as something ultimately distinct from an 'object', a '(numerical) value', etc.. And this appears to constitute an enormously more convincing alternative to his own *truth*-based early account of meaning given in the *Tractatus*, as well as to most of the alternatives to that early 'atomistic' theory, which despite avoiding the notion of *truth* seem to have remained deeply rooted on the conception of meaning based on the model of 'object and designation', 'expression and (numerical) value', etc..

Amongst the most influential research introducing techniques of proof theory to explain typical notions of computer programming was the work developed by Prof. Per Martin-Löf on the formal system called *Intuitionistic Type Theory*, suggesting a certain connection between constructive mathematics and computer programming. The fact that it is presented in proof-theoretic natural deduction style combined with the idea that *propositions-are-types*, motivating the use of operations

(computations) to explain semantical notions, and having such connection between constructive mathematics and computer programming as the underlying general objective, has attracted the attention of those interested in the logical foundations of the notion of *computation*.

One of the key points seems to have been the 'internalisation' of the meta-level *normalisation* (*reduction*) rules, as far as the clarification of the relation between proof theory and meaning and its consequences for a better understanding of notions dealt with in computer programming are concerned. However, there does not seem to emerge from the explanation of those 'internalised' rules the expected emphasis on the important part played by those rules with respect to the semantics of logical constants, because the account of meaning is kept within Gentzen's tradition which says that the *assertability conditions*, represented by the schema of *introduction* rules, capture the meaning of a logical connective. For that reason we believe that a strong semantical argument justifying that the whole formal system is not just mere notation is still missing. As a theory of meaning *Intuitionistic Type Theory* still lacks the force of a truly convincing alternative to the classical (realist) semantics of the mathematical language. Indeed, the idea that the meaning of a proposition is given by its *conditions for assertion*, which seems to characterise most of proof-theoretic 'verificationistic' semantics, including *Intuitionistic Type Theory*, is itself rather questionable.

Here I attempt to demonstrate that Prof. Martin-Löf's endeavour to provide an anti-realist formulation of meaning for the mathematical language, although essentially rooted in Frege's (realist) conception of meaning as given by '(numerical) values', can still be reformulated to a point much closer to Wittgenstein's *use*-based 'definition' of meaning, where there is no strong attachment to the usual 'object and designation', 'expression and (numerical) value', etc., models of understanding of the concept of 'meaning'.

From a rather different semantical point of view I present an alternative *Type Theory* which may be more appropriate, at least regarding the connections between proof theory and meaning, and, as a natural consequence, the connections between proof theory and computer programming where rules and meaning have a close interplay. That is the formal system named *'Meaning-As-USE' Type Theory* which has been developed as a formal counterpart to a particular reading of Wittgenstein's later account of propositions where the 'explanation of the *consequences*' of a proposition seems to play a fundamental semantical rôle. The meta-mathematical device of *normalisation*, which seems to have been developed to explain formally the effect of *elimination* inferences on the result of *introduction* steps, is taken as a formal counterpart to the just mentioned 'explanation of the *consequences*', as opposed to the more usual interpretation of later Wittgenstein's 'language-games', Prof. Jaakko Hintikka's *semantical games* (cf. *Game-Theoretic Semantics*).

Preface

Despite its rather ambitious subtitle ('*The* Logical Foundations of Computation'), the report of the investigations which culminated with the results described here is to be seen as an 'attempt' at finding intuitively appealing foundations for the notion of *computation* solely from the laws of logic. Perhaps it would be more appropriately entitled 'An *Essay* in the Logical Foundations of Computation', given the *endeavouring* nature of the studies. As a matter of fact, an endeavour to provide a reasonably coherent reading of philosophical writings in order to explain more 'concrete' notions such as the notion of ***computation***, can only be taken as one of the many possible *attempts* to 'bring down to earth', as it were, what the particular philosopher tried to teach. I have attempted to offer a possible reading of Ludwig Wittgenstein's writings on the notion of *meaning*, resulting in a reformulation of Prof. Martin-Löf's formalisation of constructive mathematics, *Intuitionistic Type Theory*, into a formal system called *'Meaning-As-USE' Type Theory*, which may be relevant for a better understanding of interesting notions such as *computation* and *constructive reasoning* only by means of logical rules. Such a reformulation of Prof. P. Martin-Löf's formalisation of *constructive* mathematics, can be seen as an alternative to Prof. Martin-Löf's theory of meaning whose underlying *values*-based semantic theory is being replaced by a *use*-based semantic theory. Its

semantic theory offers an alternative account of the traditional proof-theoretic account of meaning which, despite rejecting the idea that the meaning of a proposition is given by its *conditions for assertion*, attempts to show that by insisting on the usefulness of providing semantics by giving proof-rules, one can bring Proof Theory back to the attention of logicians, philosophers as well as mathematicians. I have then deliberately made a clear suggestion for taking 'Proof Theory and Computer Programming' as an alternative paradigm to Prof. Martin-Löf's own 'Constructive Mathematics and Computer Programming' (Martin-Löf 1982.).

Unfortunately, the word *Essay* has not been used frequently in titles of reports on scientific investigations, and I have deliberately avoided taking a position which might leave room for unwanted misunderstandings. Due to the currently prevailing usage of the word *essay* it might give the impression of a work of essentially literary nature. It may be worth remarking, however, that there is a strongly 'scientific' sense in certain uses of the word *essay* in the English language, which is precisely that of signifying an *attempt*, an *experiment*, etc.. Nevertheless, since R. Dedekind's studies on the foundations of number theory, which the English translator W. W. Beman called *Essays on the Theory of Numbers* (Dedekind 1901), and which was probably the first real attempt to deal with foundational issues concerning definitions by induction (or recursion) in a formal and precise way, laying down the basis for a mathematical theory of computations, the word *essay* has been mostly reserved for literary expositions or short 'treatises'. Looking at the etymology of the word as described in the Oxford Dictionary, one can indeed see that there is a few different meanings (uses) of the word *essay*, including that of signifying the act of *trying*, *attempting*, *endeavouring*:

Essay (e·se^{i}), *sb.* 1597. – (O)Fr. *essai*, f. *essayer*; see next. Cf. ASSAY *sb.*] **1.** The action or process of trying or testing; an ASSAY 1598. **2.** An attempt, endeavour 1598. †**3.** A first attempt in learning or practice –1734; a first draft –1739. **4.** A short composition on any particular subject; orig. 'an irregular undigested piece' (J.), but now said of a finished treatise 1597.

1. By way of trial and e. HEYLIN. A small e. of my zeal for .. your Majesty CLARENDON. **2.**

My second e. at authorship 1865. **4.** For Senacaes Epistles .. are but Essaies — that is dispersed Meditations BACON.

(From 'THE SHORTER OXFORD ENGLISH DICTIONARY. On Historical Principles.', prepared by William Little, H. W. Fowler and Jessie Coulson, revised and edited by C. T. Onions, Third Edition, completely reset with etymologies revised by, G. W. S. Friedrichsen, and with revised addenda, Volume I, A—MARKWORTHY, Oxford at Clarendon Press, 1974, p. 682.)

There does not seem to be any doubt that Dedekind's *essays*, e.g., constitute indeed a serious (scientific) *attempt* at providing logical foundations for mathematical notions such as numbers and continuity. In the preface to the first edition of the *essay* entitled *The Nature and the Meaning of Numbers*, an English translation from the German *Was sind und was sollen die Zahlen?*, Dedekind attempts to justify his foundational investigations on the nature of something he considered *an immediate result from the laws of thought*:

In science nothing capable of proof ought to be accepted without proof. Though this demand seems so reasonable yet I cannot regard it as having been met even in the most recent methods of laying the foundations of the simplest science; viz., that part of logic which deals with the theory of numbers. (...)

(Dedekind 1901, p. 31.)

Throughout Dedekind's words of motivation for the work that follows, one can see the intention of *endeavouring* a more rigorous treatment of issues that had only been dealt with in a less precise, less 'scientific' way. And he goes on to justify his deep investigations on what seems to be a most fundamental issue for the understanding of the notion of *computing*, solely from the laws of logic: induction, recursion and the interplay between the two.

(...) As such main points I mention here the sharp distinction between finite and infinite (64), the notion of the number [*Anzahl*] of things (161), the proof that the form of argument known as complete induction (or the inference from n to $n+1$) is really conclusive (59), (60), (80), and that therefore the definition by induction (or recursion) is determinate and consistent (126). (...)

(Ibid., pp. 32–3.)

It would be unreasonable to say that such writings on scientific matters so precisely performed by R. Dedekind cannot be appropriately called *Essays* on the basis that an *essay* is a report of literary, non-scientific nature. It is just a matter of checking that the word *Essay* has other senses than the more popular 'literary composition', namely that which comes straight from the French verb *essayer*, to try, to attempt. In the case of Dedekind's works the word *essay* employed by the English translator, as far as I have understood it, has been used to mean *endeavour*, and not as a classification of the writings as a certain literary composition.

G. Frege himself acknowledges the scientific relevance of Dedekind's investigations (*essays*), and in the preface of his remarkable *Grundgesetze der Arithmetik*, one can say a rival to Dedekind's own studies, he writes:

(...) My purpose necessitates many departures from what is customary in mathematics. The requirements upon the rigour of proof inevitably entail greater length; anyone not bearing this in mind will indeed be surprised at how laboriously a proposition is often proved here that he believes he can grasp in one single act of understanding. This will strike us particularly if we compare Herr Dedekind's work ***Was sind und was sollen die Zahlen?***, the most thoroughgoing work on the foundations of arithmetic that has lately come to my notice. (...)

(Furth (ed.) 1964, p. 4.)

(The emphasis is mine.)

Due to the 'unpopularity' of the *endeavouring* sense of the word *Essay*, I have decided to leave the less humble definite article '*The*' in the title of my research report. Be it as it may, . . .

LONDON, NOVEMBER 1, 1989. R. J. G. B. DE Q.

die skeptische Resignation, die Einsicht in die Unerkennbarkeit der Wirklichkeitswelt, ist keine bloße Negation, ist unser bestes Wissen.

(the sceptical resignation, the insight into the unknownability of the World of Reality, is no mere negation; it is our best knowledge.)

F. M.

In unserer Sprache ist eine ganze Mythologie niedergelegt.

(A whole mythology is deposited in our language.)

L. W.

1 *Introduction*

1.1. Motivation

Looking at a programming problem one can distinguish essentially three 'objects': a) the *problem* to be solved, which is usually expressed as a logical proposition in some kind of specification language; b) the *solution*, generally presented in some programming language; and c) the *explanation* (or *proof*) of why the latter is a correct solution to the former, where the language of some proof system (axioms + rules of inference) (cf., e.g., Hoare 1969), program logics (cf., e.g., Goldblatt 1982), or abstract concepts such as embedding of algebras and interpretations between theories, are normally used to express this third object. It is not difficult to see that the fact that one has to use different linguistic systems in order to find a solution and check that the particular solution is a correct one, makes programming something harder to deal with rigourously in all its stages. However, it becomes increasingly more important to treat the activity of programming at an appropriate level of rigour. To be able to do this it seems as though one has to look for ways of treating the three objects mentioned above within the same level of rigour and formality, (possibly) using a unified semantically sound linguistic framework. It seems that the 'language' of the mathematics of computing needs to be further clarified, and

I want to argue that *type* is a clean as well as a sufficiently general notion to have as the starting point of a better clarification of the language of the 'mathematics of computing'.

Now, one can go even further and say that if one is to identify particularly general notions which have supported most of what has been done in clarifying the mathematics of computing as well as the computing of mathematical objects, one can hardly neglect either one of the key concepts in the modern theory of computing: *types* and *theories*. Two mainstreams of modern theoretical computer science bear witness to the convenience and usefulness of such concepts as *types* and *theories* in providing an intuitive formal account of the phenomenon of *computation*, even if they do not share a single uniform view on the matter. If on the one hand, the theories of abstract data type specifications have given us useful insights into the nature of theory presentations, on the other hand, from *Type Theory* have come the lessons about how types should be presented as well as where the focus of attention is to be more appropriately placed, namely the *judgements* one can make from a type presentation.

The result of putting the good lessons from both sides into the design of an adequate body of concepts and notation for the activity of specifying/verifying data types, is a logic-free theory of constructions named *'Meaning-As-USE' Type Theory* as an explicit reference to its semantical standpoint, which provides one with such uniform presentation recipe: *theories* are presented as *types* of their 'terms'.

When talking about *presentations* of types, theories, logics, etc., one can hardly avoid taking a clear position as to what one understands by a definition, and what kind of definition one needs to characterise presentation patterns. And, indeed, an issue of much relevance which naturally arises is the question of what happens from the linguistic point of view when *definitions* are laid down and *implementations* are sought in order to realise the requirements specified. Putting it another way, what would be the best way of looking at *definitions* via rules as something which gives 'meaning' to a particular (extra-)logical sign, and also how can one prove

that the object defined via rules can be proved to exist as a particular combination of already existing objects? How can proof theory, which is apparently the most widely accepted of all instruments for operational semantics for the language of mathematics, be helpful in this case?

Frege's early awareness of the importance of a notion such as that of a non-abbreviatory definition (Frege 1914) is of significant help by showing that the idea of *implementation* in mathematical logic is neither strictly 'new' nor irrelevant. Be it in the construction of a new system of logical signs, which is the true gist of the activity of software engineering, or in the clarification of the 'definition as presentations' meta-mathematical paradigm, the notion of *analytic* definition plays a very important foundational rôle.

1.1.1. Semantics and Programming

Looking at the difficulties of providing coherent semantics to programming languages, one can readily see the benefits of semantical studies leading to a more coherent account of meaning. The ideas developed by Ludwig Wittgenstein on the semantical foundations of language, formalised as well as ordinary, constitute an admirable source of inspiration.

It was a most interesting experience to come across, just by chance, an explanation of the concept of meaning given to students of computing science, from a series of lecture notes on 'Semantics of Programming Languages', which, from my own point of view, sounded remarkably 'unmatched' with the context in which it was involved. It is usually the case that one learns to operate a machine by learning certain rules of use, and with programming languages this pattern does not seem to change radically. The idea that in order to have a good (precise) understanding of a command or a program written in a programming language one has to refer to an 'object' through a correspondence mapping to a particular 'mathematical' object, is rather different from the natural process of learning how to use a programming language. The attempt at a 'formal' explanation of the concept of meaning involved a sentence like:

(...) What are "meanings" ? In formal semantics *meanings* are mathematical *objects*. (...)

(The *emphasis* is mine.)

Meanings are *objects*. Are they really? Perhaps 'mathematical *tools*' instead of 'mathematical *objects*' would be more appropriate, for from the latter another question would still have to be answered: what meanings would then be attached to the presumably 'primary' objects? Would they have a meaning in themselves? Such a view appears to be a typical heritage of a certain philosophical tradition since St. Augustine's 'as I heard words repeatedly used in their proper places in various sentences, I gradually learnt to understand what objects they signified' (*Confessions*, Book I, §8). It is the same tradition which says that for every word there is an object to which it corresponds. An offspring of such a long philosophical tradition is to be found in theories of meaning for formalised languages where the words/sentences of a language are given meaning through an interpretation into a 'universal domain of (canonical) values'. By adopting such a view of language and meaning one would have to accept that interpretation by themselves could settle meaning. But when would such an 'infinite' chain of interpretations find a safe endpoint?

For the case of formalised languages *'Meaning-As-USE' Type Theory* would seem to be of great help.

The proof-theoretic notion of *reduction*, so far used as a mere meta-level device to prove *normalisation* theorems, seems to be a particularly suitable notion one needs to relate proof theory, which can be taken as an operational account of semantics for the language of mathematics, with computer programming where the *operational* intuition seems to prevail, regardless of the language used. According to a particular alternative characterisation of operational semantics which I call 'semantics of use' and which emerged from an 'enrichment' of traditional proof-theoretic semantics, the 'normalisation' (*reduction*) rules, not just the *introduction* rules as Gentzen's tradition claims, constitute one of the fundamental aspects of the semantics because they characterise the use (meaning) of a particular logical sign.

Taking a natural deduction style of presentation of logics, combined with the notion that *propositions-are-types*, where the elements of a type are the 'codings' of proofs of the corresponding proposition and a proposition is true if the type is non-empty, the *normalisation* rules show the effect of *eliminatory* operations on the result of an *introduction* inference. Both sides, proof theory and computer programming, can benefit from this connection. A proof-theoretic account of meaning which stands on the notion of *reduction* as a fundamental semantical notion, can be given philosophical justification ('meaning is use'), and perhaps become more convincing than the usual Gentzen semantics. Moreover, considering the utility for computer programming of the identification *propositions-as-types*, which is the basis for the semantics of *Intuitionistic Type Theory* where elements of a type are taken to be 'codings' of proofs of the corresponding proposition, computer programming can gain enormous benefit by incorporating a number of very useful notions and techniques from proof theory, such as a more 'syntactically' oriented (proof-theoretic) account of semantics, and a certain distinction of rules of construction (*introduction*, *elimination*, *reduction*), together with a general notion of a (typed) second-order (fixpoint) operator as a particular case of a more general technique of proof by induction: proof by transfinite induction. I describe the application of a reformulated version of Prof. Martin-Löf's *Intuitionistic Type Theory* (*ITT*) to theories of abstract data types specification, where the *reduction* rules (or *equality* rules, as they are called in *ITT*) are looked upon as primitive semantical rules, rather than derived (or secondary) rules. Programming notions such as 'constructors', 'destructors' (or 'selectors', as these are sometimes called) and 'equation' (*reduction* rule as I see these constituents of the semantics of the type being specified) will then fit well within the framework of this reformulated type theory, which I suggestively call *'Meaning-As-USE' Type Theory* (*MAUSE*). *MAUSE* can be looked at as a programming language as well as a formalisation of constructive mathematics which would seem to be in accordance with Wittgenstein's doctrine 'meaning is use' for the mathematical language because its semantics is based on operations covering the two fundamental aspects of use (meaning), and the notion of meaning of a proposition is taken to be insep-

arable from the consequences one can draw from it (shown by the *reduction* rules) (de Queiroz 1988e).

A number of issues commonly discussed in theories of abstract data type specifications, such as error-handling (error-recovery, error-propagation, etc.), implementation, completeness of axioms, the rôle of *hidden functions* and *if-then-else*, etc., will be dealt with in a much simpler way than most of the approaches available in the literature. This is made possible for at least two fundamental reasons: first of all, the notion of semantics is taken as based essentially on *operations* rather than on *denotations*, which means that one does not have to introduce abstract (auxiliary) concepts such as algebras and mappings to explain the semantics of the data types being specified. The semantics is explained via 'normalisation' *operations* (the rules of 'use'), which represent my own reading of (the later) Wittgenstein's notion of 'language-games' for the case of mathematical language. Secondly, due to the fact that one stays within a single linguistic framework which may include the most primitive logical operations such as *pairing* and *splitting*, *lambda abstraction* and *application*, *sum* and *choice*, and *fixpoint* and *potential infinity*, one does not have to use extra notions other than 'definitional extensions' to explain *implementations* and *proofs of correctness*, which seems to be a natural way of thinking about *implementations*.

1.1.2. Frege's notions of definition: constructive (abbreviatory) and analytic

In attempting to say something about specification of abstract data types one cannot avoid getting involved in a discussion which concerns one of the most problematic issues in mathematics: *definitions*. Without going too far into a philosophical discussion which would certainly go beyond the aims of this report, I would like to take Frege's notions of definition as the starting point for my discussion. Therefore, an assumption is made here that whenever I say something about *definitions* I will be referring to one of Frege's two distinct notions, where the distinction should be made

clear either from the context or explicitly. This assumption shall be of fundamental importance for the understanding of my development of most of the issues covered here, including 'meaning vs. rules', implementation, error handling and a number of other related topics which concern directly theories of specification of data types.

1.2. Setting the Aims

Frege's Realism seems to have prescribed a particular way of thinking about language and meaning, traces of which are found rather sharply even in the most 'radical' attempts at an 'anti-realist' theory of meaning for the language of mathematics. The model of 'object and designation', 'expression and (numerical) value', seems to have remained an essential ingredient of most of the theories of meaning, be it a *truth* or a *proof*-based theory. The profound and most remarkable teachings on an utterly distinct way of thinking about meaning offered by the great Austrian thinker Ludwig Wittgenstein appear to provide an imense new world of ideas regarding the nature and the structure of the mathematical activity which does not seem to have been properly explored so far. By attempting to 'cure us from the temptations to look about us for some object which we might call "the meaning" ' (*Blue Book*, p. 1), Wittgenstein appeared to have demonstrated that meaning can be much better understood as something ultimately distinct from an 'object', a '(numerical) value', etc.. And this appears to constitute an enormously more convincing alternative to his own *truth*-based early account of meaning given in the *Tractatus*, as well as to most of the alternatives to that early (strict) 'atomistic' theory, which despite avoiding the notion of *truth* seem to have remained deeply rooted in the conception of meaning based on the model of 'object and designation', 'expression and (numerical) value', etc.. Prof. Martin-Löf's formalisation of constructive mathematics, *Intuitionistic Type Theory*, a relatively successful attempt to provide an 'anti-realist' theory of meaning for the mathematical language, seems to show clearly its roots on Frege's 'object and designation' basic principle of Realism, by talking about meaning of propositions, types, expressions, as being determined by a certain 'canonical value', in spite of avoiding the notion of *truth*.

Here I attempt to demonstrate that Prof. Martin-Löf's endeavour to provide an intuitionistic theory of meaning for the mathematical language, although essentially rooted in Frege's (realist) conception of meaning as given by '(numerical) values', can still be reformulated to a point much closer to Wittgenstein's *use*-based 'definition' of meaning, where there is no strong attachment to the usual 'object and designation', 'expression and (numerical) value', etc., models of understanding the concept of 'meaning'.

1.3. General Overview

Chapter 2 discusses Frege's notions of definition, attempting to motivate the claim that computing science definitions are mostly what Frege called *analytic* definition. Proof-theoretic, Recursion-theoretic, etc., definitions are merely particular cases of such a general notion of non-abbreviatory definition.

Chapter 3 is a critique of Gentzen-like proof-theoretic semantics, where it is usually advocated that the *conditions for assertion* determine the meaning of a proposition.

Chapter 4 looks at Wittgenstein's writings on 'meaning', building up evidence for the view can unify most of Wittgenstein's writings on the subject in terms of a fundamental point characterised by the inseparability between the meaning of a proposition and the *consequences* one can draw from it. In order to illustrate the argument I attempt to show the usefulness and convenience of my own reading of his late semantical paradigm *meaning-is-use* as bearing close links with the paradigm 'the meaning of an assertion is inseparable from what can be done with it'. This serves the basis for the discussion that is presented in the next chapter.

Chapter 5 shows how one can formalise the notion of *use/utility* within a type-theoretic framework. I also demonstrate the usefulness of having the formal counterpart of Wittgenstein's *meaning-is-use* in the *normalisation* rules, to give a reasonable account of types, type presentations, theory presentations, logic presentations, etc., in the sense which prevails within the scope of modern computing science.

Chapter 6 describes the main differences between Prof. Martin-Löf's *Intuitionistic Type Theory* and my own *'Meaning-As-USE' Type Theory* which was developed from a different semantical perspective based on Wittgenstein's intuition on the connections between *meaning* and *use/utility*. It is demonstrated how useful such new semantical perspective, and therefore its formal counterpart, can be in dealing with typical computer programming notions such as infinity and lazy evaluation, in a rigourous manner. A new higher-order type forming operator 'Fix' is defined and an application showing how to 'implement' inductive types such as Natural Numbers, Lists and Stacks, is demonstrated.

In Chapter 7 it is argued that an *operations*-based approach to the semantics of abstract data type specification should significantly simplify typical problems associated with implementation issues, error handling, parameterisation, structuring, etc.. It is further argued that this is exactly because the notion of semantics would be based on *operations* rather than the usual detour via *denotations*, and a number of these problematic issues could then be dealt with 'syntactically' (i.e., proof-theoretically). I then describe how the reformulation of Prof. P. Martin-Löf's *Intuitionistic Type Theory* resulting in a formal system named *'Meaning-As-USE' Type Theory*, can be used to fulfil the above objectives by presenting a more adequate '*operations*-based' semantics. Describing the features of such *'Meaning-As-USE' Type Theory* I present a much simpler approach to theories of specification and implementation of abstract data types by avoiding the usual problems of explaining concepts such as 'implementation', 'parameterisation' and 'structuring' with a detour through denotational semantics via 'embedding of models', 'mappings of algebras', or even a more syntactical and much simpler but yet unsatisfactory semantical explanation via 'interpretation between theories'. Furthermore, by identifying type presentations with theory presentations, and comparing the presentation of a λ-theory given by Prof. H. Barendregt in Barendregt 1981 with the presentation of a Π-type of λ-terms in the framework of *'Meaning-As-USE' Type Theory* one gets rather simple and straightforward semantical explanations of rules such as the η- and ξ- rules of the λ-Calculus in an 'intentional' manner, as opposed to the more

customary 'extensional' way: η as extensionality; ξ as weak extensionality.

Chapter 8 attempts at a evaluation of the main achievements, and suggests directions for further research.

In the Appendix I present the formal system *'Meaning-As-USE' Type Theory*. More than just a list of inference rules, my *'Meaning-As-USE' Type Theory* should be seen as a general guideline prescribing how types, theories and logics can be presented in such a way that no appeal to *interpretations* or semantic *values* is necessary in order for one to grasp the meaning of each type, theory, logical connective, etc..

1.4. General Remark

The present dissertation was written in the course of my investigations into the logical foundations of computability. It is partly written in the form of an essay in the nature of definitions from a 'purely' logical point of view. As this is such a delicate and historically problematic issue, I make an attempt to express my own view on the subject, trying as much as possible to relate it to other new or traditional views, but restricting myself to 'definitions' used in the particular world of 'computing' science. For all this, I shall be making use of a reasonably large number of quotations which, although being justified whenever suitable, I hope will speak for themselves in most cases. At the back of my mind there is always the question of how useful such an investigation on the nature of mathematical definitions can be for the understanding of fundamental issues in the 'science of the computable' from a more 'linguistic' point of view. In order to relate the view I am advocating here in this report with the most relevant views on similar issues available in the literature I shall be deliberately repetitive, and for this I would beg patience and forgiveness to the more acquainted reader.

2 *On Frege's Notions of Definition*

2.1. Preamble

Taking the view that *computability* is the same as *definability in a language*, one cannot get away from the task of clarifying the notion of *definition* one is assuming. Church's classical identification of the class of *effectively calculable* functions with the class of *λ-definable* functions is an example of a view which is in line with the paradigm '*computability* is *definability in a language*':

(...) We now define the notion, already discussed, of an *effectively calculable* function of positive integers by identifying it with the notion of a recursive function of positive integers (or of a λ-definable function of positive integers).

(Church 1936, p. 356.)

(...) The corresponding question of the relationship between effective calculability and λ-definability had previously been proposed by the author independently.

(Id. Ibid., footnote 18.)

(The emphasis is mine.)

I believe it is fair to say that Ludwig Wittgenstein's (later) philosophical views on semantic matters (meaning vs. use) constitute an enormous source of inspiration

for a sound foundation of the theory of computation, where rules and meaning have a close interplay. From this point of view there may emerge a notion of *computability* in terms of a logical characterisation of *use* (meaning) via rules. Church's 'λ', for example, is a starting point which fits into this context just because its meaning is given by rules of use: *abstraction* and *β-reduction*.

In this chapter I attempt to discuss issues related to definitional mechanisms and the notion of computability, observing the common ground shared by the two models of computability mentioned by Church: in both the λ-calculus and recursive function theory the notion of *reduction* (*normalisation*) plays an important rôle in the definition of meaning. The reason for this seems to be that the meta-mathematical device of *normalisation* appears to be a suitable definitional mechanism for the cases where defining is not abbreviating but presenting. It is usually the case in computing science that one wants to lay down definitions which are not abbreviations, but are rather *presentations* (presentations of logics, types, theories, etc.).

For this analysis I look back at Frege's notions of *definition*. And I say that 'analytic' (non-abbreviatory) definitions require *normalisation* rules as part of the definition of meaning just because such a notion of definition of signs with a 'preconceived' meaning seems to correspond to the *definitions-as-presentations* customary to computing science. According to my interpretation of Wittgenstein's semantical views, the rules of *normalisation* capture his notion of 'the use of the sign in the language-games'. Considering the reference to some kind of *logical analysis* in the case of a non-abbreviatory definition, which is the one I am interested in due to its 'coding' nature, I say that there is something in common between proof theory and recursion theory, which is related to the fact that the meaning of (extra-)logical signs is given by rules of use and a subsequent search for an *implementation* follows the definition-as-a-presentation. My primary motivation stems from the fact that in both cases 'logical' (as well as 'extra-logical') signs are given meaning via rules of '*reduction*' ('*normalisation*') representing *use*, which would presumably allow for the unification of definitional mechanisms into a single '*reduction*'-based semantical framework. I want to investigate how adequate my own reformulation of Prof.

P. Martin-Löf's *Intuitionistic Type Theory* (Martin-Löf 1984) is as a starting point for such a framework. Proof-theoretic as well as recursion-theoretic definitions appear to be of the same fundamental nature: 'new' signs are defined (presented) via rules of use and via abbreviations. With respect to proof-theoretic definitions (presentations), which have not so far been looked at as dependent on the use of the meta-mathematical device of *normalisation*, my standpoint rests on a revision of the semantic rôle of proof rules (*introduction*, *elimination*, *reduction*) which I have performed in order to identify which rules can fully represent (operational) meaning. This revision resulted in my 'semantics of use' whose central ideas were first discussed in de Queiroz 1987c. As a consequence of my point of view regarding proof-theoretic as well as recursion-theoretic definitions as particular cases of (my understanding of) Frege's *analytic* definition (and my 'definition by use'), I look at how the proof of the 'existence theorem' for recursive definitions and for certain 'proof-theoretic'-like definitions which I say takes the form of my 'definition by use' or 'definition-as-presentation' (see Aczel 1980, and his definitions of *systems* followed by an existence lemma) relates to the *logical analysis* suggested by Frege. I aim at demonstrating that a revision of the semantics of *Intuitionistic Type Theory* can lead to a uniform type-theoretic framework to define logical as well as extra-logical 'signs', based on a single semantical principle: meaning is determined by use, and use is logically characterised by rules of *introduction* and *reduction*, the latter being formal counterparts to (later) Wittgenstein's 'the use of the sign in the language-games' because they demonstrate the effect of *elimination* inferences on the result of *introduction* steps.

2.2. Introduction

By *definability* I understand the property of an object being given by rules of use, in the form of natural deduction-like *introduction* and *reduction* rules (or *elimination*, as in the usual case). I take the view that to give *meaning* to a logical sign is to provide *rules* for the *use* of it. That is how I understand the paradigm *meaning-is-use*. I take it as my starting point, finding inspiration in the following remark by

Wittgenstein:

> For a *large* class of cases — though not for all — in which we employ the word "meaning" it can be defined thus: the meaning of a word is its use in the language.
>
> (*Investigations*, Part I, §43, p. 20e.)

This particular semantical view, which I call 'semantics of use', results essentially from a revision of proof-theoretic semantics into a richer and more coherent account of meaning than that of Gentzen where *introduction* rules alone are said to constitute definitions.

Assuming that 'use' can be characterised logically via rules of *normalisation*, which corresponds to one aspect of my own reading of Wittgenstein's later account of propositions discussed in de Queiroz 1988e ('language-games'), I attempt to provide a 'linguistic' account of meaning for the language of mathematics based on *operations* rather than on *denotations*, where no interpretation would be involved. I find inspiration in a rather 'solipsistic' remark made by Wittgenstein regarding the links between (one's) language and (one's) world:

> *The limits of my language* mean the limits of my world.
>
> (*Tractatus*, §5.6, p. 56.)

to say that the (potential) language of constructive mathematics, which would make that informal body of concepts more precise via rules of inference and appropriate definitions, could be included in that *large* class referred to in the previous remark, if we accept that a certain body of mathematics is said to be 'constructive' when its objects can be defined by (finitarily graspable) rules of *use*. Expressing his thoughts via such a 'language' of constructive mathematics, where definitions would be made by rules of *use*, the mathematician would be adopting a 'constructive' attitude towards the mathematical activity. Therefore, I say that the *language* of constructive mathematics should be taken as a member of this *large* class of cases referred to by Wittgenstein, provided that the connection rules-meaning is sufficiently clarified. By formalising the basic principles of such an 'anti-realist'

account of mathematics, the term 'realist' being used in the sense of Frege's account of meaning for the language of mathematics (a more interested reader may find a rather extensive discussion on 'Realism' in Dummett 1982), one should be able to provide an account of operational meaning based on Wittgenstein's use-based semantical paradigm for the language of mathematics. This is how I see the rôle of a formal system like Prof. P. Martin-Löf's *Intuitionistic Type Theory.*

A revision in the account of operational meaning offered by Gentzen's proof-theoretic semantics can lead to more general and more coherent semantical principles which would unify patterns of definitions of logical signs, either proof-theoretic or recursion-theoretic-like definitions. A certain feature of *Intuitionistic Type Theory* (*ITT*), which is to have the rule of *normalisation* as part of the system of rules for the definition of logical signs, rather than a mere meta-level device, makes it a suitable starting point for such a 'meta-logical' unified framework. This aspect of such a formal system, however, has not been highlighted given that its semantics is provided via Gentzen's principles where *assertability conditions* (*introduction* rules) give meaning to logical constants. A particular reformulation of *ITT* according to the 'semantics of use' seems to recover this important feature of such a formal system. The possibility for construction of such a uniform framework stands on the fact that the logical principle of *inversion*, which supports the *normalisation* rule, is a fundamental principle both in proof theory and recursion theory because it allows for the logical representation of one of the fundamental aspects of the meaning (use) of logical signs defined by rules: the aspect of the *consequences* of the 'proposition' asserted, which in my interpretation corresponds to Wittgenstein's 'the use of sign in the language-games'.

In both cases the notion of definition is a fundamental issue. By addressing the nature of definitions from a particular point of view (Frege's *analytic* definitions as my 'definitions by use', which complies with Wittgenstein's 'meaning as use' semantical principle because it covers the two fundamental aspects of the use of the particular logical sign: the conditions for asserting the sentence (grammatical

aspect – *introduction* rules) plus the conclusions that can be drawn from the sentence inferred by an *introduction* (consequences aspect – *reduction* rules)), an attempt is made to identify the kinds of rules that can capture the fundamental (syntactical as well as semantical) elements of definitions of logical symbols 'with a long-established use'. According to my understanding of proof-theoretic as well as recursion-theoretic definitions, the notion of *reduction* seems to be a fundamental notion as far as the definition of meaning by rules is concerned. I hope this can shed some light on the understanding of the nature of 'good' definitions, which is an issue of much relevance especially in recursion theory.

2.3. Frege's View

In an essay on the fundamental rôle of logic in the mathematical activity Frege attempts to analyse somewhat more closely two essential concepts of primary use to mathematicians: *inferences* and *definitions*. Considering the latter as the most characteristic as well as most problematic concept of mathematics,

> (...) Part of the mathematician's activity besides drawing inferences, is to give definitions. Most disciplines are not concerned with the latter at all; (...)
>
> (Frege 1914, p. 203.)

he devotes most of the essay to a discussion centred around the concept of 'definition'. The starting point is to say that in mathematics these two concepts are subject to the laws of logic:

> *Inferring and defining are subject to logical laws.*
>
> (Id. Ibid.)

(The *emphasis* is the Editors'.)

In the course of the discussion on the rôle of definitions in mathematics, he comes to a rather delicate point where he has to explain how a definition of a sign which had long established use can be given when 'constructing a new system of signs'.

The point stressed is that a definition is supposed to attribute meaning to a sign which had no meaning before. In the case where one wants to build a new system of logical signs and there is the need to have a sign to represent the logical symbol with long established use, the act of defining cannot be taken as just an abbreviation, given that the symbol had a meaning before, and the new sign has to reflect this 'pre-conceived' meaning in some way. In an attempt to clarify his own account of the notion of *definition*, Frege distinguishes two cases:

> We have therefore to distinguish *two quite different cases*:
>
> (1) We construct a sense out of its constituents and introduce an entirely new sign to express this sense. This may be called a 'constructive definition', but we prefer to call it a 'definition', *tout court*.
>
> (2) We have a simple sign with a long established use. We believe that we can give a logical analysis of its sense, obtaining a complex expression which in our opinion has the same sense. We can only allow something as a constituent of a complex expression if it has a sense we recognize. The sense of the complex expression must be yielded by the way in which it is put together. That it agrees with the sense of the long established simple sign is not a matter for arbitrary stipulation, but can only be recognized by an immediate insight. No doubt we speak of a definition in this case too. It might be called an 'analytic definition' to distinguish it from the first case. But it is better to eschew the word 'definition' altogether in this case, because what we should here like to call a definition is really to be regarded as an axiom. In this second case there remains no room for an arbitrary stipulation, because the simple sign already has a sense. Only a sign which as yet has no sense can have a sense arbitrarily assigned to it. So we shall stick to our original way of speaking and call only a constructive definition a definition. According to that a definition is an arbitrary stipulation which confers a sense on a simple sign which previously had none. This sense has, of course, to be expressed by a complex sign whose sense results from the way it is put together.
>
> (Ibid., p. 210.)

I say that the *analytic* definition is to have the form of proof-theoretic and recursive-theoretic 'definitions by use' or 'definition-as-presentation', where two aspects of the meaning are captured by *introduction* and *reduction* rules (rules of *grammar* and rules of *use*, respectively), according to my 'semantics of use'. One can see the need for a different notion of *equality* (rather than just the 'abbreviatory' one, which is used in the first case) in order to put in effective terms this

logical analysis suggested in the second case. In what follows I intend to show that a coherent way of doing this seems to be to take the *reduction* equality used in 'modern' mathematical languages such as the *β-reduction* in the *λ-calculus*, definition of *systems* in Aczel's definition of *Frege structures*, definition of 'standard' operators via the application of corresponding 'projections' in the language of *Category Theory*, and in the 'internalised' proof-theoretic *reduction* rules (like Prof. Martin-Löf's *equality* rules of his *Intuitionistic Type Theory*), and the equality used in *recursive* definitions which is normally seen as a 'congruence' equality. Concerning the latter, I say that it should be seen as a 'reduces to' equality, as a suitable notion of *equality* I need for 'definitions by use' or 'definition-as-presentation' (Frege's *analytic* definition) and subsequent *logical analysis*. The analysis itself will be similar to 'finding a solution for the existence theorem for recursive definitions', or simply proving the existence lemma for the symbols 'defined by use', as in Aczel's development of *Frege structures*.

2.4. Definitions in Recursion Theory

In this section I intend to highlight the 'operational', *use*-based, nature of recursive definitions, and to point out that the *logical analysis* suggested by Frege in his second notion of definition (case (2) (considering a *reduction* as a formal counterpart to an informal explanation of *use*), the *analytic* definition) can be understood as a method of solving the existence theorem for definitions by use.

Just to illustrate the point of this rôle of *reduction* rules as part of definitions, and the *logical analysis* as the proof of existence, I would like to refer to Prof. P. Aczel's development of *Frege structures*, where *systems* (*lambda system*, *pairing system*, etc.) are defined (presented) as, e.g.:

Definition 3.1. A *lambda system* for an explicitly closed family $\mathcal{F}$ consists of two $\mathcal{F}$-functionals $\lambda : \mathcal{F}_1 \rightarrow \mathcal{F}_0$ and $\mathrm{APP} : \mathcal{F}_0 \times \mathcal{F}_0 \rightarrow \mathcal{F}_0$ such that

$$\mathrm{APP}(\lambda x f(x), a) = f(a)$$

for all f in $\mathcal{F}_1$ and a in $\mathcal{F}_0$.

(Aczel 1980, p. 38.)

Definition 5.6. A *pairing system* consists of $\mathcal{F}$-functions $\text{PAIR} : \mathcal{F}_0 \times \mathcal{F}_0 \to \mathcal{F}_0$ and $p, q : \mathcal{F}_0 \to \mathcal{F}_0$ such that for all objects a, b

$$p(\text{PAIR}(a,b)) = a \qquad q(\text{PAIR}(a,b)) = b.$$

Lemma 5.7. *There is a pairing system.*

(Ibid., p. 48.)

I present informally an illustrative example of such a method of solving the Existence Theorem for recursive definitions, suggesting that it is indeed a *logical analysis* considering the fact that it turns out to be a particular case of higher-order induction proof procedures.

2.4.1. The existence theorem and the logical analysis

When defining a logical sign which will operate on a set with an inductive structure such as the natural numbers, via rules of *use*, one normally uses the following pattern, known as 'primitive recursive definition':

$$(I)\begin{cases} f(0) &= \alpha \\ f(n') &= \beta(n, f(n)) \end{cases}$$

where 'α' and 'β' are objects (concepts) already known. There are two points I want to make in this section regarding the aim of this section, which is to relate proof-theoretic definitions with recursion-theoretic definitions via a single semantical notion of 'meaning given by rules of use'.

First of all, the above pattern seems to miss the fundamentally 'imperative' nature inherent in such definitions via rules. The equality sign is hardly ever seen as a 'reduces to' sign. For this reason, the essential nature of such a pattern is hardly

seen as related to the nature of proof-theoretic definitions. As a matter of fact, the notion of meaning as understood by the prevailing principles of proof-theoretic semantics makes things even harder. Traditional proof-theoretic semantics usually takes *grammatical* rules, such as the rules for the *assertability conditions*, and the rules of *consequence* are said to having nothing to do with the meaning of the logical connective corresponding to the *grammatical* rules. However, in my reformulated proof-theoretic semantics, which I call 'semantics of use', where the notion of *reduction* is part of the meaning of the particular logical sign being defined because they demonstrate formally the *consequences* aspect in addition to the *grammatical* aspect given by the *introduction* rules, it seems to be easier to see the close links between proof-theoretic and recursion-theoretic definitions. A more appropriate way of giving such definitions, in line with Wittgenstein's *meaning-is-use* and my 'semantics of use' (de Queiroz 1987c), where the first aspect of the use of a particular logical sign is represented by the conditions under which it can be correctly used, this usually being left implicit for such 'recursive' definitions (in this case, it says that having two natural numbers one can form the 'term' '$a + b$'), and the second aspect is given by making explicit the *utility* being given to the sign '+' (in other words, the *usefulness*, the *purpose* of the particular symbol, or, even, the *consequences* that can be drawn from the term containing the particular symbol being defined), would be something like:

$$(I)\begin{cases} \texttt{EVALUATE}(f(0)) &= \alpha \\ \texttt{EVALUATE}(f(n')) &= \beta(n, \texttt{EVALUATE}(f(n))) \end{cases}$$

where '`EVALUATE`' plays a rôle similar to that which '`APPLY`' does for the case of 'λ'. It helps to code up the second aspect of the use (therefore meaning) of the (extra-)logical sign. For example, the definition of the sign '+' given by

$$(I)\begin{cases} \texttt{EVALUATE}(a + 0) &= a \\ \texttt{EVALUATE}(a + \texttt{succ}(b)) &= \texttt{succ}(\texttt{EVALUATE}(a + \texttt{PRED}(\texttt{succ}(b)))) \end{cases}$$

is in fact saying what the commitments one makes by forming '$a + b$' are, or, what can be obtained from it. It makes clear what are the consequences of 'asserting' '$a + b$', which constitutes one aspect of the meaning of any 'statement', according to Wittgenstein's use-based account of meaning:

Denn nicht das ist die Frage: was ich mache, wenn ... (dies könnte nur eine psychologische Frage sein) — sondern, welche Bedeutung die Äußerung hat, was sich aus ihr entnehmen läßt, welche Folgen sie hat.

which has been translated as

For the question is not 'What am I doing when ... ?' (for this could only be a psychological question) — but rather, 'What meaning does the statement have, what can be deduced from it, *what consequences does it have*?'

(*RPP II*, §38, p. 8^e.)

(The *emphasis* is the Translator's.)

The above (fundamental) part of the definition of '+' is showing exactly this aspect of its meaning (use) by saying what are the consequences that can be drawn from '$a + b$'. The definition of the symbol '+' is given by the rules of use, which in this case happen to be 'recursive' rules.

The second point I want to make concerning the nature of 'recursive' definitions as definitions via 'rules of use', is the fact that after writing down the definition one may want to 'logically analyse' the symbol being defined in order to find out whether there is already a logical sign with the same 'semantics of use'. In order to do that what one has to do is to find a 'solution' for the given recursive equations, which I understand as being the same as performing a *logical analysis* as suggested by Frege in order to find an 'implementation' of the function defined. That is, we prove the 'existence lemma' similarly to Aczel's procedure after the definitions of '*systems*' by rules of *introduction* (*grammar*) and *reduction* (*use*).

Several attempts have been made to find a general way of solving recursive equations, most of them intimately linked to earlier problems like justifying the possibility

of definition by recursion, and finding a justification for the axiom of mathematical induction. As early as the beginning of the century, people like Dedekind had tried to develop a method to 'solve' recursive equations by using an *ordering* relation ('$\leq$', in particular) between natural numbers as a primitive concept. Later on, around the end of the thirties, not happy with the technical difficulties of Dedekind's method, people like Prof. Lorenzen (Lorenzen 1935) and Kalmár (Kalmár 1940) independently made remarkable steps towards finding a more general and less technically difficult technique to prove the existence theorem, providing the required '*explicit* definition from an *inductive* one' (according to the terminology in use at the time).

The method developed by the Hungarian mathematician L. Kalmár is probably the first one to apply explicitly the idea of *partial* solutions to the problem of turning an *inductive* definition into an *explicit* one.

> In this note, I shall give one more proof, avoiding the order relation like Lorenzen, but also the idea of a multivalent function which is plainly more complicated than that of a function defined only for *some* natural numbers, occurring also in the heuristic argument referred to. (...)
>
> (Kalmár 1940, p. 230.)

> (...) Instead of saying, a function is not defined for a given argument, we could say, its value is ∞ or any other symbol different from the natural numbers. If we confine ourselves to functions whose values are different from 0, we can use 0 as such a symbol. (...)
>
> (Id. Ibid., footnote 13.)

(cf. the idea of using '∞' and Prof. Scott's '$\perp$' as the *undefined* in Scott 1976, 1981.)

The idea was that a certain function g would be called a *partial* solution for a schema like (I) if

(a) so far as $g(0)$ is defined, $g(0) = \alpha$;

(b) so far as $g(n')$ is defined, $g(n)$ is defined too and $g(n') = \beta(n, g(n))$.

Given that, one would then assert

(1) For every n, there is a *partial* solution g of (I) for which $g(n)$ is defined.

(2) If g_1 and g_2 are *partial* solutions of (I), and if $g_1(n)$ and $g_2(n)$ are both defined, then $g_1(n) = g_2(n)$.

For example, the function g everywhere undefined is a *partial* solution. Likewise, the following g

$$\begin{cases} g(0) = \alpha \\ g(n) \quad \text{undefined for} \quad n \neq 0 \end{cases}$$

is also a *partial* solution with $g(0)$ defined.

Following this line of reasoning Kalmár goes on to the next step, which is to analyse the case where the function is defined for some n different from 0. In other words, the process goes by successive *iterations*.

The method Kalmár had in mind does not seem to be essentially different from the well known *fixpoint* method of solving recursive equations developed by Prof. Scott in his modern theory of computable functions (see, e.g., Scott 1976, 1981.). One starts from 'no information at all' about a particular infinite object (the recursively defined function), and goes on building it up by steps, using finite *partial* approximations.

One could probably see it better through an example.

Example. Let f be inductively defined as:

$$\begin{cases} f(0) = 0 \\ f(n) = f(n-1) + 2n - 1 \end{cases}$$

To find a 'solution' for the above recursive equations, one follows the heuristic argument that at the very first stage nothing can be said about the function apart from the assumption that it is *undefined* everywhere. By *iteration* one goes on obtaining more and more defined solutions *approximating* the expected final solution,

that will turn out to be the *limit* of all those *partial* approximations. The iteration goes as something like:

$$
\begin{aligned}
f_0(n) \quad & \textit{undefined} \text{ for all '}n\text{' is a } \textit{partial} \text{ solution} \\
f_1(n) = (n = 0 \rightarrow 0, f_0(n-1) + 2n - 1) \quad & \text{is a } \textit{partial} \text{ solution} \\
f_2(n) = (n = 0 \rightarrow 0, f_1(n-1) + 2n - 1) \quad & \text{is a } \textit{partial} \text{ solution} \\
& \ldots
\end{aligned}
$$

Drawing a *table* for the successive stages of information about 'f' (where '$\perp$' represents *undefined*) we have:

	0	1	2	3	4	...
f_0	$\perp$					
f_1	0	$\perp$				
f_2	0	1	$\perp$			
f_3	0	1	4	$\perp$		
f_4	0	1	4	9	$\perp$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Looking at the table one *guesses* the behaviour of 'f': in the *limit* the function 'f' will behave like '$f(n) = n^2$ ' for all natural numbers 'n'.

It should not be too difficult to see that the *guessing* is a *logical* inference, and, in particular, an *inductive* inference. And, indeed, one can see that this logical inference was very much like an induction over *higher-order* objects (*sequences* of numbers) and it is better known in mathematical logic as *transfinite* induction. Starting from the *empty* sequence, represented by the first stage where 'f' is undefined everywhere, one goes on by building finite sequences iteratively. Therefore, one has, for the *transfinite induction* inference, the *empty* sequence as the base case, and the assumption that there is a *partial* sequence which is defined up to a certain 'n', as the induction step. This heuristic argument is formally defined by Prof. Scott, especially in the context of solving recursive equations, as *fixpoint* induction.

It should be easy to see how *fixpoint* induction (as defined by Prof. Scott) strongly relates to *transfinite* induction (as defined by Sierpiński, in Sierpiński 1958).

After having logically analysed the *recursive* definition of the function 'f' we ended up with 'f' spelled out only in terms of already existing objects: if we recognised the '*square*' function it is presumably because we had it before. So, in effect, what we have done was to choose a good candidate for our proof of existence, and just checking 'mentally' that the candidate (the *square* function) does fit in the original (recursive) definition, we felt 'sure' that the candidate was indeed a good one. It may not always be the case that we can recognise the behaviour of the function to the extent of finding a 'suitable' well-known candidate. The 'factorial' function, for example, can hardly be thought of having a candidate for replacement in a way similar to the more complicated 'f' having *square* as its more 'primitive' and recognisable candidate for replacement.

I would like to take the view that this process of logical inference to find the *fixpoint* of recursive equations is a suitable candidate to be used in the *logical analysis* suggested by Frege. It seems to be doing the job required: finding a 'complex' expression ('implementation') built up from the existing objects of the language to replace the new symbol one wanted to have in the language. In our example, the 'specification' of 'f' as $f(n) = (n = 0 \rightarrow 0, f(n-1) + 2n - 1)$ was turned into the 'implementation' '$f(n) = n^2$'. The existence of this 'complex' expression shows that the new symbol whose meaning one presumably had in mind beforehand, can be represented in the language by a certain (maybe complex) combination of already existing terms of the language.

The need for a method to solve recursive equations exists as a way of proving the existence of a function which is equivalent to the one being defined recursively. The fact that the use of the mathematical language to write definitions of these kinds, missing an essential part corresponding to the 'imperative' nature of 'definitions by use', like the absence of an element such as 'EVALUATE', has caused this proof of the 'existence lemma' to assume a rôle of such importance in recursion theory. A

certain lack of discipline in the use of the language allows for undesirable, imprecise, useless, definitions to be made. The existence theorem then plays the rôle of showing the 'goodness' of the definition. However, by identifying the fundamental elements one should provide when defining logical signs via rules, one could hope to obtain better definitional mechanisms which would shift the importance of the proof of the existence lemma to a secondary rôle.

A formal system initially thought of as an attempt to clarify the syntax and semantics of the language of intuitionistic mathematics turns out to be an attempt to settle the problem of how one can put more discipline and more coherence in the use of the language of mathematics. However, there still remain some incoherences especially with regards to the relation rule-meaning, and this, I believe, is mainly due to the fact that its semantics is based on the rather unsatisfactory Gentzen semantics. The next chapter is devoted to an analysis of the pitfalls with respect to the relation rules-meaning in what is called 'Gentzen-Prawitz' proof-theoretic semantics, which is followed by another chapter discussing the weaknesses of such a formal system thought of as a clarification of the language of intuitionistic mathematics. It does not convince because it is still insisting on the same semantical inadequacy of the *assertability conditions*-based argument. That is, Prof. P. Martin-Löf's *Intuitionistic Type Theory* is the object of my analysis in Chapter 5.

2.5. Conclusions and Final Remarks

Can proper foundations ever be formulated for the notion of 'computation' in purely linguistic terms? What is there in common among all models of computability, with respect to the relation rules-meaning? We believe that to study the foundations of computability by studying the foundations of language, the nature of concepts like meaning and understanding, can be highly fruitful. It is worth recalling one of Wittgenstein's aphorisms concerning the rôle of language in the whole of philosophy:

> Wir müssen die ganze Sprache durchpflugen.

which has been translated as

We must plough over the whole of language.

(*RFGB*, p. 7e.)

and that is how I have looked at problems related to the logical foundations of computation: concentrating on the question of how meaning is related to rules of *use*.

As I keep on insisting throughout this report, it appears recognisably fair to say that L. Wittgenstein's (later) philosophical views on semantic matters (meaning vs. use) constitute a great source of inspiration for a proper foundation of the theory of computation, where rules and meaning have a close interplay. It is exactly from this philosophical point of view that I have been working on a revision of proof-theoretic semantics for the mathematical language resulting in a particular operational semantics which I suggestively call 'semantics of use'. I have essentially tried to rework the analysis of the nature of *use* in logical terms, initiated by Prof. Michael Dummett and Prof. Dag Prawitz in their research programme on an intuitionistic (linguistic) account of meaning for the language of mathematics. The result is a reformulation of Gentzen-style proof-theoretic semantics, in line with Wittgenstein's *meaning-is-use* because it does take into account both aspects of Wittgenstein's later account of 'propositions' which he calls 'the rules of sentence formation' and 'the use of the sign in the language-games'. As an illustration of the force of this 'semantics of use' I propose a reformulation of a particular formalisation of constructive mathematics, Prof. Per Martin-Löf's *Intuitionistic Type Theory*, which I say can be seen more convincingly as a meaning theory for the mathematical language, provided it carries a better semantical account than Gentzen's traditional proof-theoretic semantics which considers the '*assertability conditions*' alone as the only aspect of the meaning of a proposition. The idea is to use *normalisation* as a semantical notion as a formal counterpart to a reading of Wittgenstein's informal notion of 'language-games'. In fact, the meta-mathematical device of *normalisation* seems to be the key point regarding the relation rules-meaning for the mathematical language, just because it is a 'formal' explanation of the *consequences* of the 'assertion'. I have shown how such

a device can be useful as an alternative reading of Wittgenstein's 'language-games', and at least as appropriate as *Game-Theoretical Semantics* (Hintikka 1983) and *Dialogue-Games* (Lorenzen 1969) in dealing with operational semantics according to Wittgenstein's views on 'language-games' and the language of mathematics (de Queiroz 1988e).

3 *A Critique of Proof-Theoretic Semantics*

3.1. Preamble

After G. Gentzen's development of the *natural deduction* style of doing metamathematics, much work has been done in analysing the methods used in mathematics in a purely syntactic way. Allowing this analysis to be done from an *operational* point of view, by giving explicit rules rather than abstract models, made proof theory a rather appealing alternative for the treatment of mathematical methods using formal systems. Remarkable results like proofs of consistency of parts of mathematics have been obtained since Gentzen's proof of consistency of arithmetic. Also various important techniques to achieve such results have been developed and the use of them have suggested how useful they can be for the treatment of computational aspects of logic and mathematics. Prof. D. Prawitz's *reduction* techniques to prove normalisation theorems, and Prof. P. Martin-Löf's formalisations of constructive mathematics are certainly good examples of this. However, no satisfactory account of the relationship between *introduction* and *elimination* rules for a logical connective seems to have been provided despite many attempts to clarify the matter once and for all. In this chapter I make an attempt to put this into better terms, taking a quite different view and arguing that the meaning of a connective is completely

given by its *introduction* rules plus the corresponding *reduction* rules. The *introduction* and *reduction* rules give the meaning to the connective being defined by indicating when and how[1] the connective can be used, respectively. To *eliminate* the connective is to *compute* the expression, getting rid of the undesired connective.

To be able to analyse mathematical proofs explicitly, treating proofs as proper mathematical objects, Gentzen, who had worked for Hilbert in his programme, developed systems of 'coding' mathematical methods into *inference* rules. The idea was that to pursue a logical analysis of metamathematics in a purely operational way, one would have to code the steps of ordinary mathematical proofs into finitarily graspable rules. Considering it to be a natural way of doing this, Gentzen argued that *introducing* and *eliminating* logical constants were the most basic procedures utilised in a mathematical proof. Therefore, a system of *natural* deduction, as he suggestively decided to call these kinds of presentation systems, would be composed of rules for *introduction* and rules for *elimination* of every logical connective to be used in the particular logic being presented.

> (...) My starting point was this: The formalization of logical deduction, especially as it has been developed by Frege, Russell and Hilbert, is rather far removed from the forms of deduction used in practice in mathematical proofs. Considerable formal advantages are achieved in return.
>
> In contrast, I intended first to set up a formal system which comes as close as possible to actual reasoning. The result was a '*calculus of natural deduction*'. (...)
>
> (Gentzen 1935, p. 68 of English translation in Szabo (ed.) 1969.)

3.2. Introduction

In a *natural deduction* system, according to Gentzen, the *introduction* rules would serve to define the connective in question. The *elimination* rule would then be just a mere consequence of the former. This account of meaning (*introduction* rules as definitions), however, seems to lack the force of a convincing account of operational

[1] Cf. Wittgenstein's '(...) I have said that "$\sim\sim p = p$" gives a rule for the use of the word "not"; it gives the meaning by stating how it functions. (...)', Ambrose (ed.) 1979, Part III (Lectures 1934-35), p. 153.

meaning. I analyse in this report (as well as in de Queiroz 1987c) a revision of proof-theoretic semantics where the definition of a particular logical connective is given by *introduction* rules plus *reduction* rules properly 'internalised', as in the case of the *equality* rules of *Intuitionistic Type Theory*. (As we shall see later on, Prof. Prawitz's *reduction* rules are of a 'meta-level' nature, and the *equality* rules of *Intuitionistic Type Theory* constitute 'internalisations' of Prof. Prawitz's rules.) This particular view, which I call 'semantics of use' in direct reference to Wittgenstein's semantical principle that meaning is determined by use, seems to offer a plausible answer to the still unsettled question: in what respect the *elimination* rules are uniquely determined by the corresponding *introduction* rules plus 'extra requirements' (not specified by Gentzen). I propose them to be the 'internalisation' of the *reduction* rules, showing the effect of particular *elimination* rules on the result of an *introduction* inference.

> (...) Gentzen suggested that it should be possible to formulate in what respect the elimination rule is uniquely determined by the corresponding introduction rule. Indeed, it strikes one as very likely that the elimination rules are in some sense the strongest rules for drawing consequences from an assertion; one sense in which this may be true was suggested in a conjecture by Prawitz. However, the question of the way (if any) in which the elimination rules occupy such a unique position is still unsettled. (...)
>
> (Prawitz 1977, p. 33.)

To be able to achieve his original goal, which was to prove consistency of parts of mathematics without having recourse to truth values but only rules of proof, Gentzen had to develop techniques to *normalise* proofs, and for that he proved what is nowadays known as the Hauptsatz, which says that every proof can be brought into a form not containing any *introduction* step followed by an *elimination* step for the same connective, one immediately after the other in the derivation. From that result he could then prove the consistency of parts of mathematics (like, e.g. elementary number theory) by showing that after *normalising* no derivation would become a *null* proof, which if it were the case would mean that a *contradiction* was being derived from the specific fragment of mathematics being analysed, just

because there is no rule of *introduction* for the *absurdity* constant.

Later on Prof. D. Prawitz carried on improving the *normalisation* procedures, and in his thorough proof-theoretical studies on *natural deduction* systems, published in Prawitz 1965, he classified these *normalisation* procedures and gave them the name of *reduction* rules. What he seems to have overlooked is that these *reduction* rules also play a definite rôle in giving meaning to the connective, and not just the *introduction* rules as himself and Gentzen both have apparently believed. *Normalisation* plays an important rôle in the operational semantics of a particular logical sign, and the *reduction* rules represent this process within proof theory, which is in itself meant to offer an operational semantics to the language of mathematics.

Following this line of thought, we then need to look back at the notion of *reduction* rule.

3.2.1. Reduction rules and the inversion principle

Based on the so-called *Inversion Principle* which says that

> (...) nothing is gained by inferring a formula through introduction for use as a major premiss in an elimination (...)
>
> (Prawitz 1965, p. 33.)

the *reduction* rules have been developed to help in transforming a *proof* (or derivation) in a *natural deduction* system to a normal form. A *proof* is in normal form if it has no *redundant* steps, where a connective is *introduced* and immediately afterwards *eliminated*. As an example:

∧-reduction

$$\dfrac{\dfrac{\dfrac{\Sigma_1}{A_1}\quad\dfrac{\Sigma_2}{A_2}}{A_1 \wedge A_2}}{A_i} \quad \rightarrow \quad \dfrac{\Sigma_i}{A_i} \qquad (i = 1,2)$$

(where '↠' means 'reduces to') which means that having derivations Σ_1 and Σ_2 for each of the formulas A_1 and A_2, respectively, we can *introduce* a logical symbol in our language to represent the *conjunction* of the two formulas and the result of the *elimination* of such a logical symbol is to get back to one of the two derivations of A_1 or A_2, the resulting *introduction* step being redundant. One aspect of our 'pre-conceived' meaning of *conjunction*, assuming we have one, is then represented by the fact that we can get back either one of the two *conjuncts*, if we apply either *elimination* operation to the result of an *introduction* inference. In other words, to represent our pre-conceived meaning of the connective in question, in addition to saying *when* we can correctly assert a conjunctive sentence (the 'grammar' aspect), we have to say *how* to use it in our language (the 'consequences' aspects: the effect of splitting operations on the result of a pairing inference), and that seems to be the rôle of the *reduction* rules. In other words, these rules show the effect of particular *elimination* operations on a formula derived from an *introduction* inference.

Other useful examples[2]:

$\vee$-*reduction*

$$\dfrac{\dfrac{\Sigma}{A_i}}{\dfrac{A_1 \vee A_2}{B}}\quad\dfrac{[A_1]}{\dfrac{\Sigma_1}{B}}\quad\dfrac{[A_2]}{\dfrac{\Sigma_2}{B}} \quad\twoheadrightarrow\quad \dfrac{\dfrac{\Sigma}{[A_i]}}{\dfrac{\Sigma_i}{B}}$$

where $[A_i]$ denotes the set of assumptions in Σ_i that are closed by the corresponding $\vee$-*elimination* ($i = 1, 2$).

$\supset$-*reduction*

$$\dfrac{\dfrac{\Sigma_1}{A_1}\quad\dfrac{\dfrac{[A_1]}{\dfrac{\Sigma_2}{A_2}}}{A_1 \supset A_2}}{A_2} \quad\twoheadrightarrow\quad \dfrac{\dfrac{\Sigma_1}{[A_1]}}{\dfrac{\Sigma_2}{A_2}}$$

[2]for a more detailed explanation, see, e.g., Prawitz 1971.

where $[A_1]$ denotes the set of assumptions in Σ_2 that are closed by the $\supset$-*introduction*.

$\forall$-*reduction*

$$\dfrac{\dfrac{\dfrac{\Sigma(a)}{A(a)}}{\forall x.A(x)}}{A(t)} \quad \twoheadrightarrow \quad \dfrac{\Sigma(t)}{A(t)}$$

where $\Sigma(a)$ indicates that the formulas in the part of the derivation above $A(a)$ may contain the parameter a. The deduction $\Sigma(t)$ is obtained from $\Sigma(a)$ by replacing every occurrence of a by the term t.

$\exists$-*reduction*

$$\dfrac{\dfrac{\dfrac{\Sigma_1}{A(t)}}{\exists x.A(x)} \quad \dfrac{[A(a)]}{\dfrac{\Sigma_2(a)}{B}}}{B} \quad \twoheadrightarrow \quad \dfrac{\dfrac{\Sigma_1}{[A(t)]}}{\dfrac{\Sigma_2(t)}{B}}$$

where $[A(a)]$ denotes the set of assumptions in $\Sigma_2(a)$ which are *closed* by the $\exists$-*elimination*.

In fact, the *Inversion Principle*, also used by Prof. Lorenzen in his development of an 'operative' view of logic and mathematics which reflects the symmetry between *introduction* and *elimination* rules, has a definite rôle to play in the case where meaning is given by natural deduction-like proof rules because it shows the second fundamental aspect of the use (operational meaning) by exhibiting, as I have just said, the effect of particular *elimination* rules on the result of an *introduction* inference.

Prof. Aczel's idea of introducing new symbols by using what he calls '*systems*' seems to be a much more natural and coherent view, where the '*reduction*' rules are given, showing how the *system* works:

Definition 3.1. A *lambda system* for an explicitly closed family $\mathcal{F}$ consists of two $\mathcal{F}$-functionals $\lambda : \mathcal{F}_1 \rightarrow \mathcal{F}_0$ and $\mathrm{APP} : \mathcal{F}_0 \times \mathcal{F}_0 \rightarrow \mathcal{F}_0$ such that

$$\mathrm{APP}(\lambda x f(x), a) = f(a)$$

for all f in $\mathcal{F}_1$ and a in $\mathcal{F}_0$.

(Aczel 1980, p. 38.)

Let us call our *lambda system* '**Π**'. We then look at the definition of 'Π' as (borrowing a bit of notation from *Intuitionistic Type Theory*):

Π-*introduction*

$$\frac{\begin{array}{c}(x \in \mathcal{F}_0)\\ f \in \mathcal{F}_1\end{array}}{\lambda x f \in \mathcal{F}_1 \to \mathcal{F}_0}$$

Π-*reduction*

$$\frac{\begin{array}{cc}\multicolumn{2}{c}{(x \in \mathcal{F}_0)}\\ a \in \mathcal{F}_0 & f \in \mathcal{F}_1\end{array}}{\mathrm{APP}(\lambda x f, a) = f(a) \in \mathcal{F}_0}$$

Another example of Prof. Aczel's use of *reduction* rules to define his *systems* is the following:

Definition 5.6. A *pairing system* consists of $\mathcal{F}$-functions $\mathrm{PAIR} : \mathcal{F}_0 \times \mathcal{F}_0 \to \mathcal{F}_0$ and $p, q : \mathcal{F}_0 \to \mathcal{F}_0$ such that for all objects a, b

$$p(\mathrm{PAIR}(a,b)) = a \qquad q(\mathrm{PAIR}(a,b)) = b.$$

(Ibid., p. 48.)

Naming our *pairing system* as '**Σ**', we look at the definition of '**Σ**' as:

Σ-*introduction*

$$\frac{a \in \mathcal{F}_0 \quad b \in \mathcal{F}_0}{\mathrm{PAIR}(a,b) \in \mathcal{F}_0}$$

Σ-*reduction*

$$\frac{a \in \mathcal{F}_0 \quad b \in \mathcal{F}_0}{p(\mathrm{PAIR}(a,b)) = a \in \mathcal{F}_0} \qquad \frac{a \in \mathcal{F}_0 \quad b \in \mathcal{F}_0}{q(\mathrm{PAIR}(a,b)) = b \in \mathcal{F}_0}$$

How can the meaning of '*lambda systems*' and '*pairing systems*' be understood if not by the *reduction* rules? One can only conclude that there cannot be any hope of getting the *elimination* rule of a logical connective just from its *introduction* rule. And this is exactly because the meaning (use) is not made completely explicit until the *reduction* rules are provided.

The controversy about which of the rules (*introduction* or *elimination*) gives the meaning to the connective being defined can be overcome by accepting that both have their parts. They constitute distinct aspects of the use (meaning) of the specific logical sign. Therefore the rules representing the *Inversion Principle* (rules of *reduction*) assume a definite rôle in a system of rules of definition, by showing how *elimination* rules operate on the result of an *introduction* inference.

What appears to have been a misleading assumption saying that *introduction* rules are definitions, consequently a misconcetion of the semantical rôle of proof rules has even left room for engagement in what might be called 'false' problems. The failure of most attempts at obtaining the *elimination* rules solely from the rules of *introduction* in natural deduction presentation systems have only shown that *introduction* and *elimination* inference rules as understood by Gentzen and Prawitz in their studies on natural deduction, are distinct aspects of the meaning of a logical sign. The *reduction* rules come in as a meta-device with the rôle of explaining the effect of *elimination* steps on *introduction* inferences.

3.2.2. Prof. Prawitz's 'elimination from introduction' general procedure

Having developed the *normalisation procedures* based on the *reduction* rules Prof. Prawitz seems to have overlooked the definitive rôle of such rules in giving the

meaning to the logical connective. In his semantical studies on the meaning and completeness of the logical constants he tried to build up a *general form of schematic introductions* aiming at a uniform way of obtaining the *elimination* rules from the corresponding schematic *introduction* rules. Quoting Gentzen's words

> (...) Gentzen suggested that "it should be possible to display the elimination rules as unique functions of the corresponding introduction rules on the basis of certain requirements". (...)
>
> (Prawitz 1978, pp. 36–7.)

and getting inspiration from them, Prof. Prawitz seems to overlook the fact that the *reduction* (*normalisation*) rules show the aspect of the meaning which is not captured by the *introduction* rules alone, by showing the effect of particular *elimination* rules on the result of an *introduction* inference. One can easily show, as I have done in a somewhat modified form in de Queiroz 1988, that there cannot be any hope of having this general procedure for obtaining the *elimination* rules just from the *introduction* rules if we consider *natural deduction* systems (rules operate on *sentences* rather than *sequents*) with only logical rules, i.e., no structural rules like *cut* or something of that sort. This is because the *introduction* rules only cover one of the aspects of the use (therefore meaning) of the particular connective. In other words, to define the meaning of a logical connective, one has to give more than just the conditions under which the connective can be correctly used ('*grammatical*'), aspect which is captured by the *introduction* rules. The other aspect of the use (meaning), which tells what is the *utility* of the connective, or, in other words, what *conclusions* can be drawn from a sentence containing the connective as the major one (*elimination* rules), still have to be given independently and cannot be 'obtained' from the first aspect. For the sake of clarity, and to make this chapter reasonably self-contained, I repeat it here, this time trying to give a more detailed account of Prof. Prawitz's semantically unsound attempt.

The *General Form of Schematic Introductions* is given as something like:

$$\frac{\begin{array}{ccccccc} \vdots & \vdots & & [H^i_{j,1}] & [H^i_{j,2}] & \cdots [H^i_{j,h^i_j}] & \\ G^i_1 & G^i_2 & \cdots & & G^i_j & \cdots & G^i_{p_i} \end{array}}{\varphi(F_1,\ldots,F_n)}$$

where F_l, G^i_j and $H^i_{j,k}$ are formula schemata built up of syntactical variables for sentences and sentential operators under the following conditions:

a) $i = 1,\ldots,s$ and s is the number of *introduction* rules for the particular connective 'φ';

b) F_l $(l = 1,\ldots,n)$ represent the formulas in the conclusion;

c) G^i_j $(j = 1,\ldots,p_i)$ represent the formulas in the premiss of the i-th *introduction* rule for the connective;

d) $H^i_{j,k}$ $(j = 1,\ldots,p_i)$ $(k = 1,\ldots,h^i_j)$ represent the assumption-formulas for the j-th formula G^i_j in the premiss of the i-th *introduction* rule;

e) p_i represents the number of formulas in the premiss of the i-th *introduction* rule;

f) h^i_j represents the number of assumption-formulas for the j-th formula of the i-th *introduction* rule.

A rule which is an instance of such a schema infers a formula $\varphi(A_1,\ldots,A_n)$ from p_i premisses $B^i_1,\ldots,B^i_{p_i}$ and may bind assumptions of the form $C^i_{j,1},\ldots,C^i_{j,h^i_j}$ that occur above the j-th premiss B^i_j, where the A's, B's and C's are formulas obtained from the corresponding F's, G's and H's by a simultaneous substitution of formulas for the syntactical variables.

The *elimination* rules for an n-ary sentential operation 'φ' whose *introduction* rule is the one just stated, are said to be given by one rule schema of the form:

$$\frac{\varphi(F_1,\ldots,F_n) \quad \begin{array}{c}\Gamma_1\\ A\end{array} \quad \cdots \quad \begin{array}{c}\Gamma_i\\ A\end{array} \quad \cdots \quad \begin{array}{c}\Gamma_s\\ A\end{array}}{A}$$

with s minor premisses of the form A and where assumptions occurring above the i-th ($i \leq s$) minor premiss may be bound when they are of one of the forms shown in Γ_i ($i = 1, \ldots, s$). For every *introduction* rule, a corresponding 'Γ' is built as:

$$\Gamma_i = \bigwedge_{k \leq h_1^i} H_{1,k}^i \rightarrow G_1^i, \ldots, \bigwedge_{k \leq h_{p_i}^i} H_{p_i,k}^i \rightarrow G_{p_i}^i$$

where

a) $\bigwedge$ is a sign for iterated conjunction;

b) If $h_j^i = 0$ (i.e., there are no assumptions), then the iterated conjunction is just to be left out and the implication is replaced by its consequent;

c) If $p_i = 0$, then there are no assumptions occurring above the i-th minor premiss bound by the inference;

d) If $s = 0$, there is only one premiss, namely $\varphi(F_1, \ldots, F_n)$.

However, the above schema does not seem to be achieving its original goal. First of all, *conjunction* and *implication* are taken as primitives, to explain themselves later on (the definition of 'Γ' involves '$\wedge$' and '$\rightarrow$'). Secondly, if we take the schema seriously and apply it to the connective '$\rightarrow$' (*implication*), we arrive at an *elimination* rule which cannot be taken as a proper *elimination* rule for the connective in question (see Example 3.2.2.3).

Examples. (Developed from the examples briefly mentioned in Prawitz 1978, p. 35.)

Example 3.2.2.1 Conjunction ('$\wedge$')

$s = 1$ (1 *introduction* rule)

$p_1 = 2$ (2 formulas in the premiss of the 1st *introduction* rule)

$h_1^1 = h_2^1 = 0$ (no assumptions for each of the formulas in the premiss of the *introduction* rule)

$G_j^1 = F_j (j = 1, 2)$ (formulas in the premiss)

$\wedge$-*introduction*

$$\frac{F_1 \qquad F_2}{\wedge(F_1, F_2)}$$

So,

$$\Gamma_1 = \bigwedge_{k \leq 0} H^i_{1,k} \rightarrow G^1_1, \bigwedge_{k \leq 0} H^i_{p_i,k} \rightarrow G^1_2$$

hence,

$$\Gamma_1 = F_1, F_2$$

therefore,

$\wedge$-*elimination*

$$\frac{\wedge(F_1, F_2) \qquad \begin{array}{c} F_1, F_2 \\ \vdots \\ A \end{array}}{A}$$

Example 3.2.2.2 Disjunction ('$\vee$')

$s = 2$ (2 *introduction* rules)

$p_1 = p_2 = 1$ (1 formula in the premiss of each *introduction* rule)

$h^1_1 = h^2_1 = 0$ (no assumptions)

$G^i_1 = F_i \quad (i = 1, 2)$ (one formula of the conclusion for the premiss of each *introduction* rule)

$\vee$-*introduction*

$$\frac{F_1}{\vee(F_1, F_2)} \qquad \frac{F_2}{\vee(F_1, F_2)}$$

So,

$$\Gamma_1 = \bigwedge_{k \leq 0} H^1_{1,k} \rightarrow G^1_1$$

$$\Gamma_2 = \bigwedge_{k \leq 0} H^2_{1,k} \rightarrow G^2_1$$

hence,

$$\Gamma_1 = F_1$$

$$\Gamma_2 = F_2$$

therefore,

$\vee$-*elimination*

$$\frac{\vee(F_1, F_2) \qquad \begin{array}{cc} F_1 & F_2 \\ \vdots & \vdots \\ A & A \end{array}}{A}$$

Example 3.2.2.3 Implication ('$\rightarrow$')

$s = 1$ (1 *introduction* rule)

$p_1 = 1$ (1 formula in the premiss)

$h^1_1 = 1$ (1 assumption-formula for the 1st formula in the premiss)

$G^1_1 = F_2$ (the 2nd formula in the conclusion is the formula in the premiss)

$H^1_{1,1} = F_1$ (the 1st formula in the conclusion is the assumption-formula)

$\rightarrow$-*introduction*

$$\frac{\begin{array}{c}[H^1_{1,1}] \\ \vdots \\ G^1_1\end{array}}{\rightarrow (F_1, F_2)} \qquad \text{which is the same as} \qquad \frac{\begin{array}{c}[F_1] \\ \vdots \\ F_2\end{array}}{\rightarrow (F_1, F_2)}$$

and

$$\Gamma_1 = \bigwedge_{k \leq 1} H^1_{1,k} \rightarrow G^1_1$$

which means

$$\Gamma_1 = H^1_{1,1} \rightarrow G^1_1$$

substituting F_1 and F_2 into $H^1_{1,1}$ and G^1_1 respectively, we have

$$\Gamma_1 = F_1 \rightarrow F_2$$

We then arrive at an *elimination* rule for '$\rightarrow$' as:

$\rightarrow$-elimination

$$\frac{\rightarrow(F_1, F_2) \qquad \begin{array}{c} F_1 \rightarrow F_2 \\ \vdots \\ A \end{array}}{A}$$

which, as already pointed out by Sundholm, can never be taken as *modus ponens* (*$\rightarrow$-elimination*).

In fact, this schema seems to be producing nothing more than a *cut* rule for every n-ary connective defined by the schematic introduction. Rather than a rule to eliminate '$\rightarrow$', the above rule looks more like a schema which one can use for any logical symbol, like e.g.:

$$\frac{\wedge(F_1, F_2) \qquad \begin{array}{c} F_1 \wedge F_2 \\ \vdots \\ A \end{array}}{A} \qquad\qquad \frac{\vee(F_1, F_2) \qquad \begin{array}{c} F_1 \vee F_2 \\ \vdots \\ A \end{array}}{A}$$

If we think in terms of a sequent calculus where the *proves* ('$\vdash$') symbol is part of the system and the assumptions are 'carried around', one can readily see that Prof. Prawitz's schema is just producing an instance of the *cut* rule for the corresponding connective:

Example 3.2.2.1 (Conjunction)

$$\frac{\vdash \wedge(F_1, F_2) \qquad \{F_1, F_2\} \vdash A}{\vdash A}$$

Example 3.2.2.2 (Disjunction)

$$\frac{\vdash \vee(F_1, F_2) \qquad \{F_1\} \vdash A \qquad \{F_2\} \vdash A}{\vdash A}$$

Example 3.2.2.3 (Implication)

$$\frac{\vdash \rightarrow (F_1, F_2) \qquad \{F_1 \rightarrow F_2\} \vdash A}{\vdash A}$$

At this point it becomes difficult to justify and to maintain the coherence of the whole body of concepts introduced by proof theory. No wonder important questions arise, such as:

1. Does a system of *natural deduction* of the kind we are considering, where *introduction* and *elimination* are assumed to be the only kinds of rules available, and those rules operate over sentences rather than sequents, accommodate an extra 'meta-concept' like a *cut* rule ?

2. Can a *cut* rule be taken as an *elimination* rule ?

The point I attempt to make when arguing against Prof. Prawitz's schema, as shown above, is that there cannot be any hope of getting the *elimination* rule of a logical connective just from its *introduction* rule. They constitute distinct aspects

of the meaning of a logical sign. So, either both *introduction* and *elimination* rules have to be provided, which works perfectly well for simple cases (not necessarily for more complicated cases such as modal logics, given that there must be something outside proof rules, like 'possible worlds', to fill in the semantical gap: e.g., the '*necessitation*' rule is an *introduction* rule for '$\Box$', and yet, it does not suffice to give meaning to '$\Box$'), or one assumes that the definition (only partially given by the *introduction* rules) is not complete until the *reduction* (*normalisation*) rules, when properly 'internalised' like in Prof. Martin-Löf's *Intuitionistic Type Theory*, are provided. They cover one of the fundamental aspects of the use (therefore meaning) of the corresponding connective by showing the effect of particular *elimination* rules on the result of an *introduction* inference.

In an attempt to correct Prof. Prawitz's schema, Schroeder-Heister argues that Prof. Prawitz's mistake was to confuse '$\Rightarrow$' ('derivability') with '$\rightarrow$' ('implication'). He then offers an alternative schema which is based on his extended notion of deduction where rules are taken as assumptions rather than only formulas as in ordinary natural deduction calculi. By adopting the same semantical principle set by Gentzen which says that *introduction* rules define the connective, Schroeder-Heister appears to comit the same fundamental mistake. The result is a schema which does not produce proper *elimination* rules.

By introducing a natural deduction calculus with rules of higher levels, i.e., a calculus where rules and not only formulae can be introduced as assumptions, it is argued that the *elimination* rule for '$\rightarrow$' can be formulated as

$$\frac{A \rightarrow B \qquad \begin{array}{c}(A \Rightarrow B)\\ \vdots \\ C\end{array}}{C}$$

where '$(A \Rightarrow B)$' means a rule taken as an assumption.

The point made is that Prof. Prawitz's mistake was due to the confusion between the logical connective '$\rightarrow$' and the rule (derivation) arrow '$\Rightarrow$'. By making this

distinction one could make the general schema for producing *elimination* rules from *introduction* rules work as desired.[3]

The standard form of basic rule schemata for an n-ary operator 'S' is given as:

Schemata for *S-introduction* (*S*I):

$$\frac{\Phi_1(A_1,\ldots,A_n)}{SA_1\ldots A_n}\ \ldots\ \frac{\Phi_m(A_1,\ldots,A_n)}{SA_1\ldots A_n},$$

Schema for *S-elimination* (*S*E):

$$\frac{SA_1\ldots A_n \qquad \overline{\Phi_1(A_1,\ldots,A_n)}\ \ldots\ \overline{\Phi_m(A_1,\ldots,A_n)}}{A}$$

where $\Phi_1(A_1,\ldots,A_n),\ldots,\Phi_m(A_1,\ldots,A_n)$ are lists of rule schemata associated with the operator 'S'.

According to this general pattern, the basic rules for '$\vee$', '$\wedge$' and '$\rightarrow$' are given as:

$$(\vee I)\quad \frac{A}{A\vee B}\quad\frac{B}{A\vee B} \qquad (\vee E)\quad \frac{A\vee B \quad \overline{C}\quad \overline{C}}{C}$$

$$(\wedge I)\quad \frac{A\quad B}{A\wedge B} \qquad (\wedge E)\quad \frac{A\wedge B \quad \overline{C}}{C}$$

$$(\rightarrow I)\quad \frac{\overline{B}}{A\rightarrow B} \qquad (\rightarrow E)\quad \frac{A\rightarrow B \quad \overline{C}}{C}$$

[3]This point was made in a letter from Dr. P. Schroeder-Heister, to whom I would like to express my deepest gratitude.

Although succeeding in producing a more coherent schema than Prof. Prawitz's, Schroeder-Heister does not seem to be too convincing. The resulting *elimination* rules for '$\wedge$' and '$\rightarrow$' are not exactly what is expected:

$$(\wedge E) \qquad \frac{A \wedge B}{A} \qquad \frac{A \wedge B}{B}$$

$$(\rightarrow E) \qquad \frac{A \quad A \rightarrow B}{B}$$

Moreover, all *elimination* rules seem to be using a 'third party' ('C', in particular), as is the case for *cut* rules. This gives the impression that there is a confusion between a *cut* rule and an *elimination* rule.

Furthermore, to verify that the *reduction* rules are indeed fundamental rules is not a difficult task. Looking at some more recent formal languages of modern mathematics (λ-Calculus, *Intuitionistic Type Theory*, Category Theory, etc.) one can find a number of examples. The rules for Π-*equality* of *Intuitionistic Type Theory*, for example, represent two of the fundamental laws of the lambda-calculus:

Π-*equality*

$$\frac{a \in A \qquad \begin{matrix}(x \in A)\\ b(x) \in B(x)\end{matrix}}{Ap((\lambda x)b(x), a) = b(a) \in B(a)}$$

$$\frac{c \in (\Pi x \in A)B(x)}{c = (\lambda x)Ap(c, x) \in (\Pi x \in A)B(x)}$$

The first one can be seen as the β-conversion rule[4]

$$(\lambda x.M)N = M[x := N], \quad (\beta - \text{conversion});$$

and the second one is the η-conversion rule[5]

[4]Barendregt 1981, definition 2.1.4.I, p. 23.

[5]Ibid., definition 2.1.28., p. 32.

$$\lambda x.Mx = M \quad (\eta - \text{conversion})$$

Can there exist any use for the lambda calculus without at least the β-conversion rule? Can the operation of '*apply*' be derived from the *introduction* (*abstraction*) rule of 'λ'? Are β- and η- conversion 'derived' rules or are they primitively defined rules? These questions all arise, as far as I can see, in demonstrating the insufficiency of the *introduction* rules with respect to the meaning of the corresponding logical connective.

The rules for bicartesian closed categories, given by P. Dybjer when attempting to relate some formal rules of Category Theory to rules of Type Theory (in Dybjer 1985, pp. 177–178.), constitute another example of how fundamental the *reduction* rules are:

Rules for binary coproduct
(weak rules)

$$\frac{f : A \to C \quad g : B \to C}{[f,g] \circ inl = f : A \to C}$$

$$\frac{f : A \to C \quad g : B \to C}{[f,g] \circ inr = g : A \to C}$$

which correspond in type theory to the *equality* rules for the sum of two types ('+' *system*, if we insist on Prof. Aczel's notion of *systems*):

+-*equality*

$$\frac{a \in A \quad \begin{matrix}(x \in A)\\ d(x) \in C(i(x))\end{matrix} \quad \begin{matrix}(y \in B)\\ e(y) \in C(j(y))\end{matrix}}{D(i(a),(x)d(x),(y)e(y)) = d(a) \in C(i(a))}$$

$$\frac{\begin{array}{ccc} & (x \in A) & (y \in B) \\ b \in B & d(x) \in C(i(x)) & e(y) \in C(j(y)) \end{array}}{D(j(b), (x)d(x), (y)e(y)) = e(b) \in C(j(b))}$$

Rules for binary product

(weak rules)

$$\frac{f : C \to A \quad g : C \to B}{fst \circ < f, g >= f : C \to A}$$

$$\frac{f : C \to A \quad g : C \to B}{snd \circ < f, g >= g : C \to B}$$

which correspond to the rules for disjoint union of a family of types, or, equivalently, to a *pairing system* 'Σ' (in Martin-Löf 1984, p. 45 & p. 46.):

Σ-*equality*

$$\frac{a \in A \quad b \in B(a)}{p(\mathrm{PAIR}(a, b)) = a \in A}$$

$$\frac{a \in A \quad b \in B(a)}{q(\mathrm{PAIR}(a, b)) = b \in B(a)}$$

Rule for exponentiation

(weak rule)

$$\frac{f : C \times A \to B}{apply \circ < \Lambda(f) \circ fst, snd >= f : C \times A \to B}$$

which corresponds to one of the *equality* rules for the cartesian product of families of types (Π-*equality*), in particular the *equality* rule which represents the lambda calculus' β-conversion rule (as seen above).

One could then ask: is the operator '[ ]', which works as an *elimination* operator, in any sense derived from '*inl*', '*inr*'? A similar question could be asked for '*fst*', '*snd*' and '< >'. And last, but not least, '*apply*' and 'Λ'.

By taking my point of view on the importance of the *reduction* rules, and the insufficiency of the *introduction* rules, one can understand why the language of, e.g., category theory, has gained a certain popularity within the computer science community: it makes available in the language not only each symbol to be *introduced* but also its projections and its corresponding *reduction* (e.g., Dybjer's '*inl*', '*inr*' and '[ ]'; '< >' and '*fst*', '*snd*'; 'Λ' and '*apply*'). This is probably what is missing in proof theory in order to make it a 'comfortable' language to express constructions. I see the work of Prof. Martin-Löf on his *Intuitionistic Type Theory*, which shall be discussed in the following chapters, as a remarkable step towards this objective.

From my own understanding of the semantical issue involved here, I can only conclude that Prof. Prawitz's mistake when demonstrating a certain 'general' schema for obtaining *elimination* rules from *introduction* rules was of a semantical nature and had nothing to do with the specific schema presented.

3.3. Prof. Prawitz's Development

The most extensive studies of proof-theoretic techniques after Gentzen has probably been done by Prof. D. Prawitz. Various important techniques and results on normalisation and strong normalisation for different logics have been obtained by this creative Scandinavian logician. His thorough studies on Natural Deduction calculi have made an important contribution to the development of a more operational account of meaning for the mathematical language.

Semantical investigations have been pursued by Prof. Prawitz, especially in connection with Prof. Dummett's development of an intuitionistic account of meaning based on the interplay between meaning and use. Both seemed to have analysed deeply the relevant aspects of the use of an assertive sentence, particularly with regards to its meaning. The result of these semantical studies, however, does not

seem to have been convincingly completed. The apparent attachment to Gentzen's view on the rôle of each of the different kinds of proof rules, has led to an incomplete and unconvincing connection between the 'meaning as use' semantic notion and the Gentzen-style proof-theoretic semantics. Although admitting that the elimination rules play a part in the definition of the use of the particular connective, Prof. Prawitz seems to stick to Gentzen's principles and takes the introduction rules as the ones which completely define the meaning of the connective arguing for some kind of 'priority' of the latter rules over the former ones. Even after saying that not all relevant aspects of the use are covered by the introduction rules, he tries to devise a general schema for obtaining elimination rules from the corresponding introduction rules. This results in a failure, which I analysed above and elsewhere (de Queiroz 1987a, 1988). The fact is that the traditional proof-theoretic semantics developed by Gentzen, certainly due to its rather 'static' notion of meaning, does not seem to tie in very well with the more intuitive and more convincing 'meaning as use' (operational) semantic principle.

3.3.1. Prof. Dummett's aspects of use and the rôle of proof rules in proof theory.

In his attempt to study the relationship between meaning and proofs, aiming at a more convincing account of operational meaning for the mathematical language as required by his and Prof. Dummett's interpretation of intuitionistic logic, Prof. Prawitz attempts to analyse how Prof. Dummett's points on meaning and use relate to proof-theoretic notions. An important point stressed in his analysis is the fact that to have the knowledge of the meaning of a sentence it does not suffice to learn to recognise a proof of the sentence. It is necessary to know the kinds of commitments that are made when the sentence is asserted. Speaking in terms of proof-theoretic inference rules, this is just saying that the 'introduction' rule, which reflects the first of these two aspects of the meaning of a sentence (the ability to recognise a proof of it), does not cover all the relevant aspects.

It thus seems more reasonable to say with the intuitionists that the meaning of a sentence is determined by the way it can be proved. Prof. Dummett suggests that the knowledge of the meaning of a sentence manifests itself in our capacity to recognise a proof of it when one is presented to us. But this conclusion may be premature at this point because, as stressed by Prof. Dummett, there are different features of the use of a sentence. What we learn is not only how to prove statements but more generally how to use statements in proofs.

Prof. Dummett points out that quite generally, there are two aspects of the use of an (assertive) sentence: (1) the conditions under which it can be correctly asserted, and (2) the commitments made by asserting it. In the case of mathematics, aspect (1) is expressed in the rules for inferring the sentence, and aspect (2) in the rules for drawing consequences from the sentence. When we learn to use sentences, what we learn are these two kinds of rules.

(Prawitz 1977, pp. 6–7.)

Looking at Prof. Prawitz's words, what we can immediately say is that it strongly suggests the existence of a close relationship between the use of a sentence and its meaning. Furthermore, by saying that there are two aspects of the use of an (assertive) sentence, and that they correspond to the aspects (1) and (2) above, he seems to admit that the meaning (use) is not made up of the condition (1) alone. This condition (1) in fact corresponds exactly to the proof-theoretic notion of introduction rule, which is the rule for inferring the sentence. The apparently evident conclusion Prof. Prawitz seems to draw, which is expressed in the last sentence of the above quotation, comes as a confirmation of the insufficiency of the introduction rule.

Nevertheless, perhaps in an attempt to be coherent with Gentzen's fundamental principle which says that the introduction rule defines the connective, the elimination rule being a mere consequence of it, Prof. Prawitz insists on a kind of 'priority' of introduction although admitting its insufficiency with regards to the meaning of the connective.

3.3.2. Insufficiency of introduction rules

To what extent the condition for asserting a sentence (introduction inference) can carry all the information necessary to understand the meaning of the sentence seems

to be fairly clear to Prof. Prawitz.

> (...) a person knowing the condition for asserting $A \wedge B$ will of course know a procedure that makes it acceptable to him to infer $A \wedge B$ from the premisses A and B, i.e., in every situation this inference is acceptable to him. But it is conceivable that although he knows the condition for asserting $A \wedge B$, he does not happen to know the procedure which given a canonical proof of $A \wedge B$ yields a canonical proof of A. As long as he does not know this procedure, the inference of A from $A \wedge B$ will not be acceptable. Hence, if we are of the opinion that a person's grasp of the meaning of $A \wedge B$ is insufficient, as long as he does not know the acceptability of this inference, we must maintain that a grasp of this meaning involves something more than the knowledge of the condition for asserting the sentence.
>
> (Ibid., p. 31.)

What I want to say is that this 'something more than the knowledge of the condition for asserting the sentence' does not necessarily have to be the elimination rule. The principle laid down by Gentzen which says that the elimination rule is a mere consequence of the definition of the connective can still be maintained. And this is done by taking the 'internalisation' of the reduction rules as the extra information needed. I just need to enrich the partial definition given by the introduction rules by incorporating the other aspect of use, which tells us what kind of commitments one makes when asserting a sentence with this particular connective as the major one. Considering the rules of introduction and reduction as constituents of the definition of a particular connective, Gentzen's claim about the unique position of the elimination rule can be kept intact. Even more, it can be fulfilled formally (described in de Queiroz 1988, de Queiroz and Maibaum 1987a).

3.3.3. The rôle of reduction rules

The reduction rules are framed in terms of the introduction rule plus a particular elimination. Nevertheless, the fact that it is indeed framed in terms of the introduction rule cannot be taken as confirming that the canonical proofs given by the introduction rule(s) constitute the meaning of the sentence because, as has been

insistently stated, the rules showing the canonical proofs do not contain all the information necessary for the understanding of the meaning of the sentence. The reduction rules play an important rôle in the definition of a particular connective because they include the extra information necessary and do not constitute an 'elimination' rule, which keeps Gentzen's claim intact. They provide the extra information by showing the 'use' of the connective. For this reason I say that the meaning of a sentence is better viewed as its 'use' rather than as its 'content' (canonical proofs).

What appears to have been a similar point of view was expressed in some studies on meaning theory by Prof. P. Martin-Löf and P. Hancock ('*Lectures in meaning theory*', unpublished at the time of publication of Prawitz 1977.). The approach is briefly discussed by Prof. Prawitz, and the idea seemed to be along the following lines: to grasp the meaning of a sentence would consist in the knowledge of a procedure which, given canonical proofs of the premisses of a corresponding elimination, would yield a canonical proof of the conclusion. As noticed by Prof. Prawitz, it would be in accordance with their ideas to equate the understanding or the knowledge of the meaning of a sentence with the ability, when knowing canonical proofs of the premisses of a corresponding elimination, to find a canonical proof of the conclusion from the result of applying the procedure which makes the elimination acceptable (in the sense discussed by Prof. Prawitz) to the given canonical proofs.

As we shall see later on, the proof-theoretic 'reduction' rules, when properly 'internalised' as in *Intuitionistic Type Theory*, seem to be doing exactly the 'coding' of that particular procedure.

Arguing that the canonical proofs of a particular sentence constitute its meaning because even within the above view the procedures that make the elimination rule acceptable are framed in terms of canonical proofs, Prof. Prawitz seems to reject what appeared to be very clear: the insufficiency of the introduction rules.

(...) If we adopt this view, we should accordingly modify the condition for asserting the sentence to comprise now also the knowledge of the procedures that justify the eliminations. For instance, the condition for asserting $A_1 \wedge A_2$ should now also embrace knowledge of the two procedures $p_i (i = 1 \text{ or } 2)$ which given a canonical proof of $A_1 \wedge A_2$ yield a canonical proof of A_i. After this

modification, the condition for asserting a sentence can still be taken to constitute the meaning of the sentence.

The canonical proofs and thus the operations associated with Gentzen's introduction rules will have a kind of priority also on this view since the operations associated with the elimination rules are framed in terms of canonical proofs. The meaning is only enriched by taking into account also the elimination rules, or, in other words, the rules for drawing consequences from a sentence are now contributing to the meaning of the sentence by being in a way incorporated into the rules for asserting the sentence.

(Ibid., pp. 32–3.)

The notion of reduction (normalisation) of proofs seems to comprise the procedures that justify the eliminations, as required by Prof. Prawitz. The corresponding rules of reduction are complementing the information given by the rules which say under which conditions the sentence can be correctly asserted by showing the kinds of conclusion one can draw from the sentence. Therefore, to say that the condition for asserting a sentence can still be taken to be a necessary part of its meaning, is all perfectly acceptable. However, it also seems tenable to say that the reduction rules constitute part of the meaning of the connective, because they bring about the sufficient conditions to understand the meaning, in addition to the necessary conditions shown by the introduction rules.

Here is an example of the rule of reduction for 'conjunction' ('$\wedge$'), taken from Prawitz 1971 (where '$\twoheadrightarrow$' is used as 'reduces to'):

$\wedge$-reduction

$$\dfrac{\dfrac{\dfrac{\Sigma_1}{A_1}\quad\dfrac{\Sigma_2}{A_2}}{A_1 \wedge A_2}}{A_i} \quad \twoheadrightarrow \quad \dfrac{\Sigma_i}{A_i} \qquad (i = 1, 2)$$

By considering these rules as 'meta'-rules it becomes difficult to realise that the knowledge of the procedures that justify the elimination rules is comprised by the reduction rules.

In Prof. Martin-Löf's *Intuitionistic Type Theory* (see a more detailed explanation

in chapter 5 below), where those kinds of rules are 'internalised' and named 'equality' rules, the corresponding rules are presented as:

Σ-*equality*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{B}(a)}{\mathtt{FST}(\mathtt{pair}(a,b)) = a \in \mathbf{A}} \qquad \frac{a \in \mathbf{A} \quad b \in \mathbf{B}(a)}{\mathtt{SND}(\mathtt{pair}(a,b)) = b \in \mathbf{B}(a)}$$

and one can see that the 'operational' behaviour (or use) of the connective 'Σ' (pairing propositional connective), whose 'constructor' is '`pair`' and 'destructors' '`FST`' and '`SND`', is made explicit by these two 'reduction' rules.

It should be easy to see that the second of Prof. Dummett's aspects of use of a sentence which embodies the commitments made by asserting the sentence, is 'captured' by these equality rules. The condition under which $A \wedge B$ can be correctly asserted is fulfilled when there is a proof of A and a proof of B. This is captured by the introduction rule:

Σ-*introduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{B}(a)}{\mathtt{pair}(a,b) \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}$$

The conclusions one can draw from the sentence '$A \wedge B$', whose condition for correct assertion is expressed by its introduction rule, are expressed by the 'reductions' presented in the equality rules above.

Synthesising in a few words the point I am trying to make, if I have understood properly Prof. Prawitz's argument on what counts as the meaning of a logical connective (a sentence, more generally), all I can say is that the conditions under which it can be correctly asserted (introduction inferences) are indeed necessary conditions. However, these conditions do not seem to be always sufficient, and this seems to be present in the following lines:

4.*Does the condition for asserting A constitute the meaning of A?* The following two facts are relevant for this question:

(i) From the rules governing when a sentence is correctly asserted, one can derive the condition which a situation S has to satisfy, in order that a given inference is to be acceptable in S, viz., that a procedure is known in S which applied to canonical proofs of the premisses yield a canonical proof of the conclusion. This is the harmony required by Dummett to hold between the conditions for asserting a sentence and the commitments made by asserting it.

(ii) From an assumption that a person knows the condition for asserting a given sentence, it does not follow that he knows the acceptability of any particular inference involving those inferences that correspond to the conditions for asserting the sentences (introduction inferences); he knows the criterion for accepting an inference, but not necessarily whether the criterion is satisfied — in particular, he does not need to know what is usually considered to be the immediate commitments made by asserting the sentences in question.

(i) and (ii) show to what extent the use of language can be derived in a meaning theory which identifies the meaning of a sentence with the condition for asserting it.

(Ibid., pp. 38–9.)

This particular conclusion made by Prof. Prawitz when saying that the use of language can be derived in a meaning theory which identifies the meaning of a sentence with the condition for asserting it, has in fact led him to failure in his semantical studies. The schema which he devises to derive elimination rules from introduction rules (see Prawitz 1978) fails to fulfil its objectives. In other reports (de Queiroz 1988, de Queiroz & Maibaum 1987a), where I analyse the failure of his schema as well as an attempt to correct it made by P. Schroeder-Heister (Schroeder-Heister 1984), I have been able to recast Gentzen's claim that the elimination rule is a mere consequence of the definition of the logical connective. I have done this by showing that elimination rules of *Intuitionistic Type Theory*[6] are derivable from *introduction* rules plus *equality* rules (and some *induction* rules for the case of non-inductive types), the latter ones being the 'internalisation' of Prof. Prawitz's *reduction* rules. To this end I had to introduce some new rules of substitution which can be proved to be harmless with respect to the consistency and the whole philosophy of that particular formal system.

[6]I have used Prof. Martin-Löf's *Intuitionistic Type Theory* as described in Martin-Löf 1984.

As a last but not less important example to illustrate the point on the insufficiency of *introduction* rules and the rôle of *reduction* rules, it is worth mentioning another example of much relevance: the rule of *necessitation* of modal logics. It is usually framed as:

$$\frac{\mathbf{A}}{\Box\mathbf{A}}$$

which says that it is indeed an *introduction* rule for the modal operator '$\Box$', and yet the semantics of the operator is not given by that particular rule alone. The second aspect of the use of '$\Box$', which should say what is the *utility* of '$\Box$' (or, what kind of conclusions one can draw from $\Box\mathbf{A}$), is not provided, and so one does not yet know, just from the *introduction* rule above, what is meant by '$\Box$'. The fact is that the commitments made by asserting '$\Box\mathbf{A}$' are not specified by this particular rule, and so all of the aspects of the use (therefore meaning) of the operator '$\Box$' are not covered. What is usually done to fill in the 'semantical gap' is to provide a 'denotational' semantics for '$\Box$' in terms of possible worlds (Kripke models), and use axioms to complement the definition of what is meant by '$\Box$'. A proper proof theory (operational semantics, via a natural deduction presentation, with rules of introduction and elimination for each connective separately) for modal logics, however, does not seem to have been provided so far.

The general picture regarding the account of semantics given by Gentzen's tradition appears to be rather confusing given that *introduction* rules of natural deduction presentation systems *à la* Prawitz are said to provide meaning for logical constants, and occasionally an 'extra' condition presented in *informal* (verbal) explanation is said to complement the semantical explanation. How to formalise such an informal explanation is something which has not been explained properly, as far as I can see. My 'semantics of use' is an attempt to offer a reasonable account of how this 'formalisation of the informal semantical explanation' can be done. I have tried to suggest that this has in fact been done inadvertently in a few 'modern' mathematical languages.

The fact is that the semantical account based on *assertability conditions* does

not seem to offer itself as a very intuitively appealing and reasonably convincing alternative. If one looks at Modal Logics, the situation becomes even less easily understood because the recourse to notions beyond natural deduction proof rules goes even further away from the main motivation taken up by Gentzen himself, which is to provide an operational account of the semantics of the mathematical language: concepts essentially denotations-based such as Prof. S. Kripke's *possible worlds* are brought in together with some natural deduction proof rules to explain the semantics of modal connectives.

The example seems to be a good illustration of the point made about the insufficiency of the proof-theoretic *introduction* rules, because the need for a clarification of the commitments becomes clearer. For simpler cases like conjunction, disjunction, implication, etc., one is led to believe that the *introduction* rules provide enough information about the meaning of the connective. When it comes to more complex cases like '$\Box$', one easily realises that it is not the case, and one has to provide some extra information in order to say exactly what is meant by the connective whose *introduction* rules were just given. Most proof methods for modal logics available in the literature have to resort to something more than the natural deduction proof rules (*introduction* and *elimination* rules), which seems to go against the very nature of the main objective of natural deduction-like techniques of presenting logics, that is to have the particular logic presented in terms of proof rules only (*introduction*, *elimination*) for each connective separately, and no axioms.

To further illustrate the weaknesses of the *assertability conditions*-based semantics, and to what extent it can complicate matters as well as 'trivialise' certain complicated problems, I would like to refer to Prof. Prawitz's natural deduction-like presentation of the modal connective corresponding to the notion of 'necessity'.

First, a syntactic clause is added to include the 'new' connective with first-order logics described in the initial chapters of his detailed treatment of *Natural Deduction* (Prawitz 1965):

> To the *languages* of Chapter I, we now add a new logical constant, the *modal operator* N (read: it is necessary that). In the definition of the formulas, we add a fourth clause:

4) If A is a formula, so is NA.

The notion of principal sign is extended accordingly. A formula is said to be *modal* if its principal sign is N.

(Prawitz 1965, p. 74.)

The (proof-theoretic) 'semantical' definition of N follows, where rules of *introduction* and *elimination* for the new logical connective are presented as:

Using the terminology of Chapter I, we thus have an *improper inference rule* NI and a *proper inference rule* NE indicated by the figures:

$$\mathrm{N}I)\ \frac{A}{\mathrm{N}A} \qquad \mathrm{N}E)\ \frac{\mathrm{N}A}{A}$$

and a corresponding *deduction rule* for NI defined by the condition: An instance of the deduction rule for NI in S4 [S5] is to have the form $<< \Gamma, A >, < \Gamma, \mathrm{N}A >>$, where every formula of Γ is modal [modal or the negation of a modal formula]. (...)

(Prawitz 1965, p. 74.)

There are at least two points to be considered here. First of all, the rule of *introduction* for N is certainly of the same nature as the usual *necessitation* rule of any modal logic, which, as it is generally known, does not by itself make clear what is meant by 'necessary'. The semantics of N-like connectives is usually given by reference to the *interpretations*-based semantics of *possible worlds*. Each modal logic, be it S1, S4, S5, K1, K4, K5, etc., is usually characterised by a certain list of axioms which attempts to indicate what can be deduced from a proposition containing N as its major sign.

Secondly, by presenting the rule of N-*elimination* simply as:

$$\mathrm{N}E)\ \frac{\mathrm{N}A}{A}$$

Prof. Prawitz fails to realise the complexity involved in expressing the consequences of a modal proposition, which depends very much on the specific modal logic one is

working with. The meaning of N is in fact made clear when these consequences are shown, which is presumably why one usually does it via axioms.

Perhaps a reasonable example of the complexity of providing a proof-theoretic presentation of modal logics can be given by reference to Prof. M. Fitting's studies on proof techniques for these logics (Fitting 1981).

3.3.4. Unconvincing Proof-Theoretic Semantics

Proof Theory seems to have emerged from the very motivation of providing a more 'operational' account of meaning for the mathematical language. However, if one looks at the proof-theoretic account of meaning, as it was first established by G. Gentzen in 1935, it seems fair to say that the belief in the force of the *assertability conditions* with respect to the meaning of propositions does not make the 'anti-interpretation' criticisms sufficiently positive and convincing. In this respect, several attempts have been made to present a more convincing argument in favour of the semantics based on *assertability conditions*. In the paper 'Proof Theory and Meaning' (Sundholm 1986), e.g., Prof. Sundholm attempts to offer stronger arguments in favour of the usual semantical point of view that 'meaning is given by the *assertability conditions* (*introduction* rules)' by recalling Gentzen-Prawitz's claim that the *elimination* rules for a certain logical constant in a natural deduction presentation system would be mere consequences of the definition of the logical constant provided by its *introduction* rules. He then goes on to discuss Prof. D. Prawitz's unsuccessful attempt to derive *elimination* rules from *introduction* rules in natural deduction presentation systems. In Prof. Prawitz's project, as far as I can see, the fact that *introduction* and *elimination* rules of ordinary natural deduction presentation systems represent two *independent* aspects of the meaning of logical constants, seems to have been overlooked. Nevertheless, the failure is claimed to have been due to Prof. Prawitz's unsuitable schema, and a solution is said to have been given by P. Schroeder-Heister with his new schema based on his extended natural deduction calculus. The semantical point of view ('meaning as given by the *introduction* rules'),

however, is not said to have had anything to do with Prof. Prawitz's failure. As a matter of fact, Schroeder-Heister's 'solution', which stands on the same (Gentzen's) semantical principle, does not appear to provide a satisfactory account of such a general schema to produce *elimination* from *introduction* rules just because it does not produce the expected *elimination* rules, but *elimination* rules framed in a certain 'generalised' fashion which follows the classical *elimination* rules for 'disjunction'. (Schroeder-Heister's alternative to Prof. Prawitz's schema is discussed, e.g., in de Queiroz 1988 and in the section *3.2.2.* above.).

Perhaps due to its rejection of the truth-based classical account of meaning, by offering an essentially 'rule-based' semantics, as well as an explicit emphasis on certain *effectiveness* criteria for the notion of *truth* ('a proposition is true if it has a proof, that is, a proof of it can be given', Prawitz 1980), proof theoretic semantics is often qualified as being a 'verificationistic' semantics. There is, of course, no reason why one should not accept that proof-theoretic semantics is a verificationistic semantics, provided one understands *verification* conditions as being given by *assertability* conditions.

Nevertheless, it is certainly unreasonable to say that, just by advocating such 'verificationistic' principles, proof-theoretic semantics as given by Gentzen, Prawitz, and all those who believe in the semantical force of the *conditions for assertion*, is congenial with what seems to be the most intuitively appealing semantical principle ever advocated to present date: Ludwig Wittgenstein's *meaning-is-use* semantical paradigm. Despite having said that 'the sense of a proposition is the method of its verification', Wittgenstein's writings on verification and meaning stand right in opposition to the principle regulating most of these 'verificationistic' theories of meaning based on the notion of *proof*. It is not difficult to demonstrate that the *consequences* one is allowed to draw from an assertion play a central rôle in Wittgenstein's teachings on the connections between meaning and verification. According to a reasonable reading of Wittgenstein's early as well as late writings on the subject, one establishes the method of verification of a proposition by stating what *consequences* should be *verified* to be true.

Aiming at further clarifying the differences between my own understanding of Wittgenstein's 'the sense of a proposition is the method of its verification' and the principle saying that 'a proposition is defined by laying down what counts as a proof of the proposition' (Prawitz 1980) I would like to discuss the issue of verification and meaning in the light of my own reading of Wittgenstein's somewhat uniform (de Queiroz 1988e) view on the close links between the meaning of an assertion and the *conclusions* one can draw from it.

3.4. Verification and Meaning

Verificationistic theories of meaning often take the *assertability conditions* as the only aspect which determines the meaning of a proposition. It seems to have been Ludwig Wittgenstein who first insisted on the rôle of *verification* in the establishment of sound foundations of objective knowledge. Incidentally, he also referred to the *sense* of a proposition as being determined by the way it can be verified. However, Wittgenstein's point of view on the rôle of *verification* for the particular case of logical propositions seems to have been very different from the general 'dogma' of the so-called 'verificationistic theories of meaning', simply because it appears to prescribe that the process of *verification* should be inseparable from the *conclusions* one is allowed to draw from the proposition, or simply, the *utility* of making the assertion. And this seems to suggest the fundamental rôle of the *consequences* one can draw from a particular assertion as a key aspect of its meaning, in addition to the equally as important *assertability conditions* taken to be the sole aspect of the meaning of an assertion by most of the 'verificationistic theories of meaning'.

3.4.1. Wittgenstein and Verification

Wittgenstein's reference to the idea that the meaning of a proposition is determined by the way it can be verified seems to have been pursued throughout his own writings in a way which is very different from what is proclaimed by those who advocate verificationistic theories of meaning for the mathematical language based primarily

on *conditions for assertion*. The great Austrian thinker seems to have brought in a particularly original idea that notions like *meaning* and *understanding* can be much more appropriately discussed in a context where concepts such as *use*, *purpose*, *utility*, *what can be done with the thing referred to*, etc., are given primary importance. In a broader context, however, it does not appear to be a rigorously original idea[7]. Regarding formalised languages, which were of Wittgenstein's concern for some time, such concepts seem to relate much more closely to the notion of *logical consequence* than to the *assertability conditions*, more usually discussed in the context of theories of meaning. It is perfectly true, however, that, to a certain extent, it does not appear to be sufficiently clear from Wittgenstein's own words in some of his explicit references to *verification* with respect to *meaning*, what is actually meant by a method of *verification*. In a conversation with Friedrich Waismann of the Vienna Circle, Wittgenstein says:

(...) The sense of a proposition is the method of its verification. A method of verification is not the means of establishing the truth of a proposition; it is the very sense of a proposition. In order to understand a proposition, you need to know the method of its verification. To specify it is to specify the sense of a proposition. You cannot look for a method of verification. A proposition can only say what is established by the method of its verification. (...)

(*LW&VC*, p. 227.)

[7]Cf. Janik & Toulmin's comments on Mauthner's views on the rôle of a critique of language: '(...) Clearly, the public aspects of language can never be accounted for purely and simply in terms of an origin in private experience. However, if one considered language as part of a man's biological equipment and, so, as an instrument of survival—a view which Mach had made fashionable at that time—one can see a way clear of these difficulties. If the original purpose of words was to insure the survival of a group of men in a situation where a single individual could not survive, it made little difference whether or no the same sensory image was before different men's minds when a word was used, just so long as they all knew how to react, what was expected of them, when such a word was used. What really matters, what really has *meaning*, is not the image a word or a sentence conjures up, but the action that it suggests or commands, warns from or prohibits. (...)', pp. 127–8 of Janik & Toulmin 1973. (The emphasis is mine). Any critique of language appears to require observation of the nature of acts which correspond to utterances.

Indeed, the above paragraph appears to leave rather open the interpretation of what is meant by the way a proposition can be verified. It does not seem to be sufficiently clear whether the *verification* of a proposition is to be specified by simply saying under which conditions it can be correctly asserted, or by stating what conclusions which one is allowed to draw from it must be *verified* to be true.

Verificationistic theories of meaning, however, often insist on the idea that the meaning of a proposition is determined solely (primarily) from the conditions under which it can be correctly *uttered*. When attempting to relate to Wittgenstein's paradigm on the connections between meaning and use, suggesting that the two views could be taken as congenial, those who advocate verificationistic theories of meaning for the language of mathematics seem to leave clear (without stating it explicitly, because it would reject the alleged congenial position) that they assume a very different point of view from that which apparently prevails in most of Wittgenstein's own teachings on *meaning-is-use*. The argument of some 'verificationist' schools in favour of a certain close connection between *verification* and *meaning* is usually formulated on the basis that all one needs to know in order to grasp the meaning of an assertion is to know the conditions under which it can be correctly asserted. By taking the *assertability conditions* as being the sole, or at least the most fundamental aspect determining the meaning of a proposition, it is customarily said that the aspect concerning the *consequences* is to be taken as just some 'extra' information which is usually identified with the *use* of the sentence, whose features can be obtained from the first aspect, that of *conditions for utterance*. A sharp separation is then made between the *meaning* of a proposition and its *use*. This seems to leave quite clear the profound difference between this particular point of view and Wittgenstein's own position when he says that 'meaning is determined by use', and that the two concepts cannot be separated:

> For a *large* class of cases — though not for all — in which we employ the word "meaning" it can be defined thus: the meaning of a word is its use in the language.
>
> (*Investigations*, Part I, §43, p. 20e.)

In the semantical explanations provided by the verificationistic theories of meaning there seems to be a clear separation between the two notions. One has a notion of *meaning* of an assertion on the one hand, and the concept of *use* of such an assertion on the other hand. The idea seems to be that the *meaning* of a proposition is to be completely determined by the statement of its *assertability conditions*, and the *use* of the proposition, which is related to its *consequences*, is just an additional information, distinct from the *meaning*, whose features, it is claimed, can be 'uniformly derived' from the latter. Prof. D. Prawitz offers a clear and elegant explanation of the point of view on the separation between *the two aspects of correct use of a particular language*, advocated by the so-called 'verificationistic theories of meaning', by referring to a certain distinction between *sense* and *force*:

> Using Frege's distinction between *sense* and *force*, Dummett suggests that a theory of meaning is to contain two parts: one part that in one way or another specifies the meanings of the sentences of the language, and a second part, a theory of force, that describes the various uses that the sentences can be put to, e.g., that of expressing an assertion, a command or a question. The general idea is that if one knows the meanings of a sentence A and the conventions concerning different forces, then one is able to derive the significance of asserting A, commanding A, asking A, etc.. This derivation is to be uniform in the meaning of A; in other words, the only specific thing about A that one needs to know in order to use A in various ways is its meaning. (...)
>
> (Prawitz 1987, pp. 471–2.)

(The emphasis is mine.)

The position assumed regarding the distinction between *meaning* and *use* is made quite definite when the *use* of an assertion is given a rather secondary rôle in contrast to the primary rôle of the *meaning* (*conditions for assertion*), owing its features to the latter. Despite prescribing the aspect concerning the *use* of a sentence as one of the fundamental aspects of a theory of meaning, the secondary position occupied by such a notion of *use* becomes clearer when it is said that the features of the *use* of a particular sentence can be 'uniformly derived' from its *meaning*:

> (...) In summary, we may say that a theory of meaning is to describe all the features of the correct use of a language by combining, on the one hand, different forces or modes with, on the

other hand, conditions for utterances and consequences of utterances; furthermore, *it is to find one central aspect of a sentence, its meaning, from which all the mentioned features of the use of the sentence can be uniformly derived.* (...)

(Ibid., p. 472.)

(The *emphasis* is mine.)

It seems to be very clear that such a 'uniform derivation' of the features of the *use* of a sentence from its *meaning* (taken to be determined by its *conditions for assertion*) is unlikely to be very convincing. The fact is that one aspect is essentially distinct from the other, and both seem to play equally fundamental rôles in establishing the meaning of a proposition. It just sounds as a much less intuitively appealing invitation for a theory of meaning based on *conditions for assertion* alone, not to mention the failure of various attempts to 'uniformly derive' the rules which capture the *consequences* aspect of a particular logical proposition (e.g., natural deduction *elimination* rules) from the rules which formalise the *assertability conditions* aspect (e.g., natural deduction *introduction* rules).

Verificationistic theories of meaning often present their own variations on the main theme — meaning of a proposition as being determined by its *conditions for utterance* — according to certain relevant aspects observed by particular 'schools'. One of these aspects has to do with the degree of precision and exactness which is given to the formulation (i.e., the whole framework of ideas and concepts consisting of formal as well as informal explanations). It is argued, e.g., by some more recent formulations of verificationistic theories of meaning, that the explanations of meaning for the logical constants, provided by people like G. Gentzen, A. Heyting, A. N. Kolmogorov, etc., all lack a certain precision on what is to count as the meaning of a sentence with respect to its conditions for proof. A recent refinement of these previous explanations is obtained by a revision of the whole body of ideas concerning the theories of meaning based on somewhat formalised descriptions of what is to count as a (general) proof of a proposition, and the introduction of the notion of *direct* proofs.

3.4.2. A refinement: canonical (direct) proofs

A further refinement of the notion of meaning as being determined by the *conditions of utterance* is given by Prof. P. Martin-Löf when he elegantly points out the crucial and often neglected distinction between *propositions* and *judgements*, and introduces the notion of *canonical* (or *direct*) proof. By making an attempt to formalise the basic principles of what is called *constructive* mathematics, as practiced by E. Bishop and his followers, his explanations of meaning in terms of *canonical* proofs offers a new perspective on the nature of logic and mathematics. A whole body of fresh ideas providing new explanations of usual mathematical concepts, and a set of logical rules named *Intuitionistic Type Theory*, presented in proof-theoretic natural deduction style, emerge as the result of thorough philosophical studies into the nature of mathematical logic as the foundations of mathematics. In his 'On the meaning of logical constants and the justification of the logical laws' (Martin-Löf 1985), he presents philosophical explanations concerning the whole of more 'technical' work presented in Martin-Löf 1982, 1984, and other seminal papers (such as, e.g., Martin-Löf 1970, 1971ab, 1973, 1975ab).

Such an elegantly presented refinement of the *assertability conditions*-based semantics finds its justification as being a germane extension of the intuitionistic explanation of the meaning of logical signs which brings the previous explanations to a more precise presentation.

(...) The intuitionists explain the notion of proposition, not by saying that a proposition is the expression of its truth conditions, but rather by saying, in Heyting's words, that a proposition expresses an expectation or an intention, and you may ask, An expectation or an intention of what? The answer is that it is an expectation or an intention of a proof of that proposition. And Kolmogorov phrased essentially the same explanation by saying that a proposition expresses a problem or task (Ger. *Aufgabe*). Soon afterwards, there appeared yet another explanation, namely, the one given by Gentzen, who suggested that the introduction rules for the logical constants ought to be considered as so to say the definitions of the constants in question, that is, as what gives the constants in question their meaning. What I would like to make clear is that these four seemingly different explanations actually all amount to the same, that is, they are not only compatible with each other but they are just different ways of phrasing one and the same explanation. (...)

(Martin-Löf 1987, p. 410.)

(The emphasis is mine.)

As we shall see later on, the step made to further improve what appears to be a number of relevant and essentially congenial explanations of the meaning of logical signs, is to draw attention to a relevant distinction between what can be called *direct* (*canonical*) and *indirect* proofs. When trying to justify the introduction of the idea of *canonical* proofs, Prof. Martin-Löf refers to the existing similarities shared by all previous intuitionistic explanations of meaning, including those given by people like Heyting and Gentzen, and argues further in favour of an improvement on the way of explaining the meaning of a proposition by the statement of its *conditions for assertion*. Such an improvement would be motivated by a certain lack of precision remaining in all explanations mentioned above, which seem to fail in distinguishing *direct* from *indirect* proofs.

3.4.3. Wittgenstein's Early Intuition Prevails

In the later phase the ghost of uncertainty seemed to have gone away, and Wittgenstein appears more incisive on the idea that there are links between the *logical consequences* of an assertion and its meaning, besides admitting that a puzzling notion such as *meaning* could not be simply defined as something that could be either 'true' or 'false', as was strongly stated in the *Tractatus*. Attempting to offer an alternative account of propositions to the 'truth-value'-based one of the *Tractatus* by means of the new concepts introduced in the later phase, which finds in the *Investigations* its *chef-d'œuvre*, he acknowledges serious drawbacks in his early account of meaning and propositions. The criticisms of the idea that a proposition is whatever can be true or false become rather explicit at the point where he attempts to formulate an alternative to that picture of truth-values developed in the *Tractatus*, replacing it by a new 'definition' of what a proposition should be. Such new definition is formulated by means of a newly introduced notion of 'language-games', an idea that seems to

pervade the whole of *Investigations*, and certainly most of his later writings. His new 'definition' of what should be accounted for as a proposition is given as:

(...) Now it looks as if the definition — a proposition is whatever can be true or false — determined what a proposition was, by saying: what fits the concept 'true', or what the concept 'true' fits, is a proposition. So it is as if we had a concept of true and false, which we could use to determine what is and what is not a proposition. What *engages* with the concept of truth (as with a cogwheel), is a proposition.

But this is a bad picture. It is as if one were to say "The king in chess is *the* piece that one can check." But this can mean no more than that in our game of chess we only check the king. Just as the proposition that only a *proposition* can be true or false can say no more than that we only predicate "true" and "false" of what we call a proposition. And what a proposition is is in one sense determined by the rules of sentence formation (in English, for example), and in another sense by the use of the sign in the language-game. (...)

(*Investigations*, Part I, §136, pp. 52^e–3^e.)

(The emphasis is mine.)

It seems to be very clear that, according to such a (later) Wittgensteinian remark, a proposition cannot be characterised by just *one* aspect, be it a *grammatical* or a *use* aspect. The *two* features of an assertion are to be given equal importance, in order to fully characterise it, and this seems to make it quite different from the teachings of most of the so-called verificationistic theories of meaning. As for the criticisms of the idea that a proposition is determined by its truth conditions, however, there is certainly a point of agreement between the later Wittgenstein's account of propositions and the teachings on the notion of meaning offered by most of the 'verificationist' schools. The primary rôle played by *verification* replaces, in both cases, such an interpretation-based ('meaning of a proposition is what results from its interpretation in a particular model') account of meaning. What appears to constitute a definite and rather clear point of divergence is the very conception of the nature of *verification*. Ludwig Wittgenstein seems to have insisted on the idea that the process of *verification* of a proposition is to be specified by stating which *consequences* do indeed *follow* from it. The 'verificationists', on the other hand, assume a different position by saying that the method of *verification* of a

proposition is to be specified by the *conditions for it to be correctly asserted.* This appears to be quite clear from Prof. Prawitz's own words:

(...) In contrast to the case for truth conditions, one does not reckon with a situation where the condition for verifying a sentence (i.e., the existence of verification) is in fact satisfied while the fact that this is so is in principle impossible for us to know. Furthermore, *the possession of a verification of a sentence is the condition for correctly asserting the sentence* (i.e., for the assertion to be warranted), and this is one aspect of the use of the sentence, which has to stand in harmony with what was called above the consequences of asserting the sentence. The connections between what counts as a verification of a sentence and the use of a sentence thus seem quite straightforward. (...)

(Prawitz 1987, p. 474.)

(The *emphasis* is mine.)

Besides the basic principle saying that a verificationistic theory of meaning is meant to offer an *effective* account of the world-language links, the idea on the primacy of the conditions for correctly asserting the sentence over other aspects of the use of the language seems to constitute the essence of a certain set of basic principles by which one has to abide when putting forward a theory of meaning which can be considered as 'verificationistic'. Variations on the formulation of this main idea have emerged and the notion of *direct* proofs appeared as a key notion to motivate further improvements on the main theme. Arguing for a certain lack of precision on the (essentially germane) formulations of semantical explanations via *conditions for utterance* given by Gentzen, Kolmogorov, etc., Prof. Martin-Löf presents his own explanations for the refinement of such equivalent semantical explanations by insisting on the need for a more precise account of what should be understood by the connections between *assertability conditions* and *meaning.* And his informal notion, with corresponding formal counterpart, of *canonical* or *direct* proofs is said to constitute a useful refinement of those previous semantical explanations:

Everything that I have said so far was said, or could have been said, already in the thirties. What made these explanations as given in the thirties lack precision somewhat, was the absence

of a clear distinction between what Dummett has called a canonical proof and a demonstration of a proposition, and which I shall call a direct as opposed to an indirect proof of a proposition. In explaining the meanings of the logical constants, I did not care whether a proof of a proposition was direct or indirect: nevertheless, we have to make that distinction, because what is explained in the meaning explanations of the logical constants, as I just gave them, is what constitutes a direct proof of a proposition formed by means of one of those constants. *Thus a proposition is defined by what counts as a direct proof of it.* (...)

(Martin-Löf 1987, pp. 412–3.)

(The *emphasis* is mine.)

It is clearly a refined explanation of meaning by *conditions for assertion*, and the reasonably wide acceptance of its formal counterpart, *Intuitionistic Type Theory*, leaves little doubt here. As we shall see later on, by remaining congenial with the usual Gentzen-like explanations of meaning, Prof. Martin-Löf seems to assume essentially the same position as to the secondary rôle of the *consequences* aspect (the *use* aspect) of a proposition. What counts as a *direct* proof of a proposition is all one needs to know in order to grasp its meaning, no attention being primarily paid to the aspect of how such *direct* proof is to be used in the proofs of other propositions. This shall become clearer from the discussion that follows below, and it will certainly be crucial in making Prof. Martin-Löf's explanation profoundly different from Wittgenstein's. The point of view taken by the Austrian philosopher shows a clear emphasis on the aspects of *use*, *consequences*, *utility*, etc..

As a matter of fact, a more careful reading of Wittgenstein's teachings on the concepts of *meaning* and *understanding* leads to the discovery of rather explicit references to the rôle of the *consequences* one can draw from an assertion with respect to its *meaning*. Not just with respect to the mathematical language (as is discussed in de Queiroz 1988e), but the remarks on the connection between the meaning of an assertion with its *utility*, *consequence*, etc., often appear in contexts where ordinary language, rather than logic, is being discussed. In a series of expositions of his own views on matters related to religious belief Wittgenstein finds convenient grounds

to speak about the connections between the *meaning* of a particular statement and the *consequences* one can draw from it:

> If you say to me — "Do you cease to exist?" — I should be bewildered, and would not know what exactly this is to mean. "If you don't cease to exist, you will suffer after death", there I begin to attach ideas, perhaps ethical ideas of responsibility. The point is, that although these are well-known words, and although I can go from one sentence to another sentence, or to pictures *[I don't know what consequences you draw from this statement]*.
>
> (*Lectures and Conversations on APRB*, pp. 69–70.)

(The *emphasis* is mine.)

Later on, when involved in (apparently) the same discussion again on issues concerning religious belief, the same intuition seems to spring up very clearly, this time with explicit reference to the idea that his own conception of *verification* of an assertion cannot be separated from the *consequences* that can be drawn from it:

> When I say he's using a picture I'm merely making a ***grammatical*** remark: [What I say] can only be verified by the consequences he does or does not draw.
>
> (*Lectures and Conversations on APRB*, p. 72.)

(The emphasis is mine.)

In his later phase, references of a somewhat explicit character to the connections between notions such as *meaning*, *understanding*, etc., and concepts like *use*, *consequences*, *conclusions*, etc., seem to be an attempt to convince by insisting on a certain unusual point of view, through the explanation of examples, recourse to situations of everyday life, etc.. For example, the rôle of *utility* of asserting a particular sentence in the understanding of what is meant by it, is referred rather explicitly in a typescript of his later phase (TS no. 229, whose underlying manuscripts cover the time from May 1946 to October 1947[8]), later on edited as the first volume of the *Remarks on the Philosophy of Psychology*. It reads precisely as:

[8] *RPP I*, Editors' preface.

We read in a story that someone had a dream and did not tell it to anyone. We don't ask how the author could learn it. – Don't we *understand* it, when Strachey makes surmises about what Queen Victoria may have seen in her mind's eye just before her death? Of course – but didn't people also *understand* the question how many souls there was room for on the point of a needle? That is to say: the question whether one understands this does not help us here; we must ask *what* we can do with such a sentence. – *That* we use the sentence is clear; *how* we use it is the question.

(*RPP I*, §366, p. 72e.)

(The emphasis is mine.)

Here the link he makes between the understanding of a sentence and its *utility*, or, in other words, what can be done with the sentence, appears to be put rather clearly. An important aspect of the *use* of sentences is highlighted, which is the aspect of *how* one can use it, in addition to the aspect which indicates *when* the sentence can be used, the latter being an aspect of essentially grammatical nature, such as the conditions for asserting the sentence.

And this is just one of many remarks collected in the same typescript, where the indicated connections are suggested. Again, later in the same *Remarks on the Philosophy of Psychology*, vol. I, a reference to the rôle of conclusions, confirmations, answers, etc., in the understanding of a particular statement is given explicitly as:

Doesn't the theme point to something outside itself? Oh, yes! But that means: – The impression that it makes on me hangs together with things in its surroundings – e.g. with the existence of our language and its intonation; but that means: with the whole field of our language-games.

When I say, e.g.: It is as if a conclusion were being drawn here, or as if here something were being confirmed, or, as if *this* were the answer to what went before, – in this way my understanding presupposes familiarity with conclusions, confirmations, answers.

(*RPP I*, §433, p. 85e.)

(The emphasis is mine.)

An important explicit reference to the meaning of statements as something inseparable from its *consequences* is also found in another typescript of his later phase, a typescript (TS no. 232, whose remarks underlying its dictation stem from the

period between November 1947 and August 1948[9]) which has been published as the second volume of the same *Remarks on the Philosophy of Psychology.* It states (what appears to be) a key connection in Wittgenstein's own critique of language between the meaning of a statement and the consequences one is allowed to draw from it. It reads exactly as:

Denn nicht das ist die Frage: was ich mache, wenn ... (dies könnte nur eine psychologische Frage sein) — sondern, welche Bedeutung die Äußerung hat, was sich aus ihr entnehmen läßt, welche Folgen sie hat.

which has been translated as

For the question is not 'What am I doing when ... ?' (for this could only be a psychological question) — but rather, 'What meaning does the statement have, what can be deduced from it, *what consequences does it have*?'

(*RPP II*, §38, p. 8^e.)

(The *emphasis* is the Translators'.)

It does not present much difficulty to find a number of other remarks in this same typescript, where rather clear suggestions are made on the close links between the meaning or the understanding of a sentence and its consequences.

One can also find very clear traces of such a key idea pursued by (the later) Wittgenstein, in the writings of more recent investigators on the rôle of language in the mathematical activity, who seem to have been familiar with the philosopher's main works. A number of highly relevant attempts to reconcile the intuitionistic conception of mathematics, often loaded with a kind of solipsistic flavour to various degrees, and the 'public' aspect of scientific knowledge, have run into what appears to be the same sort of view on the rôle of the *consequences* of an assertion with respect to its *meaning.*

Prof. M. Dummett, in his theory of meaning and truth, attempts to provide a convincing argument to support his 'anti-realist' criticisms by offering an account

[9] *RPP II*, Editors' preface.

of meaning based on the *commitments* one makes when asserting a proposition. A significant part of Prof. Dummett's explanations of meaning recall the aspect of *commitments* as a key aspect in a theory of meaning. And to recall the aspect of *commitments* one makes by asserting a sentence as a fundamental aspect of its meaning seems to be essentially the same as to recall the *consequences* one can draw from the sentence. One can hardly distinguish the *commitments* aspect from the so-called *consequences* aspect. It seems reasonable to suppose that the *commitments* one is prepared to assume by making an assertion correspond to the *consequences* of the assertion that he/she is prepared to *verify*.

It would seem to have been a strong alternative as an anti-realist theory of meaning had he taken seriously this aspect which apparently received such a denomination of *commitment* from Prof. Dummett himself. Unfortunately, by putting forward his own reading of Wittgenstein as another conceptual support for the *assertability conditions* alternative to realist semantics, he seems to leave a gap as to what place such *commitments* would take in an account of meaning based on the conditions for assertion alone. The emphasis on the *commitments* aspect is made clear when he says that his theory of meaning *links the content of an assertion with the commitment that a speaker undertakes in making that assertion*[10].

For some reason, however, Prof. Dummett's own reading of Wittgenstein seems to remain essentially in line with the *assertability conditions*-based anti-realist alternative by saying that the meaning of logical constants is determined by the *conditions for assertion*:

> (...) [The Meanings of the Logical Constants] The meaning of each constant is to be given by specifying, for any sentence in which that constant is the main operator, what is to count as a proof of that sentence, it being assumed that we already know what is to count as a proof of any of the constituents. (...)
>
> (Dummett 1977, p. 12.)

His joint research programme with Prof. Prawitz on a theory of meaning for the language of (intuitionistic) mathematics, ends up 'neglecting' the *commitment*

[10] p. 126 of Dummett 1976.

aspect in favour of a certain priority for the *conditions for assertion* with respect to the meaning of a logical proposition.

Some of Prof. S. Kripke's comments on Wittgenstein's 'private language argument' (Kripke 1982) appear to shed some light on the useful explicit connections between *meaning* and *utility*, *purpose*, etc., which seem to have been taught by (the later) Wittgenstein himself as being a key aspect of the understanding of the notion of meaning. It seems to have been an important ingredient of Wittgenstein's development of what appeared to be already present in his own intuitions on the nature of *meaning*, particularly those concerning the nature of logical propositions, but which only found firmer grounds later in the *Investigations*, where, paradoxically enough, he speaks mostly about ordinary language, and almost nothing or very little about logic or any formalised language. Prof. Kripke's remarks refer to Prof. Dummett's use of Wittgenstein's *Investigations* as an example of a source of more convincing semantical explanations for the mathematical language than the explanations provided by realist semantics, and read as follows:

> (...) So stated, the *Tractatus* picture of the meaning of declarative sentences may seem not only natural but even tautological. Nonetheless, as Dummett says, "the *Investigations* contains implicitly a rejection of the classical (realist) Frege-*Tractatus* view that the general form of explanation of meaning is a statement of the truth conditions". In the place of this view, Wittgenstein proposes an alternative rough general picture. (To call it an alternative *theory* probably goes too far. Wittgenstein disclaims (§65) any intent of offering a general account of language to rival that of the *Tractatus*. Rather, we have different activities related to each other in various ways.) Wittgenstein replaces the question, "What must be the case for this sentence to be true?" by two others: first, "Under what conditions may this form of words be appropriately asserted (or denied)?"; second, given an answer to the first question, "What is the rôle, and the utility, in our lives of our practice of asserting (or denying) the form of words under these conditions?" (...)
>
> (Kripke 1982, pp. 72–3.)

(The emphasis is mine.)

Here, besides the emphasis on Wittgenstein's replacement of one question by *two* other questions, the crucial point seems to be the reference to the word *utility*,

given that a closer look at the aspects of *consequences* of a proposition, on which we have been insisting as a key aspect in the explanation of the meaning of a particular assertion, appears to give the argument a remarkably convincing force, where a proposition is understood not only from its *conditions for utterance*, but also from its *utility*, or *what can be done with it*, what its *consequences* are. A remark which is directly related to this particular issue and which certainly appeals to the intuition because it refers to the need to understand the concept of meaning of a word as being placed in a distinct position from the word itself, and attempts to suggest the connections between *meaning* and *utility*, is a remark contained in the first half of the Part I of the *Investigations*. It reads as follows:

> You say: the point isn't the word, but its meaning, and you think of the meaning as a thing of the same kind as the word, though also different from the word. Here the word, there the meaning. The money, and the cow that you can buy with it. (But contrast: money, and its use.)
>
> (*Investigations*, Part I, §120, p. 49e.)

Going through the philosophical remarks of Part I of the *Investigations* one does not take long to find another quite intuitively appealing remark where he attempts to insist on the idea of relating the meaning of a word and the understanding of it to its corresponding *utility*, to *what can be done with it*. If one does not know the *utility* of asserting a particular sentence, ignoring what can be done with it, one certainly has not understood its meaning. That seems to be what Wittgenstein insisted on suggesting. E.g., in the remark below:

> Let us remember that there are certain criteria in a man's behaviour for the fact that he does not understand a word: that it means nothing to him, that he can do nothing with it.
>
> (*Investigations*, Part I, §269, p. 94e.)

Yet another attempt to find a more 'public' account of the activity of doing mathematics, *Game-Theoretic Semantics* (*GTS*) can be seen as an attempt to put into more effective terms Wittgenstein's suggestion regarding the notion of 'language-games' as the link between language and reality. It has been worked out for a certain time by Prof. J. Hintikka and his followers as a whole body of concepts

whose formulation has been based on their own reading of Wittgenstein's (later) account of propositions and the nature of language. The basic ideas of such a body of formal as well as informal logical notions were essentially introduced by reference to formal first-order quantificational languages. A language L is assumed to be an interpreted language, allegedly due to the possibility of talking about truth and falsity of its sentences, and its being interpreted is explained by saying that one is given some domain of individuals D on which all the nonlogical constants of L plus the names of the members of D would have a definite truth-value. The departure from Tarskian truth-based theory of meaning is made by explaining the semantics of each logical constant by means of what is called *semantical games*. In the description of the *Game-Theoretical Semantics* (*GTS*) *semantical games* are defined by Prof. Hintikka[11], where there is supposed to be two 'players' called *Myself* and *Nature*, as:

Atomic formulas:

($G.\ A$) If A is atomic, then I have won $G(A)$ and Nature has lost if A is true. If A is false, Nature has won and I have lost.

Conjunction:

($G.\ \&$) $G(S_1\ \&\ S_2)$ begins by Nature's choice of S_1 or S_2. The rest of the game is $G(S_1)$ or $G(S_2)$, respectively.

Disjunction:

($G.\ \vee$) $G(S_1 \vee S_2)$ begins by Myself's choice of S_1 or S_2. The rest of the game is $G(S_1)$ or $G(S_2)$, respectively.

[11] p. 3 of Hintikka 1983.

Universal Quantification:

(*G. U*) $G((x)S[x])$ begins by Nature's choice of a member of D. Let the name of the member chosen be 'a'. The rest of the game is then $G(S[a])$.

Existential Quantification:

(*G. E*) $G((Ex)S[x])$ is defined likewise except that 'a' is chosen by Myself.

The truth of a sentence S of L would then be defined as the existence of a winning strategy in $G(S)$ for ***Myself***. Similarly, the falsity of S would be given by the fact that the other 'player', ***Nature***, has a winning strategy.

Concerning such a notion of 'winning strategy', it might perhaps be worthwhile to recall Prof. Hintikka's own explanation on the connections between his 'reinterpretation' of the idea of ***verification*** and such a concept of winning strategy, borrowed from the mathematical theory of games to justify what he calls *strategic meaning*:

(...) Very briefly, the semantics of a sentence S is in GTS discussed by reference to a two-person zero-sum game $G(S)$ associated with it. This game $G(S)$ is played on a model or "world" in which the primitive symbols of S have been interpreted. The players are called *Myself* (I would now perhaps prefer the designation "The Competent Speaker"), who is trying to verify S, and *Nature*, who is trying to falsify S. The truth of S is defined as the existence of a winning strategy (in the game-theoretical sense of the expression) for Myself in $G(S)$; correspondingly for falsity. This definition captures the true gist of verificationist theories of meaning, in the sense that according to it S is true if it can be verified. But the idea that S "can be verified" is in GTS reinterpreted as the existence of a verifying (winning) strategy for Myself in $G(S)$. In sum, what a sentence S says is to assert the existence of a winning strategy in $G(S)$.

(Hintikka 1987, p. 499.)

A closer look at the formulation of the meaning of logical constants provided by Prof. Hintikka in his own reading of Wittgenstein's notion of 'language-games', the so-called *Game-Theoretic Semantics*, leads to the simple conclusion that the *semantical games* are essentially showing how *Nature* plays by asking *Myself* to comply with the commitments *I* make when asserting a proposition. In other words, the *consequences* aspect of a particular assertion is the relevant aspect which gives meaning to it, because that is what essentially the *semantical games* seem to be showing: what are the consequences of a proposition that have to be verified to be true. As Prof. Hintikka appears to say clearly in the paragraph above, the notion of *verification* is 'reinterpreted' as the existence of a 'winning strategy', where for each single move of the *game* an *eliminatory* operation (corresponding to the *consequences* aspect of the proposition asserted) is performed on the major connective of the proposition being 'played'. (For example, in the rule of the game for the conjunctive constant '$\wedge$', the move is made by *eliminating* the conjunction in favour of either one of the conjuncts.) The whole process follows a general pattern where *Nature* demands *Myself* to verify that the *consequences* of the assertions which *I* (*Myself*) lay down are true.

4 *Wittgenstein and the Paradigm 'Meaning-is-Use'*

The type-theoretic semantics whose development is reported in this dissertation results from a particular reading of Ludwig Wittgenstein's writings on the notions of meaning, use, proposition, understanding, etc., where *normalisation* rules, formal explanations of *consequences*, are taken to be semantical rules. I shall attempt to demonstrate in this chapter that it is not unreasonable to assume that Wittgenstein's teachings on the concepts of meaning and proposition convey a rather clear uniformity regardless of phases, which can be put simply as 'the meaning of a proposition is inseparable from the consequences one can draw from it'. For that I shall firstly discuss the connections between the concepts of 'rule' and 'meaning'. Secondly, I will attempt to fulfil the claim concerning the relative uniformity in Wittgenstein's writings on the connections between the meaning of a proposition and its consequences. Thirdly, I shall demonstrate that the view on such connections can be reconciled with the paradigm characterising the later phase – *meaning-is-use* – by demonstrating that further to the connections between *meaning* and *use*, there are explicit connections between *meaning* and *consequences* as well as between *use* and *consequences*. The resulting 'picture' is a triangle-like diagram with vertices on meaning–use–consequences which highlights the rôle of the *consequences* one can

draw from a proposition as a fundamental ingredient on Wittgenstein's development of his more celebrated semantical paradigm *meaning-is-use*. By collecting many relevant remarks from later writings, I intend to help 'filling in' the diagram aforementioned, by showing that the two usually less emphasised sides of the triangle, namely that from *meaning* to *consequences* and the other one from *use* to *consequences*, indeed have their part in Wittgenstein's semantical teachings. This, I claim, is what makes Wittgenstein's 'meaning-is-use' semantical paradigm extremely useful in dealing with formalised languages. By contrast with the general idea that if meaning is taken to be the use, one needs to take into consideration complex aspects involving the context of utterances. For the case where the consequences allowed to be drawn from assertions are formally stated, such as is the case of formalised languages (cf. semi-formal presentation of logic by the *semantical games* of *Game-Theoretical Semantics*) one does not have to put off by the complexity of dealing with context considerations because the meaning of propositions is already stated by rules demonstrating what consequences can be drawn from them, no matter the context they might be embedded in.

Finally, I shall demonstrate how one can reconcile Wittgenstein's teachings on 'meaning as explanation of meaning', which opens the *Blue Book*, as well as on 'meaning is no object' which appears in the opening lines of the *Investigations* and in the same *Blue Book*, with such a uniform account of *meaning-is-use* based on *consequences*. As to the former – 'meaning as explanation of meaning' –, the reconciliation is made by saying that the meaning of a proposition is made clear by an explanation of the consequences one is allowed to draw from it. Concerning the latter – 'meaning is no object' –, I refer to the type-theoretic account of meaning described in this dissertation in connection with Wittgenstein's connection between meaning and function, purpose, utility: the meaning of a proposition/type is not given by its proofs/elements, but by its *utility* expressed in the consequences rules of normalisation (cf. **Π**-types and β-rule of **Π**-*reduction*.).

By references to F. Mauthner's contributions to the critique of language and its links with Wittgenstein's philosophy I attempt to pay a tribute to the critique of

language which appears to have been the first to warn that 'words do not correspond to objects'.

In order to illustrate the discussion and make the chapter reasonably self-contained I shall be making use of a fairly large number of quotations. For the sake of clarity, I have found appropriate to make a few of those quotations appear more than once.

4.1. Introduction

Within the mathematical language if one looks at meaning as something determined by use rather than by an interpretation procedure, such as it was suggested by Wittgenstein':

> Perhaps we have to say that the phrase 'interpretation of signs' is misleading and instead we ought to say 'the use of signs'.
>
> (*Philosophical Remarks*, §32, p. 70.)

then to provide a precise account of (operational) meaning which can comply with this particular Wittgensteinian doctrine known as 'meaning is use', assuming that the concept of 'use' is understood in direct connection with the concept of 'utility' and is dissociated from the too broad sense of 'usage', it seems plausible to say that one needs to view meaning as something more than just content, an object, a (canonical, numerical) value, a mathematical construction, etc.. In these terms, the meaning of a proposition is not a truth-value, the meaning of a set cannot be given by what it contains — the extensional view, such as is the case for the classical explanation of how a set is to be understood —, or even by a rule showing how to build elements of the set — such as is the case for the operational account of meaning for the language of mathematics given by Prof. P. Martin-Löf in his *Intuitionistic Type Theory*[1] —. A formal counterpart to any operational account of meaning

[1]There seems to be a deliberate intention of avoiding interpretations as semantical explanations in Prof. Martin-Löf's own words:

'(...) In standard textbook presentations of first order logic, we can distinguish three quite separate

seems to have more convincing power when it captures in some way the intentional ('utility') aspect of logical signs. The operational semantics of the 'λ' logical sign of the lambda calculus, for example, is given by at least the β-rule of reduction, where the 'utility' character comes from the nature of *reduction*, a meta-mathematical device which serves to explain the effect of *elimination* inferences on the result of *introduction* steps, demonstrating the utility of 'λ' in terms of formalising the concepts of generality and substitution.

As a matter of fact, *use* is not just a concept arbitrarily chosen by Wittgenstein as the basis for his semantical investigations, for it entails the crucial concept of *intentionality*. And, as a matter of course, there is a great deal of *intentional* ingredients in Wittgenstein's concept of meaning as use. That is what makes his semantic theory fundamentally different from many of the previous accounts of the relation between language and the world, which were mostly extensional following St. Augustine's tradition 'as I heard words repeatedly used in their proper places in various sentences, I gradually learnt to understand what objects they signified'. In such an *intentional* view of language, there is no place for interpretations as endpoints for settling meaning. Moreover, in the light of such an alternative account of the links between language and reality, one can say, e.g., that the phrase 'the interpretation of signs' is misleading, and that one ought to say 'the use of signs' instead, because the former suggests that a sign is to be understood by perhaps another sign which does not need any further interpretation, whereas the latter

---

steps:

(1) inductive definitions of terms and formulas,

(2) specification of axioms and rules of inference,

(3) semantical interpretation.

Formulas and deductions are given meaning only through semantics, which is usually done following Tarski and assuming set theory.

What we do here is meant to be closer to ordinary mathematical practice. We will avoid keeping form and meaning (content) apart. Instead we will at the same time display certain forms of judgement and inference that are used in mathematical proofs and explain them semantically.'

(Martin-Löf 1984, p. 3.)

conveys a much broader conceptual basis which can accommodate intentionality.

In the *Investigations* Wittgenstein speaks of 'interpretations hanging in the air along with what they interpret':

> "But how can a rule shew me what I have to do at *this* point? Whatever I do is, on some interpretation in accord with the rule."—That is not what we ought to say, but rather: any interpretation still hangs in the air along with what it interprets, and cannot give it any support. Interpretations by themselves do not determine meaning.
>
> (*Investigations*, Part I, §198, p. 80e.)

The pictorial device of 'hanging in the air' is sometimes replaced by the figure of a 'rider' which stands there to help giving the sentence its meaning. This is the particular 'clothing' of the shortcomings of interpretations as semantical devices that appears in *Zettel*, a collection of fragments made by Wittgenstein himself and left by him in a box-file which are hardly datable although the editors believe that 'the earliest time of composition of any of these fragments was 1929 and by far the greatest number came from typescripts dictated from 1945–48'[2]:

> But an *interpretation* is something that is given in signs. It is this interpretation as opposed to a different one (running differently).—So when we wanted to say "Any sentence still stands in need of an interpretation", that meant: no sentence can be understood without a rider.
>
> (*Zettel*, §229, p. 40.)

The 'rider' of a sentence can only disappear when the element of *intentionality* is acknowledged, and the intention is uncovered by a demonstration of the conclusions one can draw from the sentence. The intentional aspect of a sentence is not made clear until one makes explicit where one intends to arrive at by asserting it. Compare, e.g., 'what do you mean?' with 'what are you trying to say?', 'what conclusion(s) do you want one to draw from what you have just said?', 'what is your intention when asserting the sentence?', etc.. And it is 'intention' which is going to play a crucial rôle in breaking the apparently infinite chain of interpretations. For Wittgenstein it is 'intention' which is going to give the 'final' interpretation. And indeed, in another

[2] *Zettel*, Editors' Preface, p. iii.

remark from the same *Zettel*, he expresses the view that *intention* can be a useful concept in the breaking of such a chain of interpretations, but does not sound too satisfied when saying that such a concept is of a 'psychological' rather than 'logical' nature:

> By "intention" I mean here what uses a sign in a thought. The intention seems to interpret, to give the final interpretation; which is not a further sign or picture, but something else—the thing that cannot be further interpreted. But what we have reached is a psychological, not a logical terminus.
>
> (*Zettel*, §231, p. 41.)

(*En passant*, it is perhaps relevant to remark here that those two passages above from *Zettel* also appear in *Philosophical Grammar* (Part I, p. 47 and p. 145, respectively) which was edited from 'a typescript completed probably in 1933, perhaps some of it 1932' (*Note in Editing*, *Philosophical Grammar*, p. 487).)

In attempting to extract from Wittgenstein's early as well as late writings a certain uniformity with respect to the connections between the meaning of a proposition and the consequences one is allowed to draw from it, it is also my endeavour to demonstrate that a logical counterpart to such a 'psychological' terminus, namely *intention*, is already present in his semantic theory, in spite of an apparent lack of explicit and definite acknowledgement that this is the case by Wittgenstein himself.

4.2. Wittgenstein's Operational View of Meaning

A particular feature of Wittgensteinian thought seems to prevail in most of his writings, be it either from the early or the later Wittgenstein. Although gaining its major force only in the later writings, the idea that meaning (especially of logical symbols) could be made explicit via *operations* (as opposed to *denotations* or *interpretations*) seems to be an underlying preoccupation of Wittgenstein. Despite the strong truth-values based logicist influence from Frege and Russell and the clear Frege-realist position taken in the *Tractatus* when saying that 'a proposition is a

truth-function of elementary propositions', one can see signs of what is now recognised as an outstandingly original idea, even in his very early writings such as the *Notebooks 1914–16*: analysis of meaning by analysis of use.

4.2.1. Against a 'static' account of representing meaning

Since his early works, such as the *Notebooks 1914–16*, Wittgenstein already demonstrated his determination in investigating how the meaning of logical symbols could be made explicit without the need of any kind of *interpretation*. In the course of his philosophical analysis of subjects such as language, meaning and understanding one of the conclusions he seems to arrive at is the weakness of pure 'coding' and the need for a more 'dynamic' (logical) account of meaning.

> If we were to try and express what we express by means of "(x).ϕ(x)" by prefixing an index to "ϕ(x)", e.g., like this: "Gen.ϕ(x)", it would not be adequate (we should not know what was being generalised).
>
> If we tried to shew[3] it by means of an index to the "x", e.g., like this: $\phi(x_G)$, it would still not be adequate (in this way we should not know the scope of the generality).
>
> If we thought of trying to do it by inserting a mark in the empty argument places, e.g., like this: "(G, G).ϕ(G, G)" it would not be adequate (we could not settle the identity of the variables).
>
> All these methods of symbolising are inadequate *because they do not have the necessary logical properties*. All those collections of signs lack the power to portray the requisite sense - in the proposed way.
>
> (*Notebooks*, p. 18[e], dated 25.10.14.)

(The *emphasis* is mine.)

Although not making clear enough what kind of logical properties would be needed for a method of symbolising to be adequate, the recognition of the need of such logical properties sets up grounds for the investigations of what form should these properties take. Later in the same *Notebooks*, a form of logical inference is suggested:

---

[3]'shew', archaic (British) form of 'show' is used in most of Wittgenstein's writings, be it a translation or an original English text.

That the *proposition* "ϕa" can be inferred from the *proposition* "(x).ϕ(x)" shews how generality is present even in the sign "(x).ϕ(x)".

And the same thing, of course, holds for any generality notation.

(Ibid., p. 32^e, dated 24.11.14.)

(aside remark: the *Tractatus* was first published in 1921.)

It seems clear from what he says about 'showing how generality is present even in the sign "(x).ϕ(x)" ' that it is the process of inference which is making the meaning of the particular sign ('generality', in this case) explicit. One can easily make a parallel between the fact that by saying that ϕa follows from (x).ϕ(x), the meaning of (x).ϕ(x) (generality) is made explicit, with the rôle of the β-rule of the lambda calculus which shows the (generality) meaning of 'λ'.

We shall see later on that by saying that a logical (*elimination*) inference is showing the meaning of the proposition he is still in accordance with the idea that 'meaning is determined by use'. That particular inference is in fact expressing one of the aspects of the use of the proposition, which indicates the kind(s) of conclusion one can draw from the proposition, or even, the *utility* of asserting it.

The underlying position Wittgenstein seems to assume quite clearly is that the meaning of logical connectives are to be expressed by logical inference rules, and yet his early philosophy set by the *Tractatus* takes a different line staying within the logicist view of his masters Frege and Russell. In later writings (the *Philosophical Remarks* can probably be taken as the beginning of what is now called the 'later Wittgenstein philosophy') he acknowledges drawbacks in the classical (realist) view expressed by the *Tractatus* and comes back to what seems to have been his original position.

Even in the *Tractatus* itself, despite the classical (Frege-realist) way of giving meaning to logical symbols via truth-tables, there is clear evidence of what appears to characterise the sole Wittgensteinian semantical intuition: meaning is to be determined by use.

Not just the 'how a sign is used is what makes us recognise it' as in:

In order to recognize a symbol by its sign we must observe how it is used with a sense.

(*Tractatus*, §3.326, p. 16.)

or the straight connection, although expressed negatively, between use and meaning as in:

If a sign is *useless*, it is meaningless. That is the point of Occam's maxim.

(*Tractatus*, §3.328, p. 16.)

but the rôle of the *utility* aspect of propositions implicit in:

(In philosophy the question, 'What do we actually use this word or this proposition for?' repeatedly leads to valuable insights.)

(*Tractatus*, §6.211, p. 65.)

plays an important part in the development of Wittgenstein's later account of meaning and propositions. And indeed, Prof. Kripke's comments on Wittgenstein's 'private language argument' (Kripke 1982) contain useful remarks on how this is to be found in the outcome of Wittgenstein's development of what seemed to be already part of his philosophical views since very early but only found firmer grounds later in the *Investigations*:

(...) So stated, the *Tractatus* picture of the meaning of declarative sentences may seem not only natural but even tautological. Nonetheless, as Dummett says, "the *Investigations* contains implicitly a rejection of the classical (realist) Frege-*Tractatus* view that the general form of explanation of meaning is a statement of the truth conditions". In the place of this view, Wittgenstein proposes an alternative rough general picture. (To call it an alternative *theory* probably goes too far. Wittgenstein disclaims (§65) any intent of offering a general account of language to rival that of the *Tractatus*. Rather, we have different activities related to each other in various ways.) Wittgenstein replaces the question, "What must be the case for this sentence to be true?" by two others: first, "Under what conditions may this form of words be appropriately asserted (or denied)?"; second, given an answer to the first question, "What is the rôle, and the utility, in our lives of our practice of asserting (or denying) the form of words under these conditions?" (...)

(Kripke 1982, pp. 72–3.)

As we have seen in the previous chapter, it is on this basis (the *two* questions) that Prof. Dummett and Prof. Prawitz attempt to analyse further the notions of meaning and use within the mathematical language.

That rules will play an important rôle in his development of firmer grounds for this 'later' philosophy is clear from much of his later writings. In Part VII of his *Remarks on the Foundations of Mathematics*, which is a collection of remarks from manuscripts written in 1941 and 1944 he says:

> We can conceive the rules of inference—I want to say—as giving the signs their meaning, because they are rules for the use of these signs. So that the rules of inference are involved in the determination of the meaning of the signs. In this sense rules of inference cannot be right or wrong.
>
> (*RFM*, Part VII, §30, p. 398.)

In later writings one can find more explicit statements on the existence of a correspondence between the concepts of rule and meaning, such as, e.g., the following remark from *On Certainty* written between 1949 and 1951:

> (...) A meaning of a word is a kind of employment of it. For it is what we learn when the word is incorporated into our language.
>
> That is why there exists a correspondence between the concepts 'rule' and 'meaning'.
>
> (*On Certainty*, §§61–2, p. 10^e.)

Now the question of what kind(s) of rule can make this correspondence exist effectively has to be analysed by an investigation of the nature of *use* in relation to rule(s).

4.2.2. On the close interplay between rules and meaning

By assuming a philosophical position such that the meaning of logical connectives is said to be given operationally, Wittgenstein insists on the close interplay between rules and meaning. The point is made rather explicit on the fact that certain rules *constitute* meaning, and are not derived from it:

Are the rules, for example, $\sim\sim p = p$ for negation, responsible to the meaning of a word? No. The rules constitute the meaning, and are not responsible to it. The meaning changes when one of the rules changes.

(Ambrose (ed.) 1979, Part I (Philosophy. Lectures 1932-33), §2, p. 4.)

Later on in the same lecture on Philosophy, Wittgenstein discusses the primary importance of rules and remarks on the interdependence between definition and rules. With respect to the correspondence between the two concepts 'rule' and 'meaning' he makes an important point on how meaning can be given if not by rules:

If someone says the rules of negation are not arbitrary because negation could not be such that $\sim\sim p = p$, all that could be meant is that the latter rule would not correspond to the English word "negation". The objection that the rules are not arbitrary comes from the feeling that they are responsible to the meaning. But how is the meaning of "negation" defined, if not by the rules? $\sim\sim p = p$ does not follow from the meaning of "not" but constitutes it. Similarly, $p.p \supset q. \supset .q$ does not depend on the meanings of "and" and "implies"; it constitutes their meaning. (...)

(Id. Ibid.)

It seems convenient to observe that the rules constituting the meaning of the logical connective "implies", represented above by the sign '$\supset$', are rules of *consequence*, rules of *elimination*, and by no means rules indicating the assertability conditions for the connective in question. On the one hand, as we shall see later on, this is congenial with what I claim to have been his own intuition regarding the semantics of the mathematical language, which is the view that the meaning of a proposition is to be inseparable from the *consequences* one is allowed to draw from it. On the other hand, this stands right in opposition to most verificationistic semantics based on *assertability conditions*, such as, e.g., Gentzen-Prawitz proof-theoretic semantics, discussed in the previous chapter.

Now, there is a particularly interesting remark in the same collection of lecture notes edited by Ambrose, where certain rules are said to be giving meaning to a logical connective because they state how it works.

I have said that "$\sim\sim p = p$" gives a rule for the use of the word "not"; *it gives the meaning by stating how it functions.*

(Ibid., Part III (Lectures 1934-35), p. 153.)

(The *emphasis* is mine.)

If, on the one hand, the emphasis on 'giving the meaning by stating how it functions' is actually present not only in the early middle period as we have seen above (1932–35), but pervades also his later phase, on the other hand, the distinction between rules of sentence formation and rules of use of the sign in the language-game, which is very much a mark of later Wittgenstein's philosophy, is already present in his early middle period as it becomes clear from his view of language as a calculus and calculus as a game in the early thirties. When reading Waismann's *Ludwig Wittgenstein and the Vienna Circle* one often comes across passages where the distinction is made clear between rules of grammar and rules of application:

If I explain the calculus to you by specifying the rule[s] of grammar and application, then you will also understand the statement '*a* cannot be substituted for *a*'.

(*LW&VC*, p. 179.)

while the key semantical aspect is clearly given in the latter:

I understand a proposition by *applying* it. Understanding is thus not a particular process; it is operating with a proposition. *The point of a proposition is that we should operate with it.*

(Ibid., p. 167.)

I have found appropriate to add the 's' to the word 'rule' in the first line, given that in the original it says:

Wenn ich Ihnen den Kalkül erkläre, indem ich Ihnen die Regeln der Grammatik und der Anwendung angebe, dan verstehen Sie auch die Aussage >a ist nicht durch a ersetzbar<.

(The emphasis is mine.)

Observe that the plural 'die Regeln' is translated into the singular 'the rule' which seems to prevent the interpretation of 'the rules of grammar and application' as 'the

rule of grammar *and* the rule of application', which is what is said with different words in Wittgenstein's later account of propositions as in 'and what a proposition is is in one sense determined by the rules of sentence formation (in English, for example), and in another sense by the use of the sign in the language-game'. As we can readily see, such a later account of propositions distinguishes *two* aspects, namely the 'grammar' aspect (the rules of sentence formation) and the 'application' aspect (the use of the sign in the language-game).

In what concerns the term 'grammar', something needs to be said about the various uses Wittgenstein makes of the word. In connection with 'the use of the sign in the language-game', rules of grammar and rules of use are sometimes collapsed into a single class of 'rules of grammar'. There is, however, a great many evidences that although the concept of 'grammar' as a certain body of rules did play an important part in Wittgenstein's teachings on the position sustained by the meaning-is-use semantical dictum, it was not considered by Wittgenstein himself the ultimate semantical instrument:

> Grammar does not tell us how language must be constructed in order to fulfil its purpose, in order to have such-and-such an effect on human beings. It only describes and in no way explains the use of signs.
>
> (*Investigations*, Part I, §496, p. 138[e].)

Nevertheless, it would seem fair to say that by using the semantical dictum 'meaning-is-use' in favour of *assertability conditions*-based semantic theories, in order to challenge theories of meaning based on the traditional logicist truth-values semantics strongly criticised by Brouwer and the intuitionists, as reminded by Prof. Prawitz in:

> As pointed out by Dummett, this whole way of arguing with its stress on communication and the role of the language of mathematics is inspired by ideas of Wittgenstein and is very different from Brouwer's rather solipsistic view of mathematics as a languageless activity. Nevertheless, as it seems, it constitutes the best possible argument for some of Brouwer's conclusions.
> (...)

> I have furthermore argued that the rejection of the platonistic theory of meaning depends, in order to be conclusive, on the development of an adequate theory of meaning along the lines suggested in the above discussion of the principles concerning meaning and use. Even if such a wittgensteinian theory did not lead to the rejection of classical logic, it would be of great interest in itself.
>
> (Prawitz 1977, p. 18.)

is to insist that the 'rules of sentence formation' and not 'the use of the sign in the language-game' are to be taken as the central aspect of the meaning of a proposition. As we shall see from this and the next chapter, the rules of assertability conditions can hardly be said to convey any intentional ingredient.

And indeed, as it will be clear later on, especially in the next chapter, the view on the relationship between meaning and function, purpose, utility, is still very much germane to the view on the rôle of the consequences one is allowed to draw from the proposition containing the logical sign as its major one. Moreover, it allows yet another device used by Wittgenstein to be understood in the light of this uniform intuition: that is the remark on the need to understand the concept of explanation of meaning in order to understand what the very concept of meaning is. The concept of explanation of meaning is perhaps firstly used in the *Blue Book* (though brief references to it are made in the *Philosophical Remarks* as well as in the *Philosophical Grammar*), when the claim that 'meaning is not an object' is made explicit, in criticism to Frege's and his own (in the *Tractatus*) account of meaning in terms of truth-values.

And indeed, it is pointed out by Waismann in his *Ludwig Wittgenstein and the Vienna Circle* one of the key factors in the understanding of the fundamental difference by Frege's realist account of meaning and Wittgenstein's own view on how meaning of signs are to be viewed as.

> WAISMANN STATES THE DIFFERENCE BETWEEN FREGE AND WITTGENSTEIN: For Frege, the alternative is this: either a sign has a meaning, i.e. it goes proxy for an object—a logical sign for a logical object, an arithmetical sign for an arithmetical object—or it is only a figure, drawn on paper in ink.

But this is not a legitimate alternative. As the game of chess shows, there is a third possibility: in chess a pawn neither has a meaning in the sense of going proxy for anything, of being a sign *for* anything, nor is it merely a piece carved in wood and pushed about on a wooden board. It is only the rules of the game of chess that define what a pawn is.

This example shows that we must not say that a sign is either a sign for something or only a structure perceivable by our senses. Thus there is a legitimate element in formalism, a true core that Frege failed to see.

The 'meaning' of a pawn is, if you like, the totality of rules holding for it. And thus you can also say that the meaning of a numeral is the totality of rules holding for it.

WITTGENSTEIN AGREES

(*LW&VC*, p. 150.)

In proof-theoretic terms, the kind of rule which he says to constitute the meaning of logical connectives such as negation and generality is certainly not a rule indicating the assertability conditions – an introduction rule in Gentzen's terminology. It seems more likely to be an elimination rule instead. However, by looking at the case of negation, we tend to think more of a reduction rule than a mere elimination. Although negation is an especially problematic case we can still say that meaning is not related to an 'introduction' rule. This would seem to arise more clearly for less problematic cases in an environment where the proof rules are properly coded and classified, such as Prof. Martin-Löf's *Intuitionistic Type Theory*, as we shall see later on. The equality sign used by Wittgenstein could be seen as the 'reduces to' operation of the proof-theoretic metamathematical notion of *reduction* rule, which appears 'internalised' in that particular formal system.

In my analysis of the traditional proof-theoretic semantics, aiming at providing a more convincing alternative by enriching its account of meaning, I attempt to identify the kinds of rules that are needed to completely define the meaning of a logical connective. I want to say that the *introduction* and *reduction* rules give the meaning to the connective being defined by indicating when and how the connective can be used, respectively.

4.2.3. On the rôle of logical reduction in making meaning explicit

In his *Remarks on the Foundations of Mathematics*, he comes back to the point on how the meaning of logical connectives (generality sign, in particular) can be made explicit. Similarly to his comments in the *Notebooks*, but this time more incisive in the sense that the word 'meaning' is used explicitly, he expresses his belief that meaning can only be made explicit, or even be 'taught', by means of something of a rather 'dynamic' character such as a logical (perhaps elimination) inference.

> One learns the meaning of 'all' by learning that 'fa' follows from '$(x).fx$'.
>
> (*RFM*, Part I, §10, p. 41, *Circa* 1937–38.)

> And the meaning of '$(x).fx$' is made clear by our insisting on 'fa' 's following from it.
>
> (*RFM*, Part I, §11, p. 42, *Circa* 1937–38.)

By repeating the same argument already put forward as early as in a remark dated 24.11.14 in the *Notebooks*, Wittgenstein suggests how the meaning of the generality sign '$(x).fx$' can be made explicit just by an explanation of what kind of *consequences* one is allowed to draw from it, therefore without resorting to any kind of 'interpretation'. The process of 'logically following from' plays a definite rôle. I then say that the rôle is to make explicit the second aspect of the use of the universal quantifier, which shows the *utility* of asserting '$(x).fx$' – to allow the inference of 'fa' for any arbitrary 'a'.

A few lines below the last remark above, there is another remark that seems to be an attempt at strengthening the point on the connection between the *use* of the proposition '$(x).fx$' and the *consequences* one can draw from it, namely 'fa'. It says exactly:

> It strikes us as if something else, something over and above the *use* of the word "all", must have changed if 'fa' is no longer to follow from '$(x).fx$'; something attaching to the word itself.
>
> (*RFM*, Part I, §13, p. 42.)

4.2.4. The nature of use and the notion of 'meaning is use' in natural language

When attempting to relate meaning and use, one of the points that naturally arise is that of what is the nature of *use*. If an analysis of meaning is to be 'shifted' to an analysis of *use* in the hope that such a controversial matter like meaning can be more effectively analysable, it is expected that the nature of use be sufficiently clarified.

In some of his writings Wittgenstein does not seem to have made explicit all relevant aspects of *use*. Even in some later writings, the nature of *use* seems to have been just suggested. Nevertheless, there are later writings in which one can see signs of his wish to clarify the nature of 'the *use* of signs'. In the notebook published as *Remarks on Colour*, for example, he points out the need of saying more than to say that a word has a *use*, that it can be used in this and that context, etc.. The information on what *kind* of use a word has plays a major rôle in making clear the meaning one wants to be conveyed. Wittgenstein remarks:

> To say that a sentence which has the form of information has a use, is not yet to say anything about the *kind* of use it has.
>
> (*Remarks on Colour*, Part III, §336, p. 62^e.)

In another context, the need to show *how* one uses a certain word/sentence is pointed out. And indeed, the feature concerning the kind of use a word/sentence has can be easily described by an explanation of the intention, the conclusions one is supposed to draw, etc..

> The soul is said to *leave* the body. Then, in order to exclude any similarity to the body, any sort of idea that some gaseous thing is meant, the soul is said to be incorporeal, non-spatial, but with the word "leave" one has already said it all. Shew me *how* you use the word "spiritual" and I shall see whether the soul is non-corporeal and what you understand by "spirit".
>
> (*Zettel*, §127, p. 22.)

The reference to the need of saying what kind of *use* a sentence has when discussing its use appears to constitute the recognition of a more context (situation) dependence of the nature of *use*. The 'how to use' ('how it works', 'what effect(s) it is supposed to have') feature of a word plays a crucial rôle in determining its

meaning. Just by saying under which conditions a particular word (sign) can be correctly used does not suffice to make its meaning clear. One still needs to clarify the intention by saying with what purpose the word was employed, what is expected to be drawn from the utterance. It is necessary to make clear the consequences one allows to be drawn from a sentence where the particular word plays a major rôle.

4.3. Meaning-is-Use and Consequences

For the language of mathematics Wittgenstein seemed to have noticed the fundamental rôle of the *consequences* of an assertion with respect to its meaning. Already in his early days of philosophical work he had a particularly bright intuition on matters regarding logic and the meaning of logical signs, which later on becomes one of his great contributions to the understanding of the nature of semantics. And this was the connection between meaning and logical *consequence*. Nevertheless, the great contrast between the main picture of what is known as the 'early Wittgenstein' with corresponding *general form of propositions* on the one hand, and hist later *the meaning of a word is its use in the language* on the other hand, seems to overshadow what appears to be a very uniform and consistent intuition on the semantics of the language of mathematics. Essentially two facts seem to lead him to the phase characterised by what is called 'the early Wittgenstein', where *truth-conditions* were said to constitute meaning: the deep impression that Russell and Frege's logical calculus gave him as the potential 'language of science', and a great enthusiasm for the possibility of challenging Mauthner's sceptical views on the rôle of language in science. However, his second account of meaning and propositions (*meaning-is-use*), which looks similar to Mauthner's point of view in certain fundamental aspects, seems to have been just a consolidation of his own early intuition present along his investigations on the semantics of the mathematical language. According to this particular intuition, the meaning of a proposition was to be something inseparable from the *consequences* one could draw from it. In his later phase Wittgenstein attempts to recover the place of the *consequences* of an assertion as constituents of the meaning of the signs used, by introducing the notion of 'language-games'. Despite the use

of the term *game*, such a semantical notion seems to have been brought in to serve as an appropriate device to represent *meaning-is-use* from the point of view of an individual, by showing how it affects him/her to assert a proposition containing a particular logical sign, or, in other words, by demonstrating what *consequences* she/he allows to be drawn from the assertion. This appears to give a rather original character to Wittgenstein's position regarding mathematics as an activity, where both the individual and the collective ingredients play equally fundamental rôles, as opposed to other views on the matter which take either one of the two as the only aspect. *En passant*, concerning the specific point on the individual vs. the collective nature of the mathematical activity, it seems appropriate to recall a remark included in a late manuscript, written around 1949 focusing mostly on the notion of *certainty* and the crucial distinction from *knowledge*, perhaps suggesting a certain criticism on some *intuition*-based and *languageless* views of mathematics, from which one can see summarised in a few lines his own view on the rôle of both intuition and language in the development of mathematical knowledge, and the semantical rôle of the *use* of mathematical propositions:

> Knowledge in mathematics: Here one has to keep on reminding oneself of the unimportance of the 'inner process' or 'state' and ask "Why should it be important? What does it matter to me?" What is interesting is how we *use* mathematical propositions.
>
> (*On Certainty*, §38, p. 7^e.)

In spite of his apparently 'intermediate' position regarding the nature of the mathematical activity (between 'private' and 'public', as it were), a relatively successful formal account of meaning for the language of mathematics based on such a notion of language-games, Prof. J. Hintikka's *Game-Theoretic Semantics*, seems to appeal to certain unusual ingredients to explain the activity of 'doing mathematics', which is the recourse to 'communitarian' notions such as *dialogue*, *games*, *begging the question*, etc.. From such an alternative account of language-games for the mathematical language, which demonstrates a certain uniformity concerning the connections between 'showing the consequences of a proposition' and 'giving meaning to an assertion', I can show how a particular reading of the same 'language-games'

can still be made without having to resort to 'communitarian' views of mathematics, which is what seems to characterise the original formulation as well as most variations of Prof. Hintikka's *game-theoretic semantics*. Taking the *normalisation* (*reduction*) rules, a useful meta-mathematical device to prove normalisation theorems, as explanations of the *consequences*, in the sense that they show the effect of particular *elimination* rules on the result of *introduction* inferences, the *operations*-based account of meaning, which proof-theory itself was meant to be, becomes more convincing than the usual *assertability conditions*-based proof-theoretic semantics. The λ-calculus, e.g., looks more convincing than traditional natural deduction calculi by stating that the meaning of 'λ' is to be understood via two, instead of just one of its aspects of use: in addition to its *assertability conditions* rule (*abstraction* rule), the calculus provides its *consequences* rule (*β-reduction* rule). This is just because this way one can *understand* it, one can see the *consequences* of it.

4.4. The Evolution of Wittgenstein's Philosophy and the Early-Later Wittgenstein Distinction

The usual distinction 'early-later Wittgenstein' seems to leave a gap as to how the two phases of the philosopher's critique of language relate to each other. It simply does not sound satisfactory to say that 'the philosopher changed his mind', and the two stages are completely separated apart. They are indeed two sharply distinct phases with respect to the 'general form of propositions' and the meaning of logical signs, but there seems to be a fundamental point of view in common, beyond the obvious one concerning the rôle of a critique of language in philosophy. A particularly bright and original intuition regarding the connection between the meaning of an assertion and the consequences one can draw from it appears in different forms throughout Wittgenstein's philosophical studies, irrespective of 'phases'. Here I attempt to put together some of the remarks concerning this particular intuition, trying to demonstrate that the two phases do share a fundamental point of view regarding the semantics of mathematical language: the *meaning* of a proposition is something inseparable from the *conclusions* one can draw from it.

The several stages of the (later) Wittgenstein's account of how the meaning of propositions is determined: 1) by means of its method of *verification* (Waismann's *Ludwig Wittgenstein and the Vienna Circle*); 2) by reducing *meaning* to *explanation of meaning* (*Blue Book*) as an objection to the view that 'words correspond to objects' (*Blue Book*, *Investigations*), similar to Mauthner's reduction of the noun *meaning* to the verb *to mean* (instead of 'what is the meaning of ...' one asks 'what is meant by ...') as an objection to the view that 'words must stand for something' (*Beiträge I*, p. 159.); 3) by 'the use of the sign in the language-game' (*Investigations*); 4) by demonstrating its function, purpose, usefulness (*Last Writings on the Philosophy of Psychology, vol. I*); etc., can perhaps be made uniform by observing the development of his own early intuition along his successive writings on the subject.

It is not always explicitly said whether the *consequences* have a definite rôle to play in the meaning of a proposition. And this seems to leave room for some rather different interpretations of his account of propositions and meaning. The reference to 'the method of *verification*' is made explicit in Waismann's *Ludwig Wittgenstein and the Vienna Circle*, and says exactly:

> (...) The sense of a proposition is the method of its verification. A method of verification is not the means of establishing the truth of a proposition; it is the very sense of a proposition. In order to understand a proposition, you need to know the method of its verification. To specify it is to specify the sense of a proposition. You cannot look for a method of verification. A proposition can only say what is established by the method of its verification. (...)
>
> (*LW&VC*, p. 227.)

There has been a number of different readings of Wittgenstein's suggestion that a proposition is to be defined by the method of its verification. On the one hand, there are those who have interpreted 'method of verification' as 'method of showing the conditions under which one can correctly assert the proposition', therefore leaving the *consequences* of an assertion in a rather secondary place with respect to its meaning. On the other hand, those who take the point of view on the fundamental rôle of 'explaining the *consequences* one can draw from proposition' in determining

its 'method of verification', introduce unusual ingredients of a rather 'communitarian' nature to explain the mathematical activity, such as *dialogue*, *games*, etc.. By pointing out the existence of a meta-mathematical device to demonstrate the consequences of a proposition in a somewhat formal manner, I have attempted to offer a different reading of Wittgenstein's teaching on the relationship between verification and meaning which could be closer to the idea that the method of verification of a proposition is to be explained by showing how to verify the *conclusions* one is allowed to draw from it.

4.4.1. Change in Spirit but not in (Early) Intuition

As a theoretical-physicist-engineer very much impressed with Boltzmann and Hertz's mathematical models of natural phenomena, Wittgenstein sets out to write a philosophical logic treatise perhaps as an attempt to challenge the scepticism of Mauthner's critique of language, finding in Frege and Russell's logical calculus a way to overcome Mauthner's alleged unsuitability of language for scientific purposes. The result is the *Tractatus Logico-Philosophicus*, whose 'statement of faith' one could take as having been phrased as:

> All philosophy is a 'critique of language' (though not in Mauthner's sense). It was Russell who performed the service of showing that the apparent logical form of a proposition need not be its real one.
>
> (*Tractatus*, §4.0031, p. 19.)

Fritz Mauthner, Austrian-born (Bohemia) German philosopher of the end of last century and the beginning of this century (b. 1849, d. 1923), performed extensive investigations into the nature of language as related to reason, apparently motivated to attempt a formulation of an alternative to Kant's philosophy as presented in a *Critique of Pure Reason* by saying that philosophy should be an activity of *Critique of Language*. He seems to be best known as a sort of pioneering 'critic of language' whose main works *Beiträge zu einer Kritik der Sprache* (1902) and *Wörterbuch der Philosophie* (1910) seem to have had a significant influence on Wittgenstein's own

critique of language, especially on his second phase where the investigations concern much more ordinary language than formalised languages, similarly to Mauthner's own point of view on the rôle and the nature of ordinary language with respect to the foundations of reason and knowledge. It seems fair to say that in the German-speaking *fin-de-siècle* philosophy he stood for 'psychologism' in the same way that Frege stood for 'logicism'.

There seem to be many similarities between Mauthner's point of view on (linguistic) 'meaning' and the late Wittgenstein's view on the rôle of language and the relationship between meaning and use: rejection of the notion of absolute truth, meaning is no entity, philosophy as a therapy, etc., not to mention Wittgenstein's use of some of Mauthner's figures regarding:

(...) Philosophy is a battle against the bewitchment of our intelligence by means of language. (...)

(*Investigations*, Part I, §109, p. 47[e].)

such as the 'ladder' image in:

My propositions serve as elucidations in the following way: anyone who understands me eventually recognizes them as nonsensical, when he has used them — as steps — to climb up beyond them. (He must, so to speak, throw away the ladder after he has climbed up it.)

He must transcend these propositions, and then he will see the world aright.

(*Tractatus*, §6.54, p. 74.)

which may have been inspired in a remark contained in the *Beiträge*, vol. I, pp. 1–2, translated by Weiler as:

(...) 'If I wish to ascend in the critique of language, which is the most important business of thinking mankind, then I must destroy every rung of the ladder once I step upon it'. (...)

(Weiler 1970, p. 296.)

and is often said to have its origin in Sextus Empiricus':

'And again, just as it is not impossible for the man who has ascended to a high place by a ladder to overturn the ladder with his foot after his ascent, so also it is not unlikely that the Sceptic after he

has arrived at the demonstration of his thesis by means of the argument proving the non-existence of proof, as it were by a step-ladder, should then abolish this very argument.'

(Hamblin 1970, p. 95.)

or even an occasional expression, such as the famous concluding sentence of the *Tractatus*:

Wovon man nicht sprechen kann, darüber man muß schweigen.

which has been translated as:

What we cannot speak about we must pass over in silence.

(*Tractatus*, §7, p. 74.)

and which appears to echo a similar remark contained in Mauthner's *Beiträge*:

Es wäre Zeit, wieder schweigen zu lernen.

(*Beiträge I*, p. 159.)

In a previous note I have even pointed out the similarities of views both philosophers seemed to have had on the process of learning the meaning of a word in a language. The aspect of what *conclusions* can be drawn from the sentence where the word appears seems to be a fundamental aspect for both Wittgenstein and Mauthner.

Despite Wittgenstein's early beliefs in concepts such as 'simple facts' and 'a proposition is a picture of reality' expressed in the *Tractatus*, this second account of meaning and propositions, which looks similar to Mauthner's point of view in certain fundamental aspects, seems to have been just a consolidation of Wittgenstein's own intuition concerning his investigations on the semantics of the mathematical language since his early days of philosophical work. According to this particular intuition, the meaning of logical signs was for Wittgenstein something inseparable from the *consequences* one could draw from the proposition containing such a sign as the major one. Regarding Mauthner's point of view on ordinary language, the

nature of words and their meaning, the rôle of the *consequences* seems to have a similar position. G. Weiler, who has made Mauthner's ideas more accessible to the English-speaking philosophical community, makes a relevant remark on this particular matter:

(...) Mauthner's alternative account is that in the learning process there is very little psychological difference between concepts, judgements and inferences, since the stuff of all these notions is the associating of experience with words. Learning how to use a word is also learning what kinds of judgements can be made about the kind of thing that the word indicates, and learning how to make these judgements is also to learn *what kinds of inferences can be drawn from that judgement.*

(Weiler 1970, p. 250.)

(The *emphasis* is mine.)

referring to p. 262, vol. III, of Mauthner's *Beiträge.*

Apparently due to a certain disillusionment with the notion of logical atomism, which he himself had worked hard to formulate precisely, Wittgenstein in the *Investigations*, the main representative of his later phase, seems to go back to the ideas and the spirit he had criticised so strongly in his early phase. Two elements of Mauthner's spirit constitute the main picture of the *Investigations*: a clear (yet positive) scepticism and an attempt to go deeper into psychological concepts rather than into proper logical investigations. Concentrating more on an investigation of ordinary language and concepts concerning the philosophy of psychology than on formalised languages and the technicalities inherent in the philosophy of logic itself, he seems to be willing to leave the impression of a profound (and yet very positive) scepticism, exactly opposite to the spirit of the Wittgenstein of the *Tractatus*, when he enthusiastically writes to Russell saying that:

(...) I've written a book called "Logisch-Philosophische Abhandlung" containing all my work of the last six years. I believe I've solved our problems finally. This may sound arrogant but I can't help believing it. (...)

(*Letters to RK&M*, dated 13.3.1919, p. 68.)

However, the 'early' intuition on the rôle of explaining the meaning of logical propositions by showing the conclusions one is allowed to draw from it seems to prevail even in the early phase, despite the clear logicist position taken in most of his significant contributions to the foundations of Russell and Frege's logical calculus reported in the *Tractatus*.

Rev. Hallett remarks on such difference in spirit of the two Wittgenstein's, where it seems to be clear that the enthusiasm of the *Tractatus* stands right in opposition to the (positive) scepticism of the *Investigations*:

> (...) "Don't think, but look!" (§66). This exhortation, so characteristic of the *Investigations*, derives its full significance from passages like those just cited from the *Notebooks*, where Wittgenstein's thinking had already determined what he had to find, or, having spotted something interesting, forthwith spun a theory to cover all other cases. The *Tractatus* was a tissue of "musts" and of corresponding essences, brought forth as though by magic from the rags and dust of "inessential" cases (§52). So Wittgenstein later insisted: "Don't say: 'There *must* be something common . . . but *look and see* whether there is anything common to all" (§66).
>
> (Hallett 1977, p. 47.)

> (...) Wittgenstein grew increasingly doubtful about standard alternatives to the mere injunction "Look!" In the *Blue Book* (26b) he still thought that a definition is a good way to clear up the 'grammar' of a word, but in the *Investigations* he was far less sanguine (§182). And he experienced enduring doubts about the effectiveness even of description. At the time he was writing the *Investigations* he warned: "How words are understood is not told by words alone" (*Z* 144); "the verbal expression casts only a dim general light on the verbal expression" (*RFM* 54). Later he wondered: "Am I not getting closer and closer to saying that in the end logic cannot be described? You must look at the practice of language, then you will see it" (*OC* 501; cp. *PG* 40f). Quite similar misgivings had caused Wittgenstein to characterize the *Tractatus* as a mere ladder, to be cast aside once the reader had come, by its means, to "see the world aright" (6.54; cp. §144). The ladder simile fits the *Investigations* better still, for there the reader does not mount on rungs of solid, though "unsayable" doctrine, but by means of mere questions, and simple samples well displayed, many of them imaginary. (...)
>
> (Id. Ibid.)

The aspect of less enthusiasm prevailing in the Wittgenstein of the *Investigations* was somewhat sadly reminded by a great logician and philosopher. In Moscow, by

the occasion of the VIIIth International Congress of Logic, Methodology and Philosophy of Science (VIII LMPS), in a conversation with Prof. Francisco Miró Quesada (Lima, Peru) about the two 'distinct' Wittgenstein's. Prof. Quesada, an eminent figure in the philosophy of logic in South America, very much an admirer of Bertrand Russell, expressed his admiration for the Wittgenstein of the *Tractatus*, who according to him had impressed Russell by formulating precisely and economically the notion Russell himself had experienced hard times in trying to get the grips on: the notion of logical atomism. His expression of disappointment with Wittgenstein's scepticism shown in the *Investigations* sounded like a testimony of how much the two Wittgenstein's are seen as so radically different from each other. And this picture of the two men, which seems to prevail even among some of Wittgenstein's admirers/followers, makes anyone think why the great contribution for the foundations of logic and mathematics given in the *Investigations* itself, perhaps more importantly than in any other of his own writings, especially regarding the nature of semantics ('meaning-is-use'), has not been so far widely recognised. 'Would it make it easier to observe whether there were any points in common among all phases of Wittgenstein's philosophical views with respect to the foundations of logic?' and 'How does the notion of *language-games*, which characterises Wittgenstein's semantical views in his later phase, contribute for a better understanding of the semantics of logics and the language of mathematics?'. These are exactly the sort of questions that comes straight into one's mind. One feels inevitably pushed to give a closer look at Wittgenstein's writings especially where he made reference to the meaning of logical signs without resorting to *values* (*true*, *false*, etc.). The rôle of an explanation of the *consequences* one is allowed to draw from a proposition in the clarification of its *meaning* has been emphasised in many places, and the my own suspicion that his views on the matter seemed to be somewhat uniform across the two sharply separated phases was being confirmed and consolidated as the re-readings of early as well as late writings were made with more careful eyes. The 'last drop', as it were, was a sign of how early such an intuition was born.

4.4.2. 'Disturbingly' Original Intuition

Concerning the mathematical language and its semantics, Wittgenstein seemed to have observed the important rôle of the *consequences* of an assertion with respect to its meaning, since his early days of doing philosophy. Already in his very beginning of philosophical work he had a particularly bright intuition on matters regarding logic and the meaning of logical signs, which later on becomes one of his great contributions to the understanding of the nature of semantics. And that was the connection of meaning with logical *consequence*. As early as 1912, in a letter to Russell, dated 1.7.12, he writes:

> Will you think that I have gone mad if I make the following suggestion?: The sign $(x).\varphi x$ is not a complete symbol but has meaning only in an inference of the kind: from $\vdash \varphi x \supset_x \psi x.\varphi(a)$ follows $\psi(a)$. Or more generally: from $\vdash (x).\varphi x.\varepsilon_0(a)$ follows $\varphi(a)$. I am—of course—most uncertain about the matter but something of the sort might really be true.
>
> (*Letters to RK&M*, Letter *R.3*, p. 12.)

(The emphasis is mine.)

Observe that the meaning of the proposition referred to ('$(x).\varphi x$') is said to have meaning only in an inference which shows the *logical consequences* of asserting it ('$\varphi(a)$'). Bringing uncertainty perhaps due to its rather disturbing originality, Wittgenstein's intuition concerning the connections between seeing the consequences of an assertion and understanding its meaning becomes an important ingredient in the development of his critique of language. It seemed to be a rather natural though not easily perceived intuition, especially concerning ordinary languages, that of making a link between the meaning of an assertion and the consequences one could draw from it (cf. the French *vouloir dire.*).

Indeed, this comes as another evidence of the early origin of an intuition that Wittgenstein seems to have carried all the way through his investigations on the semantics of the mathematical language. If one looks at more recent languages of mathematics (λ-*Calculus*, *Category Theory*, *Frege structures*, etc.) one is tempted to say that his intuitions are beginning to make themselves manifest. Perhaps a

simple and straightforward example of the force of the argument that the meaning of a logical connective is better viewed as something related to the consequences, comes from the language of λ-calculus, the language of *computable* functions. In order to make someone understand the meaning of 'λ' one has to give more than just the abstraction rule, which can be looked at as a 'grammatical' rule for 'λ'. The process of *normalisation* shown in the β-rule seems to clarify the meaning of 'λ' just because it shows how it works, or, what can be obtained from a λ-term 'assertion'. The *utility* of 'λ', which seems to be to capture the notions of generality and substitution inherent to the concept of *function*, is only made clear when a rule showing the consequences of a λ 'assertion', such as what the rule of *β-normalisation* does, is indicated.

Nevertheless, uncertain as Wittgenstein seemed to have been towards this particularly bright intuition, apart from a single remark on how the presence of *generality* in the sign '(x).φx' was made clear via its logical *consequence*, which was written in 1914 and published in the *Notebooks*, he does not come back to the point on the rôle of *consequences* with respect to meaning, until *circa* 1937–38 when he writes remarks on the foundations of mathematics. In the meantime, the enthusiasm with Russell and Frege's logical calculus seems to take him to express a position such that meaning is to be given by an object or entity, a truth-value. That is the phase characterised by what is called 'the early Wittgenstein', where *truth-conditions* were said to constitute meaning. Later on, acknowledging drawbacks in that early account of meaning presented in the *Tractatus*, he seems to go back to what appeared to be his original intuition, and attempts to recover the place of the *consequences* of an assertion as constituents of the meaning of the signs used.

At this point one would ask how such a notion of *consequences* of an assertion as being part of its meaning is expressed in his aphoristic writings of the early phase, where the semantics is clearly (Frege-) realist. And, indeed, it is interesting to observe that even in an early version of the *Tractatus*, for example, published as *Prototractatus*, with historical comments and a facsimile of the manuscript, one can

find a remark of essentially the same nature as some later ones regarding the rôle of the second 'sense' of (his later account of) a proposition, which concerns precisely the 'consequences' aspect. It says:

(...) The requirement of determinateness could also be formulated in the following way: if a proposition is to have sense, the syntactical employment of each of its parts must have been established in advance. For example, it cannot occur to one only subsequently that a certain proposition follows from it. *Before a proposition can have a sense, it must be completely settled what propositions follow from it.*

(*Prototractatus*, §3.20103, p. 65.)

(The *emphasis* is mine.)

For some reason, however, this particular remark does not appear in the final version of the *Tractatus*.

Going a bit further back one can still find remarks of the same nature. In the course of the elaboration of the *Tractatus* a number of remarks were being collected in notebooks, which were posthumously published as *Notebooks 1914–16*. In one of these remarks, dated 25.10.14, he speaks of the need of certain *logical properties* for a method of symbolising to have, in order to be an adequate method of capturing meaning via symbols:

If we were to try and express what we express by means of "(x).ϕ(x)" by prefixing an index to "ϕ(x)", e.g., like this: "Gen.ϕ(x)", it would not be adequate (we should not know what was being generalised).

If we tried to shew it by means of an index to the "x", e.g., like this: $\phi(x_G)$, it would still not be adequate (in this way we should not know the scope of the generality).

If we thought of trying to do it by inserting a mark in the empty argument places, e.g., like this: "(G, G).ϕ(G, G)" it would not be adequate (we could not settle the identity of the variables).

All these methods of symbolising are inadequate *because they do not have the necessary logical properties*. All those collections of signs lack the power to portray the required sense – in the proposed way.

(*Notebooks*, p. 18^e.)

Although not making clear what kind of logical properties would be needed for a method of symbolising to be adequate, the recognition of the need for such logical properties sets up grounds for the investigations of what (logical) form these properties should take. Later in the same *Notebooks*, in a remark dated 24.11.14, a form of logical inference is suggested:

> That the *proposition* "$\phi(a)$" can be inferred from the *proposition* "(x).ϕ(x)" shews how generality is present even in the sign "(x).ϕ(x)".
>
> And the same thing, of course, holds for any generality notation.
>
> (*Notebooks*, p. 32^e.)

Observe the clear character of a logical property speaking of the *consequences* of the assertion being suggested as adequate to show the meaning of the corresponding proposition.

Early in the *Tractatus* itself, which is known for its 'realist', *truth*-values based semantics characterising what is called the 'early Wittgenstein', he makes a remark which is a sign of an 'anti-realist' semantics only taken up more firmly and developed more intensively in what is known as the 'later Wittgenstein' phase:

> (The possibility of inference from $(x).fx$ to fa shows that the symbol $(x).fx$ itself has generality in it.)
>
> (*Tractatus*, §5.1311, p. 38.)

An explanation of the logical (elimination) inference for the logical sign of universal quantification is showing its generality meaning. And this is done together with the purely classical (realist) semantics of the *Tractatus*.

Nevertheless, uncertain as Wittgenstein seemed to be regarding this particularly bright intuition, apart from a single remark on how the presence of *generality* in the sign '(x).φx' was made clear via its logical *consequence*, which was written in 1914 and published in the *Notebooks*, he does not come back to the point on the rôle of *consequences* with respect to meaning, until *circa* 1937–38 when he writes remarks on the foundations of mathematics. In the meantime, the enthusiasm with Russell and

Frege's logical calculus seems to take him to express a position such that meaning is to be given by an object or entity (a truth-value). That is the phase characterised by what is called 'the early Wittgenstein', where *truth-conditions* were said to constitute meaning. Later on, acknowledging drawbacks in that (early) account of meaning expressed in the *Tractatus*, he seems to go back to what appeared to be his original intuition, and attempts to recover the place of the *consequences* of an assertion as constituents of the meaning of the signs used. Similarly to his comments in the *Notebooks*, but this time more incisive in the sense that the word 'meaning' is used explicitly, he expresses his belief that meaning can only be made explicit, or even be 'learnt', by means of something of a rather 'explanatory' character such as showing the *consequences* that can be drawn from the assertion, via a logical (*elimination*) inference.

> One learns the meaning of 'all' by learning that 'fa' follows from '$(x).fx$'.
>
> (*RFM*, Part I, §10, p. 41.)

> And the meaning of '$(x).fx$' is made clear by our insisting on 'fa' 's following from it.
>
> (*RFM*, Part I, §11, p. 42.)

Similarly to the early remark mentioned above, by insisting on the fact that 'fa', where 'a' is any arbitrary individual, is obtained from the utterance of '$(x).fx$', one can make clear the generality meaning of the major logical sign of the utterance made. Not quite surprisingly, such *explanation of meaning* for the logical sign of universal quantification sounds similar to the explanation of meaning for the 'λ' sign, whose rules of manipulation (especially the *β-reduction*) were apparently first thought as a convenient coding of the functional notion of generality and substitution.

In the later phase the ghost of uncertainty seemed to have gone away, and Wittgenstein appears more incisive on the idea that there are links between the *logical consequences* of an assertion and its meaning, besides admitting that a puzzling notion such as *meaning* could not be simply defined as something that could

be either 'true' or 'false' such as it was strongly said in the *Tractatus*. Attempting to offer an alternative account of propositions to that 'truth-value'-based of the *Tractatus* by means of the new concepts introduced in the later phase, which finds in the *Investigations* its *chef-d'œuvre*, he acknowledges serious drawbacks in his early account of meaning and propositions, criticising the 'words correspond to objects' picture of the *Tractatus*. The criticisms to the idea that a proposition is whatever can be true or false become rather explicit at the point where he attempts to formulate an alternative to that picture of truth-values developed in the *Tractatus*, replacing it by a new 'definition' of what a proposition should be. Such new definition is formulated by means of a newly introduced notion of 'language-games', an idea that seems to pervade the whole of the *Investigations*, and certainly most of his later writings.

4.4.3. The Objection to the View on 'Meaning-is-an-Object' and the Notion of Explanation of Meaning

In ordinary language it is usually the case that one gives an explanation of the meaning of an assertion by providing an exposition of its possible consequences. And, yet, one hardly appreciates the important rôle of the *consequences* in determining the meaning of the assertion made. When asked 'What do you mean?' or 'How do you mean?' one normally answers via an explanation of what conclusions one is entitled to draw from the assertion whose meaning is being questioned. For this reason, it would appear fair to say that the very concept of meaning can be better understood by reference to a certain process of *explanation of meaning*, or, if one can put it this way, an explanation of what kind of conclusions one can draw from the assertion. Nevertheless, most of the studies on the nature and the foundations of language, especially in this second half of the xxth century, have had a particularly strong emphasis on grammatical analysis. A number of important issues related to the (syntactic) structure of language have been clarified. It seems that little, however, has been achieved towards a better understanding of the rôle of language

and the nature of semantics. Fundamental aspects which are naturally beyond purely grammatical explanations have often been neglected. Those are the aspects I would like to call the *consequences* of assertions.

Besides explicit references concerning the idea that grammar can only describe, and in no way explain the use of signs in a particular language, such as:

> Grammar does not tell us how language must be constructed in order to fulfil its purpose, in order to have such-and-such an effect on human beings. It only describes and in no way explains the use of signs.
>
> (*Investigations*, Part I, §496, p. 138[e].)

Wittgenstein was rather persistent, especially throughout his very latest writings (notebooks, conversations, etc.), in trying to persuade the reader that the grasping of concepts like meaning and understanding of signs in a language could be made significantly easier by thinking in terms of what could be done with the signs, what conclusions could be drawn from the assertion, what utility the sign (word) had in the sentence, etc.. The reference to the understanding of the notion of *meaning* as being attached to the understanding of the idea of *explanation of meaning* comes as an idea which certainly carries a strong appeal to the intuition.

It seems to be in the *Blue Book* that Wittgenstein first attempts to make the point on how easier the understanding of the concept of *meaning* can be made when one looks at the idea of *explanation of meaning*, serving as a cure to our temptation to look for an object which would correspond to the meaning of a particular word. Such link would seem to be opening a new perspective by suggesting a radical break from the usual idea that meaning is something directly related to content, object and reference. The connections with a less 'object-oriented' concept such as *use*, starts to emerge more clearly.

> The questions "What is length?", "What is meaning?", "What is the number one?" etc., produce in us a mental cramp. We feel that we can't point to anything in reply to them and yet ought to point to something. (*We are up against one of the great sources of philosophical bewilderment: a substantive makes us look for a thing that corresponds to it.*)

Asking first "What's an explanation of meaning?" has two advantages. You in a sense bring the question "what is meaning?" down to earth. For, surely, to understand the meaning of "meaning" you ought also to understand the meaning of "explanation of meaning". Roughly: "let's ask what the explanation of meaning is, for whatever that explains will be the meaning." Studying the grammar of the expression "explanation of meaning" will teach you something about the grammar of the word "meaning" and will *cure you of the temptation to look about you for some object which you might call "the meaning"*.

(*Blue Book*, p. 1.)

(The *emphasis* is mine.)

From my own understanding of Wittgenstein's point of view on the idea that the concept of *meaning* can be better understood by looking at the notion of *explanation of meaning*, I suggest that the meta-mathematical device of *normalisation* seems to be an appropriate instrument to capture formally such *explanation of meaning* for the particular case of the language of mathematics. This is just because *normalisation* rules show the *consequences* of 'forming' or 'asserting' mathematical sentences, i.e., what can be obtained from such an assertion.

Traces of such an attractive recipe for 'curing us of the temptations to look about us for some object we might call "the meaning"' is found back in Fritz Mauthner's account of meaning for ordinary language:

(...) Mauthner wishes to dispose of the fallacy mentioned at the beginning of this section, i.e. that there are always things corresponding to words, which are their meanings: 'Most people suffer from this intellectual weakness that they believe that because there is a word it must stand for something; because there is word there must correspond the word something real'. (...)

(Weiler 1970, p. 119.)

Prof. Weiler in fact refers to the inspiring remarks on the concept of meaning as put forward by Fritz Mauthner in his *Contributions to a Critique of Language*. In the first of his 3-volume extensive analysis of the rôle and the nature of language in human reason and knowledge Mauthner refers to a certain intellectual weakness which seems to be rather close to what Wittgenstein called 'one of the great sources

of philosophical bewilderment: a substantive makes us look for a thing that corresponds to it':

(...) Die meisten Menschen leiden an dieser geistigen Schwäche, zu glauben, weil ein Wort da sei, müsse es auch das Wort für Etwas sein; weil ein Wort da sei, müsse dem Worte etwas Wirkliches entsprechen.

(*Beiträge I*, p. 159.)

Such 'intellectual weakness' referred to by Mauthner seems to have been the object of criticisms in most of Part I of Wittgenstein's *Investigations*, which starts with a quotation from St. Augustine's *Confessions* describing his word-object account of language as in 'Thus, as I heard words repeatedly used in their proper places in various sentences, I gradually learnt to understand what objects they signified'. Wittgenstein's criticisms start by saying:

These words, it seems to me, give us a particular picture of the essence of human language. It is this: the individual words in a language name objects—sentences are combinations of such names.—In this picture of language we find the roots of the following idea: Every word has a meaning. This meaning is correlated with the word. It is the object for which the word stands.

(*Investigations*, Part I, §1, p. 2^{e}.)

The point of view I take in this chapter is essentially that which motivated the development of my 'semantics of use' where *meaning* is not taken as an object or entity, as it were, but as the result of explaining what are the *consequences* of asserting the particular proposition. The profound insights that Wittgenstein's writings seem to provide as to the connections between *meaning* and *use*, *utility*, *purpose*, etc., has served as comfortable grounds for a closer look on issues regarding theories of meaning for the mathematical language where one can still find traces of the usual rather 'object-oriented' account of meaning, even in some of the more recent verificationistic theories of meaning.

In fact, the intuition concerning the strong connections between the nature of the procedure of *explaining the meaning* of a proposition and that of the act of

demonstrating the consequences one is allowed to draw from it seems to go rather far back in Wittgenstein's early writings.

4.4.4. The Aspect of Consequences and the Notion of Language-Games

Now, if such a view on the existence of a sole Wittgensteinian semantical intuition towards the language of mathematics, which takes the 'consequences' of an assertion as being constituents of its meaning, can be taken as a contentious one, how and where does it emerge in more general terms in his later phase? To answer this question I have to look at the appearance of a notion which seems to be a true 'later Wittgensteinian' mark in the whole philosophy of language: the notion of *language-games* (*Sprachspiel*), which can only find a parallel in Mauthner's concept of *game-rules* (*Spielregel*)[4]. The (later) account of propositions which characterises the *Investigations* is made explicit in the following remark:

> (...) Now it looks as if the definition — a proposition is whatever can be true or false — determined what a proposition was, by saying: what fits the concept 'true', or what the concept 'true' fits, is a proposition. So it is as if we had a concept of true and false, which we could use to determine what is and what is not a proposition. What *engages* with the concept of truth (as with a cogwheel), is a proposition.
>
> But this is a bad picture. It is as if one were to say "The king in chess is *the* piece that one can check." But this can mean no more than that in our game of chess we only check the king. Just as the proposition that only a *proposition* can be true or false can say no more than that we only predicate "true" and "false" of what we call a proposition. And what a proposition is is in one sense determined by the rules of sentence formation (in English, for example), and in another sense by the use of the sign in the language-game. (...)
>
> (*Investigations*, Part I, §136, pp. 52^{e}–3^{e}.)

(The emphasis is mine.)

[4]see, e.g., p. 109 of Weiler 1970, which refers to Mauthner's discussion on 'Sprache eine Spielregel' in *Beiträge I*, p. 25.

In the next chapter I demonstrate that the meta-mathematical device of *normalisation* can be taken as a possible reading of such a notion of *language-games* for the case of the mathematical language, and how it indeed complies with Wittgenstein's intuition: the *normalisation* rules show 'the effect of *elimination* operations on the result of *introduction* inferences'. The aspect of the *consequences* is represented by the notion of 'the use of the sign in the *language-game*'. It is also shown that the most widely known casting of the notion of *language-games* into a formal account of semantics for the language of mathematics, Prof. J. Hintikka's *Game-Theoretical Semantics*, not quite surprisingly, ends up taking the *consequences* aspect as the one having the fundamental rôle in determining the meaning of logical constants.

As a matter of fact, *Game-Theoretic Semantics* can be understood as an attempt to put into effective terms (later) Wittgenstein's suggestion regarding the *language-games* as the link between language and reality. It has been worked out for a certain time by Prof. J. Hintikka and his followers as a whole body of formal concepts based on their own reading of Wittgenstein's (later) account of propositions and language. The basic ideas of such a body of formal as well as informal logical notions were basically introduced by reference to formal first-order quantificational languages. A language L is assumed to be an interpreted language, allegedly due the possibility of talking about truth and falsity of its sentences, and its being interpreted is explained by saying that one is given some domain of individuals D on which all the nonlogical constants of L plus the names of the members of D would have a definite truth-value. The departure from Tarskian truth-value-based theory of meaning is made by explaining the semantics of each logical constant by means of what is called *semantical games*. In the description of the *Game-Theoretical Semantics* ('*GTS*') *semantical games* are defined by Prof. Hintikka, where there is supposed to be two 'players' called *Myself* and *Nature*, as:

($G.\ A$) If A is atomic, then I have won $G(A)$ and *Nature* has lost if A is true. If A is false, *Nature* has won and I have lost.

(*G.* &) $G(S_1 \,\&\, S_2)$ begins by *Nature*'s choice of S_1 or S_2.
The rest of the game is $G(S_1)$ or $G(S_2)$, respectively.

(*G.* ∨) $G(S_1 \vee S_2)$ begins by *Myself*'s choice of S_1 or S_2.
The rest of the game is $G(S_1)$ or $G(S_2)$, respectively.

(*G. U*) $G((\mathrm{x})S[\mathrm{x}])$ begins by *Nature*'s choice of a member of D.
Let the name of the member chosen be 'a'. The rest of the game is then $G(S[a])$.

(*G. E*) $G((E\mathrm{x})S[\mathrm{x}])$ is defined likewise except that 'a' is chosen by *Myself*.

The truth of a sentence S of L would then be defined as the existence of a winning strategy in $G(S)$ for *Myself*. Similarly, the falsity of S would be given by the fact that the other 'player', *Nature*, has a winning strategy.

As it can be seen from the definition above, the *semantical games* of Prof. Hintikka's *Game Semantics* show, in the end, how *Nature* plays by asking *Myself* to comply with the commitments *I* make when asserting a proposition. *Nature* demands *Myself* to verify the *consequences* of the assertions which *I* (*Myself*) lay down. In other words, every step in the game is defined by a procedure to *eliminate* the corresponding connective. The meaning of each logical constant is explained by demonstrating the *consequences* one can draw from the respective proposition.

4.5. Games and Commitments: the Rôle of Consequences

Prof. Hintikka's *Game-Theoretical Semantics* appears to be the most widely discussed formal account of the semantics of the mathematical language which hints on what I suggest to be Wittgenstein's own view on the rôle of the consequences of a proposition with respect to its meaning. However, such a formal account brings in a

particular conception of the mathematical activity where there must be a 'community' (at least two 'players', even if one is just *Nature*) acting on the process. Making a less community-oriented reading of what a language-game should be, I have attempted to make a point regarding the place of the consequences of an assertion with respect to its meaning as a particularly bright intuition which seems to go beyond 'phases' in Wittgenstein's philosophical writings. As far as the meaning of logical constants is concerned, I say that the metamathematical device of *normalisation* can in most cases without any loss be used to capture the essence of what Wittgenstein seemed to have meant by 'the use of the sign in the *language-game*': the aspect of the meaning of a proposition which concerns the *consequences* that can be drawn from it. The emphasis on the *consequences* aspect, besides the *grammar* (conditions for assertion) aspect, is in fact what makes the sharp distinction between most 'anti-realist' alternatives to classical semantics for the language of mathematics and Wittgenstein's celebrated *meaning-is-use* semantical paradigm. It is indeed a great departure from the generally known *assertability conditions*-based alternative to the 'realist' *truth conditions*-based account of meaning for the mathematical language.

Prof. M. Dummett, in his theory of meaning and truth, attempts to provide a convincing argument to support his 'anti-realist' criticisms by offering an account of meaning based on the *commitments* one makes when asserting a proposition. A significant part of Prof. Dummett's explanations of meaning recall the aspect of *commitments* as a key aspect in a theory of meaning. And to recall the aspect concerning the *commitments* one makes by asserting a sentence as a fundamental aspect of its meaning seems to amount to essentially the same as to recall the aspect regarding the *consequences* one can draw from the sentence. One can hardly distinguish the *commitments* aspect from the so-called *consequences* aspect. It seems reasonable to suppose that the *commitments* one is prepared to assume by making an assertion correspond to the *consequences* of the assertion that he/she is prepared to *verify*.

It would seem to have been a strong alternative as an anti-realist theory of meaning had he provided a logical counterpart to this aspect which apparently received

such a denomination of *commitment* from Prof. Dummett himself. Unfortunately, by putting forward his own reading of Wittgenstein as another conceptual support for the *assertability conditions* alternative to realist semantics, he seems to leave a gap as to what place such *commitments* would take in an account of meaning based on the conditions for assertion alone. The emphasis on the *commitments* aspect is made clear when he says that his theory of meaning *links the content of an assertion with the commitment that a speaker undertakes in making that assertion*[5].

Prof. Prawitz's comments on Prof. Dummett's use of *commitments* and the aspects relating meaning and use within the language of mathematics does highlight the distinction that must be made between the two aspects of the use of a sentence, namely the assertability conditions and the consequences one can draw from the sentence. It also highlights the fundamental rôle of the *consequences*:

> Dummett points out that quite generally, there are two aspects of the use of an (assertive) sentence: (1) the conditions under which it can be correctly asserted, and (2) the commitments made by asserting it. In the case of mathematics, aspect (1) is expressed in the rules for inferring the sentence, and aspect (2) in the rules for drawing consequences from the sentence. When we learn to use sentences, what we learn are these *two* kinds of rules.
>
> (Prawitz 1977, p. 7.)

(The *emphasis* is mine.)

For some reason, however, Prof. Dummett's own reading of Wittgenstein seems to remain essentially in line with the *assertability conditions*-based anti-realist alternative by saying that the meaning of logical constants is determined by the *conditions for assertion*:

> (...) [The Meanings of the Logical Constants] The meaning of each constant is to be given by specifying, for any sentence in which that constant is the main operator, what is to count as a proof of that sentence, it being assumed that we already know what is to count as a proof of any of the constituents. (...)
>
> (Dummett 1977, p. 12.)

[5] p. 126 of Dummett 1976.

His joint research programme with Prof. Prawitz on a theory of meaning for the language of (intuitionistic) mathematics, ends up 'neglecting' the *commitment* aspect in favour of a certain priority for the *conditions for assertion* with respect to the meaning of a logical proposition.

4.6. Ordinary Language

Observing how our ordinary language works we can easily come to agree with Wittgenstein's intuition. In ordinary speech it is often the case that one accepts an explanation of the meaning of an assertion by listening to an exposition of its possible consequences. And, yet, one hardly appreciates the important rôle of the *consequences* in determining the meaning of the assertion made. Questions of the sort 'What do you mean?' or 'How do you mean?' are invariably answered via an explanation of what conclusions one can draw from the assertion whose meaning is being questioned.

The connections between understanding the meaning of a statement and seeing the consequences one is allowed to draw from it is noticed by Wittgenstein in many different contexts. In Y. Smythies' notes taken from Wittgenstein's *Lectures on Religious Belief*, one can find remarks such as:

> "The separation of soul from body" [only had a public interest.] This may act like black curtains or it may not act like black curtains. *I'd have to find out what the consequences [of your saying it] are*. I am not, at least, at present at all clear. [You say this] — "So what?" — I know these words, I have certain pictures. All sorts of things go along with these words.
>
> If he says this, *I won't know yet what consequences he will draw. I don't know what he opposes this to*.
>
> (*Lectures and Conversations on APRB*, p. 69.)

(The *emphasis* is mine.)

As his exposition goes on, it becomes clearer the point on the rôle of seeing the consequences one is allowed to draw from the assertion for its understanding:

If you say to me—"Do you cease to exist?"—I should be bewildered, and would not know what exactly this is to mean. "If you don't cease to exist, you will suffer after death", there I begin to attach ideas, perhaps ethical ideas of responsibility. The point is, that although these are well-known words, and although I can go from one sentence to another sentence, or to pictures *[I don't know what consequences you draw from this statement]*. (...)

(*Lectures and Conversations on APRB*, pp. 69–70.)

(The *emphasis* is mine.)

Thinking of meaning in terms of method of verification one can still see the definite rôle of consequences in his understanding of what verification is to mean:

(...) When I say he's using a picture I'm merely making a *grammatical* remark: [What I say] can only be verified by the consequences he does or does not draw.

(*Lectures and Conversations on APRB*, p. 72.)

(The emphasis is mine.)

It might be convenient to recall a significant remark on the matter from Wittgenstein's later writings, which makes very explicit reference to the connections meaning-consequences pervading most of his writings:

For the question is not 'What am I doing when . . .?' (for this could only be a psychological question) — but rather, 'What meaning does the statement have, what can be deduced from it, *what consequences does it have*?'

(*RPP II*, §38, p. 8e.)

(The *emphasis* is the Translators'.)

4.7. Early Intuition and Modern Mathematical Languages

Looking at the development of new languages to express mathematical constructions one is tempted to say that Wittgenstein's intuition seems to be making itself manifest in 'modern' mathematical languages: *normalisation* rules often appear as an instrument of meaning explanation by showing the effect of destructors on the

constructs, or, in other words, by demonstrating the *consequences* one can obtain from a term containing a particular logical sign, how one can *use* it.

With the ever increasing 'constructivisation' of usual mathematical concepts (or better, 'linguisticisation', as I understand it) — abstraction, pairing, disjunction, etc., — there have been numerous examples of how the meaning of logical signs are coded into what I call *reduction* rules, which are merely 'formal explanations of the *consequences*'. The 'constructor' comes together with the 'destructor'. The combination of both into a *reduction*, representing what appears to be one of the fundamental principles of logic — the Inversion Principle — completes the 'definition by use' of the particular logical sign.

As an example I can recall the language of *λ-Calculus*, first thought of as the language of computable mathematics: the rule of *β-reduction*, which demonstrates the effect of the destructor 'APPLY' on the construct 'λ', is given as a primitive rule of the calculus (Barendregt 1981). Worth mentioning is the language of category theory, another example of a modern mathematical language where the general argument seems to apply: constructs come together with their projections such as '$< , >$' and '*fst*', '*snd*'; '*inl*', '*inr*' and '[,]'; 'Λ' and '*apply*' (Dybjer 1985). One of the first programming languages whose mathematical meaning was more easily understood, LISP seems to be another example of a modern language designed to express mathematical objects which again follows the same pattern: the construct 'cons' comes together with its projections 'CAR' and 'CDR', and the construct is understood in terms of how the destructors operate on it (McCarthy et. al. 1962). A formal system of natural deduction style logical inference rules called *Intuitionistic Type Theory*, developed by Prof. P. Martin-Löf as a formalisation of constructive mathematics in proof-theoretic terms, is yet another example of such 'modern' mathematical languages where meaning is given via explanations of the *consequences*: the construct for Π-types 'λ' is explained by the equality rule that shows the effect of the corresponding destructor 'APPLY' on 'λ' (Martin-Löf 1984). The language of *Frege structures* developed by Prof. P. Aczel to further explore Frege's logical notion of set has a general pattern of defining *systems* in terms of destructors operating

on constructs, which seems to suggest that it can be included as an example of a modern mathematical language where meaning is determined by *use* ('explained' by the *reduction* rules): *lambda systems* are defined by a demonstration of how the destructor 'APPLY' operates on the construct 'λ; similarly, *pairing systems* are defined by the *reduction* rules showing the effect of destructors 'p' and 'q' on the construct 'PAIR' (Aczel 1980). 'Modern' and more 'mathematical' treatments of notions inherent to computer programming, the techniques of algebraic specification of abstract data types, appear to serve as yet another example of language for mathematical definitions which demonstrates the rôle of *reduction* rules as semantic instruments: stack definitions are given in terms of the destructors 'POP' and 'TOP' operating on the construct push; binary tree definitions present the effect of destructors 'SUBLEFT', 'LABEL' and 'SUBRIGHT' on the construct 'node' (Ehrig & Mahr 1985).

4.8. Meaning, Use and Consequences

The profound insights into the connections between *meaning* and *use*, *utility*, *purpose*, etc., provided by Wittgenstein's later writings, such as the remark 'Meaning, function, purpose, usefulness – interconnected concepts' from a manuscript published as *Last Writings on the Philosophy of Psychology*, Vol. I, serve as a remarkably intuitive basis on which a semantics for the mathematical language can be constructed. Not just another semantics but one that proves more convincing than many 'verificationistic' semantics because no canonical value or mathematical construction is said to have meaning in itself, assertability conditions do not suffice, and no communitarian notions such as games or dialogue need to be introduced. Accepting the explanation of the *consequences* one is allowed to draw from a proposition as the explanation of its *meaning*, the explanation of how it is to be *used*, one can easily accept that *meaning*, *use* and *consequences* are indeed interconnected concepts. Further to the connections between *meaning* and *use*, it seems useful to consider the *consequences* one is allowed to draw from a proposition as something directly related to its *meaning/use*. And indeed, Wittgenstein's references to the

connections between *meaning* and *consequences*, as well as between *use* and *consequences* are sometimes as explicit as his celebrated 'definition' of *meaning* as *use* given in the *Investigations*. The filling in of those two lesser known sides of the semantic triangle-like diagram shall perhaps bring the notion of use a little more 'down to earth', as it were, while demonstrating that if 'the use of a sign' is understood as 'the utility of the sign' then 'use' can be made more precise and more 'concrete' for the case of mathematical language. There are logical inference rules which can be taken as being the rules playing the part of explaining the utility of logical signs by showing what can be deduced from a proposition formed with that sign as its major one. To realise such a demonstration of the conclusions one has the conventional logical rules of *elimination*, *deduction*.

In this section I collect some of those explicit references made by Wittgenstein, setting up the appropriate grounds for a discussion on how an intuitive basis for the construction of a reasonably convincing semantics for the mathematical language can arise out of these connections *meaning/use/consequences*.

As a matter of fact, signs of Wittgenstein's attempt to make the connections between explaining the *meaning* of a proposition, demonstrating how to *use* it, and showing the *consequences* one is allowed to draw from it, can be found in many different presentations. There are many relevant remarks on the topic scattered among his writings, irrespective of phases, as we have seen from the previous sections above. In the later writings, however, the remarks of that nature become much more frequent, and significantly more incisive than those of the early phase. In a series of *Lectures on the Foundations of Mathematics* given in Cambridge in 1939, for example, he explains how the meaning of a proposition can be changed by changing the way it is used, by changing the consequences that can be drawn from it. The core of the discussion can be summarised in the following:

Wittgenstein: Yes; and let us take another example; the use of "all". "If all the chairs in this room were bought at Eaden Lilley's then this one was. $(x).fx$ entails fa." Suppose I ask, "Are you sure fa follows from $(x).fx$? Can we assume that it does not follow? What would go wrong if we did assume that?"

Wisdom: One reply which might be given is that it is impossible to make such an assumption.

Wittgenstein: Yes. But let us look into this, because such things as "Let us assume that $(x).fx$ does not entail fa" have been said. Now the reply you suggested did not mean that it is psychologically impossible to assume that; for if it did, one might say that although Wisdom cannot do it yet perhaps other people can.

In what way is it impossible to assume $(x).fx$ does not entail fa?

Wisdom: Isn't the assumption like saying "Couldn't we have a zebra without stripes?"

Wittgenstein: Yes. It would be said that the meaning of '$(x).fx$' had been changed.

What then would go wrong if someone assumes that $(x).fx$ does not entail fa? I would say that all I am assuming is a different use of "all", and there is *nothing* wrong in this.

If I stick to saying that the meaning is given by the use, then I cannot use an expression in a different way without changing the meaning. But it is then misleading to say, "The expression *must* have a different meaning if used differently." It is merely that it *has* a different meaning—the different use *is* the different meaning.

And if one says, "If one assumes fa does not follow from $(x).fx$, one must use $(x).fx$ in a different way"—we reply, in assuming this one *does* it in a different way.—But if we make this assumption, nothing goes wrong.

(Diamond (ed.) 1976, p. 192.)

Just from the passage quoted above a number of things can be safely said along the same lines on the connections between *meaning*, *use* and *consequences*. From this viewpoint, it is not unreasonable to say, for example, that not only the meaning of a proposition changes when the way it is used changes, when what can be deduced from it also changes, but that the latter *determines* the former. This view becomes much more contentious and the evidences for it much clearer and much sharper in Wittgenstein's very late writings, and that is what I shall attempt to demonstrate below.

What I do here in trying to relate those three concepts of *meaning*, *use* and *consequences* by means of Wittgenstein's writings is in effect an attempt to show that the semantical standpoint 'meaning as use' is not just feasibly realisable, as far as a basis for operational semantics of formalised languages, but it is also very natural for logic and computing science as a whole. In order to define a logical sign by means of computation rules what one does is essentially demonstrate what

can be obtained from the 'term' having the sign as its major one after *computing*, *eliminating* the logical sign. (E.g., my preferred example from *λ-Calculus*: the logical sign 'λ' and its β-rule of *computation*, *normalisation*.)

4.8.1. Meaning and Use

There has been a number of different readings of Wittgenstein's definition of meaning as given in:

> For a *large* class of cases—though not for all—in which we employ the word "meaning" it can be defined thus: the meaning of a word is its use in the language.
>
> (*Investigations*, Part I, §43, p. 20^e.)

Much has already been said on the different interpretations one can give to such a 'definition'. *Use* being such a vague concept as it is in ordinary language, enough room is left for distinct and perhaps uncongenial readings of Wittgenstein's semantical paradigm. Most of the possible readings discussed in the literature do lend a vague character to Wittgenstein's use-based semantical paradigm, especially because one is usually given the idea that when describing the use of a word or a sentence one has to describe the whole context within which it is embedded. And this sounds prohibitively complex for linguistic analysis through formalised description and reasoning mechanisms.

What is remarkable about Prof. Hintikka's *Game-Theoretical Semantics* is that a (semi-)formal account of Wittgenstein's 'the *use* of logical signs in the language-game' for the case of the language of logic is given and yet there is no additional difficulty with bringing in information about the 'context'. It actually makes the explanation of use of logical constants 'context-free' by stating in each case what can be deduced from the propositions with those logical signs as the major connective, no matter what context they are embedded in. And this is done, as we have seen from a brief discussion on *semantical games* above and will be seeing in a little more detailed manner in the next chapter, by means of explanations of what consequences one is allowed to draw from the corresponding propositions.

4.8.2. Meaning and Consequences

By contrast with the wide popularity enjoyed by the connections between *meaning* and *use* that Wittgenstein's (especially later) writings convey, the remarks on the existence of close links between the *meaning* of an assertion and the *consequences* one is allowed to draw from it are almost ignored by most of the relevant literature on Wittgenstein's semantic theory. As I have demonstrated above, one can find examples of such remarks even in very early writings (cf. the letter to Russel dated 1.7.1912, quoted previously), although in most cases the explanation of *meaning* is related to an operation of *following from* and the word *consequences* is not explicitly used.

In a typescript dictated in late autumn 1947 (TS 229), whose underlying manuscripts stem from the period between May 10th 1946 and October 11th 1947, published as *Remarks on the Philosophy of Psychology*, Vol. I, one can find a remark relating what an utterance means and what one is permitted to draw from it.

> Do we also speak of a '*feeling*' of thinking in the head? Wouldn't this be like the 'feeling of meaning'?
>
> Again: Suppose someone who wouldn't have this feeling. Is he unable to think?
>
> Indeed, someone who does philosophy or psychology will perhaps say "*I* feel that I think in my head". But what that means he won't be able to say. For he will not be able to say *what* kind of feeling that is; but merely to use the expression that he 'feels'; as if he were saying "*I* feel this stitch *here*". Thus he is unaware that it remains to be investigated what his expression "I feel" means here, that is to say: what consequences we are permitted to draw from this utterance. Whether we may draw the same ones as we would from the utterance "I feel a stitch here".
>
> (*RPP I*, §350, p. 69[e].)

(The emphasis is mine.)

From another typescript dictated in early autumn 1948 (TS 232), bearing remarks which stem from manuscripts written between November 1947 and August 25th 1948, also published as *Remarks on the Philosophy of Psychology*, but this time Vol. II, one can see remarkably direct and explicit references to the connections between the meaning of a statement and what can be deduced from it.

For the question is not, 'What am I doing when . . .?' (for this could only be a psychological question) – but rather, 'What meaning does the statement have, what can be deduced from it, *what consequences does it have*?'

(*RPP II*, §38, p. 8e.)

(The *emphasis* is the Translators'.)

4.8.3. Use and Consequences

Similarly to the previous case, it does not seem to be public knowledge the attempt Wittgenstein has made in demonstrating the close links between the *use* of a proposition and the *consequences* one can draw from it. Nevertheless, one can show that it is a fact that such an attempt is made very clear in later writings, by contrast with the case of *meaning* and *consequences*, which although less emphasised in the literature, yet can be traced back in early remarks as we have seen from sections above.

From a series of *Lectures on Religious Belief*, given in Cambridge sometime around the summer of 1938, whose notes by Y. Smythies are published in *Lectures and Conversations on Aesthetics, Psychology and Religious Belief*, one can choose a passage where the links between *use* and *consequences* are made very explicit:

Yes, this might be a disagreement—*if he himself were to use the word in a way in which I did not expect, or were to draw conclusions I did not expect him to draw.* (...)

(*Lectures and Conversations on APRB*, p. 71.)

(The *emphasis* is mine.)

In the first volume of the *Last Writings on the Philosophy of Psychology*, which is not based on a typescript but on manuscript writings dating from the period between 22 October 1948 and 22 March 1949 except for the last remark dated 20 May (second half of MS 137 and the whole of MS 138), one can find examples of very explicit references to the connections between the use of words and the consequences of the corresponding utterance. Presented sometimes in a positive way such as in:

What are you telling me when you *use* the words . . .? What can I do with this utterance? *What consequences does it have?*

(*Last Writings I*, §624, p. 80e.)

(The *emphasis* is mine.)

or in a negative manner such as in:

The report "The word . . . was crammed full of its meaning" *is used quite differently, has quite different consequences*, from "It had the meaning . . .".

(*Last Writings I*, §785, p. 100e.)

(The *emphasis* is mine.)

his remarks on the connections between the use of a sentence and the consequences one is allowed to draw from it constitute a key ingredient in his investigations on the concept of meaning.

Considering that his main focus of attention in the later phase is indeed on the 'logic of ordinary language', rather than on the technicalities of Frege-Russell's logical calculus, it is not quite surprising that Wittgenstein's very late writings on meaning and use, strikes one as so intuitively appealing. By contrast, in early as well as not-so-late writings, in spite of a clear attempt to persuade the reader that 'meaning' is better understood as 'use', generally speaking there is no clear-cut, incisive account of what should be taken as the logical counterpart of the latter. There are, however, a few references to meaning-use-consequences scattered amongst early writings. Unfortunately, the remaining general picture of the transitional period suggests a certain hesitation which is revealed quite clearly when the rôle of proofs in mathematics is analysed. In the writings of the early period of his later phase, documented in the notebooks containing remarks mainly on the foundations of mathematics now published as *Philosophical Remarks* and *Philosophical Grammar*, there is a clear shift in the focus of attention from truth-values and truth-functions which were characteristic of the *Tractatus*, to mathematical proofs. The concept of mathematical proof is thoroughly investigated, and in some passages it is even placed

in a rather privileged position with respect to the semantics of the mathematical language. One of the lessons taught by those 'transitional' writings seems to be that (using Wittgenstein's own words) 'when one wants to know the meaning, the sense, of a proposition, one has to look at what its proof proves'.

It does not take too long, however, for a revision of the position on the connections between meaning and proofs to be made, and it is in the *Remarks on the Foundations of Mathematics*, written between 1941–44 (more precision with respect to dates is not provided by the editors), that the revision is made explicit:

> I once said: 'If you want to know what a mathematical proposition says, look at what its proof proves'[6]. Now is there not both truth and falsehood in this? For is the sense, the point, of a mathematical proposition really clear as soon as we can follow the proof?
>
> (*RFM*, Part VII, §10, p. 367.)

A great number of remarks which reveal a much less explicit statement of the revision made with respect to the connections between the meaning of a proposition and its proof appear in later stages of the later period. One of them, from the same *Remarks on the Foundations of Mathematics* and written around 1943/44, says:

> The proof of a proposition certainly does not mention, certainly does not describe, the whole system of calculation that stands behind the proposition and gives it its sense.
>
> (*RFM*, Part VI, §11, p. 313.)

which reveals his concern on the insufficiency of proofs with respect to meaning. The apparently holistic view is given more convincing force as to its feasibility to the formulation of a theory of meaning for the language of mathematics by his suggestions on the connections between the meaning of a proposition and the consequences one can draw from it.

Unfortunately, his acknowledgement of the inappropriateness of using mathematical proofs and assertability conditions as semantical notions does not seem to have been taken into consideration by some 'verificationist' theories of meaning which still look for a conceptual basis in Wittgenstein's semantic theory and suggest

[6]Cf. *Philosophical Grammar*, p. 369; *Philosophical Remarks*, pp. 183–4.

that 'meaning is use' is their underlying semantical paradigm. Wittgenstein's clear emphasis on the *deductions* that can be made from a proposition as tools demonstrating its meaning does not appear to have been given the attention it would seem to deserve by those who advocate a use-based semantic theory but still insist on the primacy of assertability conditions. Neither there seems to be much concern about this particular aspect of Wittgenstein's very late writings in most of the literature on Wittgenstein's philosophy, to the best of my knowledge.

4.9. Language-Games, Context of Utterances and the Consequences

When explaining the meaning of a sentence, one can hardly avoid referring to the context under which the sentence is being considered. In his use-based semantic teachings, L. Wittgenstein consistently insists on the crucial rôle played by the context of utterances, and that is very clear from the very conceptual basis provided by the notion of 'language-games'. Whenever it is relevant to ask what a certain sentence really means, Wittgenstein reminds the reader of the crucial part played by the context under which one is asserting the sentence:

> We ask "What does 'I am frightened' really mean, what am I referring to when I say it?" And of course we find no answer, or one that is inadequate.
>
> The question is: "In what sort of context does it occur?"

(*Investigations*, Part II, section ix, p. 188[e].)

Nevertheless, he does not seem to neglect the intentional ingredient which is inherent to a conception of meaning based on its identification with use. By attempting to demonstrate that the consequences one can draw from a proposition constitute one of the key ingredients of Wittgenstein's definition of meaning as use, I endeavour to show that it is not unreasonable to say (with Wittgenstein) that the meaning of a sentence depends on but is not *ultimately* given by the circumstances under which it can be correctly asserted. Taking the explanation of the what consequences one is allowed to draw from a proposition as its ultimate intentional aspect, as an explanation of how the proposition functions in the circumstances,

one remains congenial with Wittgenstein's concerns as to the semantical rôle of the consequences, and yet taking into account the important part played by the circumstances. A clear evidence of Wittgenstein's concern with the functionality aspect of the meaning of signs is present in Prof. P. Geach's notes of Wittgenstein's *Lectures on Philosophical Psychology 1946-47*:

> *Malcolm*: You say "a voluntary action" gets its sense from circumstances. And you contrast 'voluntary action' with 'feeling' as different concepts. Therefore one would expect that "a pain" should *not* get its sense from circumstances; whereas *you* say that "a pain" *also* gets its sense from circumstances. So where is the contrast between, e.g. pain and voluntary action?
>
> *Wittgenstein*: All signals get their sense from circumstances, it is true; but the way they function in the circumstances is different. Thus, impressions have a place and emotions have not; and to say an action is voluntary is not like uttering an impression *or* an emotion.
>
> (Geach (ed.) 1988, p. 76.)

(The emphasis is mine.)

Despite constituting a necessary ingredient in a semantic analysis, a description of the context of utterance must remain in the extensional territory, whereas the clarification of the consequences that can be drawn from the sentence appears to play the ultimate (intentional) rôle in the meaning explanation. It seems equally reasonable to say that, e.g., no matter how accurate the description of a context under which one can correctly assert a sentence may be, one will not make its meaning clear until the aspect of intentionality, which can hardly be said to be part of the context, is uncovered.

> "I know that that's a tree." Why does it strike me as if I did not understand the sentence? though it is after all an extremely simple sentence of the most ordinary kind? It is as if I could not focus my mind on any meaning. Simply because I don't look for the focus where the meaning is. As soon as I think of an everyday use of the sentence instead of a philosophical one, its meaning becomes clear and ordinary.
>
> Just as the words "I am here" have a meaning only in certain contexts, and not when I say them to someone who is sitting in front of me and sees me clearly,—and not because they are superfluous, but because their meaning is not *determined* by the situation, yet stands in need of such determination.

(*On Certainty*, §§347–8, p. 44e.)

(The emphasis is mine.)

Attempts have been made to develop semantic theories based on states of affairs, situations, etc., where the basic ingredients include more complex structures than the ones used in model-theoretic truth-values based semantic theories. Such ingredients in most cases find their ontological status justified in the framework of 'possible worlds'-semantics perhaps first introduced by Prof. S. Kripke's semantical studies of modalities. The important lesson that we seem to be offered by Wittgenstein's investigations into the connections meaning-use and the separation of the intentional (functional) aspect from the aspect concerning the context (state of affair, circumstance, etc.) is that more than simply devices to describe contexts must be incorporated in any workable semantic theory. And this seems to be justified by Wittgenstein in the form of a warning against our thinking that it must be possible to find different states of affairs corresponding to meaningful sentences which apparently have the same truth-conditions but have different meanings.

(...) In any case the true state of affairs can only be seen when we look into the detail of the usage of our expressions.—And it is clear that *if in our present case we should decide to say that the two expressions have different meanings, we shall certainly not be able to say that the difference is that the fact which makes the second sentence true is a different one from the fact which makes the first sentence true.*

We are justified in saying that the sentence "He can continue . . ." has a different meaning from this: "He knows the formula". *But we mustn't imagine that we can find a particular state of affairs 'which the first sentence refers to'*, as it were on a plane above that on which the special occurrences (like knowing the formula, imagining certain further terms, etc.) take place.

(*Brown Book*, Part I, §64, p. 115.)

(The *emphasis* is mine.)

Based on a distinction between sense and force, perhaps first made by Frege, and on the connections between meaning and proofs, some verificationist theories of meaning often take the *assertability conditions* as the central aspect which determines the meaning of a proposition. It seems to have been Wittgenstein who

first insisted on the rôle of *verification* in the settlement of sound foundations of objective knowledge. Incidentally, he also referred to the *sense* of a proposition as being determined by the way it can be verified. However, Wittgenstein's point of view on the rôle of *verification* for the particular case of logical propositions seems to have been much different from the general 'dogma' of the so-called 'verificationist theories of meaning', simply because it appears to prescribe that the process of *verification* should take primarily into account the *conclusions* one is allowed to draw from the proposition, or simply, the *intention*, the *utility* of making the assertion. And this seems to suggest the fundamental rôle of the *consequences* one can draw from a particular assertion as a key aspect of its meaning, in addition to the equally as important *assertability conditions* taken to be the sole aspect of the meaning of an assertion by most of the 'verificationist theories of meaning'. Prof. Prawitz's semantical investigations of a conceptual basis for a theory of meaning for the language of mathematics based on the connections between meaning and proofs clearly acknowledges that the two aspects of a proposition, namely sense and force (following Frege and Dummett), are two independent aspects, but concludes that the central aspect of its meaning must reside in the sense aspect — the *conditions for assertion* — (Prawitz 1977, pp. 38–9.).

A theory of meaning constructed on the basis of *assertability conditions* as the central aspect of the meaning of a sentence can hardly be said to incorporate any intentional ingredient. As it is clearly acknowledged in the passage mentioned above, by knowing the conditions for asserting a particular sentence one can hardly know what is called by Prof. Dummett the commitments made by asserting it, and it is this very aspect of 'commitments' which plays a central semantic rôle in what Prof. Dummett calls a 'falsificationist' theory of meaning (Dummett 1976, p. 126). And 'reflection should make us admit' that a falsificationist theory of meaning would be better than a verificationist *assertability conditions*-based theory of meaning, according to Prof. Dummett (Ibid., p. 137). Now, as I have pointed out previously, it is reasonable to suppose that the commitments one makes by asserting a sentence are inseparable from the consequences he/she allows to be taken from it. I have

attempted in de Queiroz 1989b to demonstrate that verificationist theories of meaning based on *assertability conditions*, while escaping from the denotational view of language, must remain essentially extensional. Wittgenstein's 'theory of meaning', which can be said to be verificationist but not *assertability conditions*-based, is neither denotational nor extensional given the strong intentionality element inherent to the connections between the meaning/use of a proposition and the consequences one is allowed to draw from it.

It appears to be in the *Blue Book*, perhaps where the view on 'words are tools' (p. 67) is made explicit for the first time, that the crucial rôle played by the context of utterances in a semantical theory is mostly emphasised over and over again. In some passages, however, the intentionality aspect is also brought into the focus of attention, although the general picture still reflects a clear emphasis on the importance of looking at the circumstances under which the utterances are made.

Wittgenstein's view on the insufficiency of context descriptions in making meaning explicit takes a great variety of forms. In the *Investigations*, where the use of the device of saying 'the question is: under what circumstances does one assert the sentence?' is often, if emphatic, he is already trying to warn us against thinking that there can be a totality of conditions (state of affairs) which gives an utterance its meaning:

> But did "Now I can go on" in case (151) mean the same as "Now the formula has occurred to me" or something different? We may say that, in those circumstances, the two sentences have the same sense, achieve the same thing. But also that *in general* these two sentences do not have the same sense. We so say: "Now I can go on, I mean I know the formula", as we say "I can walk, I mean I have time"; but also "I can walk, I mean I am already strong enough"; or: "I can walk, as far as the state of my legs is concerned", that is, when we are contrasting *this* condition for walking with others. But here we must be on our guard against thinking that there is some *totality* of conditions corresponding to the nature of each case (e.g. for a person's walking) so that, as it were, he *could not but* walk if they were all fulfilled.
>
> (*Investigations*, Part I, §183, p. 74[e].)

(The emphasis is mine.)

The intentionality aspect is not ultimately uncovered by an increase on the amount of information about the context. And this is what characterises the intention as the final and ultimate interpretation. In a given social practice, neither need one to define for each sentence what can be deduced from it, not need the hearer to guess, the right conclusions are readily taken due to the ability that both have of playing the language-game. Wittgenstein comes to this specific point still at Part I of the *Investigations*:

> "I was going to say . . . . ."—You remember various details. But not even all of them together shew your intention. It is as if a snapshot of a scene had been taken, but only a few scattered details of it were to be seen: here a hand, there a bit of a face, or a hat—the rest is dark. And now it is as if we knew quite certainly what the whole picture represented. As if I could read the darkness.
>
> These 'details' are not irrelevant in the sense in which other circumstances which I can remember equally well are irrelevant. But if I tell someone "For a moment I was going to say . . . . ." he does not learn those details from this, nor need he guess them. He need not know, for instance, that I had already opened my mouth to speak. But he *can* 'fill out the picture' in this way. (And this capacity is part of understanding what I tell him.)
>
> (*Investigations*, Part I, §§635–6, pp. 164^{e}–5^{e}.)

(The emphasis is mine.)

In a language-game and social practice, intentions and expectations participate in a 'give-and-take': an utterance and what will expectedly be drawn from it are linke by the very social practices. When hearing an utterance as part of a certain language-game one does not need to be given a description of details of the circumstance under which it is asserted, nor need he/she guess it. That he/she can fill out the picture in such and such a way is part of understanding what is being said.

4.10. Concluding Remarks

By an attempt to fill in the two lesser known sides of the semantic triangle-like diagram with vertices in *meaning*, *use* and *consequences* I have demonstrated how one can bring Wittgenstein's celebrated 'meaning-is-use' semantical paradigm more

'down to earth' by demonstrating that if 'the use of a sign' is understood as 'the utility of the sign' then 'use' can be made more precise and more 'concrete' for the case of formalised languages. In the context of formal logic, for example, there are logical inference rules which can be taken as being the rules playing the part of explaining the utility of logical signs by showing what can be deduced from a proposition formed with that sign as its major one. To realise such a demonstration of the conclusions one has the conventional logical rules of *elimination*, *deduction*, or even, rules of *reduction*, *normalisation*, as is the case for my own type theoretic account of meaning for the language of mathematics.

In the next chapter I will be presenting the *'Meaning-As-USE' Type Theory* and its underlying 'semantics of use', demonstrating why it should be seen as a good starting point for the construction of such an intuitive basis. With respect to Computing Science, I attempt to show that a number of ideas arising out of such a semantical reformulation of *Intuitionistic Type Theory* into the Type Theory presented formally in the Appendix of this dissertation find themselves useful in the understanding of some of the issues dealt with in theories of specification of data types, including implementation and proof of correctness, completeness of axioms, and error-handling. In the further chapters that follow I shall demonstrate how such *'Meaning-As-USE' Type Theory* can be used to explain some of the fundamental concepts of the logical foundations of computation.

5 *Formalising the Notion of Use/Utility*

In his lecture notes on the semantics of his *Intuitionistic Type Theory* Prof. Martin-Löf seems to provide an account of meaning for his proof-theoretically presented 'language' of constructive mathematics that lacks the force of a truly convincing new perspective on the connections between proof theory and meaning. Staying within Gentzen's principles, which say that the meaning of a proposition is determined by its *conditions for assertion*, he says:

> (...) The introduction rules say what are the *canonical* elements (and equal canonical elements) of the set, *thus giving its meaning*. (...)
>
> (Martin-Löf 1984, p. 24.)

(The *emphasis* is mine.)

Now, what aspect does such an *introduction* rule of (Gentzen's) proof theory cover with respect to the meaning of a proposition/type in a type-theoretic framework such as Prof. Martin-Löf's *Intuitionistic Type Theory*? I should like to say that it covers *one* and *only one* of its fundamental aspects, which corresponds roughly to the rule of sentence/proposition formation according to Wittgenstein's later account of propositions, as given in:

(...) And what a proposition is is in *one* sense determined by *the rules of sentence formation* (in English, for example) and in another sense by the use of the sign in the language-game.

(*Investigations*, Part I, §136, p. 53^e.)

(The *emphasis* is mine.)

Following Wittgenstein's account of propositions, the *introduction* rules of Gentzen's systems, which demonstrate the conditions under which one can correctly assert the sentence/proposition, do not give the proposition its meaning. And this is because the other equally fundamental aspect characterising the proposition, namely 'the use of the sign in the language-game', is not covered by the so-called rules of *assertability conditions*.

5.1. Preamble

The semantical standpoint I take in this report is essentially that which motivated the development of my 'semantics of use' where *meaning* is not taken as an object, (numerical, canonical) value or entity. It seems to me that the later Wittgenstein's semantic theory opens a definitely new perspective on the semantics of the mathematical language by making use of useful devices such as: the shift from looking at the meaning of a word to looking at the actual explanation of its meaning; placing the emphasis on what *consequences* a proposition has as a determining factor of its meaning; looking at the purpose, *utility*, usefulness, etc., of an assertion as a key to its meaning. And indeed, the profound insights that Wittgenstein's writings seem to provide as to the connections between *meaning* and *use*, *utility* (*usefulness*), *purpose*, etc., such as a remark contained in later manuscripts written between 22 October 1948 and 22 March 1949 recently published as *Last Writings on the Philosophy of Psychology*, Vol. I has served as comfortable grounds for a closer look on issues regarding semantic theories as well as theories of meaning for the mathematical language from the point of view of computing science, where meaning and utility are indeed interconnected concepts. The remark says explicitly:

Bedeutung, Funktion, Zweck, Nutzen — zusammenhängende Begriffe.

which has been translated as

Meaning, function, purpose, usefulness — interconnected concepts.

(*Last Writings I*, §291, p. 41^e.)

There have been various attempts at providing logical counterparts to (the later) Wittgenstein's semantic theory, but none seems to have succeeded in offering a more general account which could accommodate the apparently distinct teachings on the subject, namely, meaning is determined by use, the meaning of an assertion and its consequences are closely related, the meaning of a proposition is determined by its method of verification, the concept of meaning is better understood by first looking at the notion of explanation of meaning, the concepts of meaning and utility/usefulness (Ger. *Nutzen*) are closely interconnected, etc.. The latter seems to be such an important ingredient of Wittgenstein's philosophical studies that one can find clear signs and definite support even in the early writings, such as, e.g., in the following remark from the *Tractatus*:

Occam's maxim is, of course, not an arbitrary rule, not one that is justified by its success in practice: its point is that *unnecessary* units in a sign-language mean nothing.

Signs that serve *one* purpose are logically equivalent, and signs that serve *none* are logically meaningless.

(*Tractatus*, §5.47321, p. 48.)

or in a, perhaps less incisive, remark from the *Notebooks*:

Einführung des Zeichens "0" um die Dezimalnotation möglich zu machen: Die logische Bedeutung dieses Vorgehens.

(*Notebooks*, dated 16.11.14, p. 31.)

translated as:

The introduction of the sign "0" in order to make the decimal notation possible: the logical significance of this procedure.

And it is not difficult to verify that one can still find traces of the usual 'object and designation'-oriented account of meaning, even in some of the more recent verificationistic and 'anti-interpretation' theories of meaning. Despite being germane to what one could call the 'negative' side of semantical standpoint established by (the later) Wittgenstein which says that interpretations by themselves do not settle meaning[1], and in spite of also[2] making use of the concept of verification as a key aspect to understand meaning, some of those verificationistic theories of meaning are not congenial with Wittgenstein's semantical viewpoint concerning the principle that meaning is determined by use. It might be worth recalling that Wittgenstein was perhaps the first to point out the links between the meaning of a proposition and its method of verification. He was also perhaps the first to seriously challenge the principles underlying interpretations-based semantic theories.

In what follows I shall attempt to pursue the claim that, at least for the case of the 'languages' used in computing science (specification language, programming language, etc.), meaning and utility (usefulness) are indeed interconnected concepts by demonstrating that a certain type-theoretic approach to the formalisation of constructive mathematics provides us with a satisfactory apparatus to deal with propositions, types and 'computational' meaning, accommodating the various aforementioned 'clothings' of (the later) Wittgenstein's semantic 'theory'.

By means of a reasonably extensive explanation of the new ideas and concepts

[1]' "But how can a rule shew me what I have to do at *this* point? Whatever I do is, on some interpretation in accord with the rule." — That is not what we ought to say, but rather: any interpretation still hangs in the air along with what it interprets, and cannot give it any support. Interpretations by themselves do not determine meaning.', §198, p. 80[e] of the *Investigations*, Part I. (the emphasis is mine.)

[2]Following Wittgenstein's 'The sense of a proposition is the method of its verification. A method of verification is not the means of establishing the truth of a proposition; it is the very sense of a proposition. In order to understand a proposition, you need to know the method of its verification. To specify it is to specify the sense of a proposition. You cannot look for a method of verification. A proposition can only say what is established by the method of its verification.', p. 227 of F. Waismann's *Ludwig Wittgenstein and the Vienna Circle*.

introduced by Prof. P. Martin-Löf in his development of 'an intuitionistic theory of types', which since its first 'public' appearance at the *Logic Colloquium '73* (Martin-Löf 1975a) has been drawing the most eager attention of logicians, mathematicians and computer scientists alike, simultaneous with an exposition of my own views on the subject, I shall spend most of the present chapter attempting to convince the reader that the type-theoretic approach to describing logical systems can be of great utility in providing appropriate means to meet the semantical demands of many areas of study directly or indirectly relevant to computing science: (1) the formalisation of constructive mathematics; (2) the investigations into the notion of logical *judgement*; (3) the clarification of the semantical rôle of the meta-mathematical device of *normalisation*, which since G. Gentzen's formulation of the 'Inversion Principle' (Gentzen 1935), and A. Church's formalisation of the notion of 'function application' with his λ-*Calculus*(Church 1936, 1941), has been widely used as an 'operational' semantic device; (4) the search for an understanding of the nature of 'types' as conceived by computer scientists, as opposed to 'sets' as understood by (classical) mathematicians; (5) a new insight on the problem of formulating the meaning of logical connectives through the help of a combination of the propositions-are-types paradigm with the view on *normalisation* as a semantical device; and last, but not least, (6) the significance of Prof. Hintikka's *Game-Theoretical Semantics* (Hintikka 1983) to the understanding of not just general aspects of ordinary language but also of a 'semantics of computation' solely by means of the laws of formal logic.

If nothing else, I expect to have demonstrated by the end of the chapter that the so-called 'sense' view of logic, as characterised by the semantics based on *assertability conditions* and the connections between meaning and proofs, following the tradition of Gentzen, Heyting, Prawitz and others, is inadequate for a semantics of computation. I also hope to have made clear that the 'use' view, as characterised by Wittgenstein's use-based semantic theory and the connections between meaning, use, utility and consequences, is not only 'natural' as far as our intuitions on how ordinary language works, but indeed very useful for a 'workable' semantics for

computing science.[3]

5.2. Introduction

[3]From the notes taken by Mr M. Ryan of an informal talk given as part of the series 'Foundations Workshop' by Mr M. Sadler (to whom I feel grateful for his willingness to enter in many interesting and illuminating informal discussions), 22 Feb '89, one can see that although being recognisably more intuitive and more appropriate to a computer scientist view of logic than the more usual denotational approach, Wittgenstein's use-based semantic theory is sometimes regarded as a 'blow' in the hope of being able to analyse formalised languages such as logics, programming languages, etc., because it shows that one cannot neglect the 'context' of use of expressions, sentences, propositions, and moreover, one must not overlook the intentional ingredient which is brought in by the connections between meaning and use. And this view appears to leave one with the feeling that the 'sense' (Gentzen's) view should be considered more appropriate because it only involves 'syntactic relations' and no consideration of the 'contexts of utterance' is required. Take, e.g.:

'2 **Views about logic**

- **Denotational** (Frege, Carnap, Cresswell, Montague)

  This is the traditional view of logic, much popularised by the Scott/Strachey approach to programming semantics. Idea: interpret language in some valuation system ("domain").

- **Sense** (Gentzen, Heyting)

  Here, language is understood by the syntactically defined relations on it (turnstiles). We are interested in equivalences and pre- and partial orders.

- **Use** (Wittgenstein)

  On this view, language is really unanalysable. What gives it meaning is the *context of utterance*.

For the purpose of **logic engineering** (in particular, for using logic for specifications), the **sense** view is better than the **use** view, which in turn is better than the **denotational** view.'

(The emphasis is mine.)

As we have seen from the previous chapter, the use-based semantics for formalised languages, such as, e.g., Prof. J. Hintikka's *Game-Theoretical Semantics* and my own type-theoretic *Semantics of Use* described in this report, on the one hand does not have to appeal to 'non-syntactic' relations, therefore on this view a (formal) language is also understood by syntactically defined relations on it.

After D. Hilbert's pioneering attempts, G. Gentzen was certainly one of the first to put into effective terms the analysis of issues concerning the foundations of mathematics in a precise and purely syntactic way. Procedures like proving the consistency of elementary number theory using only syntactical arguments were straightforward results of his 'theory of proofs'. The key idea was that of using the notion of logical steps, *introduction* and *elimination* steps, for a suitable formal representation of mathematical proofs as objects of study in themselves. Most importantly, he also introduced the notion of *normalisation* representing the act of showing how 'redundant' steps could be removed from a logical proof without any loss. By performing an *introduction* step for a certain logical connective, and immediately afterwards applying an *elimination* step, one would be making 'redundant' steps which could be removed from the formal representation of the proof without any loss. Applying these ideas to extensions of his *calculus of natural deduction*, he arrived at a precise formulation of what is called the *cut-elimination* procedure. Being a procedure which operates on the objects of the theory — proofs in a certain logic —, in order to allow proving properties of the particular theory — the presentation of a particular logic — such as normalisation, strong normalisation, consistency, etc., it is easily understood as a 'meta-level' procedure.

Starting from Gentzen's ideas, Prof. Prawitz attempts to further explore the *calculi of natural deduction* as workable tools to define/present logics. In his thorough studies on Natural Deduction (Prawitz 1965, Prawitz 1971), he develops *reduction* rules to perform *normalisation* on proofs constructed from natural deduction inference rules. Recalling that the relationship between *introduction* and *elimination* rules of natural deduction systems is expressed by what is called 'the logical

On the other hand, it does take into account two fundamental aspects of language understanding, namely context-dependent meaning and intentionality of sentences, by prescribing that the meaning of logical signs must be stated not just by the conditions for assertion but also via the consequences one can draw from the corresponding proposition, given that the statement of the context they might be inserted in is described in the conditions for assertion aspect, therefore showing a way to simplify the analysis of what appears to be 'unanalysable'.

principle of inversion'[4], Prof. Prawitz defines procedures which operate on proof-trees, and which he initially calls reduction *steps*. Only in further developments of those normalisation procedures for natural deduction presentations of logics are those 'meta-level' procedures referred to as *rules*. However, due to the very nature of those particular presentation systems, where only *introduction* and *elimination* rules take part, the *reduction* rules remained at the 'meta-level', until a certain formalisation of the basic principles of constructive mathematics was presented via natural deduction rules from a new perspective emerging from the identification of propositions with types, and the *reduction* rules were incorporated as 'part of the system', as it were[5]. This was exactly Prof. P. Martin-Löf's *Intuitionistic Type Theory* (*ITT*) which has since had so much attention from many different perspectives on the rôle and the nature of language on the foundations of logic and mathematics, as well as on the logical basis of the 'science of computing'. As an example of such a notion of Prof. Prawitz's *reduction rule*, for the case of conjunction the *reductions* are given as:

$\wedge$-*reduction*

$$\dfrac{\dfrac{\dfrac{\Sigma_1}{A_1}\quad\dfrac{\Sigma_2}{A_2}}{A_1 \wedge A_2}}{A_1} \quad \twoheadrightarrow \quad \dfrac{\Sigma_1}{A_1}$$

and

$$\dfrac{\dfrac{\dfrac{\Sigma_1}{A_1}\quad\dfrac{\Sigma_2}{A_2}}{A_1 \wedge A_2}}{A_2} \quad \twoheadrightarrow \quad \dfrac{\Sigma_2}{A_2}$$

[4]'The inversion principle says in effect that nothing is "gained" by inferring a formula through introduction for use as a major premiss in an elimination.', Prawitz 1965, p. 33.

[5]'In the interpretation of sets as propositions, the formation rules are used to form propositions, introduction and elimination are like those of Gentzen, and the equality rules correspond to the reduction rules of Prawitz.', Martin-Löf 1984, p. 24.

(where 'Σ_i' 's represent proof-trees, and '$\twoheadrightarrow$' means 'reduces to' indicating that the proof-tree to the left of '$\twoheadrightarrow$' can be brought to a *normal* form like the proof-tree to the right of '$\twoheadrightarrow$') which are readily seen as *meta*-devices. Due to the identification of propositions with types, which leaves the counterpart of the usual *elimination* rules in need for further clarification, such as, e.g., Prof. Martin-Löf's:

Π-*elimination*

$$\frac{a \in \mathbf{A} \quad c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{APPLY}(c, a) \in \mathbf{B}(a)}$$

('APPLY' here standing in need of clarification) those meta-level rules of *normalisation* appear 'internalised' in Prof. Martin-Löf's *equality* rules of *Intuitionistic Type Theory* in order to make clear how the 'eliminatory' operators work, such as 'APPLY' in the Π case. A more detailed account of such identification of propositions with types and the rôle of such normalisation rules are explained in one of the sections that follow.

In fact, despite the clear emphasis on the *assertability conditions* (*introduction* rules) stressed by the proof-theoretic semantics of Gentzen and Prawitz, it is worth mentioning that the actual 'internalisation' of *reduction* rules appears to have been first explored as a semantical notion within proof-theoretic formalisations of intuitionistic mathematics via Gödel's seminal ideas on the so-called 'functional (Dialectica) interpretation' (Gödel 1958) by Prof. W. Tait in his development of an 'intentional' counterpart to Gödel's *T-systems* described in Tait 1967[6]. Moreover, by following Prof. Tait's seminal ideas Prof. Girard's has presented what seem to have been, together with Prof. Martin-Löf's *Intuitionistic Type Theory*, a pioneering type-theoretic framework for the formalisation of intuitionistic mathematics taking

[6]This has been pointed out by Prof. Martin-Löf in the opening lines of Martin-Löf 1971b: 'A semantical notion, specially invented for the purpose of proving normal form theorems, is Tait's 1967 notion of computability (or convertibility as he says).'

reduction (*normalisation*, *conversion*, etc.) as the central semantical notion: the *Y-* and *F-systems* described in Girard 1971.[7]

Now, what good can such an 'internalisation' of *normalisation* rules do for a better understanding of the meaning of logical signs? In what ways can it be explored and how does it relate to Wittgenstein's 'the use of the sign in the language-games'? With a brief explanation of *Intuitionistic Type Theory*, departing from its original formulation which has essentially an *assertability conditions*-based semantical explanation, but from my own perspective on the relation rules-meaning of such a particular presentation of constructive mathematics, I hope to make clear some of these important issues. More importantly, I hope to be able to explain why my own *'Meaning-As-USE' Type Theory* (*MAUSE*), which emerged from a semantical reformulation of Prof. Martin-Löf's *Type Theory* as an attempt at recovering the place of *normalisation* as a semantical device, can be seen as a more convincing theory of meaning for the language of mathematics. Roughly speaking, it would seem fair to say that it convinces better simply because it offers itself as a reasonable attempt to formalise the notion of *utility*, and put it into its proper semantical rôle.

At this point, an experienced proof-theorist who is used to prove consistency of logics presented via proof rules by making use of the canonical argument that 'the formal system is normalisable', might be left wondering whether and how the 'internalisation' (from 'meta'-level to 'object'-level) of normalisation rules would af-

---

[7]To be really fair, one would have to recall that already in 1893 Frege developed in his *Grundegesetze I* what can be seen as the early origins of the notion of *normalisation* when showing techniques for defining functions by an 'introductory' operator of abstraction producing the '*Werthverlauf*' expression (Frege 1893, §3, p. 7, translated as *course-of-values* in Furth (ed.) 1964, §3, p. 36), e.g., '$\acute{\varepsilon}f(\varepsilon)$', and the effect of its corresponding 'eliminatory' operator '$\cap$' on a 'course-of-values' expression (Frege 1893, §34, p. 52ff, translated in Furth (ed.) 1964, §34, p. 92):

'(...) it is a matter only of designating the value of the function $\Phi(\xi)$ for the argument Δ, i.e., $\Phi(\Delta)$, by means of "Δ" and "$\acute{\varepsilon}f(\varepsilon)$". I do so in this way:

"$\Delta \cap \acute{\varepsilon}f(\varepsilon)$"

which is to mean the same as "$\Phi(\Delta)$".'

(Furth (ed.) 1964, p. 92.)

fect such a universal procedure. The answer is very simple, and the witness very well known: in Church's λ-Calculus one has the 'conversion' ('reduction', 'normalisation', etc.) rule in the object-level (Church 1941, Chapter II, §6. CONVERSION, p. 12.), and yet proofs of consistency, existence of normal forms, etc., making use of the β-rule of normalisation are commonplace in mathematical studies of Church's calculus. To acknowledge the semantical rôle of *normalisation* rules is not by any means to invalidate or to ignore the part they have to play in proofs of consistency, confluence, etc.. By extending the work of A. Church of formalising mathematical concepts through explicit rules of operation, from the single notion of computable function to a much broader context of logics, types and theories Prof. Martin-Löf has also 'brought in' the rules of *normalisation* from the 'meta'-level to the 'object'-level. Again, similarly to Church, he also makes use of such semantical rules of *normalisation* in proving essential meta-mathematical properties of the formal system such as 'Normalization theorem', 'uniqueness of normal form', 'Church-Rosser property' and 'decidability of the convertibility relation' (**theorem** 3.3, and PROPOSITIONS 3.6, 3.7 and 3.8, respectively, p. 101 and p. 114 of Martin-Löf 1975a).

It seems relevant to emphasise that when I say that the *normalisation* rules have a fundamental semantic rôle to play in the meaning theory supporting what I call *'Meaning-As-USE' Type Theory*, I refer to the rôle they play in terms of explaining the meaning of the actual logical connective. It appears fair to say that the rules of *normalisation* are in some sense formalising the explanation of the *consequences* one is allowed to draw from the corresponding proposition, and for that reason I say that they serve as a semantical device which I take to be a formal counterpart to L. Wittgenstein's informal teachings on the concepts of meaning and proposition, namely 'the meaning of a word is its use in the language' and 'what a proposition is is in one sense determined by the rules of sentence formation, and in another sense by *the use of the sign in the language-game*'. This particular view is reflected directly in the title of a paper of mine, 'To Show the Consequences of a Proposition is To Give its Meaning' (de Queiroz 1988e), which attempts to demonstrate that the connections between the meaning of a proposition and the consequences one is allowed to draw

from it, seems to appear in differents forms throughout Wittgenstein's early as well as later writings. It is therefore, very different from Prof. Martin-Löf's account of the semantical rôle of the *normalisation* rules as given in:

> In the course of my semantical explanation of the elimination rule for implication, I have performed certain transformations which are very much like an implication reduction in the sense of Prawitz. Indeed, I have explained the semantical role of this syntactical transformation. The place where it belongs in the meaning theory is precisely in the semantical explanation, or justification, of the elimination rule for implication. Similarly, *the reduction rules for the other logical constants serve to explain the elimination rules associated with those constants.*
>
> (Martin-Löf 1985, p. 261.)

(The *emphasis* is mine.)

It is also different from Prof. Prawitz's emphasis on the *reduction* rules as 'methods that verify the correctness of elimination rules' in:

> The so-called reductions associated with the elimination rules in systems of natural deduction are nothing else than methods that verify the correctness of the elimination inferences in the above sense [When the conditions for correct assertions are given, however, we have also a general condition for the drawing of consequences.]. As I have described elsewhere in more detail, Gentzen's Hauptsatz or my normalizations in natural deduction are just based on considerations of meaning of the kind described in Section 5 [In order to construct a meaning theory that meets the requirement discussed in the preceding section, we may consider the possibility of identifying meaning with the two aspects of use described above. Or, since the two aspects have to stand in harmony with each other, it may be enough to identify the meaning with one of them and derive the other.].
>
> (Prawitz 1980, p. 7.)

In the framework of his *Frege structures*, Prof. P. Aczel appears to be placing the rule of *normalisation* in a fundamental rôle in the definition of what he calls *systems*: *lambda systems*, *pairing systems*, *Peano systems*, etc., are all defined by means of *normalisation*-like rules. For instance, his definition of a *lambda system* reveals the crucial rôle played by the equivalent of *β-reduction* in the characterisation of the *system* being defined:

Definition 3.1. A *lambda system* for an explicitly closed family $\mathcal{F}$ consists of two $\mathcal{F}$-functionals $\lambda : \mathcal{F}_1 \to \mathcal{F}_0$ and $\mathrm{APP} : \mathcal{F}_0 \times \mathcal{F}_0 \to \mathcal{F}_0$ such that

$$\mathrm{APP}(\lambda x f(x), a) = f(a)$$

for all f in $\mathcal{F}_1$ and a in $\mathcal{F}_0$.

(Aczel 1980, p. 38.)

Generally speaking, the rôle of *β-normalisation* rule of the λ-Calculus seems to be recognisably semantical. It seems then unfair to say that such rule of *normalisation* is a mere explanation of how the *elimination* 'APPLY' works, rather than the actual semantical explanation of 'λ' itself. A great many of the references to the rule of *β-normalisation* in the literature do acknowledge such a semantical rôle that it plays with respect to the logical sign 'λ', even if not advocating that the more general view that *normalisation* rules are semantical rules. Take, e.g., the semantical explanations of the NuPRL proof system based on *Intuitionistic Type Theory*, developed in Cornell by Prof. Constable and associates, where as a general rule the *normalisation* (*reduction*) rules are not said to play a semantical rôle, given that the semantics is still based on the *assertability conditions* (*introduction* rules) as in Prof. Martin-Löf's original semantical explanations. The meaning of λ-terms is explained in terms of its β-rule of *normalisation*:

Computation System

The meaning of lambda terms $\backslash x.b$ is given by computation rules. The basic rule, called *beta reduction*, is that $(\backslash x.b)(a)$ reduces to $b[a/x]$; (...)

(Constable et al. 1986, p. 22.)

In his book on techniques for the implementation of functional programming languages S. Peyton-Jones also explains the equality rules of the λ-Calculus as its 'operational semantics'. After the explanation of the 'syntax' where the *abstraction* rule is presented as the rule demonstrating how to form λ-abstraction terms, the conversion rules (including *β-reduction*) are used to explain 'how to calculate with the λ-Calculus':

2.2 The Operational Semantics of the Lambda Calculus

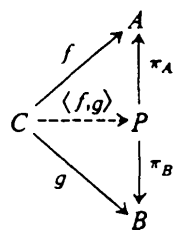
So far we have described only the *syntax* of the lambda calculus, but to dignify it with the title of a 'calculus' we must say how to 'calculate' with it. We will do this by giving three *conversion rules* which describe how to convert one lambda expression into another. (...)

(Peyton-Jones 1987, p. 14.)

The language of Category Theory would seem to be another example where one can find the device of *normalisation* being used as part of the definition of logical signs. In earlier accounts of such 'language', although in the definition of 'logical' (not in the sense of formal logistic) signs the *eliminatory* operators did play a definite rôle, the actual *normalisation* device did not always appear and was often replaced by a 'commuting' diagram. For instance, in Herrlich & Strecker 1973, products and coproducts (which would roughly correspond to our Π and Σ types, respectively) are defined by the corresponding 'constructors' (*introductory* operators) — '$<\ ,\ >$' for product, 'μ_A and 'μ_B' for coproducts —, and the 'destructors' (*eliminatory* operators) — 'π_A' and 'π_B' for products, '[,]' for coproducts —.

18.2 DEFINITION

A $\mathcal{C}$-**product** of a pair (A, B) of $\mathcal{C}$-objects is a triple (P, π_A, π_B) where P is a $\mathcal{C}$-object and $\pi_A : P \rightarrow A, \pi_B : P \rightarrow B$ are $\mathcal{C}$-morphisms (called **projections**) with the property that if C is any $\mathcal{C}$-object and $f : C \rightarrow A$, $g : C \rightarrow B$ are arbitrary $\mathcal{C}$-morphisms, then there exists a *unique* $\mathcal{C}$-morphism (usually denoted by) $< f, g >: C \rightarrow P$ such that the diagram



commutes.

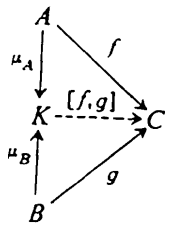
(Herrlich & Strecker 1973, p. 115.)

And similarly for coproducts:

18.3 DEFINITION

A $\mathcal{C}$-**coproduct** of a pair (A, B) of $\mathcal{C}$-objects is a triple (μ_A, μ_B, K) where K is a $\mathcal{C}$-object and $\mu_A : A \rightarrow K, \mu_B : B \rightarrow K$ are $\mathcal{C}$-morphisms (called **injections**) with the property that if C is

any $\mathcal{C}$-object and $f : A \rightarrow K$, $g : B \rightarrow K$ are arbitrary $\mathcal{C}$-morphisms, then there exists a *unique* $\mathcal{C}$-morphism (usually denoted by) $[f, g] : K \rightarrow C$ such that the diagram



commutes.

(Ibid., p. 116.)

Nonetheless, in later accounts of category theory, especially those concerned with the reconciliation of logical notions with categorical notions (and vice-versa), as in Lambek & Scott 1986, one finds the explicit demonstration of the effect of the eliminatory projections on the pairing operator as part of the definition of product:

It may be worth looking at the product of two objects A and B of $\mathcal{A}$ in some detail. It is given by an object $A \times B$ with projections $\pi_{A,B} : A \times B \rightarrow A$ and $\pi'_{A,B} : A \times B \rightarrow B$ such that, for all arrows $f : C \rightarrow A$ and $g : C \rightarrow B$, there is a unique arrow $< f, g >: C \rightarrow A \times B$ satisfying the equations:

$\boxed{\pi_{A,B} < f, g >= f}$, $\boxed{\pi'_{A,B} < f, g >= g}$.

Note that the uniqueness of $< f, g >$ may also be expressed by an equation, namely:

$< \pi_{A,B} h, \pi'_{A,B} h >= h$,

for all $h : C \rightarrow A \times B$.

(Lambek & Scott 1986, p. 20.)

(The $\boxed{\text{emphasis}}$ is mine.)

Observe that the $\boxed{\text{boxed}}$ equations correspond to the rules of Σ-*reduction* of our type-theoretic framework, and the 'uniqueness' condition corresponds exactly to our rule of minimality Σ-*induction*.

For *coproducts* the situation does not essentially change, and the effect of the *eliminatory* operator '[,]' (the parallel of our '`WHEN`') on the *introductory* operators '$\kappa_{A,B}$' and '$\kappa'_{A,B}$' are also part of the definition of the coproduct:

Cartesian closed categories were defined as positive intuitionistic propositional calculi satisfying certain equations between proofs. To complete the picture we define a *bicartesian closed category* as a full intuitionistic propositional calculus satisfying the following additional equations:

E5. $f = \Box_A$ for all $f : \bot \to A$;

E6a. $[f, g]\kappa_{A,B} = f$;

E6b. $[f, g]\kappa'_{A,B} = g$;

E6c. $[h\kappa_{A,B}, h\kappa'_{A,B}] = h$;

for all $f : A \to C$, $g : B \to C$ and $h : A \vee B \to C$.

Evidently, the defining property of $A \times B$ establishes a bijection between pairs of arrows $(C \to A, C \to B)$ and arrows $C \to A \times B$.

(Ibid., p. 65.)

Note that E6a and E6b correspond to our $+$-*reduction* rules, and E6c finds a parallel in our $+$-*induction*.

In the definition of typed λ-Calculus within the type-theoretic framework with categorical principles Lambek & Scott also include β-*reduction* as part of the definition of the calculus:

A *typed λ-calculus* is a formal theory defined as follows. It consists of classes of 'types', 'terms' of each type, and 'equations' between terms which are said to 'hold', all subject to certain closure conditions. We shall write $a \in A$ to say that a is a term and is of type A; the symbol $\in$ belongs to the metalanguage.

(a) *Types*: (...)

(...)

(b) *Terms*: (...)

(...)

(c) *Equations*: (...)

(...)

(c3) The following specific equations hold:

$a \underset{X}{=} *$ for all $a \in 1$;

$\pi(< a, b >) \underset{X}{=} a$ for all $a \in A, b \in B$,

$\pi'(< a, b >) \underset{X}{=} b$ for all $a \in A, b \in B$,

$< \pi(c), \pi'(c) > \underset{X}{=} c$ for all $c \in A \times B$;

$\lambda_{x \in A}\varphi(x)^{c}a \underset{X}{=} \varphi(a)$ for all $a \in A$ which are substitutable for x,

$\lambda_{x\in A}(f^c x) \underset{X}{=} f$ for all $f \in B^A$, provided x is not in X (hence does not occur freely in f);

(...)

$\lambda_{x\in A}\varphi(x) \underset{X}{=} \lambda_{x\in A}\varphi(x')$, if x' is substitutable for x.

(Ibid., pp. 73–5.)

The second, third and fourth equations find their parallels in our presentation of Σ types, whereas the fifth and sixth correspond to Π-*reduction* and Π-*induction*, respectively.

5.3. Formalising Constructive Mathematics

Prof. Martin-Löf's pioneering work on elaborating a formal presentation of the informal concepts regulating ordinary practice of constructive mathematics, has opened new perspectives on the rôle of language in mathematics and the way logics, as well as mathematical theories, can be presented. Instead of insisting on the use of the more customary meta-mathematical machinery such as axioms, models and interpretations, and instead of taking for granted notions such as functions and sets, Prof. Martin-Löf engaged in much deeper investigations into the nature of 'more primitive' logical concepts such as proof, proposition, judgement, types, and inference rules, in search for an alternative to the foundations of mathematics which could be at the same time reasonably self-contained and less committed to certain existing traditions. As an experienced and productive proof-theorist it is not quite surprising that the choice was made for the presentation of the formulation of a system of rules to formalise constructive mathematics in proof-theoretic natural deduction style. This way he appears to have provided a 'cleaner' and certainly innovative way of presenting the whole body of ideas and informal concepts involved in ordinary mathematical practice. (Remember that up to then proof-theoretic presentation systems had been mostly used for presentation of *logics*, not mathematical theories or foundational principles.) That seems to have been one of the reasons for the major success of his Type Theory over other systems of formalisation of constructive mathematics such as, e.g., K. Gödel's *T-systems* (Gödel 1958), E. Bishop's *Σ-sytem*

(Bishop 1970)[8], J. Myhill's *Constructive Set Theory* (Myhill 1975), S. Feferman's *Explicit Mathematics* (Feferman 1979), H. Friedman's *Theories of constructive analysis* (Friedman 1977), Prof. M. Beeson's *S-systems* (Beeson 1981), Prof. J.-Y. Girard's *Y-* and *F-systems* (Girard 1971), etc., to mention just a few. The clarification of some important logical concepts such as the crucial distinction between *propositions* and *judgements*, helped by a certain intuitive underlying conceptual paradigm saying that *propositions-are-types* (apparently first uncovered by H. Curry and given proper foundations by W. Howard (Howard 1980)), together with a reasonably coherent philosophical basis and a suitable formal instrument such as proof theory (natural deduction style) have made *Intuitionistic Type Theory* the object of careful attention by many contemporary philosophers, linguists, logicians as well as computer scientists. The result is, one can say, an attempt to 'code' the basic procedures of *informal mathematical reasoning* into logical inference rules, similarly to what the lambda calculus does for just a single concept such as *computable function*.

The key point, as far as the relations between proof theory and meaning are concerned, seems to have been the 'internalisation' of the meta-level *normalisation* (*reduction*) rules. And this, I think, is because such rules of *normalisation* seem to be capturing the second aspect of the (later) Wittgenstein's account of meaning (use) of particular propositions by demonstrating 'the use of the sign in the language-game' by means of a formal device indicating how *eliminatory* operators affect *canonical* elements built by *introduction* rules. They seem to capture the very aspect of the *consequences* of the proposition. A suitable combination of the *propositions-*

[8]In passing, Bishop seems to have been the first to suggest the use of a formal system of constructive mathematics as a programming language:

'The system Σ, with the predictive interpretation of implication, can certainly be presented as a programming language like fortran or algol.'

(Bishop 1970, p. 70.)

and

'It would be interesting to take Σ as the point of departure for a reasonable programming language, and to write a compiler.'

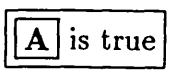
(Ibid., p. 71.)

are-types idea with the internalisation of *normalisation* rules in a way similar to what is done in *Intuitionistic Type Theory* leads to a reasonable formulation of Wittgenstein's later account of propositions, where the *introduction* rules represent the *grammar* aspect ('And what a proposition is is in *one* sense determined by the rules of sentence formation (...)') and the *reduction* rules show the *use* aspect of the proposition ('(...) and in another sense the use of the sign in the language-game.'). Such rules of *normalisation* seem to be showing the aspect of *use* as *utility* of the types (propositions) because they indicate what can be done with the (canonical) elements of the type (proposition). There has not been the expected emphasis on the rôle of these rules with respect to the semantics of the logical constants, given that the account of meaning is kept within Gentzen's tradition which says that the *assertability conditions* represented by the schema of *introduction* rules capture the meaning of a logical connective. For that reason I believe that a more convincing justification that the whole formal system is not just mere notation still lacks the force of a truly convincing alternative to the classical (realist) semantics of the mathematical language.

Nevertheless, the formal system known as *Intuitionistic Type Theory* (*ITT*) is indeed a remarkable point of departure for the construction of a meaning theory for the mathematical language. A deep revision of most of the basic logical principles and a thorough analysis of ordinary mathematical practice resulted in what is certainly the most successful of the formalisations of constructive mathematics so far developed.

5.3.1. The crucial distinction: propositions and judgements

One of the most important conceptual principles of *Intuitionistic Type Theory* is the distinction between *propositions* and *judgements*.



'**A**' is a *proposition*, whereas '**A** is true' is a *judgement*. A *judgement* of the form

'**A** is true' can only be made on the basis of the existence of a proof of the proposition **A**. Contrary to the classical view, a proposition is not the same as a truth value. And in contrast to the traditional proof-theoretic account of propositions and inferences, a logical inference is to be made from *judgement* to *judgement*, and not from *propositions* to *propositions*. Both premisses and conclusions of inference rules are not *propositions* as in the usual case even in traditional natural deduction presentations of logics, but *judgements*. This seems to be a highly relevant refinement of the usual formalisation of mathematical procedures into rules of inference, such as e.g., natural deduction style *à la* Gentzen. The difference between usual natural deduction presentation rules which have *propositions* as premises and conclusions, and *Intuitionistic Type Theory* where *judgements* are the objects on which the rules of inference operate, is explained briefly in Prof. Martin-Löf's illuminating account of the often neglected distinction between the two logical concepts of *proposition* and *judgement*, namely the written account of a series of lectures entitled 'On the Meanings of the Logical Constants and the Justifications of the Logical Laws' given in Siena, Italy, in April 1983. There he says:

(...) We must remember that, even if a logical inference, for instance, a conjunction introduction, is written

$$\frac{A \quad B}{A \,\&\, B}$$

which is the way in which we could normally write it, it does not take us from the propositions A and B to the proposition A & B. Rather, it takes us from the affirmation of A and the affirmation of B to the affirmation of A & B, which we may make explicit, using Frege's notation, by writing it

$$\frac{\vdash A \quad \vdash B}{\vdash A \,\&\, B}$$

instead. It is always made explicit in this way by Frege in his writings, and in Principia, for instance. Thus we have two kinds of entities here: we have the entities that the logical operations operate on, which we call propositions, and we have those that we prove and that appear as premises and conclusion of a logical inference, which we call assertions. (...)

(Martin-Löf 1985, pp. 204–5.)

The use of 'affirmation' is made in reference to the two distinct classes into which 'categorical' (in the philosophical sense) judgements are divided, namely 'affirmations' and 'denials'. This is made clear soon after the above passage. It reads:

> Be that as it may, what happened with Kant and the ensuing German philosophical tradition was that the word judgement came to replace the word proposition. Thus, in that tradition, a proof, Ger. Beweis, is always a proof of a judgement. In particular, the premises and conclusion of a logical inference are always called judgements. And it was the judgements, or the categorical judgements, rather, which were divided into affirmations and denials, whereas earlier it was the propositions which were so divided.
>
> (Ibid., p. 206.)

Following such tradition of taking *judgements* instead of *propositions* as the objects of logical inferences, and being germane to a particular conception of what the logic of mathematics should be, namely that which says that a *proposition* is defined by laying down what counts as a proof of it, and a *proposition* is true if and only if a proof of it can be constructed (Prawitz 1980), Prof. Martin-Löf has made a remarkable improvement on Frege's attempt to formulate a *lingua characterica*[9] based on such a notion of *judgement*. Keeping the notion of *judgement* as one of the basic notions of my own *'Meaning-As-USE' Type Theory*, but now in the light of Wittgenstein's 'language as a calculus, calculus as a game'[10], I have attempted with

[9]'My intention [in the *Begriffsschrift* (Frege 1879)] was not to represent an abstract logic in formulas, but to express a content through written signs in a more precise and clear way than it is possible to do through words. In fact, what I wanted to create was not a mere *calculus ratiocinator* but a *lingua characterica* in Leibniz's sense', reproduced in p. 2 of editor's preface of Frege 1879.

[10]On Monday 21 September 1931, in a discussion with Waismann on topics such as intention, to mean and meaning, Wittgenstein says:

> 'You must not forget that I am now not talking about propositions but about dealing with signs. I am saying that our way of using signs constitutes a calculus, and I am saying this deliberately. For there is not a mere analogy between our way of using words in a language and a calculus; I can actually construe the concept of a calculus in such a way that the use of words will fall under it. I shall presently explain how I mean this. Here I have a little bottle of benzine. What purpose does it serve? Well, it is used for cleaning. Now there is a label glued onto it saying 'benzine'. What purpose, then, does the label serve? Surely I use benzine for cleaning, not the label. (It

my own type-theoretic framework to build a *lingua characterica* which could work as a *calculus raciocinator*, by no means with the intention of fulfilling the rather ambitious dream of Leibniz.

5.3.2. Forms of Judgement

The forms of *judgement* made in *Type Theory* are of the following kind:

(1) **A** is a *type* — formally '**A** *type*';

(2) **A** and **B** are equal *types* — formally '$\mathbf{A} = \mathbf{B}$';

(3) a is an element of the *type* **A** — formally '$a \in \mathbf{A}$';

(4) a and b are equal elements of the *type* **A** — formally '$a = b \in \mathbf{A}$'.

The explanations given by Prof. Martin-Löf are as follows:

---

is of course clear that instead of this label there could be a different one.) Now this label is the point of contact of a calculus, that is, the point where it gets applied. For I can tell you, 'Get some benzine!' And in virtue of the label there is a rule in accordance with which you can proceed. When you get the benzine, this is again a step within the calculus defined by these rules. I call the whole business a calculus, since now there are two possibilities: that you proceed according to a certain rule or that you do not proceed according to this rule; for now I am in the position to say, for instance: 'Well, what you have brought is not benzine at all!'

(*LW&VC*, p. 169.)

and later on, when commenting on calculus and application he says:

'What is the difference between language (M) [Editor's note: perhaps mathematics] and a game? You might say: It ceases to be a game when things begin to become serious, and here seriousness means application. But this would not express it quite correctly. You would really have to say that a game is what is neither serious nor fun. For we call things serious if we apply the results of the calculus to everyday life. I apply e.g. the calculation $8 \times 7 = 56$ thousands of times, and this is why we take it seriously. But, at bottom, this multiplication is not the least bit different from one I merely do for fun. The difference does not lie in the calculation itself, and that is why you cannot tell by looking at the calculus whether it is a serious business or merely serves to entertain us. Hence I cannot say: If a calculus amuses me, it is a game, but only: If I *can take a calculus to be something* that amuses me, then it is a game. The calculus itself implies no relation either to seriousness or to fun.'

(*LW&VC*, p. 170.)

(1) a set **A** is defined by prescribing how a canonical element of **A** is formed as well as how two equal canonical elements of **A** are formed.[11]

Judgements of the kind '**A** *type*' are used in *formation* rules, as we shall see later on, and determine that under certain conditions a type-expression is a 'well-formed' *type*. In the explanation given above, however, it looks as though the *introduction* rules for a given type are being justified. The *introduction* rules for a type form a formal counterpart to Bishop's principle that a *constructive* set is to be defined by giving its canonical elements together with an explanation of when two canonical elements are equal. So, instead of a proper explanation of *judgements* of the form '**A** *type*', Prof. Martin-Löf's explanation is almost verbatim E. Bishop's:

As discussed previously, a *set* is defined by describing what must be done to construct an *element* of the set, and what must be done to show that two elements of the set are *equal*.

(Bishop 1967, p. 63.)

This is actually acknowledged in the introductory sections of the first 'public' description of the intuitionistic theory of types that has been developed by Prof. Martin-Löf:

(...) A type is defined by prescribing what we have to do in order to construct an object of that type. This is almost verbatim the definition of the notion of set given by Bishop [in Bishop 1967]. (...)

(Martin-Löf 1975a, p. 76.)

The informal explanation of the *judgements* of the form '**A** *type*' is given in a more simplified way in Martin-Löf 1982, although something is left to be explained, which is the notion of canonical *type* (as opposed to canonical *element*). The *judgements* of the form given above are explained as:

(...) For noncanonical A, a judgement of the form

$$A \text{ is a type}$$

means that A has a canonical type as value.

[11] Martin-Löf 1984, p. 8.

(Martin-Löf 1982, p. 163.)

There seems to be clear definitions of what a canonical *element* is, but no precise account of what is to be understood by canonical *type* seems to be given in a clear-cut manner, apart from an explanation in terms of the corresponding canonical *elements*, which takes almost exactly the same wording as (1) above:

> A canonical type A is defined by prescribing how a canonical object of type A is formed as well as how two equal canonical objects of type A are formed.

(Ibid., p. 162.)

and a fairly loose notion of 'Universes' which would contain such 'canonical' *types*. A more careful look at the table of 'primitive forms of expression used in the theory of types' given on p. 156 confirms this:

TABLE 2. PRIMITIVE FORMS OF EXPRESSION USED IN THE THEORY OF TYPES

canonical	non-canonical
$(\Pi x \in A)B, \quad (\lambda x)b$	$\boxed{c(a)}$
$(\Sigma x \in A)B, \quad \boxed{(a,b)}$	$(Ex,y)(c,d)$
$A+B, \quad i(a), \quad j(b)$	$(Dx,y)(c,d,e)$
$I(A,a,b), \quad r$	$J(c,d)$
N_0	$R_0(c)$
$N_1 \quad 0_1$	$R_1(c,c_0)$
$N_2 \quad 0_2, \quad 1_2$	$R_2(c,c_0,c_1)$
$\vdots$	$\vdots$
$N \quad 0, \quad a'$	$(Rx,y)(c,d,e)$
$(Wx \in A)B, \quad \sup(a,b)$	$(Tx,y,z)(c,d)$
$U_0, \quad U_1, \quad \cdots$	

(Ibid., p. 158.)

(The emphasis is mine.)

Note that the canonical elements of a type are placed opposite the corresponding non-canonical elements, whereas for the case of types (or type-expressions) there is no corresponding non-canonical form. The 'canonical' types, namely '$(\Pi x \in A)B$', '$(\Sigma x \in A)B$', '$A + B$', '$I(A, a, b)$', '$N_m \quad (m = 0, 1, ...)$', '$N$', '$(Wx \in A)B$', '$U_m \quad (m = 0, 1, ...)$', are listed under the corresponding column. However, everything under the column named 'non-canonical' is an *element*-expression, no *type*-expression appearing in that column. As we shall see later on, the absence of a precise definition of what a non-canonical type-expression is, is an invariant characteristic of all versions of *Intuitionistic Type Theory*, even in the latest presentations available in the (rather scarce) literature.

With respect to the notation used to present the Type Theory given in the paper 'Constructive Mathematics and Computer Programming' (Martin-Löf 1982) I should like to point out the following inconveniences, emphasised in the table quoted above: first of all, the notation for the non-canonical elements of the Π type, namely '$c(a)$', seems to leave room for undesirable confusion given that the usual 'APPLY' is not used explicitly and application is represented by mere juxtaposition. Besides generating confusion as to when terms formed by juxtaposition are to be understood as representing 'application', such notation does not help in distinguishing non-canonical terms that would be the counterparts to the canonical 'λ-terms' from terms built up by juxtaposition such as, e.g., '$i(a)$' – one of the canonical terms of the type '$A + B$'.[12]

[12]Just in passing, I thought it might be useful to recall the first attempt to deal formally with the notion of 'abstraction', namely that of Frege and the idea of 'incomplete signs'. Although in a totally different context, the omission of an explicit 'APPLY' appears to have caused problems for Frege in his logical formulation of the notion of set, as pointed out by Prof. P. Aczel when explaining the differences between the structure of Frege's *Grundgesetze der Aritmetik* (Frege 1893, 1903, partially translated in Furth (ed.) 1964), which does not use 'APPLY' explicitly in the definition of the notion of 'course of values' of a function 'f' as '$\dot{\varepsilon} f(\varepsilon)$' which was rendered later by Church as '$\lambda x f(x)$', and his own *Frege structures*, which does use an explicit 'APPLY' in the definition of a *lambda system*:

'(...) Russell's paradox can be derived in Frege's Grundgesetze and we now list the essential ways

Secondly, by using simply brackets '(,)' to represent the canonical element formation of Σ types, instead of using another symbol different from the general term formation '(,)', such as, e.g., '< , >' or 'pair', Prof. Martin-Löf leaves

---

in which the structure of the Grundgesetze differs from a Frege structure.

(...)

(4) The Grundgesetze does not treat APP as given. Instead it is explicitly postulated in Axiom V that λ is to be one-one. I.e. for all functions f, g in $\mathcal{F}_1$

$$(\lambda x f(x) \doteq \lambda x g(x)) \equiv \forall x(f(x) \doteq g(x)) \text{ is true.}$$

(Note that because of (1) Frege uses $\doteq$ instead of $\equiv$.)

In addition the Grundgesetze allows function quantification and definite descriptions and then defines APP(b, a) to be $\iota y \exists f(b \doteq \lambda x f(x) \& f(a) \doteq y)$ (informally, the y such that for some f in $\mathcal{F}_1$ $b = \lambda x f(x)$ and $f(a) = y$).'

(Aczel 1980, p. 40.)

Obviously, this is not to say that Frege's most crucial mistake was in the definition of the notion of 'course of values' without the use of an explicit application rule. As Prof. Aczel points out, 'Frege's critical mistake is in the fact that the *Grundgesetze* has exactly two propositions, namely "the true" and "the false", and "the true" is the only true proposition'. According to Prof. Aczel, the essence of Frege's mistake would seem to lie in the apparently innocuous horizontal stroke schema, and the internal definition of truth.

One can also find in H. Curry's *Outlines of a Formalist Philosophy of Mathematics* (Curry 1951) what is perhaps the earliest example of an explicit treatment of 'application' as an operator. In the chapter devoted to 'Examples of Formal Systems' (pp. 21–2) he attempts to demonstrate that Church's λ-Calculus (or 'theory of conversions, as he calls it) can be given a natural formulation as a formal system by the same general pattern which he defines as:

I. Primitive Ideas
   A. Tokens
   B. Operations
   C. Predicates

II. Primitive Theorems
   A. Axioms
   B. Rules of Procedure

Under such formulation, there appear two binary operators 'λ' and 'α' representing abstraction and application respectively. Abstraction terms are of the form '$\lambda x_i \mathcal{A}$' (where '$\mathcal{A}$' is a term with x_i occurring free), and application terms have the form '$\alpha \mathcal{A} \mathcal{B}$' to indicate that '$\mathcal{A}$' is being applied to

room for troublesome confusions. After all, the development of a new foundational language for constructive mathematics and computer programming such as *Intuitionistic Type Theory*, should actually provide us with means to avoid syntactical as well as semantical confusions[13].

In later works, such as, e.g., the lecture notes edited by Prof. G. Sambin (Martin-Löf 1984), the explicit use of 'APPLY' is made, but the notational inconvenience remains for the case of the Σ type and its corresponding pair-terms. In more recent presentations of *Intuitionistic Type Theory*, such as the one given in Nordström, Petersson & Smith 1986, this notational inconvenience is cleared by the use of angle brackets '< , >' (Nordström, Petersson & Smith 1986, p. 69.).

The second form of *judgement*, namely '$\mathbf{A} = \mathbf{B}$', is explained in Martin-Löf 1984 as:

(2) two sets **A** and **B** are equal if

'$\mathcal{B}$'. The effect of application in an abstraction term is then given by one of the 'Axioms' (II.A.3) written as:

$$\alpha\lambda x_i.\mathcal{A}\mathcal{B} \text{ cnv } \text{Subst}(\tfrac{\mathcal{B}}{x_i})\mathcal{A}$$

which is a particular presentation of the β-rule of *normalisation* '$\texttt{APPLY}(\lambda x_i.\mathcal{A}(x_i), \mathcal{B}) = \mathcal{A}(\mathcal{B}/x_i)$' where '$\alpha$' is the counterpart of 'APPLY'.

The notion of 'application', which is usually formalised by a β-like rule of *normalisation* with an explicit operation such as 'APPLY', seems to be inseparable from that of 'generality' and 'substitution' which 'λ' is supposed to encapsulate. Attempts to deal with such (functional) notions of generality and substitution without incorporating the device of 'application' are likely to be problematic.

[13]Prof. Martin-Löf's strong philosophical interest in 'mathematical logic as foundations (or philosophy) of mathematics' (Martin-Löf 1984, p. 1) is made clear in passages such as: 'What I have just said about the close connection between constructive mathematics and programming explains why the intuitionistic type theory, *which I began to develop solely with the philosophical motive of clarifying the syntax and semantics of intuitionistic mathematics*, may equally well be viewed as a programming language.', p. 156 of Martin-Löf 1982 (the *emphasis* is mine). Note that besides its clearly emphasised semantical objectives, Prof. Martin-Löf's Type Theory is an attempt to clarify syntactic issues as well.

$$\frac{a \in \mathbf{A}}{\overline{\overline{a \in \mathbf{B}}}} \quad \text{(that is,} \quad \frac{a \in \mathbf{A}}{a \in \mathbf{B}} \quad \text{and} \quad \frac{a \in \mathbf{B}}{a \in \mathbf{A}}\text{)}$$

and

$$\frac{a = b \in \mathbf{A}}{\overline{\overline{a = b \in \mathbf{B}}}}$$

for arbitrary canonical elements a, b.[14]

Similarly to the case of individuals (value-expressions, elements), as in the *judgements* of the form (4) below, one needs to judge types equal. The *judgement* '$\mathbf{A} = \mathbf{B}$' says when two type-expressions are equal.

The third form of *judgement*, while being perhaps the most fundamental of all, is the one which has apparently caused most misunderstandings. It is the *judgement* which first arises out of the identification of propositions and types, as well as the distinction between propositions and *judgements*. Prof. Martin-Löf's latest informal explanation, appearing in the Padova lecture notes, departs from the clean simplicity of its original explanation by introducing the notions of method and program, bringing with it a fair amount of imprecision as a result. It says:

> (3) an element a of a set $\mathbf{A}$ is a method (or program) which, when executed, yields a canonical element of $\mathbf{A}$ as result.[15]

Perhaps I should at this point make it clear that among the readings I wish to allow for *judgements* of the kind '$a \in \mathbf{A}$' in my own account of *Type Theory*, I do not consider the one which says 'a is a program meeting the specification $\mathbf{A}$'. Such a reading, which is explicitly considered by Prof. Martin-Löf himself by reference to the work of A. N. Kolmogorov on a 'procedural' interpretation of intuitionistic logic (Kolmogorov 1932), as a reasonable conceptual basis for the new account of the *judgement* aforementioned involving the concepts of 'program' and 'method'. He says:

---

[14] Martin-Löf 1984, p. 9.

[15] Id. Ibid.

Each form of judgement admits of several different readings. (...) The last one [a is a method of solving the problem (doing the task) **A**] is very close to programming. "a is a method ... " can be read as "a is a program ... ". Since programming languages have a formal notation for the program a, but not for **A**, we complete the sentence with " ... which meets the specification **A**". (...)

(Martin-Löf 1984, pp. 5-6.)

is sometimes used as a justification for the undecidability of the type membership relation of Prof. Martin-Löf's *Intuitionistic Type Theory*, by recourse to the undecidability of the question whether or not a program satisfies a certain specification. I do not find it reasonable to use such an argument as a justification for the undecidability of '$\in$' in Prof. Martin-Löf's latest formulations of *Type Theory*. Earlier 'decidable' versions which did not include the **W** type forming operator, such as that of 1973 (Martin-Löf 1975a), have a type membership relation '$\in$' recognisably decidable[16] and yet they do not differ so radically in terms of forms of *judgements* from later versions to the extent of not accommodating the reading 'a is a program that satisfies the specification **A**'. It sounds more reasonable to say that the undecidability of the latest formulations arose from the introduction of an undecidable transfinite recursion operator in the *elimination* rules for the type of well-ordered species $(\mathbf{W}x \in \mathbf{A})\mathbf{B}(x)$ as we shall see later on.

The explanation of the *judgement* '$a \in \mathbf{A}$' by means of vague notions such as method, program, etc., seems to have been first introduced by Prof. Martin-Löf in the context of discussions on the connections between constructive mathematics and computer programming, where the appeal to the applications of his *Type Theory* to computing science was indeed very strong. It has unfortunately turned out to be rather misleading.

---

[16]See Fact 3.13 of Martin-Löf 1975a:

'3.13. DECIDABILITY OF THE EPSILON RELATION. *It can be mechanically decided whether or not a closed term is a type symbol. And, given a closed term a and a closed type symbol* **A**, *it can be mechanically decided whether or not* $a \in \mathbf{A}$.'

(Martin-Löf 1975a, p. 116.)

It looks as though the informal explanation of the *judgements* of the kind '$a \in \mathbf{A}$' is better given simply as in the original formulation, where it is less likely that misconceptions will arise: the value-expression 'a' when *reduced* to its canonical form represents an element of the type '$\mathbf{A}$'. Or, more generally, the *judgement* '$a \in \mathbf{A}$' is the formal counterpart to the informal 'a is an object of type $\mathbf{A}$', types, propositions, elements and proofs being the most primitive concepts not including programs or specifications. This is actually verbatim the original reading of the *judgement* as given by Prof. Martin-Löf in Martin-Löf 1975a:

(...) I shall use the notation

$$a \in A$$

to express that

a is an object of type A.

(Martin-Löf 1975a, p. 76.)

which precedes the introduction of the central paradigm of the type-theoretic framework, namely 'propositions-are-types':

When a type A is thought of as a proposition,

$$a \in A$$

expresses that

a is a proof of the proposition A.

(Ibid., p. 77.)

It seems as though the (rather misleading) interpretation of '$a \in \mathbf{A}$' as being 'a is a program which meets the specification $\mathbf{A}$' (Martin-Löf 1984, p. 6.) was only introduced to justify the attempt to make connections between constructive mathematics and computer programming in the widely quoted paper delivered at the *Sixth International Congress for Logic, Methodology and Philosophy of Science*, Hannover, 1979 (Martin-Löf 1982). In that paper, the new reading of the *judgement* '$a \in \mathbf{A}$' is justified by reference to A. N. Kolmogorov's interpretation of propositions

as intentions or tasks, and proofs as methods of fulfilling the intention (Kolmogorov 1932). Perhaps under the influence of the conceptual distinction between programs and specifications that is usually made in computer programming, and in order to support the claim that there are indeed strong connections between constructive (intuitionistic) mathematics and computer programming, Prof. Martin-Löf adds the new reading which identifies the *element*-expressions to the left of '$\in$' with programs, and the *type*-expressions to the right of '$\in$' with specifications of the tasks to be fulfilled. The new reading is justified as:

> The first form of judgement admits not only the readings
>
> **A** is a type (set),
>
> **A** is a proposition,
>
> but also, and this is the reading that is most natural when the language is thought of as a programming language,
>
> **A** is a problem (task).
>
> Correlatively, the third form of judgement may be read not only
>
> *a* is an object of type (element of the set) **A**,
>
> *a* is a proof of the proposition **A**,
>
> but also
>
> *a* is a program for the problem (task) **A**.
>
> The equivalence of the first two readings is the, by now well known, correspondence between propositions and types discovered by Curry (1958) and Howard (1969), whereas the transition from the second to the third is Kolmogorov's (1932) interpretation of propositions as problems or tasks (Ger. *Aufgabe*).
>
> (Martin-Löf 1982, p. 162.)

(The emphasis is mine.)

As such a newly introduced reading (with respect to the original conception) '*program* $\in$ **SPECIFICATION**' proves rather inappropriate, I have preferred to stay with the original explanations, and in my own type-theoretic framework such a *judgement* is to be associated with the readings '*element* $\in$ **TYPE**' and '*proof* $\in$ **PROPOSITION**'.

The last form of *judgement* deals with *element*-expressions, as opposed to *type*-expressions as in the case (2) above, and involves the equality sign. So, it is essentially the *judgement* which allows one to say in the definition/presentation of types, when two *elements* are equal up to their canonical values. The informal explanation given by Prof. Martin-Löf goes as follows:

> (4) two arbitrary elements a, b of a set **A** are equal if, when executed, a and b yield equal canonical elements of **A** as results.[17]

Following the same pattern as given in the explanation of the previous form of *judgement*, '$a = b \in \mathbf{A}$' seems to be better understood as saying that the two value-expressions 'a' and 'b', when brought to their canonical form, represent equal elements of the type '**A**'. The use of terms such as 'executed' again brings an undesirable degree of vagueness which appears to be inherent in the present state of concepts such as 'program' and 'method'.

5.3.3. Canonical (Direct) Proofs

Based on the *propositions-are-types* paradigm, the semantical explanations of *ITT* say that a proposition/type is considered true (non-empty) if a *canonical* proof/element of it can be constructed. As just stated above, rules of inference deal with *judgements* rather than with *propositions* as in the conventional case. A *judgement* of the form '$a \in \mathbf{A}$' is to be understood as 'a is a proof of the proposition **A**' ('a is an element of the type **A**').

To see the relation to the intuitionistic account of logical constants, let us recall Heyting's explanation of logical constants. In order to explain the logical connectives from an intuitionistic perspective, Heyting made use of the notion of 'proof'. The explanation of each logical constant is given through exhibiting the conditions under which the proposition with such a constant as its major sign is to be considered true,

[17] Id. Ibid.

and the notion of truth is inseparable from that of 'proof'. According to Heyting, a proposition is true if a proof of it can be exhibited.

a proof of the proposition:	consists of:
$\mathbf{A} \wedge \mathbf{B}$	a proof of **A** and a proof of **B**.
$\mathbf{A} \vee \mathbf{B}$	a proof of **A** or a proof of **B**.
$\mathbf{A} \rightarrow \mathbf{B}$	a method which produces a proof of **B** from a proof of **A**.
$(\forall x)\mathbf{B}(x)$	a method which produces a proof of $\mathbf{B}(a)$ from an arbitrary individual a.
$(\exists x)\mathbf{B}(x)$	an individual (witness) a and a proof of $\mathbf{B}(a)$.

Observe that the (semi-)formal explanation of logical constants given by Heyting involves the use of similar concepts to explain both implication ('$\rightarrow$') and universal quantification ('$\forall$'). The notion of 'method' is crucial for the explanation of both logical connectives. Similarly, the logical constants of conjunction ('$\wedge$') and existential quantification ('$\exists$') are explained in terms of the mathematical concept of 'ordered pairing'. These parallels in the explanation of logical constants by means of the exhibition of what is a proof of the corresponding proposition will be clearly reflected in the explanation of the notion of propositions and proofs as given in Prof. Martin-Löf's 'verificationistic' theory of meaning called *Intuitionistic Type Theory*.

Besides the basic principle saying that a verificationistic theory of meaning is meant to offer an *effective* account of the world-language links, the idea of the primacy of the proofs of a proposition over other aspects of its use seems to constitute

the essence of certain set of basic principles by which one has to abide when putting forward a theory of meaning which can be considered as 'verificationistic'. Variations on the formulation of such a main idea have emerged and the notion of *direct* proofs appeared as a key notion to motivate further improvements on the main theme. Arguing for a certain lack of precision in the (essentially germane) formulations of semantical explanations via *conditions for proving* given by G. Gentzen, A. N. Kolmogorov, etc., Prof. Martin-Löf presents his own explanations for the refinement on such equivalent semantical explanations by insisting on the need for a more precise account of what should be understood by the connections between *proofs* and *meaning*.

Such an elegantly presented refinement of the *proof*-based semantics finds its justification as being a germane extension of the intuitionistic explanation of the meaning of logical signs which brings the previous explanations to a more precise presentation. Prof. Martin-Löf finds the necessary conceptual basis for his development of the desired refinement in the intuitionistic account of meaning and proofs in Prof. Dummett's distinction between indirect (non-canonical) and direct (canonical) proofs as well as in the corresponding explanation of the meaning of logical constants in terms of 'direct' proofs, as in:

> (...) What appear in ordinary mathematical articles and textbooks are demonstrations, not proofs in the strict sense; and a demonstration provides an adequate ground for the unqualified assertion of its conclusion. But the primary notion is that of a proof in the strict sense, which we shall refer to as a *canonical* proof: the notion of a demonstration is a secondary one, definable in terms of that of a canonical proof; and it is by reference to the notion of a canonical proof that the logical connectives are to be explained. (...)
>
> (Dummett 1977, p. 392.)

And his informal notion, with corresponding formal counterpart, of *canonical* (or *direct*) proofs is said to constitute a useful refinement of those previous semantical explanations (Gentzen, Heyting, Kolmogorov, Prawitz, etc.):

> Everything that I have said so far was said, or could have been said, already in the thirties. What made these explanations as given in the thirties lack precision somewhat, was the absence

of a clear distinction between what Dummett has called a canonical proof and a demonstration of a proposition, and which I shall call a direct as opposed to an indirect proof of a proposition. In explaining the meanings of the logical constants, I did not care whether a proof of a proposition was direct or indirect: nevertheless, we have to make that distinction, because what is explained in the meaning explanations of the logical constants, as I just gave them, is what constitutes a direct proof of a proposition formed by means of one of those constants.

(Martin-Löf 1987, pp. 412–3.)

It is clearly a refined explanation of meaning by *conditions for proof*, and the reasonably wide acceptance of its formal counterpart, *Intuitionistic Type Theory*, leaves little doubt here. As we have seen from previous chapters, by remaining congenial with the usual Gentzen-like explanations of meaning, Prof. Martin-Löf seems to assume essentially the same position as to the secondary rôle of the *consequences* aspect (the *use* aspect) of a proposition. What counts as a *direct* proof of a proposition is all one needs to know in order to grasp its meaning, no attention being primarily paid to the aspect of how such *direct* proof is to be used in proofs of other propositions. This much should be clear mainly from the discussion in the two previous chapters, and should hopefully become clearer from what follows below. It is certainly crucial in realising that Prof. Martin-Löf's explanation turns out to be profoundly different from Wittgenstein's. The point of view taken by the Austrian philosopher shows a clear emphasis on the aspects of *use*, *consequences*, *utility*, etc..

Following Heyting's conception of the logic of (intuitionistic) mathematics, where the meaning of logical connectives is given by *proof* conditions, the explanations provided by Prof. Martin-Löf in his *Intuitionistic Type Theory* constitute indeed a remarkable refinement of the previous semantical explanations based on the direct connections between meaning and proofs by making use of the notion of *direct*, *canonical* proofs, whose formal counterpart gives an important contribution to the clarification also of the *syntax* of the language of intuitionistic mathematics. The notation allows propositions and their proofs (in the sense of constructions, not of proof-trees, which finds a parallel in Prof. D. Scott's notion of mathematical construction developed in Scott 1970) to be written next to each other in a sensible way

denoting a *judgement*. The refinement from Heyting's semantical table is presented as:

a proof of the proposition:	has the (*canonical*) form of:
$\mathbf{A} \wedge \mathbf{B}$	a pair $\mathtt{pair}(a, b)$ where a is a proof of $\mathbf{A}$ and b is a proof of $\mathbf{B}$.
$\mathbf{A} \vee \mathbf{B}$	$\mathtt{inl}(a)$ where a is a proof of $\mathbf{A}$ or $\mathtt{inr}(b)$ where b is a proof of $\mathbf{B}$ ('inl' and 'inr' abbreviate 'into the left disjunct' and 'into the right disjunct', respectively).
$\mathbf{A} \rightarrow \mathbf{B}$	$(\lambda x)b(x)$ where $b(a)$ is a proof of $\mathbf{B}$ provided a is a proof of $\mathbf{A}$.
$(\forall x)\mathbf{B}(x)$	$(\lambda x)b(x)$ where $b(a)$ is a proof of $\mathbf{B}(a)$ provided a is an arbitrary individual.
$(\exists x)\mathbf{B}(x)$	a pair $\mathtt{pair}(a, b)$ where a is an individual (witness) and b is a proof of $\mathbf{B}(a)$.

Observe that the explanation of the form of canonical proofs for both implication ('$\rightarrow$') and universal quantifier ('$\forall$') appeal to the concepts of functionality and generality, resulting in both cases being presented with canonical proofs of the same general form, namely '$(\lambda x)b(x)$'. Similarly, both conjunction and existential quantification involve the notion of cartesian product, or ordered pairing. The general form of canonical proofs for both '$(\exists x)\mathbf{B}(x)$' and '$\mathbf{A} \wedge \mathbf{B}$' is that of '$\mathtt{pair}(a, b)$', or '$< a, b >$' if one prefers.

The formal counterpart to the informal explanation of canonical proofs for each logical connective would be found in the presentation of respective *introduction* rules. Moreover, the parallels between the general forms of canonical proofs, namely between implication and universal quantifier on the one hand, and conjunction and existential quantifier on the other hand, would seem to justify the presentation of 'general' types '$(\Pi x \in \mathbf{A})\mathbf{B}(x)$' and '$(\Sigma x \in \mathbf{A})\mathbf{B}(x)$', instead of the usual natural deduction presentations of logics, where even for the case of the intuitionistic interpretation the aforementioned parallels are not taken to be of primary importance, and the use of '$\forall$', '$\rightarrow$', '$\exists$', '$\wedge$', '$\vee$', etc., is directly made. In Prof. Martin-Löf's account of the logical connectives, however, the parallels between '$(\forall x)\mathbf{B}(x)$' and '$\mathbf{A} \rightarrow \mathbf{B}$', as well as between '$(\exists x)\mathbf{B}(x)$ and '$\mathbf{A} \wedge \mathbf{B}$', are definitely taken into consideration and the more 'general' types '$(\Pi x \in \mathbf{A})\mathbf{B}(x)$' and '$(\Sigma x \in \mathbf{A})\mathbf{B}(x)$' are presented as primary. The presentation of propositions such as '$(\forall x)\mathbf{B}(x)$' and '$\mathbf{A} \rightarrow \mathbf{B}$' are then taken to be particular instantiations of the general type '$(\Pi x \in \mathbf{A})\mathbf{B}(x)$'. Similarly, the presentations of propositions such as '$(\exists x)\mathbf{B}(x)$' and '$\mathbf{A} \wedge \mathbf{B}$' are obtained as particular instantiations of the general type '$(\Sigma x \in \mathbf{A})\mathbf{B}(x)$'.

Therefore, for the case of conjunction, a proposition of the form '$(\Sigma x \in \mathbf{A})\mathbf{B}(x)$' is defined as:

Σ-*introduction*

$$\frac{a \in \mathbf{A} \qquad b \in \mathbf{B}(a)}{\mathtt{pair}(a, b) \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}$$

which 'codes up' the proofs of propositions of the kind $\mathbf{A} \wedge \mathbf{B}$, and $(\exists x)\mathbf{B}(x)$ (intuitionistic interpretation, where there is a notion of 'witness').

From the aspect concerning the *assertability conditions* of a proposition of the form '$(\Sigma x \in \mathbf{A})\mathbf{B}(x)$' the canonical elements are defined to be ordered pairs of elements from $\mathbf{A}$ and $\mathbf{B}(a)[a \in \mathbf{A}]$. The aspect concerning the *utility* of these ordered pairs, which is taken to be a fundamental aspect of the semantic in my own

account of Type Theory and the meaning of logical constants, would be captured by the following *normalisation* rules:

Σ-*reduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{B}(a)}{\mathtt{FST}(\mathtt{pair}(a,b)) = a \in \mathbf{A}} \qquad \frac{a \in \mathbf{A} \quad b \in \mathbf{B}(a)}{\mathtt{SND}(\mathtt{pair}(a,b)) = b \in \mathbf{B}(a)}$$

The rules are basically indicating that one can recover either of the conjuncts from the proposition '$(\Sigma x \in \mathbf{A})\mathbf{B}(x)$', and the choice is made in the act of drawing the conclusions ('FST', 'SND'), as opposed to at the moment of the assertion, as we shall see below for the case of 'disjunction'.

For the case of propositions reflecting some kind of 'functionality', the general pattern is of the form '$(\Pi x \in \mathbf{A})\mathbf{B}(x)$', and is defined as:

Π-*introduction*

$$\frac{\begin{array}{c}[x \in \mathbf{A}] \\ b(x) \in \mathbf{B}(x)\end{array}}{(\lambda x)b(x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

which 'codes up' the proofs of propositions of the kind $\mathbf{A} \rightarrow \mathbf{B}$ and $(\forall x)\mathbf{B}(x)$.

The rule which deals with the aspect of the *utility* of the elements of the type $(\Pi x \in \mathbf{A})\mathbf{B}(x)$ is then framed as:

Π-*reduction*

$$\frac{a \in \mathbf{A} \qquad \begin{array}{c}[x \in \mathbf{A}] \\ b(x) \in \mathbf{B}(x)\end{array}}{\mathtt{APPLY}((\lambda x)b(x), a) = b(a/x) \in \mathbf{B}(a)}$$

It 'explains' the *utility* aspect by demonstrating that by applying a *canonical* proof of the proposition '$(\Pi x \in \mathbf{A})\mathbf{B}(x)$' to an *arbitrary* individual 'a' of type '$\mathbf{A}$'

one obtains a proof of '$\mathbf{B}(a)$', indicating the *generality* meaning of the proposition, and the notion of substitution which it seems to encapsulate.

The general schema for a proposition representing the disjunction of two propositions would be defined as:

+-introduction

$$\frac{a \in \mathbf{A}}{\mathtt{inl}(a) \in \mathbf{A} + \mathbf{B}} \qquad \frac{b \in \mathbf{B}}{\mathtt{inr}(b) \in \mathbf{A} + \mathbf{B}}$$

which are meant to code propositions of the kind $\mathbf{A} \vee \mathbf{B}$, and similarly to the previous cases, the *normalisation* rules complete the 'definition by use' by indicating the aspect of *utility* of the elements of the type in a way such as:

+-reduction

$$\frac{a \in \mathbf{A} \quad \begin{matrix}[x \in \mathbf{A}] \\ d \in \mathbf{C}(\mathtt{inl}(x))\end{matrix} \quad \begin{matrix}[y \in \mathbf{B}] \\ e \in \mathbf{C}(\mathtt{inr}(y))\end{matrix}}{\mathtt{WHEN}(\mathtt{inl}(a), d, e) = d(a) \in \mathbf{C}(\mathtt{inl}(a))} \qquad \frac{b \in \mathbf{B} \quad \begin{matrix}[x \in \mathbf{A}] \\ d \in \mathbf{C}(\mathtt{inl}(x))\end{matrix} \quad \begin{matrix}[y \in \mathbf{B}] \\ e \in \mathbf{C}(\mathtt{inr}(y))\end{matrix}}{\mathtt{WHEN}(\mathtt{inr}(b), d, e) = e(b) \in \mathbf{C}(\mathtt{inr}(b))}$$

No matter what choice of the disjunct one could draw from the disjunction '$\mathbf{A} + \mathbf{B}$', the conclusion one is allowed to draw from the proposition is already determined in the act of asserting it, as opposed to being decided in the act of drawing the conclusion such as was the case for the proposition '$(\Sigma x \in \mathbf{A})\mathbf{B}(x)$' above. Indeed, '`inl`' and '`inr`' are *canonical* operators, therefore introduced at the time of the assertion (in the *introduction* rules), as opposed to '`FST`' and '`SND`' which are non-canonical and introduced at the time of the demonstration of the consequences one is allowed to draw from the corresponding proposition (*elimination*, *reduction* rules).

A significant attempt to contribute to a more satisfactory account of meaning without any need to resort to the more usual world of interpretations, models, etc., has been offered by Prof. Martin-Löf's endeavour to formalise the notion of *utility*,

by means of the internalisation of the *normalisation* rules. The rules of *equality* (as the *reduction* rules are called in Prof. Martin-Löf's formulation of *Intuitionistic Type Theory*) are formulated as part of the formal system, but, nonetheless, they are not explained as fundamental rules as far as meaning of propositions is concerned. On the contrary, after giving the rules for the *assertability conditions* (the *introduction* rules), indicating the *canonical* proofs of the proposition, a certain formulation of *elimination* rules for each logical constant together with a verbal explanation of how the *elimination* rules work is provided. Such verbal explanation is justified as being some kind of 'meta-level' explanation of the 'computational' semantics associated with the body of ideas which support the formal system. From this verbal explanation, the *equality* rules are then said to become evident. As the result of a more careful look at the nature of those rules in relation to my own interpretation of Wittgenstein's teachings on the nature of meaning and 'the use of the sign in the language-game', I am led to say that the order of importance seems to have been inverted here. The rules of *normalisation* are just considered to have a less secondary rôle to play in such an endeavour to formalise the rules of constructive mathematics. They should naturally come first, as they constitute part of the meaning (use) of the particular connective by capturing its aspect of the *consequences*, *utility*, etc.. The verbal explanation would then come as the result of some individual's understanding.

5.3.4. Propositions and Types

The usual mathematical notion of set as a subset of the universe of values, which has proved adequate for most developments in the mathematics of the second half of this century, proved inadequate to support the notion of data type as understood by computer scientists. Instead of defining a type by what it contained, it was felt that a data type definition had to convey the information demonstrating how the elements of the type were to be used, computed, etc.. That was the view already put forward in 1973 by J. Morris in a paper entitled 'Types are not Sets'. He begins by saying:

> The title is not a statement of fact, of course, but an opinion about how language designers should think about types. There has been a natural tendency to look to mathematics for a consistent, precise notion of what types are. The point of view there is extensional: a type is a subset of the universe of values. While this approach may have served its purpose quite adequately in mathematics, defining programming language types in this way ignores some vital ideas.
>
> (Morris 1973, p. 120.)

A whole body of ideas developed mainly in the seventies aiming at gaining a better understanding of the concept of data types began to show that data type definitions were worth studying as a theory in its own right. With the help of algebraic techniques, the notion of data type as defined by what can be done with its elements proved a convenient alternative to the mathematical notion of set as defined by what the set contains. In the words of the ADJ Group, which was responsible for many of the seminal ideas on the mathematical characterisation of the notion of data types:

> At least since Morris, it seems to be generally accepted that types are not just sets, but sets equipped with operations, and that is *exactly* what many-sorted algebras are.
>
> (Thatcher, Wagner & Wright 1982, p. 711.)

Despite the breakthrough which was being made by such an alternative and rather convincing account of types as characterised by the operations that could be performed on their elements, the classical extensional view of mathematics proved stronger, and the notion of data types found a mathematical counterpart in the (extensional) notion of many-sorted algebras. An 'operational', or 'intentional' ingredient was introduced when the view on 'sets are characterised by their content' was challenged by 'types are characterised by their operational behaviour'. However, the strong traditional view of 'object and designation' and 'a sign denotes some object from a valuation domain' prevailed, and an equational specification of a type was looked at as a many-sorted algebra characterised by a signature (constants plus function symbols) and an axiomatics which denoted a canonical, initial, continuous, etc., algebra. The 'functions' listed in the signature characterising the particular algebra would not necessarily be describing the behaviour of the elements of the

type, but they could also be describing 'extensional' properties ('LENGTH' for the case of queues, lists, etc. – cf., e.g., the specification of 'queue' given in Ehrig & Mahr 1985, p. 55); 'HEIGHT' and 'BREADTH', for the case of trees – Ibid., pp. 56–7). By allowing the axiomatisation to have the form which could be general enough so that even functions describing 'extensional' properties would be part of the data type specification, many extra-problems were created such as, e.g., the completeness of the axiomatisation. If the 'functions' allowed in the axiomatisation do not play a definite (intentional) semantic rôle then it is not difficult to see that there cannot be any criteria to judge when a certain data type specification is complete, even if only a 'calculable' unformalised one. On the other hand, if those 'functions', or better, 'operations' as I prefer to call the constructors and destructors of a data type, are given as part of the process of explaining the (intentional) meaning of the type, the 'calculable' principle for deciding when a specification is complete could be such that when a particular axiomatisation demonstrates the effect of the destructors (TOP, POP for stacks) on the constructors (push) one can reasonably think of judging the axiomatisation complete. This can, of course, only be a 'calculable' principle and there is no question here of introducing a criterion for completeness in the formal sense.

One direct consequence of the so-called extensional view which has prevailed in most mathematical theories of data types is observed when one looks at how data type specifications are given 'mathematical meaning'. Obviously, if the axiomatics do not in principle carry any (intentional) meaning, then the quest for meaning is not settled until an interpretation in a 'domain of values' is given. And that is what happens in the case of the algebraic approach where if on the one hand the characterisation of types as many-sorted algebras is an acknowledgement of the insufficiency of the traditional notion of 'set', on the other hand a type specification is given meaning through an interpretation into extensional concepts such as initial algebras, continuous algebras, etc.. So, in summary, it seems fair to say that the chief semantical concern, even in the 'computer science' alternative for a formal characterisation of the concept of data type, has been mostly directed towards the

classical view that 'the quest for meaning is settled by denotations', 'meaning is a (numerical) value', and many difficulties were encountered to explain the meaning of oddly formed 'terms' such as the top of an empty stack, the front of an empty queue, etc..

Type Theory, an attempt to clarify the syntax and semantics of intuitionistic mathematics via notions such as *propositions*, *types* and *judgements*, developed almost simultaneously with the algebraic approaches to data types, came with a most interesting semantical innovation, which was the shift on the focus of attention from (numerical) values to *judgements*. Prior to the question of what the (numerical) value of an expression was, which nevertheless still remained but in a different level, there was a question of what kinds of *judgements* could be made from a type definition (or *presentation*). Such an interesting innovation would seem to have opened new horizons for a better understanding of what information should be taken as relevant in a type presentation. Many of the difficulties in giving a 'meaning' (as numerical value) to certain oddly formed terms would then be overcome. However, at the level of type presentations, *Type Theory* proved a step back to the mathematical principle that a set/type is to be defined by what it contains. This view was made clear on various occasions, such as, e.g., in:

> If we take seriously the idea that a proposition is defined by laying down how its canonical proofs are formed and accept that *a set is defined by prescribing how its canonical elements are formed*, then it is clear that it would only lead to unnecessary duplication to keep the notions of proposition and set (and the associated notions of proof of a proposition and element of a set) apart. Instead, we simply identify them, that is, treat them as one and the same notion. This is the formulae-as-types (propositions-as-sets) interpretation on which intuitionistic type theory is based.
>
> (Martin-Löf 1984, p. 13.)

(The *emphasis* is mine.)

In other opportunities such a view is expressed by explicit reference to the notion of data type in computer science, as in, e.g.,:

> When one speaks about data types in computer science, one might just as well say data sets.

(Martin-Löf 1984, p. 23.)

In spite of the attempt at providing a new perspective on the notion of meaning for the mathematical language, the insistence on the classical idea that meaning is something directly related to *content*, *object*, (*numerical*) *value*, etc., seems to prevent Prof. Martin-Löf's theory of meaning from being considered as a truly convincing alternative to classical (Frege) realist semantics. For Prof. Martin-Löf, types are given meaning by their content, where to say how to construct an element of the type is to give its meaning. When I say that the *reduction* (*normalisation*) rules should be taken as part of the semantics of the types/propositions, I make an attempt to provide an alternative to such a 'content'-based account of operational semantics offered by Prof. Martin-Löf, by saying that types should be defined by *what can be done with their elements*, and perhaps take a position which seems to be closer to the views of the great inspirer of an anti-interpretation and anti-realist account of semantics: Ludwig Wittgenstein. The rules of *reduction* are here taken to be a formal counterpart to Wittgenstein's informal notion of '*use* of the sign in the language-game'.

In spite of having its notation and most of its primitive concepts heavily based on Prof. P. Martin-Löf's *Intuitionistic Type Theory*, my *'Meaning-As-USE' Type Theory* relies on a radically distinct semantic theory (in the sense of Dummett 1982, p. 60.). Instead of saying that meaning is a semantic (numerical) value, the semantic theory which supports my own *Type Theory* says that meaning is no 'object' or (numerical, canonical) value but is made explicit by certain rules of *use/utility* such as the *normalisation* rules.

More than just a list of inference rules, my own *'Meaning-As-USE' Type Theory* should be seen as a general guideline prescribing how types, theories and logics can be presented in such a way that no appeal to *interpretations* or semantic numerical *values* is necessary in order for one to grasp the meaning of each type, theory, logical connective, etc.. By contrast to what seems to be understood by explanation of what it is that the formal system *Intuitionistic Type Theory* is actually offering,

there is no necessarily 'primitive' type in *'Meaning-As-USE' Type Theory*, and what is really given by the latter is a general pattern for presenting types/propositions[18]. In order to see the difference better, one can perhaps make a distinction between 'proof theory' and 'natural deduction as a body of definition/presentation tools' by saying that natural deduction is a mere body of instruments and guidelines as to how one should define/present logical systems, whereas proof theory is a *theory* about proofs. With my *'Meaning-As-USE' Type Theory* I attempt to bring the whole body of seminal (and indeed very useful!) ideas and formal tools developed by Prof. Martin-Löf in his *Intuitionistic Type Theory* to the point of being looked at as an 'enriched' natural deduction presentation tool whereby one can define/present types, propositions, logics, etc., in a uniform and disciplined way. For that purpose I had to modify the semantics of *Type Theory* from a canonical value-based semantics, which as we have seen from a few examples given above and shall be seeing more clearly in the next two chapters, is not adequate to deal with the notion of 'type' as understood by computer scientists, to a *normalisation*-based semantics.

For example, instead of presenting the logical connective of conjunction in the usual natural deduction style:

---

[18] *En passant*, an anonymous referee of one our papers reveals in his/her comments on our semantical standpoint how different a rôle the two *Type Theories* are supposed to play: whereas my *'Meaning-As-USE' Type Theory* is to be seen as a general guideline on how to present types, *Intuitionistic Type Theory* seems to be mostly regarded as a 'theory' containing certain 'primitive' types such as **Π**, **Σ**, **+**, **I**, **N**, **W**, **U**, etc., while other types such as, e.g., **Stack(A)**, **Queue(A)**, etc., are considered to be 'secondary', 'derived' types. In his/her own words: 'I have difficulty understanding the reasons given for changing the semantics in this way. It seems that the principal reason is the wish to include "algebraic data types" such as *queues* as primitive types in their type theory and not only as derived types using universes as in Nordström and Petersson's approach.' The latter is a reference to my criticism on the radically different way 'primitive' types (*formation*, *introduction*, *elimination*, *equality*) and abstract data types are presented via *Intuitionistic Type Theory* in section **17.5**, pp. 104–8 of Nordström, Petersson & Smith 1986. In the latter the specification of the 'non-primitive' type is given by means of algebraic equations 'syntactically loaded' with the notation from *Type Theory*. After all, what could be a reasonable principle supporting such a difference in treatment from 'primitive' and algebraic' types? They are just *types* altogether, are they not?

$\wedge$-*introduction*

$$\frac{\mathbf{A} \quad \mathbf{B}}{\mathbf{A} \wedge \mathbf{B}}$$

$\wedge$-*elimination*

$$\frac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{A}} \qquad \frac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{B}}$$

one has a wealth of additional tools to improve, refine, make more precise, the definition/presentation of logical conjunction, namely:

$\wedge$-*formation*

$$\frac{\mathbf{A}\ \mathit{type} \quad \mathbf{B}\ \mathit{type}}{\mathbf{A} \wedge \mathbf{B}\ \mathit{type}} \qquad \frac{\mathbf{A} = \mathbf{C} \quad \mathbf{B} = \mathbf{D}}{\mathbf{A} \wedge \mathbf{B} = \mathbf{C} \wedge \mathbf{D}}$$

$\wedge$-*introduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{B}}{\mathtt{pair}(a,b) \in \mathbf{A} \wedge \mathbf{B}} \qquad \frac{a = c \in \mathbf{A} \quad b = d \in \mathbf{B}}{\mathtt{pair}(a,b) = \mathtt{pair}(c,d) \in \mathbf{A} \wedge \mathbf{B}}$$

$\wedge$-*reduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{B}}{\mathtt{FST}(\mathtt{pair}(a,b)) = a \in \mathbf{A}} \qquad \frac{a \in \mathbf{A} \quad b \in \mathbf{B}}{\mathtt{SND}(\mathtt{pair}(a,b)) = b \in \mathbf{B}}$$

$\wedge$-*induction*

$$\frac{c \in \mathbf{A} \wedge \mathbf{B}}{\mathtt{pair}(\mathtt{FST}(c), \mathtt{SND}(c)) = c \in \mathbf{A} \wedge \mathbf{B}}$$

□

Indeed, to use the term 'wealth' is not mere exaggeration, for the latter is a much richer presentation of the logical connective of conjunction. In the former, with its *introduction* and *elimination* rules, only part of the fundamental aspects concerning the connective is made explicit, many of the other relevant aspects remaining at the 'meta'-level and left unformalised. The distinction between the *proposition* '**A**' and the *judgement* that '**A** is true' cannot be expressed in the style of the former presentation. On the other hand, one looks at the latter and see that first of all the grammatical aspects of both the proposition built up by the conjunctive connective as well as of its canonical proofs are clearly spelled out in the rules of *formation* and *introduction*. Furthermore, the aspect of *consequences*, *utility*, is, according to the semantics of use advocated in this report, formalised by the corresponding *reduction* rules, which as we shall see later on, finds an obvious parallel in the formalisation of logical constants by means of the notion of *semantical games* in Prof. J. Hintikka's *Game-Theoretical Semantics*. Last, but by no means least, the *minimality* aspect of the definition/presentation of conjunction is to find its formal counterpart in the *induction* rules (de Queiroz & Smyth 1989, and next chapter), aspect which is left unformalised in *Game-Theoretical Semantics*.

For these reasons I firmly believe that the next big step for such an extraordinary set of definitional tools as *Type Theory* is to allow itself to be looked at as an 'enriched' natural deduction presentation system, and to rid itself from its title of 'theory' (cf. the current status of what started as a theory of proofs such as Gentzen's natural deduction calculi and calculi of sequents, as well as Prof. Prawitz's improved natural deduction calculi). Such a remarkable step towards making proof theory a 'comfortable' language to express constructions that Prof. P. Martin-Löf has so brilliantly pioneered, cannot be overshadowed by the insistence that *Type Theory* is a collection of 'canonical' or 'primitive' types from which all other definable types can be instantiated.

5.3.4.1. Equality in Intuitionistic Type Theory

According to Martin-Löf 1984[19] there are four kinds of equality in Prof. Martin-Löf's *Intuitionistic Type Theory*, namely:

1. $\equiv$ or $=_{\text{def}}$
2. $\mathbf{A} = \mathbf{B}$
3. $a = b \in \mathbf{A}$
4. $\mathbf{I}(\mathbf{A}, a, b)$

The latter is introduced as a 'propositional' equality which would allow equality to be treated at the level of propositions, instead of at the level of *judgements* as in the cases of 2. and 3.. Because the equality in, e.g., '$a = b \in \mathbf{A}$', is treated at the level of a *judgement*, it is argued that (using Prof. Martin-Löf's own words) 'a form of propositional equality is nevertheless indispensable' given that one wants an equality which asserts that 'a' and 'b' are equal elements of the type '$\mathbf{A}$', but on which one can operate with the logical operations. Such a propositional equality, usually denoted by '$\mathbf{I}(\mathbf{A}, a, b)$' in the literature, would also allow arithmetic to be done in the framework of *Intuitionistic Type Theory*, where certain equalities are not to be seen as 'definitional' equalities but equality relations conveying sameness of values, such as, e.g., in '$2^2 = 2 + 2$'.

Following the general pattern of type presentations underlying *Intuitionistic Type Theory*, the proposition/type '$\mathbf{I}(\mathbf{A}, a, b)$' is presented as:

I-*formation*

$$\frac{\mathbf{A}\ \mathit{type} \qquad a \in \mathbf{A} \qquad b \in \mathbf{A}}{\mathbf{I}(\mathbf{A}, a, b)\ \mathit{type}}$$

I-*introduction*

$$\frac{a = b \in \mathbf{A}}{\mathtt{r} \in \mathbf{I}(\mathbf{A}, a, b)} \qquad \frac{a = b \in \mathbf{A}}{\mathtt{r} = \mathtt{r} \in \mathbf{I}(\mathbf{A}, a, b)}$$

[19]See chapter entitled 'Propositional equality', p. 59.

I-*elimination*

$$\frac{c \in \mathbf{I}(\mathbf{A}, a, b)}{a = b \in \mathbf{A}}$$

I-*equality*

$$\frac{c \in \mathbf{I}(\mathbf{A}, a, b)}{c = \mathrm{r} \in \mathbf{I}(\mathbf{A}, a, b)}$$

□

Definitional equality (abbr. '$\equiv$') is the main notion of equality used by Prof. Martin-Löf in his type-theoretic studies of mathematical logic (Martin-Löf 1975b). Even the rules of convertibility, such as the analogue of *λ-Calculus β-reduction*, are said to make use of a sole notion of equality, namely the notion of definitional (abbreviatory) equality[20]. It would seem fair to say that it does not seem very intuitive or natural to say that convertibility equalities should be called definitional equality for the latter involves mere abbreviation and is used when a symbol in the lefthand side is abbreviated to the 'complex' expression in the righthand side of the abbreviation sign '$\equiv$' (cf. discussion on Frege's twofold account of definitions – constructive and

[20]As a matter of fact, within the framework of proof-theoretic semantics the treatment of conversion rules as rules of definitional equality seems to find its origins in the seminal work of Prof. W. W. Tait on what he called 'intensional interpretations of functionals of finite type' (Tait 1967). In the absence of a more general conceptual framework where definitions as abbreviations would be fundamentally distinguished from definitions as presentations, he decides for using the terminology of definitional equality (which appears to be more generally identified with abbreviatory equality) to denote convertibility equality. In his own words:

'Lacking a general conception of the kinds of definitions by which an operation may be introduced, the notion of definitional equality is not very clear to me. But, if, as in the case of [Gödel's] T_i, we can regard the operations ϕ as being introduced by conversion rules

$$\phi t_1 ... t_n \Rightarrow s(t_1, ..., t_n),$$

then definitional equality has a clear meaning: s and t are *definitionally equal* if they reduce to a common term by means of a sequence of applications of the conversion rules.'

(Tait 1967, p. 198.)

analytic – in chapter 2 above). Nevertheless, one can of course start from this point, namely that of assuming every convertibility relation to be a definitional (abbreviatory) equality, and there cannot be anything absolutely wrong in that. The difficult part is to remain sound and coherent with this basic assumption throughout any serious study on mathematical theories of constructions (such as Church's λ-*Calculus*) and convertibility relations. It seems as though one is likely to run into serious conceptual difficulties and incur contradictions. In the meta-mathematical studies for the foundations of his *Type Theory*, e.g., Prof. Martin-Löf assumes that 'the formal counterpart of definitional equality is the relation of convertibility studied in combinatory logic and proof theory' (Martin-Löf 1975b, p. 93.) and runs into difficulties and incoherences. One of the primitive assumptions made in *Intuitionistic Type Theory* is that equality between terms (elements of types, proofs of propositions) is to be understood as definitional equality, and yet there are rules of equality which are explicitly ruled out from his notion of 'convertibility as definitional equality', such as, e.g., ξ- and η rules of equality between λ-terms[21]. Despite ruling them out as 'rules of conversion which do not correctly formalize the notion of definitional

[21] Cf.:

'**2. Definitional equality**

(...) The outcome of the foregoing analysis is that the rule ($\boxed{\xi}$) is unacceptable as a rule of conversion. Of course, we are free to define many different relations between terms and call them convertibility relations, but my claim is that only one of these correctly formalizes the informal notion of definitional equality. (...)

2.2. ***Further rules of conversion which do not correctly formalize the notion of definitional equality as understood in this paper***

2.2.1. Curry's rule of $\boxed{\eta}$-conversion,

$$(\lambda x)(b(x)) \text{ conv } b$$

provided the variable x does not occur free in the term b, and the combinatory axioms which correspond to it. (...)'

(Martin-Löf 1975b, p. 100.)

(The emphasis is mine.)

equality' as understood in Martin-Löf 1975b and certainly in all the other papers on *Intuitionistic Type Theory*, Prof. Martin-Löf insists that all but one equality in *Intuitionistic Type Theory* (namely, the 'propositional' equality '$\mathbf{I}(\mathbf{A}, a, b)$' presented above) are to be understood as 'definitional' equality, and yet both rules of ξ and η conversion are included in the formal apparatus of *Type Theory*. These fundamental rules of any calculus of λ-terms constitute part of the presentation of Π types, the former under Π-*introduction* and the latter under Π-*equality*.[22]

In abstract data type specifications, relations such as 'equal to', 'less than', 'less than or equal to', etc., are defined by an 'enrichment' of the theory with new extralogical symbols '*eq*', '*lt*', '*le*', etc., which return a take a pair of elements (numbers, e.g.) and return a truth-value. In Type Theory there is no such notion of 'truth-value', but the relations are defined as pairing types. Two elements a, b of a certain type are related if the pair (a, b) is an element of the type which corresponds to a type representing the desired relation.

Just as we might wish to add new symbols (or sorts) to a language, we might wish to attribute additional properties to (extralogical) symbols of the language. For instance, if L'_{nat} is the first-

[22]The conceptual uneasiness appears also in the presentation of *Intuitionistic Type Theory* by the Chalmers Group, Sweden, when both ξ and η are presented as definitional equalities between expressions with arities, where expressions are the object of study of *Type Theory*:

'**2.9 Definition of equality between two expressions of a certain arity**

(...)

5. *Abstraction 1.* (ξ-*rule*). If x is a variable of arity α and $b \equiv b' : \beta$, then

$$(x)b \equiv (x)b' : \alpha \twoheadrightarrow \beta$$

(...)

8. *Abstraction 3.* (η-*rule*). If x is a variable of arity β and b is an expression of arity $\alpha \twoheadrightarrow \beta$, then

$$(x)(b(x)) \equiv b : \alpha \twoheadrightarrow \beta$$

(...)'

(Nordström, Petersson & Smith 1986, pp. 15–16.)

order language with a diadic relation symbol *le*, we may have started by asserting the following properties for natural numbers with an ordering relation:

$$\vdash \neg(suc(n)=_{\text{nat}} zero) \tag{2.13}$$

$$\vdash suc(m)=_{\text{nat}} suc(n) \Rightarrow m=_{\text{nat}} n \tag{2.14}$$

$$\vdash le(n,n) \tag{2.15}$$

$$\vdash le(m,n)\&le(n,m) \Rightarrow m=_{\text{nat}} n \tag{2.16}$$

$$\vdash le(m,n)\&le(n,p) \Rightarrow le(m,p) \tag{2.17}$$

(Turski & Maibaum 1987, p. 74.)

Following the guidelines for writing down type-presentations, the type corresponding to the relation 'less than or equal to' over the type of natural numbers, formally abbreviated as e.g. '$\mathbf{LE}(\mathcal{N})$', would be presented as something like:

$\mathbf{LE}(\mathcal{N})$-*formation*

$$\frac{\mathcal{N}\ type}{\mathbf{LE}(\mathcal{N})\ type}$$

$\mathbf{LE}(\mathcal{N})$-*introduction*

$$\frac{a = b \in \mathcal{N}}{\mathtt{le}(a,b) \in \mathbf{LE}(\mathcal{N})} \qquad \frac{a_1 = b_1 \in \mathcal{N} \qquad a_2 = b_2 \in \mathcal{N}}{\mathtt{le}(a_1,a_2) = \mathtt{le}(b_1,b_2) \in \mathbf{LE}(\mathcal{N})}$$

$$\frac{a \in \mathcal{N}}{\mathtt{le}(0,a) \in \mathbf{LE}(\mathcal{N})} \qquad \frac{a \in \mathcal{N}}{\mathtt{le}(a,a) \in \mathbf{LE}(\mathcal{N})}$$

$$\frac{a \in \mathcal{N}}{\mathtt{le}(0,\mathtt{succ}(a)) \in \mathbf{LE}(\mathcal{N})} \qquad \frac{\mathtt{le}(a,b) \in \mathbf{LE}(\mathcal{N})}{\mathtt{le}(a,\mathtt{succ}(b)) \in \mathbf{LE}(\mathcal{N})}$$

$\mathbf{LE}(\mathcal{N})$-*reduction*

$$\frac{\mathtt{le}(a,b) \in \mathbf{LE}(\mathcal{N})}{\mathtt{SMALLER}(\mathtt{le}(a,b)) = a \in \mathcal{N}} \qquad \frac{\mathtt{le}(a,b) \in \mathbf{LE}(\mathcal{N})}{\mathtt{GREATER}(\mathtt{le}(a,b)) = b \in \mathcal{N}}$$

$\mathbf{LE}(\mathcal{N})$-*induction*

$$\frac{c \in \mathbf{LE}(\mathcal{N})}{\mathtt{le}(\mathtt{SMALLER}(c), \mathtt{GREATER}(c)) = c \in \mathbf{LE}(\mathcal{N})}$$

□

5.3.4.2. General Rules of Equality

In order to generalise the four kinds of *judgements* to handle what was described above as 'hypothetical *judgements*', which are *judgements* made under given assumptions, we need to introduce some general rules of equality and substitution as part of the framework of type-theoretic presentations of logical systems such as *Intuitionistic Type Theory.*

Equality rules are given as:

Reflexivity

$$\frac{a \in \mathbf{A}}{a = a \in \mathbf{A}} \qquad \frac{\mathbf{A}\ \mathit{type}}{\mathbf{A} = \mathbf{A}}$$

Symmetry

$$\frac{a = b \in \mathbf{A}}{b = a \in \mathbf{A}} \qquad \frac{\mathbf{A} = \mathbf{B}}{\mathbf{B} = \mathbf{A}}$$

Transitivity

$$\frac{a = b \in \mathbf{A} \qquad b = c \in \mathbf{A}}{a = c \in \mathbf{A}} \qquad \frac{\mathbf{A} = \mathbf{B} \qquad \mathbf{B} = \mathbf{C}}{\mathbf{A} = \mathbf{C}}$$

Equality of Types

$$\frac{a \in \mathbf{A} \qquad \mathbf{A} = \mathbf{B}}{a \in \mathbf{B}} \qquad \frac{a = b \in \mathbf{A} \qquad \mathbf{A} = \mathbf{B}}{a = b \in \mathbf{B}}$$

5.3.4.3. Hypothetical Judgements and Families of Types

In order to have *judgements* made under explicit assumptions, the forms of *judgement* given above are generalised, and certain general rules of substitution are given as part of the type-theoretic framework. The notion of hypothetical *judgement* gives rise to the concept of 'families of types' which, according to Martin-Löf 1975a, would be the type-theoretic counterpart to the notion of propositional function[23]. So, the first form of *judgement*, namely '**A** *type*', would be generalised to:

(1) $\mathbf{B}(x)$ *type* $[x \in \mathbf{A}]$

which would be saying that $\mathbf{B}(x)$ is a type under the assumption that x is an element of $\mathbf{A}$ ($x \in \mathbf{A}$).

The corresponding rules of *substitution* would be presented as:

$$\frac{a \in \mathbf{A} \qquad \begin{matrix}[x \in \mathbf{A}] \\ \mathbf{B}(x)\ type\end{matrix}}{\mathbf{B}(a)\ type} \qquad \frac{a = c \in \mathbf{A} \qquad \begin{matrix}[x \in \mathbf{A}] \\ \mathbf{B}(x)\ type\end{matrix}}{\mathbf{B}(a) = \mathbf{B}(c)}$$

The second form of *judgement*, namely '$\mathbf{A} = \mathbf{B}$' ('**A** and **B** are equal types'), would be generalised to:

(2) $\mathbf{B}(x) = \mathbf{D}(x)$ $[x \in \mathbf{A}]$

and the corresponding substitution rules:

$$\frac{a \in \mathbf{A} \qquad \begin{matrix}[x \in \mathbf{A}] \\ \mathbf{B}(x) = \mathbf{D}(x)\end{matrix}}{\mathbf{B}(a) = \mathbf{D}(a)} \qquad \frac{a = c \in \mathbf{A} \qquad \begin{matrix}[x \in \mathbf{A}] \\ \mathbf{B}(x) = \mathbf{D}(x)\end{matrix}}{\mathbf{B}(a) = \mathbf{D}(c)}$$

[23] 'A propositional function defined on a type A is called a *property* or, in intuitionistic terminology, a *species* of objects of type A. Because of the correspondence between propositions and types, it will sometimes be convenient to call a type valued function defined on A a species of objects of type A as well. Note that the notion of property is not a primitive notion: it is explained in terms of the notion of proposition (or type); and the notion of function.', p. 77 of Martin-Löf 1975a.

The third form of *judgement* — '$x \in \mathbf{A}$', informally 'x is an element/proof of $\mathbf{A}$' —, would be generalised to:

(3) $b(x) \in \mathbf{B}(x)$ $[x \in \mathbf{A}]$

and the respective substitution rules:

$$\frac{a \in \mathbf{A} \qquad \begin{matrix}[x \in \mathbf{A}] \\ b(x) \in \mathbf{B}(x)\end{matrix}}{b(a) \in \mathbf{B}(a)} \qquad \frac{a = c \in \mathbf{A} \qquad \begin{matrix}[x \in \mathbf{A}] \\ b(x) \in \mathbf{B}(x)\end{matrix}}{b(a) = b(c) \in \mathbf{B}(a)}$$

The last of the four forms of *judgement* explained above, namely '$a = c \in \mathbf{A}$', which would be informally explained as 'a and c are two equal elements/proofs of the type/proposition $\mathbf{A}$', would have its generalised form as:

(4) $b(x) = d(x) \in \mathbf{B}(x)$ $[x \in \mathbf{A}]$,

motivating the following substitution rule:

$$\frac{a \in \mathbf{A} \qquad \begin{matrix}[x \in \mathbf{A}] \\ b(x) = d(x) \in \mathbf{B}(x)\end{matrix}}{b(a) = d(a) \in \mathbf{B}(a)}$$

5.3.4.4. Finite Types

Finite subsets of Natural Numbers are presented in *Intuitionistic Type Theory* as primitive types:

$\mathbf{N}_n$-*formation*

$$\mathbf{N}_n \ type$$

$\mathbf{N}_n$-*introduction*

$$m_n \in \mathbf{N}_n \qquad (\text{m} = 0, 1, ..., \text{n-1})$$

N_n-*elimination*

$$\frac{c \in \mathrm{N}_n \qquad c_m \in \mathrm{C}(m_n) \qquad (m = 0, 1, ..., n-1)}{\mathrm{R}_n(c, c_0, ..., c_{n-1}) \in \mathrm{C}(c)}$$

N_n-*equality*

$$\frac{c_m \in \mathrm{C}(m_n) \qquad (m = 0, 1, ..., n-1)}{\mathrm{R}_n(m_n, c_0, ..., c_{n-1}) = c_m \in \mathrm{C}(m_n)}$$

□

An alternative approach to introducing the finite subsets of Natural Numbers as the Finite sets would be, according to Prof. Martin-Löf, 'to postulate the rules for n equal to 0 and 1 only, and define $\mathrm{N}_2 \equiv \mathrm{N}_1 + \mathrm{N}_1$, $\mathrm{N}_3 \equiv \mathrm{N}_1 + \mathrm{N}_2$, etc., and then derive all other rules' (Martin-Löf 1984, p. 66)[24]. We shall see in the next chapter how those finite subsets of inductively defined sets are given in such an alternative way using the higher-order type forming operator 'Fix'.

5.3.4.5. Cartesian Product of a Family of Types

The type of cartesian product of a family of types is presented as:

Π-*formation*

$$\frac{\mathrm{A}\ \textit{type} \qquad \mathrm{B}(x)\ \textit{type}\ [x \in \mathrm{A}]}{(\Pi x \in \mathrm{A})\mathrm{B}(x)\ \textit{type}} \qquad \frac{\mathrm{A} = \mathrm{C} \qquad \mathrm{B}(x) = \mathrm{D}(x)\ [x \in \mathrm{A}]}{(\Pi x \in \mathrm{A})\mathrm{B}(x) = (\Pi x \in \mathrm{C})\mathrm{D}(x)}$$

[24]This alternative approach was already pointed out in the earlier paper of 1973:

'1.8. *Finite types*

For each nonnegative integer n, we introduce a type $\mathrm{N_n}$ with precisely the n objects 1,2,...,n. Actually, it would suffice to introduce $\mathrm{N_0}$ and $\mathrm{N_1}$, because, for n greater than one, we can define $\mathrm{N_n}$ to be the disjoint union of $\mathrm{N_1}$ with itself n times.'

(Martin-Löf 1975a, p. 81.)

Π-*introduction*

$$\frac{\begin{array}{c}[x \in \mathbf{A}] \\ b(x) \in \mathbf{B}(x)\end{array}}{(\lambda x)b(x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)} \qquad \frac{\begin{array}{c}[x \in \mathbf{A}] \\ b(x) = d(x) \in \mathbf{B}(x)\end{array}}{(\lambda x)b(x) = (\lambda x)d(x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

Π-*elimination*

$$\frac{a \in \mathbf{A} \quad c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{APPLY}(c, a) \in \mathbf{B}(a)}$$

or, alternatively[25]:

$$\frac{c \in (\Pi x \in \mathbf{A})\mathbf{B}(x) \qquad \begin{array}{c}[y \in \mathbf{B}(x)[x \in \mathbf{A}]] \\ d(y) \in \mathbf{C}(\lambda(y))\end{array}}{\mathtt{F}(c, d) \in \mathbf{C}(c)}$$

Π-*equality*

$$\frac{\begin{array}{cc}& [x \in \mathbf{A}] \\ a \in \mathbf{A} & b(x) \in \mathbf{B}(x)\end{array}}{\mathtt{APPLY}((\lambda x)b(x), a) = b(a/x) \in \mathbf{B}(a)}$$

$$\frac{c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{(\lambda x)\mathtt{APPLY}(c, x) = c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

or, alternatively[26]:

$$\frac{\begin{array}{cc}[x \in \mathbf{A}] & [y \in \mathbf{B}(x)[x \in \mathbf{A}]] \\ b(x) \in \mathbf{B}(x) & d(y) \in \mathbf{C}(\lambda(y))\end{array}}{\mathtt{F}(\lambda(b), d) = d(b) \in \mathbf{C}(\lambda(b))}$$

□

[25]Given in the Preface of Martin-Löf 1984.

[26]Id. Ibid.

In some alternative presentations of the Π type, such as the one given in Nordström, Petersson & Smith 1986, the *eliminatory* operator 'F' is sometimes written 'FUNSPLIT' (function-splitting operator).

Another important remark about the 'Π' type is that, according to the intuitionistic interpretation of '$\forall$' and '$\rightarrow$', the type '$(\Pi x \in \mathbf{A})\mathbf{B}(x)$' happens to capture the notion of 'method' used in Heyting's explanation of these logical constants in terms of proofs, therefore having similar counterparts in the refined (semi)-formal explanation given by Prof. Martin-Löf through the concept of canonical proofs. In the latter, both '$(\forall x)\mathbf{B}(x)$' and '$\mathbf{A} \rightarrow \mathbf{B}$' have something like '$(\lambda x)b(x)$' as canonical proofs. For this reason, the type '$(\Pi x \in \mathbf{A})\mathbf{B}(x)$' is regarded as a generalisation of both types aforementioned. Moreover, together with the view that *Intuitionistic Type Theory* is regarded as a *theory* rather than a guideline plus a set of formal *instruments*, the 'Π' type is considered to be one of the 'canonical' types, so to speak, of the *theory* called *Intuitionistic Type Theory*. Therefore, in order to talk about universal quantification and implication in Type Theory, Prof. Martin-Löf has to 'define' these logical connectives as 'particular instantiations' of the type 'Π':

> First, using definitional equality, we can now define $\mathbf{B}^{\mathbf{A}}$ by putting
>
> $$\mathbf{B}^{\mathbf{A}} \equiv \mathbf{A} \rightarrow \mathbf{B} \equiv (\Pi x \in \mathbf{A})\mathbf{B},$$
>
> provided $\mathbf{B}$ does not depend on x. We next consider the Π-rules in the interpretation of propositions as sets. If, in the first rule, Π-formation, we think of $\mathbf{B}(x)$ as a proposition instead of a set, then, after the definition
>
> $$(\forall x \in \mathbf{A})\mathbf{B}(x) \equiv (\Pi x \in \mathbf{A})\mathbf{B}(x),$$
>
> it becomes the rule [of $\forall$-*formation*] (...)
>
> (Martin-Löf 1984, p. 32.)

5.3.4.6. Disjoint Union of a Family of Types

The type of disjoint union of a family of types is given as:

Σ-*formation*

$$\frac{\mathbf{A}\ type \quad \mathbf{B}(x)\ type}{(\Sigma x \in \mathbf{A})\mathbf{B}(x)\ type} \qquad \frac{\mathbf{A} = \mathbf{C} \quad \mathbf{B}(x) = \mathbf{D}(x)}{(\Sigma x \in \mathbf{A})\mathbf{B}(x) = (\Sigma x \in \mathbf{C})\mathbf{D}(x)}$$

Σ-*introduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{B}(a)}{< a, b > \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)} \qquad \frac{a = c \in \mathbf{A} \quad b = d \in \mathbf{B}(a)}{< a, b > = < c, d > \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}$$

Σ-*elimination*

$$\frac{c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x) \qquad \begin{matrix}[x \in \mathbf{A}, y \in \mathbf{B}(x)] \\ d(x,y) \in \mathbf{C}(< x, y >)\end{matrix}}{\mathrm{E}(c, (x,y)d(x,y)) \in \mathbf{C}(c)}$$

Σ-*equality*

$$\frac{a \in \mathbf{A} \qquad b \in \mathbf{B}(a) \qquad \begin{matrix}[x \in \mathbf{A}, y \in \mathbf{B}(x)] \\ d(x,y) \in \mathbf{C}(< x, y >)\end{matrix}}{\mathrm{E}(< a, b >, (x,y)d(x,y)) = d(a,b) \in \mathbf{C}(< a, b >)}$$

□

Similarly to the 'Π' case, the type 'Σ' is considered to be a sort of 'canonical' type of the *theory* which 'formalises' the intuitionistic interpretation of both the existential quantifier and the logical conjunction. So, in order to talk about the logical connectives of conjunction and existential quantification, one needs 'definitional equalities' as in the previous case. For the case of existential quantification it is defined as:

As we have already done with the cartesian product, we shall now see what are the logical interpretations of the disjoint union. If we put

$$(\exists x \in \mathbf{A})\mathbf{B}(x) \equiv (\Sigma x \in \mathbf{A})\mathbf{B}(x),$$

then, from the Σ-rules, interpreting $\mathbf{B}(x)$ as a propositional function over $\mathbf{A}$, we obtain as particular cases: [$\exists$-formation and $\exists$-introduction] (...)

(Martin-Löf 1984, p. 42.)

and conjunction is defined as:

The rules of disjoint union deliver also the usual rules of conjunction and the usual properties of the cartesian product of two sets if we define

$$\mathbf{A\&B} \equiv \mathbf{A} \times \mathbf{B} \equiv (\Sigma x \in \mathbf{A})\mathbf{B},$$

where $\mathbf{B}$ does not depend on x. (...)

(Martin-Löf 1984, p. 43.)

5.3.4.7. Disjoint Union of Two Types

The type of the disjoint union (sum) of two types is presented as:

+-formation

$$\frac{\mathbf{A}\ \mathit{type} \quad \mathbf{B}\ \mathit{type}}{\mathbf{A}+\mathbf{B}\ \mathit{type}} \qquad \frac{\mathbf{A}=\mathbf{C} \quad \mathbf{B}=\mathbf{D}}{\mathbf{A}+\mathbf{B}=\mathbf{C}+\mathbf{D}}$$

+-introduction

$$\frac{a \in \mathbf{A}}{\mathtt{i}(a) \in \mathbf{A}+\mathbf{B}} \qquad \frac{a=c \in \mathbf{A}}{\mathtt{i}(a)=\mathtt{i}(c) \in \mathbf{A}+\mathbf{B}}$$

$$\frac{b \in \mathbf{B}}{\mathtt{j}(b) \in \mathbf{A}+\mathbf{B}} \qquad \frac{b=d \in \mathbf{B}}{\mathtt{j}(b)=\mathtt{j}(d) \in \mathbf{A}+\mathbf{B}}$$

+-elimination

$$\frac{\begin{array}{cccc} & & [x \in \mathbf{A}] & [y \in \mathbf{B}] \\ c \in \mathbf{A}+\mathbf{B} & d \in \mathbf{C}(\mathtt{i}(x)) & e \in \mathbf{C}(\mathtt{j}(y)) \end{array}}{\mathtt{D}(c,(x)d,(y)e) \in \mathbf{C}(c)}$$

+-equality

$$\frac{\begin{array}{cccc} & & [x \in \mathbf{A}] & [y \in \mathbf{B}] \\ a \in \mathbf{A} & d \in \mathbf{C}(\mathtt{i}(x)) & e \in \mathbf{C}(\mathtt{j}(y)) \end{array}}{\mathtt{D}(\mathtt{i}(a),(x)d,(y)e)=d(a) \in \mathbf{C}(\mathtt{i}(a))} \qquad \frac{\begin{array}{cccc} & & [x \in \mathbf{A}] & [y \in \mathbf{B}] \\ b \in \mathbf{B} & d \in \mathbf{C}(\mathtt{i}(x)) & e \in \mathbf{C}(\mathtt{j}(y)) \end{array}}{\mathtt{D}(\mathtt{j}(b),(x)d,(y)e)=e(b) \in \mathbf{C}(\mathtt{j}(b))}$$

□

Again, similarly to the previous cases, the type/proposition '+' is another canonical type having the proposition corresponding to logical disjunction as a particular instantiation:

The disjunction of two propositions is now interpreted as the sum of two sets. We therefore put:

$$\mathbf{A} \vee \mathbf{B} \equiv \mathbf{A} + \mathbf{B}.$$

From the formation and introduction rules for +, we then obtain the corresponding rules for ∨ [∨-formation and ∨-introduction] (...)

(Martin-Löf 1984, p. 57.)

5.3.4.8. Natural Numbers

The type of natural numbers is presented in *Intuitionistic Type Theory* as follows:

N-*formation*

$$\mathbf{N}\ \textit{type}$$

N-*introduction*

$$0 \in \mathbf{N} \qquad \frac{a \in \mathbf{N}}{a' \in \mathbf{N}}$$

N-*elimination*

$$\frac{c \in \mathbf{N} \qquad d \in \mathbf{C}(0) \qquad \begin{matrix}[x \in \mathbf{N}, y \in \mathbf{C}(x)] \\ e(x,y) \in \mathbf{C}(x')\end{matrix}}{\mathrm{R}(c, d, (x,y)e(x,y)) \in \mathbf{C}(c)}$$

N-*equality*

$$\frac{d \in \mathrm{C}(0) \qquad \begin{array}{c}[x \in \mathrm{N}, y \in \mathrm{C}(x)] \\ e(x,y) \in \mathrm{C}(x')\end{array}}{\mathrm{R}(0,d,(x,y)e(x,y)) = d \in \mathrm{C}(0)}$$

$$\frac{a \in \mathrm{N} \qquad d \in \mathrm{C}(0) \qquad \begin{array}{c}[x \in \mathrm{N}, y \in \mathrm{C}(x)] \\ e(x,y) \in \mathrm{C}(x')\end{array}}{\mathrm{R}(a',d,(x,y)e(x,y)) = e(a,\mathrm{R}(a,d,(x,y)e(x,y))) \in \mathrm{C}(a')}$$

□

5.3.4.9. Lists

The type of 'lists of items' is presented in *Intuitionistic Type Theory* in a rather different form than the usual computing science, abstract data type, manner (cf. the usual LISP-list presentation through 'cons' *and* 'CAR-CDR'). Due to the semantics based on 'canonical values', where the actual *normalisation* does not play any rôle in giving meaning to a type (the type is given meaning by its *introduction* rules which show how to form the canonical elements of the type), the type of lists is not presented through its usual operations 'CAR-CDR' or 'HEAD-TAIL', but only the rule demonstrating how to form the canonical values of the type is exhibited, following the semantical principle that the meaning of a type is given by its canonical elements. Such a semantical standpoint does not sound too convincing for a computer scientist. One has to make a sharp distinction between the type of 'queues of items', which is 'destroyed', 'computed' via the discipline 'first-in-first-out', and the type of 'lists of items' which is computed through a rather distinct discipline 'last-in-first-out', despite both having 'isomorphic' canonical elements: both contain sequences of items, but are only recognised to be either 'lists' or 'queues' as soon as the *normalisation* of non-canonical elements is shown. The canonical element by itself does not make one see the difference, apart from a judicious choice of 'names' ('cons' for lists, as opposed to 'insertinqueue' for queues.) Looking at (algebraic) specifications of the two types one realises that the way a non-canonical element of the type 'list of items' is *normalised* via its DESTRUCTORS 'CAR' and 'CDR' is very different from the way a non-canonical element of the type 'queue of items' is *normalised* by means of

its DESTRUCTORS 'FRONT' and 'REMOVE' (see algebraic specification of queue later in this same chapter, taken from Ehrig & Mahr 1985.).[27]

The presentation of the type '**List(A)**' usually given in *Intuitionistic Type Theory* is similar to the following presentation (where '$a.b$' is used instead of '$\mathtt{cons}(a, b)$'):

List-*formation*

$$\frac{\mathbf{A}\ type}{\mathbf{List(A)}\ type} \qquad \frac{\mathbf{A} = \mathbf{B}}{\mathbf{List(A)} = \mathbf{List(B)}}$$

List-*introduction*

$$\mathtt{nil} \in \mathbf{List(A)} \qquad \mathtt{nil} = \mathtt{nil} \in \mathbf{List(A)}$$

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{List(A)}}{a.b \in \mathbf{List(A)}} \qquad \frac{a = c \in \mathbf{A} \quad b = d \in \mathbf{List(A)}}{a.b = c.d \in \mathbf{List(A)}}$$

[27]Cf. J. Guttag's explanation of the axiomatic approach to data type definitions using the same example of the data type 'queue of items', where the point is made as to the insufficiency of the 'syntactic' information given by procedures to form the 'canonical' values of types:

'(...) One could, for example, define a type Queue (of Items) with the operations

NEW: → Queue
ADD: Queue × Item → Queue
FRONT: Queue → Item
REMOVE: Queue → Queue
IS-EMPTY?: Queue → Boolean

Unfortunately, however, short of supplying a representation, the only mechanism for denoting what these operations "mean" is a judicious choice of names. Except for intuitions about the meaning of such words as Queue and FRONT, the operations might just as well be defining type Stack as type Queue. (...)'

(Guttag 1977, p. 397.)

(The emphasis is mine.)

List-*elimination*

$$\frac{c \in \mathbf{List}(\mathbf{A}) \qquad d \in \mathbf{C}(\mathtt{nil}) \qquad \begin{matrix}[x \in \mathbf{A}, y \in \mathbf{List}(\mathbf{A}), z \in \mathbf{C}(y)] \\ e(x,y,z) \in \mathbf{C}(x.y)\end{matrix}}{\mathtt{LISTREC}(c, d, (x,y,z)e(x,y,z)) \in \mathbf{C}(c)}$$

List-*equality*

$$\frac{d \in \mathbf{C}(\mathtt{nil}) \qquad \begin{matrix}[x \in \mathbf{A}, y \in \mathbf{List}(\mathbf{A}), z \in \mathbf{C}(y)] \\ e(x,y,z) \in \mathbf{C}(x.y)\end{matrix}}{\mathtt{LISTREC}(\mathtt{nil}, d, (x,y,z)e(x,y,z)) = d \in \mathbf{C}(\mathtt{nil})}$$

$$\frac{a \in \mathbf{A} \qquad b \in \mathbf{List}(\mathbf{A}) \qquad d \in \mathbf{C}(\mathtt{nil}) \qquad \begin{matrix}[x \in \mathbf{A}, y \in \mathbf{List}(\mathbf{A}), z \in \mathbf{C}(y)] \\ e(x,y,z) \in \mathbf{C}(x.y)\end{matrix}}{\mathtt{LISTREC}(a.b, d, (x,y,z)e(x,y,z)) = e(a, b, \mathtt{LISTREC}(b, d, (x,y,z)e(x,y,z))) \in \mathbf{C}(a.b)}$$

□

5.3.4.10. Wellorderings

There are essentially two different presentations of **W** types both from the same semantical standpoint, namely that of Prof. G. Sambin's notes for Prof. Martin-Löf's lectures given in Padova in 1980 (Martin-Löf 1984) and that of an introductory book on *Intuitionistic Type Theory* by B. Nordström, K. Petersson and J. Smith (Nordrström, Petersson & Smith 1986):

W-*formation*

$$\frac{\mathbf{A}\ \mathit{type} \qquad \begin{matrix}[x \in \mathbf{A}] \\ \mathbf{B}(x)\ \mathit{type}\end{matrix}}{(\mathbf{W}x \in \mathbf{A})\mathbf{B}(x)\ \mathit{type}}$$

W-*introduction*

$$\frac{a \in \mathbf{A} \qquad b \in \mathbf{B}(a) \to (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x)}{\mathrm{sup}(a,b) \in (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x)}$$

Observe that in spite of the inductive character of the type **W**, a base or bottom case is not given (just think of the natural numbers without '0'!). Prof. Martin-Löf justifies the absence of a bottom case in the *introduction* rules of **W** by saying that such '$0 \in (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x)$' can be obtained by taking one type in the family of types $\mathbf{B}(x)[x \in \mathbf{A}]$ to be the empty type. Therefore, according to Prof. Martin-Löf, if $a_0 \in \mathbf{A}$ and $\mathbf{B}(a_0) = \mathbf{N}_0$, then $\mathtt{R}_0(y) \in (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x)[y \in \mathbf{B}(a_0)]$ so that $\sup(a_0, (\lambda y)\mathtt{R}_0(y)) \in (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x)$ is a bottom element.

As we shall see below, the presentation given by the Chalmers Group looks more like one might expect of the presentation of an inductive type by having the bottom clause explicitly given.

The presentation of the **W**-type is completed by Prof. Martin-Löf as:

W-*elimination*

$$\frac{c \in (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x) \qquad \begin{array}{c}[x \in \mathbf{A}, y \in \mathbf{B}(x) \to (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x), z \in (\Pi v \in \mathbf{B}(x))\mathbf{C}(\mathtt{Ap}(y,v))] \\ d(x,y,z) \in \mathbf{C}(\sup(x,y))\end{array}}{\mathtt{T}(c,(x,y,z)d(x,y,z)) \in \mathbf{C}(c)}$$

W-*equality*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{B}(a) \to (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x) \qquad \begin{array}{c}[x \in \mathbf{A}, y \in \mathbf{B}(x) \to (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x), z \in (\Pi v \in \mathbf{B}(x))\mathbf{C}(\mathtt{Ap}(y,v))] \\ d(x,y,z) \in \mathbf{C}(\sup(x,y))\end{array}}{\mathtt{T}(\sup(a,b),(x,y,z)d(x,y,z)) = d(a,b,(\lambda v)\mathtt{T}(\mathtt{Ap}(b,v),(x,y,z)d(x,y,z))) \in \mathbf{C}(\sup(a,b))}$$

□

The presentation given by the Programming Methodology Group of Chalmers is in its turn given in Nordström, Petersson & Smith 1986 as:

W-*formation*

$$\frac{\mathbf{A}\ \textit{type} \quad \begin{array}{c}[x \in \mathbf{A}] \\ \mathbf{B}(x)\ \textit{type}\end{array}}{(\mathbf{W}x \in \mathbf{A})\mathbf{B}(x)\ \textit{type}} \qquad \frac{\mathbf{A}_1 = \mathbf{A}_2 \quad \begin{array}{c}[x \in \mathbf{A}_1] \\ \mathbf{B}_1(x) = \mathbf{B}_2(x)\end{array}}{(\mathbf{W}x \in \mathbf{A}_1)\mathbf{B}_1(x) = (\mathbf{W}x \in \mathbf{A}_2)\mathbf{B}_2(x)}$$

W-*introduction*

$$\nabla \in (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x) \qquad \nabla = \nabla \in (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x)$$

$$\frac{a \in \mathbf{A} \quad b(x) \in (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x)}{\sup(a,b) \in (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x)}[x \in \mathbf{B}(a)] \qquad \frac{a_1 = a_2 \in \mathbf{A} \quad b_1(x) = b_2(x) \in (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x)}{\sup(a_1,b_1) = \sup(a_2,b_2) \in (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x)}[x \in \mathbf{B}(a_1)]$$

(In passing, observe that here the bottom clause is given explicitly as '∇', similarly to A. Church's use of '0_0' as the bottom clause for the constructive second number class (Church 1938, p. 228.).)

W-*elimination*

$$\frac{a \in (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x) \quad b \in \mathbf{C}(\nabla) \qquad \begin{matrix}[y\in\mathbf{A},z(x)\in(\mathbf{W}x\in\mathbf{A})\mathbf{B}(x)[x\in\mathbf{B}(y)],u(x)\in\mathbf{C}(z(x))[x\in\mathbf{B}(y)]] \\ c(y,z,u) \in \mathbf{C}(\sup(y,z))\end{matrix}}{\mathtt{W}-\mathrm{REC}(a,b,c) \in \mathbf{C}(a)}$$

$$\frac{a_1=a_2\in(\mathbf{W}x\in\mathbf{A})\mathbf{B}(x) \quad b_1=b_2\in\mathbf{C}(\nabla) \qquad \begin{matrix}[y\in\mathbf{A},z(x)\in(\mathbf{W}x\in\mathbf{A})\mathbf{B}(x)[x\in\mathbf{B}(y)],u(x)\in\mathbf{C}(z(x))[x\in\mathbf{B}(y)]] \\ c_1(y,z,u)=c_2(y,z,u)\in\mathbf{C}(\sup(y,z))\end{matrix}}{\mathtt{W}-\mathrm{REC}(a_1,b_1,c_1)=\mathtt{W}-\mathrm{REC}(a_2,b_2,c_2)\in\mathbf{C}(a_1)}$$

W-*equality*

$$\frac{b \in \mathbf{C}(\nabla) \qquad \begin{matrix}[y \in \mathbf{A}, z(x) \in (\mathbf{W}x \in \mathbf{A})\mathbf{B}(x)[x \in \mathbf{B}(y)], u(x) \in \mathbf{C}(z(x))[x \in \mathbf{B}(y)]] \\ c(y,z,u) \in \mathbf{C}(\sup(y,z))\end{matrix}}{\mathtt{W}-\mathrm{REC}(\nabla,b,c) = b \in \mathbf{C}(\nabla)}$$

$$\frac{d\in\mathbf{A} \quad \begin{matrix}[x\in\mathbf{B}(d)] \\ e(x)\in(\mathbf{W}x\in\mathbf{A})\mathbf{B}(x)\end{matrix} \quad b\in\mathbf{C}(\nabla) \qquad \begin{matrix}[y\in\mathbf{A},z(x)\in(\mathbf{W}x\in\mathbf{A})\mathbf{B}(x)[x\in\mathbf{B}(y)],u(x)\in\mathbf{C}(z(x))[x\in\mathbf{B}(y)]] \\ c(y,z,u)\in\mathbf{C}(\sup(y,z))\end{matrix}}{\mathtt{W}-\mathrm{REC}(\sup(d,e),b,c) = c(d,e,(x)\mathtt{W}-\mathrm{REC}(e(x),b,c)) \in \mathbf{C}(\sup(d,e))}$$

□

5.3.4.11. Universes

Having made precise the concept of *type* via the notions of canonical elements, non-canonical elements, normalisation and induction, the second-order notion of *type of types*, where types themselves would be treated as elements of a 'bigger' type, still remained to be explained/formalised. Finding inspiration in the somewhat formal techniques for providing 'modern' counterparts to Zermelo-Fraenkel's replacement axiom developed in category theory (sets of sets), Prof. Martin-Löf introduced the notion of 'universes' which would 'link the generation of objects and types' and would also 'make the [intuitionistic type] theory adequate for the formalisation of certain constructions in category theory, like the construction of the category of all small categories' (p. 73 of Martin-Löf 1975a).

Two presentations of the 'type of small types' are given in Martin-Löf 1984: one formulated '*à la* Tarski', and one '*à la* Russell'. The formulation '*à la* Tarski' is given as follows:

U-*formation*

$$\mathbf{U}\ \textit{type} \qquad \frac{\mathrm{a} \in \mathbf{U}}{\mathrm{T(a)}\ \textit{type}}$$

where **U** and T(x)[x ∈ **U**] are said to be defined by a simultaneous transfinite induction, and according to the semantics of *Intuitionistic Type Theory*, can be read off the following introduction rules:

U-*introduction*

$$\frac{\begin{array}{c}[\mathrm{x} \in \mathrm{T(a)}] \\ \mathrm{a} \in \mathbf{U} \quad \mathrm{b(x)} \in \mathbf{U}\end{array}}{\pi(\mathrm{a}, (\mathrm{x})\mathrm{b(x)}) \in \mathbf{U}} \qquad \frac{\begin{array}{c}[\mathrm{x} \in \mathrm{T(a)}] \\ \mathrm{a} \in \mathbf{U} \quad \mathrm{b(x)} \in \mathbf{U}\end{array}}{\mathrm{T}(\pi(\mathrm{a}, (\mathrm{x})\mathrm{b(x)})) = (\mathbf{\Pi x} \in \mathrm{T(a)})\mathrm{T(b(x))}}$$

$$\frac{\begin{array}{c}[\mathrm{x} \in \mathrm{T(a)}] \\ \mathrm{a} \in \mathbf{U} \quad \mathrm{b(x)} \in \mathbf{U}\end{array}}{\sigma(\mathrm{a}, (\mathrm{x})\mathrm{b(x)}) \in \mathbf{U}} \qquad \frac{\begin{array}{c}[\mathrm{x} \in \mathrm{T(a)}] \\ \mathrm{a} \in \mathbf{U} \quad \mathrm{b(x)} \in \mathbf{U}\end{array}}{\mathrm{T}(\sigma(\mathrm{a}, (\mathrm{x})\mathrm{b(x)})) = (\mathbf{\Sigma x} \in \mathrm{T(a)})\mathrm{T(b(x))}}$$

$$\frac{a \in U \quad b \in U}{a + b \in U} \qquad \frac{a \in U \quad b \in U}{T(a + b) = T(a) + T(b)}$$

$$\frac{a \in U \quad b \in T(a) \quad c \in T(a)}{i(a, b, c) \in U} \qquad \frac{a \in U \quad b \in T(a) \quad c \in T(a)}{T(i(a, b, c)) = I(T(a), b, c)}$$

$$n_0 \in U \quad n_1 \in U \quad \cdots \quad n \in U \qquad T(n_0) = N_0 \quad T(n_1) = N_1 \quad \cdots \quad T(n) = N$$

$$\frac{a \in U \quad \begin{matrix}[x \in T(a)] \\ b(x) \in U\end{matrix}}{w(a, (x)b(x)) \in U} \qquad \frac{\begin{matrix}[x \in T(a)] \\ a \in U \quad b(x) \in U\end{matrix}}{T(w(a, (x)b(x))) = (Wx \in T(a))T(b(x))}$$

□

Using Prof. Martin-Löf,s own words, in the formulation '*à la* Russell' one considers Π, Σ, ..., both as type forming operations and as operations to form canonical elements of the type **U**, the universe, making it look like B. Russell's ramified type theory. In this formulation, the 'decodifier' 'T' disappears and the types themselves, instead of their 'code' written in lower case letters, are said to be in the universe **U**. Furthermore, one says that a 'type **A** is small, or a **U**-type, if it has a code $a \in U$ such that $T(a) = A$' (p. 90 of Martin-Löf 1984). The presentation of the universe in the formulation '*à la* Russell' is the following:

U-*formation*

$$U \textit{ type} \qquad \frac{A \in U}{A \textit{ type}}$$

U-*introduction*

$$\frac{A \in U \quad \begin{matrix}[x \in A] \\ B(x) \in U\end{matrix}}{(\Pi x \in A)B(x) \in U} \qquad \frac{A \in U \quad \begin{matrix}[x \in A] \\ B(x) \in U\end{matrix}}{(\Sigma x \in A)B(x) \in U}$$

$$\frac{\mathbf{A} \in \mathbf{U} \quad \mathbf{B} \in \mathbf{U}}{\mathbf{A} + \mathbf{B} \in \mathbf{U}} \qquad \frac{\mathbf{A} \in \mathbf{U} \quad b, c \in \mathbf{A}}{\mathbf{I}(\mathbf{A}, b, c) \in \mathbf{U}}$$

$$\mathbf{N}_0 \in \mathbf{U} \qquad \mathbf{N}_1 \in \mathbf{U} \qquad \cdots \qquad \mathbf{N} \in \mathbf{U}$$

$$\frac{\mathbf{A} \in \mathbf{U} \quad \begin{matrix}[x \in \mathbf{A}] \\ \mathbf{B}(x) \in \mathbf{U}\end{matrix}}{(\mathbf{W}x \in \mathbf{A})\mathbf{B}(x) \in \mathbf{U}}$$

□

In Nordström, Petersson & Smith 1986, the 'type of small types (the first universe)'[28] is presented as:

U-*formation* 1:

$$\mathbf{U}\ \textit{type}$$

U-*introduction* 1:

$$\overline{\{i_1, \cdots, i_n\}} \in \mathbf{U}$$

$$\mathbf{Type}(\overline{\{i_1, \cdots, i_n\}}) = \{i_1, \cdots, i_n\}$$

U-*introduction* 2:

$$\overline{\mathbf{N}} \in \mathbf{U}$$

$$\mathbf{Type}(\overline{\mathbf{N}}) = \mathbf{N}$$

U-*introduction* 3:

[28] *Sic.*

$$\frac{\mathbf{A} \in \mathbf{U}}{\overline{\mathbf{List}}(\mathbf{A}) \in \mathbf{U}}$$

$$\frac{\mathbf{A} = \mathbf{B} \in \mathbf{U}}{\overline{\mathbf{List}}(\mathbf{A}) = \overline{\mathbf{List}}(\mathbf{B}) \in \mathbf{U}}$$

$$\mathbf{Type}(\overline{\mathbf{List}}(\mathbf{A})) = \mathbf{List}(\mathbf{Type}(\mathbf{A}))$$

U-*introduction* 4:

$$\frac{\mathbf{A} \in \mathbf{U} \qquad a \in \mathbf{Type}(\mathbf{A}) \qquad b \in \mathbf{Type}(\mathbf{A})}{\overline{\mathbf{I}}(\mathbf{A}, a, b) \in \mathbf{U}}$$

$$\frac{\mathbf{A}_1 = \mathbf{A}_2 \in \mathbf{U} \qquad a_1 = a_2 \in \mathbf{Type}(\mathbf{A}_1) \qquad b_1 = b_2 \in \mathbf{Type}(\mathbf{A}_1)}{\overline{\mathbf{I}}(\mathbf{A}_1, a_1, b_1) = \overline{\mathbf{I}}(\mathbf{A}_2, a_2, b_2) \in \mathbf{U}}$$

$$\mathbf{Type}(\overline{\mathbf{I}}(\mathbf{A}, a, b)) = \mathbf{I}(\mathbf{Type}(\mathbf{A}), a, b)$$

(*En passant*, looking at the second rule of '**U**-*introduction* 4', it seems as though it would be more sensibly formulated as:

$$\frac{\mathbf{A}_1 = \mathbf{A}_2 \in \mathbf{U} \qquad a_1 = b_1 \in \mathbf{Type}(\mathbf{A}_1) \qquad a_2 = b_2 \in \mathbf{Type}(\mathbf{A}_2)}{\overline{\mathbf{I}}(\mathbf{A}_1, a_1, b_1) = \overline{\mathbf{I}}(\mathbf{A}_2, a_2, b_2) \in \mathbf{U}}$$

This is just because, according to the rules of **I**-*formation*, none of the types '$\overline{\mathbf{I}}(\mathbf{A}_1, a_1, b_1)$' and '$\overline{\mathbf{I}}(\mathbf{A}_2, a_2, b_2)$' would be correctly formed type-expressions had not a_1 and b_1 been said to belong to the same type $\mathbf{Type}(\mathbf{A}_1)$ (a_2 and b_2, $\mathbf{Type}(\mathbf{A}_2)$, respectively.).

U-*introduction* 5:

$$\frac{\mathbf{A} \in \mathbf{U} \qquad \mathbf{B} \in \mathbf{U}}{\mathbf{A} \overline{+} \mathbf{B} \in \mathbf{U}}$$

$$\frac{\mathbf{A}_1 = \mathbf{A}_2 \in \mathbf{U} \qquad \mathbf{B}_1 = \mathbf{B}_2 \in \mathbf{U}}{\mathbf{A}_1 \overline{+} \mathbf{B}_1 = \mathbf{A}_2 \overline{+} \mathbf{B}_2 \in \mathbf{U}}$$

$$\mathbf{Type}(\mathbf{A}\overline{+}\mathbf{B}) = \mathbf{Type}(\mathbf{A}) + \mathbf{Type}(\mathbf{B})$$

U-*introduction* 6:

$$\frac{\mathbf{A} \in \mathbf{U} \qquad \begin{array}{c}[x \in \mathbf{Type}(\mathbf{A})] \\ \mathbf{B}(x) \in \mathbf{U}\end{array}}{(\overline{\Pi} x \in \mathbf{A})\mathbf{B}(x) \in \mathbf{U}}$$

$$\frac{\mathbf{A}_1 = \mathbf{A}_2 \in \mathbf{U} \quad \begin{array}{c}[x \in \mathbf{Type}(\mathbf{A}_1)] \\ \mathbf{B}_1(x) = \mathbf{B}_2(x) \in \mathbf{U}\end{array}}{(\overline{\Pi} x \in \mathbf{A}_1)\mathbf{B}_1(x) = (\overline{\Pi} x \in \mathbf{A}_2)\mathbf{B}_2(x) \in \mathbf{U}}$$

$$\mathbf{Type}((\overline{\Pi} x \in \mathbf{A})\mathbf{B}(x)) = (\Pi x \in \mathbf{Type}(\mathbf{A}))\mathbf{Type}(\mathbf{B}(x))$$

U-*introduction* 7:

$$\frac{\mathbf{A} \in \mathbf{U} \qquad \begin{array}{c}[x \in \mathbf{Type}(\mathbf{A})] \\ \mathbf{B}(x) \in \mathbf{U}\end{array}}{(\overline{\Sigma} x \in \mathbf{A})\mathbf{B}(x) \in \mathbf{U}}$$

$$\frac{\mathbf{A}_1 = \mathbf{A}_2 \in \mathbf{U} \quad \begin{array}{c}[x \in \mathbf{Type}(\mathbf{A}_1)] \\ \mathbf{B}_1(x) = \mathbf{B}_2(x) \in \mathbf{U}\end{array}}{(\overline{\Sigma} x \in \mathbf{A}_1)\mathbf{B}_1(x) = (\overline{\Sigma} x \in \mathbf{A}_2)\mathbf{B}_2(x) \in \mathbf{U}}$$

$$\mathbf{Type}((\overline{\Sigma} x \in \mathbf{A})\mathbf{B}(x)) = (\Sigma x \in \mathbf{Type}(\mathbf{A}))\mathbf{Type}(\mathbf{B}(x))$$

U-*introduction* 8:

$$\frac{\mathbf{A} \in \mathbf{U} \qquad \begin{array}{c}[x \in \mathbf{Type}(\mathbf{A})] \\ \mathbf{B}(x) \in \mathbf{U}\end{array}}{(\overline{\mathbf{W}} x \in \mathbf{A})\mathbf{B}(x) \in \mathbf{U}}$$

$$\frac{\mathbf{A}_1 = \mathbf{A}_2 \in \mathbf{U} \quad \begin{array}{c}[x \in \mathbf{Type}(\mathbf{A}_1)] \\ \mathbf{B}_1(x) = \mathbf{B}_2(x) \in \mathbf{U}\end{array}}{(\overline{\mathbf{W}} x \in \mathbf{A}_1)\mathbf{B}_1(x) = (\overline{\mathbf{W}} x \in \mathbf{A}_2)\mathbf{B}_2(x) \in \mathbf{U}}$$

$$\mathbf{Type}((\overline{\mathbf{W}} x \in \mathbf{A})\mathbf{B}(x)) = (\mathbf{W} x \in \mathbf{Type}(\mathbf{A}))\mathbf{Type}(\mathbf{B}(x))$$

U-*elimination* 1:

$$\frac{\mathbf{A} \in \mathbf{U}}{\mathbf{Type(A)}\ \textit{type}}$$

U-*elimination* 2:

$$\frac{\mathbf{A} = \mathbf{B} \in \mathbf{U}}{\mathbf{Type(A)} = \mathbf{Type(B)}}$$

□

One can draw a straight parallel between Prof. Martin-Löf's presentation of Universes and the presentation given by the Chalmers Group. Such a parallel is easily done by noticing that the above presentation is essentially a combination of both '*à la* Tarski' and '*à la* Russell' given by Prof. Martin-Löf in the following way: the 'decodifier' 'T(**type**)' in Prof. Martin-Löf's presentation *à la* Tarski would correspond to '**Type**(**TYPE**)', and the lower case types ('π', 'σ', '**i**', etc.) correspond to the overlined types. The 'finite' types '$\mathbf{n}_0$', '$\mathbf{n}_1$', $\cdots$, and '$\mathbf{N}_0$', '$\mathbf{N}_1$', $\cdots$, correspond to the 'enumeration' types '$\overline{\{i_1, \cdots, i_n\}}$', and '$\{i_1, \cdots, i_n\}$'.

5.3.5. Meaning and Utility

It would seem fair to say that Ludwig Wittgenstein's idea concerning *meaning* as being inseparable from concepts like *use*, *utility*, *consequences*, etc., works as a much more intuitively appealing explanation of meaning than most of the meaning explanations offered by the usual verificationist theories of meaning.

Rather than a rejection of the 'anti-realist' criticisms put forward by those verificationist theories of meaning, the point made here seems to be offering an improvement on the 'anti-realist' criticisms by attempting to enrich the notion of verification in the light of Wittgenstein's teachings. What is essentially suggested here is that Wittgenstein appeared to have had a very different view on the matter, which strikes one as having a significantly stronger convincing force than the profoundly distinct teachings offered by most of the verificationistic theories of meaning including *Intuitionistic Type Theory*. Moreover, it does not appear to sound more or less

intuitively appealing in the analysis of ordinary languages than in the observance of formalised languages. It is just a very simple and straightforward conceptual link which can significantly help forming a better understanding of such a key notion for any discussion on the nature of any rational discourse: the notion of *meaning*.

The idea of relating the notion of *meaning* with that of *utility* finds its roots in Wittgenstein's very early writings. It helps one to look at the so-called 'logical constants' as linguistic devices like any other, instead of as a 'constant' in the usual sense of the word.

(...) Logic is still in the melting pot but one thing gets more and more obvious to me: The propositions of Logic contain *only apparent* variables and whatever may turn out to be the proper explanation of apparent variables, its consequences *must* be that there are *no logical* constants.

Logic must turn out to be a *totally* different kind than any other science.

(*Letters to RK&M*, letter *R.2*, dated 22.6.12, p. 10.)

Logical connectives do exist insofar as they serve a certain purpose, insofar as are linguistic devices which have a meaning/use/utility, but for that very reason they can even 'disappear':

With the logical constants one need never ask whether they exist, for they can even *vanish*!

(*Notebooks*, dated 25.10.14, p. 19^e.)

The view on 'Words are Tools', leading to 'logical connectives are devices which can be characterised by their use/utility', is in fact made clear both in the *Investigations*:

Think of the tools in a tool-box: there is a hammer, pliers, a saw, a screw-driver, a rule, a glue-pot, glue, nails and screws.—The functions of words are as diverse as the functions of these objects. (And in both cases there are similarities.)

(*Investigations*, Part I, §11, p. 6^e.)

and in the *Blue Book*:

Think of words as instruments characterized by their use, and then think of the use of a hammer, the use of a chisel, the use of a square, of a glue pot, and of the glue. (Also, all that we

say here can be understood only if one understands that a great variety of games is played with the sentences of our language: Giving and obeying orders; asking questions and answering them; describing an event; telling a ficticious story; telling a joke; describing an immediate experience; making conjectures about events in the physical world; making scientific hypotheses and theories; greeting someone, etc., etc.)

(*Blue Book*, p. 68.)

It would seem fair to say that by looking at logical connectives as linguistic devices Wittgenstein has attempted to break with a long tradition whose primary concern was truth rather than meaning, the chief concept being that of correspondence rather than application. It is perhaps relevant to recall here a passage from the lectures on philosophical psychology which he gave in Cambridge in 1946–47, notes of which have been taken down and recently published (Geach (ed.) 1988). It says:

Linguistic devices are like the handles of a machine. They are all made to fit the hand, but what they do and what you have to do with them are different. One has to be pulled hard, one can be adjusted continuously, one will take only two positions.

The traditional logic is concerned only with the handles, not with what they do. E.g. the copula doesn't matter. In German we say "es blaut", "it's blue"; there might be a language where all colour words were verbs.

(Geach (ed.) 1988, p. 25.)

In order to relate my own view of types and type presentations to other views in the literature I have pointed out in chapter 3 that my own notion of *types* is closer to the notion of *systems* as developed by Prof. P. Aczel in his paper on *Frege structures* (Aczel 1980) than Prof. P. Martin-Löf's own notion of *types*. And this is because Prof. Aczel's *systems* — namely *lambda system*, *pairing system*, *Peano system* — are presented/defined not just by what they contain, but by their *utility*, by what can be done with them.

5.3.6. Neither (Strict) Atomism nor (Unmanageable) Holism

From the standpoint that logical connectives can be explained through the use of the concept of types as identified with propositions, and given that the theories of abstract data types shows us how to uniquely characterise types by demonstrating their functionality, their utility, rather than simply by what they are formed of, it would seem reasonable to say that while such a conception of the logical connectives would not be fairly classified as atomistic in the strict sense, it could not be said to be incapable of systematisation. If types are not given meaning in terms of some primitive elements which would be meaningful in themselves, it would appear untenable to say that the account of propositions based on types (as characterised by their use/utility) is strictly atomistic or, if one prefers, molecular. As I understand Wittgenstein's semantic theory based on the dictum 'meaning is determined by use', (no exegetical claim being made here), I feel that a theory of meaning based on types and their use/utility such as my own *'Meaning-As-USE' Type Theory*, named after Wittgenstein's semantical paradigm, would seem to deserve being called as it is. Obviously, the paradigm is not as unambiguous as it would first seem. And, indeed, much has already been said on the different interpretations one can give to Wittgenstein's 'definition' of 'the meaning of a word is its use in the language'. *Use* being such a vague concept as it is in ordinary language, enough room is left for distinct and perhaps uncongenial readings of Wittgenstein's semantical paradigm. There are readings discussed in the literature which lend too vague a character to Wittgenstein's use-based semantical paradigm, some attempting to discuss use in the sense of 'usage', some in the sense of 'practice', etc.. The general picture which remains is that of vagueness, usually in the least positive sense. Therefore, as a general rule, one's first reaction to the adoption of a use-based semantics in the analysis of formalised languages is that of scepticism with respect to the feasibility of the enterprise. One is induced to think that the enterprise may turn out to require unfeasible tasks to be realised, because when describing the use of a word or sentence one has to refer to the whole context within which it is embedded, as well as to its intentionality aspect. And from this, one is almost inevitably led to conclude that, according to Wittgenstein, 'language is really unanalysable' (*not* Wittgenstein's own

words!). One might then justify the arrival at such a (not quite positive) conclusion by saying that if one has to deal with contexts and intentionality, without having the support of some 'universe' of values (cf. interpretations-based semantical theories), then one makes the semantical standpoint sound inadequate for linguistic analysis through formalised description and reasoning mechanisms, given the complexities one would be inevitably faced with.

In his attempt to answer the question of 'what is a theory of meaning?' Prof. Dummett clearly favours the principle that the meaning of an assertion is determined by the conditions under which one can correctly assert it because 'in a theory of meaning in which truth plays the central role', as his verificationist theory of meaning in fact does, 'the content of any assertion is fully determined by the condition that the sentence uttered be true' (Dummett 1976, p. 112). And this would seem to find its intuitive justification in the view that one knows the meaning of any sentence when one simply knows what has to be the case for it to be true. With the possession of such knowledge, one knows 'the content of any utterance of that sentence with assertoric force, and equally, with imperatival, interrogative, optative force, etc.' (Id. Ibid.).

> The possibility that a language may stand in need of adjustment, that, in particular, the principles of inference conventionally recognized may require reassessment, is implicit in the idea that a language ought to be capable of systematization by a theory of meaning which determines the use given to each sentence from the internal structure of that sentence, that is, by an atomistic or molecular theory of meaning; for there can be no guarantee that a complex of linguistic practices which has grown up by piecemeal historical evolution in response to needs felt in practical communication will conform to any systematic theory.
>
> (Dummett 1976, p. 104.)

Later in the same discussion the approach taken by Wittgenstein is recalled, and the argument in favour of what seems to be a rather strict atomistic (or molecular) theory of meaning based on canonical proofs (canonical values) is centered around a criticism to what appears *prima facie* as what I am calling here 'an unmanageable holism':

(...) The idea that Wittgenstein had, that acceptance of any principle of inference contributes to determining the meanings of the words involved, and that therefore, since the speakers of a language may confer on their words whatever meanings they choose, forms of inference generally accepted are unassailable by philosophical criticism, has its home only within a holistic view of language. If a language ought to be capable of systematization by means of an atomistic or molecular theory of meaning, we are not free to choose any logic that we like, but only one for which it is possible to provide a semantics which also accords with the other uses to which our sentences are put; in accepting or rejecting any particular form of inference, we are responsible to the meanings of the logical constants, thought of as given in some uniform manner (e.g. by two- or many-valued truth tables).

(Dummett 1976, p. 105.)

Now, what I find remarkable about, for instance, Prof. Hintikka's *Game-Theoretical Semantics* is that a (semi-)formal account of Wittgenstein's 'the *use* of the logical sign in the language-game' for the case of the language of mathematics is given, and yet no prohibitively insurmountable difficulty is dealt with by bringing in information about the 'context' and the corresponding intentional ingredients. It actually makes the explanation of use of logical constants 'context-free', so to speak, by laying down the 'rules of the game', by stating in each case what can be deduced from the propositions with those logical signs as the major connective, no matter what context they might be embedded in. The intentional ingredients are brought to logical form, so to speak. And this is done by means of explanations of what consequences one is allowed to draw from the corresponding propositions. It is then left implicit, (as in the context of a proper game where moves other than those stated are simply invalid, nonsensical), that any consequence beyond the one(s) indicated cannot be drawn from the propositions, whatever context they may be uttered in (wherever the game is played!). It does not seem to be the case that the adoption of the Wittgensteinian approach turned the corresponding theory of meaning incapable of systematising language analysis. What it does seem very reasonable to say is that a theory of meaning constructed on the basis of *assertability conditions* as the central aspect of the meaning of a sentence can hardly be said to incorporate any intentional ingredient. By knowing the conditions for asserting a particular sentence one can

hardly know the commitments made by asserting it, and it is this very aspect of 'commitments' which plays a central semantic rôle in what Prof. Dummett calls a 'falsificationist' theory of meaning:

> (...) a falsificationist theory [of meaning] does not relate the meaning of a sentence directly to the grounds of an assertion made by means of it at all. Instead, it links the content of an assertion with the commitment that a speaker undertakes in making that assertion; an assertion is a kind of gamble that the speaker will not be proved wrong. Such a theory therefore has obvious affinities, not only with Popper's account of science, but also with the language-game type of semantics developed by Hintikka and others.
>
> (Dummett 1976, p. 126.)

And 'reflection should make us admit' that a falsificationist theory of meaning would be better than a verificationist *assertability conditions*-based theory of meaning, according to Prof. Dummett (Ibid., p. 137). Now, as I have pointed out earlier, it is reasonable to suppose that the commitments one makes by asserting a sentence are inseparable from the consequences he/she allows to be taken from it, and a (falsificationist) theory of meaning based on the connections between the meaning of a proposition and the consequences one can draw from it such as *Game-Theoretical Semantics* (to which Prof. Hintikka refers as verificationist in a recent paper: Hintikka 1987, p. 499), should be ranked better than a verificationist theory of meaning based on *assertability conditions*.

In my type-theoretic framework, the 'atomic' or 'molecular' units, namely propositions, are explained by what can be done with them (*introduction* and *reduction* rules), rather than by an explanation of how its (canonical) proofs (or canonical elements as in the case of types) are to be recognised.

5.4. Type Theory and The Meaning of Logical Constants

What appears to be a simple and straightforward problem has still not met a universally agreed solution on a satisfactory basis. The question of finding a suitable formal account of meaning for the (primitive) logical signs has troubled many philosophers

and logicians since the early days of formal logic, whenever it is even recognised as a problem.

In this section I would like to point out how two *operational* (as opposed to *denotational*, or *truth*-based) accounts for such a problem can still be shown to be technically equivalent, despite having emerged from two different readings of a single philosophical account, and being essentially distinct with respect to the nature of the mathematical activity (public vs. private): on the one hand, the 'semantics of use' of *'Meaning-As-USE' Type Theory*, my own reformulation of Prof. P. Martin-Löf's *Intuitionistic Type Theory* semantics where the *normalisation* rules are taken to be semantical rules; and, on the other hand, Prof. J. Hintikka's *Game-Theoretical Semantics*, where the meaning of logical signs is given via '*semantical games*'. The philosophical account from which both emerge is precisely Wittgenstein's later account of the nature of propositions where the notion of 'language-games' is a key semantical concept. In the course of the argument an attempt is made to motivate the use of *normalisation* as a semantical notion by looking at the evolution of a rather early and persistent intuition of Wittgenstein concerning the rôle of the *consequences* that one can draw from a proposition in determining its meaning. By making use of quotations from the early as well as from the later Wittgenstein's writings I have shown in the previous chapter that this feature appears to have been a constant in Wittgenstein's own intuition on the semantics of the mathematical language, despite the sharp distinction of the 'early-later' Wittgenstein.

Finally, observing that the *normalisation* rules seem to be able to capture the very aspect of the meaning of a proposition concerning its consequences, I say that such a meta-mathematical device could be a semantically useful notion which would lead to a reasonable solution of the problem of formulating the meaning of logical constants on a more satisfactory basis.

5.4.1. Wittgenstein's Early Awareness

L. Wittgenstein, since his early days, was aware of the fundamental nature of the issue, and in a letter to Russell he expresses his preoccupations on the question of

the meaning of logical signs:

What troubles me most at present is not the apparent-variable-business, but rather the meaning of "$\vee$", ".", "$\supset$", etc. This latter problem is — I think — still more fundamental and, if possible, still less recognized as a problem.

(*Letters to RK&M*, Letter *R.4*, dated Summer 1912, p. 13.)

In his *Philosophical Investigations* Wittgenstein provides a more uniform account of what a proposition is, and, it seems to me, more faithful to his original intuition on the nature of semantics. A proposition is not defined by its *truth conditions*, which characterised the realist semantics of the *Tractatus*. Neither is a proposition defined by its *assertability conditions* alone, as it is customarily taken to be in most of the alternatives to realist semantics, and which characterises, for example, Gentzen's proof-theoretic semantics. Rather, a proposition is in fact defined by two distinct and fundamental aspects of its *use*: the *grammar* aspect and the *consequences* aspect.

In the *Investigations*, which is the main representative of the later Wittgenstein and is considered to be his *chef d'œuvre* due to its thorough and indeed original studies on many topics of central concern to philosophers, such as meaning, understanding, propositions, logic, states of consciousness, etc., he brings in a new notion which he calls '*Sprachspiel*' (often translated as 'language-games') in order to define his (later) account of what a proposition should be:

(...) Und was ein Satz ist, ist in *einem* Sinne bestimmt durch die Regeln des Satzbaus (der deutschen Sprache z.B.), in einem andern Sinne durch den Gebrauch des Zeichens im Sprachspiel.

which has been translated as

(...) And what a proposition is is in one sense determined by the rules of sentence formation (in English, for example) and in another sense by the use of the sign in the language-game.

(*Investigations*, Part I, §136, p. 53^e.)

(*En passant*, notice that the translation does not show the emphasis on the first '*one*' ('*einem*') with regards to the aspects (or 'senses', Ger. *Sinn*, to use Wittgenstein's own words) of a proposition. The *emphasis* seems to play the important rôle

of stressing the *conjunctive* character of Wittgenstein's assertion on the nature of propositions: a proposition has a first aspect *and* a second aspect. The absence of the emphasis in the translation appears to leave room for unwanted readings of his later account of semantics, given that the *assertability conditions*, represented by the first aspect ('the rules of sentence formation'), were put by Wittgenstein as *one of both* and not *the one aspect which could be chosen from either aspects* of a proposition. Apparently overlooking the conjunctive nature of the assertion, most verificationist and 'anti-realist' alternatives to classical semantics seem to insist on a certain 'priority' of this first aspect over the second one, which makes them lose the force of a truly convincing alternative.)

However, Wittgenstein himself does not seem to provide clear-cut, fully worked out examples of logical counterparts to the informal notion of 'use of the sign in the language-game' which would make more precise what such a notion would correspond to. A number of interpretations of such a notion of games of language have emerged, among them are certainly Prof. M. Dummett's commitment-based theory of meaning and Prof. Prawitz's attempt to develop such a theory of meaning for the language of mathematics in proof-theoretic terms. Nevertheless, what seems to have been the only reading of the *Investigations* which has originated a significant and fairly well-established body of ideas regarding the application of the notion of 'language-games' as such, to the foundations of logic and mathematics appears to have been Prof. J. Hintikka and his followers' *Game-Theoretical Semantics*[29]. Although uncovering the semantical rôle of a demonstration of the consequences one can draw from a proposition, this approach brings in a particular conception of the mathematical activity where there must be a 'community' (at least two 'players', even if one is just *Nature*) acting on the process. Despite its great merits in clarifying a number of issues regarding the application of the later Wittgenstein's 'meaning-

[29]None of the other '*dialogue*-based' approaches to semantics, such as e.g., Prof. P. Lorenzen's '*Dialogue-Games*' (Lorenzen 1969), C. Hamblin's '*Imperatives*' (Hamblin 1987), and a few other relevant contributions to the approach, seem to have mentioned explicitly Wittgenstein's 'language-games' as the direct source of inspiration.

is-use' to the semantics of the language of mathematics, this view does not seem to match with the truly individual nature of the mathematician's activity. We shall then see how an alternative reading can still be made of 'language-games' without having to resort to such 'dialogical' views of mathematics, which is what seems to characterise most variations of Prof. Hintikka's reading and even early and similar approaches by Prof. P. Lorenzen: dialogue-games, game-theoretical semantics, logics of dialogue, etc. (Lorenzen 1969).

5.4.2. Proof Theory, Meaning and Language-Games

The notion of 'language-games' has often been taken from a 'dialogical' point of view to serve as the basis for game-theoretic semantics and logics of dialogue. Recalling Frege's account of the mathematician's activity ('*Part of the mathematicians' activity besides drawing inferences is to give definitions*'[30]), I want to have a more 'individual-to-community' account of the language-games for the particular case of the mathematical language. I believe that the central question is to identify elements of the language for the mathematician to be able to express 'non-solipsistically' his own thoughts in a way understandable to others. Therefore, making a less community-oriented reading of what a language-game should be, I would like to say that the metamathematical device of *normalisation* can in most cases without any loss be used to capture the very essence of what Wittgenstein seemed to have meant by 'language-games': the aspect of the meaning of a proposition which concerns the *consequences* that can be drawn from it. The emphasis on the *consequences* aspect, besides the *grammar* aspect, is in fact what makes the sharp distinction between most 'anti-realist' alternatives to classical semantics for the language of mathematics and Wittgenstein's celebrated *meaning-is-use* semantical paradigm. It is indeed a great departure from the generally known *assertability conditions*-based alternative to the 'realist' '*truth conditions*'-based account of meaning for the mathematical language.

[30] p. 203 of Frege 1914.

Prof. M. Dummett, in his theory of meaning and truth, attempts to provide a convincing argument to support his 'anti-realist' criticisms by offering an account of meaning based on the 'commitments' one makes when asserting a proposition. It would seem to have been a fruitful approach to an anti-realist theory of meaning had he given a logical counterpart to this informally presented aspect which apparently received such a denomination of 'commitment' from Prof. Dummett himself. Unfortunately, by putting forward his own reading of Wittgenstein as another philosophical support for the '*assertability conditions*' alternative to realist semantics, he seems to leave a gap which concerns precisely the place such 'commitments' would take in an account of meaning based on the conditions for assertion alone. The emphasis on the 'commitments' aspect is made clear when he says that his 'falsificationist' (as opposed to 'verificationist') theory of meaning *links the content of an assertion with the commitment that a speaker undertakes in making that assertion*[31], 'content' being used in the sense of 'meaning'.

For some reason, however, Prof. Dummett's reading of Wittgenstein seems to stay essentially in line with the *assertability conditions*-based anti-realist alternative by saying that:

> [The Meanings of the Logical Constants] The meaning of each constant is to be given by specifying, for any sentence in which that constant is the main operator, what is to count as a proof of that sentence, it being assumed that we already know what is to count as a proof of any of the constituents.
>
> (Dummett 1977, p. 12.)

His 'joint' programme with Prof. D. Prawitz[32] on a semantic theory as well

[31] p. 126 of Dummett 1976.

[32] 'I am in very great agreement with what Prawitz says. This is not surprising, since he is expounding ideas which, for some years, he and I have been developing, each influenced by the other in a way that makes it now impossible, even if there were any point of doing so, to disentangle who contributed what.', opening lines of Dummett 1980, which is a comment/response to Prawitz 1980.

as a theory of meaning[33] for the language of (intuitionistic) mathematics, ends up neglecting the very aspect of 'commitment' in favour of a certain priority for the 'conditions for assertion' with respect to the meaning of a proposition.

Another account of Wittgenstein's views on the nature of propositions which again makes reference to Prof. Dummett's 'commitment' aspect is that of Prof. S. Kripke. Although leaving somewhat undefined as to exactly what kind of 'anti-realist' alternative he is providing, and not concerned with the logical counterparts to Wittgenstein's informal explanations, Prof. Kripke seems to make clear that there are indeed *two* fundamental aspects in Wittgenstein's later account of propositions, offering a very intuitively appealing explanation by introducing a rather distinctive mode of phrasing Wittgenstein's account. Prof. Kripke's reading of the account of propositions expressed in the *Investigations* makes use of the notion of *utility*, allowing the very late connections *meaning-use-utility* to be easily accommodated. For this reason I believe that it constitutes a particularly interesting and highly relevant reading of the later Wittgenstein's account on the nature of a '*Satz*':

> (...) [From the *Tractatus* to the *Investigations*] Wittgenstein replaces the question, "What must be the case for this sentence to be true?" by two others: first, "Under what conditions may this form of words be appropriately asserted (or denied)?"; second, given an answer to the first question, "What is the rôle, and the utility, in our lives of our practice of asserting (or denying) the form of words under these conditions?"
>
> (Kripke 1982, pp. 72–3.)

(The emphasis is mine.)

It seems to be a useful rephrasing of Wittgenstein's own remark (*Investigations*, Part I, §136) although no account as to how this could possibly have a logical counterpart seems to have been suggested. Nevertheless, such a notion of *utility* seems to be a particularly useful notion indeed, given that, e.g., one can easily make a parallel between, on the one hand, the fact that by saying that ϕa follows from

[33] see Dummett 1982 for more details on such a distinction between a 'semantic theory' and a 'theory of meaning', apparently made by Prof. Dummett himself.

$(x).\phi(x)$, the (generality) meaning of $(x).\phi(x)$ is made explicit, and, on the other hand, the rôle of the β-rule of the λ-*Calculus* which shows the (generality) meaning and, in the end, the *utility* of 'λ', by reflecting the second aspect of a 'proposition' such as $\lambda x.f(x)$. As I have pointed out previously, my own interpretation of (the later) Wittgenstein's semantical teachings seem to be somewhat germane to (my own understanding of) Prof. Kripke's views: I say that the *normalisation* rules express in a somewhat 'cryptic' though useful way the aspect concerning the '*utility*' of the particular logical sign. We shall see how this is done in the following sections.

Intuitionistic Type Theory would seem to be a reasonable framework where one could find a logical counterpart to the notion of *utility* had its semantics been given without the exclusive emphasis on (numerical, canonical) values, and had *normalisation* been used to play its naturally semantical rôle. In such a framework, as it has been presented in different versions by Prof. Martin-Löf due to the identification of propositions with types, the rules of *elimination*, which would *prima facie* represent the *consequences* aspect of a particular proposition/type, are framed using the notion of 'noncanonical' proofs/elements which represent eliminatory operators over the canonical proofs/elements. For instance:

Σ-*elimination*

$$\frac{c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{FST}(c) \in \mathbf{A}} \qquad \frac{c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{SND}(c) \in \mathbf{B}(\mathtt{FST}(c))}$$

Π-*elimination*

$$\frac{c \in (\Pi x \in \mathbf{A})\mathbf{B}(x) \quad a \in \mathbf{A}}{\mathtt{APPLY}(c, a) \in \mathbf{B}(a)}$$

$+$-*elimination*

$$\frac{\begin{array}{ccc} & [x \in \mathbf{A}] & [y \in \mathbf{B}] \\ c \in \mathbf{A} + \mathbf{B} & d \in \mathbf{C}(\mathtt{inl}(x)) & e \in \mathbf{C}(\mathtt{inr}(y)) \end{array}}{\mathtt{WHEN}(c, d, e) \in \mathbf{C}(c)}$$

Nevertheless, those rules only show which are the non-canonical operators without actually indicating how they work. The resort to verbal explanation is necessary, from which the *equality* (or *computation*) rules are said to become evident. No account is given, however, as to how primitive those *elimination* rules and noncanonical proofs/elements are; but the meaning of a proposition/type is claimed to be given by the corresponding *introduction* rules.

According to my own reading of Wittgenstein's account of propositions, I say that one must settle the *consequences* aspect of the explanation given above which, as given, includes only the 'grammar' aspect. In this way, the 'noncanonical' operators, which perform the eliminations on the 'canonical' ones, have to be made explicit as part of the definition of the particular proposition/type. The rules for this purpose are provided by Prof. Martin-Löf but in a rather secondary place with respect to the definition of each proposition. In other words, they are not considered as primitive rules with respect to the meaning of the proposition/type being defined, which amounts to the same thing as saying that they are not taken as 'semantical rules'. They are called '*equality*' rules, but in my own reformulated version, attempting to be faithful to their original purpose, I call them *reduction* rules. The *reduction* rules of my *'Meaning-As-USE' Type Theory* are framed as, e.g.:

('$\wedge$' being the pairing (type) operator meant to 'code up' the logical connective of conjunction):

$\wedge$-reduction

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{B}}{\mathtt{FST}(\mathtt{pair}(a,b)) = a \in \mathbf{A}} \qquad \frac{a \in \mathbf{A} \quad b \in \mathbf{B}}{\mathtt{SND}(\mathtt{pair}(a,b)) = b \in \mathbf{B}}$$

($\forall$ being the (type) forming operator meant to provide a schema for the universal quantifier):

$\forall$-reduction

$$\frac{\begin{array}{cc} & [x \in \mathbf{A}] \\ a \in \mathbf{A} & b(x) \in \mathbf{B}(x) \end{array}}{\mathtt{APPLY}((\lambda x)b(x), a) = b(a/x) \in \mathbf{B}(a)}$$

(and ∨, meant to do the same for the case of disjunction):

∨-*reduction*

$$\frac{\begin{array}{ccc} & [x \in \mathbf{A}] & [y \in \mathbf{B}] \\ a \in \mathbf{A} & d \in \mathbf{C}(\mathtt{inl}(x)) & e \in \mathbf{C}(\mathtt{inr}(y)) \end{array}}{\mathtt{WHEN}(\mathtt{inl}(a), d, e) = d(a) \in \mathbf{C}(\mathtt{inl}(a))} \qquad \frac{\begin{array}{ccc} & [x \in \mathbf{A}] & [y \in \mathbf{B}] \\ b \in \mathbf{B} & d \in \mathbf{C}(\mathtt{inl}(x)) & e \in \mathbf{C}(\mathtt{inr}(y)) \end{array}}{\mathtt{WHEN}(\mathtt{inr}(b), d, e) = e(b) \in \mathbf{C}(\mathtt{inr}(b))}$$

5.4.3. Game-Theoretical Semantics and Semantical Games

As an attempt to provide a logical counterpart to the later Wittgenstein's 'the use of the sign in the language-game', Prof. J. Hintikka and associates have developed what is known as *Game-Theoretical Semantics*, which would be roughly described as a body of formal as well as informal framework to explain the activity of the mathematician as a game against an abstract entity named 'Nature', where the latter would be trying to falsify the statements of the former. In this framework, the notion of truth would not be identified with that of Tarski, but would be redefined by reference to formal first-order quantificational languages, together with the notions of players, moves and strategic meaning. A language L is assumed to be an interpreted language, and its being interpreted is explained by saying that one is given some domain of individuals D on which all the nonlogical constants of L plus the names of the members of D would have a definite truth-value. The departure from Tarskian truth-based theory of meaning is made by explaining the meaning of each logical constant by means of what is called *semantical games*. The formal devices introduced as counterparts to informal notions were construed in such a way that they would allow the meaning of logical connectives to be explained in a purely rule-based manner. In the description of the *Game-Theoretical Semantics* ('*GTS*')

semantical games are defined by Prof. Hintikka (p. 3 of Hintikka 1983), where there is supposed to be two 'players' called *Myself* and *Nature*, as:

($G.\ A$) If A is atomic, then I have won $G(A)$ and *Nature* has lost if A is true. If A is false, *Nature* has won and I have lost.

($G.\ \&$) $G(S_1\ \&\ S_2)$ begins by *Nature*'s choice of S_1 or S_2. The rest of the game is $G(S_1)$ or $G(S_2)$, respectively.

($G.\ \vee$) $G(S_1 \vee S_2)$ begins by *Myself*'s choice of S_1 or S_2. The rest of the game is $G(S_1)$ or $G(S_2)$, respectively.

($G.\ U$) $G((\mathrm{x})S[\mathrm{x}])$ begins by *Nature*'s choice of a member of D. Let the name of the member chosen be 'a'. The rest of the game is then $G(S[a])$.

($G.\ E$) $G((E\mathrm{x})S[\mathrm{x}])$ is defined likewise except that 'a' is chosen by *Myself*.

The truth of a sentence S of L would then be defined as the existence of a winning strategy in $G(S)$ for *Myself*. Similarly, the falsity of S would be given by the fact that the other 'player', *Nature*, has a winning strategy. In every move of the game the major logical connective of the current 'position' in the game is being eliminated. The *semantical games* are readily seen to be explanations of the consequences that the starting player *Myself* would be allowing the other player *Nature* to ask for.

5.4.4. Normalisation and Semantical Games

Looking at *GTS*'s semantical rules as rules for showing how one can 'eliminate' each particular logical connective, and continue with the game until the 'winner' is known,

it is not difficult to see the identical rôle that the *reduction* rules of *MAUSE* play by showing how noncanonical (eliminatory) operators act on canonical (introductory) elements. Furthermore, given the fact that in *GTS I* (*Myself*) am supposed to play with *Nature* by laying down my own assertions, and *Nature* as the opponent is always asking *Myself* to comply with the commitments *I* (*Myself*) make, it should be easy to see the parallel between 'canonical element vs. *Myself*'s move', and 'noncanonical element vs. *Nature*'s move'. For that comparison it suffices to look at the conclusion of *MAUSE*'s *reduction* rules (the *judgement* at the bottom of each of those rules) and confront it with the corresponding '*semantical games*'.

As a first example, let us look at how the *consequences* aspect of the logical constant of conjunction is coded via *MAUSE*'s *reduction* rules as in:

$\wedge$-*reduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{B}}{\mathtt{FST}(\mathtt{pair}(a,b)) = a \in \mathbf{A}} \qquad \frac{a \in \mathbf{A} \quad b \in \mathbf{B}}{\mathtt{SND}(\mathtt{pair}(a,b)) = b \in \mathbf{B}}$$

In *GTS* the rules for conjunction are given as:

(*G.* &) $G(S_1 \ \& \ S_2)$ begins by *Nature*'s choice of S_1 or S_2.
The rest of the game is $G(S_1)$ or $G(S_2)$, respectively.

Following the above brief account of the connections between the duality canonical vs. noncanonical operations of *MAUSE* and the duality *Myself* vs. *Nature* of *GTS*, we can easily relate the two semantical rules: *Nature* chooses first in (*G.* &) of *GTS*, and the 'noncanonical' operation (either 'FST' or 'SND') makes the choice first in $\wedge$-*reduction* of *MAUSE*.

For the case of the universal quantifier the connections are also rather clear:

In my own *'Meaning-As-USE' Type Theory* the rule of *normalisation* for the universal quantification (type) operator '$\forall$', which is meant to be a general schema to define/present the logical connective of 'universal quantifier', is given as:

$\forall$-*reduction*

$$\frac{\begin{array}{cc} & [x \in \mathbf{A}] \\ a \in \mathbf{A} & b(x) \in \mathbf{B}(x) \end{array}}{\mathtt{APPLY}((\lambda x)b(x), a) = b(a/x) \in \mathbf{B}(a)}$$

and the rules of the game for the universal quantifier ('U') in *GTS* are given as:

(*G. U*) $G((\mathrm{x})S[\mathrm{x}])$ begins by *Nature*'s choice of a member of D.
Let the name of the member chosen be 'a'. The rest of the game is then $G(S[a])$.

Observe how both notions of *generality* (*Nature* choosing an arbitrary 'a' from the domain D of individuals) and *substitution* (where 'x' is replaced by the individual 'a') are clearly present in both semantical rules (*G. U*) of *GTS* and $\forall$-*reduction* of *MAUSE*. Furthermore, similarly to the corresponding *GTS semantical game*, the 'choice' of 'a' is part of the 'noncanonical' element 'APPLY' move.

For the case of disjunction the connections seem to work as nicely as in the other cases, except that the interchange of positions (*Nature* playing second instead of first) may make it slightly less straightforward to see the similarities. Nonetheless, this is certainly the most interesting evidence for the straight connections between the two formal counterparts to Wittgenstein's semantic theory, namely *Game-Theoretical Semantics* and *'Meaning-As-USE' Type Theory*. We shall see that the 'interchange' of positions is indeed reflected in the interplay between 'canonical' vs. 'non-canonical' operators.

Here are the *normalisation* rules for the connective '∨' of *MAUSE*, which is meant to be a schema for the presentation of logical disjunction:

∨-*reduction*

$$\frac{a \in \mathbf{A} \quad \begin{matrix}[x \in \mathbf{A}] \\ d \in \mathbf{C}(\mathtt{inl}(x))\end{matrix} \quad \begin{matrix}[y \in \mathbf{B}] \\ e \in \mathbf{C}(\mathtt{inr}(y))\end{matrix}}{\mathtt{WHEN}(\mathtt{inl}(a), d, e) = d(a) \in \mathbf{C}(\mathtt{inl}(a))} \qquad \frac{b \in \mathbf{B} \quad \begin{matrix}[x \in \mathbf{A}] \\ d \in \mathbf{C}(\mathtt{inl}(x))\end{matrix} \quad \begin{matrix}[y \in \mathbf{B}] \\ e \in \mathbf{C}(\mathtt{inr}(y))\end{matrix}}{\mathtt{WHEN}(\mathtt{inr}(b), d, e) = e(b) \in \mathbf{C}(\mathtt{inr}(b))}$$

The game-theoretical semantic rules for disjunction ('∨') are given as:

(*G.* ∨) $G(S_1 \vee S_2)$ begins by *Myself*'s choice of S_1 or S_2.
The rest of the game is $G(S_1)$ or $G(S_2)$, respectively.

Again, the connections are clear: on the one hand *I* (*Myself*) make the choice first in (*G.* ∨) of *GTS*, and on the other hand the 'canonical' operation of ∨-*reduction* of *MAUSE* starts 'marking' the choice (either '`inl`' – 'into the left disjunct', or '`inr`' – 'into the right disjunct').

By drawing a parallel between *normalisation* rules of *MAUSE* and semantical games of *GTS*, we seem to be able to provide means for a better understanding of how such a notion of 'operational semantics' via the explanation of the *consequences* embedded in Wittgenstein's 'the use of the sign in the language-game' is made possible via a particular 'meta-mathematical' device such as *normalisation*. The fact that both cases show that the explanation of the *consequences* of an assertion have an important part to play regarding its meaning, comes as some sort of confirmation for the soundness of Wittgenstein's (quite early!) intuition regarding his (later) interpretation-free account of meaning. If we look at some 'modern' mathematical languages such as, e.g., A. Church's λ-Calculus (Church 1941), Prof. P. Aczel's language of Frege Structures (Aczel 1980), the language of Category Theory (e.g., Lambek & Scott 1986), etc., where the device of *normalisation* is clearly present,

even if not made explicit as a semantical tool, we can see that the wide use of such an 'operational semantics' device seems to be a precursor for some kind of 'linguisticisation' of well-established mathematical concepts, if I can put it this way.

5.4.5. A Reflection on Public vs. Private Mathematics

At this point one can see how such an apparently minor problem of formulating precisely the meaning of (primitive) logical signs can raise much deeper enquiries into the nature of the mathematical activity. The question concerning the rôle of language in mathematics is inevitably brought into focus. Without going too far into an issue which is fundamental, complex, and still very much open to discussion, I would express my own views on the matter by saying that the meta-mathematical device of *normalisation* seems to play the 'public' part of an activity which appears to be of an essentially 'private' nature. That this can be taken as a plausible attempt to find a point of reconciliation between Brouwer's rather solipsistic views and some more recent 'dialogical' views on the nature of 'doing mathematics', is still to be judged by a comparison with other views on the matter.

5.5. Incoherences of Intuitionistic Type Theory Due to Canonical Value-Based Semantics

A significant attempt to contribute to a more satisfactory account of meaning without any need to resort to the more usual world of interpretations, models, etc., would have been offered by Prof. Martin-Löf's had his semantic theory taken into account the semantical rôle of the meta-mathematical device of *normalisation*. A semantical viewpoint justified by an endeavour to formalise the notion of *utility*, through the internalisation of the *normalisation* rules would seem to make the semantic theory of *Intuitionistic Type Theory* a much more convincing alternative to classical denotations-based semantics. The rules of *equality* (as the *reduction* rules are called in Prof. Martin-Löf's formulation of *Intuitionistic Type Theory*) are given as part of the formal system, but, nonetheless, they are not explained as fundamental rules

as far as the meaning of types/propositions is concerned. On the contrary, after giving the rules for the *assertability conditions* (the *introduction* rules), indicating the *canonical* proofs of the proposition, and a certain formulation of *elimination* rules for each logical constant, a verbal explanation of how the *elimination* rules work is provided. Such verbal explanation is justified as being some kind of 'meta-level' explanation of the 'computational' semantics associated with the body of ideas which support the formal system. From this verbal explanation, the *equality* rules are then said to become evident. As the result of a more careful look at the nature of those rules in relation to a certain interpretation of Wittgenstein's teachings on the nature of meaning and language-games, I am led to say that the order of importance seems to have been inverted here. The rules of *normalisation* are just considered to have a less secondary rôle to play in such an endeavour to formalise the rules of mathematics. They should naturally come first, as they constitute part of the meaning (use) of the particular connective by capturing its aspect of the *consequences*, *utility*, etc.. The verbal explanation would then come as the result of some individual's understanding.

As a matter of fact, the semantical explanations given by Prof. Martin-Löf, when saying that the meaning of a type is given by the rules demonstrating what are the canonical elements of the type, the so-called *introduction* rules, appear to leave enough room for misconceptions. Essentially the same semantical misconception which led Prof. Prawitz to failure and circular arguments in his attempts to derive natural deduction *elimination* rules from *introduction* rules:

> Gentzen suggested that "it should be possible to display the elimination rules as unique functions of the corresponding introduction rules on the basis of certain requirements". One has indeed a strong feeling that the elimination rules are obtained from the corresponding introduction rules in a uniform way.
>
> (Prawitz 1978, p. 37.)

as I have already pointed out in chapter 3, appears to have been repeated within Prof. Martin-Löf's extended account of natural deduction. An attempt has been recently made to devise 'a general method to obtain all the remaining rules for a type just

from the corresponding *formation* and *introduction* rules of a type defined/presented in *Intuitionistic Type Theory*':

> The main contribution that we make here is to describe a scheme for inferring the elimination rule and computation rules for a newly introduced type constructor. In other words, we show that it suffices to provide the type formation rule and the introduction rule for a new type constructor; together these provide sufficient information from which the remaining details can be deduced. The significance of this result comes from the twin benefits of reducing the burden of understanding and the burden of definition. It reduces the burden of understanding and the burden of definition. It reduces the burden of understanding, since we now need to understand only the formation and introduction rules and the general scheme for inferring the remaining rules. The burden of definition is reduced, since it suffices to state the formation and introduction rules, the others being inferred automatically.
>
> (Backhouse et. al. 1989, p. 29.)

(The emphasis is mine.)

At least from a computer scientist's point of view, the idea of deriving the computation rules of a type from the rules which indicate what canonical elements the type contains does not sound truly convincing.

In order to illustrate how counter-intuitive this looks to a computer scientist, let us take the example of the type of 'queues of elements from an alphabet', which is usually defined algebraically as an 'abstract data type' in a way similar to:

queue = bool + nat +

sorts: *alphabet*

queue

opns: $K1, ..., Kn :\rightarrow$ *alphabet*

error $:\rightarrow$ *alphabet*

new $:\rightarrow$ *queue*

addq : *queue alphabet* $\rightarrow$ *queue*

REMOVE : *queue* $\rightarrow$ *queue*

FRONT : *queue* $\rightarrow$ *alphabet*

IS − EMPTY : *queue* → *bool*

LENGTH : *queue* → *nat*

IF − THEN − ELSE : *bool queue queue* → *queue*

eqns: $a \in alphabet; q, q1, q2 \in$ queue

(e1) REMOVE(new) = new

(e2) REMOVE(addq(q, a)) = IF IS − EMPTY(q) THEN new
ELSE addq(REMOVE(q), a)

(e3) FRONT(new) = error

(e4) FRONT(addq(q, a)) = IF IS − EMPTY(q) THEN a
ELSE FRONT(q)

(e5) IS − EMPTY(new) = true

(e6) IS − EMPTY(addq(q, a)) = false

(e7) LENGTH(new) = 0

(e8) LENGTH(addq(q, a)) = succ(LENGTH(q))

(e9) IF true THEN $q1$ ELSE $q2 = q1$

(e10) IF false THEN $q1$ ELSE $q2 = q2$

(Ehrig & Mahr 1985, p. 55.)

Now, the rules of *introduction*, which show what is needed to form the canonical elements of the type '*queue* of *alphabet*', namely 'new' and 'addq(q, a) [$q \in$ *queue*, $a \in$ *alphabet*]' can be said to be given in the first three lines of the paragraph '*opns*'. According to the semantics based on canonical values, the type would be understood just by those rules. Moreover, the remaining lines of the above definition would not have any part to play in determining the meaning of the type in question, given that the meaning had been established by the rules of canonical elements formation, (mis)leading one to think that the computation rules, namely (e2) and (e4), could be automatically derived from those first lines of '*opns*'. Such a '(numerical) value-based' semantical standpoint allows one to go even further and say that, if all one needs to know about a type is to know its canonical el-

ements' then all the lines below the third line of the paragraph '*opns*' shall be a mere consequence of the meaning-giving rules which show how to form '**new**' and 'addq(q, a) [$q \in queue, a \in alphabet$]', the canonical elements of the type '*queue* of *alphabet*'[34].

[34]Cf. J. Guttag's account of abstract data type definitions which points out the insufficiency of the 'syntactical' information indicating how to build canonical elements of the type, and argues in favour of axiomatic definitions as adequate means of describing the distinguishing characteristic of data types. Axioms, he claims, would not only avoid the introduction of extraneous details in the definition of data types but would also keep the level of abstraction at an appropriate degree. In Guttag 1977, he says:

'(...) An algebraic specification of an abstract type consists of two pairs: a syntactic specification and a set of relations. (...)

(...) Consider type Queue (of Items) with the operations listed in previous section. The syntactic specification is as above:

NEW: → Queue
ADD: Queue × Item → Queue
FRONT: Queue → Item
REMOVE: Queue → Queue
IS-EMPTY?: Queue → Boolean

The distinguishing characteristic of a queue is that it is a first in-first out storage device. A good axiomatic definition of the above operations must therefore assert that and only that characteristic. The relations (or axioms) below comprise just such a definition. (...)

(1) IS-EMPTY? (NEW) = true
(2) IS-EMPTY? (ADD (q,i)) = false
(3) FRONT (NEW) = error
(4) FRONT (ADD (q,i)) = **if** IS-EMPTY? (q)
then i
else FRONT(q)
(5) REMOVE (NEW) = error
(6) REMOVE (ADD (q,i)) = **if** IS-EMPTY? (q)
then NEW
else ADD (REMOVE (q),i)

(...)'

Observe that the distinguishing characteristic of the data type 'queue of items' is not clear from the '*opns*' part of the definition, but is only made explicit by 'equations' (e2) and (e4) of the '*eqns*' part. As Guttag says, the distinguishing characteristic of a type is made explicit in the axiomatisation, not in the 'syntax' part of a type definition. Following these principles, the semantics of use says that it is there in the axiomatics where the semantical information concerning the effect of the destructors 'FRONT' and 'REMOVE' over the constructor 'addq' is demonstrated by means of what one could call *normalisation* rules (cf. λ-Calculus and its β-rule of *normalisation*). It would not make sense to think of 'obtaining' the information given in the axiomatisation just from the 'syntax' part (the *introduction* rules in a type-theoretic framework).

5.5.1. Definition of Arithmetical Functions

After Brouwer's rather solipsistic *mathematics is a languageless activity*, Heyting's semi-formal description of intuitionistic logic, and Dummett-Prawitz investigations on the connections between proof theory and meaning, (to mention a few attempts to make more precise the 'anti-realist' criticisms), Prof. Martin-Löf's philosophy and formal system appear to be a most significant step forward. It is an attempt to put into effective terms the claim that the meaning of logical signs can be given without the recourse to any interpretation.

This would presumably mean that using the language of *Intuitionistic Type Theory* to define mathematical objects would be a straightforward and coherent thing to do. The concepts 'rule' and 'meaning' would be clearly related and completely clarified. Unfortunately, this does not seem to be the case for the presentations of *Intuitionistic Type Theory*, including the one given in Martin-Löf 1984. In fact, there seem to be serious incoherencies on how the relation meaning-rule is handled in *Intuitionistic Type Theory*. One reason for the confusions might be identified as

(Guttag 1977, p. 398.)

(The emphasis is mine.)

being a certain incoherence in the notion of *equality*. Definitional equality seems to be the prevailing notion in the notion meaning theory. Such an equality notion, namely definitional or abbreviatory equality, can indeed be a useful equality relation. It can also be the source of serious misconceptions.

When defining the arithmetical functions of addition and multiplication, Prof. Martin-Löf uncovers an often conceptually mistaken employment of the notion of definitional equality. And indeed, if we look at the way arithmetical functions are defined, we realise that there is some incoherence as far as the relation rule-meaning is concerned. First, with predecessor and addition:

Example. (the predecessor function). We put

$$pd(a) \equiv \mathrm{R}(a, 0, (x, y)x)$$

(...)

Example. (addition). We define

$$a + b \equiv \mathrm{R}(b, a, (x, y)y')$$

The meaning of '$a + b$' is to perform 'b' times the successor operation on 'a'. Then one easily derives the rules:

$$\frac{a \in \mathbf{N} \qquad b \in \mathbf{N}}{a + b \in \mathbf{N}}$$

$$\frac{a \in \mathbf{N}}{a + 0 = a \in \mathbf{N}} \qquad \frac{a \in \mathbf{N} \qquad b \in \mathbf{N}}{a + b' = (a + b)' \in \mathbf{N}}$$

from which we can also derive the corresponding axioms of first order arithmetic, like in the preceding example.

(...)

Note that (more incisively in the example of addition) the rule of the 'grammar' (under what conditions an +-expression can be formed) — the first rule —, and the rule of 'use'/consequences (what is the rôle and the utility of '+') — the two remaining rules of equality —, are said to be derived from the definitional equality. Now, in the light of my own interpretation of Frege's analytic definition (chapter

2) and my attempts at a mathematical/logical account of 'implementation' (more details in the next two chapters), it would make more sense to say that the two groups of rules, namely grammar plus use/consequences, would be the very meaning-giving rules of '+', which would have to be satisfied by any 'implementation'. One should be able to show that the rules do indeed verify for the particular implementation with the primitive recursive operator 'R', rather than saying that the former are 'derived' from the latter.

Then, with multiplication:

Example. (multiplication). We define

$$a \cdot b \equiv \mathrm{R}(b, 0, (x, y)(y + a))$$

Usual properties of the product '$a \cdot b$' can then easily be derived.

(...)

(Martin-Löf 1984, p. 73 & p. 74.)

It seems that the very notion of equality as used in *Intuitionistic Type Theory* is itself not without incoherences. For example, in another previous paper, Prof. Martin-Löf uses the notion of (abbreviatory) definitional equality to explain recursive definitions. When describing the properties of the type N of natural numbers he uses '$=_{\mathrm{def}}$' in the definition of primitive recursive functions over the type N:

1.9. *Natural Numbers*

N is a type, namely, the type of ***natural numbers***, 0 is an object of type N, and, if x is an object of type N, so is its successor $s(x)$. These are the first two Peano axioms.

Let C be a function which to an arbitrary natural number assigns a type. Then, given an object c of type $C(0)$ and a binary function g which to a natural number x and an object of type $C(x)$ assigns an object of type $C(s(x))$, we may define a unary function f which to a natural number x assigns an object of type $C(x)$ by the *recursion* schema

$$\begin{cases} f(0) & =_{\mathrm{def}} \quad c, \\ f(s(x)) & =_{\mathrm{def}} \quad g(x, f(x)). \end{cases}$$

If $C(x)$ represents a proposition for every natural number x, then f is the proof of the universal proposition $(\forall x \in N)C(x)$ which we get by applying the principle of mathematical *induction* to the proof c of $C(0)$ and the proof g of $(\forall x \in N)(C(x) \supset C(s(x)))$.

(Martin-Löf 1975a, p. 82.)

An attempt to relate Prof. Martin-Löf's kinds of definition to those of Frege (as we have seen in chapter 2), reveals an uncomfortable incoherence. Recalling Frege's account of abbreviatory (constructive, in Frege's words) definitions, only signs which had been already defined (or taken for granted) should appear at the righthand side of the abbreviatory definitional symbol '$\equiv$' (or '$=_{\text{def}}$' if one prefers). Now, in the definition of the function '$f(x)$' above, the sign 'f' is appearing in the righthand side of the definitional equality symbol.

Now, coming back to the employment of definitional equality in the definition of arithmetical functions given in Martin-Löf 1984, let us look more closely to the conceptual unease that seems to exist in *Intuionistic Type Theory*. As I have stated above, when defining the arithmetical functions of addition and multiplication, Prof. Martin-Löf uncovers what is to my view a misleading employment of the notion of definitional equality.

So, as we can see mainly from the example of *addition*, the *introduction* rule:

$$\frac{a \in \mathrm{N} \qquad b \in \mathrm{N}}{a + b \in \mathrm{N}}$$

which says under which conditions one can correctly use the symbol '+' (the first aspect of its use, according to my 'semantics of use'), therefore stating the '*grammar*' aspect, and the *reduction* rules:

$$\frac{a \in \mathrm{N}}{\mathtt{EVALUATE}(a + 0) = a \in \mathrm{N}}$$

$$\frac{a \in \mathrm{N} \qquad b \in \mathrm{N}}{\mathtt{EVALUATE}(a + \mathtt{succ}(b)) = \mathtt{succ}(\mathtt{EVALUATE}(a + \mathtt{PRED}(\mathtt{succ}(b)))) \in \mathrm{N}}$$

which state what kinds of conclusions one can draw from the 'sentence' '$a+b$', or, in other words, what is the *utility* of the symbol '+' (*consequence* aspect, which is the second of the aspects of use) should complete the 'definition by use'. Nevertheless, the definition of addition is given using a 'definitional' equality (which makes it look like the 'abbreviatory' or 'constructive' (case (1)) definition of Frege) and rules of *reduction* written in the more conventional notation such as:

$$\frac{a \in \mathrm{N}}{a + 0 = a \in \mathrm{N}} \qquad \frac{a \in \mathrm{N} \quad b \in \mathrm{N}}{a + b' = (a + b)' \in \mathrm{N}}$$

are said to be derived from the 'abbreviatory' definition of '+'.

Due to this confusion, the 'term' '$a + b$' is sometimes taken as 'canonical' (as in the *introduction* rule, which is meant to define 'canonical' elements), and sometimes as 'non-canonical' (as in the definitional equality $a + b \equiv \mathrm{R}(b, a, (x, y)y')$, given that 'R' is 'non-canonical'). The introduction of 'EVALUATE' would seem to establish a reasonable discipline in the use of definitions in the sense that a proper definition via *reduction* must have a 'canonical' expression ('succ' for the case of '+') on the righthand side of the equality sign '=', otherwise the sign for equality is to be understood as a 'definitional equality' ('$\equiv$'), or just an 'abbreviatory' definition.

The notion of 'canonical' elements is a fundamental one in *Intuitionistic Type Theory*. It seems to find coherent justification in most cases. However, when it comes to arithmetical expressions, the coherence seems to fade away.

(...) I shall call an expression, in whatever notation, canonical or normal if it is already fully evaluated, which is the same as saying that it has itself as value. Thus, in decimal arithmetic,

$$0, 1, ..., 9, 10, 11, ...$$

are canonical (normal) expressions, whereas

$$2 + 2, 2 \times 2, 2^2, 3!, 10^{10^{10}}, ...$$

are not. (...)

(Martin-Löf 1982, p. 157.)

where, in contrast to all the other cases, there is no notational difference between a canonical and a non-canonical element (cf. 'λ' vs. 'APPLY', 'pair' vs. 'FST' and 'SND', where a clear notational distinction is made between canonical and non-canonical elements, etc.). It is not made clear why the latter expressions are not canonical. That one accepts the argument on an intuitive basis does not imply that justification is fully coherent. For example, one can start by asking what is the type of expressions like '$2 + 2$', '2×2', etc.. To simply say that they are of type 'Natural Numbers' is to bring in more confusion, given that the type N of Natural Numbers as presented by Prof. Martin-Löf has '0' and a' as canonical elements and '$\mathrm{R}(b, a, (x, y)y')$' as non-canonical.

In fact, the operator 'R' seems to have been brought in to remedy this situation. It seems to be meant as a 'definitional instrument', to represent the definition of evaluation of arithmetical expressions, especially designed to avoid 'bad' definitions allowed by the more traditional notation where, as we have seen above, the equality sign does not appear to have the rôle of indicating a 'reduction'. This is why I think of the definition of '+', assuming 'R' as primitive, would be more coherently given as something like:

$$\mathtt{EVALUATE}(a + b) \equiv \mathrm{R}(b, a, (x, y)y')$$

where now both sides of the definitional equality have 'non-canonical' terms.

Prof. Aczel introduces an operator similar to such an 'R' in order to provide means of defining 'functions' over the set of natural numbers:

Lemma 5.11. There is an $\mathcal{F}$-functional $R : \mathcal{F}_0 \times \mathcal{F}_0 \times \mathcal{F}_2 \rightarrow \mathcal{F}_0$ such that

$$R(0, b, f) = b$$

and

$$R(s(a), b, f) = f(a, R(a, b, f))$$

for all a, b in $\mathcal{F}_0$ and f in $\mathcal{F}_2$.

(Aczel 1980, p. 49.)

And again similarly to Prof. Martin-Löf he gives a definition of addition as a mere abbreviation, leaving what seem to be the rules which express the distinguishing characteristic of such an arithmetical function, namely the (recursive) rules of *reduction*, a secondary rôle to play in characterising addition '+':

Using R we may define the usual arithmetical functions. For example we may define $+$ in $\mathcal{F}_2$ by

$$(a + b) = R(b, a, < s(y) \mid x, y >)$$

for all objects a, b. Then $(a + 0 = a)$ and $(a + s(b)) = s(a + b)$ for all objects a, b.

(Ibid., p. 50.)

In an attempt to provide a more coherent semantical framework, I have developed a modified set of rules to formalise constructive mathematics, taking *normalisation* as a primitive and fundamental semantic concept. The formal system is in fact a reformulation of Prof. Martin-Löf's *Intuitionistic Type Theory* (*ITT*) complying with my alternative operational semantics which I call 'semantics of use'. The main difference is that a type/proposition is not defined just by the *formation* and *introduction* rules, but the *reduction* rules also play an important part in making explicit the meaning of the type/proposition being defined. This is justified by the basic principle of my semantics of use: a proposition is defined by the two aspects of its use, which are the 'grammar' aspect (captured by natural-deduction *introduction* rules) and the *consequences* aspect (characterised by the 'internalised' *reduction* rules of *ITT*). Therefore, alternatively to Prof. Martin-Löf's *ITT*, my formal system requires the rules of *formation*, *introduction* and *reduction* to complete the definition of a particular type (proposition). For the case of inductive types the *reduction* rules equally represent one aspect of the meaning, and rules for induction/recursion operators such as Prof. Martin-Löf's 'R' are derived from the definition of the type

plus the set of higher-order rules of the 'Fix' operator, as has been done in de Queiroz & Maibaum 1987a for the case of natural numbers and lists, and shall be repeated in the next chapter. This set of higher-order rules were developed in de Queiroz & Maibaum 1987a and are presented again in the next chapter, with the purpose of showing how key programming notions such as infinite objects and general recursion can be handled in such a proof-theoretically presented formal system. They are meant to capture second-order objects which in Prof. Martin-Löf's formulation of *ITT* had to be dealt with either by the well-ordering rules ('**W**' type-forming operator) or by the rules for Universes ('**U**' type-forming operator).

5.5.2. Functions and Operations

Many of the incoherences of *Intuitionistic Type Theory* appear to stem from the semantical point of view, and one striking evidence for this is the difficulty in explaining the non-canonical operators. All the efforts made by the constructivists towards establishing the distinction between the concepts of function and operation seem to be overlooked, and despite being an approach to the formalisation of constructive mathematics the framework of *Intuitionistic Type Theory* does not seem to allow the distinction to be made. The first example of the definition of arithmetical symbols above shows 'predecessor' ('*pd*') as a 'function' on the natural numbers, where according to my 'semantics of use' it should in fact be inseparable from the very definition of the type of natural numbers, representing the second aspect of its *use* (therefore *meaning*), which is the *consequences* aspect: the *normalisation* effect of showing how the *eliminatory* operator 'PRED' acts on the *introductory* constructor 'succ' is taken in my *'Meaning-As-USE' Type Theory* as the formal counterpart to (the later) Wittgenstein's informal definition of one of the two aspects of a proposition, which is 'the *use* of the sign in the language-game'. The 'definition by use' or 'definition-as-presentation' of the type $\mathcal{N}$ would look like (as in de Queiroz 1988):

$\mathcal{N}$-formation

$$\mathcal{N}\ type \qquad \mathcal{N} = \mathcal{N}$$

$\mathcal{N}$-introduction

$$0 \in \mathcal{N} \qquad 0 = 0 \in \mathcal{N}$$

$$\frac{a \in \mathcal{N}}{\mathrm{succ}(a) \in \mathcal{N}} \qquad \frac{a = b \in \mathcal{N}}{\mathrm{succ}(a) = \mathrm{succ}(b) \in \mathcal{N}}$$

$\mathcal{N}$-reduction

$$\frac{a \in \mathcal{N}}{\mathrm{PRED}(\mathrm{succ}(a)) = a \in \mathcal{N}}$$

□

(*$\mathcal{N}$-induction* has been deliberately omitted.)

In the development of his *Frege structures* (Aczel 1980), Prof. P. Aczel defines a *Peano system* in a way similar to our *MAUSE* presentation of type '$\mathcal{N}$', except that a distinguishing function for the elements of the *system*, either '0' or '$PRED(a)$', is given explicitly.

Definition 5.9. A *Peano system* consists of an object 0 and $\mathcal{F}$-functions s, PRED and DEC : $\mathcal{F}_0 \to \mathcal{F}_0$ such that for all objects a

$$\mathrm{PRED}(s(a)) = a,$$
$$\mathrm{DEC}(s(a)) = \Delta_2$$

and

$$\mathrm{DEC}(0) = \Delta_1.$$

(Aczel 1980, p. 49.)

In fact, the use of 'predecessor' in the presentation of *$\mathcal{N}$-types*, *Peano systems*, *Numbers* (Beeson 1981, p. 168), follows the general pattern of having 'predecessor' as the 'DESTRUCTOR' of the '**constructor**' 'successor'. In the development of his

own formal system S of constructive mathematics Prof. M. Beeson defines/presents numbers via a series of axioms (N1–N5) which include the following:

(N2) $0 \in N \;\&\; \forall x \in N(s_N(x) \in N \;\&\; p_N(s_N(x)) = x) \;\&\; \forall x \in N(x \neq_N 0 \rightarrow p_N(x) \in N \;\&\; s_N(p_N(x)) = x)$ $\&\; \forall x \in N(s_N(x) \neq_N 0)$.

(Beeson 1981, p. 169.)

Breaking the whole 'axiom' (N2) into its conjunctive parts we have:

(i) $0 \in N$

(ii) $\forall x \in N(s_N(x) \in N \;\&\; p_N(s_N(x)) = x)$

(iii) $\forall x \in N(x \neq_N 0 \rightarrow p_N(x) \in N \;\&\; s_N(p_N(x)) = x)$

(iv) $\forall x \in N(s_N(x) \neq_N 0)$.

A careful look at the form of these axioms reveals the close parallels with a definition of the type $\mathcal{N}$ in our type theory:

a) (i) and the first half of (ii) find their counterparts in *$\mathcal{N}$-introduction*;

b) the second half of (ii) has the form of *$\mathcal{N}$-reduction*;

c) the second conjunct of (iii), namely '$s_N(p_N(x)) = x$', finds a parallel in the *induction* rules for non-inductive types such as **Π**- and **Σ**-*induction*. Putting it into an inference rule form:

$\mathcal{N}$-induction

$$\frac{a \in \mathcal{N}}{\mathtt{succ}(\mathtt{PRED}(a)) = a \in \mathcal{N}}$$

and we immediately find that the problematic case of having 'a' as '0' in the rule above makes it impossible to accept it as a sound rule. Observe that in the first conjunct of (iii) Prof. Beeson includes a conditional on 'x' being different from '0'. In the type-theoretic framework developed in this report the rôle of the 'conditional' is played by the type '+' of disjunction of two types, with constructors '`inl`', '`inr`', and destructor '`WHEN`', no 'discriminator function' (such as Prof. Aczel's 'DEC' in his *Peano system*) to distinguish between '0' and '`succ`(a)'. This is one of the reasons

why the induction rules for inductive types are not given directly in my own type theory, but resort to the higher-order type forming operator 'Fix'.

Now, in the framework of the semantics of use underlying the type theory I am attempting to offer as an alternative to Prof. Martin-Löf's, 'predecessor' is not seen as a 'function', but rather as an operation characterising the DESTRUCTOR, *eliminatory* operator for the natural numbers, which can only operate on the result of an *introduction* of the 'constructor' 'successor'. Similarly to 'APPLY' for 'λ', 'FST' and 'SND' for 'pair', 'HEAD' and 'TAIL' for 'cons', etc., 'PRED' should be better regarded as an (*eliminatory*) 'operation' (over the *introductory* operation 'succ') rather than as a 'function'. This should perhaps ratify the claim that there must be a distinction between the concepts of 'function' and 'operation', emerging out of metamathematical studies on how to formalise constructive mathematics.[35]

[35]See, e.g., Prof. M. Beeson's metamathematical studies on how to formalise constructive mathematics, Beeson 1981, 1985. In the former he says:

'We find that functions are not conceptually fundamental, since they can be defined in terms of sets and operations. A fundamental point of our analysis is that in the past, insufficient care has been taken to distinguish operations and functions, and this has led to certain confusions.'

(Beeson 1981, p. 149.)

In the latter he points out:

'Some people feel that each operation should "come equipped" with a set called the "intended domain"; (...) Others do not insist on this feature of operations. It is our own view that this is a natural feature not of operations, but of functions — two notions which have not always been carefully distinguished. (...) We point out that there is a natural example of an *operation* without any fixed "intended domain": the *pairing operation*, which produces from any two objects x and y, a new object $< x, y >$, from which x and y can be immediately recovered ("read off"). There are two operations p_0 and p_1 such that $\mathrm{p}_0(< x, y >) = x$ and $\mathrm{p}_1(< x, y >) = y$.'

(Beeson 1985, p. 34.)

It is also pointed out in Beeson 1981, p. 184, that a list of the primitive concepts of Prof. Martin-Löf's theories would not include 'function'. According to Prof. Beeson, those primitive concepts would be set, proof and operation. In any case, given that *Intuitionistic Type Theory* is to be seen as a foundational framework for constructive mathematics — where the concept of 'operation' would seem to be more sensibly considered more primitive than that of 'function' — it really looks

There are at least two important reasons for that: first of all, the concept 'function' is not supposed to be taken as a primitive concept in *Intuitionistic Type Theory*. The actual purpose of a formalisation of constructive mathematics such as the work developed by Prof. Martin-Löf, as I understand it, is to revise and make precise the whole of the basic principles of (constructive) mathematics, taking *proofs*, *propositions* (*types*) and *judgements* as primitive concepts, from which other mathematical concepts, including 'function', would be derived. Nevertheless, in Prof. Martin-Löf's 'school' there seem to be different views on this particular matter, where the *elimination* rules are sometimes said to show how to define 'functions' as in the general explanation of the rôle of *elimination* rules:

(...) The elimination rule shows how we may define functions on the set defined by the introduction rules. (...)

(Martin-Löf 1984, p. 24.)

or in more specific cases as the explanation of non-canonical operator 'Ap' for Π-*types*:

Π-*equality*

$$\frac{a \in \mathbf{A} \qquad \begin{array}{c}[x \in \mathbf{A}] \\ b(x) \in \mathbf{B}(x)\end{array}}{\mathtt{Ap}((\lambda x)b(x), a) = b(a/x) \in \mathbf{B}(a)} \qquad \frac{c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{c = (\lambda x)\mathtt{Ap}(c, x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

The first equality rule shows how the new function Ap operates on the canonical elements of $(\Pi x \in \mathbf{A})\mathbf{B}(x)$. (...)

(Ibid., p. 29.)

(The emphasis is mine.)

Instead of being explained as 'the rules showing how we may define functions on the set (type) defined by the introduction rules' as above, the *elimination* rules are

rather self-contradicting to present an explanation of some basic rules of type definition, such as the *elimination* rules, using the concept of 'function'.

Perhaps one should also recall L. Wittgenstein's '(Operations and functions must not be confused with each other.)', §5.25, p. 42 of the *Tractatus*.

sometimes explained as being the rules expressing an 'induction principle', as in B. Nordström's account of *Intuitionistic Type Theory* as a programming logic. It is said explicitly:

> (...) The *elimination rules* for F expresses an induction principle for $F(A_1, ..., A_n)$. If $C(x)$ is a family of sets indexed by elements in $F(A_1, ..., A_n)$ the elimination rule gives conditions for the judgement $e \in C(p)$, where e is (in general) a noncanonical expression involving the selector associated with F. If we read $C(x)$ as property of elements in $F(A_1, ..., A_n)$ we can see the elimination rule as giving conditions for proving that $C(p)$ is true for $p \in F(A_1, ..., A_n)$. (...)
>
> (Nordström 1986, p. 8.)

The fact that one needs induction rules to ensure *minimality* of the types seems to be absolutely clear as long as one understands a framework such as *Type Theory* as a system of definitional mechanisms to define/present types. In that sense it is not surprising that some of the *elimination* and *equality* rules of *Intuitionistic Type Theory* have the form of formal counterparts to inductive (as minimality) clauses. (When I say that *some* of the *elimination* and *equality* rules do bear resemblance with inductive schema whereas some others do not, I have in mind the counterexample of the usual **Π**-*elimination* and -*equality* with 'APPLY', despite its 'more general' presentation with 'F' bearing such resemblance with an inductive schema.) Nonetheless, it seems difficult not to accept that one also needs to make explicit how DESTRUCTORS operate on constructors, which is the rôle of the *reduction* rules. In my own *'Meaning-As-USE' Type Theory* those two aspects of a type, namely the *use* as *utility* aspect, and the *minimality* aspect, have each their rôle to play in the framework, as well as their corresponding formal counterparts as *reduction* and *induction*, respectively.

On other occasions, the 'operations' concerning the *elimination* rules are considered as new 'constants' introduced to define functions. Prof. Martin-Löf himself sometimes refers to those *eliminatory* operators as *constants*:

Π-*elimination*

$$\frac{c \in (\Pi x \in \mathbf{A})\mathbf{B}(x) \quad a \in \mathbf{A}}{\mathtt{Ap}(c,a) \in \mathbf{B}(a)} \qquad \frac{c = d \in (\Pi x \in \mathbf{A})\mathbf{B}(x) \quad a = b \in \mathbf{A}}{\mathtt{Ap}(c,a) = \mathtt{Ap}(d,b) \in \mathbf{B}(a)}$$

(...) We have to explain the meaning of the new constant **Ap** (**Ap** for application). (...)

(Martin-Löf 1984, pp. 28–9.)

(The emphasis is mine.)

In an attempt to show the usefulness of *Intuitionistic Type Theory* in solving programming problems, J. Smith refers to these *eliminatory* operators as constants. E.g., in:

(...) We want to define functions on the natural numbers by primitive recursion and we therefore introduce the constant **rec**, which is computed according to the rules

$$\begin{cases} \mathtt{rec}(0,c,e) & = & c \\ \mathtt{rec}(s(a),c,e) & = & e(a,\mathtt{rec}(a,c,e)) \end{cases}$$

So, defining f by $f(n) \equiv \mathtt{rec}(n,c,e)$ corresponds to the introduction of f by the primitive

$$\begin{cases} f(0) & = & c \\ f(s(a)) & = & e(a,f(a)) \end{cases}$$

Let $C(x)$ be a type for $x \in N$. We then have the following rule, which should be obvious from the computation rule for '**rec**',

$$\frac{n \in N \quad c \in C(0) \quad e(x,y) \in C(s(x)) \quad [x \in N, y \in C(x)]}{\mathtt{rec}(n,c,e) \in C(n)}$$

For instance, defining *plus* by $plus(m,n) \equiv \mathtt{rec}(n,m,(x,y)s(y))$ we can use $N-elimination$, by putting $C(x) \equiv N$ and $e(x,y) \equiv s(y)$, to get $plus(m,n) \in N[m \in N, n \in N]$. (...)

(Smith 1983, p. 448.)

(The emphasis is mine.)

Even more recent attempts to provide a uniform explanation of the rôle of *elimination* rules in *Intuitionistic Type Theory* remain unclear as to what those *eliminatory* operators are to be understood as. E.g., in an introductory book on their own account of *Type Theory*, the Göteborg Group present an explanation of the *elimination* rules as:

> First, we introduce some general rules about equality and substitution and then, for each type forming operation, we give four kinds of rules.
>
> 1. The *formation rules* for A describe under which conditions we may infer that A is a type and when two types A and B are equal.
> 2. The *introduction rules* define the type A in that they prescribe how the canonical elements are formed and when two canonical elements are equal.
> 3. The *elimination rules* show how to prove a proposition about an arbitrary object of the type.
> 4. The *equality rules* describe the equality relation on elements of the type. Besides the equalities between canonical elements which are given when the type is defined, the computation rules for the noncanonical primitive constants give rise to equality rules.
>
> (Nordström, Smith & Petersson 1986, p. 33.)

(The emphasis is mine.)

Secondly, by looking at *elimination* rules as rules to provide means of defining 'functions', one has to go back to the same problematic point regarding definitions of data types which would presumably be overcome by an operational semantics such as that of *ITT*, and get involved in the same old problem (typical of algebraic theories of abstract data type specification), which is to have to explain what is the result of applying 'functions' like 'PRED' to an argument such as '0', 'TAIL' and 'HEAD' to 'nillist', 'POP' and 'TOP' to 'emptystack', and so on. Looking at all these 'operations' as 'names' for the *eliminatory* operations, non-canonical operators, therefore seeing them as inseparable from the meaning of the type being defined together with the *introduction* rules, indicating the second aspect of the use (meaning) of the type, and operating only on the result of an *introduction*, it becomes natural to see that those standard problems are just artificially created. *Judgements*

like '$\texttt{PRED}(0) \in \mathcal{N}$', '$\texttt{TAIL}(\texttt{nillist}) \in \mathbf{List(A)}$' and '$\texttt{POP}(\texttt{emptystack}) \in \mathbf{Stack(A)}$' could never be reached because the rules of definition (by use) would not include the possibility of arriving at such *judgements*. In a type-theoretic framework based on the semantics of use, such as my *'Meaning-As-USE' Type Theory*, judgements of the form '$\texttt{PRED}(0) \in \mathcal{N}$', '$\texttt{TAIL}(\texttt{nillist}) \in \mathbf{List(A)}$' and '$\texttt{HEAD}(\texttt{nillist}) \in \mathbf{A}$', '$\texttt{POP}(\texttt{emptystack}) \in \mathbf{Stack(A)}$' and '$\texttt{TOP}(\texttt{emptystack}) \in \mathbf{A}$', would not be 'reachable' through the rules of manipulation of the respective types.

In a *denotations*-based semantical framework, due to the unnecessary detour through concepts such as functions, sets, embeddings, etc., 'spurious' constant values like '`error`', '$\perp$', etc., have to be introduced. Complex structures are then constructed to deal with what is called 'error-recovery', 'error-propagation', and the study of parameterisation and structuring is made significantly more difficult than one would normally expect it to be. And this is exactly because in this setting, where *interpretations* are supposed to give meaning, one is forced to ask such questions about what one could call 'false' problems. Instead of asking what can be produced by the rules of construction and destruction, one is tempted to ask about the *values* of expressions. Similarly, the need to use partial algebras and, to some extent, the abandonment of initial algebraic semantics in favour of certain 'loose' semantics, are both examples of complex approaches to solving these, as it were, 'false' problems (a similar view on th (non-)handling of 'errors' was developed by Veloso & Maibaum, and is reported in, e.g., Veloso 1985).

5.6. Final Remarks

It has been a most illuminating experience to use the lessons learnt from many years of continuous studies attempting to make precise the notion of data type, namely the investigation of the theory of abstract data types, to reformulate what is certainly a seminal work on the foundations of logic based on the concepts of *types* and *judgements*: *Intuitionistic Type Theory*. The classical mathematical notion of set, which was felt sufficiently satisfactory for most developments in mathematics, proved inadequate to support the notion of data type as understood by computer scientists.

Instead of defining a type by what it contained, it was felt that a data type definition had to convey the information demonstrating how the elements of the type were to be used, computed, etc.. A reasonably self-contained body of concepts developed mainly in the seventies aiming at gaining a better understanding of the concept of data types in mathematical terms began to show that data type definitions were worth studying as a theory in itself. With the help of algebraic techniques, the notion of data type as defined by what could be done with its elements proved a convenient alternative to the mathematical notion of set as defined by what the set contains. The semantical concern, however, was still directed towards 'meaning as (numerical) value', and many difficulties were encountered to explain the meaning of oddly formed 'terms' such as the top of an empty stack, the predecessor of '0', the node of an empty tree, etc..

Now, *Intuitionistic Type Theory*, the result of an attempt to clarify the syntax and semantics of intuitionistic mathematics via *types* and *judgements*, developed almost simultaneously with the algebraic approaches to data types, came with a most interesting and highly relevant semantical innovation, which was the shift on the focus of attention from (numerical) values to *judgements*. Prior to the question of what the value of an expression was, which nevertheless still remained but at a different level, there was a question of what kinds of *judgements* could be made from a type definition (*presentation*). Such an interesting innovation was to open new horizons for a better understanding of what is to be taken as relevant from a type presentation. Many of the difficulties in giving a 'meaning' (as numerical value) to certain oddly formed terms would then be overcome. However, at the level of type definitions/presentations, the approach advocated by Prof. Martin-Löf in his *Intutionistic Type Theory* proved a step back to the mathematical principle that a set is to be defined by what the set contains, as we have seen from various passages mentioned above.

In the next two chapters it is my main objective to present the features of a reformulated *Type Theory* where the good lessons from both sides come together in a more coherent framework to express constructions. For instance, instead of Prof.

Martin-Löf's pattern for his *Intuitionistic Type Theory*:

formation

introduction

elimination

equality

what I call *'Meaning-As-USE' Type Theory* prescribes the following rules, with corresponding purpose:

formation rules: which describe the type's *grammar*, showing how to form the type expression.

introduction rules: as a description of the (canonical) elements' *grammar*, showing how to form the value-expression – `constructor`.

reduction rules: which demonstrate the elements' *use* by describing the effect of DESTRUCTORS on the `constructors`.

induction rules: 'meta'-reasoning statement saying 'the type contains the elements given by the rules of *introduction* and only those, therefore the meaning (*use*, *utility*) of the type is only the one given by the *reduction* rules'.

By describing in more detail the rôle of each of these rules in the type-theoretic framework and the corresponding implications for a view on the logical foundations of computation, I shall attempt to convince the reader that with the help of the identification of *propositions* with *types* one can even provide a reasonable framework to present logics as theories, not via axioms but via natural deduction-like proof rules. Bearing in mind that by presenting a *proposition* one is presenting the rules for its major logical connective, and that theories can be presented as types, it is reasonable to think of a logic presentation as being the presentation of as many theories/types as the number of different connectives the logic might have. Furthermore, by having a reasonable discipline on how to define/present types which would seem to avoid the recourse to denotations in order to find the distinguishing characteristic and fix the meaning of the type defined, it shall become clear to what extent the type-theoretic framework here developed is helpful in understanding some of the key issues dealt with by theories of abstract data type specification.

6 *'Meaning-As-USE' Type Theory and Computer Programming*

6.1. Introduction

Looking for connections between proofs and computations, and taking proof theory as a 'coding mechanism' potentially much richer than the λ-Calculus, one can find in Prof. P. Martin-Löf's *Intuitionistic Type Theory (ITT)*, a proof-theoretic presentation of constructive mathematics, where *meaning* and *proof* are fundamental concepts based on *operations* rather than on *denotations*, a potential candidate to fulfil a philosophical as well as a logical claim[1]. Prof. P. Martin-Löf's seminal work on the connections between constructive mathematics and computer programming (Martin-Löf 1982), which has since the middle senventies been the object of careful study for many contemporary philosophers, logicians as well as computer scientists, has served as an enormous source of inspiration for what I report here. It appears to

[1] '(...) any interpretation still hangs in the air along with what it interprets, and cannot give it any support. Interpretations by themselves do not determine meaning. (...)', §198, Part I, p. 80^e of Wittgenstein's *Investigations*,

and

'a proof *is* a computation', p. 354 of Bishop's *A Constructivist Manifesto*, Bishop 1967.

constitute a pioneering attempt at constructing a 'verificationist' theory of meaning for the mathematical language, which may offer an extra benefit in providing proper means of talking about operational semantics of programming languages. However, despite being 'verificationist', Prof. Martin-Löf's *ITT* does not seem to be fully coherent and congenial with Wittgenstein's verificationist use-based semantical principle, given that the semantic rôle of proof rules is kept within Gentzen's tradition, which says that *introduction* rules constitute definitions (de Queiroz 1988f).

In an attempt to provide a more coherent and (hopefully) more convincing account of the basic principles of constructive mathematics, I say that proof theory, insofar as being an instrument of operational semantics, should be regarded as something which comes before constructive mathematics in the sense that the latter, which is in itself just a body of informal concepts concerning the mathematical activity, can be made precise by means of the former. For this reason I also say that proof theory can be looked at as a suitable starting point for the development of reasonable foundations for operational semantics of computer programming languages where the *operational* intuition seems to prevail regardless of the language used.

One of my main underlying objectives, as has been repeatedly stated previously, is to offer a 'non-dialogical' reading of Ludwig Wittgenstein's writings on the notion of *meaning* for the language of mathematics (Cf. Dialogue and Game-like interpretations), resulting in a reformulation of *Intuitionistic Type Theory* into a formal system called *'Meaning-As-USE' Type Theory*, which seems to be relevant for a better understanding of interesting notions such as *computation* and *constructive reasoning* only by means of logical rules. Such a reformulation of Prof. P. Martin-Löf's formalisation of *constructive* mathematics, would seem to offer itself as an alternative account of the traditional proof-theoretic account of meaning, suggesting 'Proof Theory and Computer Programming' as an alternative paradigm to Prof. Martin-Löf's own 'Constructive Mathematics and Computer Programming'.

In previous chapters, as well as on other occasions (de Queiroz 1988, 1987c, 1988abf), I have attempted to discuss the formalisation of fundamental aspects of *use* (meaning) of logical constants via proof-theoretic inference rules, insisting on

the insufficiency of the *assertability conditions* (which find their formal counterpart in the natural deduction *introduction* rules) with respect to the two aspects of Wittgenstein's later account of propositions.

Based on a single fundamental argument (*reduction* rules as semantical instruments) the aim of this chapter is twofold:

Firstly I shall prove the derivability of natural deduction-like *elimination* rules of a type-theoretic framework such as *Intuitionistic Type Theory*, from the corresponding *introduction* and *reduction* rules. This should be regarded as an attempt to offer an answer to an unsettled question due to Gentzen-Prawitz:

> Gentzen suggested that it should be possible to formulate in what respect the elimination rule is uniquely determined by the corresponding introduction rule. Indeed, it strikes one as very likely that the elimination rules are in some sense the strongest rules for drawing consequences from an assertion; one sense in which this may be true was suggested in a conjecture by Prawitz. However, the question of the way (if any) in which the elimination rules occupy such a unique position is still unsettled.
>
> (Prawitz 1977, p. 33.)

Secondly, I propose a general fixpoint operator for *Intuitionistic Type Theory* which is meant to handle infinite objects in that particular proof-theoretic presentation of constructive mathematics. So, instead of having a principle of (finite) induction for each infinite type ('R' for Natural Numbers, 'LISTREC' for Lists, etc.) and operators to deal with higher-order principles such as well-orderings ('**W**' operator) and universes ('**U**' operator), one can have a general fixpoint operator which can be used to construct well-orderings and universes (types of types), and from which operators for proofs by finite induction over inductively defined types can be derived. I prove the derivability of such operators for the case of Natural Numbers, Lists and Stacks.

6.1.1. Equality rules of ITT

As to the particular presentation of constructive mathematics constituted by *Intuitionistic Type Theory*, one is led to draw a natural conclusion which is that the

question of which one is primitive, and which one is a derived rule, or, in other words, which rules determine the meaning of the connective/type/proposition, does not seem to be easily answered for the case of Prof. Martin-Löf's formulation of *ITT*. By taking the *normalisation* rules to be the main semantical rules I am in effect proposing to offer a, what appears to me, plausible answer to the that question, which seems to help in making the desired connections between proof theory and computer programming (I shall be referring to proof theory as the natural deduction style of presenting logics *à la* Prof. Prawitz, where for each connective separately there are rules of *introduction* and *elimination*, and rules operate on sequents with antecedent empty): I say with the traditional view of proof theory that the *elimination* rules are indeed a mere consequence of the definition of the connective, but alternatively to this same tradition, I also say that the definition of the connective is not complete until the *reduction* (*normalisation*) rules (or *equality* rules, as they are called in *ITT*), which show how particular *elimination* rules operate on the result of an *introduction* inference (capturing one of the fundamental aspects of the use of the connective), are provided. I shall be doing this by showing how to derive *elimination* rules from the corresponding *introduction* and *reduction* rules (*equality* rules of *ITT*) within Prof. Martin-Löf's *Intuitionistic Type Theory*. For this enterprise, I need to add new rules to *ITT* keeping the basic philosophy of the whole system and, equally important, maintaining its consistency (a consistency proof for my type-theoretic framework without the higher-order 'type of types' can be done similarly to the one given in Martin-Löf 1975a). For example, to be able to prove that the *elimination* rules for a certain logical connective defined within *ITT* (type/proposition) are derivable from its *introduction* and *reduction* (*equality*) rules, I need to add at least two rules which are perfectly in accordance with the meaning of judgements of the form '$a = b \in \mathbf{A}$' ('a and b are equal elements of the type $\mathbf{A}$', at p. 5 of Martin-Löf 1984.), their common premiss:

$$= -\mathcal{L} \quad \frac{a = b \in \mathbf{A}}{a \in \mathbf{A}} \qquad\qquad = -\mathcal{R} \quad \frac{a = b \in \mathbf{A}}{b \in \mathbf{A}}$$

Assuming we have these rules available in the formal system, which Prof. Martin-Löf takes as implicitly given from the meaning of judgements of the form '$a = b \in \mathbf{A}$'[2], we can put into more formal terms the conceptual argument regarding the insufficiency of the *introduction* rules and the fundamental rôle of the *reduction* (*normalisation*) rules[3]. In fact, many have already pointed out the insufficiency of the so-called *introduction* rules, and by insisting on this very weakness of Prof. Martin-Löf's verificationist semantics, I have attempted to offer more convincing arguments for my criticisms of the (Gentzen's style) semantics of *Intuitionistic Type Theory*. This is because I look at the meaning of a proposition as something that would come from two aspects of *use*: the conditions for assertion, which are usually formalised by natural deduction *introduction* rules, and the conclusions that can be drawn from the proposition, which would seem to be captured by rules of *normalisation* (the effect of particular *elimination* rules on the result of *introduction* inferences). And this appears to amount to the same thing as saying that a definition of the circumstances under which a logical sentence having the connective as the major connective can be asserted does not suffice as far as the understanding of its (the sentence's) meaning is concerned[4]. What I do here is fundamentally different

---

[2]'(...) and from $c \in \mathbf{A}$, which is *implicit* in $a = c \in \mathbf{A}$ (...)' (the *emphasis* is mine), Martin-Löf 1984, p. 17.

[3]Cf. Wittgenstein's 'And the meaning of '$(x).fx$' is made clear by our insisting on 'fa' 's following from it.', *RFM*, Part I, §11, p. 42, which remarks on the effect of an *elimination* inference on the result of an *introduction* inference making the meaning of the corresponding proposition clear. That seems to be exactly the rôle of the *reduction* rules.

[4]Cf. Prof. Prawitz's comment on Prof. Dummett's argument about the aspects relating meaning and use within the language of mathematics: 'Dummett points out that quite generally, there are two aspects of the use of an (assertive) sentence: (1) the conditions under which it can be correctly asserted, and (2) the commitments made by asserting it. In the case of mathematics, aspect (1) is expressed in the rules for inferring the sentence, and aspect (2) in the rules for drawing consequences from the sentence. When we learn to use sentences, what we learn are these *two* kinds of rules.', (the *emphasis* is mine), p. 7 of Prawitz 1977,

Cf. also:

'(...) For if we are asked what is the meaning of the word 'and', at least in the purely conjunctive sense (as opposed to, e.g., its colloquial use to mean 'and then'), the answer is said to be *completely*

from Prof. R. Backhouse's 'method to construct the elimination and computation rules from the formation and introduction rules for a type in Martin-Löf's theory of types' (Abstract of Backhouse 1985). The semantical standpoint for my investigations, as it should be clear by now, is rather different and does not rely on Gentzen's semantics.

6.1.2. Elimination on the constructors

The modifications performed on the rules for Natural Numbers reflect how fundamentally different from Prof. Martin-Löf's is my own view on the rôle of proof rules: besides the differences on the semantic rôle, I regard the proof rules as operating on the 'constructors' (canonical elements/proofs) of the type/proposition, rather than on the types themselves. As in, for example, the *elimination* rule for inductive types (which are the ones where the difference is clear) operates on the 'constructor' ('succ' for Natural Numbers, 'cons' for Lists, etc.) and not on the whole type as is the case for Prof. Martin-Löf's formulation: for non-inductive types in *ITT* such as Π and Σ types, the *elimination* rules happen to eliminate not only the 'constructor' ('λ' and 'pair' respectively), but also the types themselves. (Observe that the type 'N' does not appear in the conclusion of N-*elimination*, whereas the type '$\mathcal{N}$' does

given by saying that (i) from any pair of statements P and Q we can infer the statement formed by joining P to Q by 'and' (which statement we hereafter describe as 'the statement P-and-Q'), that (ii) from any conjunctive statement P-and-Q we can infer P, and (iii) from P-and-Q we can always infer Q. Anyone who has learnt to perform those inferences knows the meaning of 'and', for there is simply nothing more *to* knowing the meaning of 'and' than being able to perform these inferences. (...)', p. 38 of Prior 1960.

Note that in both accounts a proposition or a logical connective is characterised by two *independent* aspects, namely the *assertability conditions* and the *consequences*. Even if those two aspects are not recognised as being the sufficient information one needs to uniquely characterise a particular logical connective, it is quite clear that they constitute at least the necessary amount of information needed for the characterisation. What does not sound intuitive, however, is to suggest that 'it should be possible to formulate in what respect the elimination rule is uniquely *determined* by the corresponding introduction rule'.

appear in the premiss as well as in the conclusion of $\mathcal{N}$-*elimination*). By adopting the view where the *elimination* rules operate on the 'constructors' rather than the type itself one does not affect the coherence of the conceptual framework. On the other hand, an immediate benefit of this view is to obtain a more coherent system of rules, at least in the sense that the *equality* rules indeed represent the 'internalisation' of the proof-theoretic (meta-level) notion of *reduction* rules developed by Prof. D. Prawitz which are meant to reflect the logical principle of inversion of '*elimination* over *introduction*' (Prawitz 1965, 1971). This does not seem to be the case for Prof. Martin-Löf's N-*equality* rules:

N-*equality*

$$\frac{d \in \mathrm{C}(0) \qquad \begin{matrix}[x \in \mathrm{N}, y \in \mathrm{C}(x)] \\ e(x,y) \in \mathrm{C}(x')\end{matrix}}{\mathrm{R}(0, d, (x,y)e(x,y)) = d \in \mathrm{C}(0)}$$

$$\frac{a \in \mathrm{N} \qquad d \in \mathrm{C}(0) \qquad \begin{matrix}[x \in \mathrm{N}, y \in \mathrm{C}(x)] \\ e(x,y) \in \mathrm{C}(x')\end{matrix}}{\mathrm{R}(a', d, (x,y)e(x,y)) = e(a, \mathrm{R}(a, d, (x,y)e(x,y))) \in \mathrm{C}(a')}$$

despite having stated the correspondence between the *equality* rules of *ITT* and Prof. Prawitz's *reduction* rules[5].

The N-*elimination* rule is presented in such a way that it is the 'type' N which is being eliminated:

N-*elimination*

$$\frac{c \in \mathrm{N} \qquad d \in \mathrm{C}(0) \qquad \begin{matrix}[x \in \mathrm{N}, y \in \mathrm{C}(x)] \\ e(x,y) \in \mathrm{C}(x')\end{matrix}}{\mathrm{R}(c, d, (x,y)e(x,y)) \in \mathrm{C}(c)}$$

[5]'In the interpretation of sets as propositions, the formation rules are used to form propositions, introduction and elimination rules are like those of Gentzen, and the equality rules correspond to the reduction rules of Prawitz.', p. 24 of Martin-Löf 1984.

whereas in my $\mathcal{N}$-*elimination* rules:

$\mathcal{N}$-*elimination*

$$\frac{a \in \mathcal{N}}{\mathtt{PRED}(\mathtt{succ}(a)) \in \mathcal{N}} \qquad \frac{a = b \in \mathcal{N}}{\mathtt{PRED}(\mathtt{succ}(a)) = \mathtt{PRED}(\mathtt{succ}(b)) \in \mathcal{N}}$$

the 'destructor' 'PRED' is eliminating the 'constructor' succ'.

A straightforward advantage that can be obtained by adopting such a different point of view is the fact that the membership relation '$\in$' is clearly seen to be decidable, as opposed to the undecidable '$\in$' of *Intuitionistic Type Theory*. It is well known that by allowing (non-canonical) elements of inductive types to 'code' finite as well as transfinite induction over the corresponding type ('$\mathtt{R}(c, d, (x, y)e(x, y)) \in \mathrm{C}(c)$' for the type of Natural Numbers, and '$\mathtt{T}(c, (x, y, z)d(x, y, z)) \in \mathrm{C}(c)$' for the type of Well-Orderings[6]), respectively), Prof. Martin-Löf's Type Theory has an undecidable type membership relation. For the case of my *'Meaning-As-USE' Type Theory* higher-order concepts such as induction and recursion are treated separately by a proper higher-order set of rules (rules for 'Fix', as we shall see below). Therefore, the membership relation '$\in$' is decidable for 'ordinary' types (non-'Fix' types) of my *'Meaning-As-USE' Type Theory* (a decidability proof can be given along the same lines as the one given in Martin-Löf 1975a. The undecidability is not 'removed' with a stroke of magic, so to speak, but it is simply 'shifted up' to the higher-order 'type of types', where one would expect higher-order notions to be properly treated[7].

There is also clearly a great benefit in terms of the semantics: my 'semantics of use', where the effect of 'destructors' on 'constructors' (*reduction* rules) is taken as part of the semantics, presents itself as a suitable operational semantics for abstract data type specifications.

---

[6] p. 81 of Martin-Löf 1984.

[7] This relevant feature of my *Type Theory* was uncovered in an informal discussion with Dr Leen Helmink and Dr René Ahn, both from Phillips Research Laboratories, Eindhoven, The Netherlands, to whom I feel grateful for the intellectual stimulus.

It is worth recalling that by introducing 'PRED' I am not bringing in an extra problem of how to deal with 'PRED(0)' because this term can never be derived as a member of $\mathcal{N}$. This is because I do not take 'PRED' as a 'function' as is customary in algebraic specification of natural numbers as an abstract data type, in functional programming, and even in the usual (Gentzen's style) semantics of *Intuitionistic Type Theory*[8]. So, I do not have to say what value should such a 'function' take with '0' as argument. Instead, I say that 'PRED' is an operation of elimination over the introductory 'succ'. And *judgement* on terms can only be made by the rules of manipulation. Furthermore, there is no way of reaching the *judgement* '$\mathtt{PRED}(0) \in \mathcal{N}$', just because there is no possible sequence of rule applications for arriving at it.

Coming back to the differences between my own view and Prof. Martin-Löf's on the effect of *elimination* rules, I see the 'constructor' as the object of the *elimination*, rather than the type itself. One of the benefits of this particular view is that the *reduction* rules for inductive types can be seen as 'evaluation' rules, such as '$\mathtt{PRED}(\mathtt{succ}(a)) = a$' is the evaluation rule for natural numbers, '$\mathtt{HEAD}(\mathtt{cons}(a, b)) = a$' and '$\mathtt{TAIL}(\mathtt{cons}(a, b)) = b$' are evaluation rules for **List**, and similarly '$\mathtt{TOP}(\mathtt{push}(a, b)) = a$' and '$\mathtt{POP}(\mathtt{push}(a, b)) = b$' are evaluation rules for **Stack**. The 'computation' rules 'R' (or 'NATREC' as I prefer to call it) for Natural Numbers, 'LISTREC' for Lists, 'STACKREC' for Stacks, etc., will turn out to be derived rules from a general fixpoint operator (like 'Fix' as we shall see further on) plus those *reduction* rules specific to a type.

As we have seen from the previous chapter and shall see more in the next one, this view proves helpful in relating proof theory, in particular *Intuitionistic Type Theory*, with specification of abstract data types (which, one can say, essentially constitutes a modern view of computer programming), where the notions of 'constructor', 'destructor' and *reduction* rule (equation) are fundamental notions. Looking at *reduction* rules as instruments of the semantics, one can offer an explanation

[8] 'The elimination rule shows how we may define functions on the set defined by the introduction rules.', Id. Ibid., (the emphasis is mine).

of why such a general form of equations is used in specification of abstract data types (like the ones seen above), by saying that these rules are the extra information one needs to be able to completely specify the operational semantics of the particular type being defined. They complete the information given by the 'syntax' part of an abstract data type specification (which can be seen as *introduction* rules), by saying how the elements of the type are used (Stacks are used in such a way that when POP'ing a stack on which an element was just push'ed, one gets the stack one had before the push operation). One can also provide a simpler semantics for the specification of abstract data types, by avoiding standard problems such as giving meaning to 'odd'-terms like 'PRED(0)', 'POP(emptystack)', 'TAIL(nillist)', etc..

6.2. Derivability of elimination rules in Type Theory

Notational Remarks. (i) The sign '□' denotes the end of a definition, and '■' indicates the end of a proof. (ii) Judgements included in '[ ]' are assumptions that are discharged by the particular rule.

DEFINITION 6.2.1. A rule of *ITT* consists of a relation between a set of (possibly empty) hypothetical judgements called *assumptions*, together with a set of judgements called *premisses*, and a judgement taken as the result of the application of the rule (*conclusion*). □

DEFINITION 6.2.2. A rule 'R' is derivable from a set of rules $R_1, \ldots, R_n$ if its conclusion can be related to its premisses (and possible assumptions) by applying only rules from $R_1, \ldots, R_n$. □

LEMMA 6.2.3. The $\mathcal{N}$-*elimination* rules:

$$\frac{a \in \mathcal{N}}{\mathtt{PRED}(\mathtt{succ}(a)) \in \mathcal{N}} \qquad \frac{a = b \in \mathcal{N}}{\mathtt{PRED}(\mathtt{succ}(a)) = \mathtt{PRED}(\mathtt{succ}(b)) \in \mathcal{N}}$$

are derivable from the presentation of $\mathcal{N}$.

PROOF.

(i) For the first $\mathcal{N}$-*elimination*

$$\frac{a \in \mathcal{N}}{\mathsf{PRED}(\mathsf{succ}(a)) \in \mathcal{N}}$$

we can derive it as

$$\dfrac{\dfrac{a \in \mathcal{N}}{\mathsf{PRED}(\mathsf{succ}(a)) = a \in \mathcal{N}}\,(1)}{\mathsf{PRED}(\mathsf{succ}(a)) \in \mathcal{N}}\,(2)$$

where

(1) $\mathcal{N}$-*equality*

(2) $= -\mathcal{L}$

(ii) For the second $\mathcal{N}$-*elimination*

$$\frac{a = b \in \mathcal{N}}{\mathsf{PRED}(\mathsf{succ}(a)) = \mathsf{PRED}(\mathsf{succ}(b)) \in \mathcal{N}}$$

we derive it as

$$\dfrac{\dfrac{\dfrac{\dfrac{a = b \in \mathcal{N}}{a \in \mathcal{N}}\,(1)}{\mathsf{PRED}(\mathsf{succ}(a)) = a \in \mathcal{N}}\,(3) \qquad a = b \in \mathcal{N}}{\mathsf{PRED}(\mathsf{succ}(a)) = b \in \mathcal{N}}\,(5) \qquad \dfrac{\dfrac{\dfrac{a = b \in \mathcal{N}}{b \in \mathcal{N}}\,(2)}{\mathsf{PRED}(\mathsf{succ}(b)) = b \in \mathcal{N}}\,(4)}{b = \mathsf{PRED}(\mathsf{succ}(b)) \in \mathcal{N}}\,(6)}{\mathsf{PRED}(\mathsf{succ}(a)) = \mathsf{PRED}(\mathsf{succ}(b)) \in \mathcal{N}}\,(7)$$

where

(1) $= -\mathcal{L}$

(2) $= -\mathcal{R}$

(3) $\mathcal{N}$-*equality*

(4) $\mathcal{N}$-*equality*

(5) *Transitivity*

(6) *Symmetry*

(7) *Transitivity*

■

Similarly to the above case, where we proved the derivability of the *elimination* rules from *introduction* plus *reduction* rules for the case of the type of Natural Numbers presented according to my 'semantics of use', now I want to demonstrate that the same argument applies to $\mathbf{\Pi}$-*elimination* rules. For the case of a non-inductive type such as $(\mathbf{\Pi}x \in \mathbf{A})\mathbf{B}(x)$, I need to make use of what I call 'rules of *induction*' for non-inductive types, which are needed to capture the 'meta-reasoning' which says 'the meaning of the type defined is just the one defined by the corresponding rules of *introduction* and *reduction*'[9]. This 'meta-level' reasoning is captured by the general fixpoint operator 'Fix' for the case of inductive types, and for these *induction* rules for the case of non-inductive types.

The rule of *induction* for the $\mathbf{\Pi}$ type, for example, is the well-known η-rule of the λ-Calculus:

$\mathbf{\Pi}$-*induction*

$$\frac{c \in (\mathbf{\Pi}x \in \mathbf{A})\mathbf{B}(x)}{(\lambda x)\mathsf{APPLY}(c, x) = c \in (\mathbf{\Pi}x \in \mathbf{A})\mathbf{B}(x)}$$

For the case of $\mathbf{\Sigma}$, a rule such as

[9]This particularly crucial point concerning my 'semantics of use' for the *'Meaning-As-USE' Type Theory*, grew out of many illuminating conversations with Dr M. Smyth, to whom I feel greatly indebted. Two joint papers entitled 'Induction Rules for Non-Inductive Types in Type Theory' (de Queiroz & Smyth 1988) and 'The Inductive Rôle of λ-Calculus' η-rule' (in preparation) as well as the section on 'Non-Inductive Types' below describe the enlightening results of those discussions.

$$\frac{c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}{< \mathrm{p}(c), \mathrm{q}(c) >= c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}$$

which is essentially the same rule as my Σ-*induction*, is said to be a derived rule[10].

DEFINITION 6.2.3. Let $\mathbf{A}$ be a type and $\mathbf{B}(x)[x \in \mathbf{A}]$ be a family of types over $\mathbf{A}$. The type $(\Pi x \in \mathbf{A})\mathbf{B}(x)$ representing the product of the type $\mathbf{A}$ and the family of types $\mathbf{B}(x)$ is defined as:

Π-*formation*

$$\frac{\mathbf{A}\ \textit{type} \quad \begin{matrix}[x \in \mathbf{A]} \\ \mathbf{B}(x)\ \textit{type}\end{matrix}}{(\Pi x \in \mathbf{A})\mathbf{B}(x)\ \textit{type}} \qquad \frac{\mathbf{A} = \mathbf{C} \quad \begin{matrix}[x \in \mathbf{A}] \\ \mathbf{B}(x) = \mathbf{D}(x)\end{matrix}}{(\Pi x \in \mathbf{A})\mathbf{B}(x) = (\Pi x \in \mathbf{C})\mathbf{D}(x)}$$

Π-*introduction*

$$\frac{\begin{matrix}[x \in \mathbf{A}] \\ b(x) \in \mathbf{B}(x)\end{matrix}}{(\lambda x)b(x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)} \qquad \frac{\begin{matrix}[x \in \mathbf{A}] \\ b(x) = d(x) \in \mathbf{B}(x)\end{matrix}}{(\lambda x)b(x) = (\lambda x)d(x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

Π-*reduction*

$$\frac{a \in \mathbf{A} \quad \begin{matrix}[x \in \mathbf{A}] \\ b(x) \in \mathbf{B}(x)\end{matrix}}{\mathrm{APPLY}((\lambda x)b(x), a) = b(a/x) \in \mathbf{B}(a)}$$

Π-*induction*

[10]'(...) We can now prove that the inference rule ... is derivable. It is an analogue of the second Π-*reduction* rule, which could also be derived, provided the Π-rules were formulated following the same pattern as the other rules. (...)', Martin-Löf 1984, pp. 62–3. (note the remark on the similarity of this rule with Π-*induction* rule, which is confirmed by my formulation of both as *induction* rules)

$$\frac{c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{(\lambda x)\mathtt{APPLY}(c, x) = c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

□

I shall demonstrate how the rules of Π-*elimination* can be derived from the above definition of the type $(\Pi x \in \mathbf{A})\mathbf{B}(x)$. By doing this I shall fulfil J. Smith's claim:

The functions are formed by means of lambda-abstraction.

$$\Pi - \textit{introduction}: \qquad \frac{b(x) \in \mathbf{B}(x) \quad [x \in \mathbf{A}]}{(\lambda x)b(x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

Function application is expressed by the constant *ap*, which is computed in the usual way:

$$ap((\lambda x)b(x), a) = b(a/x)$$

From this computation rule, we get the elimination rule for Π:

$$\Pi - \textit{elimination}: \qquad \frac{a \in \mathbf{A} \quad c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{ap(c, a) \in \mathbf{B}(a)}\ .$$

(Smith 1983, p. 449.)

LEMMA 6.2.4. Π-*elimination* rules

Π-*elimination*

$$\frac{a \in \mathbf{A} \quad c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{APPLY}(c, a) \in \mathbf{B}(a)} \qquad \frac{a = b \in \mathbf{A} \quad c = d \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{APPLY}(c, a) = \mathtt{APPLY}(d, b) \in \mathbf{B}(a)}$$

are derivable from Π-*introduction* plus Π-*reduction* rules, and the Π-*induction* rule.

PROOF. (i) The first rule

$$\frac{a \in \mathbf{A} \quad c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{APPLY}(c, a) \in \mathbf{B}(a)}$$

is derived as

$$\dfrac{a \in \mathbf{A} \qquad \dfrac{\dfrac{c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{\lambda x.\mathtt{APPLY}(c,x) = c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}(1)}{\lambda x.\mathtt{APPLY}(c,x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}(2)}{\dfrac{\mathtt{APPLY}(\lambda x.\mathtt{APPLY}(c,x),a) = \mathtt{APPLY}(c,a) \in \mathbf{B}(a)}{\mathtt{APPLY}(c,a) \in \mathbf{B}(a)}(4)}(3)$$

where

(1) Π-*induction*

(2) $= -\mathcal{L}$

(3) Π-*reduction*

(4) $= -\mathcal{R}$

(ii) To prove that the second rule

$$\dfrac{c = d \in (\Pi x \in \mathbf{A})\mathbf{B}(x) \quad a = b \in \mathbf{A}}{\mathtt{APPLY}(c,a) = \mathtt{APPLY}(d,b) \in \mathbf{B}(a)}$$

is also a derivable rule should be easy and is left to the reader. ■

Now, as I shall need the type $(\Sigma x \in \mathbf{A})\mathbf{B}(x)$ for the remainder of my discussion on the connections between proof theory and computer programming, I define it as:

DEFINITION 6.2.5. Let $\mathbf{A}$ be a type and $\mathbf{B}(x)$ $[x \in \mathbf{A}]$ be a family of types over $\mathbf{A}$. The type $(\Sigma x \in \mathbf{A})\mathbf{B}(x)$ representing the disjoint union of a family of types is defined as:

Σ-*formation*

$$\dfrac{\mathbf{A}\ \textit{type} \quad \overset{[x \in \mathbf{A}]}{\mathbf{B}(x)\ \textit{type}}}{(\Sigma x \in \mathbf{A})\mathbf{B}(x)\ \textit{type}} \qquad \dfrac{\mathbf{A} = \mathbf{C} \quad \overset{[x \in \mathbf{A}]}{\mathbf{B}(x) = \mathbf{D}(x)}}{(\Sigma x \in \mathbf{A})\mathbf{B}(x) = (\Sigma x \in \mathbf{C})\mathbf{D}(x)}$$

Σ-*introduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{B}(a)}{\mathtt{pair}(a,b) \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)} \qquad \frac{a = c \in \mathbf{A} \quad b = d \in \mathbf{B}(a)}{\mathtt{pair}(a,b) = \mathtt{pair}(c,d) \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}$$

Σ-*reduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{B}(a)}{\mathtt{FST}(\mathtt{pair}(a,b)) = a \in \mathbf{A}} \qquad \frac{a \in \mathbf{A} \quad b \in \mathbf{B}(a)}{\mathtt{SND}(\mathtt{pair}(a,b)) = b \in \mathbf{B}(a)}$$

Σ-*induction*

$$\frac{c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{pair}(\mathtt{FST}(c), \mathtt{SND}(c)) = c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}$$

□

It is routine to prove

LEMMA 6.2.6. Σ-*elimination* rules

Σ-*elimination*

$$\frac{c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{FST}(c) \in \mathbf{A}} \qquad \frac{c = d \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{FST}(c) = \mathtt{FST}(d) \in \mathbf{A}}$$

$$\frac{c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{SND}(c) \in \mathbf{B}(\mathtt{FST}(c))} \qquad \frac{c = d \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{SND}(c) = \mathtt{SND}(d) \in \mathbf{B}(\mathtt{FST}(c))}$$

are derivable from Σ-*introduction* plus Σ-*equality* rules, and the Σ-*induction* rule.

■

6.3. Non-Inductive Types

Throughout the whole of the previous chapter it has been argued that the meaning of a type in a type-theoretic framework is made explicit by the corresponding *reduction*

rules. This leaves open the question of how to explain the rôle of certain equality rules which are essentially different from *normalisation* (*reduction*) equality. The η-rule of the λ-calculus, for example, is an equality rule where the nature of equality appears to be of a very different character from β-rule equality. Let us call these, for the moment, 'special' equality rules for non-inductive types such as Π and Σ. And, indeed, when attempting to say that the meaning of Π-types, whose elements are λ-terms, can be explained by the *β-normalisation* rule alone, one is almost inevitably confronted with the question of what rôle the analogue of the λ-Calculus η-rule is meant to play in a type-theoretic framework. In this section I suggest that the right way to view the 'special' equality rules is as formalised induction rules for non-inductive types. I shall attempt to demonstrate that their rôle is, in effect, to ensure *minimality* of the type. Recalling Dedekind's quest for minimality, when in a letter responding to Keferstein who criticised his use of the notion of chain, claiming it would be 'unnecessary', proposing an amendment in Dedekind's definition of simply infinite sets which, in the words of van Heijenoort (Dedekind 1890, p. 98), would amount to the abandonment of mathematical induction, we can see that Dedekind rightly pointed out the need to have a minimality clause when defining a set:

> (6) I have shown in my reply (III), however, that these facts are still far from being adequate for completely characterizing the nature of the number sequence N. All these facts would hold also for every system S that, besides the number sequence N, contained a system T, of arbitrary additional elements t, to which the mapping φ could always be extended while remaining similar and satisfying $\varphi(T) = T$. But such a system S is obviously something quite different from our number sequence N, and I could so choose it that scarcely a single theorem of arithmetic would be preserved in it. What, then, must we add to the facts above in order to cleanse our system S again of such alien intruders t as disturb every vestige of order and to restrict it to N? (...)
>
> (Dedekind 1890, p. 100.)

The analogue of η-rule for Π-types seems to be useful in formalising the usually informal *inductive* clause of the definition of types by saying that 'every element of the *Π-type* is a *λ-abstraction* term', and therefore, no meaning beyond what is indicated by the β-rule is to be taken as the meaning of Π-types.

6.3.1. Induction on the Π*-type of* λ*-terms and on the* Σ*-type of* `pair`*-terms*

When the *introduction* rule(s) for a certain type/proposition contain(s) the type both in the premiss and in the conclusion, as, e.g., is the case for Natural Numbers, the rules of *introduction* and *reduction* define inductively the corresponding type. For the case of inductive types the rules for construction of canonical elements of the types embody an inductive step. This is not the case for non-inductive types such as Π and Σ types.

DEFINITION 6.3.1. Let a Π type be a type defined as:

Π-*formation*

$$\frac{\mathbf{A}\ \textit{type} \quad \begin{matrix}[x \in \mathbf{A}] \\ \mathbf{B}(x)\ \textit{type}\end{matrix}}{(\Pi x \in \mathbf{A})\mathbf{B}(x)\ \textit{type}} \qquad \frac{\mathbf{A} = \mathbf{C} \quad \begin{matrix}[x \in \mathbf{A}] \\ \mathbf{B}(x) = \mathbf{D}(x)\end{matrix}}{(\Pi x \in \mathbf{A})\mathbf{B}(x) = (\Pi x \in \mathbf{C})\mathbf{D}(x)}$$

Π-*introduction*

$$\frac{\begin{matrix}[x \in \mathbf{A}] \\ b(x) \in \mathbf{B}(x)\end{matrix}}{(\lambda x)b(x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)} \qquad \frac{\begin{matrix}[x \in \mathbf{A}] \\ b(x) = d(x) \in \mathbf{B}(x)\end{matrix}}{(\lambda x)b(x) = (\lambda x)d(x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

Π-*reduction*

$$\frac{a \in \mathbf{A} \quad \begin{matrix}[x \in \mathbf{A}] \\ b(x) \in \mathbf{B}(x)\end{matrix}}{\mathtt{APPLY}((\lambda x)b(x), a) = b(a/x) \in \mathbf{B}(a)}$$

Π-*induction*

$$\frac{c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{(\lambda x)\mathtt{APPLY}(c, x) = c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

□

Observe that in the rules of **Π**-*introduction* there is no 'inductive step', where the type being defined would be both in the premiss and in the conclusion of the rules of *introduction*.

The **Π** type just defined is an example of a non-inductive type as understood in this section.

Now, if one accepts that the meaning of the **Π**-type of λ-terms can be given via *rules* (*operations*), the rule of *β-reduction* seems to be its 'main' semantical rule, simply because it shows the *use* (operational meaning) of 'λ'. And, indeed, according to my own type-theoretic characterisation of the semantics of logical constants based on Wittgenstein's semantical dictum *meaning-is-use*, where the *normalisation* rules are taken to be semantical rules in a similar way to Prof. Hintikka's *semantical games* because in both cases the meaning of logical signs is taken to be inseparable from *logical consequence*, the β-rule of the λ-calculus gives the (operational) meaning of **Π**-types of λ-terms because it demonstrates what can be *obtained* from a λ-term, or what is the *utility* of forming such a term. Both semantical accounts, the *semantical games* and the *semantics of use*, emerge from (the later) Wittgenstein's notion of 'language-games', which seems to be a great source of inspiration for an 'anti-interpretation' account of meaning for the mathematical language. However, the rôle of the η-rule '$\lambda x.\mathtt{APPLY}(c, x) = c$', which is certainly one of the fundamental rules of the λ-calculus, is not easily justified as part of the definition of 'λ' from the point of view of operational semantics, unless one accepts its *inductive* rôle of saying 'every element of a **Π**-type is a λ-term'[11]. This makes it easier to see why such an

[11]The question of whether or not the η-rule of the λ-calculus is indeed a rule to express the extensionality principle seems to be very much open to debate. In my type-theoretic framework, however, it seems to be working as an 'induction' rule, similarly to Profs. Hindley & Seldin's analogue rule of Combinatory Logic:

'REMARK 8.6 (The role of (η)). In λ-calculus, (η) ranked as an extensionality-principle equivalent to (*ext*) and (ζ). But in combinatory logic it is not an extensionality-principle at all. All it says is that *every term is equal to an abstraction*, and by itself this has nothing to do with extensionality.

inductive rule is of the same nature as another induction-based rule, this time for the Σ-type of *pairing*-terms: the rule '$\mathtt{pair}(\mathtt{FST}(c), \mathtt{SND}(c)) = c$', which seems to be saying that 'every element of a Σ-type is a pair-term'.[12]

I present the corresponding rules of *induction* for non-inductive types, which are needed to capture the 'meta-reasoning' of saying 'the meaning of the type defined is just the one defined by the corresponding rules of *introduction* and *reduction*'. This 'meta'-reasoning is captured by the general fixpoint 'Fix' for the inductive types, and by these *induction* rules for the case of non-inductive types.

The rule of *induction* for the Π type, for example, is an analogue of the well-known η-rule from the λ-calculus:

(...)', p. 79 of Hindley & Seldin 1986. (The *emphasis* is mine.)

[12]They seem to be rules of the same character, e.g., cf. Prof. Martin-Löf's placement of both as distinct rules of equality from other kinds of equality such as *conversion*-equality: '(...)

2. Definitional equality

(...) The outcome of the foregoing analysis is that the rule (ξ) is unacceptable as a rule of conversion. Of course, we are free to define many different relations between terms and call them convertibility relations, but my claim is that only one of these correctly formalizes the informal notion of definitional equality. (...)

2.2. ***Further rules of conversion which do not correctly formalize the notion of definitional equality as understood in this paper***

2.2.1. Curry's rule of η-conversion,

$$(\lambda x)(b(x)) \text{ conv } b$$

provided the variable x does not occur free in the term b, and the combinatory axioms which correspond to it. Equally unacceptable is the corresponding rule for cartesian products,

$$(p(c), q(c)) \text{ conv } c,$$

although, as shown below, the abstract objects denoted by $(p(c), q(c))$ and c can be proved to be *identical*. (...)'

(Martin-Löf 1975b, p. 100.)

In the semantical explanations of his *Intuitionistic Type Theory*, however, Prof. Martin-Löf places both the analogue of β-rule and the analogue of η-rule (first rule above), under Π-*equality* rules (Martin-Löf 1984, p. 29.).

Π-*induction*

$$\frac{c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{(\lambda x)\mathtt{APPLY}(c, x) = c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

A rule of essentially the same nature is part of the definition of the Σ type forming operator:

Σ-*induction*

$$\frac{c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{pair}(\mathtt{FST}(c), \mathtt{SND}(c)) = c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}$$

These rules of *induction* are not strictly new within type-theoretic frameworks, given that for the case of 'Π' the rule of Π-*induction*, which is clearly an analogue of the η-rule of the λ-Calculus, is already available in *Intuitionistic Type Theory* as presented in Martin-Löf 1984, but playing a rather different rôle[13]. The rules of *equality* for the type forming operator Π include a rule with such an 'inductive' structure:

Π-*equality*

$$\frac{c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{(\lambda x)\mathtt{Ap}(c, x) = c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

For the case of Σ, a rule such as

$$\frac{c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}{< \mathtt{p}(c), \mathtt{q}(c) >= c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}$$

[13]see the second Π-*equality* rule in Martin-Löf 1984, p. 29, where it is said that '(...) Similarly, the second rule is needed to obtain a notation, $\mathtt{Ap}(c, x)$ for a program of which we know only the name 'c' (...)'.

where '< , >' is the pairing constructor, and 'p' and 'q' are the 'eliminatory' operations ('projections') corresponding respectively to the first and the second components of the ordered pair, which is essentially the same rule as our Σ-*induction*, is said to be a derived rule[14].

6.3.2. Π-*types and* λ-*theories*

I have attempted in this section to motivate the use of formalised induction rules for non-inductive types such as Π and Σ types, as formal counterparts to the usual informal inductive clause of definition of a type, suggesting, e.g., that the analogue of the λ-Calculus η-rule seems to play such an inductive rôle in a type-theoretic framework. In a more detailed account of my type-theoretic treatment of typical λ-Calculus notions given in the next chapter I discuss how one can justify not just the inductive rôle of the η-rule but the rôle of other fundamental notions and rules of the calculus (it shall also be discussed in a paper still in preparation: de Queiroz & Smyth 198?a). For example, I endeavour to explain the ξ-rule as the rule which says when two canonical elements of a Π-type are equal, (see second rule of Π-*introduction*, DEFINITION 6.3.1 above), rather than a rule of *weak extensionality*, as is often done in mathematical studies of λ-Calculus (see, e.g., Barendregt 1981, p. 32).

6.4. Inductively Defined Types

When the *introduction* rule(s) for a certain logical connective[15] contain(s) the type both in the premiss and in the conclusion, as, e.g., is the case for Natural Numbers,

[14]'(...) Example (proof of the converse of the projection laws) We can now prove that the inference rule ... is derivable. It is an analogue of the second Π-*equality* rule, which could also be derived, provided the Π-rules were formulated following the same pattern as the other rules. (...)', pp. 62–3 of Martin-Löf 1984. Note the remark on the analogy made of this rule with the Π-*equality* rule, which is confirmed by my formulation of both as *induction* rules for the non-inductive types Π and Σ.

[15]Perhaps Prof. Aczel's term '*system*' (*lambda system*, *pairing system*, *Peano system*, etc.)

the rules of *introduction* and *reduction* (*equality*) define inductively the corresponding type (set). As it happens, for each inductively defined type there is a principle of primitive recursion, or, in the more general case, structural recursion. What I shall be doing in this section is to show how we can derive operators for this kind of recursion (primitive recursion) over types defined by their *introduction* plus *reduction* rules using a general fixpoint operator 'Fix' which handles higher-order objects in my reformulated version of Prof. Martin-Löf's *Intuitionistic Type Theory*. My procedure will be similar to Prof. P. Aczel's in his development of *Frege structures*. Hence, no explicit induction rules will be required for these types, unlike the cases of Π, Σ, and $+$ above.

It turns out that primitive recursive operators for inductively defined types such as the type of Natural Numbers, Lists, Stacks, etc., all follow a certain general pattern, and this indicates that one can devise algorithms for deriving recursion operators for inductively defined types in *ITT*, as P. Dybjer has tried to show (Dybjer 1987).

An important point about inductively defined types, especially in connection with computer programming, is that they are meant to represent infinite objects. As it happens, the problem of representing infinite objects in Type Theory has been looked at from similar perspectives by P. Dybjer as well as by the PRL group (Constable & Mendler 1985 and Mendler, Panangaden & Constable 1986, respectively). To be able to develop my unified framework, I shall follow a similar trend, by introducing a general fixpoint operator on the level of types and a general fixpoint operator on the level of elements of types. In other words, I shall be defining a typed version of a higher order type forming operator which I suggestively call 'Fix' (in reference to the notion of fixpoints). As we shall see later on, this recursive operator can be seen as a particular case of the well-ordering type forming operators, which can be looked at as operators for 'internalising' second-order principles,

would be more appropriate, given that, like Prof. Aczel, I am chiefly concerned with the 'constructors' and 'destructors' of elements of each type rather than with the type itself, and each '*system*' is defined by its corresponding '*reduction*' rule(s), as it were (Aczel 1980).

in the same way as the modal operator for necessity ('$\Box$') when interpreted as a *provability* operator. By introducing the well-ordering type forming operator '**W**' and the higher-order type forming for the notion of 'universes' (operator '**U**') in *ITT*, perhaps following one of Bishop's principles of how to formalise constructive mathematics:

> (...) Another important foundational problem is to find a formal system that will efficiently express existing predictive[16] mathematics. I think we should keep the formalism as primitive as possible, starting with a minimal system and enlarging it only if the enlargement serves a genuine mathematical need.
>
> (...)
>
> Thus we can develop Brouwer's ideas as a metatheory. In particular, Brouwer's result that $f : R \to R$ is continuous emerges as a metatheorem, which states that if we construct a continuous function $f : R \to R$ having a certain property, then we can construct within Σ a continuous f having the same property. In fact more is true, the canonical function which is constructed by unwinding the formalised proof of the existence of f is continuous. After these results have been proved as metatheorems, it becomes constructively meaningful to build a new formal system to incorporate the metatheory. (...)
>
> (p. 60 & p. 68 of Bishop 1970.)

Prof. Martin-Löf has made an attempt to provide a unified framework to describe infinite objects in Type Theory without having to introduce the notion of partial objects (Martin-Löf 1982 and 1984), and so keeping the identification 'propositions-are-types' intact.

However, these operators, as they are presented in Martin-Löf 1984, do not seem to offer a clear account of how useful this internalisation can be. By unifying the definition of the proof rules for the well-ordering and the higher-order (universes) type forming operators into a single higher-order operator (or Fixpoint operator, as we prefer to call it) I shall try to motivate the use of these higher-order notions

[16] '(...) Thus by 'constructive' I shall mean a mathematics that describes or predicts the results of certain finitely performable, albeit hypothetical, computations within the set of integers. If a word is needed to delimit this special variety of constructivism, I propose the term 'predictive'. (...)', p. 53 of Bishop 1970.

more effectively. It is my hope that this reformulation will help in understanding the important rôle of such higher-order notions, especially in relating a particular proof theory with inductive definitions like *ITT*, with computer programming notions such as general recursion, lazy evaluation, infinite objects, etc..

As I have just said, I shall define a higher-order type forming operator 'Fix', which will capture both the notions of well-orderings like '**W**' and higher-order objects like '**U**', so it can be demonstrated how the notions of infinite objects and partiality are dealt with in my own *'Meaning-As-USE' Type Theory*. The basic idea is that the notion of partiality exists in terms of iteration: at every stage of the 'computation' the objects are total in themselves, but each stage is part of the whole process, and so, each stage contains partial information about the whole computation. This is in fact another way of looking at infinite objects as *limits* of successive iterations similar to the idea of 'lazy evaluation'. Moreover, such a higher-order operator seems to be exactly what would allow definitions in a *Type Theory*-like framework to go beyond primitive recursion. A proper formulation of such an operator would perhaps get around the criticisms regarding the restrictions imposed by Prof. Martin-Löf's *Intuitionistic Type Theory* which, as it is well known, does not provide means for definitions beyond primitive recursion, despite the attempt to motivate the use of higher-order objects by the introduction of operators such as '**W**' and '**U**'.

This view will allow us to have a more general account of the kind of objects that can be described in *Intuitionistic Type Theory*, given that the finite objects will turn out to be particular cases of infinite objects, in the same way as the finitary induction/recursion operators appear as particular cases of the infinitary induction/recursion operator. So, 'finite' types like Prof. Martin-Löf's $\mathrm{N_m}(\mathrm{m} = 0, 1, 2, ...)$ as defined in the chapter on Finite sets, pp. 65–8 of Martin-Löf 1984 will turn out to be particular cases of the *potentially infinite* '$\mathcal{N}$', which in its turn, can be *implemented* in terms of the general fixpoint type forming operator 'Fix' plus its 'basis' $\{0\}$.

6.5. Adding a Fixpoint Operator to MAUSE

The operator on the level of types will be defined as:

DEFINITION 6.5.1. Let ∇ be a type and $\mathbf{B(X)}$ be a type expression depending on the type variable $\mathbf{X}$. The higher-order type forming operator 'Fix' is defined as:

Fix-*formation*

$$\frac{\begin{array}{c}[\mathbf{X}\ \textit{type}]\\ \nabla\ \textit{type} \quad \mathbf{B(X)}\ \textit{type}\end{array}}{\mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B(X)}\ \textit{type of types}} \qquad \frac{\begin{array}{c}[\mathbf{X}\ \textit{type}]\\ \nabla_1 = \nabla_2 \quad \mathbf{B_1(X)} = \mathbf{B_2(X)}\end{array}}{\mathsf{Fix}\ \mathbf{X}.\nabla_1 + \mathbf{B_1(X)} = \mathsf{Fix}\ \mathbf{X}.\nabla_2 + \mathbf{B_2(X)}}$$

where

(i) '∇', '∇_1' and '∇_2' are constant type expressions (no type variables allowed to occur in them) which must be inhabited. This is to ensure the *predicativity* of the operator Fix, in the sense that '∇' will serve as the basis from which the *big* type will be built; furthermore, because ∇, ∇_1 and ∇_2 must be constant type expressions (no type variables), the (undesirable) type Fix $\mathbf{X}.\mathbf{X}$ cannot be formed. The only restriction imposed on ∇ is that it cannot be (Prof. Martin-Löf's) $\mathbf{N_0}$, $\varnothing$ or any non-inhabited type, allowing the type '$\mathsf{Fix}\ \mathbf{X}.\mathbf{N_0} + \mathbf{B(X)}$' to be formed, even though $\mathbf{N_0}$, $\varnothing$, . . ., etc., have no element from which the elements of each of the iterative stages would then be formed.

(ii) '$\mathbf{X}$' is a type variable,

(iii) '$\mathbf{B(X)}$' is a type expression formed with Π, Σ, $\times$ (particular case of Σ), $\rightarrow$ (particular case of Π), and $+$.

The 'canonical' elements of the 'type of types' '$\mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B(X)}$' will be types whose elements are gen'erated by the operator 'gen' on the level of the elements:

Fix-*introduction*

$$\varnothing \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B(X)} \qquad \varnothing = \varnothing \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B(X)}$$

$$\frac{\begin{array}{c}[z \in \mathbf{X}] \\ g(z) \in \nabla + \mathbf{B}(\mathbf{X})\end{array}}{\mathsf{gen}(z.g(z)) \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

$$\frac{\begin{array}{c}[z \in \mathbf{X}] \\ g_1(z) = g_2(z) \in \nabla + \mathbf{B}(\mathbf{X})\end{array}}{\mathsf{gen}(z.g_1(z)) = \mathsf{gen}(z.g_2(z)) \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

$$\frac{\mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}{\nabla + \mathbf{B}(\mathbf{A}/\mathbf{X}) \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

$$\frac{\mathbf{A}_1 = \mathbf{A}_2 \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}{\nabla + \mathbf{B}(\mathbf{A}_1/\mathbf{X}) = \nabla + \mathbf{B}(\mathbf{A}_2/\mathbf{X}) \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

where '$\varnothing$' is the 'base' for the type of iterative types represented by '$\mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})$' in the same way as Church's base ordinal '0_0' is the base for the constructive second number class (Church 1938), and '∇' is the *small* type from which the *big* type (type of types) '$\mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})$' is iteratively built.

For example: (i) for Natural Numbers, $\nabla = \{0\}$ and $\mathbf{B}(\mathbf{X}) = \mathbf{X}$;

(ii) for Lists of $\mathbf{A}$, $\nabla = \{\mathtt{nillist}\}$ and $\mathbf{B}(\mathbf{X}) = \mathbf{A} \times \mathbf{X}$;

(iii) for Stacks of $\mathbf{A}$, $\nabla = \{\mathtt{emptystack}\}$ and $\mathbf{B}(\mathbf{X}) = \mathbf{A} \times \mathbf{X}$;

(iv) for Binary Trees of $\mathbf{A}$, $\nabla = \{\mathtt{niltree}\}$ and $\mathbf{B}(\mathbf{X}) = \mathbf{A} \times \mathbf{X} \times \mathbf{X}$.

This restriction on the existence of an inhabited ∇ in the definition of an inductive type via the 'Fix' operator seems to provide a more reasonable alternative to the 'positiveness' restriction used by Dybjer as well as by the PRL group (Dybjer 1987, and Mendler, Panangaden & Constable 1986). This can be justified as follows: the syntactic restriction on the definition of inductive types via Fix-like operators requiring that $\mathbf{X}$ occurs only positively in the expression taken by Fix, says that $\mathbf{X}$ must not occur *negatively* in any case, which is to say that $\mathbf{X}$ must not occur as a type quantified over by a 'Π' type forming operator. As the function space type forming operator '$\rightarrow$' can be seen as a particular case of 'Π' within type theory, the restriction says that $\mathbf{X}$ must not occur as the antecedent of an 'arrow' as in e.g.

$\mathbf{X} \to \mathbf{X}$, $\mathbf{X} \to \mathbf{A}$, and $\mathbf{A} \to (\mathbf{X} \to \mathbf{B})$. This is where the term *positive* comes from: the antecedent part of an 'arrow' type-expression is called its *negative* part, whereas the consequent part is called the *positive* part.

However, such a *positiveness* restriction on the occurrence of '$\mathbf{X}$' in the expression $\mathbf{B}(\mathbf{X})$ of e.g. '$\mathsf{Fix}\ \mathbf{X}.\mathbf{B}(\mathbf{X})$' (where there is no '$\nabla$') does not seem to be sufficient to avoid undesirable definitions such as '$\mathsf{Fix}\ \mathbf{X}.\mathbf{X}$'. In this case $\mathbf{X}$ is occurring *strictly positively* in $\mathbf{B}(\mathbf{X})$ (i.e., it is not the antecedent of any arrow) and yet there seems to remain some *impredicativity*. No such *impredicativity* can occur in my reformulated definition[17].

[17] An anonymous referee of one of my joint papers with Dr Maibaum has remarked on the claim that the existence of an inhabited ∇ in the definition of an inductive type via the Fix operator would seem to be providing a reasonable alternative to the 'positiveness' restriction. He/she then argues that 'this is a misunderstanding of the reason for positiveness' and finishes by saying 'Anyway why would you want to forbid $\mathsf{Fix}\ \mathbf{X}.\mathbf{X}$?'. The simplest answer I could think of giving for that particular question needs reference to Prof. Martin-Löf's own concern on the wellfoundedness and predicativeness of the theory of types. When referring to the impredicativeness of an early version of the intuitionistic theory of types he says:

'In its first version, the present theory was based on the strongly impredicative axiom that there is a type of all types whatsoever, in symbols, $V \in V$, which is at the same time a type and an object of that type. This axiom had to be abandoned, however, after it had been shown to lead to a contradiction (...). The incoherence of the idea of a type all types whatsoever made it necessary to distinguish, like in category theory, between small and large types. Thus the universe V appears, not as the type of all types, but as the type of small types, whereas V itself and all types which are built up from it are large. *This makes the types wellfounded and the theory predicative.*'

(Martin-Löf 1975a, pp. 73–74.)

(The *emphasis* is mine.)

I do not want a type such as $\mathsf{Fix}\ \mathbf{X}.\mathbf{X}$ in my type theory because I want to have only wellfounded types and a predicative type theory. For the empty type, which would correspond to '$\mathbf{N}_0$' of *Intuitionistic Type Theory*, one has '$\varnothing$' in the base case of the definition of 'Fix'. This is a different approach to accommodating the notion of (potential) infinity into a type-theoretic framework than Prof. Martin-Löf's recent 'nonstandard type theory' presented at the *Logic Colloquium '89* in Padova, whose written account has just come to my hands: Martin-Löf 1989. It is perhaps too early to state how big is the difference, but with 'Fix' as defined here we have no circular definitions.

Now, to complete the definition of Fix, I give its *reduction* rules as:

Fix-*reduction*

At the level of elements

$$\frac{a \in \mathbf{A} \quad \mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X}) \quad \begin{matrix}[z \in \mathbf{X}] \\ g(z) \in \nabla + \mathbf{B}(\mathbf{X})\end{matrix}}{\mathtt{UNWIND}(\mathsf{gen}(z.g(z)), a) = g(a/z) \in \nabla + \mathbf{B}(\mathbf{A}/\mathbf{X})}$$

At the level of types

$$\frac{\mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}{\mathtt{PREDTYPE}(\nabla + \mathbf{B}(\mathbf{A}/\mathbf{X})) = \mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

which means that to obtain the elements of the 'next' type in the increasing chain of types generated iteratively as members of Fix $\mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})$, one has to 'run' through all the elements of the last type already 'acquired' ($\mathbf{A}$ for the case of the first Fix-*reduction* rule above) applying the 'generator' 'gen' to each of its elements. And, to obtain the next type in the increasing chain of types, one applies a kind of higher-order successor operation '$\nabla + \mathbf{B}(\mathbf{A}/\mathbf{X})$', where the previous one is obtained by 'PREDTYPE' in a way which is analogous to successor and predecessor of Natural Numbers.

□

The same referee who asked the question aforementioned certainly held another view on the rôle of predicativeness and wellfoundedness in a type-theoretic framework, which is revealed in his/her following remarks: '(...) the authors seem to confuse two notions of infinite objects: well-founded ones (which can be represented in Martin-Löf's type theory) and non-well-founded ones (which is the topic of the paper by Mendler, Panangaden and Constable) and which cannot be represented in Martin-Löf's type theory'. From what I understand of the remark, I would say that he/she would want to see a type-theoretic framework where wellfoundedness and predicativeness could be relaxed (perhaps this was indeed the objective of the paper referred to), which is a quite different view from both Prof. Martin-Löf (at least in the standard type theory, anyway) and myself.

From the rules just given for the 'Fix' type forming operator, we can see that this operator works as a kind of 'inductive typed higher-order λ operator':

a) Inductive component:

(i) it contains a 'base' case: $\varnothing$.

(ii) a kind of 'successor' operator:

$$\frac{\mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}{\nabla + \mathbf{B}(\mathbf{A}/\mathbf{X}) \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

(iii) and a 'predecessor' *reduction*:

$$\frac{\mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}{\mathtt{PREDTYPE}(\nabla + \mathbf{B}(\mathbf{A}/\mathbf{X})) = \mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

b) Lambda component:

(i) it contains a kind of *abstraction* rule:

$$\frac{\begin{array}{c}[z \in \mathbf{X}] \\ g(z) \in \nabla + \mathbf{B}(\mathbf{X})\end{array}}{\mathsf{gen}(z.g(z)) \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

(ii) and an *application* rule:

$$\frac{a \in \mathbf{A} \quad \mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X}) \quad \begin{array}{c}[z \in \mathbf{X}] \\ g(z) \in \nabla + \mathbf{B}(\mathbf{X})\end{array}}{\mathtt{UNWIND}(\mathsf{gen}(z.g(z)), a) = g(a/z) \in \nabla + \mathbf{B}(\mathbf{A}/\mathbf{X})}$$

c) Higher-order component:

(i) it contains rules to deal with objects at the level of types:

$$\frac{\mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}{\nabla + \mathbf{B}(\mathbf{A}/\mathbf{X}) \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

$$\frac{\mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}{\mathtt{PREDTYPE}(\nabla + \mathbf{B}(\mathbf{A}/\mathbf{X})) = \mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

It should not be too difficult to see that the rules for 'Fix' constitute kinds of higher-order **Π** rules, and in fact this does not come as a big surprise if we think of 'Fix' as a typed version of a *lambda structure*. Untyped *lambda structures* have been used by Prof. P. Aczel to offer a foundational basis for *ITT*'s interpretations of λ-Calculus (Aczel 1980). Another untyped formulation of a general fixpoint operator within *Intuitionistic Type Theory* similar to 'Fix' has been used by the PRL group to provide means of defining types recursively in Nuprl (Constable & Mendler 1985).

Now, the fact that operators like 'R' for Natural Numbers (I shall be using the name 'NATREC' for the primitive recursive operator for Natural Numbers) are shown to be *derived* operators (similarly for 'LISTREC' for lists, 'STACKREC' for stacks, 'TREEREC' for trees, etc.), and not primitive operators as is the case for *ITT*, would seem to support my point of view. Furthermore, in order to make the operators work properly when defining computations over the type inductively defined, the notion of '*reduction*' appears fundamental, as can be seen easily from the example of Fix-*reduction* rules which show how the 'generator' 'gen' works on the level of the elements, and how the 'successor' '$\nabla + \mathbf{B}(\mathbf{A}/\mathbf{X})$' operates on the level of types.

When proving the derivability of such recursive operators, the point is not just to show a well-known result that primitive recursion can be defined in terms of general recursion plus conditionals, but to show how those primitive recursive operators like 'NATREC' can be obtained in *ITT* from the definition of the type given by its *formation*, *introduction* and *reduction* rules, plus the general fixpoint operator 'Fix', in a way similar to that of Prof. Aczel's derivability of an 'R' operator from a *Peano system* plus a *lambda system* (**Lemma 5.11**, p. 49 of Aczel 1980). For each case (Natural Numbers, Lists, etc.) I first prove the existence of the inductive type by showing how it can be obtained as a type formed from the already existing types with the same 'semantics of use'. I then prove the existence of the corresponding inductive/recursive operator.

6.6. Natural Numbers

As the most common example of an inductive type, the type of natural numbers is given its meaning in *Intuitionistic Type Theory* by the corresponding *introduction* rules which show the *canonical* elements of the type N. This reflects how much difference it makes when the semantical notion is fundamentally distinct. Prof. Martin-Löf's idea of a set (or type) does not seem to offer itself as a true departure from the classical idea that a set is to be defined by what it contains, instead of by what can be done with it. The semantics of *Intuitionistic Type Theory* says that a type is given meaning by the rules of how to build a canonical element of it, which seems to represent their meaning because they are exactly what the types *contain*. From my own understanding of Wittgenstein's 'meaning-is-use' paradigm, I present my *'Meaning-As-USE' Type Theory* as a candidate for the realisation of a framework where types (propositions) are defined by what can be done with their elements (proofs) and not merely by what they contain.

As we have seen from the introductory chapters and sections, my 'semantics of use' prescribes the definition of a type/proposition to be given by at least *grammatical* rules (rules of *introduction*), and *consequence* rules (*reduction* rules, which indicate how to *use* the element of the type). Therefore, an inductive type such as the type of Natural Numbers would not have to be taken as an exception from such a general pattern, given that we have a general schema for the inductively defined types such as the type-of-types forming operator 'Fix'. This way we do not have to introduce the type of Natural Numbers as a primitive type, as is done in Prof. Martin-Löf's *Intuitionistic Type Theory*. Instead, we have this general higher-order inductive type forming operator whose particular instantiations can 'implement' inductive types such as the type of Natural Numbers, Lists, Stacks, Binary Trees, etc., each in their turn being defined by the rules of my 'semantics of use'.

DEFINITION 6.6.1. Let $\mathcal{N}$ be the type inductively defined as:

$\mathcal{N}$-formation

$$\mathcal{N}\ type \qquad \mathcal{N} = \mathcal{N}$$

$\mathcal{N}$-introduction

$$0 \in \mathcal{N} \qquad 0 = 0 \in \mathcal{N}$$

$$\frac{a \in \mathcal{N}}{\mathtt{succ}(a) \in \mathcal{N}} \qquad \frac{a = b \in \mathcal{N}}{\mathtt{succ}(a) = \mathtt{succ}(b) \in \mathcal{N}}$$

$\mathcal{N}$-reduction

$$\frac{a \in \mathcal{N}}{\mathtt{PRED}(\mathtt{succ}(a)) = a \in \mathcal{N}}$$

□

LEMMA 6.6.2. There is a type $\mathcal{N}$.

PROOF. Put

$$N = \{0\}$$
$$0 \equiv \mathtt{inl}(0)$$
$$\mathtt{succ}(a) \equiv \mathtt{inr}(a)$$
$$e(\mathtt{PRED}(a)) \equiv \mathtt{WHEN}(a, d, e)$$

and take Fix $\mathbf{X}.N + \mathbf{X}$ as the generator of $\mathcal{N}$.

It is easy to check that

$$e(\mathtt{PRED}(\mathtt{succ}(a))) = e(a)$$

does verify, given that

$$e(\mathtt{PRED}(\mathtt{succ}(a))) \equiv \mathtt{WHEN}(\mathtt{inr}(a), d, e)$$

and, according to one of the rules of $+$-*reduction*,

$$\mathtt{WHEN}(\mathtt{inr}(a), d, e) = e(a)$$

Now, regarding the *inductive* structure of $\mathcal{N}$ as well as our (alleged) *realisation* of the (potentially) infinite nature of $\mathcal{N}$ via Fix $\mathbf{X}.N + \mathbf{X}$, we have to prove that such a *realisation* is faithful to our 'definition by use' of $\mathcal{N}$.

According to the first rule of the second pair of Fix-*introduction* rules, the generation of Fix $\mathbf{X}.N + \mathbf{X}$ would give us the following iteration:

$$\begin{aligned}
&\varnothing \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X}\\
&N + \varnothing \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X}\\
&N + (N + \varnothing) \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X}\\
&N + (N + (N + \varnothing)) \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X}\\
&N + (N + (N + (N + \varnothing))) \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X}\\
&N + (N + (N + (N + (N + \varnothing)))) \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X}\\
&\vdots
\end{aligned}$$

(recall that '$\varnothing$' is always a member of 'Fix $\mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})$' whatever '$\nabla$' and '$\mathbf{B}(\mathbf{X})$' may be.)

Concerning the elements of the types being generated, if, just for notational convenience, we use '$\perp$' as 'undefined' or the object representing the absence of elements in $\varnothing$, we see the following pattern:

We know from the 'base' case rule (first rule of Fix-*introduction*) that $\varnothing \in$ Fix $\mathbf{X}.N + \mathbf{X}$.

Now, starting from '$\varnothing$', we have:

At the level of the types:

$$\frac{\varnothing \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X}}{N + \mathbf{X}(\varnothing/\mathbf{X}) \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X}}$$

which means that $N + \varnothing \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X}$.

At the level of the elements:

$$\frac{\perp \in \varnothing \quad \varnothing \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X} \quad \begin{array}{c}[z \in \mathbf{X}] \\ \mathtt{inr}(z) \in N + \mathbf{X}\end{array}}{\mathtt{UNWIND}(\mathtt{gen}(z.\mathtt{inr}(z)), \perp) = \mathtt{inr}(z)(\perp/z) \in N + \mathbf{X}(\varnothing/\mathbf{X})}$$

so, $\mathtt{inr}(\perp) \in N + \varnothing$.

$$\frac{\perp \in \varnothing \quad \varnothing \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X} \quad \begin{array}{c}[z \in \mathbf{X}] \\ \mathtt{inl}(0) \in N + \mathbf{X}\end{array}}{\mathtt{UNWIND}(\mathtt{gen}(z.\mathtt{inl}(0)), \perp) = \mathtt{inl}(0)(\perp/z) \in N + \mathbf{X}(\varnothing/\mathbf{X})}$$

so, $\mathtt{inl}(0) \in N + \varnothing$.

and we have generated '$N + \varnothing$' as

$$\{\mathtt{inl}(0), \mathtt{inr}(\perp)\}$$

One more iteration, starting from the previously generated type, and we obtain:

At the level of the types:

$$\frac{N + \varnothing \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X}}{N + \mathbf{X}(N + \varnothing/\mathbf{X}) \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X}}$$

which means that $N + (N + \varnothing) \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X}$.

At the level of the elements:

$$\frac{\begin{array}{ccc}\qquad & \qquad & [z \in \mathbf{X}] \\ \mathtt{inr}(\bot) \in N + \varnothing & N + \varnothing \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X} & \mathtt{inr}(z) \in N + \mathbf{X}\end{array}}{\mathtt{UNWIND}(\mathtt{gen}(z.\mathtt{inr}(z)), \mathtt{inr}(\bot)) = \mathtt{inr}(z)(\mathtt{inr}(\bot)/z) \in N + (N + \varnothing/\mathbf{X})}$$

so, $\mathtt{inr}(\mathtt{inr}(\bot)) \in N + (N + \varnothing)$.

$$\frac{\begin{array}{ccc}\qquad & \qquad & [z \in \mathbf{X}] \\ \mathtt{inr}(\bot) \in N + \varnothing & N + \varnothing \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X} & \mathtt{inl}(0) \in N + \mathbf{X}\end{array}}{\mathtt{UNWIND}(\mathtt{gen}(z.\mathtt{inl}(0)), \mathtt{inr}(\bot)) = \mathtt{inl}(0)(\mathtt{inr}(\bot)/z) \in N + (N + \varnothing/\mathbf{X})}$$

so, $\mathtt{inl}(0) \in N + (N + \varnothing)$.

$$\frac{\begin{array}{ccc}\qquad & \qquad & [z \in \mathbf{X}] \\ \mathtt{inl}(0) \in N + \varnothing & N + \varnothing \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X} & \mathtt{inr}(z) \in N + \mathbf{X}\end{array}}{\mathtt{UNWIND}(\mathtt{gen}(z.\mathtt{inr}(z)), \mathtt{inl}(0)) = \mathtt{inr}(z)(\mathtt{inl}(0)/z) \in N + (N + \varnothing/\mathbf{X})}$$

so, $\mathtt{inr}(\mathtt{inl}(0)) \in N + (N + \varnothing)$.

$$\frac{\begin{array}{ccc}\qquad & \qquad & [z \in \mathbf{X}] \\ \mathtt{inl}(0) \in N + \varnothing & N + \varnothing \in \mathsf{Fix}\ \mathbf{X}.N + \mathbf{X} & \mathtt{inl}(0) \in N + \mathbf{X}\end{array}}{\mathtt{UNWIND}(\mathtt{gen}(z.\mathtt{inl}(0)), \mathtt{inl}(0)) = \mathtt{inl}(0)(\mathtt{inl}(0)/z) \in N + (N + \varnothing/\mathbf{X})}$$

so, $\mathtt{inl}(0) \in N + (N + \varnothing)$.

and we have generated '$N + (N + \varnothing)$' as

$$\{\mathtt{inl}(0), \mathtt{inr}(\mathtt{inl}(0)), \mathtt{inr}(\mathtt{inr}(\bot))\}$$

One more iteration, this time starting from '$N + (N + \varnothing)$', and we would generate '$N + (N + (N + \varnothing))$'.

$$\vdots$$

The iterative process goes on generating a new type at every stage.

The 'potentially' infinite content of Fix $\mathbf{X}.N + \mathbf{X}$ can always be 'expanded' a step further. So, the above iteration is generating the following schema of types:

$$\{\boxed{\perp}\}$$
$$\{\mathtt{inl}(0), \boxed{\mathtt{inr}(\perp)}\}$$
$$\{\mathtt{inl}(0), \mathtt{inr}(\mathtt{inl}(0)), \boxed{\mathtt{inr}(\mathtt{inr}(\perp))}\}$$
$$\{\mathtt{inl}(0), \mathtt{inr}(\mathtt{inl}(0)), \mathtt{inr}(\mathtt{inr}(\mathtt{inl}(0))), \boxed{\mathtt{inr}(\mathtt{inr}(\mathtt{inr}(\perp)))}\}$$
$$\vdots$$

where the elements framed in a box, like e.g. '$\boxed{\mathtt{inr}(\mathtt{inr}(\perp))}$', were introduced for mere convenience of notation, and would be crossed out. The application of the rules turn out to generate an increasing sequence of finite types containing: *no* element; *one* element; *two* elements; etc..

The *implementation* of the inductive type '$\mathcal{N}$' would then be given via a certain 'loose' notion of *supremum* type. The type we are looking for would turn out to be the *supremum* of such an increasing chain of finite types contained in Fix $\mathbf{X}.N + \mathbf{X}$.

So, $\mathcal{N} \equiv \mathbf{SUP}(\mathsf{Fix}\ \mathbf{X}.N + \mathbf{X})$ ■

(Note that '**SUP**' represents an 'open' ('loose') concept, similar to a notion such as 'lazy evaluation', which reflects the 'potential' nature of the infinity.)

Having proved the existence of the inductive type $\mathcal{N}$, we now want to prove the existence of an operator for proofs by induction/recursion on the type. The operator will then serve as a definitional instrument to define computations over the inductive type $\mathcal{N}$.

LEMMA 6.6.3. There is an $\mathcal{N}$-recursive operator '`NATREC`' such that we have

$\mathcal{N}$-Fix-*equality*

$$\frac{\begin{array}{cc} & [x \in \mathcal{N}, y \in \mathrm{C}(x)] \\ d \in \mathrm{C}(0) & e(x,y) \in \mathrm{C}(\mathtt{succ}(x)) \end{array}}{\mathtt{NATREC}(0, d, (x,y)e(x,y)) = d \in \mathrm{C}(0)}$$

$$\frac{\begin{array}{ccc} & & [x \in \mathcal{N}, y \in \mathrm{C}(x)] \\ a \in \mathcal{N} & d \in \mathrm{C}(0) & e(x,y) \in \mathrm{C}(\mathtt{succ}(x)) \end{array}}{\mathtt{NATREC}(\mathtt{succ}(a), d, (x,y)e(x,y)) = e(a, \mathtt{NATREC}(a, d, (x,y)e(x,y))) \in \mathrm{C}(\mathtt{succ}(a))}$$

and the rule of *$\mathcal{N}$-induction/recursion*

$\mathcal{N}$-induction/recursion

$$\frac{\begin{array}{ccc} & & [x \in \mathcal{N}, y \in \mathrm{C}(x)] \\ c \in \mathcal{N} & d \in \mathrm{C}(0) & e(x,y) \in \mathrm{C}(\mathtt{succ}(x)) \end{array}}{\mathtt{NATREC}(c, d, (x,y)e(x,y)) \in \mathrm{C}(c)}$$

will be a derived rule.

PROOF. First identify

Base Case: 0.

Constructor: succ.

Destructor: PRED.

Then let

$$H(d, e, 0) \equiv \mathtt{WHEN}(\mathtt{inl}(0), d, e)$$

$$H(d, e, \mathtt{succ}(a)) \equiv \mathtt{WHEN}(\mathtt{inr}(\mathtt{succ}(a)), d, e(\mathtt{PRED}(\mathtt{succ}(a)), \mathtt{WHEN}(\mathtt{PRED}(\mathtt{succ}(a)), d, e)))$$

Put

$$\mathtt{NATREC}(c, d, e) \equiv H(d, e, c)$$

Putting

$$0 \equiv \mathtt{inl}(0)$$
$$\mathtt{succ}(a) \equiv \mathtt{inr}(a)$$
$$e(\mathtt{PRED}(a)) \equiv \mathtt{WHEN}(a, d, e)$$

one can find an iterative formulation of 'NATREC' via the 'Fix' operator. ■

6.7. Parameterised Types

In a type expression '**B**(**A**)', '**B**' is said to be parameterised by **A**. Now, to have 'parameters' in the specification of a certain type in the framework of our *'Meaning-As-USE' Type Theory* would not seem to bring in extra problems such as those treated by the more *denotations*-oriented approaches, such as, for instance, error-propagation. Instantiating a type parameter such as '**A**' with another type presented within the type-theoretic framework described here, would not seem to inherit the eventual problems in dealing with odd terms of **A** in the semantical explanations of the definition/presentation of **B**(**A**). As we can see from a typical example of a parameterised specification such as that of binary trees of elements of type **A**, the rules of *formation*:

BinTree-*formation*

$$\frac{\mathbf{A}\ \mathit{type}}{\mathbf{BinTree(A)}\ \mathit{type}} \qquad \frac{\mathbf{A} = \mathbf{B}}{\mathbf{BinTree(A)} = \mathbf{BinTree(B)}}$$

certify that the parameter **A** is indeed a type. Furthermore, to bring in the semantics of the 'target type' to match the 'formal parameter type' **A** would be as simple as bringing in its rules of *use* for the target type, no interference in the semantics (*normalisation*) of the uninstantiated parameterised type taking place. (Recall that we do not have 'error equations' such as, e.g., '$\mathtt{TOP}(\mathtt{emptystack}) = 0 \in \mathbf{N}$', where the

instantiation the 'formal parameter' '**A**' of a parameterised definition of '**Stack(A)**' would have to take into account whether the target type would have '0' in it.)

And, again, similarly to Natural Numbers, Lists, Stacks, etc., given the rules of *introduction* and *reduction* as:

BinTree(A)-*introduction*

$$\texttt{niltree} \in \mathbf{BinTree(A)} \qquad \texttt{niltree} = \texttt{niltree} \in \mathbf{BinTree(A)}$$

$$\frac{a \in \mathbf{BinTree(A)} \quad b \in \mathbf{A} \quad c \in \mathbf{BinTree(A)}}{\texttt{node}(a, b, c) \in \mathbf{BinTree(A)}}$$

$$\frac{a_1 = a_2 \in \mathbf{BinTree(A)} \quad b_1 = b_2 \in \mathbf{A} \quad c_1 = c_2 \in \mathbf{BinTree(A)}}{\texttt{node}(a_1, b_1, c_1) = \texttt{node}(a_2, b_2, c_2) \in \mathbf{BinTree(A)}}$$

BinTree(A)-*reduction*

$$\frac{a \in \mathbf{BinTree(A)} \quad b \in \mathbf{A} \quad c \in \mathbf{BinTree(A)}}{\texttt{LABEL}(\texttt{node}(a, b, c)) = b \in \mathbf{A}}$$

$$\frac{a \in \mathbf{BinTree(A)} \quad b \in \mathbf{A} \quad c \in \mathbf{BinTree(A)}}{\texttt{LEFT}(\texttt{node}(a, b, c)) = a \in \mathbf{BinTree(A)}}$$

$$\frac{a \in \mathbf{BinTree(A)} \quad b \in \mathbf{A} \quad c \in \mathbf{BinTree(A)}}{\texttt{RIGHT}(\texttt{node}(a, b, c)) = c \in \mathbf{BinTree(A)}}$$

there would be no way of reaching *judgements* like '$\texttt{LABEL}(\texttt{niltree}) \in \mathbf{A}$', or '$\texttt{LEFT}(\texttt{niltree}) \in \mathbf{BinTree(A)}$', or even '$\texttt{RIGHT}(\texttt{niltree}) \in \mathbf{BinTree(A)}$' to be made, and the 'errors' would not be propagated, even if the type parameter '**A**' is to be an instance of **BinTree(A)** itself. (I shall come back to a discussion on parameterisation of data types in the next chapter.)

Now, I shall demonstrate how parameterised inductively defined types such as the usual (LISP-like) **Lists** of elements of **A**, and **Stacks** of elements of **A**, can be

defined and proved to exist in terms of the primitive type forming operators of my *'Meaning-As-USE' Type Theory.*

DEFINITION 6.7.1. Let **List(A)** be the type of lists of elements from the type **A**, defined inductively as:

List(A)-*formation*

$$\frac{\mathbf{A}\ type}{\mathbf{List(A)}\ type} \qquad \frac{\mathbf{A} = \mathbf{B}}{\mathbf{List(A)} = \mathbf{List(B)}}$$

List(A)-*introduction*

$$\mathtt{nillist} \in \mathbf{List(A)} \qquad \mathtt{nillist} = \mathtt{nillist} \in \mathbf{List(A)}$$

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{List(A)}}{\mathtt{cons}(a,b) \in \mathbf{List(A)}} \qquad \frac{a = c \in \mathbf{A} \quad b = d \in \mathbf{List(A)}}{\mathtt{cons}(a,b) = \mathtt{cons}(c,d) \in \mathbf{List(A)}}$$

List(A)-*reduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{List(A)}}{\mathtt{HEAD}(\mathtt{cons}(a,b)) = a \in \mathbf{A}} \qquad \frac{a \in \mathbf{A} \quad b \in \mathbf{List(A)}}{\mathtt{TAIL}(\mathtt{cons}(a,b)) = b \in \mathbf{List(A)}}$$

□

LEMMA 6.7.2. There is a type **List(A)**.

PROOF. Put

$$\begin{aligned}
&\mathbf{List}_1 = \{\mathtt{nil}\}\\
&\mathtt{nillist} \equiv \mathtt{inl}(\mathtt{nil})\\
&\mathtt{cons}(a,b) \equiv \mathtt{inr}(\mathtt{pair}(a,b))\\
&\mathtt{HEAD}(c) \equiv \mathtt{WHEN}(c,d,\mathtt{FST})\\
&\mathtt{TAIL}(c) \equiv \mathtt{WHEN}(c,d,\mathtt{SND})
\end{aligned}$$

and take Fix $\mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}$ as the generator of $\mathbf{List}(\mathbf{A})$.

Regarding the *correctness* of the *implementation* it is easy to check that:

$$\mathtt{HEAD}(\mathtt{cons}(a, b)) = a$$

given that

$$\begin{aligned} \mathtt{HEAD}(\mathtt{cons}(a,b)) &\equiv \mathtt{WHEN}(\mathtt{inr}(\mathtt{pair}(a,b)), d, \mathtt{FST}) \\ &= \mathtt{FST}(\mathtt{pair}(a,b)) \\ &= a \end{aligned}$$

Similarly, it is straightforward to check that:

$$\mathtt{TAIL}(\mathtt{cons}(a, b)) = b$$

given that

$$\begin{aligned} \mathtt{TAIL}(\mathtt{cons}(a,b)) &\equiv \mathtt{WHEN}(\mathtt{inr}(\mathtt{pair}(a,b)), d, \mathtt{SND}) \\ &= \mathtt{SND}(\mathtt{pair}(a,b)) \\ &= b \end{aligned}$$

Now, I should then provide the proof of *implementation* of the *potentially* infinite structure of our inductively defined data type $\mathbf{List}(\mathbf{A})$, which shall be done via the general fixpoint operator 'Fix'.

According to the first rule of the second pair of Fix-*introduction* rules, the generation of Fix $\mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}$ would give us the following iteration:

$$\varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}$$
$$\mathbf{List}_1 + \mathbf{A} \times \varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}$$
$$\mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times \varnothing) \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}$$
$$\mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times \varnothing)) \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}$$
$$\vdots$$

(recall that '$\varnothing$' is always a member of '$\mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})$' whatever '$\nabla$' and '$\mathbf{B}(\mathbf{X})$' may be.)

Concerning the elements of the types being generated, if, again just for notational convenience, we use '$\perp$' as the 'undefined' or the object representing the absence of elements in $\varnothing$, we see the following pattern:

We know from the 'base' case rule (first rule of Fix-*introduction*) that $\varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}$.

Now, starting from '$\varnothing$', we have:

At the level of the types:

$$\frac{\varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}}{\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}(\varnothing/\mathbf{X}) \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}}$$

which means that $\mathbf{List}_1 + \mathbf{A} \times \varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}$.

At the level of the elements:

$$\frac{\perp \in \varnothing \quad \varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X} \quad \begin{matrix}[a \in \mathbf{A}, z \in \mathbf{X}] \\ \mathtt{inr}(\mathtt{pair}(a,z)) \in \mathbf{List}_1 + \mathbf{A} \times \mathbf{X}\end{matrix}}{\mathtt{UNWIND}(\mathtt{gen}(z.\mathtt{inr}(\mathtt{pair}(a,z)), \perp) = \mathtt{inr}(\mathtt{pair}(a,z))(\perp/z) \in \mathbf{List}_1 + \mathbf{A} \times \mathbf{X}(\varnothing/\mathbf{X})}$$

so, $\mathtt{inr}(\mathtt{pair}(a, \perp)) \in \mathbf{List}_1 + \mathbf{A} \times \varnothing$.

$$\frac{\perp \in \varnothing \quad \varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X} \quad \mathtt{inl}(\mathtt{nil}) \in \mathbf{List}_1 + \mathbf{A} \times \mathbf{X}}{\mathtt{UNWIND}(\mathtt{gen}(z.\mathtt{inl}(\mathtt{nil})), \perp) = \mathtt{inl}(\mathtt{nil})(\perp/z) \in \mathbf{List}_1 + \mathbf{A} \times \mathbf{X}(\varnothing/\mathbf{X})}\ [z \in \mathbf{X}]$$

so, $\mathtt{inl}(\mathtt{nil}) \in \mathbf{List}_1 + \mathbf{A} \times \varnothing$.

and we have generated '$\mathbf{List}_1 + \mathbf{A} \times \varnothing$' as

$$\{\mathtt{inl}(\mathtt{nil}), \mathtt{inr}(\mathtt{pair}(a, \perp))\}$$

One more iteration, starting from the previously generated type, and we obtain:

At the level of the types:

$$\frac{\mathbf{List}_1 + \mathbf{A} \times \varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}}{\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}(\mathbf{List}_1 + \mathbf{A} \times \varnothing/\mathbf{X}) \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}}$$

which means that $\mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times \varnothing) \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}$.

At the level of the elements:

$$\frac{\mathtt{inr}(\mathtt{pair}(a,\perp)) \in \mathbf{List}_1 + \mathbf{A} \times \varnothing \quad \mathbf{List}_1 + \mathbf{A} \times \varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X} \quad \mathtt{inr}(\mathtt{pair}(a,z)) \in \mathbf{List}_1 + \mathbf{A} \times \mathbf{X}}{\mathtt{UNWIND}(\mathtt{gen}(z.\mathtt{inr}(\mathtt{pair}(a,z))), \mathtt{inr}(\mathtt{pair}(a,\perp))) = \mathtt{inr}(\mathtt{pair}(a,z))(\mathtt{inr}(\mathtt{pair}(a,\perp))/z) \in \mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times \varnothing/\mathbf{X})}\ [a \in \mathbf{A}, z \in \mathbf{X}]$$

so, $\mathtt{inr}(\mathtt{pair}(a, \mathtt{inr}(\mathtt{pair}(a, \perp)))) \in \mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times \varnothing)$.

$$\frac{\mathtt{inr}(\mathtt{pair}(a,\perp)) \in \mathbf{List}_1 + \mathbf{A} \times \varnothing \quad \mathbf{List}_1 + \mathbf{A} \times \varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X} \quad \mathtt{inl}(\mathtt{nil}) \in \mathbf{List}_1 + \mathbf{A} \times \mathbf{X}}{\mathtt{UNWIND}(\mathtt{gen}(z.\mathtt{inl}(\mathtt{nil})), \mathtt{inr}(\mathtt{pair}(a,\perp))) = \mathtt{inl}(\mathtt{nil})(\mathtt{inr}(\mathtt{pair}(a,\perp))/z) \in \mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times \varnothing/\mathbf{X})}\ [z \in \mathbf{X}]$$

so, $\mathtt{inl}(\mathtt{nil}) \in \mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times \varnothing)$.

$$\frac{\mathtt{inl}(\mathtt{nil}) \in \mathbf{List}_1 + \mathbf{A} \times \varnothing \quad \mathbf{List}_1 + \mathbf{A} \times \varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X} \quad \mathtt{inr}(\mathtt{pair}(a,z)) \in \mathbf{List}_1 + \mathbf{A} \times \mathbf{X}}{\mathtt{UNWIND}(\mathtt{gen}(z.\mathtt{inr}(\mathtt{pair}(a,z))), \mathtt{inl}(\mathtt{nil})) = \mathtt{inr}(\mathtt{pair}(a,z))(\mathtt{inl}(\mathtt{nil})/z) \in \mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times \varnothing/\mathbf{X})}\ [a \in \mathbf{A}, z \in \mathbf{X}]$$

so, $\mathtt{inr}(\mathtt{pair}(a, \mathtt{inl}(\mathtt{nil}))) \in \mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times \varnothing)$.

$$\frac{\mathtt{inl}(\mathtt{nil}) \in \mathbf{List}_1 + \mathbf{A} \times \varnothing \quad \mathbf{List}_1 + \mathbf{A} \times \varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X} \quad \overset{[z \in \mathbf{X}]}{\mathtt{inl}(\mathtt{nil}) \in \mathbf{List}_1 + \mathbf{A} \times \mathbf{X}}}{\mathtt{UNWIND}(\mathtt{gen}(z.\mathtt{inl}(\mathtt{nil})), \mathtt{inl}(\mathtt{nil})) = \mathtt{inl}(\mathtt{nil})(\mathtt{inl}(\mathtt{nil})/z) \in \mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times \varnothing / \mathbf{X})}$$

so, $\mathtt{inl}(\mathtt{nil}) \in \mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times \varnothing)$.

and we have generated '$\mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times \varnothing)$' as

$$\{\mathtt{inl}(\mathtt{nil}), \mathtt{inr}(\mathtt{pair}(a, \mathtt{inl}(\mathtt{nil}))), \mathtt{inr}(\mathtt{pair}(a, \mathtt{inr}(\mathtt{pair}(a, \bot))))\}$$

One more iteration, this time starting from '$\mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times \varnothing)$', and we would generate '$\mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times (\mathbf{List}_1 + \mathbf{A} \times \varnothing))$'.

$$\vdots$$

The iterative process goes on generating a new type at every stage.

The 'potentially' infinite content of $\mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}$ can always be 'expanded' a step further. So, the above iteration is generating the following chain of types:

$\{\boxed{\bot}\}$

$\{\mathtt{inl}(\mathtt{nil}), \boxed{\mathtt{inr}(\mathtt{pair}(a, \bot))}\}$

$\{\mathtt{inl}(\mathtt{nil}), \mathtt{inr}(\mathtt{pair}(a, \mathtt{inl}(\mathtt{nil}))), \boxed{\mathtt{inr}(\mathtt{pair}(a, \mathtt{inr}(\bot)))}\}$

$\{\mathtt{inl}(\mathtt{nil}), \mathtt{inr}(\mathtt{pair}(a, \mathtt{inl}(\mathtt{nil}))), \mathtt{inr}(\mathtt{pair}(a, \mathtt{inr}(\mathtt{pair}(a, \mathtt{inl}(\mathtt{nil}))))),$
$\boxed{\mathtt{inr}(\mathtt{pair}(a, \mathtt{inr}(\mathtt{pair}(a, \mathtt{inr}(\mathtt{pair}(a, \bot))))))}\}$

$\vdots$

where the elements framed in a box, like e.g. '$\boxed{\mathtt{inr(pair}(a,\bot))}$', were introduced for mere convenience of notation, and would be crossed out. By applying the rules we obtained an increasing sequence of finite types containing the following lists: a list with *no* element; a list with *no* element, and a list with *one* arbitrary element; a list with *no* element, a list with *one* arbitrary element, and a list with *two* arbitrary elements; etc..

The *implementation* of the inductive type '$\mathbf{List(A)}$' would be the *supremum* of the increasing sequence of types contained in $\mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X}$.

So, $\mathbf{List(A)} \equiv \mathbf{SUP}(\mathsf{Fix}\ \mathbf{X}.\mathbf{List}_1 + \mathbf{A} \times \mathbf{X})$ ■

Having proved the existence of the inductive parameterised type '$\mathbf{List(A)}$', I now want to prove the existence of an operator for proofs by induction/recursion on the type. The operator will then serve as a definitional instrument to define computations over the inductive type '$\mathbf{List(A)}$'.

LEMMA 6.7.3. There is a **List**-recursive operator 'LISTREC' such that we have

List-Fix-*equality*

$$\frac{d \in \mathbf{C}(\mathtt{nillist}) \qquad [x \in \mathbf{A}, y \in \mathbf{List(A)}, z \in \mathbf{C}(y)]\ e(x,y,z) \in \mathbf{C}(\mathtt{cons}(x,y))}{\mathtt{LISTREC}(\mathtt{nillist}, d, (x,y,z)e(x,y,z)) = d \in \mathbf{C}(\mathtt{nillist})}$$

$$\frac{a \in \mathbf{A} \qquad b \in \mathbf{List(A)} \qquad d \in \mathbf{C}(\mathtt{nillist}) \qquad [x \in \mathbf{A}, y \in \mathbf{List(A)}, z \in \mathbf{C}(y)]\ e(x,y,z) \in \mathbf{C}(\mathtt{cons}(x,y))}{\mathtt{LISTREC}(\mathtt{cons}(a,b),d,(x,y,z)e(x,y,z)) = e(a,b,\mathtt{LISTREC}(a,b,d,(x,y,z)e(x,y,z))) \in \mathbf{C}(\mathtt{cons}(a,b))}$$

and the rule of $\mathbf{List(A)}$-*induction/recursion*

$\mathbf{List(A)}$-*induction/recursion*

$$\frac{c \in \mathbf{List(A)} \qquad d \in \mathbf{C}(\mathtt{nillist}) \qquad [x \in \mathbf{A}, y \in \mathbf{List(A)}, z \in \mathbf{C}(y)]\ e(x,y,z) \in \mathbf{C}(\mathtt{cons}(x,y))}{\mathtt{LISTREC}(c, d, (x,y,z)e(x,y,z)) \in \mathbf{C}(c)}$$

will be a derived rule.

PROOF. First identify

Base Case: `nillist`.

Constructor: `cons`.

Destructors: `HEAD`, `TAIL`.

Then let

$$H(d, e, \texttt{nillist}) \equiv \texttt{WHEN}(\texttt{inl}(\texttt{nillist}), d, e)$$

$$H(d,e,\texttt{cons}(a,b)) \equiv \texttt{WHEN}(\texttt{inr}(\texttt{cons}(a,b)),d,e(\texttt{HEAD}(\texttt{cons}(a,b)),\texttt{TAIL}(\texttt{cons}(a,b)),\texttt{WHEN}(\texttt{TAIL}(\texttt{cons}(a,b)),d,e)))$$

Put

$$\texttt{LISTREC}(c, d, e) \equiv H(d, e, c)$$

Putting

$$\begin{aligned}
&\texttt{nillist} \equiv \texttt{inl}(\texttt{nil}) \\
&\texttt{cons}(a, b) \equiv \texttt{inr}(\texttt{pair}(a, b)) \\
&\texttt{HEAD}(c) \equiv \texttt{WHEN}(c, d, \texttt{FST}) \\
&\texttt{TAIL}(c) \equiv \texttt{WHEN}(c, d, \texttt{SND})
\end{aligned}$$

similarly to the 'NATREC' case, one can find an iterative formulation of 'LISTREC' via 'Fix'. ■

DEFINITION 6.7.4. Let **Stack(A)** be the type of stacks of elements from **A**, inductively defined as:

Stack-*formation*

$$\frac{\mathbf{A}\ \mathit{type}}{\mathbf{Stack(A)}\ \mathit{type}} \qquad \frac{\mathbf{A} = \mathbf{B}}{\mathbf{Stack(A)} = \mathbf{Stack(B)}}$$

Stack-*introduction*

$$\mathtt{emptystack} \in \mathbf{Stack(A)} \qquad \mathtt{emptystack} = \mathtt{emptystack} \in \mathbf{Stack(A)}$$

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{Stack(A)}}{\mathtt{push}(a, b) \in \mathbf{Stack(A)}} \qquad \frac{a = c \in \mathbf{A} \quad b = d \in \mathbf{Stack(A)}}{\mathtt{push}(a, b) = \mathtt{push}(c, d) \in \mathbf{Stack(A)}}$$

Stack-*reduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{Stack(A)}}{\mathtt{TOP}(\mathtt{push}(a, b)) = a \in \mathbf{A}} \qquad \frac{a \in \mathbf{A} \quad b \in \mathbf{Stack(A)}}{\mathtt{POP}(\mathtt{push}(a, b)) = b \in \mathbf{Stack(A)}}$$

□

LEMMA 6.7.5. There is a type **Stack(A)**.

PROOF. Put

$$\begin{aligned}
&\mathbf{Stack}_1 = \{\mathtt{empty}\} \\
&\mathtt{emptystack} \equiv \mathtt{inl}(\mathtt{empty}) \\
&\mathtt{push}(a, b) \equiv \mathtt{inr}(\mathtt{pair}(a, b)) \\
&\mathtt{TOP}(c) \equiv \mathtt{WHEN}(c, d, \mathtt{FST}) \\
&\mathtt{POP}(c) \equiv \mathtt{WHEN}(c, d, \mathtt{SND})
\end{aligned}$$

and take Fix $\mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}$ as the generator of **Stack(A)**.

Regarding the *correctness* of the *implementation* it is easy to see that:

$$\mathtt{TOP}(\mathtt{push}(a, b)) = a$$

given that

$$\begin{aligned}\texttt{TOP}(\texttt{push}(a,b)) &\equiv \texttt{WHEN}(\texttt{inr}(\texttt{pair}(a,b)),d,\texttt{FST})\\ &= \texttt{FST}(\texttt{pair}(a,b))\\ &= a\end{aligned}$$

Similarly, to check that:

$$\texttt{POP}(\texttt{push}(a,b)) = b$$

we verify

$$\begin{aligned}\texttt{POP}(\texttt{push}(a,b)) &\equiv \texttt{WHEN}(\texttt{inr}(\texttt{pair}(a,b)),d,\texttt{SND})\\ &= \texttt{SND}(\texttt{pair}(a,b))\\ &= b\end{aligned}$$

Now, I should then provide the proof of *implementation* of the *potentially* infinite structure of our inductively defined data type **Stack(A)**, which shall be done via the general fixpoint operator 'Fix'.

According to the first rule of the second pair of Fix-*introduction* rules, the generation of Fix $\mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}$ would give us the following iteration:

$\varnothing \in \text{Fix } \mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}$

$\mathbf{Stack}_1 + \mathbf{A} \times \varnothing \in \text{Fix } \mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}$

$\mathbf{Stack}_1 + \mathbf{A} \times (\mathbf{Stack}_1 + \mathbf{A} \times \varnothing) \in \text{Fix } \mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}$

$\mathbf{Stack}_1 + \mathbf{A} \times (\mathbf{Stack}_1 + \mathbf{A} \times (\mathbf{Stack}_1 + \mathbf{A} \times \varnothing)) \in \text{Fix } \mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}$

$\vdots$

(recall that '$\varnothing$' is always a member of 'Fix $\mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})$' whatever '$\nabla$' and '$\mathbf{B}(\mathbf{X})$' may be.)

Concerning the elements of the types being generated, if, just for notational convenience, we use '$\perp$' as the 'undefined' or the object representing the absence of elements in $\varnothing$, we see the following pattern:

We know from the 'base' case rule (first rule of Fix-*introduction*) that $\varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}$.

Now, starting from '$\varnothing$', we have:

At the level of the types:

$$\frac{\varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}}{\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}(\varnothing/\mathbf{X}) \in \mathsf{Fix}\ \mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}}$$

which means that $\mathbf{Stack}_1 + \mathbf{A} \times \varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}$.

At the level of the elements:

$$\frac{\perp \in \varnothing \quad \varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X} \quad \begin{array}{c}[a \in \mathbf{A}, z \in \mathbf{X}] \\ \mathtt{inr}(\mathtt{pair}(a,z)) \in \mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}\end{array}}{\mathtt{UNWIND}(\mathtt{gen}(z.\mathtt{inr}(\mathtt{pair}(a,z)), \perp) = \mathtt{inr}(\mathtt{pair}(a,z))(\perp/z) \in \mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}(\varnothing/\mathbf{X})}$$

so, $\mathtt{inr}(\mathtt{pair}(a, \perp)) \in \mathbf{Stack}_1 + \mathbf{A} \times \varnothing$.

$$\frac{\perp \in \varnothing \quad \varnothing \in \mathsf{Fix}\ \mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X} \quad \begin{array}{c}[z \in \mathbf{X}] \\ \mathtt{inl}(\mathtt{empty}) \in \mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}\end{array}}{\mathtt{UNWIND}(\mathtt{gen}(z.\mathtt{inl}(\mathtt{empty})), \perp) = \mathtt{inl}(\mathtt{empty})(\perp/z) \in \mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}(\varnothing/\mathbf{X})}$$

so, $\mathtt{inl}(\mathtt{empty}) \in \mathbf{Stack}_1 + \mathbf{A} \times \varnothing$.

and we have generated '$\mathbf{Stack}_1 + \mathbf{A} \times \varnothing$' as

$$\{\mathtt{inl}(\mathtt{empty}), \mathtt{inr}(\mathtt{pair}(a, \perp))\}$$

One more iteration, starting from the previously generated type, and we obtain:

At the level of the types:

$$\frac{\mathbf{Stack_1 + A \times \varnothing \in Fix\ X.Stack_1 + A \times X}}{\mathbf{Stack_1 + A \times X(Stack_1 + A \times \varnothing/X) \in Fix\ X.Stack_1 + A \times X}}$$

which means that $\mathbf{Stack_1 + A \times (Stack_1 + A \times \varnothing) \in Fix\ X.Stack_1 + A \times X}$.

At the level of the elements:

$$\frac{\mathtt{inr(pair}(a,\bot))\in\mathbf{Stack_1+A\times\varnothing} \quad \mathbf{Stack_1+A\times\varnothing\in Fix\ X.Stack_1+A\times X} \quad \overset{[a\in\mathbf{A},z\in\mathbf{X}]}{\mathtt{inr(pair}(a,z))\in\mathbf{Stack_1+A\times X}}}{\mathtt{UNWIND(gen}(z.\mathtt{inr(pair}(a,z))),\mathtt{inr(pair}(a,\bot)))=\mathtt{inr(pair}(a,z))(\mathtt{inr(pair}(a,\bot))/z)\in\mathbf{Stack_1+A\times(Stack_1+A\times\varnothing/X)}}$$

so, $\mathtt{inr(pair}(a, \mathtt{inr(pair}(a, \bot)))) \in \mathbf{Stack_1 + A \times (Stack_1 + A \times \varnothing)}$.

$$\frac{\mathtt{inr(pair}(a,\bot))\in\mathbf{Stack_1+A\times\varnothing} \quad \mathbf{Stack_1+A\times\varnothing\in Fix\ X.Stack_1+A\times X} \quad \overset{[z\in\mathbf{X}]}{\mathtt{inl(empty)}\in\mathbf{Stack_1+A\times X}}}{\mathtt{UNWIND(gen}(z.\mathtt{inl(empty)}),\mathtt{inr(pair}(a,\bot)))=\mathtt{inl(empty)}(\mathtt{inr(pair}(a,\bot))/z)\in\mathbf{Stack_1+A\times(Stack_1+A\times\varnothing/X)}}$$

so, $\mathtt{inl(empty)} \in \mathbf{Stack_1 + A \times (Stack_1 + A \times \varnothing)}$.

$$\frac{\mathtt{inl(empty)}\in\mathbf{Stack_1+A\times\varnothing} \quad \mathbf{Stack_1+A\times\varnothing\in Fix\ X.Stack_1+A\times X} \quad \overset{[a\in\mathbf{A},z\in\mathbf{X}]}{\mathtt{inr(pair}(a,z))\in\mathbf{Stack_1+A\times X}}}{\mathtt{UNWIND(gen}(z.\mathtt{inr(pair}(a,z))),\mathtt{inl(empty)})=\mathtt{inr(pair}(a,z))(\mathtt{inl(empty)}/z)\in\mathbf{Stack_1+A\times(Stack_1+A\times\varnothing/X)}}$$

so, $\mathtt{inr(pair}(a, \mathtt{inl(empty)})) \in \mathbf{Stack_1 + A \times (Stack_1 + A \times \varnothing)}$.

$$\frac{\mathtt{inl(empty)}\in\mathbf{Stack_1+A\times\varnothing} \quad \mathbf{Stack_1+A\times\varnothing\in Fix\ X.Stack_1+A\times X} \quad \overset{[z\in\mathbf{X}]}{\mathtt{inl(empty)}\in\mathbf{Stack_1+A\times X}}}{\mathtt{UNWIND(gen}(z.\mathtt{inl(empty)}),\mathtt{inl(empty)})=\mathtt{inl(empty)}(\mathtt{inl(empty)}/z)\in\mathbf{Stack_1+A\times(Stack_1+A\times\varnothing/X)}}$$

so, $\mathtt{inl(empty)} \in \mathbf{Stack_1 + A \times (Stack_1 + A \times \varnothing)}$.

and we have generated '$\mathbf{Stack_1 + A \times (Stack_1 + A \times \varnothing)}$' as

$$\{\mathtt{inl(empty)}, \mathtt{inr(pair}(a, \mathtt{inl(empty)})), \mathtt{inr(pair}(a, \mathtt{inr(pair}(a, \bot)))))\}$$

One more iteration, this time starting from '$\mathbf{Stack}_1 + \mathbf{A} \times (\mathbf{Stack}_1 + \mathbf{A} \times \varnothing)$', and we would generate '$\mathbf{Stack}_1 + \mathbf{A} \times (\mathbf{Stack}_1 + \mathbf{A} \times (\mathbf{Stack}_1 + \mathbf{A} \times \varnothing))$'.

$$\vdots$$

The iterative process goes on generating a new type at every stage.

The 'potentially' infinite content of Fix $\mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}$ can always be 'expanded' a step further. So, the above iteration is generating the following chain of types:

$$\{\boxed{\bot}\}$$

$$\{\mathtt{inl(empty)}, \boxed{\mathtt{inr(pair}(a, \bot))}\}$$

$$\{\mathtt{inl(empty)}, \mathtt{inr(pair}(a, \mathtt{inl(empty)})), \boxed{\mathtt{inr(pair}(a, \mathtt{inr}(\bot)))}\}$$

$$\{\mathtt{inl(empty)}, \mathtt{inr(pair}(a, \mathtt{inl(empty)})), \mathtt{inr(pair}(a, \mathtt{inr(pair}(a, \mathtt{inl(empty)})))),$$
$$\boxed{\mathtt{inr(pair}(a, \mathtt{inr(pair}(a, \mathtt{inr(pair}(a, \bot))))))}\}$$

$$\vdots$$

where the elements framed in a box, like e.g. '$\boxed{\mathtt{inr(pair}(a, \bot))}$', were introduced for mere convenience of notation, and would be crossed out. The application of the rules generate an increasing sequence of finite types containing: *no* element, then a **stack** with *one* (arbitrary) element, then a **stack** with *one* (arbitrary) element and another **stack** with *two* (arbitrary) elements, ..., and so on and so forth.

The *implementation* of the inductive type '**Stack(A)**' would be given by a certain 'loose' notion of *supremum*. The *implementation* required is then the *supremum* of the increasing sequence of types contained in $\mathsf{Fix}\ \mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X}$.

So, $\mathbf{Stack}(\mathbf{A}) \equiv \mathbf{SUP}(\mathsf{Fix}\ \mathbf{X}.\mathbf{Stack}_1 + \mathbf{A} \times \mathbf{X})$ ■

Having proved the existence of the inductive parameterised type '**Stack(A)**', we now want to prove the existence of an operator for proofs by induction/recursion on the type. The operator will then serve as a definitional instrument to define computations over the inductive type '**Stack(A)**'.

LEMMA 6.7.6. There is a **Stack**-recursive operator 'STACKREC' such that we have

Stack-Fix-*equality*

$$\frac{d \in \mathbf{C}(\mathtt{emptystack}) \qquad \begin{array}{c}[x \in \mathbf{A}, y \in \mathbf{Stack}(\mathbf{A}), z \in \mathbf{C}(y)] \\ e(x,y,z) \in \mathbf{C}(\mathtt{push}(x,y))\end{array}}{\mathtt{STACKREC}(\mathtt{emptystack}, d, (x,y,z)e(x,y,z)) = d \in \mathbf{C}(\mathtt{emptystack})}$$

$$\frac{a \in \mathbf{A} \qquad b \in \mathbf{Stack}(\mathbf{A}) \qquad d \in \mathbf{C}(\mathtt{emptystack}) \qquad \begin{array}{c}[x \in \mathbf{A}, y \in \mathbf{Stack}(\mathbf{A}), z \in \mathbf{C}(y)] \\ e(x,y,z) \in \mathbf{C}(\mathtt{push}(x,y))\end{array}}{\mathtt{STACKREC}(\mathtt{push}(a,b),d,(x,y,z)e(x,y,z)) = e(a,b,\mathtt{STACKREC}(a,b,d,(x,y,z)e(x,y,z))) \in \mathbf{C}(\mathtt{push}(a,b))}$$

and the rule of **Stack(A)**-*induction/recursion*

Stack(A)-*induction/recursion*

$$\frac{c \in \mathbf{Stack}(\mathbf{A}) \qquad d \in \mathbf{C}(\mathtt{emptystack}) \qquad \begin{array}{c}[x \in A, y \in \mathbf{Stack}(\mathbf{A}), z \in \mathbf{C}(y)] \\ e(x,y,z) \in \mathbf{C}(\mathtt{push}(x,y))\end{array}}{\mathtt{STACKREC}(c, d, (x,y,z)e(x,y,z)) \in \mathbf{C}(c)}$$

will be a derived rule.

PROOF. First identify

Base Case: emptystack.

Constructor: push.

Destructors: TOP, POP.

Then let

$$H(d, e, \texttt{emptystack}) \equiv \texttt{WHEN}(\texttt{inl}(\texttt{emptystack}), d, e)$$

$$H(d,e,\texttt{push}(a,b)) \equiv \texttt{WHEN}(\texttt{inr}(\texttt{push}(a,b)),d,e(\texttt{TOP}(\texttt{push}(a,b)),\texttt{POP}(\texttt{push}(a,b)),\texttt{WHEN}(\texttt{POP}(\texttt{push}(a,b)),d,e)))$$

Put

$$\texttt{STACKREC}(c, d, e) \equiv H(d, e, c)$$

Putting

$$\begin{aligned}
&\texttt{emptystack} \equiv \texttt{inl}(\texttt{empty}) \\
&\texttt{push}(a, b) \equiv \texttt{inr}(\texttt{pair}(a, b)) \\
&\texttt{TOP}(c) \equiv \texttt{WHEN}(c, d, \texttt{FST}) \\
&\texttt{POP}(c) \equiv \texttt{WHEN}(c, d, \texttt{SND})
\end{aligned}$$

similarly to the 'NATREC' case, one can find an iterative formulation of 'STACKREC' via 'Fix'. ■

We have just seen examples of 'definitions by use' which represent my understanding of Frege's *analytic* definitions, where my proof of existence corresponds to his *logical analysis* (see chapter 2 for a closer insight into my understanding of Frege's notions of definition and the rôle of rules in relation to meaning). This pattern of 'definitions by use'/proof of existence is used in my formal explanation of the notion of *implementation* of abstract data types, where a new type 'defined by use' is proved to exist by a proof of the existence of a particular combination of already

existing types in a type with the same 'semantics of use' — the effect of the representative(s) of the DESTRUCTOR(S) on the representative(s) of the constructor(s) is shown to be the same as in the original definition/presentation—.

As an example of such a process of implementation I show how it is done in the case of implementing a (potentially) infinite type '**Stack(A)**' in terms of the primitive operations available in the formal system: *pairing* ('Σ' type forming operator) and *fixpoint* ('Fix' type-of-type forming operator) to handle inductive types.

An interesting point which arose from my proof of existence for the types '**List(A)**' and '**Stack(A)**' above is the fact that both can be implemented in the same way via 'Σ' (*pairing/splitting*) and 'Fix' (*induction/potential infinity*). This suggests that, according to my 'semantics of use', **stacks** with 'push', 'POP' and 'TOP', and **lists** with 'cons', 'HEAD' and 'TAIL' are essentially the same data type, just because they present the same 'semantics of use'. I shall be dealing with this particular issue in more detail in the next chapter.

6.8. Final Remarks

By introducing a higher-order concept of 'type of types' which would seem to unify both the notions of well-ordering and cumulative universes of *Intuitionistic Type Theory* into a single higher order type forming operator which I call 'Fix', I hope to have made a (very small) step towards demonstrating how Type Theory can be made adequate in formalising certain constructions in category theory such as the construction of the category of all categories.

> (...) [in the intuitionistic theory of types] there are axioms for universes (in the sense of category theory) which link the generation of objects and types and play somewhat the same role for the present theory as does the replacement axiom for Zermelo-Fraenkel set theory. They also make the theory adequate for the formalization of certain constructions in category theory, like the construction of the category of all small categories.
>
> (Martin-Löf 1975a, p. 73.)

And indeed, type theory, whose first seminal ideas were laid down by B. Russell in his famous paper 'Mathematical logic as based on the theory of types' (Russell 1908) should prove a useful framework for a broader and more coherent conceptual basis for the foundations of mathematics than classical set theory. Some enthusiasts go further to claim that 'type theory is the proper foundations for mathematics' (Lambek & Scott 1986, Preface, p. viii). And indeed, the interplay between the new algebraic ideas and concepts brought in by category theory on the one hand, and the fresh ideas and revised logical notions given by Prof. Martin-Löf's seminal investigations on types, propositions, proofs and judgements, on the other hand, should have a significant rôle to play in the construction of a new foundational basis for mathematics, logics and computing science alike.

Concerning type theory and the logical foundations of computing science, it would seem reasonable to say that having obtained a formal, mathematical, counterpart to a philosophical argument on the rôle of the proof-theoretic notion of *reduction* rule, by providing a way of unifying the notion of well-ordering and cumulative hierarchies of well-founded types, I have provided a useful perspective on the connections between proof theory and computer programming. The idea that a *reduction* (*normalisation*) rule can be looked at as a semantic instrument should prove a useful conceptual view that could allow the unification of techniques from theories of abstract data type specification with techniques from proof theory, constructive mathematics and λ-Calculus. For that reason I also believe that these connections stand as an important hint of a new foundational view on the form of frameworks for formal software development, which could prove fruitful for a more methodological view of computer programming.

In the next chapter I shall be making an attempt to deal with typical problems of a certain mainstream of modern theoretical computer science, namely the theory of

abstract data types, using the framework of my own reformulation of *Intuitionistic Type Theory*.

7 *'Meaning-As-USE' Type Theory and Abstract Data Types*

7.1. Preamble

If one is to identify particularly general notions which have supported most of what has been done in clarifying the mathematics of computing as well as the computing of mathematical objects, one can hardly neglect either one of the key concepts in the modern theory of computing: *types* and *theories*. Two mainstreams of modern theoretical computer science bear witness to the convenience and usefulness of such concepts as *types* and *theories* in providing an intuitive formal account of the phenomenon of *computation*, even if they do not share a single uniform view on the matter. Despite the serious drawbacks both sides appear to have, and their divergences on the way a type is to be defined, there are good lessons to be learnt from both of them. If on the one hand, the theories of abstract data type specifications have provided us with useful insights into the nature of theory presentations, on the other hand, from *Type Theory* the lessons come from how types should be presented as well as where the focus of attention is to be more appropriately placed, namely the *judgements* one can make from a type presentation. Attempting to take the good

lessons from both sides, I have been engaged in the development of what seems to be a more coherent *Type Theory*, than, e.g., *Intuitionistic Type Theory*. The result is a logic-free theory of constructions named *'Meaning-As-USE' Type Theory* as an explicit reference to its semantical standpoint, which provides one with such a uniform presentation recipe: *theories* are presented as *types* of their 'terms'.

7.2. Introduction

An *operations*-based approach to the semantics of abstract data types specification should significantly simplify typical problems associated with implementation issues, error handling, parameterisation, structuring, etc.. This is exactly because the notion of semantics would be based on *operations* rather than the usual detour via *denotations*, and a number of these problematic issues could then be dealt with 'syntactically' (i.e., proof-theoretically). I shall describe how a particular reformulation of Prof. P. Martin-Löf's *Intuitionistic Type Theory* via an enrichment of its operational semantics, called 'semantics of use', where *normalisation* rules (rules of *use*) are taken as part of the meaning of the logical signs, resulting in a formal system named *'Meaning-As-USE' Type Theory*, can be used to fulfil the above objectives by presenting a more adequate '*operations*-based' semantics. Describing the features of such *'Meaning-As-USE' Type Theory* I present a much simpler approach to theories of specification and implementation of abstract data types by avoiding usual problems of explaining concepts such as 'implementation', 'parameterisation' and 'structuring' with a detour through denotational semantics via 'embedding of models', 'mappings of algebras', or an even more syntactical and simpler but yet unsatisfactory semantical explanation via 'interpretations between theories'.

Moreover, as an example of the identification of type presentations with theory presentations aforementioned, I demonstrate how a presentation of a Π-type of λ-terms compares to Prof. H. Barendregt's axiomatic presentation of a λ-theory. In addition to that, I present a discussion on how one deals with the notion of *implementation* within this setting, where *theory* presentations are identified with *type* presentations.

A number of issues commonly discussed in theories of abstract data type specifications, such as error-handling (error-recovery, error-propagation, etc.), implementation, completeness of axioms, the rôle of *hidden functions* and *if-then-else*, etc., is dealt with in a much simpler way than most of the approaches available in the literature. This is made possible for at least two fundamental reasons: first of all, the notion of semantics is taken as based essentially on *operations* rather than on *denotations*, which means that one does not have to introduce abstract concepts such as algebras and mappings to explain the semantics of the data types being specified. The semantics is explained via *normalisation operations* which I call here the rules of 'use'. Secondly, due to the fact that one stays within a single linguistic framework which may include the most primitive logical operations such as *pairing* and *splitting*, *lambda abstraction* and *application*, *sum* and *choice*, as well as *fixpoint* and *potential infinity*, one does not have to use extra notions to explain *implementations* and *proof of correctness* other than 'definitional extensions'.

7.2.1. Frege's notions of definition: constructive (abbreviatory) and analytic (presentatory)

In attempting to say something about specification of abstract data types one cannot avoid being involved in a discussion which concerns one of the most problematic issues in mathematics: *definitions*. In computing science the kind of definition one is mostly interested in seems to be definitions as *presentations*, instead of the perhaps more traditional definitions as *abbreviations*. One usually defines a logic, a data type, a recursive function, etc., via certain rules of *presentation*. Without going too far into a philosophical discussion which would certainly go beyond the aims of this chapter, I would like to take Frege's notions of definition as the starting point for my discussion. And the objective is clear: associate what I call *presentatory* definitions with Frege's *analytic* definition. I strongly feel that Frege has rightly pointed out the view that definitions are not always abbreviations, and both proof theory and the theory of abstract data types deal exactly with non-abbreviatory

definitions which are sometimes called 'presentations'. An assumption is made here that whenever I say something about *definitions* I will be referring to one of Frege's two distinct notions, where the distinction should be made clear either from the context or explicitly. This assumption shall be of fundamental importance for the understanding of my development of most of the issues covered here, including my attempt at giving a mathematical account of the notion of 'implementation as proof of existence', error handling and a number of other related topics which concern directly theories of specification of data types. I shall refer to non-abbreviatory (or *analytic* according to Frege) definitions by *presentatory* definitions or 'definition by use'.

In an essay on the fundamental rôle of logic in the mathematical activity G. Frege attempts to analyse somewhat more closely two essential concepts of primary use to mathematicians: *inferences* and *definitions*. Considering the latter as the most characteristic as well as most problematic concept of mathematics,

> (...) Part of the mathematician's activity besides drawing inferences, is to give definitions. Most disciplines are not concerned with the latter at all; (...)
>
> (Frege 1914, p. 203.)

he devotes most of the essay to a discussion centered around the concept of *definition*. The starting point is to say that in mathematics these two concepts, *inferences* and *definitions*, are subject to the laws of logic:

> *Inferring and defining are subject to logical laws.*
>
> (Id. Ibid.)

In the course of the discussion on the rôle of definitions in mathematics, he comes to a rather delicate point where he has to explain how a definition of a sign which had long established use can be given when 'constructing a new system of signs'. The point stressed is that a *definition*, strictly speaking, is supposed to attribute meaning to a sign which had no meaning before. In the case where one wants to build a new system of logical signs and there is the need to have a sign to represent

the logical symbol with long established use, the act of defining cannot be taken as just an abbreviation, given that the symbol had a meaning before, and the new sign has to reflect this 'pre-conceived' meaning in some way. That is perhaps where *definitions-as-presentations* replaces *definitions-as-abbreviations*.

In an attempt to clarify his own account of the distinction between these two notions of *definition*, Frege talks about two different cases: (1) One constructs a sense out of its constituents and introduce an entirely new sign to express this sense. Frege refers to this kind of definition as a 'constructive definition', but he prefers to call it a 'definition', *tout court*; (2) One has a simple sign with a long established use, which he believe he can give a logical analysis of its sense, obtaining a complex expression which in his opinion has the same sense. Only something with a recognizable sense is allowed as a constituent of a complex expression which is being abbreviated by the new sign being defined. The sense of the complex expression must be yielded by the way in which it is put together. That it agrees with the sense of the long established simple sign is not a matter for arbitrary stipulation, Frege argues, but can only be recognised by an immediate insight. One speaks of a definition in this case too. As Frege suggests, it might be called an *analytic* definition to distinguish it from the first case.

As is rightly pointed out by Frege, in the second case there remains no room for an arbitrary stipulation, because the simple sign already has a sense. Only a sign which as yet has no sense can have a sense arbitrarily assigned to it. As with the case of most definitions used in computing science, a science which deals thoroughly and mostly with codings, one usually wants to define something (a type, a logic, etc.) which already has a sense for the definer, and for which a suitable *presentation* is required.

From my own understanding of Frege's point of view, I say that the *analytic* definition is to have the form of 'definitions by use', where two aspects of the meaning are captured by (natural deduction-like *introduction* and *reduction* rules, plus *Intuitionistic Type Theory* notion of *propositions-are-types*), according to my 'semantics of use'. One can see the need for a different notion of *equality* (rather than the just

'abbreviatory' one, which is used in the first case) in order to clarify the nature and the rôle of such *logical analysis* suggested in the second case. From my own understanding of Frege's studies on *definitions*, I shall consider such *logical analysis* as the process of finding *implementations* for my definition as *presentation*, or 'definition by use', which occurs when abstract data types are specified (or *presented*). Such analysis will be similar to proving the existence lemma for the *systems* 'defined by use', as in Prof. Aczel's development of *Frege structures* (Aczel 1980). And looking at the problem of finding *implementations* as a procedure of proving an *existence* theorem, we can only think of 'meta-level' arguments as explanations of the constructions performed to arrive at object(s) which support(s) the proof of existence. I am not concerned here with the 'internalisation' of such a meta-linguistic step. For my present purposes, the explanation of such 'meta-linguistic' steps via notions such as 'mappings between algebras', 'interpretations between theories', etc., is not particularly suitable. Correctness of *implementations* shall again be proved similarly to Prof. Aczel's case, by showing that the 'objects' sought for do have the property required by the initial definition as *presentation*, or 'definition by use'.

7.3. Algebraic Specifications

Using the lesson learnt from theories of abstract data types, I want to argue that type presentations must show what can be done with the elements of the type. In other words, a type presentation has to show the *use* of the elements. Now, assuming that *use* can be characterised logically via rules of *normalisation*, which corresponds to a fundamental aspect of the meaning of a proposition, according to my own reading of Wittgenstein's later account of propositions, I attempt to provide a 'linguistic' account of meaning for the language of mathematics based on *operations* rather than on *denotations*, where no interpretation is involved.

Departing from the usual axiomatic presentation of theories, my *Type Theory* provides proper grounds for presenting theories via natural deduction-like proof rules. A number of equality rules given for each presentation of a type allows for the presentation of what is usually given as axioms of the theory. General substitution

and equality rules take part in the type-theoretic framework independently of a particular type presentation. I shall demonstrate below how to relate an axiomatic presentation of a λ-theory with a proof-theoretic presentation of a Π-type.

7.3.1. Type Definitions/Presentations

In algebraic specifications of data types the 'semantics of use' appears in the form of 'equations': complementing the syntactic information corresponding to the 'signature' of the type, one essentially states the rules of *reduction* for the elements of the particular type by showing how the destructors operate on the constructors. Once an element is built via the constructor(s), the semantics of use explains what its utility is, or, in other words, how it can be 'destroyed', what can be obtained from it.

One may remark that the meaning explanations given in framework such as *'Meaning-As-USE' Type Theory* appear to suggest a certain shift between use of a type and use of an element of a type. One could presumably justify such 'shift' by saying that in a type-theoretic framework there is hardly as much interest in the type-expressions as there is in the value-expressions (elements of the type). (Cf. the type-expression **Stack(A)** and the value-expressions `emptystack` and `push`(a, b)). The identification of *propositions* with *types*, which plays a crucial part in allowing the distinction '*propositions* and *judgements*' to be made, seems to bring into our type-theoretic explanation of meaning such inevitable and hardly undesirable shift in the focus of attention.

Now, given by the very nature of the 'semantics of use' where the meta-mathematical device of *normalisation* is taken as a formal counterpart to Wittgenstein's 'explanation of meaning' and 'the use of the sign in the language-games', *reduction* rules correspond to the *utility* aspect of the type. One gives the 'grammar' (*introduction* rules) aspect, which tells us how to *build* elements of the type, and the 'consequences' aspect, which shows how to *destroy* elements of the type, thus indicating what can be obtained from them, what their *utility* is. Therefore, instead of writing

down an arbitrary number of axioms for each type, we have a calculable principle for determining when we have enough.

Following the long mathematical tradition, a theory presentation in formal treatments of abstract data types is usually given by a definition of the *syntax* of the terms with which the theory will be concerned, and a list of axioms demonstrating the *semantics* of the terms. (Or 'pre-semantics', if one is not satisfied with the degree of precision given by the meaning definition of terms until a certain correspondence with 'mathematical objects' is established.)

Here is an example of a theory presentation (algebraic specification of the type of *stacks*), as given by H. Ehrig and B. Mahr in their compendium of techniques for algebraic specification of abstract data types:

stack =

sorts: *alphabet*

stack

opns: $K1, ..., Kn :\rightarrow$ *alphabet*

`error` $:\rightarrow$ *alphabet*

`empty` $:\rightarrow$ *stack*

`push` : *alphabet stack* $\rightarrow$ *stack*

`POP` : *stack* $\rightarrow$ *stack*

`TOP` : *stack* $\rightarrow$ *alphabet*

eqns: $x \in$ *alphabet*, $s \in$ *stack*

$\texttt{POP}(\texttt{push}(x, s)) = s$

$\texttt{TOP}(\texttt{push}(x, s)) = x$

$\texttt{POP}(\texttt{empty}) = \texttt{empty}$

$\texttt{TOP}(\texttt{empty}) = \texttt{error}$

(Ehrig & Mahr 1985, p. 52.)

Attempting to relate the above presentation of a theory of *stacks* to what prescribes the mathematical tradition of theory presentations, one could look at the lines under the heading *opns* as the syntactic definition of terms. They tell us what are the terms by demonstrating how to *form* the terms of the theory, e.g., 'empty', 'push(x, s)', etc.. The axioms listed under *eqns* could be associated with what we referred to above as *semantics* (or 'pre-semantics', if one prefers).

Looking at such a specification of **stacks** in the light of our semantics of use, we can say that the 'equations' (*eqns*) give the semantics to the type **stack** because they demonstrate the two ways of 'destroying' an element of the type built by the constructor 'push': via 'POP' and 'TOP', the two *eliminatory* operations. Regarding the special constant 'error', introduced to represent the effect of 'TOP' on the 'empty stack', it is worth pointing out a particularly crucial difference between the *operations*-based and the *denotations*-based approaches to semantics. In *MAUSE*, a framework where the semantics is based on *operations* (via proof-rules), 'equations' such as the ones referring to the effect of 'POP' and 'TOP' operations on the 'empty stack' simply do not exist just because the rules of manipulation of the type would never allow the *judgements* involving the membership of such terms to be made: *judgements* of the kind 'POP(empty) $\in$ **Stack(A)**' and 'TOP(empty) $\in$ **Stack(A)**' would never be reachable. The issue is 'shifted up' to the meta-level, so to speak. Compare, for instance, the simple case of the λ-calculus, where the destructor 'APPLY' is to operate on λ-*abstraction* terms only: in the axiomatic presentation of a λ-theory Prof. Barendregt includes an axiom corresponding to β-reduction, but none corresponding to the effect of 'applying' a term which is not a λ-abstraction — an arbitrary variable, for example — to an argument, (Barendregt 1981, p. 23). One could say that while the type Π of λ-terms is not inductive (does not have a base case), the types of natural numbers, lists, stacks, etc., are inductively generated from the 'base' (0, nillist, emptystack, etc.) and therefore one has to consider the result of the destructors over these boundary cases somehow. Now, it is true that in our type-theoretic framework the Π-type is not inductively generated and contains only 'λ-*abstraction*' terms, but this is not the case for the set 'Λ' of λ-

terms considered in Prof. Barendregt's axiomatisation because 'Λ' is an inductive set generated from a set of variables and does not contain only λ-*abstraction* terms (see clause '(1) $x \in \Lambda$, where x is an arbitrary variable', in Barendregt 1981, p. 22). Similarly to the usual definitions/presentations of inductive data types (natural numbers, lists, stacks, etc.) in Prof. Barendregt's definition/presentation of the set 'Λ' the destructor operates on specific constructors only (application on λ) and efforts to explain odd terms showing the effect of destructors on other than those (such as, e.g., '$\mathtt{APPLY}(x, a)$') might as well be spared.

What I do not think is a good lesson to be learnt from such an algebraic approach to *theory* presentations is the emphasis on the *values* of expressions, where constructors as well as destructors are taken to be *functions*, and the *value* of the expression formed by applying, e.g., a function like `TOP` to the `empty` stack is artificially introduced as an 'odd' constant value such as `error`, $\perp$, etc.. By shifting the focus of attention to the kinds of *judgements* one can make from a theory presentation, one realises that *judgements* of the form '`POP(empty)` is a *stack*' and '`TOP(empty)` is a data value from the *alphabet*' are simply not 'reachable'. The destructors `TOP` and `POP` are to be seen as *eliminatory* operators acting on the constructor `push`.

As we have already seen and shall see again later on, *Type Theory* teaches us where to place the emphasis, namely the *judgements* one can make from a *type* presentation.

In a *denotations*-based semantical framework, due to the unnecessary detour through concepts like functions, sets, embeddings, etc., 'spurious' constant values like '`error`', '$\perp$', etc., have to be introduced. And this is exactly because in this setting, where *interpretations* are supposed to give meaning, one is forced to ask such questions about what one could call 'false' problems. Instead of asking what can be produced by the rules of construction and destruction, one is tempted to ask about the *values* of expressions. Similarly, the need to use partial algebras and, to some extent, the abandonment of initial algebraic semantics in favour of certain 'loose' semantics, are both examples of complex approaches to solving these, as it were, 'false' problems.

7.3.2. Hidden Functions

We have no difficulty explaining the rôle of the so-called *hidden functions*, which are introduced in the specifications of certain data types but should remain 'hidden' from the 'user' of the data type.

For example, looking at the specification of an 'inductive' data type such as the type of **queues** of elements:

queue = bool + nat +

sorts:	*alphabet*
	queue
opns:	$K1, ..., Kn :\rightarrow$ *alphabet*
	error $:\rightarrow$ *alphabet*
	new $:\rightarrow$ *queue*
	addq : *queue alphabet* $\rightarrow$ *queue*
	REMOVE : *queue* $\rightarrow$ *queue*
	FRONT : *queue* $\rightarrow$ *alphabet*
	IS − EMPTY : *queue* $\rightarrow$ *bool*
	LENGTH : *queue* $\rightarrow$ *nat*
	IF − THEN − ELSE : *bool queue queue* $\rightarrow$ *queue*
eqns:	$a \in$ *alphabet*; $q, q1, q2 \in$ *queue*
(e1)	REMOVE(new) = new
(e2)	REMOVE(addq(q, a)) = IF IS − EMPTY(q) THEN new
	ELSE addq(REMOVE$(q), a)$
(e3)	FRONT(new) = error
(e4)	FRONT(addq(q, a)) = IF IS − EMPTY(q) THEN a
	ELSE FRONT(q)
(e5)	IS − EMPTY(new) = true
(e6)	IS − EMPTY(addq(q, a)) = false
(e7)	LENGTH(new) = 0

(e8) $\texttt{LENGTH}(\texttt{addq}(q, a)) = \texttt{succ}(\texttt{LENGTH}(q))$

(e9) $\texttt{IF true THEN } q1 \texttt{ ELSE } q2 = q1$

(e10) $\texttt{IF false THEN } q1 \texttt{ ELSE } q2 = q2$

(Ehrig & Mahr 1985, p. 55.)

it would seem fair to say that according to the semantics of our *'Meaning-As-USE' Type Theory* the rôle of 'IF-THEN-ELSE', 'IS-EMPTY' and 'LENGTH' in the above specification could be explained as follows:

$\texttt{IS} - \texttt{EMPTY}(q)$ (defined in (e5) and (e6)) is an example of what is usually called a *hidden function*, given that it is not concerned directly with the semantics of the type being specified. Such a 'function' is just acting as an 'auxiliary' function in the specification of the rules of *use* for the type **queue**.

$\texttt{IF} - \texttt{THEN} - \texttt{ELSE}$ is used to establish conditions for the 'destruction' operations on elements built by the constructors. In *MA USE* we have the conditional (eliminatory) operator 'WHEN' to realise the 'choice' made by such IF-THEN-ELSE without having to use 'extra' data such as **bool**.

$\texttt{LENGTH}(q)$, in (e7) and (e8), is effectively unnecessary as far as the (*use*) semantics of the type **queue** is concerned. It only describes a particular extra property of the elements of the type **queue**, which could perhaps be defined after the semantics of the type is given completely. Observe that by saying that types are many-sorted algebras is to leave completely arbitrary the criterion regarding the completeness of axioms w.r.t. the intended type, and the non-existence of a definite criterion to decide what are the 'functions' to express the distinguishing characteristic of the data type justifies the introduction of irrelevant (to the characterisation of the type!) functions such as 'LENGTH' in this case. A (rough) recipe such as 'a type is to be characterised by its constructors and by the effect of its destructors on the

terms built up by the constructors, would perhaps be a remedy. It is in fact our prescription with *MAUSE*.

Therefore, according to my 'semantics of use', the type **queue** is given its 'meaning' by the (*normalisation*) rules of *use* (e2 and e4), where the effect of 'destroying' an element built by the 'constructor' addq is demonstrated. Furthermore, considering the operations 'FRONT' and 'REMOVE' as *eliminatory* operations acting on canonical terms built by *introduction* inferences, there would be no need to say anything about 'unreachable' terms such as 'FRONT(new)' and 'REMOVE(new)'. The rules of use for the type **queue** would not include the possibility of making a *judgement* of the form 'FRONT(new) $\in$ **A**' or 'REMOVE(new) $\in$ **queue(A)**'.

7.3.3. Completeness of Axioms

Given by the very nature of the 'semantics of use', *reductions* corresponding to the *utility* aspect of the use of the type. One gives the 'grammar' (*introduction* rules) aspect, which tells us how to *build* elements of the type, and the 'consequences' aspect, which shows how to *destroy* elements of the type, thus indicating what can be obtained from them, what their *utility* is. Therefore, instead of writing down an arbitrary number of axioms for each type, we have a calculable principle for determining when we have enough. Let us take as an example of the specification of a data type the following specification of binary trees:

bintree-base =

sorts:	*alphabet*
	bintree
opns:	$K1, ..., Kn :\rightarrow$ *alphabet*
	`leaf` : *alphabet* $\rightarrow$ *bintree*
	`left` : *bintree alphabet* $\rightarrow$ *bintree*
	`right` : *alphabet bintree* $\rightarrow$ *bintree*
	`both` : *bintree alphabet bintree* $\rightarrow$ *bintree*

(Ehrig & Mahr 1985, pp. 16–7.)

bintree = bintree-base + nat + bool

opns: BREADTH : *bintree* → *nat*

EDGE : *bintree* → *nat*

NODE : *bintree* → *nat*

DEG : *bintree* → *bool*

eqns: $a \in alphabet; b, b1, b2 \in bintree$

$\texttt{BREADTH}(\texttt{leaf}(a)) = \texttt{succ}(0)$

$\texttt{BREADTH}(\texttt{left}(b, a)) = \texttt{BREADTH}(b)$

$\texttt{BREADTH}(\texttt{right}(a, b)) = \texttt{BREADTH}(b)$

$\texttt{BREADTH}(\texttt{both}(b1, a, b2)) = \texttt{add}(\texttt{BREADTH}(b1), \texttt{BREADTH}(b2))$

$\texttt{EDGE}(\texttt{leaf}(a)) = 0$

$\texttt{EDGE}(\texttt{left}(b, a)) = \texttt{succ}(\texttt{EDGE}(b))$

$\texttt{EDGE}(\texttt{right}(a, b)) = \texttt{succ}(\texttt{EDGE}(b))$

$\texttt{EDGE}(\texttt{both}(b1, a, b2)) = \texttt{succ}(\texttt{succ}(\texttt{add}(\texttt{EDGE}(b1), \texttt{EDGE}(b2))))$

$\texttt{NODE}(b) = \texttt{succ}(\texttt{EDGE}(b))$

$\texttt{DEG}(\texttt{leaf}(a)) = \texttt{true}$

$\texttt{DEG}(\texttt{left}(b, a)) = \texttt{DEG}(b)$

$\texttt{DEG}(\texttt{right}(a, b)) = \texttt{DEG}(b)$

$\texttt{DEG}(\texttt{both}(b1, a, b2)) = \texttt{false}$

(Ehrig & Mahr 1985, pp. 56–7.)

On the basis of Guttag's principle that a good axiomatisation must assert the distinguishing characteristic of the type and only that (Guttag 1977), it seems fair to say that the above specification of **bintree** is not a good one. Observe that none of the operations defined produce an element of the sort *alphabet*, which are

presumably the elements inserted into the data structure to be accessed later in some particular order. It is not indicated how to access the contents of each node in the binary tree. The purpose of building the data structure corresponding to a binary tree is not being made clear. Only general properties of the elements of the type are being given (breadth, number of edges, number of nodes, etc.). (In passing, this is, I believe, a clear sign of the extensional view inherited from classical mathematics.) Therefore, it cannot in any reasonable sense be said to be complete. In fact, the above seems to be essentially an extension by definition of some underlying undefined type.

An 'enriched' specification of **bintree**, including a new operation called HEIGHT is given as:

bintree&height = **bintree** +

opns: HEIGHT : *bintree* → *nat*

MAX : *nat nat* → *nat*

eqns: $a \in alphabet; b, b1, b2 \in bintree; m, n \in nat$

$\texttt{HEIGHT}(\texttt{leaf}(a)) = (0)$

$\texttt{HEIGHT}(\texttt{left}(b, a)) = \texttt{succ}(\texttt{HEIGHT}(b))$

$\texttt{HEIGHT}(\texttt{right}(a, b)) = \texttt{succ}(\texttt{HEIGHT}(b))$

$\texttt{HEIGHT}(\texttt{both}(b1, a, b2)) =$

$\quad \texttt{succ}(\texttt{MAX}(\texttt{HEIGHT}(b1), \texttt{HEIGHT}(b2)))$

$\texttt{MAX}(0, n) = n$

$\texttt{MAX}(m, 0) = m$

$\texttt{MAX}(\texttt{succ}(m), \texttt{succ}(n)) = \texttt{succ}(\texttt{MAX}(m, n))$

(Ehrig & Mahr 1985, pp. 161–2.)

Again, none of the 'equations' of the above enrichment show the real *utility* of the data structure, or, to use Guttag's terms, the actual 'distinguishing characteristic' of the data type. Or, put it another way, they show how to further *extend* its use

once it has been defined. None of them concern the aspect of how to obtain an element (of type *alphabet*) from the tree. As in the previous case, it is not properly shown how an element of the type **bintree** can be 'destroyed', therefore leaving a gap in the 'semantics of use' for the particular type. Obviously, if the semantics is to be given by an interpretation into an initial algebra, continuous algebra, etc., there is little concern in making the 'distinguishing characteristic' of the data structure simply by its axiomatisation.

In contrast, in my type-theoretic framework one would write down semantical *reduction* rules corresponding to what can be 'obtained' from each element of the type, like 'TOP' and 'POP' for the case of **Stack**. For the case of binary trees one writes:

DEFINITION 7.3.1. Let **Tree(A)** be the type of binary trees of elements of type **A**, defined inductively as:

BinTree-*formation*

$$\frac{\mathbf{A}\ \mathit{type}}{\mathbf{BinTree(A)}\ \mathit{type}} \qquad \frac{\mathbf{A} = \mathbf{B}}{\mathbf{BinTree(A)} = \mathbf{BinTree(B)}}$$

BinTree-*introduction*

$$\mathtt{niltree} \in \mathbf{BinTree(A)} \qquad \mathtt{niltree} = \mathtt{niltree} \in \mathbf{BinTree(A)}$$

$$\frac{a \in \mathbf{BinTree(A)} \quad b \in \mathbf{A} \quad c \in \mathbf{BinTree(A)}}{\mathtt{node}(a, b, c) \in \mathbf{BinTree(A)}}$$

$$\frac{a_1 = a_2 \in \mathbf{BinTree(A)} \quad b_1 = b_2 \in \mathbf{A} \quad c_1 = c_2 \in \mathbf{BinTree(A)}}{\mathtt{node}(a_1, b_1, c_1) = \mathtt{node}(a_2, b_2, c_2) \in \mathbf{BinTree(A)}}$$

BinTree-*reduction*

$$\frac{a \in \mathbf{BinTree(A)} \quad b \in \mathbf{A} \quad c \in \mathbf{BinTree(A)}}{\mathtt{LABEL}(\mathtt{node}(a, b, c)) = b \in \mathbf{A}}$$

$$\frac{a \in \mathbf{BinTree(A)} \quad b \in \mathbf{A} \quad c \in \mathbf{BinTree(A)}}{\mathtt{LEFT}(\mathtt{node}(a,b,c)) = a \in \mathbf{BinTree(A)}}$$

$$\frac{a \in \mathbf{BinTree(A)} \quad b \in \mathbf{A} \quad c \in \mathbf{BinTree(A)}}{\mathtt{RIGHT}(\mathtt{node}(a,b,c)) = c \in \mathbf{BinTree(A)}}$$

BinTree-*induction*

(instantiated from the general fixpoint rules for the 'Fix' operator.) □

The example should make clear my own point of view regarding what is needed in order to specify a data type: rules saying how to 'construct' elements of the type, and rules demonstrating how to 'destroy' an element of the type. This should hopefully give us reasonable criteria for judging whether a certain specification is good or bad, complete or incomplete, semantically 'clean', and so on.

Taking the example of specifying the type of natural numbers, I say that instead of specifying such a type as:

$$\begin{cases} \mathbf{succ}(x) = \mathbf{succ}(y) \rightarrow x = y \\ \neg(\mathbf{succ}(x) = 0) \end{cases}$$

(Maibaum, Sadler & Veloso 1981, p. 17.)

or

$$\begin{cases} \vdash \neg(\mathbf{succ}(m) =_{nat} zero) \\ \vdash \mathbf{succ}(m) =_{nat} \mathbf{succ}(n) \rightarrow m =_{nat} n \end{cases}$$

(Turski & Maibaum 1987, p. 74, eqs. 2.13 and 2.14.)

which are usually written down to represent the first two of Peano's axioms, my 'semantics of use' of *MAUSE* prescribes a specification in the form of the $\mathcal{N}$ specification given in previous sections, where the rôle of each step on the definition is made semantically clear. It is perhaps worth recalling that instead of having an *axioms*-based presentation of the logical signs, we have a *judgement* rules-based

presentation system. As we have seen from a previous explanation of the kind of presentation system we have in my *'Meaning-As-USE' Type Theory*, we have the usual natural deduction style of presenting logics, combined with the conceptually crucial distinction between *propositions* and *judgements*, and the identification *propositions-as-types* as the basis for the semantical explanations needed to justify the whole body of new concepts introduced by such Type Theory-like frameworks. In addition to the usual semantical explanations via *introduction* rules, such as in *Intuitionistic Type Theory*, we take the *reduction* (*normalisation*) rules as part of the formalisation of the semantics of the logical (as well as extra-logical) constants.

The whole idea of my semantical explanation could perhaps be summarised for the present example as follows: the rules of *introduction* show how to build the elements 0 and $\mathsf{succ}(a)$ of $\mathcal{N}$, and the rules of *reduction* show the effect of the destructor 'PRED' on the constructor 'succ', covering the respective fundamental aspects of the *use* of the type $\mathcal{N}$. And similarly to the other cases mentioned above (for instance 'APPLY' for the Π-type), the destructor 'PRED' would not be taken as a function but as an operation, following the important distinction between functions and operations that is necessary to make in constructive mathematics (Beeson 1985, p. 34).

As a matter of fact, one of the two axioms of the examples given above can be derived in the form of an inference of the kind:

$$\frac{\mathsf{succ}(a) = \mathsf{succ}(b) \in \mathcal{N}}{\mathsf{PRED}(\mathsf{succ}(a)) = \mathsf{PRED}(\mathsf{succ}(b)) \in \mathcal{N}}$$

from a definition of $\mathcal{N}$ in our type-theoretic framework. The other axiom, where the logical notion of negation is used to say that 0 is not the successor of any natural number, would be shown in *$\mathcal{N}$-introduction* (cf. the presentation of **Bintree(A)** and its two canonical elements given in its *introduction* rule. Such *negative* information provided by the equation shown carries information that is 'shifted up' to the meta-level, so to speak. More importantly, note that the specification of types in *MAUSE* expresses only *positive* information about their elements.

In some cases of usual algebraic specification of data types, no use of negative information is made, and only positive information is given in the specification, but another usual misconception, as far as my semantical point of view is concerned, is the inclusion of *recursive* functions over the elements of the type. For instance, in Ehrig & Mahr 1985, a specification for the type of natural numbers is given as:

nat =

sorts: $\rightarrow$ *nat*

opns: $0 :\rightarrow$ *nat*

$\mathtt{succ} : nat \rightarrow nat$

$\mathtt{ADD} : nat\ nat \rightarrow nat$

eqns: $n, m \in nat$

$\mathtt{ADD}(n, 0) = n$

$\mathtt{ADD}(n, \mathtt{succ}(m)) = \mathtt{succ}(\mathtt{ADD}(n, m))$

(Ehrig & Mahr 1985, p. 14.)

Note that the distinguishing characteristic of the type of natural numbers is hardly acknowledged to be the fact that we can perform addition with the elements. We can also perform multiplication, exponentiation, etc.. What seems to characterise the type of natural numbers is its canonical elements and their use, together with its inductive structure.

7.3.4. Parameterisation and Error-propagation

In the framework of my type-theoretic *normalisation*-based semantics, which is characterised, I insistedly repeat, by taking the rules of *introduction* and *reduction* as corresponding to the 'definition by use' of the type in *MAUSE*, one has a much simpler way of dealing with parameterised types, given that, for instance, 'error' *judgements* are neither part of nor derivable from the type presentation. One simply takes the problem of parameterised specification simply as that of bringing in

the rules of use corresponding to the parameters. Problems typical of *denotations*-based approaches such as dealing with error-propagation, finding proper semantical explanations for the act of instantiating a parameterised specification making sure that the target type matches the properties of the formal parameter type, and so on, are not of concern when the semantics is given via *normalisation* rules, and there is no need to make a detour via interpretations.

The algebraic approach is not free of big difficulties, an two entire chapters of Ehrig & Mahr 1985 are dedicated to the topics of Parameterised Specification and Parameter Passing. A formal definition of Parameterised Specification is given as:

7.1 DEFINITION (Parameterized Specification)

A parameterized specification PSPEC = (SPEC,SPEC1) consists of a pair of specifications

SPEC = (S, OP, E), called formal parameter specification, and

SPEC1 = SPEC + (S1, OP1, E1), called target specification.

(Ehrig & Mahr 1985, p. 171.)

The above '+' operation on specifications is regarded in our case as just a juxtaposition of the rules of *use* (including *introduction* and *reduction* rules) of the parameterised type and the type corresponding to the instantiation of the parameter.

To have 'parameters' in the specification of a certain type in a framework like our *'Meaning-As-USE' Type Theory* does not seem to bring in extra problems such as those treated by the more *denotations*-based approaches. As we could see from the DEFINITION 7.3.1 of binary trees of elements of type **A**, the rules of *formation*:

BinTree-*formation*

$$\frac{\mathbf{A}\ \mathit{type}}{\mathbf{BinTree(A)}\ \mathit{type}} \qquad \frac{\mathbf{A} = \mathbf{B}}{\mathbf{BinTree(A)} = \mathbf{BinTree(B)}}$$

certify that the parameter **A** is indeed a type. Furthermore, to bring in the semantics of the type **A** is as simple as to bring in its rules of *use*. And, again,

similarly to Natural Numbers, Lists, Stacks, etc., there is no rule to allow *judgements* like 'LABEL(niltree) $\in$ **A**', or 'LEFT(niltree) $\in$ **BinTree(A)**', or even 'RIGHT(niltree) $\in$ **BinTree(A)**' to be made.

7.3.5. Implementation

As we have seen from all the explanation given in previous sections, the explanation of the nature of *implementations* is given via 'definitional extensions', according to my own understanding of Frege's '*analytic* definition' which prescribes a definition by normalisation (*use*) rules, and subsequent proof of existence via '*constructive* definitions', rather than a certain translation between theories. This means that we have here a purely syntactic means rather than algebraic embedding of models or mapping of algebras. It constitutes a much simpler approach than 'interpretations between theories', just because the implementation process is explained via 'definitional extensions', rather than a certain translation between (languages and) theories.

For example, in Aczel 1980 a 'definition by use' of a *pairing system* and its corresponding proof of existence, which serves as proxy to our understanding of the process of laying down a specification of a certain type and the subsequent search for an *implementation* of such a 'type' (or *system*) in terms of more primitive ones, is given as:

Definition 5.6. A *pairing system* consists of $\mathcal{F}$-functions PAIR : $\mathcal{F}_0 \times \mathcal{F}_1 \rightarrow \mathcal{F}_0$ and $p, q : \mathcal{F}_0 \rightarrow \mathcal{F}_0$ such that for all objects a, b

$$p(\mathrm{PAIR}(a,b)) = a \text{ and } q(\mathrm{PAIR}(a,b)) = b.$$

Lemma 5.7. *There is a pairing system.*

Proof. $\mathrm{PAIR}(a,b) = \lambda x \mid x|_2(a,b)$ for all objects a, b and for each object c let $p(c) = \mathrm{APP}(c, \Delta_1)$ and $q(c) = \mathrm{APP}(c, \Delta_2)$.

Clearly these are $\mathcal{F}$-functions and for all objects a, b

$$p(\mathrm{PAIR}(a,b)) = \mathrm{APP}(\mathrm{PAIR}(a,b), \Delta_1)$$

$$= \mathrm{APP}(\lambda x \mid x|_2(a,b), \Delta_1)$$
$$=| \Delta_1|_2(a,b)$$
$$= a$$

and

$$q(\mathrm{PAIR}(a,b)) =| \Delta_2|_2(a,b) = b.$$

(Aczel 1980, p. 48.)

(Without going into a detailed explanation of Prof. Aczel's notation, it is relevant to point out that 'Δ_1', 'Δ_2' and '|' are all previously defined in terms of the *lambda system* constituted by 'λ' and 'APP'.)

In order to make it easier for a comparison between Prof. Aczel's proof of the Existence Lemma and my interpretation of how to prove the existence of an *implementation* from a *definition by use*, I rewrite the above proof as:

PROOF. Put

$\mathrm{PAIR}(a,b) \equiv \lambda x \mid x|_2(a,b)$

$p(c) \equiv \mathrm{APP}(c, \Delta_1)$

$q(c) \equiv \mathrm{APP}(c, \Delta_2)$

For the *correctness* of the implementation, we have to check that:

$p(\mathrm{PAIR}(a,b)) = a$ and

$q(\mathrm{PAIR}(a,b)) = b.$

and we do that by the following sequence of substitutions and rewritings:

$$\begin{aligned} p(\mathrm{PAIR}(a,b)) &\equiv \mathrm{APP}(\mathrm{PAIR}(a,b), \Delta_1) \\ &\equiv \mathrm{APP}(\lambda x \mid x|_2(a,b), \Delta_1) \\ &=| \Delta_1|_2(a,b) \\ &= a \end{aligned}$$

Similarly:

$$\begin{aligned} q(\mathrm{PAIR}(a,b)) &\equiv \mathrm{APP}(\mathrm{PAIR}(a,b),\Delta_2) \\ &\equiv \mathrm{APP}(\lambda x \mid x|_2(a,b),\Delta_2) \\ &= \mid \Delta_2|_2(a,b) \\ &= b \end{aligned}$$

■

Note that an *implementation* of the *pairing system* is found in terms of more primitive *systems* (*lambda system*, in this case), and the proof that such an *implementation* obeys the properties given in the 'definition by use' comes afterward. Finding such an *implementation*, e.g., when 'constructing' the expression '$\lambda x \mid x|_2(a,b)$' to do what 'PAIR(a,b)' does, was very much like proving an *existence* lemma, and so, the very *process* of choosing the 'object' sought is kept at the 'meta-level', as it were.

Furthermore, observe that Prof. Aczel's procedure is indeed to provide definitional extensions. Although using '=' for the definition of 'PAIR', 'p' and 'q' in terms of the (previously defined) *lambda system* objects, such definition is better seen as a definitional (abbreviatory) equality '$\equiv$'. Therefore, despite introducing 'new' objects such as 'PAIR', 'p' and 'q', they can be considered later on (after the proof of correctness) as mere *abbreviations*. That is exactly the point of view we take regarding the notion of *implementation* via definitional extensions. (see the *implementation* – proof of existence – of the type **Stack(A)** in terms of 'Σ' (*pairing/splitting*), '+' (*marking/choosing*) and 'Fix' (*potential infinity*), chapter 6.)

Similarly, a *Peano system* is defined by a presentation of the constructors and DESTRUCTORS, and 'equations' showing how the latter act on the former, and the existence lemma (implementation lemma, according to my interpretation) is stated and the proof of correctness is given as the exhibition of 'representatives' of the constructors and DESTRUCTORS in terms of the constructors and DESTRUCTORS of the previously defined *pairing system*:

Definition 5.9. A *Peano system* consists of an object 0 and $\mathcal{F}$-functions s, PRED, and DEC : $\mathcal{F}_0 \rightarrow \mathcal{F}_0$ such that for all objects a

$$\text{PRED}(s(a)) = a,$$

$$\text{DEC}(s(a)) = \Delta_2$$

and

$$\text{DEC}(0) = \Delta_1.$$

Lemma 5.10. *There is a Peano system.*

Proof. Let $0 = \text{PAIR}(\Delta_1, \Delta_2)$, $s =< \text{PAIR}(\Delta_2, x) \mid x >$, PRED $= q$ and DEC $= p$.

(Aczel 1980, p. 49.)

The 'proof of correctness', which would prove that indeed the representative of the DESTRUCTOR 'PRED' (namely 'q') act on the constructor 's' (namely '$<$ PAIR$(\Delta_2, x)|x >$') in the same way as required by the definition/presentation of the *Peano system* (**Definition 5.10.**), was omitted, but could easily be done as:

PROOF. Let

$0 \equiv \text{PAIR}(\Delta_1, \Delta_2)$

$s \equiv< \text{PAIR}(\Delta_2, x) \mid x >$

$\text{PRED} \equiv q$

$\text{DEC} \equiv p.$

It is required that:

$$\begin{aligned}\text{PRED}(s(a)) &\equiv q(s(a)) \\ &\equiv q(< \text{PAIR}(\Delta_2, x) \mid x > (a)) \\ &= q(\text{PAIR}(\Delta_2, a)) \\ &= a\end{aligned}$$

■

This example is a rather adequate case for an outline of my point of view on the whole discussion of how *implementations* relate to specifications of abstract data

types, and what form the specification itself as well as the 'proof of correctness' of the *implementation* should generally take.

Another example of an 'implementation' of a type into another which could fit the above explanation of the notion of implementation is that of S. Peyton-Jones in his book on Functional Languages (Peyton-Jones 1987):

2.2.2.3 A larger example

As a larger example, we will demonstrate the somewhat surprising fact that data constructors can actually be modelled as pure lambda abstractions. We define CONS, HEAD and TAIL in the following way:

CONS = (λa.λb.λf.f a b)
HEAD = (λc.c (λa.λb.a))
TAIL = (λc.c (λa.λb.b))

These obey the rules for CONS, HEAD and TAIL given in Section 2.1.3. ['HEAD (CONS a b) → a' and 'TAIL (CONS a b) → b'] For example,

	HEAD (CONS p q)
=	(λc.c (λa.λb.a)) (CONS p q)
→	CONS p q (λa.λb.a)
=	(λa.λb.λf. f a b) p q (λa.λb.a)
→	(λb.λf. f p b) q (λa.λb.a)
→	(λf. f p q) (λa.λb.a)
→	(λa.λb.a) p q
→	(λb.p) q
→	p

(Peyton-Jones 1987, p. 17.)

The proof of correctness would be completed when the same sequence of substitutions and rewritings would be done for the case of the destructor TAIL acting on the constructor CONS.

Another relevant example of a similar procedure of 'implementing' and 'proving correctness' is to be found in Lambek & Scott's account of the connections between type theory and category theory:

Definition 17.1. An *extended λ-calculus* is given by a set of terms and equations which are said to 'hold'. (...)

(...) Finally, the following specific equations are postulated, for a, b, c, f and $\varphi(x)$:

L1. $\lambda_x \varphi(x)^{\varsigma} x = \varphi(x)$;

L2. $\lambda_x (f^{\varsigma} x) = f$, if x is not free in f;

L3. $\pi((a, b)) = a$;

L4. $\pi'((a, b)) = b$;

L5. $(\pi(c), \pi'(c)) = c$.

(Lambek & Scott 1986, p. 101.)

followed by the proposition that would correspond to an 'existence lemma':

Proposition 17.4. Every extended λ-calculus $\mathcal{L}$ gives rise to a C-monoid $M(\mathcal{L})$, whose elements are (equivalence classes of) closed terms of $\mathcal{L}$ and where

$$\begin{aligned} 1 &\equiv \lambda_x x, \\ gf &\equiv \lambda_x (g^{\varsigma}(f^{\varsigma} x)), \\ \pi &\equiv \lambda_x \pi(x), \\ \pi' &\equiv \lambda_x \pi'(x), \\ < f, g > &\equiv \lambda_x (f^{\varsigma} x, g^{\varsigma} x), \\ \varepsilon &\equiv \lambda_z (\pi(z)^{\varsigma} \pi'(z)), \\ h^* &\equiv \lambda_x \lambda_y (h^{\varsigma}(x, y)). \end{aligned}$$

Moreover, $a = b$ in $M(\mathcal{L})$ if $a = b$ holds in $\mathcal{L}$.

Proof. The equations of a C-monoid are easily checked. (...)

(...) Unfortunately, it is not obvious that these give an equivalence of categories. The reason for this is that in Proposition 17.3 [where pairing with '(,)' and 'π, π'' were defined] the definitions of $\pi(-)$, and $\pi'(-)$ and (–,–) were badly chosen.

(Lambek & Scott 1986, p. 103.)

The 'proof of correctness' comes right after the choice of (better) candidates for the representatives of `constructors` and `DESTRUCTORS` of both the abstraction/application ('λ' / '$^{\varsigma}$', respectively) part of the definition as well as of the pairing/splitting ('(,)' / 'π' and 'π'', respectively), in a similar way by proving

that the effect of DESTRUCTORS on corresponding constructors is the same as in the original definitions given in L1–L5:

> Let us change the term forming operations $\pi(-)$, $\pi'(-)$ and $(-,-)$ in Proposition 17.3 as follows:
> $\pi(c) \equiv \pi^{\varsigma} c$,
> $\pi'(c) \equiv \pi'^{\varsigma} c$,
> $(a,b) \equiv < \lambda_x a, \lambda_x b >^{\varsigma} 1$,
> where it is assumed that x is not freee in a or b. We now calculate:

$$\begin{aligned}
\pi((a,b)) &= \pi^{\varsigma}(< \lambda_x a, \lambda_x b >^{\varsigma} 1), \\
&= (\pi < \lambda_x a, \lambda_x b >)^{\varsigma} 1 && \text{by C10,} \\
&= (\lambda_x a)^{\varsigma} 1 && \text{by C1,} \\
&= a;
\end{aligned}$$

$$\begin{aligned}
(\pi(c), \pi'(c)) &= < \lambda_x(\pi^{\varsigma} c), \lambda_x(\pi'^{\varsigma} c) >^{\varsigma} 1, \\
&= < \pi\lambda_x c, \pi'\lambda_x c >^{\varsigma} 1 \quad \text{by C11,} \\
&= (\lambda_x c)^{\varsigma} 1 \\
&= c.
\end{aligned}$$

(Ibid., pp. 103–4.)

To that 'proof of correctness' one could easily add the proof that the effect of the other DESTRUCTOR for the pairing constructor '(,)', namely '$\pi'((a,b)) = b$', also verifies.

7.3.5.1. Implementation and Lists *vs.* Stacks

In order to demonstrate a simple (practical) consequence of looking at the distinguishing characteristic of types as being given primarily by the effect of the DESTRUCTORS on the terms built by the constructors, I choose the (extremely) simple example of implementing a theory of stacks (**Stack(A)**) in terms of a theory of lists (**List(A)**). The aim here is twofold: first of all I want to give a further ('practical') example of how the process of finding an *implementation* and proving its correctness relates to the usual process of proving an existence theorem by

choosing a suitable candidate and proving that such a candidate conforms to the requirements set up by the definition-as-presentation. The choice of a candidate is understood here as a meta-linguistic step and its being formalisable is a question which is not considered here as I have mentioned above, for obvious reasons. The rôle of definitional extensions is made clear in my logical account of the process of *implementation* where the 'newly' defined operators *abbreviate* the (perhaps complex) candidates. Secondly, I purposedly choose the example of implementing stacks in terms of lists to show how a more intuitively appealing semantical view can make artificial distinctions disappear such as the distinction between what seem to be two instances of the same data structure. As will become clear, the only distinction one can think of making between a stack and a (LISP-like) list is that stacks are usually thought of as structures growing/shrinking 'vertically', whereas lists are imagined as an 'horizontal' data structure. The *utility* of both data types, however, which is to provide a 'data memory' accessed according to the protocol 'last in, first out', is essentially the same if one abstracts from the names: `cons`, as opposed to `push`; `CAR` and `CDR`, as opposed to `TOP` and `POP`. It seems that the distinction is merely cosmetic, or even of 'geometric' positioning. Nevertheless, there seems to be unnecessary effort to draw a distinction between a stack and a (`cons`/`CAR`/`CDR`-like) list, even in texts dedicated to list-structured programming, such as, e.g.:

> The mechanism which we described in the initial blocks model occurs quite naturally in computer science. It is called a **stack**. The important characteristics of stacks are that they are lists such that additions and deletions can only be made at the front of the list.
>
> (Allen 1978, pp. 290–1.)

There are certainly many different notions of lists in computer science, such as, e.g., linked lists, doubly linked lists, etc.. However, the simple notion of list as that of LISP-lists with cons/car/cdr, where the additions and deletions can only be made at the front of the list, does not differ from stacks in that respect. An awareness of the *utility* of data structures as being the determining factor of their *meaning*, would certainly help to see that one can have a stack in a LISP-like programming language just by using the usual primitive operations cons/car/cdr as

stack operations (push/top/pop). And this would prevent spending much effort in constructing a, perhaps complicated, LISP implementation of stacks. E.g., after a certain renaming of list operations, such as:

> Though our representation of sequences is such that *first*, *rest* and *concat* are identical to *car*, *cdr* and *cons* respectively, we should use the names *first*, *rest*, and *concat* to make it clear that we are operating on lists.
>
> (Allen 1978, p. 40.)

an implementation of stacks is given essentially as:

$push \Leftarrow \lambda[[i; st]concat[i; st]]$
$top \Leftarrow \lambda[[st]first[st]]$
$pop \Leftarrow \lambda[[st]rest[st]]$

(Allen 1978, p. 392.)

It would certainly make things much easier to think of stacks as being essentially the same data type as cons/car/cdr-lists. Observe that the 'representation' of *push*, *top* and *pop* is no more than a change in the names *cons*, *car* and *cdr*, respectively.

Now, I give a presentation of both data structures following the semantics of use which regulates my *'Meaning-As-USE' Type Theory*, and demonstrate how one can immediately see that there can only be an artificial distinction between cons/car/cdr lists and push/top/pop stacks. The procedure can even be considered as essentially similar to the example aforementioned, namely that of giving abbreviatory definitions ('$\equiv$'). But the point here is that I shift the focus of attention to the *utility* aspect of the presentation of the data type, and I prove the *correctness* of the representation by showing that the representation satisfies the *utility* requirements captured by the *reduction* rules.

The theory of lists is presented as the type **List(A)** as follows:[1]

[1]Note that, as expected, it bears close resemblance to the definition of LISP-lists with cons/car/cdr:

DEFINITION 7.3.1. Let $\mathbf{List(A)}$ be the type of lists of elements from the type $\mathbf{A}$, defined inductively as:

$\mathbf{List(A)}$-*formation*

$$\frac{\mathbf{A}\ type}{\mathbf{List(A)}\ type} \qquad \frac{\mathbf{A} = \mathbf{B}}{\mathbf{List(A)} = \mathbf{List(B)}}$$

$\mathbf{List(A)}$-*introduction*

$$\mathtt{nillist} \in \mathbf{List(A)} \qquad \mathtt{nillist} = \mathtt{nillist} \in \mathbf{List(A)}$$

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{List(A)}}{\mathtt{cons}(a,b) \in \mathbf{List(A)}} \qquad \frac{a = c \in \mathbf{A} \quad b = d \in \mathbf{List(A)}}{\mathtt{cons}(a,b) = \mathtt{cons}(c,d) \in \mathbf{List(A)}}$$

$\mathbf{List(A)}$-*reduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{List(A)}}{\mathtt{CAR}(\mathtt{cons}(a,b)) = a \in \mathbf{A}} \qquad \frac{a \in \mathbf{A} \quad b \in \mathbf{List(A)}}{\mathtt{CDR}(\mathtt{cons}(a,b)) = b \in \mathbf{List(A)}}$$

$\mathbf{List(A)}$-*induction*

(instantiated from the generalised induction operator 'Fix'.)

□

'Given any S-expression, it is possible to produce any subexpression of it by a suitable composition of car's and cdr's. If x and y represent any two S-expressions, the following identities are true:

car[cons[x;y]]=x

cdr[cons[x;y]]=y.

The following identity is also true for any S-expression x such that x is composite (non-atomic):

cons[car[x];cdr[x]]=x.'

(McCarthy et. al. 1962, p. 3.)

The theory of stacks, presented as a type which I name **Stack(A)**, again follows the pattern of presentation of types as prescribed by the semantics of use.

DEFINITION 7.3.2. Let **Stack(A)** be the type of lists of elements from the type **A**, defined inductively as:

Stack(A)-*formation*

$$\frac{\mathbf{A}\ \mathit{type}}{\mathbf{Stack(A)}\ \mathit{type}} \qquad \frac{\mathbf{A} = \mathbf{B}}{\mathbf{Stack(A)} = \mathbf{Stack(B)}}$$

Stack(A)-*introduction*

$$\mathtt{emptystack} \in \mathbf{Stack(A)} \qquad \mathtt{emptystack} = \mathtt{emptystack} \in \mathbf{Stack(A)}$$

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{Stack(A)}}{\mathtt{push}(a,b) \in \mathbf{Stack(A)}} \qquad \frac{a = c \in \mathbf{A} \quad b = d \in \mathbf{Stack(A)}}{\mathtt{push}(a,b) = \mathtt{push}(c,d) \in \mathbf{Stack(A)}}$$

Stack(A)-*reduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{Stack(A)}}{\mathtt{TOP}(\mathtt{push}(a,b)) = a \in \mathbf{A}} \qquad \frac{a \in \mathbf{A} \quad b \in \mathbf{Stack(A)}}{\mathtt{POP}(\mathtt{push}(a,b)) = b \in \mathbf{Stack(A)}}$$

Stack(A)-*induction*

(instantiated from the generalised induction operator 'Fix'.)

□

LEMMA 7.3.3. There is a type **Stack(A)**.

PROOF. Let

$\mathtt{emptystack} \equiv \mathtt{nillist}$,

$\mathtt{push}(a,b) \equiv \mathtt{cons}(a,b)$.

and

$$\mathtt{TOP}(s) \equiv \mathtt{CAR}(s),$$
$$\mathtt{POP}(s) \equiv \mathtt{CDR}(s).$$

Now I prove the *correctness* of the implementation by showing that indeed:

$$\mathtt{TOP}(\mathtt{push}(a,b)) = a,$$
$$\mathtt{POP}(\mathtt{push}(a,b)) = b.$$

I just have to use our definitions:

$$\mathtt{TOP}(\mathtt{push}(a,b)) \equiv \mathtt{CAR}(\mathtt{push}(a,b))$$
$$\equiv \mathtt{CAR}(\mathtt{cons}(a,b))$$
$$= a.$$

$$\mathtt{POP}(\mathtt{push}(a,b)) \equiv \mathtt{CDR}(\mathtt{push}(a,b))$$
$$\equiv \mathtt{CDR}(\mathtt{cons}(a,b))$$
$$= b.$$

■

Whenever the *utility* aspect of a data type is neglected in favour of names or even (numerical) values, a number of misunderstandings are likely to arise. The above example demonstrates that rather clearly. The artificiality of such a distinction between a stack and a cons/CAR/CDR-list has in fact remained unquestioned for a long time. Even in more recent textbooks on Functional Programming, a paradigm of computer programming which was born out of LISP-like programming languages, one can see that little attention is paid to the fact that the *utility* of stacks ('last-in-fist-out' memory), expressed by its *reduction* rules $\mathtt{TOP}(\mathtt{push}(a,b)) = a$ and $\mathtt{POP}(\mathtt{push}(a,b)) = b$, is essentially the same as that of LISP-lists with $\mathtt{CAR}(\mathtt{cons}(a,b)) = a$ and $\mathtt{CDR}(\mathtt{cons}(a,b)) = b$. In Henderson 1980, e.g., 'implementing' a stack on a LISP-like programming language is not just reduced to saying that one uses cons as the stack push constructor and CAR/CDR as the stack destructors TOP/POP but an explanation of how one should represent a

stack is given with no reference to the fact that they cannot be distinguished up to names:

(...) we can implement a stack structure as an S-expression, simply by using a list of the stacked elements and pushing and poping values from the left-hand end of that list. Thus, for example,

$$(A\ B\ C\ D)$$

could represent a stack of exactly 4 items.

(Henderson 1980, p. 166.)

Such an explanation, however, amounts to the simple conclusion that stacks are the same data structures as (LISP-like) lists.

As a matter of fact, most of the English dictionaries I have encountered explain the word 'list' merely as an enumeration, a roll, etc., no reference being made to the order of item recovery from the so-named information structure. By giving the name 'list' to the data structure which works on the basis of a 'last-in-first-out' discipline, perhaps the LISP language designers meant 'list-as-stack', as opposed to 'list-as-queue' where the discipline would be the opposite 'first-in-first-out'.

7.4. Theory Presentations and Type Presentations

7.4.1. The λ-theory of Prof. Barendregt and the Π-type of λ-terms

A modern 'language' designed to express mathematical constructions, the λ-Calculus has been developed as an *operational* theory of functions. To use Prof. Barendregt's own words, 'the lambda calculus is a type free theory about functions as *rules*, rather than as graphs' (Barendregt 1981). Rules are used to capture the notion of function in the 'old' sense, as opposed to what is usually referred to as the 'modern' view of 'functions as graphs'.

A λ-theory, as defined by Prof. H. Barendregt in his mathematical treatise entitled *The Lambda Calculus. Its Syntax and Semantics*, is presented in the traditional axiomatic style, where the 'syntactic' definitions of terms is given as:

2.1.1. DEFINITION. (i) *Lambda terms* are words over the following alphabet:

$v_0, v_1, \ldots$	variables,
λ	abstractor,
$(\ ,\ )$	parentheses.

(ii) The set of λ-terms is defined inductively as follows:

(1) $x \in \Lambda$;

(2) $M \in \Lambda \Longrightarrow (\lambda x.M) \in \Lambda$;

(3) $M, N \in \Lambda \Longrightarrow (MN) \in \Lambda$;

where x in (1) or (2) is an arbitrary variable.

(Barendregt 1981, p. 22.)

Prof. Barendregt's presentation of a λ-theory follows the 'syntactic' definition of terms, and is given in the axiomatic style as:

2.1.4. DEFINITION. The theory λ has as formulas

$M = N$

where $M, N \in \Lambda$ and is axiomatized by the following axioms and rules:

(I)	$(\lambda x.M)N = M[x := N]$,	(β-conversion);
(II.1)	$M = M$;	
(II.2)	$M = N \Longrightarrow N = M$;	
(II.3)	$M = N, N = L \Longrightarrow M = L$;	
(II.4)	$M = N \Longrightarrow MZ = NZ$;	
(II.5)	$M = N \Longrightarrow ZM = ZN$;	
(II.6)	$M = N \Longrightarrow \lambda x.M = \lambda x.N$,	(rule ξ).

(Barendregt 1981, p. 23.)

As I have highlighted in the introductory sections, the presentation of a theory as a type in my type-theoretic framework brings together the 'syntactic' definition of

terms as well as the 'semantics'. The *introduction* rules indicate how to construct the elements of the type, corresponding approximately to the usual syntactic definition of terms, as in Prof. Barendregt's presentation shown above.

The *reduction* rules, in their turn, demonstrate what is essential in the semantics by showing how to *destroy* the terms contained in the type, or, in more appropriate terminology, they indicate how to *use* the elements of the type.

The Π-type of λ-terms, according to the semantics of use of *'Meaning-As-USE' Type Theory*, is presented as:

DEFINITION 7.4.1. Let a Π type be a type defined as:

Π-*formation*

$$\frac{\mathbf{A}\ \textit{type} \quad \mathbf{B}(x)\ \textit{type}\ [x \in \mathbf{A}]}{(\Pi x \in \mathbf{A})\mathbf{B}(x)\ \textit{type}} \qquad \frac{\mathbf{A} = \mathbf{C} \quad \mathbf{B}(x) = \mathbf{D}(x)\ [x \in \mathbf{A}]}{(\Pi x \in \mathbf{A})\mathbf{B}(x) = (\Pi x \in \mathbf{C})\mathbf{D}(x)}$$

Π-*introduction*

$$\frac{b(x) \in \mathbf{B}(x)\ [x \in \mathbf{A}]}{(\lambda x)b(x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)} \qquad \frac{b(x) = d(x) \in \mathbf{B}(x)\ [x \in \mathbf{A}]}{(\lambda x)b(x) = (\lambda x)d(x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

Π-*reduction*

$$\frac{a \in \mathbf{A} \qquad b(x) \in \mathbf{B}(x)\ [x \in \mathbf{A}]}{\mathtt{APPLY}((\lambda x)b(x), a) = b(a/x) \in \mathbf{B}(a)}$$

Π-*induction*

$$\frac{c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{(\lambda x)\mathtt{APPLY}(c, x) = c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

□

Comparing the λ-theory presentation of Prof. Barendregt with our presentation of the Π-type of λ-terms, one can make the following parallels:

Axiom (I) (β-conversion) bears a resemblance to the rule of Π-*reduction*.

Axioms (II.1), (II.2) and (II.3) seem to be particular cases of general equality rules (*reflexivity*, *symmetry*, *transitivity*), which enter a type-theoretic framework such as my *'Meaning-As-USE' Type Theory* at a general level for all types, not just for Π-types. In Prof. Barendregt's case, such rules are repeated for every theory presentation, e.g., the theory **CL** of Combinators, whose syntax is given in DEFINITION 7.1.1., p. 151, and semantics (axioms, rules) in DEFINITION 7.1.2., p. 152.

Axioms (II.4) and (II.5) appear to be variants of one of the rules of Π-*elimination*:

Π-*elimination*

$$\frac{a = b \in \mathbf{A} \quad c = d \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{APPLY}(c, a) = \mathtt{APPLY}(d, b) \in \mathbf{B}(a)}$$

which can be shown to be mere consequences of the definition of Π-types (as in DEFINITION 4.1. above) similarly to what has been done in previously. That is, firstly, to show that one can make the *judgement* '$\mathtt{APPLY}(c, a) \in \mathbf{B}(a)$' from the *judgements* '$a \in \mathbf{A}$' and '$c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)$', one has to show the following derivation tree:

$$\dfrac{\dfrac{a \in \mathbf{A} \qquad \dfrac{\dfrac{c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{\lambda x.\mathtt{APPLY}(c, x) = c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}\,(1)}{\lambda x.\mathtt{APPLY}(c, x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}\,(2)}{\mathtt{APPLY}(\lambda x.\mathtt{APPLY}(c, x), a) = \mathtt{APPLY}(c, a) \in \mathbf{B}(a)}\,(3)}{\mathtt{APPLY}(c, a) \in \mathbf{B}(a)}\,(4)$$

where

(1) Π-*induction*

(2) $= -\mathcal{L}$

(3) Π-*reduction*

$(4) = -\mathcal{R}$

Now, with the help of two general substitution rules available in *Intuitionistic Type Theory* as well as in my *'Meaning-As-USE' Type Theory*:

$$\frac{a \in \mathbf{A} \qquad \begin{array}{c}[x \in \mathbf{A}] \\ b(x) = d(x) \in \mathbf{B}(x)\end{array}}{b(a) = d(a) \in \mathbf{B}(a)}$$

$$\frac{a = c \in \mathbf{A} \qquad \begin{array}{c}[x \in \mathbf{A}] \\ b(x) \in \mathbf{B}(x)\end{array}}{b(a) = b(c) \in \mathbf{B}(a)}$$

it is easy to obtain the counterparts of II.4 as:

$$\frac{z \in \mathbf{A} \qquad \begin{array}{c}[x \in \mathbf{A}] \\ m(x) = n(x) \in \mathbf{B}(x)\end{array}}{\mathtt{APPLY}(m, z) = \mathtt{APPLY}(n, z) \in \mathbf{B}(z)}$$

and II.5 as:

$$\frac{m = n \in \mathbf{A} \qquad \begin{array}{c}[x \in \mathbf{A}] \\ z(x) \in \mathbf{B}(x)\end{array}}{\mathtt{APPLY}(z, m) = \mathtt{APPLY}(z, n) \in \mathbf{B}(m)}$$

Axiom (II.6) (rule ξ) finds in the second **Π**-*introduction* rule its counterpart in our presentation of **Π**-types.

By making such a parallel between an axiomatic and a 'proof-theoretic' presentation of a simple theory, the intention here is to show how such a natural deduction-based presentation system as my reformulated *Type Theory*, which has a more coherent semantics and still incorporates the identification of propositions with types, can be useful in providing appropriate definition/presentation tools. The focus of attention is placed rather conveniently on a semantics based on the notion of *utility* and on what seems to be the most interesting aspect of a theory presentation:

the *judgements* one can make about the *type/theory* presentation, instead of on the values of expressions.

7.4.2. Logics as Theories

With the help of the identification of *propositions* with *types* one can even provide a reasonable framework to present logics as theories, not via axioms but via natural deduction-like proof rules. Bearing in mind that by defining/presenting a logical connective one is defining the *proposition* of which it is the major connective, and that theories can be presented as types, it seems reasonable to think of a logic presentation as being the presentation of as many theories/types as the number of different connectives the logic might have. For instance, a definition of the logical connective of implication would be something like:

→-formation

$$\frac{\mathbf{A}\ \mathit{type} \quad \mathbf{B}(x)\ \mathit{type}\ [x \in \mathbf{A}]}{\mathbf{A} \rightarrow \mathbf{B}\ \mathit{type}} \qquad \frac{\mathbf{A} = \mathbf{C} \quad \mathbf{B}(x) = \mathbf{D}(x)\ [x \in \mathbf{A}]}{\mathbf{A} \rightarrow \mathbf{B} = \mathbf{C} \rightarrow \mathbf{D}}$$

→-introduction

$$\frac{b(x) \in \mathbf{B}(x)\ [x \in \mathbf{A}]}{(\lambda x)b(x) \in \mathbf{A} \rightarrow \mathbf{B}} \qquad \frac{b(x) = d(x) \in \mathbf{B}(x)\ [x \in \mathbf{A}]}{(\lambda x)b(x) = (\lambda x)d(x) \in \mathbf{A} \rightarrow \mathbf{B}}$$

→-reduction

$$\frac{a \in \mathbf{A} \quad b(x) \in \mathbf{B}(x)\ [x \in \mathbf{A}]}{\mathtt{APPLY}((\lambda x)b(x), a) = b(a/x) \in \mathbf{B}(a)}$$

→-induction

$$\frac{c \in \mathbf{A} \rightarrow \mathbf{B}}{(\lambda x)\mathtt{APPLY}(c, x) = c \in \mathbf{A} \rightarrow \mathbf{B}}$$

□

7.5. Concluding Remarks

By presenting a rather *operations*-oriented point of view on the issues concerning theories of abstract data type specification, whenever appropriate attempting to compare it with other more usual *denotations*-based approaches, I hope to have motivated the importance of adopting *interpretations*-free explanations of meaning in frameworks for software development.

Regarding other points of view based on *operational* semantics, it is worth making a few remarks concerning the differences, for example, between my own view on the rôle and the semantics of formal systems like my *'Meaning-As-USE' Type Theory* and the point of view taken by Prof. Martin-Löf and his followers.

The semantics of Gentzen adopted by Prof. Martin-Löf seems to make it difficult to justify a number of concepts, especially those related to theories of abstract data type specification, within *Intuitionistic Type Theory*. The definition of **Lists**, for example, is given in Martin-Löf 1984 as:

We can follow the same pattern used to define natural numbers to introduce other inductively defined sets. We see here the example of lists.

List-*formation*

$$\frac{\mathbf{A}\ \mathit{type}}{\mathbf{List(A)}\ \mathit{type}}$$

where the intuitive explanation is: **List(A)** is the set of list of elements of the set **A** (finite sequences of elements of **A**).

List-*introduction*

$$\mathtt{nil} \in \mathbf{List(A)} \qquad \frac{a \in \mathbf{A} \quad b \in \mathbf{List(A)}}{\mathtt{cons}(a,b) \in \mathbf{List(A)}}$$

where we may also use the notation $() \equiv \mathtt{nil}$.

(Martin-Löf 1984, p. 77.)

There is no reference to the usual 'destructors' 'HEAD' and 'TAIL', and the *elimination* rules for the type **List** are framed using a finite induction operator 'LISTREC' in a way similar to the finite induction operator for Natural Numbers 'R', given in his presentation of the type of Natural Numbers. As a matter of fact, as we have seen from previous sections, such induction operators can be obtained as particular instantiations of the generalised induction operator 'Fix' and should not take part in the definition of the inductive type **List**.

Confronting such an example with, say, the rules defining the type of **queue** of elements of **A**, we can easily see that it does not seem intuitively appealing to think of the meaning of a particular type/proposition as something given just by the rules indicating how to construct canonical elements (*introduction* rules). The 'semantics' (of *use*) of the particular type is only made explicit when the rules of 'destruction' of elements of the type (the rules of *reduction*, which show the *consequences*, or the *utility*, of the elements of the type) are laid down. The rules of *formation* and *introduction* for a type **Queue(A)**, corresponding to the grammar of type- and value-expressions respectively, in Prof. Martin-Löf's *Intuitionistic Type Theory* would not seem to be able to make the distinction between **lists** and **queues**, because the rules of 'construction' of canonical elements would be identical apart from the names of the operations ('**emptyqueue**' as opposed to 'nil', and 'addtoqueue' as opposed to 'cons'):

Queue(A)-*formation*

$$\frac{\mathbf{A}\ \mathit{type}}{\mathbf{Queue(A)}\ \mathit{type}}$$

Queue(A)-*introduction*

$$\mathtt{emptyqueue} \in \mathbf{Queue(A)} \qquad \frac{a \in \mathbf{A} \quad b \in \mathbf{Queue(A)}}{\mathtt{addtoqueue}(a, b) \in \mathbf{Queue(A)}}$$

When the rules of 'destruction' (rules of *reduction*) for both types are laid down, one then sees the sharp difference in the meaning (semantics, *use*) of the two types: for the case of **lists**, the 'destruction' is made by removing the last element inserted, whereas in the case of **queues**, the first element inserted is the one to be removed by the 'destructors'. That is perhaps what Guttag meant when saying that 'the distinguishing characteristic of a queue is that it is a first in first out storage device' and 'a good axiomatic definition [...] must therefore assert that and only that characteristic' (Guttag 1977, p. 398), after explaining why a mere explanation of how to form the canonical elements of the type 'queue' would not suffice to uniquely characterise it:

(...) One could, for example, define a type Queue (of Items) with the operations

NEW:	→ Queue
ADD:	Queue × Item → Queue
FRONT:	Queue → Item
REMOVE:	Queue → Queue
IS-EMPTY?:	Queue → Boolean

Unfortunately, however, short of supplying a representation, the only mechanism for denoting what these operations "mean" is a judicious choice of names. Except for intuitions about the meaning of such words as Queue and FRONT, the operations might just as well be defining type Stack as type Queue. (...)'

(Guttag 1977, p. 397.)

(The emphasis is mine.)

By defending the use of an axiomatisation in order to make precise which type is being defined/presented, one is acknowledging the insufficiency of any explanation

of the form of canonical elements the type might have. The axiomatisation is then supposed to settle the quest for a precise characterisation of the type by showing what kind of operations can be done with the elements of the type. The notion of type based on canonical values, such as the one given by the semantics of *Intuitionistic Type Theory*, is clearly incompatible with the concept of abstract data type in the tradition of Guttag.

Obviously, in his lecture notes published in Martin-Löf 1984, Prof. Martin-Löf is hardly concerned with the application of his Type Theory to theories of specification of abstract data types. The lecture notes seem to have been produced with the purpose of explaining the foundational principles and some important formal details of the framework. An example of such an application of *Intuitionistic Type Theory* as formulated by Prof. Martin-Löf's followers at Chalmers University of Technology, Göteborg, Sweden, is clearly seen as less convincing than a more intuitively appealing alternative such as the 'semantics of use' of my *'Meaning-As-USE' Type Theory*. In a rather brief section on specification of abstract data types, at the end of a draft book (perhaps provisionally) entitled '*An Introduction to Martin-Löf's Theory of Types*', it reads:

17.5 Specification of abstract data types

A simple example of an abstract data type specification is a specification of a stack. In type theory, the specification is the type:

$$
\begin{array}{l}
(\Sigma StackN \in \mathbf{U}) \\
\quad (\Sigma empty \in StackN) \\
\quad\quad (\Sigma push \in N \times StackN \rightarrow StackN) \\
\quad\quad\quad (\Sigma pop \in StackN \rightarrow StackN) \\
\quad\quad\quad\quad (\Sigma top \in StackN \rightarrow N) \\
\quad\quad\quad\quad\quad (\Pi t \in StackN)(\Pi n \in N) \\
\quad\quad\quad\quad\quad\quad ([pop[[empty]]=_{StackN} empty] \times \\
\quad\quad\quad\quad\quad [pop[[push[[< n, t >]]\ ]]=_{StackN} t] \times \\
\quad\quad\quad\quad\quad\quad [top[[empty]]=_{N} 0] \times \\
\quad\quad\quad\quad\quad [top[[push[[< n, t >]]\ ]]=_{N} n])
\end{array}
$$

where we have used the abbreviation

$$p[[a]] \equiv \mathbf{apply}(p, a)$$

We have also used the convention of confusing a type with its coding in **U**, all but the first occurrence of *StackN* stands for the type **Type**(*StackN*).

(Nordström, Petersson & Smith 1986, pp. 104–5.)

As a brief explanation of (my own understanding of) their intention, one should look at the specification given above as an attempt to say 'formally' the following: where 'Σ' represents existential quantification, and 'Π' codes the universal quantifier, there is a type *StackN* in the Universe of types '**U**', there is a constant *empty* in the type, there are functions *push*, *pop* and *top* with certain arities, such that for all t in *StackN* and for all n in N the 'equations' verify.

Having no apparent motivation as to why the use of such a 'complex' notation when specifying an abstract data type, their point of view on the utility of *Intuitionistic Type Theory* for a better understanding of theories of specification of abstract data types makes the whole body of new concepts and new notation brought in by such a formal system look as just another symbolic notation, perhaps even more unnecessarily complicated than the usual ones, for coding up exactly the same information (see corresponding algebraic specification in Ehrig & Mahr 1985, p. 52.). The force of a truly convincing alternative to traditional approaches to typical problems when dealing with theories of specification seems to be missing. The apparently intuitively appealing 'computational semantics' does not seem to convince. As a matter of fact, it does not even seem to be properly formalised by *ITT*, given that the explanation of the meaning of logical constants involves recourse to verbal (informal) explanation: the *equality* (*reduction*) rules of *ITT* are said to become evident from a certain informal explanation of the *elimination* operators. On this basis, it is fair to say that from such a point of view, *Intuitionistic Type Theory* looks like a mere 'coding'. This is not necessarily the case for all type-theoretic frameworks, because the drawbacks are not in *Type Theory* itself, but rather in the

semantical approach taken by Prof. P. Martin-Löf. A type-theoretic presentation system can be made more coherent and more convincing than *Intuitionistic Type Theory* when the semantics is 'enriched' by taking the *reduction* rules as semantical rules.

Moreover, regarding higher-order objects, it is well-known that *Intuitionistic Type Theory* is rather restrictive. I claim that the less intuitive Gentzen's proof-theoretic semantics adopted by Prof. Martin-Löf has even prevented suitable explanations of (potentially) higher-order operators such as **W** and **U** to be given in a reasonably convincing manner. *Denotations*-based approaches to the semantics of abstract data type specifications such as, e.g., Prof. Scott's *Theory of Domains* (Scott 1976, 1981) and the Lehmann-Smyth (Lehmann & Smyth 1981) categorical constructions (Cf. the idea of using a 'Fix'-like operator for the definition of infinite types such as Stacks and Natural Numbers with their pattern of definition of such infinite types: List$A \cong A +$ Stack(List$A$), p. 122, and $X \cong \mathbf{1} + X$, p. 126, respectively, of Lehmann & Smyth 1981) manage to motivate better the use of higher-order notions than Prof. Martin-Löf's *Intuitionistic Type Theory*. By taking a rather different semantical point of view, but staying within the *interpretations*-free semantical paradigm, I have introduced a higher-order type forming operator which I call 'Fix', and which seems to make the explanations of *infinite* objects via my *'Meaning-As-USE' Type Theory* more intuitively appealing.

8 *Epilogue*

Can proper foundations ever be formulated for the notion of *computation* in purely logico-linguistic terms? What is there in common amongst all models of computability, with respect to the relation rules-meaning?

I believe that to study the foundations of computability by studying the foundations of language and the nature of concepts like *meaning* and *understanding*, can be extremely fruitful and interesting. It is worth recalling one of L. Wittgenstein's remarks concerning the attitude one should take when analysing human knowledge which is to keep reminding oneself of the crucial rôle of language in the development and the transmission of knowledge:

> Wir müssen die ganze Sprache durchpflugen.

which has been translated as

> We must plough over the whole of language.
>
> (*RFGB*, p. 7^{e}.)

8.1. Evaluation of the Results

I have not presented here any new algorithm, a novel computation technique, a substantially better proof of a fundamental theorem, or anything of the sort. Neither have I devised a new decision procedure for a particular logic, which would perhaps be said to characterise a substantial contribution to a better understanding of the logical foundations of the 'science of computing'. Nevertheless, it has been my purpose to motivate the need to look at the problems with which the present report is concerned from a certain angle that I believe may offer new insights into the nature of such 'science'. This reflects my own point of view on the questions concerning the problem of laying down the foundations of a reasonable theory dealing with concepts like *computation* and *constructive reasoning*, based solely on the laws of logic. It is faithful to my own attitude towards such foundational questions, which is to believe that the most important steps are those leading to *elucidations*, where *solutions* emerge out of the (constant) activity of clarification of thoughts. In the course of my investigations I have felt that a better understanding of the nature of *meaning* could help in clarifying a number of issues concerning the use of the logical tools available, such as proof-theoretic presentation systems. To have attempted to bring Wittgenstein's ideas and spirit into the discussion of the foundations of the science of computing is an initiative which followed naturally from my sympathy with (my understanding of) the philosopher's views as in:

> Philosophy aims at the logical clarification of thoughts.
>
> Philosophy is not a body of doctrine but an activity.
>
> A philosophical work consists essentially of elucidations.
>
> Philosophy does not result in 'philosophical propositions', but rather in the clarification of propositions.
>
> (*Tractatus*, §4.112, p. 25.)

and in:

> It was true that our considerations could not be scientific ones. It was not of any possible interest to us to find out empirically 'that, contrary to our preconceived ideas, it is possible to think such-and-such'—whatever that may mean. (The conception of thought as a gaseous medium.) And we may not advance any kind of theory. There must not be anything hypothetical in our

considerations. We must do away with all *explanation*, and description alone must take place. And this description gets its light, that is to say its purpose, from the philosophical problems. These are, of course, not empirical problems; they are solved, rather, by looking into the workings of our language, and that in such a way as to make us recognize those workings: *in despite of* an urge to misunderstand them. The problems are solved, not by giving new information, but by arranging what we have always known. Philosophy is a battle against the bewitchment of our intelligence by means of language.

(*Investigations*, Part I, §109, p. 47^e.)

I believe his investigations on the nature and the rôle of language in knowledge acquisition and representation have served as a rather solid basis for a better understanding of foundational issues of disciplines such as (formal) logic, mathematics, as well as 'the science of computing'. They have certainly elucidated many key points regarding a crucial concept of any rational discourse: the very concept of *meaning*.

Concerning the *elucidations* offered by the investigations reported here, it seems fair to say that the following achievements constitute coherent clarifications of the logical basis for a notion of *computation*:

8.1.1. Definitions as Presentations

A look at Frege's notions of definition has demonstrated that in logic, mathematics, as well as in computing science, one needs to make *defintions* which are not *abbreviations*. This is particularly true with computing science where one is usually involved in coding concepts with pre-conceived meanings into a certain 'computable' language. Such a view of the rôle of Frege's *analytic* definition has led to a better account of definitions as *presentations* rather than as *abbreviations*. It has also uncovered Frege's awareness of the need for a logical account of the notion of *implementation*.

8.1.2. Weaknesses of Gentzen-style semantics

A reformulation of semantics via inference rules has helped to uncover weaknesses of Gentzen-style semantics.

It seems fair to say that Wittgenstein's (later) philosophical views on semantic matters (*meaning-is-use*) constitute a great source of inspiration for more convincing logical foundations of the theory of *computation*, where concepts like 'rules' and 'meaning' are clearly seen to be particularly close. From this philosophical point of view I have performed a revision on Gentzen-style proof-theoretic semantics for the mathematical language (which in itself constitutes an attempt to realise the connection rules-meaning), resulting in a particular operational semantics which I suggestively call 'semantics of use'. I have essentially tried to rework the analysis of the nature of *use* in logical terms, initiated by Prof. Michael Dummett and Prof. Dag Prawitz in their research programme on an intuitionistic (linguistic) account of meaning for the language of mathematics, and followed somewhat independently by Prof. Per Martin-Löf's attempt to clarify the syntax and the semantics of *constructive* mathematics. An attempt to reach a better understanding of the *Constuctivist* Manifesto (Chapter 1 of Bishop 1967) put forward by Errett Bishop and a few others has also served as one of my underlying motivations.

8.1.3. Verification and Consequences

Still concerning proof-theoretic semantics, I have offered a coherent alternative perspective on the connections between *verification* and *meaning*, again as part of my understanding of Wittgenstein's teachings.

> Asking whether and how a proposition can be verified is only a particular way of asking "How d'you mean?" The answer is a contribution to the grammar of the proposition.
>
> (*Investigations*, Part I, §353, p. 112^e.)

8.1.4. Wittgenstein, Consequences and Meaning

An important aspect of the semantical studies reported here is that it offers a new perspective on the evolution of Wittgenstein's *meaning-is-use* semantical paradigm

which shows a certain uniformity in his intuition, regardless of phases (early, later, etc.), concerning the connections between 'showing the *consequences* of a proposition' and 'giving the meaning to a proposition'. Since 1912 when he writes to Russell about his intuition on the rôle of logical *consequence* in determining the meaning of an assertion, until circa 1949 when he makes such an explicit reference to the connection between the questions 'what meaning does the statement have?' and 'what consequences does the statement have?', his writings bear witness to the persistence of an intuition which has significantly improved our understanding of key concepts such as meaning, understanding, etc..

8.1.5. Semantics with no reference to (numerical) values

By taking *use/utility* rather than (canonical, numerical) *values* as being the key aspect of semantics, a semantical reformulation of Prof. P. Martin-Löf's *Intuitionistic Type Theory* has been performed.

A direct comparison between my *Semantics of Use* and *Game-Theoretical Semantics* has been made (Chapter 5), where the point was made that both give *meaning* to logical signs by showing the *consequences* of the corresponding proposition. By explaining the *consequences* one is allowed to draw from a proposition one is demonstrating its *meaning*, and I have argued that the device of *normalisation* represents such *explanation of meaning* for the case of the mathematical language.

> "The meaning of a word is what is explained by the explanation of the meaning". I.e.: if you want to understand the use of the word "meaning", look for what are called "explanations of meaning".
>
> (*Investigations*, Part I, §560, p. 149[e].)

8.1.6. The Rôle of Proof-Rules in Type Theory

I have also proved the derivability of *elimination* rules such as those presented in *Intuitionistic Type Theory*, from what I believe is to be taken as the set of rules for definition-as-presentation of types in *'Meaning-As-USE' Type Theory.* This has been

done as a formal counterpart to the informal argument concerning the irrelevance of Prof. Martin-Löf's *elimination* rules of *Intuitionistic Type THeory* with respect to the presentation of types in a type-theoretic framework. The proof of derivability of *elimination* rules from the type presentation via *formation*, *introduction*, *reduction* and *induction* has also shown that the *reduction* rules do play a fundamental rôle in the definition (as presentation) of a type, and so the 'verbal' explanation of how the *eliminatory* operations work, as given by Prof. Martin-Löf's account of proof rules in *Type Theory*, is completely irrelevant to the understanding of the meaning of the type. For the case of inductive types such as the type of natural numbers, the formal derivation of *elimination* rules from the type presentation also shows that undesirable *judgements* such as '$\mathsf{PRED}(0) \in \mathcal{N}$' can never be made from the presentation of the type.

8.1.7. Computer Programming and Infinity

The development of a higher-order type-of-types Fix operator has shown how the notion of *infinity* is dealt with in *'Meaning-As-USE' Type Theory.* With such a higher-order inductive operator one can easily see the general pattern to obtain recursion operators for particular inductive types, such as natural numbers, lists, stacks, binary trees, etc., instead of having a specific recursion operator being given in the definition (presentation) of the type such as is done in *Intuitionistic Type Theory*. Furthermore, one can easily see that finite chunks of infinite sets come as particular cases of the definition of the infinite case via the Fix operator. This shows that one does not need to have types such as $\mathbf{N}_0$, $\mathbf{N}_1$, $\mathbf{N}_2$, . . ., $\mathbf{N}_m$ defined as primitive types, such as is done in the chapter on Finite sets of Prof. Martin-Löf's lecture notes (Padova) on *Intuitionistic Type Theory* (Martin-Löf 1984).

8.1.8. Abstract Data Types

The application of *'Meaning-As-USE' Type Theory* to the development of a new approach to theories of Abstract Data Types specification has proved useful in many

respects. As a matter of fact, it has been a most illuminating experience to use the lessons learned from many years of continuous studies attempting to make precise the notion of data type, namely the investigation of the theory of abstract data types, to reformulate what is certainly a seminal work on the foundations of logic based on the concepts of *types* and *judgements*: *Type Theory*. The usual mathematical notion of set, which was felt sufficiently satisfactory for most developments in the mathematics of this second half of the century, proved inadequate to support the notion of data type as understood by computer scientists. Instead of defining a type by what it contained, it was felt that a data type definition had to convey the information demonstrating how the elements of the type were to be used, computed, etc.. A whole body of ideas developed mainly in the seventies aiming at gaining a better understanding of the concept of data types began to show that data type definitions were worth studying as a theory in itself. With the help of algebraic techniques, the notion of data type as defined by what can be done with its elements proved a convenient alternative to the mathematical notion of set as defined by what the set contains. The semantical concern, however, was still directed towards 'meaning as (numerical) value', and many difficulties were encountered to explain the meaning of oddly formed 'terms' such as the top of an empty stack.

Type Theory, an attempt to clarify the syntax and semantics of intuitionistic mathematics via *types* and *judgements*, developed almost simultaneously with the algebraic approaches to data types, came with a most interesting semantical innovation, which was the shift on the focus of attention from (numerical) values to *judgements*. Prior to the question of what the value of an expression was, which nevertheless still remained but in a different level, there was a question of what kinds of *judgements* could be made from a type definition (*presentation*). Such an interesting innovation opened new horizons for a better understanding of what is to be taken as relevant from a type presentation. Many of the difficulties in giving a 'meaning' (as numerical value) to certain oddly formed terms would then be overcome. However, at the level of type presentations, *Type Theory* proved a step back to the mathematical principle that a type (set) is to be defined by what the type

(set) contains.

By presenting a reformulated *Type Theory* where the good lessons from both sides come together in a more coherent framework to express constructions, I have demonstrated that a number of issues concerning data types can be better understood if one accepts both that a type is to be presented by what can be done with its elements, and that the question one should ask about a type presentation is what sort of *judgements* one can make from that particular presentation rather than what values the expressions have.

And the semantical approach opened a new perspective on how to solve a number of typical problems dealt with in theories of specification of data types. Most of those typical problems such as, e.g., error handling, the rôle of hidden functions, parameterisation, a formal account of implementation, the completeness of axioms, etc., were given reasonably convincing 'solutions'. The identification of *theory* presentations with *type* presentations allowed for a useful comparison between a type-theoretic presentation of a Π-type of λ-terms with Prof. H. Barendregt's axiomatic presentation of a λ-theory (in his *The Lambda Calculus. Its Syntax and Semantics*, Barendregt 1981.). From the parallel drawn between the two presentations of a theory has emerged many of the advantages of using a type-theoretic framework as a presentation 'meta-logic' as an alternative to the more usual axiomatic framework. Among the advantages one can mention a greater degree of generality especially concerning general equality rules, and a better account for an explanation of the rôle of certain rules such as e.g. the η and ξ rules of the λ-calculus, for the case of the example mentioned. The semantical approach suggesting the approach of looking at the meaning of types as inseparable from their *utility*, which in *'Meaning-As-USE' Type Theory* is shown by the effect of the DESTRUCTORS on the constructors of the type, has also allowed for an identification of two data structures which have been unnecessarily distinguished: cons/CAR/CDR-lists and push/TOP/POP-stacks.

8.1.9. The Resulting Formal System

The result is a reformulation of Gentzen-style proof-theoretic semantics, apparently

closer to Wittgenstein's *meaning-is-use* semantical paradigm than *assertability conditions*-based semantics, because it does take into account the second fundamental aspect of Wittgenstein's later account of propositions which he calls 'the use of the sign in the language-games', and others (e.g., Prof. Dummett) sometimes call 'the commitments one makes by asserting the proposition' (Prof. Dummett's 'falsificationist' theory of meaning 'links the content of an assertion with the commitment that a speaker undertakes in making that assertion', at p. 126 of Dummett 1976.), or even (e.g., Prof. Kripke) 'the rôle, and the utility, in our lives of our practice of asserting (or denying) [the proposition]' (pp. 72–3 of Kripke 1982.). As an illustration of the force of this 'semantics of use' I have proposed a reformulation of a particular formalisation of constructive mathematics, Prof. Per Martin-Löf's *Intuitionistic Type Theory*, which I say can be seen more convincingly as a meaning theory for the mathematical language provided it carries a better semantical account than Gentzen's traditional proof-theoretic semantics which considers the *assertability conditions* alone as the only aspect of the meaning of a proposition. The resulting formal system, described in the *Appendix*, is then intentionally called *'Meaning-As-USE' Type Theory.*

8.2. Perspectives on Further Work

There seems to be enough room for an investigation into how to provide the 'semantics of use' for Modal Connectives via rules of higher-order like the Fix operator. Observe that the iterative structure underlying the definition of the Fix operator seems to reflect the structure of a collection of ordered 'worlds'.

An improvement of the Fix operator would be most useful, and this could be done perhaps by using a more generalised formulation without the need of +.

Further examples of data type definitions using *MAUSE* and proofs of correctness of *implementations* for more complicated structures would be of much interest.

Appendix

Here I present in reasonably concrete terms a result of my semantical studies of the mathematical language by exhibiting a theory of meaning based on *types* and *judgements*, whose semantic theory can be said to comply with L. Wittgenstein's paradigm *meaning-is-use* and F. Mauthner's teachings on *meaning-is-no-entity.*

In spite of having its notation and most of its primitive concepts heavily based on Prof. P. Martin-Löf's *Intuitionistic Type Theory*, my *'Meaning-As-USE' Type Theory* relies on a radically distinct semantic theory (in the sense of Dummett 1982, p. 60.). Instead of saying that meaning is a semantic (numerical) value, the semantic theory which supports my own *Type Theory* says that meaning is no 'object' or (numerical, canonical) value but is made explicit by certain rules of *use* such as the *normalisation* rules.

More than just a list of inference rules, my own *'Meaning-As-USE' Type Theory* should be seen as a general guideline prescribing how types, theories and logics can be presented in such a way that no appeal to *interpretations* or semantic *values* is necessary in order for one to grasp the meaning of each type, theory, logical connective, etc..

'Meaning-As-USE' Type Theory

Ruy J. G. B. de Queiroz

(November 1, 1989)

GENERAL RULES

Reflexivity

$$\frac{a \in \mathbf{A}}{a = a \in \mathbf{A}} \qquad \frac{\mathbf{A}\ type}{\mathbf{A} = \mathbf{A}}$$

Symmetry

$$\frac{a = b \in \mathbf{A}}{b = a \in \mathbf{A}} \qquad \frac{\mathbf{A} = \mathbf{B}}{\mathbf{B} = \mathbf{A}}$$

Transitivity

$$\frac{a = b \in \mathbf{A} \qquad b = c \in \mathbf{A}}{a = c \in \mathbf{A}} \qquad \frac{\mathbf{A} = \mathbf{B} \qquad \mathbf{B} = \mathbf{C}}{\mathbf{A} = \mathbf{C}}$$

Equality Left and *Equality Right*

$$= \text{-}\mathcal{L} \quad \frac{a = b \in \mathbf{A}}{a \in \mathbf{A}} \qquad\qquad = \text{-}\mathcal{R} \quad \frac{a = b \in \mathbf{A}}{b \in \mathbf{A}}$$

Equality of Types

$$\frac{a \in \mathbf{A} \qquad \mathbf{A} = \mathbf{B}}{a \in \mathbf{B}} \qquad \frac{a = b \in \mathbf{A} \qquad \mathbf{A} = \mathbf{B}}{a = b \in \mathbf{B}}$$

Substitution

(1) From the *judgement* $\mathbf{B}(x)$ *type* $[x \in \mathbf{A}]$

$$\frac{a \in \mathbf{A} \qquad \begin{matrix}[x \in \mathbf{A}] \\ \mathbf{B}(x)\ type\end{matrix}}{\mathbf{B}(a)\ type} \qquad \frac{a = c \in \mathbf{A} \qquad \begin{matrix}[x \in \mathbf{A}] \\ \mathbf{B}(x)\ type\end{matrix}}{\mathbf{B}(a) = \mathbf{B}(c)}$$

(2) From the *judgement* $\mathbf{B}(x) = \mathbf{D}(x)$ $[x \in \mathbf{A}]$

$$\frac{a \in \mathbf{A} \qquad \begin{matrix}[x \in \mathbf{A}] \\ \mathbf{B}(x) = \mathbf{D}(x)\end{matrix}}{\mathbf{B}(a) = \mathbf{D}(a)} \qquad \frac{a = c \in \mathbf{A} \qquad \begin{matrix}[x \in \mathbf{A}] \\ \mathbf{B}(x) = \mathbf{D}(x)\end{matrix}}{\mathbf{B}(a) = \mathbf{D}(c)}$$

(3) From the *judgement* $b(x) \in \mathbf{B}(x)$ $[x \in \mathbf{A}]$

$$\frac{a \in \mathbf{A} \qquad \begin{matrix}[x \in \mathbf{A}] \\ b(x) \in \mathbf{B}(x)\end{matrix}}{b(a) \in \mathbf{B}(a)} \qquad \frac{a = c \in \mathbf{A} \qquad \begin{matrix}[x \in \mathbf{A}] \\ b(x) \in \mathbf{B}(x)\end{matrix}}{b(a) = b(c) \in \mathbf{B}(a)}$$

(4) From the *judgement* $b(x) = d(x) \in \mathbf{B}(x)$ $[x \in \mathbf{A}]$

$$\frac{a \in \mathbf{A} \qquad \begin{matrix}[x \in \mathbf{A}] \\ b(x) = d(x) \in \mathbf{B}(x)\end{matrix}}{b(a) = d(a) \in \mathbf{B}(a)}$$

Assumption

$$[x \in \mathbf{A}]$$

Instead of Prof. Martin-Löf's usual rules for the definition of types/propositions:

formation

introduction

elimination

equality

we have the following rules, with corresponding purpose:

formation

type's *grammar*. They show how to form the 'canonical' types — type-expression.

introduction

elements' *grammar*. They demonstrate how to form the 'canonical' elements (`constructors`) — value-expressions.

reduction

elements' *use*. These rules demonstrate the effect of `DESTRUCTORS` on terms built up by means of the `constructors`.

induction

These rules are what one could call 'meta'-reasoning statements saying 'the type contains the elements given by the rules of *introduction* and only those, therefore the meaning (*use*, *utility*) of the type is only the one given by the *reduction* rules'.

CARTESIAN PRODUCT OF A FAMILY OF TYPES

Π-*formation*

$$\frac{\mathbf{A}\ \mathit{type} \quad \mathbf{B}(x)\ \mathit{type}\ [x \in \mathbf{A}]}{(\Pi x \in \mathbf{A})\mathbf{B}(x)\ \mathit{type}} \qquad \frac{\mathbf{A} = \mathbf{C} \quad \mathbf{B}(x) = \mathbf{D}(x)\ [x \in \mathbf{A}]}{(\Pi x \in \mathbf{A})\mathbf{B}(x) = (\Pi x \in \mathbf{C})\mathbf{D}(x)}$$

Π-*introduction*

$$\frac{b(x) \in \mathbf{B}(x)\ [x \in \mathbf{A}]}{(\lambda x)b(x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)} \qquad \frac{b(x) = d(x) \in \mathbf{B}(x)\ [x \in \mathbf{A}]}{(\lambda x)b(x) = (\lambda x)d(x) \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

Π-*reduction*

$$\frac{\begin{array}{c}[x \in \mathbf{A}] \\ a \in \mathbf{A} \qquad b(x) \in \mathbf{B}(x)\end{array}}{\mathtt{APPLY}((\lambda x)b(x), a) = b(a/x) \in \mathbf{B}(a)}$$

Π-*induction*

$$\frac{c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}{(\lambda x)\mathtt{APPLY}(c, x) = c \in (\Pi x \in \mathbf{A})\mathbf{B}(x)}$$

□

DISJOINT UNION OF A FAMILY OF TYPES

Σ-*formation*

$$\frac{\begin{array}{c}[x \in \mathbf{A}] \\ \mathbf{A}\ \textit{type} \qquad \mathbf{B}(x)\ \textit{type}\end{array}}{(\Sigma x \in \mathbf{A})\mathbf{B}(x)\ \textit{type}} \qquad \frac{\begin{array}{c}[x \in \mathbf{A}] \\ \mathbf{A} = \mathbf{C} \qquad \mathbf{B}(x) = \mathbf{D}(x)\end{array}}{(\Sigma x \in \mathbf{A})\mathbf{B}(x) = (\Sigma x \in \mathbf{C})\mathbf{D}(x)}$$

Σ-*introduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{B}(a)}{\mathtt{pair}(a, b) \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)} \qquad \frac{a = c \in \mathbf{A} \quad b = d \in \mathbf{B}(a)}{\mathtt{pair}(a, b) = \mathtt{pair}(c, d) \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}$$

Σ-*reduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{B}(a)}{\mathtt{FST}(\mathtt{pair}(a, b)) = a \in \mathbf{A}} \qquad \frac{a \in \mathbf{A} \quad b \in \mathbf{B}(a)}{\mathtt{SND}(\mathtt{pair}(a, b)) = b \in \mathbf{B}(a)}$$

Σ-*induction*

$$\frac{c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}{\mathtt{pair}(\mathtt{FST}(c), \mathtt{SND}(c)) = c \in (\Sigma x \in \mathbf{A})\mathbf{B}(x)}$$

□

DISJOINT UNION OF TWO TYPES

+-formation

$$\frac{\mathbf{A}\ \textit{type} \quad \mathbf{B}\ \textit{type}}{\mathbf{A}+\mathbf{B}\ \textit{type}} \qquad \frac{\mathbf{A}=\mathbf{C} \quad \mathbf{B}=\mathbf{D}}{\mathbf{A}+\mathbf{B}=\mathbf{C}+\mathbf{D}}$$

+-introduction

$$\frac{a \in \mathbf{A}}{\mathtt{inl}(a) \in \mathbf{A}+\mathbf{B}} \qquad \frac{a=c \in \mathbf{A}}{\mathtt{inl}(a)=\mathtt{inl}(c) \in \mathbf{A}+\mathbf{B}}$$

$$\frac{b \in \mathbf{B}}{\mathtt{inr}(b) \in \mathbf{A}+\mathbf{B}} \qquad \frac{b=d \in \mathbf{B}}{\mathtt{inr}(b)=\mathtt{inr}(d) \in \mathbf{A}+\mathbf{B}}$$

+-reduction

$$\frac{a \in \mathbf{A} \quad \begin{matrix}[x \in \mathbf{A}] \\ d \in \mathbf{C}(\mathtt{inl}(x))\end{matrix} \quad \begin{matrix}[y \in \mathbf{B}] \\ e \in \mathbf{C}(\mathtt{inr}(y))\end{matrix}}{\mathtt{WHEN}(\mathtt{inl}(a),d,e)=d(a) \in \mathbf{C}(\mathtt{inl}(a))}$$

$$\frac{b \in \mathbf{B} \quad \begin{matrix}[x \in \mathbf{A}] \\ d \in \mathbf{C}(\mathtt{inl}(x))\end{matrix} \quad \begin{matrix}[y \in \mathbf{B}] \\ e \in \mathbf{C}(\mathtt{inr}(y))\end{matrix}}{\mathtt{WHEN}(\mathtt{inr}(b),d,e)=e(b) \in \mathbf{C}(\mathtt{inr}(b))}$$

+-induction

$$\frac{c \in \mathbf{A}+\mathbf{B} \quad \begin{matrix}[a \in \mathbf{A}] \\ d \in \mathbf{C}(\mathtt{inl}(a))\end{matrix} \quad \begin{matrix}[b \in \mathbf{B}] \\ e \in \mathbf{C}(\mathtt{inr}(b))\end{matrix}}{\mathtt{inl}(\mathtt{WHEN}(c,d,e))=c \in \mathbf{A}+\mathbf{B}}$$

$$\frac{c \in \mathbf{A}+\mathbf{B} \quad \begin{matrix}[a \in \mathbf{A}] \\ d \in \mathbf{C}(\mathtt{inl}(a))\end{matrix} \quad \begin{matrix}[b \in \mathbf{B}] \\ e \in \mathbf{C}(\mathtt{inr}(b))\end{matrix}}{\mathtt{inr}(\mathtt{WHEN}(c,d,e))=c \in \mathbf{A}+\mathbf{B}}$$

□

FINITE TYPE

$\{0\}$ *type*

RULES FOR HIGHER-ORDER OBJECTS AND GENERALISED INDUCTION

Fix-*formation*

$$\frac{[\mathbf{X}\ \textit{type}]}{\nabla\ \textit{type} \quad \mathbf{B}(\mathbf{X})\ \textit{type}}{\mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})\ \textit{type of types}} \qquad \frac{[\mathbf{X}\ \textit{type}]}{\nabla_1 = \nabla_2 \quad \mathbf{B_1}(\mathbf{X}) = \mathbf{B_2}(\mathbf{X})}{\mathsf{Fix}\ \mathbf{X}.\nabla_1 + \mathbf{B_1}(\mathbf{X}) = \mathsf{Fix}\ \mathbf{X}.\nabla_2 + \mathbf{B_2}(\mathbf{X})}$$

Fix-*introduction*

$$\varnothing \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X}) \qquad \varnothing = \varnothing \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})$$

$$\frac{\begin{array}{c}[z \in \mathbf{X}] \\ g(z) \in \nabla + \mathbf{B}(\mathbf{X})\end{array}}{\mathsf{gen}(z.g(z)) \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

$$\frac{\begin{array}{c}[z \in \mathbf{X}] \\ g_1(z) = g_2(z) \in \nabla + \mathbf{B}(\mathbf{X})\end{array}}{\mathsf{gen}(z.g_1(z)) = \mathsf{gen}(z.g_2(z)) \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

$$\frac{\mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}{\nabla + \mathbf{B}(\mathbf{A}/\mathbf{X}) \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

$$\frac{\mathbf{A_1} = \mathbf{A_2} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}{\nabla + \mathbf{B}(\mathbf{A_1}/\mathbf{X}) = \nabla + \mathbf{B}(\mathbf{A_2}/\mathbf{X}) \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

Fix-*reduction*

At the level of elements

$$\frac{a \in \mathbf{A} \quad \mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X}) \quad \begin{matrix}[z \in \mathbf{X}] \\ g(z) \in \nabla + \mathbf{B}(\mathbf{X})\end{matrix}}{\mathtt{UNWIND}(\mathsf{gen}(z.g(z)), a) = g(a/z) \in \nabla + \mathbf{B}(\mathbf{A}/\mathbf{X})}$$

At the level of types

$$\frac{\mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}{\mathtt{PREDTYPE}(\nabla + \mathbf{B}(\mathbf{A}/\mathbf{X})) = \mathbf{A} \in \mathsf{Fix}\ \mathbf{X}.\nabla + \mathbf{B}(\mathbf{X})}$$

□

INDUCTIVE TYPES

Natural Numbers

$\mathcal{N}$-*formation*

$$\mathcal{N}\ \textit{type} \qquad \mathcal{N} = \mathcal{N}$$

$\mathcal{N}$-*introduction*

$$0 \in \mathcal{N} \qquad 0 = 0 \in \mathcal{N}$$

$$\frac{a \in \mathcal{N}}{\mathsf{succ}(a) \in \mathcal{N}} \qquad \frac{a = b \in \mathcal{N}}{\mathsf{succ}(a) = \mathsf{succ}(b) \in \mathcal{N}}$$

$\mathcal{N}$-*reduction*

$$\frac{a \in \mathcal{N}}{\mathtt{PRED}(\mathtt{succ}(a)) = a \in \mathcal{N}}$$

$\mathcal{N}$-induction

(instantiated from the Fix- rules of generalised induction.)

□

Parameterised Types

(Using Propositions (Types) as Parameters)

Examples

Lists

List(A)-*formation*

$$\frac{\mathbf{A}\ \mathit{type}}{\mathbf{List(A)}\ \mathit{type}} \qquad \frac{\mathbf{A} = \mathbf{B}}{\mathbf{List(A)} = \mathbf{List(B)}}$$

List(A)-*introduction*

$$\mathtt{nillist} \in \mathbf{List(A)} \qquad \mathtt{nillist} = \mathtt{nillist} \in \mathbf{List(A)}$$

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{List(A)}}{\mathtt{cons}(a,b) \in \mathbf{List(A)}} \qquad \frac{a = c \in \mathbf{A} \quad b = d \in \mathbf{List(A)}}{\mathtt{cons}(a,b) = \mathtt{cons}(c,d) \in \mathbf{List(A)}}$$

List(A)-*reduction*

$$\frac{a \in \mathbf{A} \quad b \in \mathbf{List(A)}}{\mathtt{HEAD}(\mathtt{cons}(a,b)) = a \in \mathbf{A}} \qquad \frac{a \in \mathbf{A} \quad b \in \mathbf{List(A)}}{\mathtt{TAIL}(\mathtt{cons}(a,b)) = b \in \mathbf{List(A)}}$$

List(A)-*induction*

(instantiated from the generalised induction rules.)

□

Binary Trees

BinTree(A)-*formation*

$$\frac{\mathbf{A}\ \mathit{type}}{\mathbf{BinTree(A)}\ \mathit{type}} \qquad \frac{\mathbf{A} = \mathbf{B}}{\mathbf{BinTree(A)} = \mathbf{BinTree(B)}}$$

BinTree(A)-*introduction*

$$\mathtt{niltree} \in \mathbf{BinTree(A)} \qquad \mathtt{niltree} = \mathtt{niltree} \in \mathbf{BinTree(A)}$$

$$\frac{a \in \mathbf{BinTree(A)} \quad b \in \mathbf{A} \quad c \in \mathbf{BinTree(A)}}{\mathtt{node}(a, b, c) \in \mathbf{BinTree(A)}}$$

$$\frac{a_1 = a_2 \in \mathbf{BinTree(A)} \quad b_1 = b_2 \in \mathbf{A} \quad c_1 = c_2 \in \mathbf{BinTree(A)}}{\mathtt{node}(a_1, b_1, c_1) = \mathtt{node}(a_2, b_2, c_2) \in \mathbf{BinTree(A)}}$$

BinTree(A)-*reduction*

$$\frac{a \in \mathbf{BinTree(A)} \quad b \in \mathbf{A} \quad c \in \mathbf{BinTree(A)}}{\mathtt{LABEL}(\mathtt{node}(a, b, c)) = b \in \mathbf{A}}$$

$$\frac{a \in \mathbf{BinTree(A)} \quad b \in \mathbf{A} \quad c \in \mathbf{BinTree(A)}}{\mathtt{LEFT}(\mathtt{node}(a, b, c)) = a \in \mathbf{BinTree(A)}}$$

$$\frac{a \in \mathbf{BinTree(A)} \quad b \in \mathbf{A} \quad c \in \mathbf{BinTree(A)}}{\mathtt{RIGHT}(\mathtt{node}(a, b, c)) = c \in \mathbf{BinTree(A)}}$$

BinTree(A)-*induction*

(instantiated from the general fixpoint rules for the 'Fix' operator.)

□

Bibliography

A

Aczel, Peter H. G.: 1978, 'The Type Theoretic Interpretation of Constructive Set Theory', in *Logic Colloquium '77*, Proceedings of the colloquium held in Wrocław, A. Macintyre, L. Pacholski & J. Paris (eds.), August 1977, series Studies in Logic and The Foundations of Mathematics, vol. 96, North-Holland, Amsterdam, x+311pp, pp. 55–66.

— : 1980, 'Frege Structures and the Notions of Proposition, Truth and Set', in *The Kleene Symposium*, Proceedings of the Symposium held in June 18–24, 1978, at Madison, Wisconsin, USA, J. Barwise, H.-J. Keisler & K. Kunen (eds.), series Studies in Logic and The Foundations of Mathematics, vol. 101, North-Holland Publishing Co., Amsterdam, xx+425pp, pp. 31–59.

— : 1982, 'The Type Theoretic Interpretation of Constructive Set Theory: Choice Principles', in *The L. E. J. Brouwer Centennary Symposium*, Proceedings of the Conference held in Noordwijkerhout, June 8–13 1981, A. S. Troelstra & D. van Dalen (eds.), series Studies in Logic and The Foundations of Mathematics, vol. 110, North-Holland, Amsterdam, x+523pp, pp. 1–40.

— : 1986, 'The Type Theoretic Interpretation of Constructive Set Theory: Inductive Definitions', in *Logic, Methodology and Philosophy of Science VII*, Proceedings of the Congress held July 11–16 1983, in Salzburg, Austria, R. Barcan Marcus, G. J. W. Dorn & P. Weingartner (eds.), series Studies in Logic and The Foundations of Mathematics, vol. 114, North-Holland Publishing Co., Amsterdam, xiii+738pp, pp. 17–49.

Allen, John: 1978, *Anatomy of LISP*, part of both McGraw-Hill Computer Science Series (Consulting Editors: Edward A. Feigenbaum & Richard W. Hamming), and McGraw-Hill Series in Artificial Intelligence (Consulting Editor: Edward A. Feigenbaum), McGraw-Hill Book Company, New York, xviii+446pp.

Ambrose, Alice: 1959, 'Proof and the theorem proved', in *Mind*, vol. 68, pp. 435–445.

— (ed.): 1979, *Wittgenstein's Lectures: Cambridge. 1932-35*, Basil Blackwell, Oxford, xii+225pp.

B

Backhouse, Roland: 1986, *On the Meanings and Construction of the Rules in Martin-Löf's Theory of Types*, Note CS 8606, series Computing Science Notes, Department of Mathematics and Computing Science, University of Groningen, Groningen, The Netherlands, 15pp.

Backhouse, Roland, Paul Chisholm, Grant Malcolm, Erik Saaman: 1989, 'Do-it-Yourself Type Theory', in *Formal Aspects of Computing*, Vol. 1, N. 1, pp. 19–84.

Barendregt, Henk P.: 1981, *The Lambda Calculus. Its Syntax and Semantics*, (revised edition 1984), series Studies in Logic and The Foundations of Mathematics, vol. 103, North-Holland, Amsterdam, xv+621pp.

Beeson, Michael J.: 1981, 'Formalizing Constructive Mathematics: Why and How?', in Proceedings of the workshop on *Constructive Mathematics*, New Mexico, 1980, series Lecture Notes in Mathematics, vol. 873, Springer-Verlag, pp. 146–190.

— : 1985, *Foundations of Constructive Mathematics. Metamathematical Studies*, series Ergebnisse der Mathematik und ihrer Grenzgebiete, vol. 6, Springer-Verlag, xxiv+466pp.

— : 1986, 'Proving Programs and Programming Proofs', in *Logic, Methodology and Philosophy of Science VII*, Proceedings of the Congress held in July 11–16, 1983, in Salzburg, Austria, R. Barcan Marcus, G. J. W. Dorn & P. Weingartner (eds.), series Studies in Logic and The Foundations of Mathematics, vol. 114, North-Holland Publishing Co., Amsterdam, xiii+738pp, pp. 51–82.

Bishop, Errett: 1967, *Foundations of Constructive Analysis*, McGraw-Hill series in Higher Mathematics, (Consulting Editor: E. H. Spanier), McGraw-Hill Book Company, New York, xiv+371pp.

— : 1970, 'Mathematics as a numerical language', in *Intuitionism and Proof Theory*, Proceedings of the *Summer Conference* at Buffalo, New York, 1968, A. Kino, J. Myhill & R. E. Vesley (eds.), series Studies in Logic and The Foundations of Mathematics, North-Holland, vii+516pp, pp. 53–71.

— : 1975, 'The Crisis in Contemporary Mathematics', in *Historia Mathematica*, vol. 2, pp.

507–517.

Bishop, Errett & H. Cheng: 1972, 'Constructive measure theory', in *Memoirs of the American Mathematical Society*, vol. 116.

Brand, Gerd: 1979, *The Central Texts of Wittgenstein*, (translated and with an introduction by Robert E. Innis), Basil Blackwell, Oxford, xxv+182pp.

Brouwer, Luitzen Egbertus Jan: 1975, *Collected Works. Vol 1. Philosophy and Foundations of Mathematics*, (Edited by A. Heyting), North-Holland Publishing Company, Amsterdam and Oxford, American Elsevier Publishing Company, New York, XV+628pp.

C

Church, Alonzo: 1936, 'An unsolvable problem of elementary number theory', in *American Journal of Mathematics*, vol. LVIII, pp. 345–363.

— : 1938, 'The Constructive Second Number Class', in *Bulletin of the American Mathematical Society*, vol. 44, pp. 224–232.

— : 1941, *The Calculi of Lambda-Conversion*, series Annals of Mathematical Studies, N. 6, Princeton University Press, Princeton, 77pp.

Constable, Robert L., & Nax P. Mendler: 1985, 'Recursive Definitions in Type Theory', in *Logics of Programs*, Proceedings of the symposium held in Brooklyn, June 1985, R. Parikh (ed.), series Lecture Notes in Computer Science, vol. 193, Springer-Verlag, vi+424pp, pp. 61–78.

Constable R. L., S. F. Allen, H. M. Bromley, W. R. Cleveland, J. F. Cremer, R. W. Harper, D. J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, J. T. Sasaki, S. F. Smith: 1986, *Implementing Mathematics with the Nuprl Proof Development System*, Prentice-Hall, New York, x+299pp.

Coquand, Thierry & Gérard Huet: 1988, 'The calculus of constructions', in *Information and Computation* (Special Issue edited by Gordon Plotkin, with a selection of papers given at the International Conference on *Semantics of Data Types*, held at Sophia-Antipolis, June 27–29, 1984), vol. 76, pp. 95–120.

da Costa, Newton C. A.: 1980, *Ensaio sobre os Fundamentos da Lógica*, (in Portuguese), Editora HUCITEC (Editora da Universidade de São Paulo), São Paulo, xii+255pp.

Curry, Haskell B.: 1951, *Outlines of a Formalist Philosophy of Mathematics*, series Studies in Logic and The Foundations of Mathematics, (Edited by L. E. J. Brouwer, E. W. Beth & A. Heyting), North-Holland Publishing Company, Amsterdam, viii+75pp.

D

Dedekind, Richard: 1890, 'Letter to Keferstein', in J. van Heijenoort (ed.) 1967, pp. 98–103.

— : 1901, *Essays on the Theory of Numbers. I. Continuity and Irrational Numbers. II. The Nature and Meaning of Numbers*, (authorised translation by Wooster Woodruff Beman), Open Court Mathematical series, The Open Court Publishing Company, Chicago, 115pp.

Diamond, Cora (ed.): 1976, *Wittgenstein's Lectures on the Foundations of Mathematics, Cambridge, 1939*, (From the notes of R. G. Bosanquet, Norman Malcolm, Rush Rhees and Yorick Smythies), The Harvester Press, Hassocks, Sussex, 300pp.

Dummett, Michael A. E.: 1959, 'Truth', in *Proceedings of the Aristotelian Society*, (new series), vol. 59, pp. 141–162.

— : 1973, 'The justification of deduction', in *Proceedings of the British Academy*, vol. 59, pp. 1–34.

— : 1975a, 'The Philosophical Basis of Intuitionistic Logic', in *Logic Colloquium '73*, H. E. Rose & J. C. Shepherdson (eds.), series Studies in Logic and The Foundations of Mathematics, North-Holland, Amsterdam, pp. 5–40.

— : 1975b, 'What is a Theory of Meaning?', in *Mind and Language. Wolfson College Lectures 1974*, Samuel Guttenplan (ed.), Claredon Press, Oxford, 158pp, pp. 97–138.

— : 1976, 'What is a Theory of Meaning? (II)', in *Truth and Meaning: Essays in Semantics*, Gareth Evans & John McDowell (eds.), Clarendon Press, Oxford, xviii+419pp, pp. 67–137.

— : 1977, *Elements of Intuitionism*, (with the assistance of Roberto Minio), (Reprinted 1978, 1985), series Oxford Logic Guides, (General Editors: Dana Scott & John Shepherdson), Clarendon Press, Oxford, xii+467pp.

— : 1980, 'Comments on Professor Prawitz's Paper', in *Logic and Philosophy*, *Entretiens* of the International Institute of Philosophy, Symposium held in Düsseldorf, August 27 – September 1 1978, G. H. von Wright (ed.), Martinus Nijhoff Publishers, The Hague, viii+84pp, pp. 11–18.

— : 1981, *The Interpretation of Frege's Philosophy*, Duckworth, xviii+621pp.

— : 1982, 'Realism', in *Synthese*, Special Issue: Realism, Part II, (edited by Esa Saarinen), vol. 52, pp. 55–112.

Dybjer, Peter: 1985, 'Category Theory and Programming Language Semantics: an Overview', in *Category Theory and Computer Programming*, D. Pitt, S. Abramsky, A. Poigné & D. Rydeheard (eds.), Proceedings of the Workshop held in Guildford, U.K., September 16–20 1985, series Lecture Notes in Computer Science, vol. 240, Springer-Verlag, vii+519pp, pp. 165–181.

— : 1987, 'Inductively Defined Sets in Martin-Löf's Set Theory', (Draft April 1987. A pub-

lished version appears in *Workshop on General Logic*, A. Avron, R. Harper, F. Honsell, I. Mason & G. Plotkin (eds.), Report ECS-LFCS-88-52, Department of Computer Science, University of Edinburgh, February 1987.)

E

Engelmann, Paul: 1967, *Letters from Ludwig Wittgenstein. With a Memoir*, (edited by B. F. McGuinness), (transl. by L. Furtmüller), Basil Blackwell, Oxford, xv+150pp.

Ehrig, Hartmut & Bernd Mahr: 1985, *Fundamentals of Algebraic Specifications 1. Equations and Initial Semantics*, series EATCS Monographs on Theoretical Computer Science, vol. 6, W. Brauer, G. Rozenberg & A. Salomaa (eds.), Springer-Verlag, Berlin, XI+321pp.

F

Feferman, Solomon: 1979, 'Constructive Theories of Functions and Classes', in *Logic Colloquium '78*, M. Boffa, D. van Dalen & K. McAloon (eds.), Proceedings of the colloquium held at Mons, Belgium, August 24 – September 1 1978, series Studies in Logic and The Foundations of Mathematics, North-Holland, pp. 159–224.

Fenstad, Jens E. (ed.): 1971, *Proceedings of the Second Scandinavian Logic Symposium*, held in Oslo, June 18–20 1970, series Studies in Logic and The Foundations of Mathematics, vol. 63, North-Holland, Amsterdam, viii+405pp.

Fitting, Melvin: 1981, *Proof Methods for Modal and Intuitionistic Logics*, series Synthese Library. Studies in Epistemology, Logic, Methodology and Philosophy of Science, vol. 169, D. Reidel, Dordrecht, viii+555pp.

Frege, Gottlob: 1879, *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*, (English translation in van Heijenoort (ed.) 1967, pp. 1–82.)

— : 1884, *Die Grundlagen der Arithmetik. (Eine logisch mathematische Untersuchung über den Begriff der Zahl)*, Auf der Grundlage der Centenerausgabe, herausgegeben von Christian Thiel, series Philosophische Bibliothek, Band 366, Felix Meiner Verlag, Hamburg, 1988, XXIV+144pp.

— : 1891, 'Funktion und Begriff', first published in the *Proceedings of the Jena Medical and Scientific Society*, 1891, English translation by Peter Geach 'Function and Concept' published in Frege 1984, pp. 137–156.

— : 1892, 'Über Sinn und Bedeutung', first published in *Zeitschrift für Philosophie und*

philosophische Kritik vol. 100, 1892, pp. 25–50, English translation by Max Black 'On Sense and Meaning' published in Frege 1984, pp. 157–177.

— : 1893: *Grundgesetze der Arithmetik. Begriffsschirftlich abgeleitet. I*, Jena, (reprinted in series Olms Paperbacks, Band 32, Georg Olms Verlagsbuchhandlung, Hildesheim, 1966), XXXII+254pp.

— : 1903: *Grundgesetze der Arithmetik. Begriffsschriftlich abgeleitet. II*, (Pohle, Jena), (reprinted in series Olms Paperbacks, Band 32, Georg Olms Verlagsbuchhandlung, Hildesheim, 1966), XVI+266pp.

— : 1914, 'Logic in Mathematics', in *Posthumous Writings*, Hans Hermes, Friedrich Kambartel & Friedrich Kaulbach (eds.), (transl. by Peter Long & Roger White), Basil Blackwell, Oxford, 1979, XIII+288pp, pp. 203–250.

— : 1984, *Collected Papers on Mathematics, Logic and Philosophy*, (Edited by Brian McGuinness), (Translated by Max Black, V. H. Dudman, Peter Geach, Hans Kaal, E.-H. W. Kludge, Brian McGuinness, R. H. Stoothoff), Basil Blackwell, Oxford, viii+412pp.

Friedman, Harvey: 1977, 'Set theoretic foundations for constructive analysis', in *Annals of Mathematics*, vol. 105, pp. 1–28.

Furth, Montgomery (ed.): 1964, *The Basic Laws of Arithmetic. Exposition of the System*, (partial English translation of Gottlob Frege's *Grundgesetze der Arithmetik*), University of California Press, Berkeley and Los Angeles, lxiv+143pp.

G

Geach, Peter T. (ed.): 1988, *Wittgenstein's Lectures on Philosophical Psychology 1946–47*, (Notes by Peter T. Geach, Kanti J. Shah & A. C. Jackson), Harvester-Wheatsheaf, Hemel Hempstead, xv+348pp.

Gentzen, Gerhard: 1935, 'Untersuchungen über das logische Schliessen', in *Mathematische Zeitschrift*, B. 39, pp. 176–210, 405–431. (English translation 'Investigations into Logical Deductions' in *The Collected Papers of Gerhard Gentzen*, M. E. Szabo (ed.), series Studies in Logic and The Foundations of Mathematics, North-Holland, Amsterdam, 1969, xiv+338pp, pp. 68–131.)

Gillies, Donald A.: 1980, 'Brouwer's Philosophy of Mathematics', (review article of Brouwer 1975), in *Erkenntnis*, vol. 15, pp. 105–126.

— : 1982, 'Wittgenstein and Revisionism', (review article of Crispin Wright's *Wittgenstein on the Foundations of Mathematics*, Duckworth, xix+481pp), in *The British Journal for the Philosophy of Science*, vol. 33, pp. 422–433.

Girard, Jean-Yves: 1971, 'Une Extension de l'Interpretation de Gödel à l'Analyse, et son Ap-

plication à l'Elimination des Coupures dans l'Analyse et la Theorie des Types', in Fenstad (ed.) 1971, pp. 63–92.

Gödel, Kurt: 1958, 'Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes', in *Dialectica*, vol. 12, pp. 280–287.

Goldblatt, Robert: 1982, *Axiomatising the Logic of Computer Programming*, series Lecture Notes in Computer Science, Springer-Verlag, New York, Heidelberg, Berlin, XI+304pp.

Gudmunsen, Chris: 1977, *Wittgenstein and Buddhism*, (Reprinted 1986), Macmillan Education, Houndmills and London, xii+128pp.

Guttag, John: 1977, 'Abstract Data Types and the Development of Data Structures', in *Communications of the ACM*, June 1977, Volume 20, Number 6, pp. 396–404.

H

Hallett, Garth L.: 1977, *A Companion to Wittgenstein's "Philosophical Investigations"*, Cornell University Press, Ithaca and London, 801pp.

Hamblin, Charles L.: 1970, *Fallacies*, Methuen & Co. Ltd., London, 326pp.

— : 1987, *Imperatives*, (Foreword by Nuel Belnap), Basil Blackwell, Oxford and New York, ix+262pp.

van Heijenoort, Jean (ed.): 1967, *From Frege to Gödel: A Source Book in Mathematical Logic. 1879-1931*, series Source Books in the History of the Sciences, Harvard University Press, Cambridge, Massachusetts, xii+664pp.

Henderson, Peter: 1980, *Functional Programming. Application and Implementation*, Prentice-Hall International Series in Computer Science, (Series Editor: C. A. R. Hoare), Prentice-Hall, Hemel Hempstead, xi+355pp.

Henkin, Leon: 1960, 'On Mathematical Induction', in *The American Mathematical Monthly*, vol. 67, n. 4, pp. 323–337.

Herrlich, Horst & George E. Strecker: 1973, *Category Theory. An Introduction*, Allyn and Bacon Inc., Boston, xi+400pp.

Hermes, Hans: 1959, 'Zum Inversionsprinzip der operativen Logik', in *Constructivity in Mathematics*, Proceedings of the Colloquium held at Amsterdam, 1957, A. Heyting (ed.), series Studies in Logic and The Foundations of Mathematics, North-Holland, viii+297pp, pp. 62–68.

Heyting, Arend: 1956, *Intuitionism. An Introduction*, series Studies in Logic and the Foundations of Mathematics, North-Holland, viii+133pp.

Hindley, J. Roger & Jonathan P. Seldin: 1986, *Introduction to Combinators and λ-Calculus*, series London Mathematical Society Student Texts, (Managing Editor: Prof. E. B. Davies), vol. 1,

Cambridge University Press, Cambridge, vi+360pp.

Hintikka, K. Jaakko: 1955, 'Form and Content in Quantification Theory', in *Acta Philosophica Fennica*, vol. 8, Helsinki, pp. 11–55.

— : 1983, 'Game-Theoretical Semantics: Insights and Prospects', in *The Game of Language*, J. Hintikka & J. Kulas (eds.), series Synthese Language Library, vol. 22, D. Reidel, Dordrecht, xii+319pp, pp. 1–31.

— : 1987, 'Language Understanding and Strategic Meaning', in *Synthese* (Special Issue on *Theories of Meaning*, Guest Editor: Maria Luisa Dalla Chiara, collecting articles originally presented as contributions to the conference "Theories of Meaning", organised by the Florence Center for the History and Philosophy of Science, Firenze, Villa di Mondeggi, June 1985), vol. 73, pp. 497–529.

Hoare, C. A. R.: 1969, 'An axiomatic basis for computer programming', in *Communications of the ACM*, vol. 12, pp. 576–580 & 583.

Howard, William A.: 1980, 'The formulae-as-types notion of construction', in *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, J. P. Seldin & J. R. Hindley (eds.), Academic Press, London, xxv+606pp, pp. 479–490.

Hughes, G. E. & M. J. Cresswell: 1968, *An Introduction to Modal Logic*, (reprinted 1982), Methuen and Co., xii+388pp.

J

James, William: 1890, *The Principles of Psychology. Vol I*, MacMillan and Co., Ltd., London, vii+689pp.

— : 1890, *The Principles of Psychology. Vol. II*, MacMillan and Co., Ltd., London, vi+704pp.

Janik, Allan (ed.): 1989, 'Gottlob Frege: Briefe an Ludwig Wittgenstein', in *Wittgenstein in Focus — Im Brennpunkt: Wittgenstein*, (Edited by Brian McGuinness & Rudolf Haller), series Grazer Philosophische Studien, (Edited by Rudolf Haller), vol. 33/34, Rodopi, Amsterdam/Atlanta, pp. 5–33.

Janik, Allan & Stephen Toulmin: 1973, *Wittgenstein's Vienna*, Touchstone (Published by Simon and Schuster), New York, 314pp.

Jaśkowski, Stanislaw: 1934, 'On the Rules of Suppositions in Formal Logic', reprinted in *Polish Logic 1920–1939*, S. McCall (ed.), 1967, Clarendon Press, Oxford, viii+406pp, pp. 232–258.

K

Kalmár, László: 1940, 'On the Possibility of Definition by Recursion', in *Acta Scientiarum Mathematicarum (Szeged)*, Tom. IX, Fasc. IV, pp. 227–232.

Kleene, Stephen C.: 1945, 'On the Interpretation of Intuitionistic Number Theory', in *Journal of Symbolic Logic*, vol. 10, pp. 109–124.

Kleene, Stephen C. & Richard Vesley: 1965, *The Foundations of Intuitionistic Mathematics*, North-Holland.

Kolmogorov, A. N.: 1932, 'Zur Deutung der intuitionistischen Logik', in *Mathematische Zeitschrift*, vol. 35, pp. 58–65.

Kreisel, Georg: 1968, 'A Survey of Proof Theory', in *Journal of Symbolic Logic*, vol. 33, n. 3, pp. 321–388.

Kripke, Saul A.: 1982, *Wittgenstein on Rules and Private Language. An Elementary Exposition*, Basil Blackwell, Oxford, x+150pp.

von Kutschera, F.: 1968, 'Die Vollständigkeit des Operatorensystems $\{\wedge, \vee, \neg, \supset\}$ für die intuitionische Aussagenlogik im Rahmen der Gentzensemantik', in *Archiv für mathematische Logik und Grundlagenforschung*, vol. 11, pp. 3–16.

L

Lakatos, Imre: 1976, *Proofs and Refutations. The Logic of Mathematical Discovery*, John Worrall & Elie Zahar (eds.), Cambridge University Press, Cambridge, xii+174pp.

Lambek, J. & P. J. Scott: 1986, *Introduction to higher order categorical logic*, series Cambridge Studies in Advanced Mathematics, (Editorial Board: D. J. H. Garling, D. Gorenstein, T. tom Dieck, P. Walters), vol. 7, Cambridge University Press, Cambridge, ix+293pp.

Lee, Desmond (ed.): 1980, *Wittgenstein's Lectures. Cambridge. 1930–32*, (from the notes of John King & Desmond Lee), Basil Blackwell, Oxford, xvii+124pp.

Lorenzen, Paul: 1939, 'Die Definition durch vollständige Induktion', in *Monatshefte für Mathematik und Physik*, B. 47, pp. 356–358.

— : 1955, *Einführung in die operative Logik und Mathematik*, series Die Grundlehren der mathematischen Wissenschaften, (Edited by R. Grammel, E. Hopf, H. Hopf, F. Rellich, F. K. Schmidt & B. L. van der Waerden), Band LXXVIII, Springer-Verlag, Berlin, iv+298pp.

— : 1969, *Normative Logic and Ethics*, series B.I-Hochschultaschenbücher. Systematische Philosophie, vol. 236*, Bibliographisches Institut, Mannheim/Zürich, 89pp.

M

Maibaum, Thomas S. E., Martin R. Sadler & Paulo A. S. Veloso: 1984, 'Logical Specification and Implementation', in Proceedings of the *4th International Conference on the Foundations of Software Technology and Theoretical Computer Science*, M. Joseph & R. Shyamasundar (eds.), Bangalore, India, series Lecture Notes in Computer Science, Springer-Verlag, vol. 181, 468pp, pp. 13–30.

Malcolm, Norman: 1958, *Ludwig Wittgenstein. A Memoir*, (With a Biographical Sketch by G. H. von Wright), Oxford University Press, London, 100pp.

Markwald, Werner: 1954, 'Zur Theorie der konstruktiven Wohlordnungen', in *Mathematische Annalen*, B. 127, pp. 135–149.

Martin-Löf, Per: 1970, *Notes on Constructive Mathematics*, Almqvist & Wiksell, Stockholm, 109pp.

— : 1971a, 'Hauptsatz for the intuitionistic theory of iterated inductive definitions', in Fenstad (ed.) 1971, pp. 179–216.

— : 1971b, 'Hauptsatz for the theory of species', in Fenstad (ed.) 1971, pp. 217–233.

— : 1973, 'Hauptsatz for intuitionistic simple type theory', in *Proceedings of the Fourth International Congress for Logic, Methodology and Philosophy of Science*, P. Suppes et. al. (eds.), held in Bucharest, 1971, North-Holland, Amsterdam, pp. 279–290.

— : 1975a, 'An intuitionistic theory of types: predicative part', in *Logic Colloquium '73*, H. E. Rose & J. C. Shepherdson (eds.), series Studies in Logic and The Foundations of Mathematics, North-Holland, Amsterdam, pp. 73–118.

— : 1975b, 'About Models for Intuitionistic Type Theories and the Notion of Definitional Equality', in *Proceedings of the Third Scandinavian Logic Symposium*, 1973, S. Kanger (ed.), North-Holland, pp. 81–109.

— : 1982, 'Constructive Mathematics and Computer Programming', in the Proceedings of the International Congress for *Logic, Methodology and Philosophy of Science VI*, Hannover, August 22–29 1979, L. J. Cohen, J. Los, H. Pfeiffer & K.-P. Podewski (eds.), series Studies in Logic and The Foundations of Mathematics, North-Holland, Amsterdam, xiii+738pp, pp. 153–175.

— : 1984, *Intuitionistic Type Theory*, (Notes by Giovanni Sambin of a series of lectures given in Padova, June 1980), series Studies in Proof Theory, Bibliopolis, Naples, iv+91pp.

— : 1985, 'On the meanings of the logical constants and the justifications of the logical laws', in *Atti degli incontri di logica matematica*, vol. 2, C. Bernardi & P. Pagli (eds.), series Scuola di Specializzazione in Logica Matematica, Dipartimento di Matematica, Università di Siena, pp. 203–281.

— : 1987, 'Truth of a Proposition, Evidence of a Judgement, Validity of a Proof', in *Synthese*, (Special Issue on *Theories of Meaning*, Guest Editor: Maria Luisa Dalla Chiara, collecting articles originally presented as contributions to the conference "Theories of Meaning", organised by the Florence Center for the History and Philosophy of Science, Firenze, Villa di Mondeggi, June 1985), vol. 73, pp. 407–420.

— : 1989, *Mathematics of Infinity*, typed manuscript, Stockholm, 51pp.

Mauthner, Fritz: 1906, *Die Sprache*, series Die Gesellschaft: Sammlung Sozial-Psychologischer Monographien, Martin Buber (Herausg.), Vol. 9, Rütten & Loening, Frankfurt am Main, 120pp.

— : 1911, *Schopenhauer*, Georg Müller Verlag, Munich & Leipzig, VII+129pp.

— : 1923, *Beiträge zu einer Kritik der Sprache. Vol. I. Zur Sprache und Zur Psychologie*, (3rd edition), Verlag von Felix Meiner, Leipzig, XVIII+719pp. (sometimes referred to simply as '*Beiträge I*' *passim*.)

— : 1923, *Beiträge zu einer Kritik der Sprache. Vol. II. Zur Sprachwissenschaft*, (3rd edition), Verlag von Felix Meiner, Leipzig, VI+718pp. (sometimes referred to simply as '*Beiträge II*' *passim*.)

— : 1923, *Beiträge zu einer Kritik der Sprache. Vol. III. Zur Grammatik und Logik*, (3rd edition), Verlag von Felix Meiner, Leipzig, XI+663pp. (sometimes referred to simply as '*Beiträge III*' *passim*.)

— : 1923, *Wörterbuch der Philosophie: Neue Beiträge zu einer Kritik der Sprache. Vol. I. A=A to GOETHES WEISHEIT*, (2nd edition), Verlag von Felix Meiner, Leipzig, CXXX+661pp. (sometimes referred to simply as '*Wörterbuch I*' *passim*.)

— : 1924, *Wörterbuch der Philosophie: Neue Beiträge zu einer Kritik der Sprache. Vol. II. GOTT to QUIETIV*, (2nd edition), Verlag von Felix Meiner, Leipzig, 586pp. (sometimes referred to simply as '*Wörterbuch II*' *passim*.)

— : 1924, *Wörterbuch der Philosophie: Neue Beiträge zu einer Kritik der Sprache. Vol. III. RATIONALISMUS to ZWECK. Personen- und Sachregister*, (2nd edition), Verlag von Felix Meiner, Leipzig, 560pp. (sometimes referred to simply as '*Wörterbuch III*' *passim*.)

McCarthy, John, Paul W. Abrahams, Daniel J. Edwards, Timothy P. Hart, Michael I. Levin: 1962, *LISP 1.5 Programmer's Manual*, (Second Edition 1965, Eleventh printing 1979), The Computation Center and Research Laboratory of Electronics, Massachussetts Institute of Technology, The M.I.T. Press, Cambridge, Massachussetts, vi+106pp.

Mendler, Nax P., P. Panangaden & Robert L. Constable: 1986, 'Infinite Objects in Type Theory', in *Proceedings of the Symposium on Logic in Computer Science (LICS)*, Cambridge, Massachussetts, June 16–18 1986, pp. 249–255.

Moravia, Sergio: 1986, *L'Enigma della Mente. Il 'Mind-Body Problem' nel Pensiero Contem-*

poraneo, series Biblioteca di Cultura Laterza, vol. 932, Gius. Laterza & Figli Spa, Roma, XXXVI+327pp.

Morris, John H., Jr.: 1973, 'Types are not sets', in *Conference Record of ACM Symposium on Principles of Programming Languages*, held in Boston, Massachussetts, October 1–3 1973, pp. 120–124.

Myhill, John: 1975, 'Constructive Set Theory', in *Journal of Symbolic Logic*, vol. 40, pp. 347–383.

N

Nedo, Michael & Michele Ranchetti (eds.): 1983, *Ludwig Wittgenstein. Sein Leben in Bildern und Texten*, (Vorwort von B. F. McGuinness), (Gestaltet von Willy Fleckhaus), Suhrkamp Verlag, Frankfurt am Main, XV+394pp.

Nordström, Bengt: 1986, *Martin-Löf's Type Theory as a Programming Logic*, Technical Report 27, Programming Methodology Group, University of Göteborg, September 1986, 25pp.

Nordström, Bengt, Kent Petersson & Jan Smith: 1986, *An Introduction to Martin-Löf's Type Theory*, iv+114pp. (Draft Midsummer 1986. To be published by Oxford University Press. If not yet published, at least a draft copy should be obtainable from the authors at Programming Methodology Group, Department of Computer Sciences, University of Göteborg, Chalmers University of Technology, S-412 96 Göteborg, Sweden.)

P

Paulson, Lawrence: 1984, *Constructing Recursion Operators in Intuitionistic Type Theory*, Technical Report No. 57, University of Cambridge, Computing Lab, Cambridge, 46pp.

Peano, Giuseppe: 1889, *Arithmetices principia*, (English translation *The principles of arithmetic, presented by a new method* published in van Heijenoort (ed.) 1967, pp. 83–97.).

Péter, Rósza: 1967, *Recursive Functions*, (3rd revised ed.), Academic Press, London, 300pp.

Peyton-Jones Simon L.: 1987, *The Implementation of Functional Programming Languages*, Prentice-Hall International Series in Computer Science, (Series Editor: C. A. R. Hoare), Prentice-Hall, Hemel Hempstead, xviii+445pp.

Prawitz, Dag: 1965, *Natural Deduction. A Proof-Theoretical Study*, series Acta Universitatis Stockholmiensis. Stockholm Studies in Philosophy 3, Almqvist & Wiksell, Uppsala, 113pp.

— : 1971, 'Ideas and Results in Proof Theory', in Fenstad (ed.) 1971, pp. 235–307.

— : 1973, 'Towards a Foundation of a General Proof Theory', in *Proceedings of the Fourth International Congress for Logic, Methodology and Philosophy of Science*, held in Bucharest, 1971, P. Suppes, L. Henkin, A. Joja & Gr. C. Moisil (eds.), North-Holland, pp. 225–250.

— : 1974, 'On the Idea of a General Proof Theory', in *Synthese*, vol. 27, pp. 63–77.

— : 1975, 'Comments on Gentzen-type procedures and the classical notion of truth', in *Proof Theory Symposium*, J. Diller & G. H. Müller (eds.), series Lecture Notes in Mathematics, vol. 500, Springer-Verlag, Berlin, pp. 290–319.

— : 1977, 'Meaning and proofs: on the conflict between classical and intuitionistic logic', in *Theoria* (Sweden), vol. XLIII, pp. 2–40.

— : 1978, 'Proofs and the Meaning and Completeness of the Logical Constants', in *Essays on Mathematical and Philosophical Logic*, Proceedings of the Fourth Scandinavian Logic Symposium and of the First Soviet-Finnish Logic Conference, Jyväskylä, Finland, June 29–July 6 1976, J. Hintikka, I. Niiniluoto & E. Saarinen (eds.), series Synthese Library, vol. 122, D. Reidel Publishing Company, Dordrecht, viii+462pp, pp. 25–40.

— : 1980, 'Intuitionistic Logic: A Philosophical Challenge', in *Logic and Philosophy, Entretiens* of the International Institute of Philosophy, Symposium held in Düsseldorf, August 27 – September 1 1978, G. H. von Wright (ed.), Martinus Nijhoff Publishers, The Hague, viii+84pp, pp. 1–10.

— : 1985, 'Remarks on some approaches to the concept of logical consequence', in *Synthese*, vol. 62, pp. 153–171.

— : 1987, 'Some Remarks on Verificationistic Theories of Meaning', in *Synthese* (Special Issue on *Theories of Meaning*, Guest Editor: Maria Luisa Dalla Chiara, collecting articles originally presented as contributions to the conference "Theories of Meaning", organised by the Florence Center for the History and Philosophy of Science, Firenze, Villa di Mondeggi, June 1985), vol. 73, pp. 471–477.

Prior, A. N.: 1960, 'The runabout inference-ticket', in *Analysis*, vol. 21, pp. 38–39.

Q

de Queiroz, Ruy J. G. B.: 1987a, 'Note on Frege's notions of definition and the Relationship Proof Theory vs. Recursion Theory', (extended abstract published in *Abstracts of the VIIIth International Congress of Logic, Methodology and Philosophy of Science*, Institute of Philosophy of the Academy of Sciences of the USSR, Vol. 5, Part I, pp. 69–73, Moscow, USSR, August '87), 47pp. (submitted for publication)

— : 1987b, 'A Word on Rules and Meaning', (presented at the *ASL/LSA Conference on Logic*

and Linguistics, Centre for the Study of Language and Information (CSLI), Stanford University, Stanford, USA, July 10–12 1987.) (abstract to be published in *Journal of Symbolic Logic*).

— : 1987c, 'Meaning as use, not content: an alternative view on the proof-theoretic account of meaning', 42pp. (submitted for publication)

— : 1988, 'A Proof-Theoretic Account of Programming and the Rôle of Reduction Rules', in *Dialectica*, vol. 42, n. 4, pp. 265–282.

— : 1988a, 'Meaning as grammar *plus* consequences', 5pp. (submitted for publication)

— : 1988b, 'Normalisation and Language-Games', 19pp. (submitted for publication)

— : 1988c, 'The Two Wittgensteins and the Semantics of the Mathematical Language', 13pp. (submitted for publication).

— : 1988d, 'Normalisation and the Semantics of Use', (abstract published in *Abstracts of the Logic Colloquium '88*, 150pp, p. 52, Padova, Italy, August 23–30 1988. Abstract also to appear in the *Journal of Symbolic Logic*.).

— : 1988e, 'The Mathematical Language and its Semantics: to Show the Consequences of a Proposition is to Give its Meaning', Abstract published in the *Abstracts of the Thirteenth International Wittgenstein Symposium*, organised by the Österreichische Ludwig Wittgenstein Gesellschaft, Institut für Wittgensteinforschung, Analytische Philosophie und Wissenschaftstheorie, 93pp, p. 17, Kirchberg/Wechsel, Austria, August 14–21 1988. Full paper in the *Reports of the Thirteenth International Wittgenstein Symposium 1988*, (Edited by Paul Weingartner & Gerhard Schurz), series Schriftenreihe der Wittgenstein-Gesellschaft, Vol. 18, (Edited by Elisabeth Leinfellner, Rudolf Haller, Adolf Hübner, Werner Leinfellner & Paul Weingartner), Hölder-Pichler-Tempsky, Vienna, 304pp, pp. 259–266.

— : 1988f, 'Verification and Meaning', 33pp. (submitted for publication)

— : 1988g, 'Intuitionistic Type Theory and Frege's Realism', (in preparation, partial draft March 9, 1989: 26pp.)

— : 1989a, 'Meaning, Function, Purpose, Usefulness, *Consequences* – Interconnected Concepts', Abstract in p. 20 of *Abstracts of Fourteenth International Wittgenstein Symposium (Centenary Celebration)*, 112pp, Österreichische Ludwig Wittgenstein Gesellschaft, Institut für Wittgensteinforschung, Analytische Philosophie und Wissenschaftstheorie, Kirchberg/Wechsel, August 13–20 1989, 14pp. Full paper to appear in the *Reports* of the Symposium.

— : 1989b, 'Meaning, Use, Consequences – Interconnected Concepts', (presented as an invited paper at the *International Symposium on Communication, Meaning and Knowledge*, organised by the Associação Portuguesa para o Desenvolvimento das Comunicações and by the Sociedade Portuguesa de Filosofia, Lisbon, Portugal, September 13–15 1989.), 16pp. To

appear in the *Proceedings* of the Symposium.

de Queiroz, Ruy J. G. B. & Thomas S. E. Maibaum: 1987a, 'Proof Theory and Computer Programming', (to appear in vol. 36 (1990) of the *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*), 28pp.

— : 1987b, 'Intuitionistic Type Theory and Abstract Data Types Specification: Reduction Rules as Instruments of 'Semantics of Use' ', 38pp. (submitted for publication)

— : 1988, *''Meaning-As-USE' Type Theory* and Theories of Abstract Data Types Specification: Reduction Rules (Equations) constituting the 'Semantics of Use' of the Type', (accepted for presentation at the *Sixth Workshop on Specification of Abstract Data Types*, Technische Universität Berlin, August 29 – September 2 1988.).

— : 1988a, *''Meaning-As-USE' Type Theory* and Abstract Data Types: Types and their *Utility*', 31pp. (submitted for publication)

— : 1988b, 'Abstract Data Types and Type Theory: Theories as Types', 21pp. (submitted for publication)

de Queiroz, Ruy J. G. B. & Michael B. Smyth: 1989, 'Induction Rules for Non-Inductive Types in Type Theory', (presented at the *Fifth British Colloquium for Theoretical Computer Science*, Royal Holloway and Bedford New College, Egham, Surrey, UK, April 11–13 1989. Abstract to appear in the *Bulletin of European Association for Theoretical Computer Science (EATCS)*.).

— : 198?a, 'The Inductive Rôle of λ-Calculus η-Rule' (in preparation).

R

Reynolds, John C.: 1974, 'Towards a theory of type structure', in *Programming Symposium*, Colloque sur la Programmation, Paris, April 9–11 1974, B. Robinet (ed.), series Lecture Notes in Computer Science, vol. 19, Springer-Verlag, Berlin/New York, 425pp, pp. 408–425.

Rhees, Rush (ed.): 1968, 'Wittgenstein's Notes for Lectures on 'Private Experience' and 'Sense Data'', in *Philosophical Review*, vol. 77, pp. 271–320.

— : 1976, 'Wittgenstein's 'Ursache und Wirkung: Intuitives Erfassen'', in *Philosophia*, vol. 6, pp. 392–408, 427–446.

Russell, Bertrand: 1908, 'Mathematical logic as based on the theory of types', in van Heijenoort (ed.) 1967, pp. 150–182.

S

Schroeder-Heister, Peter: 1983a, 'The Completeness of Intuitionistic Logic with Respect to a Validity Concept Based on an Inversion Principle', in *Journal of Philosophical Logic*, vol. 12, pp. 359–377.

— : 1983b, 'Generalized Rules for Quantifiers and the Completeness of the Intuitionistic Operators &, ∨, ⊃, ⊥, ∀, ∃', in *Computation and Proof Theory*, Proceedings of the *Logic Colloquium '83*, Aachen, West Germany, Part II, E. Börger, W. Oberschelp, M. M. Richter, B. Schinzel & W. Thomas (eds.), series Lecture Notes in Mathematics, vol. 1104, Springer-Verlag, viii+475pp, pp. 399–426.

— : 1984, 'A Natural Extension of Natural Deduction', in *Journal of Symbolic Logic*, vol. 49, n. 4, pp. 1284–1300.

Scott, Dana S.: 1970, 'Constructive Validity', in *Symposium on Automated Deduction*, series Lecture Notes in Mathematics, vol. 125, Springer-Verlag, pp. 237–275.

— : 1976, 'Data Types as Lattices', in *SIAM Journal on Computing*, vol. 5, n. 3, pp. 522–587.

— : 1981, *Lectures on a Mathematical Theory of Computation*, Oxford University Computing Laboratory, Programming Research Group, Technical Monograph PRG-19, iv+148pp.

— : 1982, 'Domains for Denotational Semantics', in Proceedings of the *ICALP '82*, Edited by M. Nielsen & E. M. Schmidt, series Lecture Notes in Computer Science, Springer-Verlag, vol. 140, pp. 577–613.

Sierpiński, Wacław: 1958, *Cardinal and Ordinal Numbers*, Polska Akademia Nauk, series Monografie Matematyczne, Tom 34, Warsaw, 487pp.

Sluga, Hans: 1985, 'Mauthner, Wittgenstein, and the Critique of Language', (typed manuscript of the written account of a talk delivered at a conference on Austrian philosophy before Wittgenstein, Stanford University, May 17, 1985), 25pp.

Smith, Jan: 1983, 'The Identification of Propositions and Types in Martin-Löf's Type Theory: A Programming Example', in Proceedings of the International Conference on *Foundations of Computation Theory*, Borgholm, Sweden, August 21–27 1983, M. Karpinski (ed.), series Lecture Notes in Computer Science, vol. 158, Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, pp. 445–456.

van Stigt, Walter P.: 1979, 'The rejected parts of Brouwer's Dissertation on the foundations of mathematics', in *Historia Mathematica*, vol. 6, pp. 385–404.

— : 1982, 'L. E. J. Brouwer, the signific interlude', in *The L. E. J. Brouwer Centenary Symposium*, Proceedings of the Conference held in Noordwijkerhout, June 8–13 1981, A. S. Troelstra & D. van Dalen (eds.), series Studies in Logic and The Foundations of Mathematics, vol. 110, North-Holland, Amsterdam, x+523pp, pp. 505–512.

Sundholm, Göran: 1983, 'Constructions, proofs and the meaning of logical constants', in *Journal*

of Philosophical Logic, vol. 12, pp. 151–172.

— : 1986, 'Proof Theory and Meaning', in *Handbook of Philosophical Logic*, vol. III *Alternatives to Classical Logic*, Dov M. Gabbay & Franz Guenthner (eds.), series Synthese Library, vol. 166, D. Reidel Publishing Company, Dordrecht, xii+520pp, pp. 471–506.

T

Tait, William W.: 1967, 'Intentional interpretation of functionals of finite type I', in *Journal of Symbolic Logic*, vol. 32, pp. 198–212.

— : 1968, 'Constructive Reasoning', in *Logic, Methodology and Philosophy of Science III*, B. van Rootselaar & J. F. Staal (eds.), North-Holland, pp. 185–199.

Tennant, Neil: 1978, *Natural Logic*, Edinburgh University Press, Edinburgh, 196pp.

Thatcher, J. W., E. G. Wagner & J. B. Wright: 1982, 'Data Type Specification: Parameterization and the Power of Specification Techniques', in *ACM Transactions on Programming Languages and Systems (TOPLAS)*, Vol. 4, N. 4, October 1982, pp. 711–732.

Thomason, Richmond (ed.): 1974, *Formal Philosophy. Selected Papers of Richard Montague*, Yale University Press, New Haven and London, 369pp.

Troelstra, A. S.: 1977, *Choice Sequences. A Chapter of Intuitionistic Mathematics*, series Oxford Logic Guides, (General Editor: Dana Scott), Clarendon Press, Oxford, ix+170pp.

Turner, Raymond: 1984, *Logics for Artificial Intelligence*, Ellis Horwood series in Artificial Intelligence, (Series Editor: John Campbell), Ellis Horwood Limited, Chichester.

— : 1987, 'A Theory of Properties', in *Journal of Symbolic Logic*, vol. 52, n. 2, pp. 445–472.

— : 1988, 'Properties, Propositions and Semantic Theory', Technical Report, Department of Computer Science, University of Essex, Colchester, UK, 15pp.

Turski, Władysław M. & Thomas S. E. Maibaum: 1987, *The Specification of Computer Programs*, International Computer Science Series, (Consulting Editors: A. D. McGettrick & J. van Leeuwen), Addison-Wesley, x+278pp.

V

Veloso, Paulo A. S.: 1985, *Programme development as theory manipulations*, series Monografias em Ciência da Computação, n. 4/85, (Series Editor: Paulo A. S. Veloso), Departamento de Informática, Pontifícia Universidade Católica, Rio de Janeiro, May 1985, 30pp.

W

Waismann, Friedrich: 1979, *Ludwig Wittgenstein and the Vienna Circle*, (Reprinted 1983), (edited by Brian McGuinness), (German edition 1967), (translated by Joachim Schulte and Brian McGuinness), Basil Blackwell, Oxford, 266pp. (sometimes referred to simply as '*LW&VC*' *passim*.)

Weiler, Gershon: 1970, *Mauthner's Critique of Language*, Cambridge University Press, xiii+346pp.

Werhane, Patricia H.: 1987, 'Some paradoxes in Kripke's interpretation of Wittgenstein', in *Synthese*, vol. 75, pp. 253–273.

Winskel, Glynn & Kim G. Larsen: 1983, *Using Information Systems to Solve Recursive Domain Equations Effectively*, Technical Report No. 51, Computer Laboratory, University of Cambridge, 41pp.

Wittgenstein, Ludwig: 1922, *Tractatus Logico-Philosophicus*, (transl. by David F. Pears & Brian F. McGuinness, with Introd. by Bertrand Russell), (this translated edition first published 1961, reprinted 1981), series International Library of Philosophy, Routledge & Kegan Paul, London, xxii+89pp. (sometimes referred to simply as '*Tractatus*' *passim*.)

— : 1929, 'Some Remarks on Logical Form', in *Aristotelean Society Supplementary Volume 9*, pp. 162–171.

— : 1953, *Philosophical Investigations*, (transl. by G. E. M. Anscombe), (2nd edition 1958, reprinted 1984), Basil Blackwell, Oxford, viii+250^{e}pp. (sometimes referred to simply as '*Investigations*' *passim*.)

— : 1956, *Remarks on the Foundations of Mathematics*, G. H. von Wright, R. Rhees & G. E. M. Anscombe (eds.), (transl. by G. E. M. Anscombe), (3rd edition, revised and reset, 1978), (Reprinted 1989), Basil Blackwell, Oxford, 444pp. (sometimes referred to simply as '*RFM*' *passim*.)

— : 1958, *The Blue and Brown Books. (Preliminary Studies for the 'Philosophical Investigations')*, (2nd edition 1969, reprinted 1989), Basil Blackwell, Oxford, xiv+192pp. (sometimes referred to simply as '*TB&BB*' as a whole, or separately '*Blue Book*' and '*Brown Book*' *passim*.)

— : 1961, *Notebooks 1914-1916*, G. H. von Wright & G. E. M. Anscombe (eds.), (with an English translation by G. E. M. Anscombe), Basil Blackwell, Oxford, 140+91^{e}pp. (sometimes referred to simply as '*Notebooks*' *passim*.)

— : 1965, 'Lectures on Ethics', in *Philosophical Review*, vol. 74, pp. 3–12.

— : 1966, *Lectures and Conversations on Aesthetics, Psychology and Religious Belief*, (Compiled from Notes taken by Yorick Smythies, Rush Rhees and James Taylor), Cyril Barrett (ed.), (Fourth Impression 1978, Reprinted 1987), Basil Blackwell, Oxford, viii+72pp. (some-

times referred to simply as '*Lectures and Conversations on APRB*' *passim*.)

— : 1967, *Zettel*, G. E. M. Anscombe & G. H. von Wright (eds.), (transl. by G. E. M. Anscombe), (2nd revised edition 1981), Basil Blackwell, Oxford, vi+138pp.

— : 1969, *Briefe an Ludwig von Ficker*, (Edited by G. H. von Wright in collaboration with Walter Methlagl), Otto Müller Verlag, Salzburg, 110pp. (sometimes referred to simply as '*Briefe an LvF*' *passim*.)

— : 1969, *On Certainty/ Über Gewissheit*, G. E. M. Anscombe & G. H. von Wright (eds.), (transl. by Denis Paul & G. E. M. Anscombe), Harper & Row Publishers, New York, vii+90+90^{e}pp. (sometimes referred to simply as '*OC*' *passim*.)

— : 1971, *Prototractatus*, (An early version of *Tractatus Logico-Philosophicus*), B. F. McGuinness, T. Nyberg & G. H. von Wright (eds.), (With a translation by D. F. Pears & B. F. McGuinness), (an historical introduction by G. H. von Wright), (and a facsimile of the author's manuscript), Routledge & Kegan Paul, London, 256pp.

— : 1973, *Letters to C. K. Ogden*, (with Comments on the English Translation of the *Tractatus Logico-Philosophicus*), (Edited with an Introduction by G. H. von Wright), (and an Appendix of Letters by Frank Plumpton Ramsey), Basil Blackwell, Oxford, and Routledge & Kegan Paul, London and Boston, ix+90pp. (sometimes referred to simply as '*Letters to CKO*' *passim*.)

— : 1974, *Letters to Russell, Keynes & Moore*, Edited with an Introduction by G. H. von Wright, (assisted by B. F. McGuinness), Basil Blackwell, Oxford, 190pp. (sometimes referred to simply as '*Letters to RK&M*' *passim*.)

— : 1974, *Philosophical Grammar*, Rush Rhees (ed.), (transl. by Anthony Kenny), (reprinted 1984), Basil Blackwell, Oxford, 495pp. (sometimes referred to simply as '*PG*' *passim*.)

— : 1975, *Philosophical Remarks*, Rush Rhees (ed.), (transl. by R. Hargreaves & R. White), Basil Blackwell, Oxford, 357pp. (sometimes referred to simply as '*PR*' *passim*.)

— : 1977, *Culture and Value / Vermischte Bemerkungen*, G. H. von Wright (ed.) (in collab. with H. Nyman), (transl. by P. Winch), (2nd Edition 1978, Amended second Edition with English Translation 1980, Reprinted 1988), Basil Blackwell, Oxford, vii+94+87^{e}pp. (sometimes referred to simply as '*C&V*' *passim*.)

— : 1977, *Remarks on Colour*, G. E. M. Anscombe (ed.), (transl. by Linda L. McAlister & Margarete Schättle), (Reprinted 1988), Basil Blackwell, Oxford, 63+63^{e}pp. (sometimes referred to simply as '*RC*' *passim*.)

— : 1977, *Wörterbuch für Volksschulen*, (Edited with an Introduction by Adolf Hübner-Werner and Elisabeth Leinfellner), series Schriftenreihe der Wittgenstein-Gesellschaft, Band 1, Verlag Hölder-Pichler-Tempsky, Vienna, XXXV+44pp. (sometimes referred to simply as '*Wörterbuch*' *passim*.)

— : 1979, *Remarks on Frazer's GOLDEN BOUGH / Bemerkungen über Frazers GOLDEN BOUGH*, R. Rhees (ed.), (English translation by A. C. Miles), (revised by R. Rhees), (reprinted 1987), The Brynmill Press Ltd, Doncaster, & Humanities Press International Inc., New Jersey, vi+18+18^{e}pp. (sometimes referred to simply as '*RFGB*' *passim*.)

— : 1980, *Remarks on the Philosophy of Psychology*, Vol. I, G. E. M. Anscombe & G. H. von Wright (eds.), (transl. by G. E. M. Anscombe), Basil Blackwell, Oxford, vi+218+198^{e}pp. (sometimes referred to simply as '*RPP I*' *passim*.)

— : 1980, *Remarks on the Philosophy of Psychology*, Vol. II, G. H. von Wright (ed.), (transl. by C. G. Luckhardt & M. A. E. Aue), Basil Blackwell, Oxford, 143+121^{e}pp. (sometimes referred to simply as '*RPP II*' *passim*.)

— : 1982, *Last Writings on the Philosophy of Psychology*, Vol. I, *(Preliminary Studies for Part II of 'Philosophical Investigations')*, G. H. von Wright & H. Nyman (eds.), (transl. by C. G. Luckhardt & Maximillian A. E. Aue), Basil Blackwell, Oxford, 148+127^{e}pp. (sometimes referred to simply as '*Last Writings I*' *passim*.)

von Wright, Georg H.: 1982, *Wittgenstein*, University of Minnesota Press, Minnesota, vi+218pp.

Z

Zucker, Jeffrey I. & R. S. Tragesser: 1978, 'The Adequacy Problem for Inferential Logic', in *Journal of Philosophical Logic*, vol. 7, pp. 501–516.

Index of Names

A

Abramsky, S., xiii
Aczel, P., xiii, 13, 19, 34, 35, 47, 124, 125, 152, 153, 165, 166, 214, 231, 242, 245, 276, 277, 285, 286, 333, 334, 336
Ahn, R., xiv, 262
Allen, J., 340, 341
Ambrose, A., 30, 91

B

Backhouse, R., 234, 260
Barendregt, H., xiii, 9, 46, 124, 276, 314, 321, 322, 345–348, 364
Beeson, M., 158, 245–247, 330
Beman, W. W., xviii
Benevides, M., xiv
Bishop, E., 157, 158, 163, 255, 278, 360
Boltzmann, L., 102
Bradshaw, R., xiv
Brouwer, L. E. J., 93, 232, 237, 278

C

Cantell, F., xiv
Church, A., 11, 12, 151, 165, 190, 206, 231, 281
Constable, R. L., 153, 277, 281, 283, 285
Costa, M., xiv
da Costa, N., xiii
Cunha, J., xiv
Cunha, P. R. F., xiii
Curry, H., 158, 166

D

Dawson, M., xiv
Dedekind, R., xviii–xx, 271
Diamond, C., 127
Dummett, M., 15, 27, 49–51, 55, 56, 74–76, 89, 90, 120, 121, 136, 174, 184, 216–218, 221, 223, 224, 237, 259, 360, 365, 367
Dybjer, P., 47, 49, 124, 277, 281

E

Ehrig, H., 125, 182, 203, 235, 320, 324, 326, 327, 331, 332, 355

F

Feferman, S., 158
Fernandes, C., xiv
Fitting, M., 60
Frege, G., xv, xvi, xx, 3, 6–8, 11–13, 15–18, 30, 86, 88, 89, 94, 102, 103, 108, 109, 112, 124, 131, 136, 150, 161, 165, 166, 189, 238, 240, 308, 315–318, 359
Friedman, H., 158
Fuks, H., xiv
Furth, M., xx, 165

G

Gödel, K., 149, 157
Gabbay, D., xiii
Geach, P., 134, 214
Gentzen, G., xvi, 4, 5, 8, 14, 15, 26, 27, 29–32, 36, 37, 44, 49–54, 56, 60, 61, 66, 70, 91, 141, 142, 145–147, 149, 159, 160, 174, 187, 220, 256, 257, 259–261, 263, 351, 356, 360, 364, 365
Gillies, D., xiii
Girard, J.-Y., 149, 150, 158
Gochet, P., xiii
Goldblatt, R., 1
Gondim, J., xiv
Guttag, J., 203, 236, 237, 326, 327, 353, 354

H

Haigh, E., xiv
Halford, A., xiv
Hallett, G., 106
Hamblin, C., 104, 221
Hancock, P., xiii, 53
Harrison, K., xiv

Harwood, W., xiv
van Heijenoort, J., 271
Helmink, L., xiv, 262
Henderson, P., 344, 345
Herrlich, H., 154
Hertz, H., 102
Heyting, A., 66, 145, 172–176, 198, 237
Hilbert, D., 30, 147
Hindley, J., 273, 274
Hintikka, J., xvi, 28, 77–80, 118, 119, 128, 145, 146, 187, 217–219, 221, 222, 227, 228, 273
Hoare, C. A. R., 1
Howard, W., 158

J

Janik, A., 63
Jeremaes, P., xiv

K

Kalmár, L., 22, 23
Kant, I., 102
Keferstein, H., 271
Khosla, S., xiv
Kolmogorov, A. N., 66, 70, 168, 170, 171, 174
Kreisel, G., xiii
Kripke, S., 57, 58, 76, 89, 135, 224, 225, 365

L

Lambek, J., 155, 156, 231, 310, 337, 338
Lehmann, D., 356
Leibniz, 162
Leinfellner, E., xiii
Levy, M., xiii
Lopes, M. A., xiv
Lorenzen, P., 22, 28, 34, 221, 222

M

Mahr, B., 125, 182, 203, 235, 320, 324, 326, 327, 331, 332, 355
Maibaum, T. S. E., xiii, 52, 56, 192, 252, 282, 329
Martin-Löf, P., xiii, xv–xviii, 5, 7–9, 13, 15, 18, 26, 27, 29, 44, 48, 49, 53, 54, 56, 67, 68, 70, 71, 83, 84, 95, 124, 141, 145, 148–153, 157, 160–175, 179, 180, 183–185, 187–191, 194, 196–201, 204, 205, 207, 208, 212, 214, 219, 225, 226, 232, 233, 237–240, 242–244, 247–250, 253–256, 258–263, 267, 274–280, 282, 283, 286, 287, 309, 314, 351, 352, 354, 356, 360–362, 365, 367, 369
Mauthner, F., 82, 98, 101–105, 115–117, 367
McCarthy, J., 124
Mendler, N. P., 277, 281, 283, 285
Miller, D., xiii
Moravia, S., xiii
Morris, J., 180, 181
Myhill, J., 158

N

Niskier, C., xiv
Nordström, B., 167, 185, 191, 198, 204, 205, 209, 249, 251, 355

P

Panangaden, P., 277, 281, 283
Pequeno, M., xiv
Petersson, K., 167, 185, 191, 198, 204, 205, 209, 251, 355
Peyton-Jones, S., 153, 154, 337
Pitt, J., xiv
Poigné, A., xiii
Prawitz, D., 26, 27, 29, 31–33, 36, 37, 39, 43, 44, 46, 49–56, 58–62, 65, 70, 75, 90, 91, 93, 94, 121, 122, 136, 145, 147–149, 152, 161, 174, 187, 221, 223, 233, 237, 257–259, 261, 360
Prior, A., 260

Q

Quesada, F. M., 107

R

Rice, L., xiv
Russell, B., 30, 86, 88, 102, 105, 107–109, 111, 129, 131, 208, 310, 361
Ryan, M., xiv, 146

S

Sadler, M., xiii, 146, 329
Sambin, G., xiii, 167, 204
Schroeder-Heister, P., 44–46, 56, 60, 61
Scott, D., 22–25, 175, 356
Scott, P. J., 155, 156, 231, 310, 337, 338
Seldin, J., 273, 274

Sextus Empiricus, 103
Sierpiński, W., 25
Smith, J., 167, 185, 191, 198, 204, 205, 209, 250, 251, 268, 355
Smith, R., xiv
Smyth, M., xiii, 187, 266, 356
Smythies, Y., 122, 130
St. Augustine, 4, 84, 116
Strecker, G., 154
Sundholm, G., 42, 60
Szabo, M., 30

T

Tait, W. W., 149, 189
Tam, C. C., xiv
Tarski, A., 227
Taylor, C., xiv
Thatcher, J., 181
Toulmin, S., 63
Turner, R., xiii
Turski, W., 192, 329

V

Veloso, P. A. S., xiii, 252, 329

W

Wagner, E., 181
Waismann, F., 92, 94, 101, 144, 161
Weiler, G., xiii, 103, 105, 115
Wittgenstein, L., xv–xvii, 3, 5–9, 11–15, 20, 21, 27, 28, 30, 31, 61–64, 68, 69, 71, 74–78, 80–109, 111, 112, 114–123, 125–138, 141–146, 150–152, 158, 159, 161, 175, 180, 184, 212–217, 219–227, 230, 231, 233, 244, 248, 255–257, 259, 273, 286, 318, 319, 357, 358, 360, 365, 367
von Wright, G. H., xiii
Wright J., 181

Z

Zermelo-Fraenkel, 207
Zucker, J., xiii

Index of References

A

Aczel 1980, 13, 19, 35, 125, 153, 166, 214, 231, 243, 245, 277, 285, 286, 318, 333, 334, 336
Allen 1978, 340, 341
Ambrose (ed.) 1979, 30, 91

B

Backhouse 1985, 260
Backhouse et. al. 1989, 234
Barendregt 1981, 9, 46, 124, 276, 321, 322, 345, 346, 364
Beeson 1981, 158, 245–247
Beeson 1985, 247, 330
Beiträge, 102, 104, 115
Beiträge I, 101, 103, 104, 116, 117
Beiträge III, 105
Bishop 1967, 163, 255, 360
Bishop 1970, 158, 278
Blue Book, 7, 82, 101, 106, 114, 115, 137, 214
Brown Book, 135

C

Church 1936, 11, 145
Church 1938, 206, 281
Church 1941, 145, 151, 231
Confessions, 4
Constable & Mendler 1985, 277, 285
Constable et. al. 1986, 153
Curry 1951, 166

D

Dedekind 1890, 271
Dedekind 1901, xviii, xix
Diamond (ed.) 1976, 127
Dummett 1976, 75, 121, 136, 216–218, 223, 365
Dummett 1977, 75, 121, 174, 223
Dummett 1980, 223
Dummett 1982, 15, 184, 224, 367
Dybjer 1985, 47, 124
Dybjer 1987, 277, 281

E

Ehrig & Mahr 1985, 125, 182, 203, 235, 320, 324, 326, 327, 331, 332, 355

F

Feferman 1979, 158
Fitting 1981, 60
Frege 1879, 161
Frege 1893, 150
Frege 1914, 3, 16, 222, 316
Friedman 1977, 158
Furth (ed.) 1964, xx, 150, 165

G

Gödel 1958, 149, 157
Geach (ed.) 1988, 134, 214
Gentzen 1935, 30, 145
Girard 1971, 150, 158
Goldblatt 1982, 1
Grundgesetze, xx, 150, 165
Guttag 1977, 203, 236, 237, 326, 353

H

Hallett 1977, 106
Hamblin 1970, 104
Hamblin 1987, 221
Henderson 1980, 344, 345
Herrlich & Strecker 1973, 154, 155
Hindley & Seldin 1986, 274
Hintikka 1983, 28, 78, 145, 228
Hintikka 1987, 79, 218
Hoare 1969, 1
Howard 1980, 158

I

Investigations, 14, 64, 68, 69, 76, 77, 82, 85, 89, 93, 101, 103, 105–107, 113, 114,

116, 117, 126, 128, 133, 137, 138, 142, 213, 220, 221, 224, 255, 359–361

J

Janik & Toulmin 1973, 63

K

Kalmár 1940, 22
Kolmogorov 1932, 168, 171
Kripke 1982, 76, 89, 224, 365

L

Lambek & Scott 1986, 155–157, 231, 310, 338, 339
Last Writings I, 101, 125, 130, 131, 142, 143
Lectures and Conversations on APRB, 72, 122, 123, 130
Lehmann & Smyth 1981, 356
Letters to RK&M, 105, 108, 213, 220
Lorenzen 1935, 22
Lorenzen 1969, 28, 221, 222
LW&VC, 92, 94, 95, 101, 162

M

Maibaum, Sadler & Veloso 1981, 329
Martin-Löf 1970, 67
Martin-Löf 1971a, 67
Martin-Löf 1971b, 67, 149
Martin-Löf 1973, 67
Martin-Löf 1975a, 67, 145, 151, 163, 169, 170, 194, 196, 207, 240, 258, 262, 282, 309
Martin-Löf 1975b, 67, 189–191, 274
Martin-Löf 1982, xviii, 67, 163–165, 167, 170, 171, 242, 255, 278
Martin-Löf 1984, 13, 48, 56, 67, 84, 124, 141, 148, 163, 167–170, 172, 183, 184, 188, 196–201, 204, 207, 208, 237, 239, 240, 248, 250, 258, 259, 261, 262, 267, 274–276, 278, 279, 351, 352, 354, 362
Martin-Löf 1985, 67, 152, 160, 161
Martin-Löf 1987, 68, 71, 175
Martin-Löf 1989, 282
McCarthy et. al. 1962, 124, 342
Mendler, Panangaden & Constable 1986, 277, 281
Morris 1973, 181
Myhill 1975, 158

N

Nordström 1986, 249
Nordström, Petersson & Smith 1986, 167, 185, 191, 198, 204, 205, 209, 251, 355
Notebooks, 87, 96, 106, 109–112, 143, 213

O

On Certainty, 90, 99, 106, 135

Q

de Queiroz 1987a, 50
de Queiroz 1987c, 13, 20, 31, 256
de Queiroz 1988, 37, 50, 52, 56, 61, 244, 256
de Queiroz 1988a, 256
de Queiroz 1988b, 256
de Queiroz 1988e, 5, 14, 28, 62, 71, 151
de Queiroz 1988f, 256
de Queiroz 1989b, 137
de Queiroz & Maibaum 1987a, 52, 56, 244
de Queiroz & Smyth 1988, 266
de Queiroz & Smyth 1989, 187
de Queiroz & Smyth 198?a, 276

P

Peyton-Jones 1987, 154, 337
Philosophical Grammar, 86, 94, 106, 131, 132
Philosophical Remarks, 83, 131, 132
Philsosophical Remarks, 94
Prawitz 1965, 32, 58, 59, 147, 148, 261
Prawitz 1971, 33, 54, 147, 261
Prawitz 1977, 31, 51, 53, 94, 121, 136, 257, 259
Prawitz 1978, 37, 39, 56, 233
Prawitz 1980, 61, 62, 152, 161, 223
Prawitz 1987, 65, 70
Prior 1960, 260
Prototractatus, 109, 110

R

Remarks on Colour, 97
RFGB, 27, 357
RFM, 90, 96, 106, 112, 132, 259
RPP I, 72–74, 129
RPP II, 21, 74, 123, 129, 130
Russell 1908, 310

S

Schroeder-Heister 1984, 56
Scott 1970, 175
Scott 1976, 22, 23, 356
Scott 1981, 22, 23, 356
Sierpiński 1958, 25
Smith 1983, 250, 268
Sundholm 1986, 60
Szabo (ed.) 1969, 30

T

Tait 1967, 149, 189
Thatcher, Wagner & Wright 1982, 181
Tractatus, 14, 68, 76, 86, 88, 89, 94, 102–107, 109–113, 131, 143, 220, 248, 358
Turski & Maibaum 1987, 192, 329

V

Veloso 1985, 252

W

Wörterbuch der Philosophie, 102
Weiler 1970, 103, 105, 115, 117

Z

Zettel, 85, 86, 97, 106