DENOTATIONAL SEMANTICS IN THE DEFINITION OF NATURAL LANGUAGE USES

João Bernardo de Sena Esteves Falcão e Cunha

July, 1989

Department of Computing
Imperial College of Science, Technology and Medicine
London SW7

A thesis submitted for the degree of *Doctor of Philosophy* of the University of London and for the Diploma of Membership of the Imperial College

To Maria João

ABSTRACT

Given a theory of computation such as Scott and Strachey's, and Wittgenstein's analysis of grammar and meaning, we investigate the consequences of using them when defining natural languages to be utilized in communication with computers. This view assumes that the mathematical study of language is divided into syntax, the study of the form, and into semantics, the study of the use. It also assumes that a well known notation and methodology is essential for successful engineering and, consequently, Denotational Semantics, *DS*, has been chosen for the mathematical framework.

We introduce versions of Montague Semantics, *MS*, and Barwise and Perry's Situation Semantics, *B&PS*, with computational characterizations given in the framework of *DS*, so we can compare these two mathematical approaches to semantics of natural language. We show how Montague's compositional treatment of intentionality should be simplified, by showing that the cap-cup operators are counter-intuitive and unnecessary in his definitions of English, and showing how to eliminate the meaning postulates. The main advantage is that when translating Montague English we will never need higher order quantification. We also propose a functional definition of meaning for *B&PS*. This treatment is shown to suffer in general from omniscience, one of the main problems which *B&PS* was supposed to solve. We discuss Hintikka's original criticism of Possible-Worlds Semantics and modal logics in the treatment of attitudes, and we find it wanting from a computational standing.

The simplifications to *MS* and the presentation of *B&PS* are examples of how *DS* can be used to construct computational grammars for natural language. We believe that the main difference between the two approaches is the choice of the domain of meanings for sentences, intensional booleans in *MS* and situation constraints in *B&PS*. We argue that the choice for this domain is methodological and justified depending on application, determined by the context of the natural language communication. We stress the importance of the knowledge acquisition process in definitions of grammars, and we believe that more research should be undertaken on relating this model of grammar with automatic models of learning such as the ones given by connectionism. As a guiding principle we only accept grammars defined with functions which are representable and learnable by computers as connectionist machines.

Valeu a pena? Tudo vale a pena
Se a alma não é pequena.
Quem quer passar além do Bojador
Tem que passar além da dor.
Deus ao mar o perigo e o abismo deu,
Mas nele é que espelhou o Céu.

Was it worth while? It is worth while, all,
If the soul is not small.
Whoever means to sail beyond the Cape
Must double sorrow — no escape.
Peril and abyss has God to the sea given
And yet made it the mirror of heaven.

Fernando Pessoa, *Mensagem*

O binómio de Newton é tão belo como a Vénus de Milo.
O que há é pouca gente para dar por isso.

Newton's Binomial Theory is as beautiful as the Venus of Milo.
The fact is, precious people care.

Alvaro de Campos, *Poesias*

> You, who wish to know, do you know you are? I know it. Whence are you? I know not. Do you feel yourself single or multiple? I know not. Do you feel yourself moved? I know not. Do you know that you think? I do.
>
> St Augustine, *Soliloquia*

ACKNOWLEDGMENTS

I would like to thank my supervisor, Jim Cunningham, for having introduced me to the subject of this work, for the helpful discussions we had and for the encouraging interest and support he showed during all of its development. Martin Sadler read substantial parts of this thesis and his comments on omniscience and compositionality were extremely useful in particular to produce the last versions of chapters one and four. Mike Smith also read substantial parts of this thesis and commented particularly on chapters two and three on the subject of computability for quantification and equality. Dov Gabbay was very much a source of illuminating ideas and particularly helpful were his comments on Montague semantics and finite hereditary languages. I would also like to express my thanks to the staff of the Computing Theory and Systems Technology Group and of the Department, in particular Jennifer Eastwood and Linden Rice.

I am also grateful to many friends and colleagues for helpful discussions, for their perceptive criticism and continuous cheerful support. I am particularly indebted for their careful reading of different versions of this typescript and for their suggestions. Many important improvements were made and if errors of fact or judgement remain they are of my own making. In particular I should mention Jeremy Pitt and Stuart Kent of the natural language group for many technical and

philosophical discussions on the vast subject of natural language analysis, and for reading and commenting on the entire typescript. Michael Gera and the other members of the neural computing group, were tremendously helpful in showing what connectionist theories are all about and making sure my comments on that subject were sensible. Ruy de Queiróz enlightened me in various aspects on the foundations of logic and computing sciences. Pablo de la Quintana deserves my thanks for his clarifications on modal logic and automatic theorem proving. Mark Ryan helped tremendously in clarifying my thoughts on computability and he read and commented on all of the typescript. From Logic Programming to Programming in Logic, I would also like to thank Reem Bahgat, Marcos Costa, David Fuller, Mark Reynolds, C. C. Tam and Silvana Zappacosta.

Some of the central ideas in this thesis were outlined to a small group at an international meeting in logic and linguistics held at Stanford University in August 1987. Their questions and criticisms came at a crucial stage and chapter four resulted from an attempt to understand the criticisms Barwise's Situation Semantics levels at Montague Semantics. I remember in particular two extremely helpful discussions with Stanley Peters and Fernando Pereira. I would also like to thank the anonymous referees of the *FGCS'88* international conference for the very encouraging comments on an early draft of what has become chapter four.

Without his knowledge Douglas Hofstadter made my interests focus on Logic and Computing Sciences. Bill Swan from *ICI* was very influential in suggesting the analysis of natural language as a subject for my thesis, when I was in very early stages of this work.

I should also mention Professors Vasco Sá, Rui Guimarães, Carlos Moreira da Silva and my colleagues of *Universidade do Porto* in particular of *GEIN* and *DEE*. I would like to thank them for their effective support and continued interest. Their enthusiasm and dedication has shown me the value of academic research and how to be able to carry this work further.

My parents and my brother deserve these special words for *everything*. In particular for all the questions on the subject of this thesis, many of which are still waiting for sensible answers.

This work would not have been possible without the continuous financial support of the Portuguese *Laboratório Nacional de Engenharia e Tecnologia Industrial*. I am also grateful to *Universidade do Porto* for giving me leave of absence for the last two years.

LIST OF CONTENTS

ABSTRACT 3

ACKNOWLEDGMENTS 5

LIST OF CONTENTS 8

LIST OF ABBREVIATIONS 15

THESIS OVERVIEW 16

CHAPTER 1

INTRODUCTION

1.1 Overview 20

1.2 Philosophy of Meaning for Natural Language 21

Natural and Artificial Languages 21

Montague's Thesis 22

Wittgenstein's Meaning as Use 23

Connectionism and Learning 24

1.2.1 Positivistic Views on Meaning 25

Logical Positivism 25

Use rather than Meaning 26

Hintikka and Malmgren's Criticisms 26

Psychology and Connectionism 27

1.2.2 Compositionality of Meaning 28

Possible Worlds, Indices and Contexts of Use 29

Situations and Contexts of Use 30
1.2.3 The Study of Language and the Definition of Grammar 31
1.3 Mathematics and Computation for Natural Language 33
1.3.1 Formality and Formal Systems 34
Formal Reduction of Semantics 35
Formal Grammar 35
Denotational Semantics 36
1.3.2 The Typed λ-Calculus and Scott's Theory of Computation 37
Simply Typed λ-Calculus 38
Constructively Typed λ-Calculus 38
1.3.3 Complexity in a Formal Analysis 39
The Formal Syntax 40
The Formal Semantics 40
Formal and Connectionist Systems 41
The Computational Meaning of Quantification 42
Generalized and Binary Quantifiers 43
1.3.4 Computational Linguistics 44
Types and Types of Sentences 45
Traditional and Formal Linguistics 47
1.4 Conclusion 48

CHAPTER 2

COMPUTABLE NOTATIONS AND DENOTATIONS FOR MONTAGUE'S *PTQ*

2.1 Introduction 49
2.2 Notation, Primitives and Computable Constants 51
2.2.1 Domains and Computability 52
2.2.2 The Boolean Domain 52
2.2.3 Constructions on Domains 53
2.2.4 Computable Elements 54
Logical Connectives 54
Equality 54
Quantification 55

2.3 A Semantically Clear Version of *PTQ* 60
2.3.1 Montague's Syntax of English 60
Syntactic Categories 61
Basic Expressions 61
Phrases 62
Example from *LR*-Grammars 64
2.3.2 Montague's Syntax of Intensional Logic *IL* 64
Types 64
Symbols 65
Expressions 65
2.3.3 Domain Spaces and Elements for *IL*'s Semantics 66
Entities and Contexts of Use 66
Montague's Modal and Temporal Logics 67
Meaning in Montague 67
2.3.4 A Domain Semantics of *IL* 69
Semantic Interpretation 69
Turner's Principle on Interpretations 70
Semantic Extension and Intension 70
Logical Equivalence and Meaning Postulates 71
2.3.5 The Semantics of *M-NL* through Translation into *IL* 73
Translation on Syntactic Categories 73
Translation on Phrases 74
Extension and Intension of Phrases 76
2.4 Intentionality in *PTQ* 77
2.4.1 Montague's Definitions 77
Extensional and Intensional Meaning for Expressions in *IL* 78
Intensional Operator 79
Extensional Operator 79
2.4.2 Using the Intensional Operators 80
Dowty's Example 80
Dowty's Conclusion 82
Gallin's Observation Criticized 82

2.4.3 The Double Formalization of Intentionality in *PTQ* 83
Failure of Compositionality in *NL* 83
Compositionality in *IL* 84
Translation from nl into *IL* 85
Compositionality Restored in *NL* 86
2.4.4 Excessive Expressiveness of *IL* 87
2.5 Conclusions 88

CHAPTER 3

A SIMPLIFIED TREATMENT OF MONTAGUE'S ENGLISH

3.1 Introduction 89
3.2 The Simplified Intensional Language 91
3.2.1 Syntax of the Intensional Language *L* 92
Types 92
Symbols 92
Expressions 92
3.2.2 Semantics for Language *L* 93
Meanings 94
Semantic Interpretation 96
3.2.3 About the Semantic Function 97
Semantic Extension and Intension 97
Meaning Postulates 98
Examples 99
Surviving without Meaning Postulates 100
3.3 The Simplified Translation 101
Translation on Syntactic Categories 102
Translation on Phrases 102
Is the Intensional Translation of English High-Order ? 106
3.4 Isomorphic Semantics for English 106
3.5 Computable Grammars without Meaning Postulates 111
Overview 112
3.5.1 Some Basic Characteristics of Montague's Framework 113

Compositionality, Logic and Postulates 113
Limited Changeability 114
3.5.2 Properties Required for a Computable Grammar 114
Computable Syntax 115
Computable Semantics 116
3.5.3 Computable Grammars to Approximate English 117
About the Notation 118
Approximation $^{0}English$ 119
Examples 122
Comments 123
Approximation $^{1}English$ 123
Comment on Meaning Postulates 125
Examples 126
3.6 Conclusion 127

CHAPTER 4

TOWARDS COMPUTABLE DENOTATIONS FOR SITUATION SEMANTICS

4.1 Introduction 129
Objectives 130
Overview 131
4.2 Barwise & Perry's English 133
4.3 A Formal System for Situation Semantics: *SSL* 136
4.3.1 Primitive Domains 137
4.3.2 Non Primitive Basic Domains 138
Situations 139
Notation and Abbreviations 140
Properties of Situations 141
Discourse Situations and Connections 142
Constraints and Structures of Situations 143
4.4 The Interpretation of *DFA-NL* with *SSL* 144
4.4.1 Interpretation of Lexicon 145
4.4.2 Semantic Features and Settings 145

4.4.3 Semantics of *DFA-NL* 146
4.5 About Omniscience and Perception 153
Compositional Semantics 153
Omniscience 153
Overview 154
4.5.1 Formal Semantics, Synonymy and Omniscience 155
Compositionality and Semantic Omniscience 156
Example with Sentences 158
Example with Statements 159
4.5.2 Barwise's Puzzle of Logical Equivalence 160
The Puzzle of 'Seeing' 161
Logic and the Choice of Formal Logic 163
4.5.3 On a Formal Logic for Perception 166
Objectives 166
Other Propositional Words and Phrases 167
Hintikka's Treatment of Propositional Attitudes 167
Axiomatization 168
Semantics 169
4.5.4 On Formal Inference and Interpretation for Perception 170
Formal Representations and Meaning 170
4.6 Conclusions 173

CHAPTER 5
CONCLUSIONS

5.1 On Formal Analysis of Natural Language 176
Montague's Operators and the Postulates 176
Computability of Grammar 176
Propositional Attitudes, Perception and Formal Inference 177
Language Use instead of Meaning and Grammar 177
5.2 Summary of Results 178
5.3 Recommendations For Further Study 179
Deduction, Induction and Quantification 179

Logical Grammatic Approximations and Connectionism 180
Applications to other Syntactic and Semantic Objects 180

REFERENCES 181

LIST OF ABBREVIATIONS

Abbreviation	*for*
B&PS	Barwise and Perry Semantics
DFA-NL	Determiner-free Aliass Natural Language, Barwise and Perry[1983]
DS	Denotational Semantics
IL	Intensional Logic (chapter 2)
L	Simplified Intensional Logic (chapter 3)
M-NL	Montague's Natural Language
MS	Montague Semantics
NL	Natural Language
PI	'Philosophical Investigations', Wittgenstein[1953]
PTQ	'The Proper Treatment of Quantification in Ordinary English', Montague[1970d]
PWS	Possible Worlds Semantics
S&A	'Situations and Attitudes', Barwise and Perry[1983]
SSL	Theory of Situations, (from *Situation Semantics Logic*)
UG	'Universal Grammar', Montague[1970c]

THESIS OVERVIEW

Chapter 1 is an account of the philosophical analysis of meaning for natural language and of its mathematical counterparts. The main purpose of this chapter is to place Montague and Barwise & Perry's treatments of English in this century's philosophical and mathematical developments. The second purpose is to justify the choice of a computational approach and treatment, and of mathematical frameworks which have intuitive computationally treatable and tractable properties. We argue that the treatments of Montague and Barwise & Perry follow the ideas of early Wittgenstein or late Russell on the philosophical and mathematical analysis of language. We think that they take an absolute approach to the classification of syntactic objects. For instance, Montague semantics is mainly a treatment of declarative sentences and situation semantics is mainly a treatment of indicative sentences. Furthermore, we think that due to philosophical motivations their analysis of natural language does not give a fundamental importance to constructive or computational properties. They are both motivated by classical principles originating from Tarski in Montague semantics and from Hintikka in situation semantics. We argue that Scott and Strachey's Denotational Semantics, as a methodology and a mathematical theory of computation with explicit notational conventions, provides what we need for the constructive definitions of language uses. We want to stress the computational aspects of Scott's domain theory not its ability to model information. We also believe in using Wittgenstein's *use* instead of *meaning* when speaking of natural language. Meaning has infinitistic connotations while use has finitistic or computational ones. The reduction of meaning to use requires grammar to be defined as syntax and semantics, with traditional pragmatics being a part of the latter. We also argue that mathematical discussions of human cognitive processes, such as *learning information*, must be related to constructive models for those human cognitive processes. Connectionist theories provide, we believe, the best examples for such models.

Chapter 2 is a review and introduction to the main features of *PTQ* on which the discussion of the following chapters will be based. It is also a summary of the notational conventions that will be used and of possible computational definitions for logical constants and quantification. We start by introducing a version of Montague's treatment using denotational semantics with an intensional logic embedded on a typed λ-calculus. This version employs a similar framework to the one used by Turner in his treatment of nominalization, but with an improved notation, where meaning functions have proper names. We examine its computational characteristics, the definition of sentence and of its meaning, the definitions of contexts of use and the treatment of intentionality. We discuss the use of the cap-cup operators and criticize Gallin's observations. By proposing a correct description, in terms of functional operators, we explain some of their problems and expose their non intuitive usages. As a side result we believe that our description makes it easier to *read PTQ*, in particular to understand the use of the meaning postulates. We also show why those operators are unnecessary in the intensional treatment of English, and point out some of the necessary steps to simplify it in order to have a semantic analysis with better known and more tractable computational properties. We show that parts of the expressive power of *IL* are never used. This observation for instance lead us into asking again whether the minimal formal system necessary to classify Montague's English in *PTQ* is a higher order logic with its computationally intractable properties. This question will be negatively answered in the following chapter. We also discuss the computational treatment of determiners when interpreted as logical quantifiers over primitive domains.

Chapter 3 introduces the simplification to *PTQ* suggested in chapter 2 of not using the cap-cup operators. It is then shown that those operators are unnecessary in that treatment of English. We prove that the resulting semantics is isomorphic to the one presented in chapter 2. This brings the whole semantics closer to a simple Modal or Temporal Logic, for which direct theorem proving starts to emerge. This logic is embedded in a constructively typed λ-calculus system and has only quantification over domains of primitively given entities and indices. We defend the view that

meaning postulates, which introduce higher-order quantification in PTQ, are unnecessary and that they should to be avoided. In order to show how to eliminate the postulates, we define two approximations to English using computable grammars. The second approximation captures the intensional aspects of PTQ satisfying the requirements of the postulates. We believe the notion of computational grammar to be important because for instance it satisfies principles of economy when defining the meaning of the lexicon.

Chapter 4 starts with a presentation of a simplified version of situation semantics inspired by Barwise & Perry's *Situations and Attitudes*. We again use Denotational Semantics and typed λ-calculus. Basically we introduce primitive objects and define semantic domains on which syntactically defined objects such as sentences and utterances are to be interpreted. We provide a compositional definition for *ALIASS*, the natural language defined in one of the appendices of *Situations and Attitudes*. We then extend the definition of this language to take account of attitude and perception sentences which embed sentences. This allows us to compare this presentation of Situation Semantics with Montague Semantics and is mainly an attempt to understand Barwise's claim that no logical omniscience is observable in the former. We believe that the responsibility for that is not possible worlds semantics, as Barwise originally stated by referring Hintikka's work. We show that in general omniscience is present in situation semantics, but as in the definition of strong equivalence between statements no quantification is made on the interpretation, it is possible to avoid it in particular cases. We discuss the role of omniscience by using the formalization of perception sentences, which we believe is more objective than discussions on using propositional attitudes. In this respect we discuss Hintikka's views on formalization and we argue that his platonistic view on the formalization of propositional attitudes has been an obstacle for an objective logic of attitudes and perception.

Chapter 5 is a summary of the main technical and methodological conclusions which we believe to be original in this work. For instance we believe that our observation on Montague's operators is the first correct one, and together with our criticism of constraints on interpretations, provides the most elegant and

computationally tractable definition of *PTQ*. If it is accepted that our formalization of situation semantics is acceptable, then our proof of it having its own omniscience, contributes to accepting a constructive analysis of natural language, very unlike the one defended by Hintikka. Our analysis provides new constructive views on grammar of natural language and new reasons to reject Hintikka's philosophical criticism of Wittgenstein. We believe the logic behind the construction of computable grammars, as approximations to natural language, has a direct relation with Popper's logic of scientific discovery. This chapter also indicates some of the areas which we would like to see clarified and we would like to investigate further. If we accept that to formalize a language and its meaning, we must know its use, it is obvious that in order to computationally process natural language we must formalize the possible contexts of use and formalize the possible uses of syntactic expressions. For real applications, this requires studies in the direction of more complex syntactic expressions other than sentences, and in the direction of uses of syntactic expressions other than declarative or indicative. Formal systems, such as logics, provide us with certainty and reproducibility on deduction and reasoning from formal information. Connectionist theories provide us with a computational model for induction, but yielding only a probability, not certainty. We believe that the links between Formal System Theory and Connectionist Theory should be further investigated, particularly by studying how they approach the analysis and synthesis of natural languages.

> How should we explain to someone what a game is? I imagine that we should describe *games* to him, and we might add: "This *and similar things* are called 'games' ". And do we know any more about it ourselves? Is it only other people whom we cannot tell exactly what a game is? — But this is not ignorance. We do not know the boundaries because none have been drawn. To repeat, we can draw a boundary — for a special purpose. Does it take that to make the concept usable? Not at all! (Except for that special purpose) No more than it took the definition: 1 pace = 75 cm. to make the measure of length 'one pace' usable. And if you want to say "But still, before that it wasn't an exact measure", then I reply: very well, it was an inexact one. — Though you still owe me a definition of exactness.
>
> Wittgenstein[1953], §69, pp. 33e

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

In this thesis we use written Natural Language, to be more specific some type or dialect of written English. In this chapter we provide a very brief summary of the main attempts to provide Natural Language with a formal logical and mathematical analysis. Our main interest in doing so is to expose some of the philosophical and mathematical ideas on which Montague Semantics and Situation Semantics have developed. This chapter can also be seen as a short foundational discussion on the possibility and advantages of a formal analysis of natural language, particularly when we desire such analysis to be computationally treatable and tractable.

We start with a simple overview of the philosophy of meaning, mainly the positivistic view, and we discuss some basic notions such as compositionality and grammaticality. In a second part we briefly discuss the role of formality and of different types of formal systems, and of their computational properties. We argue about our use of Denotational Semantics as a functional theory and metalanguage. We end with a short reference to current application-oriented work on Computational Linguistics, and very briefly indicate some relations with this work.

1.2 PHILOSOPHY OF MEANING FOR NATURAL LANGUAGE

Natural and Artificial Languages

The distinction which is often made between natural languages and artificial languages is we believe somehow inappropriate. We use the word 'artificial' for computer languages and mathematical logic languages because they have been constructed to enable us to be precise or exact in a mathematical sense of the word. However natural languages have been invented by men and in that respect are un-natural or artificial. They have also been used successfully to express and communicate information across time and space. For instance, our present knowledge of the early Egyptian civilization is due to Champollion's interpretation of natural language forms expressed in hieroglyphs. As Karl Popper has argued, knowledge growth is closely associated with the use of natural languages [1]. Our claim that natural languages are as artificial as formal languages is equivalent to saying that in both exactness and ambiguity are variable and dependent on their use. We believe that this conclusion is supported for instance by the thesis explicit in the works of Richard Montague. Intuitively Ludwig Wittgenstein's observation in the citation above is criticizing the general idea that natural language is not exact or precise.

[1] Popper[1972], chapter 3, distinguishes four functions in human language, the expressive, communicative, descriptive and argumentative functions, of which the first two are shared with animal languages. We could say that all functions have been used across time and space.

Nevertheless, most of the mathematicians and analytical philosophers of this century, starting with Frege and through Tarski, have recognized a fundamental difference between artificial and natural languages, the latter being impossibly incoherent or having inherent ontological deficiencies. The solution could rest on developing new artificial languages free of all the problems with natural languages. However this view is being continuously challenged.

Montague's Thesis

The first to express the thesis and to give objective evidence that there is no important theoretical difference between natural languages and artificial formal languages was Richard Montague in the early 1970's (see for instance Thomason[1974]). Montague's Thesis, as we will call it, assumes that Linguistics and in particular the study of meaning should be regarded as a branch of Mathematics. Following Frege and Tarski, Montague adopted a classical referential theory for meaning which maintains that any meaningful expression refers to some mathematical object [2]. More recently, Barwise and Perry[1983] claim more evidence in support of Montague's thesis while building on what is claimed to be an improved ontology for meaning. This improved theory is based on the philosophical ideas of Wittgenstein[1921], Russell[1918] and Austin[1950]. Barwise and Cooper[1981], pp. 204, are explicit in attributing to Montague *the revolutionary idea* [...] *that natural language is not impossibly incoherent, as his teacher Tarski had led us to believe.* Before that Popper[1960], pp. 223, had already made a more general remark concerning Tarski's theory, which can be interpreted as pointing in the same direction: *The view that his* [Tarski's] *theory is applicable only to formalized languages is, I think, mistaken. It is applicable to any consistent and even to a 'natural' language* [...].

We think that Montague's Thesis is essential to justify the study of natural language using the methods and tools available in computer science. In particular we believe

[2] Alston[1967] provides an excellent introduction to traditional theories of meaning, which can be grouped into three types, Referential, Ideational and Stimulus-response and how they relate with more modern views. Meaning and use, Rules and regularities, Word use and Synonymy are discussed.

that the methodology known by the name of Denotational Semantics offers the most natural framework for the computational characterization of Montague's work and of natural language. It provides a constructive understanding for the classical notions of sets and functions used by Montague. It offers a mathematical theory closely related with recursion theory to deal with infinitary objects through the use of finite approximations, and it offers a well known notational system.

Wittgenstein's Meaning as Use

Our understanding of meaning for natural language will be based on ideas put forward by Wittgenstein in his 'Philosophical Investigations' [1953]. There discussions of meaning are replaced by discussions of use: *Look at the sentence as an instrument, and at its sense as its employment* (§421, pp. 126e). We will accept Wittgenstein's statement in the sense that it is more useful in computational linguistics to be concerned with characterizations of use than of meaning. It seems to be more useful to use the word *use* than the word *meaning* for studying linguistic objects. Use has finitistic and operational connotations, while meaning has infinitistic and static ones. We stress that our understanding of *use* should not be confused with that of *useful.*

An important observation of Wittgenstein's *PI* is that the understanding of the use of words depends on the understanding of a language-game (see 1.2.1 and 1.3.4 below). Wittgenstein's ideas on language-games have been extremely fruitful. For instance *Game Theoretical Semantics*, as proposed by Hintikka[1976], results from a direct criticism of the ideas in *PI*. However we believe that Hintikka's rejection of Wittgenstein's finitistic attitude towards language-games, which according to him is defended in *PI*, is not acceptable from a computational point of view. Hintikka's rejection is not acceptable because we want to model human cognitive processes using computational implementations which have finite basis. Hintikka seems to disagree that the possible uses of language, language-games, have to be learnt *a priori*. The consequence of this for a mathematical approach is that it is necessary to start with language-games, or contexts of use, fully circumscribed and at least partially and finitely defined. We can use mathematical languages and techniques to

characterize possible uses or applications of words and languages in order to define its use relative to the application. But we have to provide a finite starting point on which to build. For instance, a formal system which models elements of some kind, must have a finite set of axioms and rules of inference which will provide a characterization for a possibly infinite number of elements. Similarly, a parallel computer network, 'neural', finitely captures the ability to recognize an infinite number of elements.

Connectionism and Learning

The connectionist theory can be regarded as an attempt to provide a model for cognitive human processes, for instance to explain how people learn and then answer questions based on what was learnt (Gera[1988]). A connectionist system goes through learning and answering phases. The two phases could be identified with function definition and function evaluation. The informative content of the function in such systems is also changing with each learning phase and it can be expected to change from states of less information to states of more information. If we do not consider probabilistic systems, such function can always be described as a finite state machine. We believe that it should be possible to compare the theory of connectionist systems with the theory of formal systems when applied to processing language. A fundamental observation in this respect is that in a connectionist paradigm the answer to some question, or the computation of an associated function, takes always a finite amount of time, even if the answer corresponds to a state of insufficient information. We will accept that an individual's understanding of the use of natural language can be identified with such a finite, although large, function which suffers incremental changes, as for instance we propose to do in chapter three, section 3.5. The present study, using formal presentations of natural language with a theory of computable functions as partial objects, such as domain theory, can be seen as an initial contribution for the comparison.

1.2.1 Positivistic Views on Meaning

The reduction of Linguistics to a branch of Mathematics implicit in Montague's Thesis can be seen as a natural development of attitudes towards philosophy and language initiated in the beginning of this century after Russell's recognition of Frege's work on logic.

Logical Positivism

The rational ideas of logical atomism and the Vienna Circle's positivism proposed a shift from metaphysical excesses in philosophy to a study centred around the mathematical analysis of ordinary language. The works of Wittgenstein[1921] and Russell[1918, 1924, 1950] provide the basic arguments on this influential view. Philosophy should *be concerned not with morals, but with the language of morals; not with science but with the language of science* (Devitt and Sterelny[1987]). Furthermore as Russell[1950], pp. 367 explains, *What is distinctive is attention to mathematics and logic, and emphasis upon linguistic aspects of traditional philosophical problems.* Boole[1854] but mostly Frege's [1879] 'Begriffsschrift' provided the foundation for much of this view when devising their logical calculus from the study of ordinary language. However, Frege and the logical positivists regarded the use of natural language as fundamentally inadequate in achieving the goal of precision. A logical or mathematical language had to be developed to eliminate the ambiguities inherent in the use of natural language. This was the objective of Whitehead and Russell's 'Principia Mathematica' [1910] with the known consequences shown by Gödel[1931] [3]. This originated a reappraisal of formalistic attempts on mathematics.

[3] Popper[1945], in note 8 to chapter 24 of Vol. 2, provides an interesting view on uncritical logical positivism in particular of Wittgenstein's 'Tractatus', by equating examples of that theory with the paradox of the Liar and indirectly to Gödel's proofs. For instance in pp. 355, on Wittgenstein's proposition that 'The totality of true propositions is . . the totality of natural science.': *Since this proposition does not belong to natural science (but, rather, to a meta-science, i.e. a theory that speaks about science) it follows that it asserts its own untruth* [...].

Use rather than Meaning

In 'Philosophical Investigations' Wittgenstein[1953] regards this doctrine as too dogmatic and as Manno[1969] writes, he sets the *magna carta* of analytical or ordinary language philosophy. Wittgenstein proposes that there is not just one logical form in natural language but multiple dimensions in which to characterize its use:

> It is not our aim to refine or complete the system of rules for the use of our words in unheard-of-ways.
>
> For the clarity that we are aiming at is indeed complete clarity. But this simply means that the philosophical problems should completely disappear.
>
> The real discovery is the one that makes me capable of stopping doing philosophy when I want to. — The one that gives philosophy peace, so that it is no longer tormented by questions that bring itself in question. — Instead, we now demonstrate a method, by examples; and the series of examples can be broken off. Problems are solved (difficulties eliminated), not a single problem.
>
> There is not a philosophical method, though there are indeed methods, like different therapies.
>
> Wittgenstein[1953], §133, pp. 51e

Under this view, it only makes sense to reduce philosophy to the study of language or of its grammar: *our investigation* [...] *is a grammatical one* (§90). Its consequences, as Kenny[1973] (pp. 205) summarizes, are that words like doubt, knowledge and certainty are to be characterized through rules of use in language-games. The collection of rules would be the grammar of a language-game, which needs to be learnt, and outside which they are meaningless. As the classical referential study of meaning for language was so deeply concerned with relations with metaphysical objects or relations with reality, only characterizations of use would be valid in avoiding the infinite trap of a relational theory of meaningwhich always creates a new language to explain the old, as Manno[1969] on *Analytic Philosophy* tells us .

Hintikka and Malmgren's Criticisms

Hintikka[1976], pp. 22, criticizes Wittgenstein's finitistic position on understanding language-games and calls it a *ultra-constructivistic doctrine*. According to

Hintikka's interpretation, language-games are a *link between language and reality*, and should be characterized as *games in the precise sense of mathematical game theory*. However he argues that we must adopt a *realistic (Platonistic) attitude* [which is] *deeply antithetical to Wittgenstein's holism*. We believe that this position is at the heart of the attitude Hintikka[1975] takes on logically possible worlds. It ultimately leads Barwise[1981] to reject the modal logician's approach to model-theoretic semantics.

In a different perspective, Malmgren[1971] criticizes Wittgenstein's replacement of the ordinary conception of *truth* by *correct use* on the grounds that there is no *regular empirical connection* between word and object which would explain a judgement on *correct use*. Hintikka[1976] criticizes ostensive definitions of names as *logical Adamism*, and also elucidation, a similar process for language-games on the same anti-finitistic grounds. As pragmatics, the study of the use, is not regarded by Hintikka an empirical science, a psychological explanation is not easily accepted as a working hypothesis for what correct use is.

Psychology and Connectionism

The interpretation of experiments in the psychology of learning has shown that children can only learn to recognize and use words if these words are part of routines in which they act as participants. A routine is a sequence of physical and linguistic actions performed several times in an environment where the children is present (as Gonçalves[1988]). If we identify these routines with language-games, then what Wittgenstein says on understanding a word-rule as being derived from understanding a language-game seems to be perfectly acceptable, but not just as a mathematical explanation.

The extrapolation of the recent work on learning and understanding in the connectionist model, based on Rumelhart and McClelland's [1986] proofs of its generality, could provide the foundations for a finitistic approach to meaning such as the one Wittgenstein proposes. It provides the regular empirical connection missing in Malmgren and basically works by accepting ostensive definitions,

ostensive or as learning from examples. The strong limitations found by Minsky and Papert[1969] in simple perceptrons do not exist in the more recent complex connectionist systems. These may enable Wittgenstein's elucidation, or ostensive definition of language-games, to take place. Learning or ostensive definitions in these systems could also be seen as a form of induction on a finite number of examples. This would provide for Russell's ontology what he calls the *sensical* basis of logic. Induction would complement deduction (see Russell[1946], pp. 209). The importance of induction is clearly defended by Popper[1953] although definitely not as a basic principle of logic which is complementary of deduction.

1.2.2 Compositionality of Meaning

The mathematical referential theory of meaning for natural language starts when Frege[1892] gives an interpretation of classical linguistic objects in terms of concepts such as functions and arguments. Frege's analysis is oriented by a version of the principle of compositionality according to which the meaning of a given expression is a function of the meanings of its constituents and of the way they are composed. This principle is used for instance to reject a naive theory of reference because of intensional or oblique phenomena in language (see Frege[1892] or Alston[1967]). In order to conserve compositionality Frege decided that the meaning of an expression needs to be relativized to the context in which it occurs, the *context of use*. This relative value is usually named the sense or intension of the expression to distinguish it from its absolute value or extension in some particular context. For a declarative sentence Frege concluded that extension was to be identified with a truth value [4]. As more recently Wilfrid Hodges clearly proposes,

> The truth-value of a sentence in a situation is determined by the phrase-marker [syntactic structure] (underlying if need be) of the sentence together with the intensions of its constituent phrases; the truth-value can be calculated, at least in principle, by working out the intensions of the short constituent phrases first, then the longer ones in terms of the shorter ones, and so on up the phrase-marker, until the intension of the whole is

[4] We will argue that this identification works as a definition for a declarative sentence, in so far as we classify a sentence as being declaratively used when we can judge it with a boolean value. See on section 1.3.4 *Types and Types of Sentences* for further discussion.

> reached; the intension of the whole sentence immediately tells us the truth-value in a given situation.
>
> Hodges[1977], pp. 250

There are many reasons for accepting this principle and although it seems counter intuitive to some authors, see for instance Popper[1974], pp. 22, no reasonable procedure has ever been suggested to replace it. There has been a long discussion about the most correct and usable form of this principle, exposed for instance by Partee[1982]. However introspection seems to confirm that compositionality is present in the cognitive process of understanding a sentence (Hodges[1977], pp. 251). Compositionality is also essential for a question of computational economy in the choice for the representations of linguistic objects and the associated characterizations of meaning or use, and it is a form of the scientific principle that in order to know an object we need to know its parts.

Possible Worlds, Indices and Contexts of Use

Beginning in the late 60's, Richard Montague introduces tools from mathematical logic to give the most precise and successful form to Frege's proposal. For instance in 'Universal Grammar' Montague[1970c] preserves compositionality by introducing a pair of primitive sets J and I, so called respectively contexts of use and possible-worlds, with respect to which the meaning of expressions is valued. However he is aware of some of the problems in doing so:

> It is a slight oversimplification to call the members of I possible worlds. In connection with tensed languages, for instance, it is convenient to take I as the set of all ordered pairs consisting of a possible world and a moment of time, and J as the set of all complexes of remaining features of possible contexts of use."
>
> Montague[1970c], pp. 228

The last option is taken in his 'The Proper Treatment of Quantification in Ordinary English' [1970d], for short *PTQ*. Gallin[1975] in his introduction, refers to other possible definitions for indices, where each element can be thought of as a sequence $(t, p, w, s, ...)$ where t is the time, p is the spatial position, w the world, s is the speaker and the remaining coordinates specify, in general, various other aspects of the context of use. Lewis[1986], §1.4, argues that this separation of coordinates is

bounded for any practical purpose, and this is the same conclusion Barwise and Perry[1983], pp. 127, came to, when characterizing utterances. The choice of coordinates seems still to depend on the use of language. Lewis[1986] discusses the acceptability of using possible worlds as contexts of use from philosophical and technical points of view, and defends their use based on what he calls *modal realism.*

Once we have a definition of the contexts of use and knowledge of the relative values of simple expressions in the language for every context of use, the meaning of any expression is constructed following given compositional rules.

Situations and Contexts of Use

Situation Semantics, as in Barwise[1981] or Barwise and Perry's [1983] 'Situations and Attitudes', for short *S&A*, rejects a mathematical theory of meaning based on contexts of use defined with possible worlds. The methodological reasons are based in Austin and Russell's logical positivism and on technical reasons, such as logical omniscience, which are discussed for instance in Hintikka[1975]. As Barwise expresses,

> [...] The "possible-worlds" approach suffers from several well-known defects (logical omniscience, for example) and seems to some of us to rest on dubious metaphysical and epistemological assumptions.[...]
> This paper only hints at the technical details of situation semantics. It is rather the story of my own transition from looking at model theory for natural language in the tradition of Frege, Tarski, Davidson and Montague to a model theory more in accord with the ideas of Austin and (late) Russell, a theory in which sentences describe types of situations in the world, with truth an important but derivative notion.
>
> Barwise[1981], pp. 369-372

For illustration, Situation Semantics defines the meaning of an indicative sentence as a constraint between situations [5]. Its semantic treatment is based on a non well founded set theory based on Aczel's Hypersets [6]. Perry[1987] argues that Situations

[5] About indicative sentences see on section 1.3.4 *Types and Types of Sentences.*

[6] For an introduction to Hyperset Theory and its role in Situation Semantics see for instance Barwise and Etchemendy[1987] and Barwise[1984]. The original theory follows from Aczel's [1980] 'Frege Structures and the Notions of Proposition, Truth and Set'.

are better suited than possible worlds for a mathematical analysis of meaning. Situation Semantics claims to solve some problems with the acceptability of natural language inference by adopting a new approach to model-theoretic semantics. As we will see in chapter four, we will be claiming that it achieves those results by accepting a compositional definition of meaning, but rejecting a compositional definition of boolean meaning and therefore the standard definition of logical equivalence. In general the problem of omniscience is still present. Landman[1986] presents an alternative view of Situation Semantics and again claims to solve some puzzles in natural inference on perception which remained to be solved in *S&A*. Landman's account instead of using a theory of hypersets uses a form of the more familiar domain theory to model the partiality of situations and information. In this respect it is a step which makes possible our computable characterization of Situation Semantics. We believe however that omniscience is also present in Landman's account of Situation Semantics.

1.2.3 The Study of Language and the Definition of Grammar

The word *grammar* has various uses. In the framework of computational linguistics, some grammarians speak of grammar as embracing rules not only for syntax but for phonological, lexical, semantic and pragmatic specification as well. This view has permeated through to more classical grammarians:

> [...] Words must be combined into larger units, and grammar encompasses the complex set of rules specifying such combination. Meaning relations in the language system are the business of SEMANTICS, the study of meaning, and semantics therefore has relevance equally within lexicology and within grammar. Finally, the meaning of linguistic expressions when uttered in particular types of situation is dealt with in PRAGMATICS, which is concerned with the communicative force of linguistic utterances.
>
> Quirk et al[1985], pp. 12

Morris[1938] who is the source for Montague's definitions, divides the study of language, in syntax, semantics and pragmatics, and defines a language L as $L = L_{syn} + L_{sem} + L_p$. As Montague's summarizes,

> Syntax is concerned solely with relations between linguistic expressions; semantics with relations between expressions and the objects to which they refer; and pragmatics with

> relations among expressions, the objects to which they refer and the users or contexts of use of the expressions.
>
> Montague[1968], pp. 95

> The basic aim of semantics is to characterize the notions of a true sentence (under a given interpretation) and of entailment, while that of syntax is to characterize the various syntactical categories, especially the set of declarative sentences. It is to be expected, then, that the aim of syntax could be realized in many different ways, only some of which would provide a suitable basis for semantics. [...] I fail to see any great interest in syntax except as a preliminary to semantics.
>
> Montague[1970c], pp. 223

This divisions are basically also shared by Hintikka,

> Syntax is the study of language (and its logic) without reference to its representative function, while this function is studied in semantics. Pragmatics is defined as the study of the *use* of language.
>
> Hintikka[1976], pp. 18

We believe that grammar should be divided only into two parts, syntax and semantics with the following 'definition': Grammar = Syntax + Semantics.

> Syntax is the study of the forms.
> Semantics is the study of the uses.

Under this view, syntax corresponds roughly to the previous ones, but semantics is seen as either what Montague adopts in 'Universal Grammar', or a combination of his early semantic plus pragmatic notions. We believe that this view on semantics corresponds to more than a linguistic change, as it reflects current views on semantics, meaning and use. Wittgenstein's definitions of meaning as use, and for instance Barwise's [1981] comment that the basic aim of semantics is not just to characterize the notions of a true sentence and of entailment, but to characterize the uses of expressions of the language, support this view. A syntactically acceptable expression of natural language should be semantically characterized with the mathematical tools we know in order to achieve precision in its use. If we are solely interested in the use of declarative sentences, see section 1.3.4, we are interested for instance on how to precisely derive its boolean valuation.

We could also argue that this choice corresponds to the distinction in Quirk et al [1985], pp. 55, between form and function in grammar. We should also note that once we start using formal systems to characterize a grammar, syntax and semantics, the distinction between form and use is more of a regulative principle and of method. As we shall see in section 1.3.1 on *Formal Reduction of Semantics*, even semantics will have to be formalized, and indirectly will also need to be a study of the form. Transformational views on grammar in our view are untenable exactly because even taking an open minded view[7], the possibility that direct studies on forms and syntactic structures can characterize the use of natural language seem remote.

1.3 MATHEMATICS AND COMPUTATION FOR NATURAL LANGUAGE

The mathematical formalization of logic and language initiated by Boole[1854] and Frege[1879] culminated in Whitehead and Russell's 'Principia Mathematica' [1910] and in the philosophies of logical atomism and logical positivism originating from Wittgenstein's 'Tractatus' [1921]. As van Heijenoort[1967] observes, Mathematical Logic comes into a new period with Brouwer's intuitionistic insights into the foundations of mathematics and with the influences of Gödel's constructive results and methods upon Hilbert's program. Mathematical Logic and the development of the theory of recursive functions with methods originating from Gödel, provided the theoretical basis for the studies of computation (see Cutland[1980]). Church's[1936] definition of a formal system called the λ-Calculus, a form of notation to express and evaluate functions, Strachey's use of it (see Milne and Strachey[1976]) to study programming languages and Scott's[1976] constructive models used in its interpretation based on ordinary recursion theory provide one of the most satisfactory and intuitively valid theories of computability.

[7] See Montague's [1970c] opening note on Transformational views on grammar, for the more strict judgement.

We will briefly discuss the methodological importance of formality and formal systems as a means of achieving precision and relative certainty. We give a brief definition of our understanding of Denotational Semantics, and of the possibilities of the λ-Calculus. We discuss the complexity of a computational analysis of natural language by analysing implementations of the syntactic and semantic functions. We compare the formalistic paradigm of language definition with the problem of learning a language. We end this section by discussing current lines of research on grammatical formalization, i.e. on computational linguistics.

1.3.1 Formality and Formal Systems

The importance of formality has been recognized as a means of achieving precision and relative certainty and therefore having a vital role to play in the definition of computational uses of natural language. We use the expression 'relative certainty' because of Lakatos's [1976] analysis of the mathematical method, and because we regard formality an essential property of that method. Formality can therefore be used to strengthen the mathematical method against ambiguity.

We believe that a mathematical theory of computation when used in the study of natural language should provide an explanation of how a machine, which only has a basic ability to distinguish different *forms* can still be able to use a language in some intuitively *meaningful* way. The explanation, should not only be precise but also intuitive. The intuitive mathematical abstraction for such machine is for instance given by Turing's automata theory. As we have referred, connectionist theory (Rumelhart and McClelland[1986]) also has strong intuitive appeal, particularly as a computational model of human learning. Although connectionist theory can be stated using automata theory, in our following use of the words *formal system* we will be thinking of theories directly comparable with Turing's, such as discussed in Cutland[1980].

A formal system is defined, for example, by providing a finite number of basic expressions and of transformation rules. We say that all the basic expressions and all the expressions derived by application of the transformation rules are

expressions of that formal system. According to Martin-Löf's[1983], to be completely formal means precisely that there must be no semantic conditions involved in its definition. The step we take of providing the definitions should bring along all the certainty available. It should also codify what we call the meaning or the logic of the system that is being formalized. A formal system handles in the end only syntactic objects, that is what 'formal' means.

If a formal system attempts to reproduce the way we think we would call it a formal logic, its basic expressions are called axioms and its transformation rules are called logical inference rules. As we know, there are formal systems for classical logic, intuitionistic logic and for other logics, each having different properties.

Formal Reduction of Semantics

Our previous definition of syntax and semantics, in section 1.2.3, was to some extent influenced by Martin-Löf's [1983] views on formality, logic and on syntax and semantics. Objects, such as expressions, formulas and sentences, defined by rules on form, are syntactic objects. Objects, such as propositions and utterances, defined on syntactic objects, but not directly with rules on form, are semantic objects. The rules that determine the values for the semantic objects are rules that take into account the contexts of use. It is obvious that to achieve a correct computational use, we must accept that in principle formality can be restored, by convenient formalization of the contexts of use, with rules on the form of the latter. We will call this the principle of formal reduction of semantics [8].

Formal Grammar

We can think that in *PTQ* Montague defines a formal system which we will call *M-NL* using a formal grammar [9]. Montague's definition formally captures or reduces

[8] We believe that the discussion between Barwise and Fodor on 'languages of thought', following the introduction of situation semantics, is a consequence of different views on how far can we take the formal reduction for the human cognitive thinking process (see for instance Barwise[1986] and Fodor[1986]).

[9] An adaptation of Lakatos' dialogues could be made on Montague grammar in *PTQ*. Under this view one would say that it is Gallin[1975] who 'really' provides a formal system for Montague's English. Its non-tensed translation is formalized with axioms and inference rules.

for instance semantic notions of time and modality; therefore the formal logic is called temporal and modal. He also uses a typed λ-Calculus and so it formally reduces the semantic notions of classes, arguments and functions proposed by Frege[1892]. Although the formal semantic rules are only presented in English, and defined over syntactically presented objects such as expressions and contexts of use, they can and have been formally reduced, for instance by Gallin[1975]. Any computational implementation of *PTQ* makes such a reduction.

Denotational Semantics

We understand Denotational Semantics to be a methodology. In this respect it is more than a particular style of notation and of a mathematical theory of computation. The notation is based on the λ-Calculus and has been widely used in the characterization of programming languages since Strachey started using it in the early 1960's. The mathematical theory of computation known as Domain Theory was introduced by Scott[1976] as a foundation for the λ-Calculus formal system. It constructed models for that formal system using sets of integers.

> Having repeatedly and outspokenly condemned the λ-Calculus as a formal system without clear mathematical foundation, it was up to me to provide some alternative. [...] The present paper evolved particularly from the project of making the connections with ordinary recursion theory easier to comprehend, since a satisfactory theory of computability and programming language semantics had to face this problem.
>
> Scott [1976], pp. 575-576

Stoy[1977], Schmidt[1986] and Gordon[1979], the latter at a more elementary level, made a large contribution to the diffusion of Strachey and Scott's methods among the computer science community. The fact that its style of notation and its use of the mathematical theory is so well known in computer science is the reason, we believe, that it can be used also as a medium to express unambiguously definitions for the uses of natural language. Lack of a constructive mathematical foundation would compromise the scientific choice and lack of diffusion would jeopardize its logic engineering use which ultimately has to test such choice [10].

[10] This statement was included in order to satisfy Kuhn's[1962] engineering requirement, under which theories must be well known to be usable.

Another advantage of Denotational Semantics is that through the use of recursion theory it is possible to give an elegant and clear treatment for a number of paradoxes in natural language. As Visser[1988] has shown, the liar paradox and a whole group of similar ones can be modeled using fix-point semantics in a very intuitive way.

1.3.2 The Typed λ-Calculus and Scott's Theory of Computation

Denotational Semantics can in general be used to characterize extensions to the pure λ-Calculus, as was shown by Scott[1976]. However we will be interested only in the λ-Calculus constructively typed in the sense of Scott[1981]. It provides expressive power to 'program' any computable function, or in the terminology of recursion theory, any partial recursive function. Scott claims that if we are careful in the use of constants and types we can guarantee that any expression, a λ-term, defines a computable function.

> The major consequence of what we have done concerns the typed λ-Calculus. *Any* expression involving only *effectively given types* and, perhaps, some *computable constants* using only the λ, ! -notation [! x. τ = fix (λx. τ)] defines a computable function of its free variables. And such functions applied to computable arguments give computable values. And such functions have computable least fixed points. Etc., etc. In a definite sense then we have in the "metalanguage", as people say, a quite precise and fully *mathematical programming language* for defining computable operators. [...] And when we combine the usual type-definition facility together with *domain equations*, we have an especially powerful language.
>
> Scott[1981], pp. 121

Through the use of the fix operator Scott is able to define in his typed λ-Calculus the least number operator, the μ-scheme (pp. 84) and so the system can define any computable function. A standard result from recursion theory shows that when a function is defined solely in terms of decidable predicates or total computable functions the use of the μ-scheme can result in the definition of a non-total function (see Cutland[1980], pp. 45); therefore in general the execution of a typed λ-Calculus program in Scott's sense may not terminate for some argument. We need in general to have an independent proof whether or not a λ-term expresses a total function, which computation will eventually halt.

Simply Typed λ-Calculus

There is another use of the name 'typed λ-Calculus' usually called the simply typed λ-Calculus and originally defined by Church[1941] from Russell's[1908] ramified theory of types (see Fortune *et al*[1984] for a recent definition). This system does not use a fix operator and the representable functions form a proper subset of the elementary functions (pp. 163). It is claimed that normalization for this calculus is provable in Peano Arithmetic. Assuming that all primitive functions are total, and with computation seen as normalization or reduction, any simply typed λ-Calculus program will terminate. Fortune *et al*[1984] also presents the second-order typed λ-Calculus or polymorphic λ-Calculus. It is a generalization of the simply typed Calculus, obtained by adding a type abstraction mechanism to the simple types. The representable functions are a subset of the computable functions called the ε_0-recursive functions. Proof of normalization is discussed but it is not provable in Peano Arithmetic nor in augmented second-order Peano Arithmetic (pp. 179). However we will not be using this system by itself, but with a constructive semantics as introduced by Scott[1976].

Constructively Typed λ-Calculus

The typed λ-Calculus, with Scott's [1981] constructions, is the system we will be using to characterize Montague's English and Situation Semantics' English with computable denotations. We want for the metalanguage a computable notation, other than a high-order logic. The mathematical and computational properties of such logics are unattractive since for instance they constrain theorem provers to be inefficient. As will be shown, we think that Montague's English is not necessarily classifiable through a high-order logic. If, as we argue, meaning postulates are to be dropped and their effect replaced by syntactical and semantical rules, every translatable expression of English should be equivalent to an expression of a typed λ-Calculus language with only computable constants. If we want our syntactic and semantic analysis to provide terminating implementations, we just need to choose the appropriate basic constants, and the appropriate syntactic and semantic composition rules.

For instance, in Montague's *PTQ* meaning postulates are the direct source of high-order problems, since we are not making the distinction between intensional and extensional verbs at the syntactic or semantic stage of analysis, and since at that late stage we need to quantify over properties. In this case, we think that the distinction between intensional and extensional verbs should be explicit in the syntactic or semantic definitions, as we propose in chapter three, and so we would avoid the introduction of unnecessary complexities, such as meaning postulates. In general we believe that in an effective computational study of natural language, we can and should avoid the use of for instance high-order logics and functions which computation may not terminate, through the choice and use of correct definitions [11]. We will discuss the motivation for this in the next paragraph.

1.3.3 Complexity in a Formal Analysis

The computational analysis of natural language can be discussed in terms of the complexity of its grammar, the syntactic and semantic systems being defined. There is a difference between a formal analysis of language which has real computational applications in view and one which has not. Montague's analysis although formal has been criticized as being computationally unattractive, because its Intensional Language is a higher order logic. Computational implementations of higher order theorem provers are inefficient and ill-understood (Turner[1988a])

As we proposed, the definition of a formal grammar for natural language can be divided in two closely related parts. We have to provide a definition for a formal system which characterizes the form of the language, its formal syntax, and for a formal system which characterizes the use, the formal semantics. Each implementation may have different properties concerning computational complexity and efficiency.

[11] We accept that in a non effectively computational study of natural language we may accept more complex analysis, with various benefits. In this case we can still use the same typed λ-Calculus as it has all the expressive power we need to program any computable function, including non effectively computable functions with non-terminating programs.

The Formal Syntax

In *PTQ* Montague defined a syntax for a dialect of natural language with 17 rules, which used 15 syntactic functions [12], two of which indexed on the integers (see Thomason [1974], pps. 7, 23). An alternative and more intuitive definition for syntactic formal systems would be one in which we have only one mode of combination, concatenation, but more rules. We could then choose the formal representation to reflect the type of the syntactic part of the grammar in Chomsky's hierarchy (see for instance Harrison[1978]). By restricting the representations, we could restrict the syntax to be for example context-free, and by using parser generators we could generate efficient parsers directly from the definition of the syntax [13].

The Formal Semantics

In *PTQ* Montague defined the semantics as an interpretation function. For instance, when given an expression of the category sentence and an index, or context of use, the interpretation would evaluate to a boolean value. To be computable this function has to be definable as a program in some programming language. Montague has chosen to translate English into an interpreted intermediate language, called Intensional Logic, in which computations could be regarded both as in theorem proving and as in function evaluation. There are different choices of how to provide the formal semantics for Montague's English, through variations on the intermediate language and on the interpretation of the latter. For instance Turner[1983a] can essentially be seen as using Scott's Lambda Calculus to give an interpretation to the intermediate language and therefore express a formal semantics for Montague's English. In the same way ours chapters 2 and 3 provide interpretations for Montague's English with alterations over the intermediate language and its

[12] Also called modes of combination or structural operations.

[13] See Pitt[1989] for an *LR* deterministic context-free presentation for *PTQ*'s syntactic formal system. The argument whether natural language is or is not context free depends on our understanding of what are 'intuitively acceptable expressions' and depends on our formal reduction being directed towards the syntactic or the semantic formal system. It is a moot point that we can have a formal syntax that over-generates, so it accepts every string. In Cunha and Pitt[1989] we discuss the possibility and advantages of restricting the syntax to a context-free form. There we propose such a syntax for Shieber's [1985] example, which provides evidence against his claim against the context-freeness of natural language.

interpretation, in order to make explicit the computational characteristics of his treatment and in order to make it more tractable.

Formal and Connectionist Systems

It is not in the scope of this thesis to compare formal systems with Connectionist Systems. But we think it is important to notice the different approaches followed to capture information. Suppose someone *A* knows a language, as a collection of strings, and *A* is going to tell us what he knows about that language.

As *A* is a formalist he has decided to provide me with a Post production system to define the language. As an illustration we will use the *MIU* formal system presented in Hofstadter[1979], pp. 33. This system utilizes only three letters, *M*, *I* and *U*. It has one basic expression *MI* and four transformation rules:

1. If you possess a string whose last letter is *I*, you can add a *U* at the end.
2. If you possess *Mx*, you can add *Mxx* to your collection.
3. If *III* occurs in one of the strings in your collection you can make a new string with *U* in place of *III* .
4. If *UU* occurs inside one of the strings you can drop it.

The collection of strings is the *MIU* language. It is possible to prove that the string *MU* is not in the language, but such proofs are not possible in general for an arbitrary string. Given any string, there should be a procedure to decide if it is or if it is not in the language. If it is known, for instance by *A*, that some string is in the language, we can find a derivation for it. Of course this does not tell me how in particular *A* knows that. Probably *A* has derived it himself, or knows more about the system than we were told above. But if *A* has given me a string which he knows that it is not in the system, we might never finish in the process to find its derivation.

If we were not happy with this state of affairs, we could try to use a connectionist system. We would only accept that *A* provides examples of strings in the language,

and strings not in the language in the training or learning phase. After having gone through a sufficient number of examples, the system should have induced some rule to be able to decide always in a finite amount of time whether any given string was or was not in the language. We could also allow the training of explicit cases of 'not known'. For simplicity we also regard the system as deterministic. This means that the function it computes, after the training or learning stage is finished, does not change with time, and can therefore be represented as a decidable formal system.

There are two important points we believe should be noticed. The first is that in the connectionist case we are never sure that the results are correct, as they depend on whether the learning and induction process was sufficiently complete. The second one, which is an open question, is whether the human processing capacity is better simulated by variations of the first formal system or of the second. One of the implications of accepting the second model, is that any question produces an answer in a finite time, even if that answer has to be something like the 'not known' case.

The Computational Meaning of Quantification

The logical and computational study of quantifiers has been closely related to the logical study of the infinite, and of the problems of scientific induction and generalization [14]. Russell[1946], pp. 647, thinks of induction as an independent logical principle, without which generalization in natural science is impossible. To generalize corresponds to use the principle of induction on a finite number of observed examples of some phenomena that is under study. Linguistic expressions which describe the results of scientific induction contain at least one determiner, usually a universal quantifier.

> The Greeks in general attached more importance to deduction as a source of knowledge than modern philosophers do. In this respect, Aristotle was less at fault than Plato; he repeatedly admitted the importance of induction, and he devoted considerable attention to the question: how do we know the first premisses from which deduction must start? Nevertheless, he, like other Greeks, gave undue prominence to deduction in his theory of knowledge. We shall agree that Mr Smith (say) is mortal, and we may, loosely, say that

[14] This is not to be confused with mathematical induction, for instance on integers. Scientific induction is present for instance when a universal law is proposed as an attempt to explain some natural phenomena.

> we know this because we know that all men are mortal. But what we really know is not 'all men are mortal'; we know rather something like 'all men born more than one hundred and fifty years ago are mortal, and so are almost all men born more than one hundred years ago'. This is our reason for thinking that Mr Smith will die. But this argument is an induction not a deduction. It has less cogency than a deduction, and yields only a probability, not certainty; but on the other hand it gives *new* knowledge, which deduction does not.
>
> Russell[1946], pp. 209

In the 'Tractatus', Wittgenstein tried to account for quantifiers by taking universally quantified statements as long conjunctions and existentially quantified statements as long disjunctions. In order to do this however Wittgenstein would need to introduce an axiom of finitude, what was in contradiction with Russell's necessity of an axiom of infinity to get number theory off the ground. As Kenny[1973] refers to, pps. 92, 116, Wittgenstein was never satisfied with this account, except when the conjunctions or disjunctions were clearly finite. Statements about all numbers could not be treated as conjunctions of statements about each of the numbers.

Karl Popper has extensively criticized Russell's views on induction (see for instance in Popper[1953], particularly sections IV to X). The criticism of induction as a logical principle, and as a consequence that scientific universal statements cannot in general be logically deduced, and a parallel criticism of the unscientific character of existential statements (see for instance in Popper[1935], section 15), have proved to be an interesting proposal. In chapter two we will provide a computable definition for universal and existential quantifier operators, which are monotonic and continuous. Our definition resulted from attempts to find Plotkin's [1981], pp. 15, maximal quantifier in a computable space of functions. Our quantifier intuitively shares the properties we associate with the use of generalization sentences. We believe that our definition is consistent with Popper's views on such sentences.

Generalized and Binary Quantifiers

Barwise and Cooper[1981] have drawn attention to the fact that there is a much larger number of quantifiers other than the universal and existential quantifiers

introduced by Frege. They also argue that the syntactic structure of quantified sentences in predicate calculus is very much different from the syntactic structure of quantified sentences in natural language. Barwise and Cooper's work on generalized quantifiers claims that many determiners of natural language cannot be modeled naturally with the quantifiers of first order logic. Their observations seem to bring again into logical discussions of natural language Aristotle's views on logic, particularly its 'Syllogistics' (see for instance Lear[1980]). Westerstähl[1988] covers much of the current state of research on this area. We think it is particularly promising for computational analysis of language Westerstähl's theory of binary quantifiers.

1.3.4 Computational Linguistics

Symbolic processing of natural language applies in its most general form to a syntactic object called text. Text can be formalized as an ordered collection of characters from a finite set. Characters form words, sentences, documents. Documents are of different types, as are words and sentences.

> [...] Two terms are employed for the interconnection of grammar and the uses of grammar: TEXT LINGUISTICS and DISCOURSE ANALYSIS. All types of organization (but notably lexicology and grammar) enter into the structure of TEXTS, which constitute spoken and written discourse.
>
> Quirk et al[1985], pp. 12

In the formal processing of natural language we often assume that any kind of communication can be reduced to some form of text. We will also assume that contexts of use can be formalized as text, in order to be amenable to formal processing.

The linguistic analysis of natural language produces knowledge. It creates grammars which attempt to characterize the correct forms and uses of languages. In part, this process takes place through the introduction of technical languages in order to classify the linguistic objects and derive rules. The definition of a stable dictionary, although partial and incomplete, is essential in establishing the rules of grammar. Its definitions provide the basic vocabulary, basic primitive classifications

and a basic set of informal equality relations on use. Although most of the definitions are ambiguous, i.e., there are multiple definitions for the same words, the disambiguation is usually decidable given the context of use. In a computational implementation this ambiguity should always be decidable, if we accept the principle of formal reduction referred to previously.

Types and Types of Sentences

Linguistics accepts that natural language is essentially typed. The need for regularity demands that to happen. It is basically the same reason that Scott[1976] refers to for the definition of data types in programming languages,

> What then is a *data type* ? Answer: a type of data. That is to say, a collection of data that have been grouped together for reasons of similarity of structure or perhaps mere convenience. Thus the collection may very well be a mixed bag, but more often than not canons of taste or demands of simplicity dictate an adherence to regularity. [...].
>
> Scott[1976], pp. 539

This is not a claim that natural language is equivalent to a typed formal system, but could be used to indicate so. As the choices of categories, or types, in traditional grammar result from comprehensive studies, we would think that they represent effective classifications, at least for our use. The adherence to regularities is consistent with mechanisms to deal with ambiguity. For instance the division of words into classes of verbs and nouns is dictated by simplicity, but there are rules which also allow a word form classified as a verb to be converted and classified as a noun. Quirk *et al* [1985], for instance group these rules under the name of conversion rules (pp. 1520). The mathematical method applied to linguistics attempts to make explicit these regularities and the choices of categories.

One of the working hypothesis in this thesis is that Montague and Barwise are just formalizing different uses of language using different types of sentences: declarative sentences and indicative sentences (see further discussion and examples on chapter three, section 3.5.2). A sentence in a declarative use, and in a given context, has the same type of meaning that Barwise attributes to statements of indicative sentences. In general our hypothesis extends to other types of sentences such as interrogative.

In general we believe this is the kind of typing in Wittgenstein's famous list of different kinds of language-games,

> But how many kinds of sentence are there? Say assertion, question, and command? — There are *countless* kinds: countless different kinds of use of what we call "symbols", "words", "sentences". And this multiplicity is not something fixed, given once and for all; but new types of language, new language-games, as we may say, came into existence, and others become obsolete and get forgotten. (We can get a *rough picture* of this from the changes in mathematics.)
>
> Here the term "language-*game*" is meant to bring into prominence the fact that the *speaking* of language is part of an activity, or of a form of life.
>
> Review the multiplicity of language-games in the following examples, and in others:
>
> Giving orders, and obeying them —
>
> Describing the appearance of an object, or giving its measurements —
>
> Constructing an object from a description (a drawing) —
>
> [...]
>
> Solving a problem in practical arithmetic —
>
> Translating from one language into another —
>
> Asking, thanking, cursing, greeting, praying.
>
> — It is interesting to compare the multiplicity of the tools in the language and of the ways they are used, the multiplicity of kinds of word and sentence, with what logicians have said about the structure of language. (Including the author of the *Tractatus Logico-Philosophicus*.)
>
> Wittgenstein[1953], §23, pp. 11e

This view of language would allow us to conclude that the different formal definitions of meaning for sentences between Frege and Montague on one side and Barwise and Perry on the other, are just different perceptions of their uses. Frege and Montague's meaning object of a boolean intension would be seen as formalizing the meaning of a declarative sentence used in a language-game which requires a judgement using 'yes' or 'no'. Barwise and Perry's meaning of referred situations would be seen as formalizing the meaning of a sentence used in a language-game which consists of describing possible referred situations. However we believe that this is not the kind of argument that would be accepted by Montague, Barwise or Perry, whose inspiration for natural language semantics comes in many respects from logical positivism. From a methodological point of view, we would argue that the choice for meaning and its formalization are two independent problems. We are not claiming however that they are not related, in the

same way as an indicative sentence has an associated boolean value, when used declaratively.

Traditional and Formal Linguistics

We will call informal or traditional, linguistic systems which explicitly use in their constructions mathematical methods. Quirk et al[1985], Halliday[1961] are examples of traditional grammars of English that fall under under this classification. We will call them formal if mathematical methods play an explicit role in their construction. If mathematical is identified with computational we would say that computational linguistic theories on grammar differ from the traditional ones because they are interested in grammars that can be used both by people and by machines. However, we may not always accept that mathematical theories of grammar are usable by both. For completeness, we will very briefly refer to some research paradigms in computational linguistics. Pitt[1989] provides an excellent review of the developments in the field of computational linguistics, and of the state of the art in parsing and generation.

Transformational grammar, originating from Chomsky[1957] and Chomsky[1965] provides a formal theory of grammar. Very generally , it uses a syntactical paradigm based on a formal theory of languages and automata theory. As far as we are concerned, it fails to recognize the importance of semantics and mathematical logic by concentrating in the syntactic definition of the set of declarative sentences of the language. Winograd[1983], particularly chapter four, provides an interesting overview of the computational paradigm in transformational grammar.

Systemic grammar originating from Halliday[1961] is a special presentation of traditional grammar. It was formalized and used in Winograd[1972] to implement *SHRLDU*, perhaps the first successful natural language processing system. Although not originally designed for computing applications, systemic grammar can be characterized with mathematical methods. Functional-unification grammars, Kay[1985], and logical treatments of feature graphs, Rounds and Kasper[1986],

can be seen as formal treatments for systemic grammars. Halliday and Martin[1981] present more recent developments of the original paradigm.

Functional-unification grammars can also been seen as a consequence of investigations in logic, automated theorem proving and parsing-generation. For instance the work on logical grammars and parsing initiated by Colmerauer[1976] and Pereira and Warren[1978] has its foundations in the study of logic and theorem proving.

1.4 CONCLUSION

We believe that Montague Semantics and Situation Semantics are the two most successful formal paradigms for natural language analysis. They have strong philosophical and mathematical foundations, and accept that most of the traditional work on linguistics can be usefully formalized. However we believe that those systems should be both simplified and given clear computable presentations. We propose to use the methods and notations of Denotational Semantics.

We believe that recent developments in philosophy of mathematics and science, which can be called constructivist, demand a greater preoccupation with the implementability of formal theories, or proofs of such possibility. They also require discussions on how they relate to human cognitive processes, such as for instance whether it is possible for such formal systems to be learnt. Because Denotational Semantics provides a computable theory of functions and because functions can be discussed on terms of learnability we believe it offers a suitable framework for the analysis of natural language.

> To most logicians (like the first author) trained in model-theoretic semantics, natural language was an anathema, impossibly vague and incoherent. To us, the revolutionary idea in Montague's paper PTQ (and earlier papers) is the claim that natural language is not impossibly incoherent, as his teacher Tarski had led us to believe, but that large portions of its semantics can be treated by combining known tools from logic, tools like functions of finite type, the λ-calculus, generalized quantifiers, tense and modal logic, and all the rest.
>
> Montague had a certain job that he wanted to do and used whatever tools he had at hand to do it. If the product he built looks a bit like a Rube Goldberg machine, well, at least it works pretty well.
>
> Barwise and Cooper[1981], pp. 204

CHAPTER 2

COMPUTABLE NOTATIONS AND DENOTATIONS FOR MONTAGUE'S *PTQ*

2.1 INTRODUCTION

In 'The Proper Treatment of Quantification in Ordinary English', usually shortened as *PTQ*, Montague[1970d] defines *a certain fragment of a certain dialect of English.* He provides for that fragment of English a formal syntax and semantics, using mathematical tools developed by Tarski, and supporting Popper's [1960] writings which claimed that Tarski's theories were applicable not only to formalized languages but as well to natural languages.

Montague's fragment of Natural Language, which we will call *M-NL*, is a syntactically typed language, whose categories are the ones we could find in a

traditional grammar of English such as Quirk *et al*[1985]. In order to define the interpretation of *M-NL* Montague introduces an intensional language *IL*, gives its semantics in a space $\mathcal{S}$ of of functions over truth-values, entities and indices, and the interpretation of *M-NL* is induced by translation into the intermediate language:

$$M\text{-}NL \xrightarrow[\text{translation}]{t} IL \xrightarrow[\text{interpretation}]{u} \mathcal{S}$$

In the presentation of *M-NL* of this chapter we are mainly interested in highlighting its computational properties. As was proposed in chapter one we will be using to achieve that objective the treatment of Denotational Semantics. In this chapter, with a few limitations, and in the next chapter we will show how the intuitions in *PTQ* can be captured using only computable functions defined over effectively given domains of partial objects, with the advantages referred to in the previous chapter:

$$M\text{-}NL_{\perp} \xrightarrow[\text{translation}]{t} IL_{\perp} \xrightarrow[\text{interpretation}]{u} \mathcal{S}_{\perp}$$

Our presentation has been greatly influenced by the presentations of Gallin[1975], Dowty *et al*[1981], Turner[1983a] and Jowsey[1986]. The use we make of the functional notation is new in Montague semantics, particularly in the form of the definitions for the semantic functions which make explicit the name of the functions and of the arguments. Making those names explicit at notation level is, we believe, a great advantage in the discussions of meaning, intentionality and computability [1]. In the use of Dana Scott's Domain Theory for semantics our treatment follows directly from Turner[1983a, 1983b]. We believe it is more important to use Scott's theory as a theory of computation in the definition of grammars for natural language, and not mainly to use it to show the existence of non-trivial solutions for denotations of

[1] The word *intension* was introduced by Carnap[1946] in logical terminology and has been widely used in formal treatments of *intentionality*. In Philosophy intentionality is involved when dealing with the objects of knowledge, belief and necessity, as opposed to immediate sense objects involved in using the physical senses; see for instance Malmgren[1971]. As Hodges[1977] says *Alas for the infelicities of logical terminology*. In general remarks we will be using the word *intentionality*, but we will use in the logical tradition *intension* and *intensional* verbs (as opposed to *extension* and *extensional* verbs).

other linguistic phenomena in a Montagovian ontology, as Turner does with the objective of providing a treatment for nominalization.

We start in the following section 2.2 with an introduction to the notation and primitives which will be used in our computational study of natural language. This notation has to be understood in the Scott-Strachey style of Denotational Semantics [2]. Section 2.3 introduces the definition of English in *PTQ* using this notation and theory. We will comment on Montague's definition of grammar, the meaning of natural language expressions, the treatment of intentionality and the definition of contexts of use. This presentation will be a first step towards a simplified treatment with clear computational presentation. The definition for the semantics of *IL* follows from Turner[1983a] by using Scott[1981] improved notation. In section 2.4 we discuss the definition and use of the intensional operators and provide evidence which shows that they should be avoided. We also discuss whether we need intensional operators in the treatment of intentionality in natural language, and of intensional verbs in particular.

2.2 Notation, Primitives and Computable Constants

Most of the notation which we will use is standard and should cause no problems for the reader. Notation for semantics will be that standardly used in Denotational Semantics. As we regard functions and arguments as having equal status we will tend to write ab instead of $a(b)$. We will use different types of parenthesis to facilitate reading. We will tend to use square brackets on arguments when using functions with objects in syntactic domains. The notation $g_{[a \leftarrow v]}$ denotes a function like g, except that variable v in the domain of g is assigned the value a. Application associates to the left, so we write abc instead of $(ab)c$. Arrow associates to the right, and so we write $a \rightarrow b \rightarrow c$ instead of $a \rightarrow (b \rightarrow c)$. Sum, product and arrow have the usual precedences, so we write $a \times b \rightarrow c$ instead of $(a \times b) \rightarrow c$ and

[2] Although the name *Denotational Semantics* can be used to refer to any mathematical semantics, say as oposed to axiomatic (Hoare) semantics or operational semantics, in Computer Science it is usually used to refer to a particular type of mathematical semantics developed by Strachey and Scott to formalize programming languages. See for instance Stoy[1973], Milne and Strachey[1976] or Scott[1981].

$a + b \rightarrow c$ instead of $(a + b) \rightarrow c$. In our definitions we shall use $=$ for definitional equality. The symbol $\cong$ stands for isomorphic equality between domains.

2.2.1 Domains and Computability

The primitive domains we will be using are complete partially ordered sets having a a least element, or *CPO's*. Usually they are just Montague sets to which a least element $\perp$ has been added. We just impose that for all the elements of the set are incomparable by the ordering relation $\sqsubseteq$ and that each element x is better defined than $\perp$, and we write $\perp \sqsubseteq x$. For a primitive set X we denote the new domain by $X_{\perp}$, or if there is no confusion just by X. The operation described is usually named *lifting*. We will do exactly as in Turner[1983a] for all of Montague's primitive sets. This type of domains is important because it has the properties which must be respected in order to model effective computability. On this subject see for instance chapter one of Plotkin[1981] or Scott[1981].

Together with our domains we introduce a few operations that will be used to construct other domains and which preserve the properties of effective presentation. Using such domains, the effective computability of functions can be proved based on the acceptance of a few theses, equivalent to Church's thesis. For instance, a function to be effectively computable must be defined on a *CPO*, and must be proved to be monotonic and continuous on that domain. For example the functions ***cond*** and ***and*** are part of the structure of the boolean domain, that we will define in the next section, and are proved to be computable in Scott[1981], pp. 57. However, the mathematical properties of monotonicity and continuity are particularly helpful to prove computability of functions with arguments of non finite size, such as functions on functions over infinite domains. This is the case for instance of the semantic interpretation functions in this thesis.

2.2.2 The Boolean Domain

There are different ways to model declarative sentences in Logic. In order to do so Montague used the set of truth values. As we want to model the computability of

such semantics we will use instead a domain of booleans as defined by Scott[1981], or Turner[1983a]. Intuitively the total elements are Montague's true and false, and the undefined element is a truth-value gap which corresponds to a state of information for which we do not want to judge an declarative sentence either as being true or false. This presentation is the one which corresponds to accepting a complete symmetry between truth and falsity, as for instance was formally introduced by Boole[1854]. As we will see in the semantics of quantified declarative sentences, there are other presentations for the same elements of the boolean set, which correspond to rejecting their symmetry.

Definition 2.2.1 The domain $\mathscr{B}$ of *booleans* has two total elements t and f, and a partial element $\perp_{\mathscr{B}}$ or just $\perp$. Its structure satisfies the complete partial ordering $\perp \sqsubseteq t$ and $\perp \sqsubseteq f$.

$$\mathscr{B} = \{ t, f, \perp \} .$$



2.2.3 Constructions on Domains

In the definitions of domains we will be using the following constructions where if A and B are effectively given domains the defined domain is also effectively given. The page numbers refer to the definitions in Scott[1981].

$A + B$	*disjoint sum*	*pp. 47*
$\Sigma_n A_n$	*finite sum* $(n > 0)$	*pp. 47*
$A \times B$	*product*	*pp. 33*
$\Pi_n A_n$	*finite product* $(n > 0)$	*pp. 45*
$A \to B$	*function space*	*pp. 38*

The condition we require on primitive domains to be effectively given has been taken from Scott[1981]. Intuitively we will only accept as primitive domains the ones which have at most a countable infinity of finite elements, and for this finite elements we must know how they are 'related to' each other (pp. 113).

When it is obviously provable that a presentation of domains can be indexed by integers and there is a finite number of such domains, we will use subscripts other than *n*. This is the case for instance in definition 2.3.1.4.

2.2.4 Computable Elements

In our definitions we will be using some predefined constants as computable elements. If *A*, *B* and *C* are any effectively given domains, the functions ***curry*** and ***cond*** with the following types are computable. The page numbers refer again to the full definitions in Scott[1981].

curry	$(A \times B \to C) \to (A \to B \to C)$	*pp. 42*
cond	$\mathcal{B} \times A \times A \to A$	*pp. 49*

Intuitively for instance, $cond(t, x, y)=x$, $cond(f, x, y)=y$ and $cond(\perp_{\mathcal{B}}, x, y)=\perp_A$.

Logical Connectives

In our definitions we will also be using the usual truth-functional connectives. Given that there are many choices for the computational implementation of the connectives, we would need in general to use disambiguation techniques. We could for instance think of them defined on the boolean domain as the standard strict functions. However, following Turner[1983b] we will use Kleene's strong three-valued connectives. The connectives ***and***, ***not*** and ***implies*** have the following definitions, and ***or***, ***xor***, and ***equival*** can be derived in the standard way for instance from the first two.

and	t	f	$\perp$
t	t	f	$\perp$
f	f	f	f
$\perp$	$\perp$	f	$\perp$

not	
t	f
f	t
$\perp$	$\perp$

implies	t	f	$\perp$
t	t	f	$\perp$
f	t	t	t
$\perp$	t	$\perp$	$\perp$

Equality

As we will see in the definition 2.3.4.1 of the semantic function on *IL* we need to use a function ***equal*** defined in general on every type of domain of denotations of

expressions. However, given any translation of English into *IL* we could show that we will only need to have this relation defined on elements of the domain of entities. If this is a flat domain we do not have problems to prove its computability. Such function is necessary to formalize Turner's [1983a] relation *is the same element as* for which the proof of computability was not given. In general we would need to check exactly how such function is used in Montague's treatment of English, as it uses equality between expressions of categories which require as denotation a function on intension of entities. For instance, the translation of *'the α'* uses equality on variables of *IL* of type $s \rightarrow e$. However, as in our treatment of the following chapter will not need more than equality defined in a flat domain, we will not discuss this matter further. In fact the same happens with Turner[1983a] as he uses Bennett's [3] simplification to *PTQ*.

Quantification

We will also need to use operators and functions corresponding to the quantifiers which we will name ***exists*** and ***forall***. There have been a number of problems in the attempt to define quantifiers and in answering questions on their computability. Plotkin[1981], chapter one, pp. 15, indicates a property that existential quantifiers should have and as a problem asks the reader to find the maximal computable quantifier. Unfortunately no answer is provided. Given that the work of Turner[1983a, 1983b] has concentrated on the problem of computability for the universal quantifier, we will discuss it here and we will assume that there is a similar treatment for the existential quantifier.

Turner's [1983b] initial definition for the universal quantifier, which we will name here as ***forall***, is basically the following one, where $\mathcal{N}_\perp$ is the flat domain (CPO) of integers and $[\mathcal{N}_\perp \rightarrow \mathcal{B}]$ is the domain of partially computable boolean functions on integers:

$$\textit{forall} : [\mathcal{N} \rightarrow \mathcal{B}] \rightarrow \mathcal{B}$$ [4]

[3] For Bennett's simplification see for instance Dowty *et al*[1981], pps. 188 & ffs.

[4] For simplicity, and if there is no ambiguity, we will avoid the subscript.

$$\textit{forall } f \quad = \quad \begin{cases} t & \text{if } f(n) = t \quad \text{for all elements } n \text{ of } \mathcal{N}. \\ f & \text{if } f(n) = f \quad \text{for some element } n \text{ of } \mathcal{N}. \\ \perp & \text{otherwise.} \end{cases}$$

If $<f_n>$ is an *ω-sequence* of the form $f_0 \sqsubseteq f_1 \sqsubseteq \ldots \sqsubseteq f_n \sqsubseteq \ldots$ in $[\mathcal{N} \rightarrow \mathcal{B}]$, it admits a least upper bound. We write for the least upper bound of such sequence $\bigcup_n f_n$. In order to prove that the quantifier operator, as defined, satisfies monotonicity and continuity Turner has to show that,

$$\textit{forall } (\bigcup_n f_n) = \bigcup_n \textit{forall } (f_n) \qquad \text{(T1)}$$

Given the structure of the boolean domain, i.e. that it is finite with two total elements, proving (T1) is equivalent to proving both (T2) and (T3),

$$\textit{forall } (\bigcup_n f_n) = t \quad \text{iff} \quad \bigcup_n \textit{forall } (f_n) = t \qquad \text{(T2)}$$

$$\textit{forall } (\bigcup_n f_n) = f \quad \text{iff} \quad \bigcup_n \textit{forall } (f_n) = f \qquad \text{(T3)}$$

Turner proves (T2) and (T3). The most critical step is in the proof of (T2) on the left-to-right implication: first observe that $\textit{forall } (\bigcup_n f_n) = t$ iff for each d in $\mathcal{N}$, $(\bigcup_n f_n)(d) = t$. By the structure of $\mathcal{B}$ there must be some n such that for each $m \geq n$ it is the case that $f_m(d) = t$. But then $f_m(d) = t$ for each d in $\mathcal{N}$. In particular for $d = \perp$, $f_m(\perp) = t$. But because of the continuity of f_m this guarantees that $\textit{forall } (f_m) = t$ for each $m \geq n$. Hence $\bigcup_n \textit{forall } (f_n) = t$.

As Turner observes, because the predicates or functions f on $\mathcal{N}$ are continuous and therefore monotonic, $\textit{forall}(f)$ is true if and only if f is a function which holds on the undefined element of $\mathcal{N}$. *But this rather trivializes the idea of quantification.* (Turner[1984b], pp. 31). This happens for instance in the case where the function has the form $f = \lambda n.\, t$ of a constant predicate.

Intuitively we should expect that $\textit{forall}(f_n) = f$ for any sequence which for some d, $f_n(d) = f$. Following the diagram below, we should only expect that $\textit{forall}(f_n) = t$

for an infinite total sequence, in the case where for each $d, f_n(d) = \checkmark$. Intuitively again, we would not expect for any partial sequence f_n that ***forall***(f_n) would be true.

n	$\perp$	0	1	2	3	...	***forall***(f_n)
	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
			...				
f_i	$\perp$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\perp$	$\perp$
			...				
f_1	$\perp$	$\checkmark$	$\checkmark$	$\perp$	$\checkmark$	$\perp$	$\perp$
f_0	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

But, as we will see, quantification presented in this intuitive way suffers from a major problem. We know as well that showing continuity can also be done by proving monotonicity,

$$\textit{forall}\ (\bigcup_n f_n) \supseteq \bigcup_n \textit{forall}\ (f_n)$$

and then to prove continuity, we just need to show,

$$\textit{forall}\ (\bigcup_n f_n) \sqsubseteq \bigcup_n \textit{forall}\ (f_n)$$

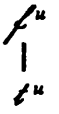
It may seem that the example in the diagram above violates continuity. However we should observe that the *ω-sequence* shown does not correspond to a continuous element in $[\mathcal{N} \rightarrow \mathcal{B}]$. If it did, and using Turner's argument, for some finite n, we should have that $f_n(\perp) = \checkmark$, when it is the case that for each $d, f_n(d) = \checkmark$. In other words, a sequence which only has all elements $\checkmark$ in the infinite limit cannot be computably presented.

Turner's attempt to define a computable quantifier based on restricting the definition on the undefined element becomes rather difficult as it involves the use and restrictions on E_∞, the limit of the constructions which form a solution of the domain equation $E = \mathcal{B} + [E \rightarrow E]$. The value of a quantifier ***forall***(p) where p is now of type $E_\infty \rightarrow \mathcal{B}$, is defined according to the value of $p(d)$ for d's in $E_\infty - \bigcup_n E_n$ and therefore depends on an integer parameter n for each predicate. Although this latter dependency is said by Turner to be intuitively acceptable and expected, it seems that the argument has been found wanting.

There are good intuitive reasons for supporting the computability of quantifiers. Given any finite approximation f_n, we usually make the assertion that ***forall***(f_n) is true based on an effective computation on that partial information. We can in fact define a universal operator which satisfies this notion of assertibility. However, we will need to change the structure of the boolean set and the definition of the universal operator as follows.

Definition 2.2.3.1 The domain $\mathcal{B}^u$ of *refutable booleans* $\acute{t}$ has one total element $\acute{f}$ and a partial element $\acute{t}$. Its structure satisfies the complete partial ordering $\acute{t} \sqsubseteq \acute{f}$. The upperscript u stands for *universal*, as this domain is introduced for the treatment of universal statements. We will avoid superscripts when context is not ambiguous[5].

$$\mathcal{B}^u = \{\acute{t}^u, \acute{f}^u\}.$$



Definition 2.2.3.2 The following operator is called a refutable universal quantifier:

$$\textbf{\textit{forall}} : [\mathcal{N} \rightarrow \mathcal{B}] \rightarrow \mathcal{B}^u$$

$$\textbf{\textit{forall}}\ f = \begin{cases} \acute{f}^u & \text{if } f(n) = \acute{f} \quad \text{for some element } n \text{ of } \mathcal{N}. \\ \acute{t}^u & \text{otherwise.} \end{cases}$$

Using this definition we can prove monotonicity and continuity. To prove monotonicity we need to show that

$$\text{if } f_i \sqsubseteq f_j \quad \text{then } \textbf{\textit{forall}}\ (f_i) \sqsubseteq \textbf{\textit{forall}}\ (f_j).$$

This much is obvious as an increase in the definition of f_n can only be achieved either by having for some d the value of $f_{n+1}(d)$ changing from undefined into true or false. In the later, ***forall*** (f_{n+1}) becames $\acute{f}^u$, in the former it does not change value.

[5] We could have used another domain with an undefined element below $\acute{t}$. That option might be necessary to treat universally quantified sentences for which no truth value would be an appropriate value, such as syntactically unacceptable ones. However, for a semantic treatment involving the notion of refutation it seems that the domain above is sufficient.

To prove continuity and given the structure of $\mathcal{B}^u$, we need to show that,

$$forall\ (\bigcup_n f_n) = \not\in^u \quad \text{iff} \quad \bigcup_n forall\ (f_n) = \not\in^u$$

But, because $\mathcal{B}^u$ is finite the equality on the right of the equivalence holds iff for some n, $forall(f_n)=\not\in^u$. Given the definition of *forall*, the previous equality holds iff for some d in $\mathcal{N}$ and for some n, $f_n(d)=\not\in$. And this iff $(\bigcup_n f_n)\ (d)=\not\in$. But again from the definition of *forall* this happens when $forall\ (\bigcup_n f_n) = \not\in^u$.

We believe that there are good reasons to accept this definition of universal quantification as well as many questions to answer. Let us observe the following examples E1 and E2, covered by this definition.

E1	⊥	0	1	2	3	...	*forall*(f_n)
	⊥	∉	∉	∉	∉	∉	$\not\in^u$
			...				
f_i	⊥	∉	∉	∉	∉	⊥	$\not\in^u$
			...				
f_1	⊥	∉	∉	⊥	∉	⊥	$\not\in^u$
f_0	⊥	⊥	⊥	⊥	⊥	⊥	$\not\in^u$

E2	⊥	0	1	2	3	...	*forall*(f_n)
	⊥	∉	∉	∉	∈	∉	$\in^u$
			...				
f_i	⊥	∉	∉	∉	∈	⊥	$\in^u$
			...				
f_1	⊥	∉	∉	⊥	⊥	⊥	$\not\in^u$
f_0	⊥	⊥	⊥	⊥	⊥	⊥	$\not\in^u$

We believe that this characterization for the universal quantifier corresponds to having a refutability theory of meaning in the lines defended by Popper. (see for instance Popper[1960]). Under this view any universal statement which has not been falsified in any finite approximation can be computed as being true. In this respect, we need truth to be less defined than falsity, as the former can always be assumed if no information is available [6]. Some of the questions which will need to be answered are related with the semantics of embedded quantifiers, for instance what happens when giving the semantics of quantified relative clauses of

[6] We propose to discuss this matter in a forthcoming paper. We believe this is the proper treatment we need for quantified scientific sentences. For instance in Newton's theory of gravitation, universal statements were made and these were generally assumed to be true because, if we accept Popper's claim, under the known collection of facts they cannot be falsified. The treatment of existential quantifiers should be done in a similar way to the one proposed here, with the role of the elements true and false of the boolean domain inverted. The falsity of an existential statement for which there is no exemplification known can, and according to Popper should, always be assumed at least in a scientific environment.

propositional attitudes. We will not take further this discussion, except to assume that the boolean domain which is to be used throughout this work will need possibly to be defined together with other semantic disambiguation processes, such as for instance the choice of the meaning of a connective, say, between a strong and a weak meaning.

2.3 A SEMANTICALLY CLEAR VERSION OF *PTQ*

In this section we take a critical view of *PTQ*'s semantics of English and propose for it a new and improved representation. In order to have a self-contained section, we will reexpress its definitions of syntax with minor changes. This presentation will be a step towards a simplified semantical treatment with clear computational presentation, which will follow in chapter three. As we said, the definition for the semantics of *IL* is based on Turner[1983a] using Scott[1981] notation. The main improvements corresponds to make explicit references to function names through out, for instance for the interpretation function of *IL* and for the translation functions on categories and expressions of English. As will be discussed on section 2.4 this is a major improvement. By using the quantifiers defined previously and making explicit use of an equality function, we improve on Turner's presentation, which goes towards our objective of obtaining a fully computational treatment.

2.3.1 Montague's Syntax of English

The definition and presentation of the syntactic system for *PTQ*'s English is basically equal to the one in Montague's original paper. Although syntax is not our main concern here, we note that as the basis for a computational definition of grammar, and to allow for instance efficient parser generation and efficient parsing, Montague's English syntax should be defined and presented as in Pitt[1989]. At the end of this section we give a very simple example of how it could have been done. Pitt's form of definition, based on *LR*-grammars, provides for the automatic construction of an efficient parser generator from the definition.

Syntactic Categories

Definition 2.3.1.1 The set ***CAT*** of *syntactic categories* of Montague's Natural language, *M-NL*, is defined as follows:

1. Basic categories e and t are syntactic categories.
2. If a and b are syntactic categories so is $(a->b)$ and $(a=>b)$.

Definition 2.3.1.2 The following names will be used for the syntactic categories:

t	or *declarative sentences* or just *sentences*.	
IV	or *Intransitive Verb phrases*, is to be the category of	$(e->t)$.
T	or *Terms*,	$(IV->t)$.
TV	or *Transitive Verb phrases*,	$(T->IV)$.
IAV	or *IV-modifying Adverbs*,	$(IV->IV)$.
CN	or *Common Noun phrases*,	$(e=>t)$.
SN	or *Sentence-modifying Adverbs*,	$(t->t)$.
IAVP	or *AV-making Prepositions*,	$(T->IAV)$.
SV	or *Sentence taking Verb phrases*,	$(t->IV)$.
IVV	or *IV-taking Verb phrases*,	$(IV=>IV)$.

Basic Expressions

Definition 2.3.1.3 The sets B_a of *basic expressions* b_a of category a are given:

$B_{IV} = \{run, walk, talk, rise, change\}$.

$B_T = \{John, Mary, Bill, ninety, he_1, he_2, he_3, ...\}$.

$B_{TV} = \{find, lose, eat, love, date, be, seek, conceive\}$.

$B_{IAV} = \{rapidly, slowly, voluntarily, allegedly\}$.

$B_{CN} = \{man, woman, park, fish, pen, unicorn, price, temperature\}$.

$B_{SN} = \{necessarily\}$.

$B_{IAVP} = \{in, about\}$.

$B_{SV} = \{believe\ that, assert\ that\}$.

$B_{IVV} = \{try\ to, wish\ to\}$.

The sets of *basic expressions* of other categories are empty. A *basic expression b* is a member of $B = \Sigma_a B_a$.

Phrases

Definition 2.3.1.4 The sets P_a of *phrases* p_a of category a are introduced by the rules:

Basic rules:

1. For all categories a, any member of B_a is in P_a .
2. If $\alpha \in P_{CN}$ then $F_0(\alpha), F_1(\alpha), F_2(\alpha) \in P_T$, where:
   $F_0(\alpha)$ = ***every*** α,
   $F_1(\alpha)$ = ***the*** α,
   $F_2(\alpha)$ = ***a*** α or $F_2(\alpha)$=***an*** α,
   according as the first word in α takes ***a*** or ***an***.
3. If $\alpha \in P_{CN}$ and $\beta \in P_t$ then $F_{3,n}(\alpha,\beta) \in P_{CN}$, where:
   $F_{3,n}(\alpha,\beta)$= α ***such that*** β',
   and β' comes from β by replacing each occurrence of he_n or him_n by ***he***, ***she***, ***it*** or ***him***, ***her***, ***it*** respectively, according as the first B_{CN} in β is of masculine, feminine or neuter gender.

Rules of functional application:

4. If $\alpha \in P_T$ and $\beta \in P_{IV}$ then $F_4(\alpha,\beta) \in P_t$, where
   $F_4(\alpha,\beta) = \alpha\, \beta'$,
   and β' is the result of replacing the first verb in β by its third person singular present.
5. If $\alpha \in P_{TV}$ and $\beta \in P_T$ then $F_5(\alpha,\beta) \in P_{IV}$, where
   $F_5(\alpha,\beta) = \alpha\, \beta$, if β does not have the form he_n or
   $F_5(\alpha,he_n)=\alpha\, him_n$.
6. If $\alpha \in P_{IAVP}$ and $\beta \in P_T$ then $F_5(\alpha,\beta) \in P_{IAV}$.
7. If $\alpha \in P_{SV}$ and $\beta \in P_t$ then $F_6(\alpha,\beta) \in P_{IV}$, where
   $F_6(\alpha,\beta) = \alpha\, \beta$.
8. If $\alpha \in P_{TIV}$ and $\beta \in P_{IV}$ then $F_6(\alpha,\beta) \in P_{IV}$.

9. If $\alpha \in P_{SN}$ and $\beta \in P_t$ then $F_6(\alpha,\beta) \in P_t$.
10. If $\alpha \in P_{IAV}$ and $\beta \in P_{IV}$ then, $F_7(\alpha,\beta) \in P_{IV}$, where
    $F_7(\alpha,\beta) = \beta\ \alpha$.

Rules of conjunction and disjunction:

11. If $\phi, \psi \in P_t$ then $F_8(\phi,\psi) \in P_t$ and $F_9(\phi,\psi) \in P_t$, where
    $F_8(\phi,\psi) = \phi$ ***and*** ψ ,
    $F_9(\phi,\psi) = \phi$ ***or*** ψ .
12. If $\phi, \psi \in P_{IV}$ then $F_8(\phi,\psi) \in P_{IV}$ and $F_9(\phi,\psi) \in P_{IV}$.
13. If $\phi, \psi \in P_T$ then $F_9(\phi,\psi) \in P_T$.

Rules of quantification:

14. If $\alpha \in P_T$ and $\phi \in P_t$ then $F_{10,n}(\alpha,\phi) \in P_t$, where either 1 or 2:
    1. α does not have the form he_k, and $F_{10,n}(\alpha,\phi)$ comes from ϕ by replacing the first occurrence of he_n or him_n by α, and all other occurrences of he_n or him_n by ***he, she, it*** or ***him, her, it*** respectively, according as the first B_{CN} or B_T in ϕ is of masculine, feminine or neuter gender.
    2. $\alpha = he_k$, and $F_{10,n}(\alpha,\phi)$ comes from ϕ by replacing all occurrences of he_n or him_n by he_k or him_k respectively.
15. If $\alpha \in P_T$ and $\zeta \in P_{CN}$ then $F_{10,n}(\alpha,\zeta) \in P_{CN}$.
16. If $\alpha \in P_T$ and $\delta \in P_{IV}$ then $F_{10,n}(\alpha,\delta) \in P_{IV}$.

Rules of tense and sign:

17. If $\alpha \in P_T$ and $\delta \in P_{IV}$ then
    $F_{11}(\alpha,\delta), F_{12}(\alpha,\delta), F_{13}(\alpha,\delta), F_{14}(\alpha,\delta), F_{15}(\alpha,\delta) \in P_t$, where:
    $F_{11}(\alpha,\delta) = \alpha\ \delta'$, and δ' is the result of replacing the first verb in δ by its negative third person singular present;
    $F_{12}(\alpha,\delta) = \alpha\ \delta''$, and δ'' is the result of replacing the first verb in δ by its third person singular future;

$F_{13}(\alpha,\delta) = \alpha\,\delta'''$, and δ''' is the result of replacing the first verb in δ by its negative third person singular future;

$F_{14}(\alpha,\delta) = \alpha\,\delta''''$, and δ'''' is the result of replacing the first verb in δ by its third person singular perfect;

$F_{15}(\alpha,\delta) = \alpha\,\delta'''''$, and δ''''' is the result of replacing the first verb in δ by its negative third person singular perfect.

A *meaningful* or *well-formed expression p* of *M-NL* is a member of $P = \Sigma_a\ P_a$.

Example from LR-Grammars

Specification languages for grammars have been used to define Montague's English Syntax. With such language we would for instance rewrite rules 7 and 11 on which the syntactic operations correspond to concatenation, e.g. operations F_6 and F_8, as follows:

7. (F_6) *ivphrase ::= svphrase sent*
11. (F_8) *sent ::= sent and sent*

In the work of Pitt[1989], Attribute-translation Grammar, a formalism for grammars based on this kind of notation, is used to define an extension to Montague's fragment of English. Attribute-translation Grammar does not use syncategorematic definitions, but rather extends the definition of the lexicon to include for instance the words corresponding to the logical connectives.

Although forms of syntactic expression are not in our main concern, we will make use of this simplified and more effective type of notation in chapter 3, section 3.5, when introducing and discussing computational grammars.

2.3.2 Montague's Syntax of Intensional Logic *IL*

Types

Definition 2.3.2.1 The set $\mathbb{T}$ of *types* of Montague's Intensional Logic, *IL*, is defined as follows:

1. Basic categories e and t are types.
2. If α and β are types so is $(\alpha \rightarrow \beta)$.
3. If α is a type so is $(s \rightarrow \alpha)$.

Symbols

Definition 2.3.2.2 We will define nine different kind of symbols in the alphabet of the language:

1. Connectives: $\wedge$ $\vee$ $\sim$ $\leftrightarrow$ $\rightarrow$ $=$
2. Quantifiers: $\exists$ $\forall$
3. Modal operators: $\Diamond$ $\Box$
4. Temporal operators: F P
5. Cap cup operators: $\hat{}$ $\check{}$
6. Punctuation marks: . ( )
7. Abstractor: λ
8. Constants $\mathbb{C}_\alpha$ of type α: $C_{\alpha 1}$ $C_{\alpha 2}$ $C_{\alpha 3}$ ... $C_{\alpha i}$ ...
   The set of all constants C of *IL* is $\mathbb{C} = \Sigma_\alpha \mathbb{C}_\alpha$.
9. Variables $\mathbb{V}_\alpha$ of type α: $h_{\alpha 1}$ $h_{\alpha 2}$ $h_{\alpha 3}$ ... $h_{\alpha i}$ ...
   The set of all variables h of *IL* is $\mathbb{V} = \Sigma_\alpha \mathbb{V}_\alpha$.

While in *PTQ* there is no requirement on the denumerability of constants, we impose such condition, as does Turner[1983a].

Expressions

Definition 2.3.2.3 The set $\mathbb{E}_\alpha$ of well-formed expressions of type α in the language *IL*, is defined by the rules:

1. Every constant $C_{\alpha i}$ and every variable $h_{\alpha i}$ is in $\mathbb{E}_\alpha$.
2. If $e_{(\beta \rightarrow \alpha)}$ is of type $(\beta \rightarrow \alpha)$ and e_β is of type β, then $e_{(\beta \rightarrow \alpha)}(e_\beta)$ is in $\mathbb{E}_\alpha$.
3. If α can be analysed as $(\beta \rightarrow \rho)$ then $\lambda h_\beta . e_\rho$ is in $\mathbb{E}_\alpha$.
4. If α is t, e_t, e_{t1} and e_{t2} are any expressions of type t, $e_{\pi 1}$ and $e_{\pi 2}$ are any expressions of the same type π, then the following are in $\mathbb{E}_\alpha$:

$e_{\pi 1} = e_{\pi 2}$	$\sim e_t$
$\Diamond e_t$	$\Box e_t$
$\boldsymbol{F}\, e_t$	$\boldsymbol{P}\, e_t$
$\exists h_\alpha \,.\, e_t$	$\forall h_\alpha \,.\, e_t$
$e_{t1} \wedge e_{t2}$	$e_{t1} \vee e_{t2}$
$e_{t1} \rightarrow e_{t2}$	$e_{t1} \leftrightarrow e_{t2}$

5. If α can be analysed as $(s \rightarrow p)$ then $\hat{}e_p$ is in $\mathbb{E}_\alpha$.
6. If $e_{(s \rightarrow \alpha)}$ is of type $(s \rightarrow \alpha)$ then $\check{}e_{(s \rightarrow \alpha)}$ is in $\mathbb{E}_\alpha$.

The set of all expressions e of *IL* is $\mathbb{E} = \Sigma_\alpha \mathbb{E}_\alpha$.

2.3.3 Domain Spaces and Elements for *IL*'s Semantics

In order to present the semantics of *IL* we will be using a space of domains. As said in section 2.2.1, we will be using Montague's primitive sets extended with an undefined element. Such domains can be seen as complete partial ordered sets with a least element, and are usually named cpo's. The basic elements in each of Montague's set will not be comparable between themselves and the undefined element will be less defined than any of them. The functions to be introduced are now partial and we believe they will capture the natural intuitions. See for instance Turner[1983a] for this choice of domains and functions.

Entities and Contexts of Use

Definition 2.3.3.1 The domain $\mathscr{E}$ of *entities* e. It must be given. An entity can be seen as the most primitive reference we can have.

Definition 2.3.3.2 The domain $\mathscr{I}$ of *contexts of use* or *indices* i. We will define each *index* as a pair of one *world* w, and one *time referent* t. So $\mathscr{I}$ will be a domain built from given $\mathscr{W}$ and $\mathscr{T}$ domains, $\mathscr{I} = \mathscr{W} \times \mathscr{T}$ where $\mathscr{W}$ and $\mathscr{T}$ are given.

In the original *PTQ* Montague does not give a Kripke style semantics for *IL*. For the sake of generality we will assume the existence of an accessibility function, and

define the interpretation of the modal and temporal symbols of our language using Kripke's standard method, in particular, following Fitting's [1983] definitions. Given r', which is the *accessibility function* between world referents and l' which is the *lower or equal than relation* between time referents, we define the appropriate functions on indices or contexts of use.

Definition 2.3.3.3 If indices i_1 and i_2 have the form (ω_1, t_1) and (ω_2, t_2), then given a partial accessibility relation on worlds, r': $\mathcal{W} \times \mathcal{W} \rightarrow \mathcal{B}$ and a partial ordering relation on time, l': $\mathcal{T} \times \mathcal{T} \rightarrow \mathcal{B}$, we define the partial ordering relations r and l as:

$$r, l : \mathcal{I} \times \mathcal{I} \rightarrow \mathcal{B}$$
$$r(i_1, i_2) = \mathbf{cond}(\, r'(\omega_1, \omega_2),\ t, f\,) .$$
$$l(i_1, i_2) = \mathbf{cond}(\, l'(t_1, t_2),\ t, f\,) .$$

Montague's Modal and Temporal Logics

IL is a modal logic, but Montague's formalization of modality and tense was in many ways naive, as Kuhn[1988] points out. However he had the merit of making plausible the use of modal and tense logic in the formalization English. There is no doubt that we can use English in some idiomatic way, which happens to be consistent with the English Montague defined. There are many other uses of English for which the modal characterization is better achieved with other constraints on the semantic primitive domains and relations. Modal characterizations of English usage can be associated with a great variety of modal logics, for instance, *S5, S4, B, K4* as Gallin[1975] proposes. Characterizations of the use of tense in English can also made in many different ways, see for instance Kuhn[1988]. Gabbay[1985] uses a discrete branching presentation of the temporal part of the model and Allen[1984] uses ordering on time intervals.

Meaning in Montague

In natural language every expression could be seen as referring unambiguously to some primitive object, usually called its *extension.* As we have seen this notion of meaning is not enough for compositionality to hold in a theory of meaning. With

different uses expressions have different objects as extensions. This enlarged notion of meaning is usually called the *sense* or *intension* of an expression. In Montague's *PTQ* meaning is reduced to take into account the time and world in which the expression is used. In the formalization of these notions we will use domain spaces instead of sets, to have a computable framework for Montague's denotations.

Definition 2.3.3.4 The domains $\mathcal{M}_\alpha$ of *possible denotations* of type α are constructed as follows.

1. If $\alpha = e$, $\mathcal{M}_\alpha = \mathcal{E}$
2. If $\alpha = t$, $\mathcal{M}_\alpha = \mathcal{B}$
3. For all α such that $\alpha=(\rho\rightarrow\beta)$, $\mathcal{M}_\alpha = \mathcal{M}_\rho \rightarrow \mathcal{M}_\beta$
4. For all α such that $\alpha=(s\rightarrow\beta)$, $\mathcal{M}_\alpha = \mathcal{I} \rightarrow \mathcal{M}_\beta$

The domain of all *possible denotations* m is $\mathcal{M} = \Sigma_\alpha \mathcal{M}_\alpha$.

Definition 2.3.3.5 The domains $\mathcal{S}_\alpha$ of *senses* of type α are defined as:

$$\mathcal{S}_\alpha = \mathcal{I} \rightarrow \mathcal{M}_\alpha$$

The domain of all *senses* s is $\mathcal{S} = \Sigma_\alpha \mathcal{S}_\alpha$.

Definition 2.3.3.6 For each type α, g_α is a *variable-assignment* function defined over the syntactic domain of variables $\mathbb{V}_\alpha$. The function g is the element of the domain $\mathcal{V}$ which 'sums' [7] all g_α,

$$g_\alpha : \mathcal{V}_\alpha = \mathbb{V}_\alpha \rightarrow \mathcal{M}_\alpha, \qquad g: \mathcal{V} = \mathbb{V} \rightarrow \mathcal{M}$$

Definition 2.3.3.7 For each type α, f_α is the given *constant-interpretation* function defined over domain $\mathbb{C}_\alpha$. The function f is the element of the domain $\mathcal{C}$ which sums all f_α,

$$f_\alpha: \mathcal{C}_\alpha = \mathbb{C}_\alpha \rightarrow \mathcal{S}_\alpha, \qquad f: \mathcal{C} = \mathbb{C} \rightarrow \mathcal{S}$$

[7] The precise construction of g from g_α involves the selection of the elements of the disjoint sum domains $\mathbb{V}$ and $\mathcal{M}$. See Stoy[1977] pps. 92-94.

2.3.4 A Domain Semantics of *IL*

Semantic Interpretation

Definition 2.3.4.1 The semantic *interpretation* function u_α, which gives the value in domain $\mathcal{S}_\alpha$ of an expression e_α of $\mathbb{E}_\alpha$, given g of $\mathcal{G}$,

$$u_\alpha : \mathbb{E}_\alpha \rightarrow \mathcal{G} \rightarrow \mathcal{I} \rightarrow \mathcal{M}_\alpha$$

or

$$u_\alpha : \mathbb{E}_\alpha \rightarrow \mathcal{G} \rightarrow \mathcal{S}_\alpha$$

is defined recursively for each type α, following the syntactic definition 2.3.2.2 of language *IL*:

1. If $e_\alpha \in \mathbb{C}_\alpha$ then $u_\alpha[\![e_\alpha]\!]g\,i = f_\alpha[\![e_\alpha]\!]\,i$
   If $e_\alpha \in \mathbb{V}_\alpha$ then $u_\alpha[\![e_\alpha]\!]g\,i = g_\alpha[\![e_\alpha]\!]$

2. If $e_{(\beta\rightarrow\alpha)}(e_\beta)$ is in $\mathbb{E}_\alpha$ then,
   $u_\alpha[\![e_{(\beta\rightarrow\alpha)}(e_\beta)]\!]g\,i = u_{(\beta\rightarrow\alpha)}[\![e_{(\beta\rightarrow\alpha)}]\!]g\,i\,(u_\beta[\![e_\beta]\!]g\,i)$

3. If α is $(\beta\rightarrow\rho)$ and $\lambda v_\beta.\, e_\rho$ is in $\mathbb{E}_\alpha$ then,
   $u_\alpha[\![\lambda v_\beta.\, e_\rho]\!]g\,i = \lambda d_\beta.\, u_\rho[\![e_\rho]\!]g_{[d \leftarrow v]}\,i$

4. If α is t, e_t, e_{t1} and e_{t2} are in $\mathbb{E}_\alpha$, and $e_{\pi1}$, $e_{\pi2}$ are any expressions of the same type π then,

$u_\alpha[\![e_{\pi1} = e_{\pi2}]\!]g\,i$ = **cond** ($u_\pi[\![e_{\pi1}]\!]g\,i$ **equal** $u_\pi[\![e_{\pi2}]\!]g\,i$, t, f)
$u_\alpha[\![\sim e_t]\!]g\,i$ = **cond** ($u_\alpha[\![e_t]\!]g\,i$ **equival** f, t, f)
$u_\alpha[\![\Diamond e_t]\!]g\,i$ = **cond** (**exists** i'. $r(i, i')$ **and** $u_\alpha[\![e_t]\!]g\,i'$, t, f)
$u_\alpha[\![\Box e_t]\!]g\,i$ = **cond** (**forall** i'. $r(i, i')$ **implies** $u_\alpha[\![e_t]\!]g\,i'$, t, f)
$u_\alpha[\![\mathbf{F}\, e_t]\!]g\,i$ = **cond** (**exists** i'. $l(i, i')$ **and** $u_\alpha[\![e_t]\!]g\,i$, t, f)
$u_\alpha[\![\mathbf{P}\, e_t]\!]g\,i$ = **cond** (**exists** i'. $l(i', i)$ **and** $u_\alpha[\![e_t]\!]g\,i$, t, f)
$u_\alpha[\![\exists\, v_\alpha.\, e_t]\!]g\,i$ = **cond** (**exists** d_α. $u_\alpha[\![e_t]\!]g_{[d \leftarrow v]}\,i$, t, f)
$u_\alpha[\![\forall v_\alpha.\, e_t]\!]g\,i$ = **cond** (**forall** d_α. $u_\alpha[\![e_t]\!]g_{[d \leftarrow v]}\,i$, t, f)
$u_\alpha[\![e_{t1} \wedge e_{t2}]\!]g\,i$ = **cond** ($u_\alpha[\![e_{t1}]\!]g\,i$ **and** $u_\alpha[\![e_{t2}]\!]g\,i$, t, f)
$u_\alpha[\![e_{t1} \vee e_{t2}]\!]g\,i$ = **cond** ($u_\alpha[\![e_{t1}]\!]g\,i$ **or** $u_\alpha[\![e_{t2}]\!]g\,i$, t, f)

$$u_\alpha[\![e_{t1} \rightarrow e_{t2}]\!]g\,i\,j = \textit{cond}\,(\,u_\alpha[\![e_{t1}]\!]g\,i\,\textit{implies}\,u_\alpha[\![e_{t2}]\!]g\,i,\,t,f)$$
$$u_\alpha[\![e_{t1} \leftrightarrow e_{t2}]\!]g\,i\,j = \textit{cond}\,(\,u_\alpha[\![e_{t1}]\!]g\,i\,\textit{equival}\,u_\alpha[\![e_{t2}]\!]g\,i,\,t,f)$$

5. If α is $(s\rightarrow\rho)$ and $\hat{}e_\rho$ is in $\mathbb{E}_\alpha$ then,
   $u_\alpha[\![\hat{}e_\rho]\!]g\,i\,j = \lambda_{j'}.\,u_\rho[\![e_\rho]\!]g_{j'}\,i\,j$.

6. If $\check{}e_{(s\rightarrow\alpha)}$ is in $\mathbb{E}_\alpha$ then,
   $u_\alpha[\![\check{}e_{(s\rightarrow\alpha)}]\!]g\,i\,j = u_{(s\rightarrow\alpha)}[\![e_{(s\rightarrow\alpha)}]\!]g\,i\,i\,j$.

The function u is the element which sums all u_α

$$u:\ \mathbb{E}\rightarrow\mathscr{V}\rightarrow\mathscr{I}\rightarrow\mathscr{M}$$

or

$$u:\ \mathbb{E}\rightarrow\mathscr{V}\rightarrow\mathscr{S}$$

Although originally independent, the material above is similar to Turner[1983a] except that the interpretation u is named, and the interpretation of expressions with the cap operator is explicitly given. The critical analysis of *PTQ* which resulted in expressions 5 and 6 above, can be found in section 2.4.1.

Turner's Principle on Interpretations

As we have said one of the objectives of Turner[1983a] was to define a version of Montague's English for which it could be proved that the interpretation function was computable, or more precisely, continuous in each of its arguments. The proof is given by Turner for the basic cases as we referred to in section 2.2. As we discussed in chapter one computability is a basic point and its acceptance can be even used to justify the principle of compositionality. Thus together with the principle of compositionality and the principle of formal reduction for semantics, we should require the interpretation function to be proved continuous, because we want it to be effectively computable.

Semantic Extension and Intension

As a side result of the above treatment we make Montague's notions of extension and intension explicit.

Definition 2.3.4.2 The *extension* of an expression e at i, *ext:* $[\mathbb{E}\times\mathcal{I}]\rightarrow\mathcal{M}$, is defined as:

$$ext(e, i) = u[\![e]\!]g\,i\,.$$

Definition 2.3.4.3 The *intension* of an expression e, *int:* $\mathbb{E}\rightarrow\mathcal{I}$, is defined as:

$$int(e) = \lambda i.\, u[\![e]\!]g\,i$$ [7]

It follows from definitions 2.3.4.2 and 2.3.4.3 that the functions *int* and *ext* are related by the expression,

$$int = curry\,(ext)\,.$$

Thus the definitions we are proposing for intension and extension can be regarded as two different ways of expressing the interpretation of an expression. We should say that the extensional meaning of an expression given a context computes a value, its extension in that context. We should also say that the intensional meaning of an expression is a functional object on contexts of use.

Logical Equivalence and Meaning Postulates

Logical equivalence between sentences is a difficult subject in *PTQ*. According to Montague, a logically possible interpretation is any interpretation u which is logical, i.e., it 'respects' the meaning of the logical constants such as *and*, *or* and *necessity*, and which is possible, i.e., 'respects' a set of expressions, usually called the *meaning postulates*, in the sense that it makes them true in all contexts of use.

Definition 2.3.4.5 Given a set of expressions m_p, the meaning postulates of *IL*, a logically possible interpretation is any interpretation which makes all those expressions true in all indices i,

$$logically_possible\ u = cond(\,forall\,p, i\,.\,u[\![m_p]\!]\,i = t,\ t,\ f\,)$$

[7] We must note that regarding the use of the η reduction rule we should accept the axiom of extensionality, see Barendregt[1981], to be able to reduce *int(e)* to $u[\![e]\!]g$. This relates to Montague's requirement that in general the intension of an expression e, *int(e)*, is the same as $u[\![e]\!]g$.

As Montague says logical truth, logical consequence and logical equivalence are to be characterized accordingly.

The difficulty we mentioned with logical equivalence is that the meaning postulates are supposed to cover all the distinctions of words or phrases of English which were not directly made in the syntactic or in the semantic definitions. If we exclude the postulates, there is no difference between intensional and extensional verbs. It may seem rather contrived to use meaning postulates to define the use of words, which should be clearly and uniformly made in the definition of the language, in its grammar. As well synonymy between words or phrases of English is defined by including in the set of meaning postulates expressions which will guarantee their equivalent use.

Perhaps the most noticeable problem, for a computational treatment, is that quantification on variables ranging over predicates is introduced in *PTQ*'s treatment of English exclusively through the meaning postulates. It is simple to check that there are no such expressions introduced in the translation of English into *IL*.

The restriction on interpretations of Montague's meaning postulates is equivalent to restricting the meaning of the constants of the intermediate language. For instance meaning postulate 1, $\exists x_e . \Box[x_e=c_e]$, is equivalent to imposing a constraint on the constant-interpretation function, 2.3.3.7, for constants of type e in *IL*. It constrains that function to have a constant value on all indices (if *necessarily*, $\Box$, is interpreted as true in all indices). This meaning postulate is required so that proper nouns will be 'logically determinate', i.e. have extensions invariant with respect to indices, possible worlds and moments of time. The proof is straightforward. We require that $u[\![\exists x_e . \Box[x_e=c_e]]\!]g\, i$ should be true in all indices i, and so we have using e as a general variable to decompose the expression,

$$
\begin{aligned}
u[\![\exists x_e.\ e]\!]g\, i &= \textit{cond}\ (\ \textit{exists}\ d_e.\ u[\![e]\!]g_{[d\leftarrow x]}\, i,\ t, f) \\
u[\![\Box e]\!]g_{[d\leftarrow x]}\, i &= \textit{cond}\ (\textit{forall}\ i'.\ r(i, i')\ \textit{implies}\ u[\![e]\!]g_{[d\leftarrow x]}\, i',\ t, f) \\
u[\![x=c]\!]g_{[d\leftarrow x]}\, i' &= \textit{cond}\ (\ u[\![x]\!]g_{[d\leftarrow x]}\, i'\ \textit{equal}\ u[\![c]\!]g_{[d\leftarrow x]}\, i',\ t, f) \\
u[\![c]\!]g_{[d\leftarrow x]}\, i' &= f[\![c]\!]\, i'
\end{aligned}
$$

$$u[\![x]\!]g_{[\alpha \leftarrow x]}\, i' \quad = g_{[\alpha \leftarrow x]}[\![x]\!] = \alpha_e$$

Therefore,

$$u[\![x=c]\!]g_{[\alpha \leftarrow x]}\, i' \quad = \mathit{cond}\,(\,\alpha_e \,\mathit{equal}\, f[\![c]\!]\, i',\ t, f)$$

If r is always true, $r(i, i') = t$, we require that for all indices i there is an entity α such that $f[\![c]\!]\, i$ is equal to it. This is the same as saying that $f[\![c]\!]$ is a constant function on indices, or that proper nouns are logically determinate. The other direction of the argument is also valid. If $f[\![c]\!]$ is a constant function, $\exists x_e.\ \Box[x_e = c_e]$ is true everywhere.

To accept the other expressions as meaning postulates is equivalent to placing similar constraints on the constant-interpretation of the other constants of *IL*. We will see some of the constraints in chapter three, section 3.2.3 on *Meaning Postulates*. This analysis raises the question of where is the best place to have the restrictions which we will try to answer in the last section of chapter 3. If we have the intuition that proper nouns have extensions invariant with respect to indices, i.e., that the constant-interpretation is constant on indices, should we restrict the interpretation directly or introduce an expression which we declare true? The same comment applies to extensional versus intensional verbs. We learn to classify verbs into extensional and intensional, and to distinguish their interpretation and use. Our recognition of that difference can be associated with the recognition of their different forms. The verb 'love' is an extensional verb, the verbs 'seek' and 'believe' are intensional. In most of the contexts, of course.

2.3.5 The Semantics of *M-NL* through Translation into *IL*

Translation on Syntactic Categories

In the following we basically reexpress the translation function described by Montague in a more natural prefix form.

Definition 2.3.5.1 The function from syntactic categories of English to types of language *IL*, τ_{IL}: $CAT \rightarrow \mathbb{T}_{IL}$ is defined as follows:

1. $\tau(e) = e$
2. $\tau(t) = t$
3. $\tau(a->b) = (s \rightarrow \tau(a)) \rightarrow \tau(b)$ and
   $\tau(a=>b) = (s \rightarrow \tau(a)) \rightarrow \tau(b)$, for all categories a and b.

Translation on Phrases

Definition 2.3.5.2 The following translation rules from the domain of well-formed phrases P of *M-NL* to the domain of well-formed expressions $\mathbb{E}$ of language *IL*, define function t_{IL}, t_{IL}: $P \rightarrow \mathbb{E}$. We will be using the following names in this definition,

j	is $C_{e\,1}$,	x	is $h_{(s\rightarrow e)\,1}$,	p	is $h_{(s\rightarrow t)\,1}$,
m	is $C_{e\,2}$,	y	is $h_{(s\rightarrow e)\,3}$,		
b	is $C_{e\,3}$,			P	is $h_{(s\rightarrow((s\rightarrow e)\rightarrow t))\,1}$,
n	is $C_{e\,4}$,	x_n	is $h_{(s\rightarrow e)\,2n}$,	Z	is $h_{(s\rightarrow((s\rightarrow((s\rightarrow e)\rightarrow t))\rightarrow t))\,1}$.

Basic Rules:

1.1. For all n:

$$t\{he_n\} = \lambda P.\ \check{}P(x_n) \qquad = he_{IL\,n} \qquad = h_{\tau(T)\,n} \quad \in \mathbb{V}_{\tau(T)}.$$

1.2. For every other *basic expression* b_a of *M-NL*, $t\{b_a\}$ is a given constant $C_{\tau(a)}$ of *IL*. For simplification of use, if ***bexp*** is the identifier used as b_a, then *bexp* is to be the identifier used for the translation in *IL*, $t\{bexp\}=bexp$; e.g., the following constants have the form shown:

$$t\{be\} = \lambda Z.\ \lambda x.\ Z(\lambda y.\ \check{}x = \check{}y) \quad = be_{IL} \quad = C_{\tau(TV)\,6} \quad \in \mathbb{C}_{\tau(TV)},$$
$$t\{necessarily\} = \lambda p.\ \Box\,\check{}p \quad = necessarily_{IL} \quad = C_{\tau(SA)\,1} \quad \in \mathbb{C}_{\tau(SA)},$$
$$t\{John\} = \lambda P.\ \check{}P(\hat{}j) \quad = John_{IL} \quad = C_{\tau(T)\,1} \quad \in \mathbb{C}_{\tau(T)},$$
$$t\{Mary\} = \lambda P.\ \check{}P(\hat{}m) \quad = Mary_{IL} \quad = C_{\tau(T)\,2} \quad \in \mathbb{C}_{\tau(T)},$$
$$t\{Bill\} = \lambda P.\ \check{}P(\hat{}b) \quad = Bill_{IL} \quad = C_{\tau(T)\,3} \quad \in \mathbb{C}_{\tau(T)},$$
$$t\{ninety\} = \lambda P.\ \check{}P(\hat{}n) \quad = ninety_{IL} \quad = C_{\tau(T)\,4} \quad \in \mathbb{C}_{\tau(T)}.$$

2. If $\alpha \in P_{CN}$ then,

$$t\{every\ \alpha\} = \lambda P.\ \forall x.\ (t\{\alpha\}(x) \rightarrow \check{}P(x)),$$
$$t\{the\ \alpha\} = \lambda P.\ \exists y.\ (\forall x.\ (t\{\alpha\}(x) \leftrightarrow x=y) \wedge \check{}P(y)),$$

$$t\{F_2(\alpha)\} = \lambda P.\ \exists x.\ (\ t\{\alpha\}(x) \wedge \check{}P(x)\ )\ .$$

3. If $\alpha \in P_{CN}$ and $\beta \in P_t$ then,

$$t\{F_{3,n}(\alpha,\phi)\} = \lambda x_n.\ (\ t\{\alpha\}(x_n) \wedge t\{\beta\}\ )\ .$$

Rules of functional application:

4. If $\alpha \in P_T$ and $\beta \in P_{IV}$ then,

$$t\{F_4(\alpha,\beta)\} = t\{\alpha\}(\hat{}t\{\beta\})\ .$$

5. If $\alpha \in P_{TV}$ and $\beta \in P_T$ then,

$$t\{F_5(\alpha,\beta)\} = t\{\alpha\}(\hat{}t\{\beta\})\ .$$

6. If $\alpha \in P_{IAVP}$ and $\beta \in P_T$ then,

$$t\{F_4(\alpha,\beta)\} = t\{\alpha\}(\hat{}t\{\beta\})\ .$$

7. If $\alpha \in P_{SV}$ and $\beta \in P_t$ then,

$$t\{F_6(\alpha,\beta)\} = t\{\alpha\}(\hat{}t\{\beta\})\ .$$

8. If $\alpha \in P_{TIV}$ and $\beta \in P_{IV}$ then,

$$t\{F_6(\alpha,\beta)\} = t\{\alpha\}(\hat{}t\{\beta\})\ .$$

9. If $\alpha \in P_{SN}$ and $\beta \in P_t$ then,

$$t\{F_6(\alpha,\beta)\} = t\{\alpha\}(\hat{}t\{\beta\})\ .$$

10. If $\alpha \in P_{IAV}$ and $\beta \in P_{IV}$ then,

$$t\{F_7(\alpha,\beta)\} = t\{\alpha\}(\hat{}t\{\beta\})\ .$$

Rules of conjunction and disjunction:

11. If $\phi, \psi \in P_t$ then,

$$t\{\phi \textbf{ and } \psi\} = t\{\phi\} \wedge t\{\psi\}\ ,$$

$$t\{\phi \textbf{ or } \psi\} = t\{\phi\} \vee t\{\psi\}\ .$$

12. If $\phi, \psi \in P_{IV}$ then,

$$t\{\phi \textbf{ and } \psi\} = \lambda x.\ (t\{\phi\}(x) \wedge t\{\psi\}(x))\ ,$$

$$t\{\phi \textbf{ or } \psi\} = \lambda x.\ (t\{\phi\}(x) \vee t\{\psi\}(x))\ .$$

13. If $\phi, \psi \in P_T$ then,

$$t\{\phi \textbf{ or } \psi\} = \lambda P.\ (t\{\phi\}(P) \vee t\{\psi\}(P))\ .$$

Rules of quantification:

14. If $\alpha \in P_T$ and $\phi \in P_t$ then,

$$t\{F_{10,n}(\alpha,\phi)\} = t\{\alpha\}(\lambda x_n.\ t\{\phi\}) .$$

15. If $\alpha \in P_T$ and $\zeta \in P_{CN}$ then,

$$t\{F_{10,n}(\alpha,\zeta)\} = \lambda y.\ t\{\alpha\}(\lambda x_n.\ t\{\zeta\}(y)) .$$

16. If $\alpha \in P_T$ and $\delta \in P_{IV}$ then,

$$t\{F_{10,n}(\alpha,\delta)\} = \lambda y.\ t\{\alpha\}(\lambda x_n.\ t\{\delta\}(y)) .$$

Rules of tense and sign:

17. If $\alpha \in P_T$ and $\delta \in P_{IV}$ then,

$$t\{F_{11}(\alpha,\delta)\} = \sim(t\{\alpha\}(\hat{}t\{\delta\})) ,$$
$$t\{F_{12}(\alpha,\delta)\} = F(t\{\alpha\}(\hat{}t\{\delta\})) ,$$
$$t\{F_{13}(\alpha,\delta)\} = \sim F(t\{\alpha\}(\hat{}t\{\delta\})) ,$$
$$t\{F_{14}(\alpha,\delta)\} = P(t\{\alpha\}(\hat{}t\{\delta\})) ,$$
$$t\{F_{15}(\alpha,\delta)\} = \sim P(t\{\alpha\}(\hat{}t\{\delta\})) .$$

Extension and Intension of Phrases

The definitions of intension and extension for the intensional language can be extended for expressions of natural language in the obvious way:

Definition 2.3.5.3 The *extension* of a well-formed expression *exp* of *M-NL* for a given context of use i, *extension:* $[P \times \mathcal{I}] \rightarrow \mathcal{M}$, is defined as,

$$extension(exp, i) = u[\![t\{exp\}]\!]g\, i .$$

Definition 2.3.5.4 The *intension* of a well-formed expression *exp* of *M-NL*, *intension:* $P \rightarrow \mathcal{I}$, is defined as:

$$intension(exp) = \lambda i .\ u[\![t\{exp\}]\!]g\, i .$$

Definition 2.3.5.5 The *meaning* of a well-formed expression *exp* of *M-NL* is identified with the *intension* function defined before,

$$meaning = intension$$

2.4 INTENTIONALITY IN *PTQ*

There are two formalizations for intentionality [8] in *PTQ*. There is the intension, interpretation, of an expression ζ, and the intension, concept $\hat{}\zeta$, of an expression ζ. We argue that the latter is counter intuitive, and it only survived in *PTQ* because Montague used two distinct functional notations.

In this section we summarize Montague's definitions for intensional and extensional interpretations, and for the cap cup operators. The use of these operators creates counter-intuitive results as shown by Dowty et al[1983]. We claim that Gallin's [1975] observations, referred to in Dowty, are not correct, and that their use is puzzling and their definitions are generally counter-intuitive. We then follow the semantical treatment of compound expressions involving oblique or intensional contexts from their English definition, through their translation into *IL* and interpretation. We conclude that Montague's formalization of English, with correct compositional intentionality, was achieved through a difficult use of the cap cup operators and in spite of a counter intuitive definition.

2.4.1 Montague's Definitions

Montague accepted Frege's functional analysis of natural language. In order to provide a correct formalization it was necessary to define formally Frege's notion of intentionality. In order to do this Montague introduced a language for which the principle of compositionality of extensions holds (see Gallin[1975]) and he defined for expressions of this language both an extensional and intensional semantics. Then he provided a translation (syntax to syntax) from English phrases into expressions of this language, so securing indirectly an intensional and extensional interpretation for English. In its intermediate language he had introduced two operators $\hat{}$ and $\check{}$, usually referred to as the intensional operators. Intuitively, the intensional operator was used in the translation of compound expressions when oblique contexts might require intensional meanings and the extensional operator if the translation of compound expressions did not require intensional meanings. We

[8] See footnote 1 of this chapter.

will reproduce here the notation used by Montague and some of its definitions. We will argue that Montague used an intermediate language with excessive expressive power by introducing the intensional operators. Dowty's example shows that its full use is counterintuitive. Gallin shows that Montague's translations of natural language phrases only covers a subset of expressions of *IL*.

In the following section we show how Montague's double notation contributed to keep the double formalization of intentionality unnoticed. It is remarkable how Montague managed to reconcile the semantic definitions of intension and extension with the definitions of the cap cup operators and the definition of the translation from English, in such a way that the counterintuitive results to be found in pure *IL* never have to be used. We will definitions M.1-M.4 to refer to Montague's original definitions.

Extensional and Intensional Meaning for Expressions in IL

Definition M.1 The *extension* of an expression *exp* with respect to U (interpretation), i (world-time index) and g (variable-assignment), is denoted by $exp^{U,g,i}$. This extension is defined recursively and compositionally on the expressions of the intensional language.

Definition M.2 The *intension* of an *exp* with respect to U and g, is denoted by $exp^{U,g}$ and is defined as the function h with the domain of indices i such that $h(i) = exp^{U,g,i}$:

$exp^{U,g} = h$ *such that*
$h(i) = exp^{U,g,i}$

or

$exp^{U,g} = \lambda i.\ exp^{U,g,i}$

The following equality holds by definition:

$exp^{U,g,i} = exp^{U,g}(i)$

Intensional Operator

Definition M.3 The extension of the intensional operator, $[\hat{}exp]^{U,g,i}$, is defined as the function h with the domain of indices i such that $h(i) = exp^{U,g,i}$. This form of the definition is ambiguous because i is used in three different places. But everyone agreed that it was obvious which was the scope intended:

$$[\hat{}exp]^{U,g,i} = h \text{ such that}$$
$$h(i) = exp^{U,g,i}$$

or

$$[\hat{}exp]^{U,g,i} = \lambda j.\ exp^{U,g,j}$$

We notice the similarity of this definition for extensional meaning of the intensional operator with the definition of intensional meaning in M.2. Its intension under the obvious interpretation using M.2 is,

$$[\hat{}exp]^{U,g} = \lambda i.\ [\hat{}exp]^{U,g,i} = \lambda i.\ \lambda j.\ exp^{U,g,j}$$

Extensional Operator

Definition M.4 The extension of the extensional operator, $[\check{}exp]^{U,g,i}$, is defined as,

$$[\check{}exp]^{U,g,i} = exp^{U,g,i}(i)$$

The intensional meaning under the obvious interpretation is,

$$[\check{}exp]^{U,g} = \lambda i.\ exp^{U,g,i}(i)$$

This cannot be further reduced as the η reduction rule cannot in general be applied if i is a free variable,

$$[\check{}exp]^{U,g} \neq exp^{U,g,i}$$

What we have done in section 2.3 was to identify the interpretation U with u, the world-time index i with i and the variable-assignment g with g and made use of a functional notation to define the interpretation function. Then the intensional and

extensional meaning functions are defined for expressions of *IL* using the interpretation function.

2.4.2 Using the Intensional Operators

Dowty's Example

Returning to our notational system, we will present an example where the definitions of intension, extension and the interpretation of the intensional operators have been found to create an intuitively problematic result, the failure in the equivalence between the interpretations of k and $\hat{}\check{}k$, where k denotes a constant of Montague's *IL*. It is also counter intuitive that the intension of $\check{}k$ is not the extension of k.

We will calculate as in Dowty *et al*[1981] the meaning of a constant k of type $s \rightarrow e$ and of the related expressions $\check{}k: e$, $\hat{}k: s \rightarrow (s \rightarrow e)$, $\check{}\hat{}k: s \rightarrow e$ and $\hat{}\check{}k: s \rightarrow e$. We will use §5 and §6 of definition 2.3.4.1 for the meaning of the intensional operators. The interpretation $\boldsymbol{u}$ is constructed on a finite set of entities $\mathcal{E} = \{a, b, c, d\}$, and a finite set of points of reference $\mathcal{I} = \{1, 2, 3, 4\}$. Let f, the interpretation function for constants of the language, be given with the value for k as shown below in a convenient graphic representation:

$$f[\![k]\!] = \begin{bmatrix} 1 & \begin{bmatrix} 1 & a \\ 2 & a \\ 3 & a \\ 4 & a \end{bmatrix} \\ 2 & \begin{bmatrix} 1 & a \\ 2 & b \\ 3 & c \\ 4 & d \end{bmatrix} \\ 3 & \begin{bmatrix} 1 & c \\ 2 & b \\ 3 & d \\ 4 & a \end{bmatrix} \\ 4 & \begin{bmatrix} 1 & c \\ 2 & d \\ 3 & a \\ 4 & b \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & A \\ 2 & B \\ 3 & C \\ 4 & D \end{bmatrix}$$

Meaning of k. The intension of the constant expression k is $f[\![k]\!]$, as shown above.

$$int[\![k]\!] = f[\![k]\!]$$

Meaning of ˇk. The intension of the expression $\check{}k$ is:

$$int[\![\check{}k]\!] = \lambda i.\, u[\![\check{}k]\!]\, g\, i = \lambda i.\, u\, [\![k]\!]\, g\, i\, i = \lambda i.\, f[\![k]\!]\, i\, i$$

$$int[\![\check{}k]\!] = \begin{bmatrix} 1 & a \\ 2 & b \\ 3 & d \\ 4 & b \end{bmatrix}$$

Conclusion 2.4.2.1 $\sim (\forall i.\, ext\,([\![k]\!], i) = int\,[\![\check{}k]\!])$

Meaning of ^k. The intension of the expression $\hat{}k$ is:

$$int[\![\hat{}k]\!] = \lambda i.u[\![\hat{}k]\!]\, g\, i = \lambda i.\lambda j.\, u\, [\![k]\!]\, g_j\, i = \lambda i.\, f[\![k]\!]$$

$$int[\![\hat{}k]\!] = \begin{bmatrix} 1 & \begin{bmatrix} 1 & A \\ 2 & B \\ 3 & C \\ 4 & D \end{bmatrix} \\ 2 & \begin{bmatrix} 1 & A \\ 2 & B \\ 3 & C \\ 4 & D \end{bmatrix} \\ 3 & \begin{bmatrix} 1 & A \\ 2 & B \\ 3 & C \\ 4 & D \end{bmatrix} \\ 4 & \begin{bmatrix} 1 & A \\ 2 & B \\ 3 & C \\ 4 & D \end{bmatrix} \end{bmatrix}$$

Conclusion 2.4.2.2 $\forall i.\, ext\,([\![\hat{}k]\!], i) = int\,[\![k]\!]$

Meaning of $\hat{}\check{}k$. The intension of the expression $\hat{}\check{}k$ is:

$$int\ [\![\hat{}\check{}k]\!] = \begin{bmatrix} 1 & \begin{bmatrix} 1 & a \\ 2 & b \\ 3 & d \\ 4 & b \end{bmatrix} \\ 2 & \begin{bmatrix} 1 & a \\ 2 & b \\ 3 & d \\ 4 & b \end{bmatrix} \\ 3 & \begin{bmatrix} 1 & a \\ 2 & b \\ 3 & d \\ 4 & b \end{bmatrix} \\ 4 & \begin{bmatrix} 1 & a \\ 2 & b \\ 3 & d \\ 4 & b \end{bmatrix} \end{bmatrix}$$

Conclusion 2.4.2.3 $int\ [\![\hat{}\check{}k]\!] \neq int\ [\![k]\!]$

Meaning of $\check{}\hat{}k$. The intension of the expression $\check{}\hat{}k$ is:

$$int\ [\![\check{}\hat{}k]\!] = \begin{bmatrix} 1 & A \\ 2 & B \\ 3 & C \\ 4 & D \end{bmatrix}$$

Conclusion 2.4.2.4 $int\ [\![\check{}\hat{}k]\!] = int\ [\![k]\!]$

Dowty's Conclusion

Dowty *et al* does not explain why the equivalence does not hold. However it is observed that there are two reasons to ignore the failure of the equivalence between $\hat{}\check{}k$ and k. The first one is that in interpreting English, constants of type $s \rightarrow \alpha$ are never used. The second reason is that expressions of the form $\hat{}\check{}\alpha$ will never arise in translating English.

Gallin's Observation Criticized

Gallin[1975] suggested that $\hat{}$ works as a kind of hidden functional abstractor on indices. Although indices do not appear in *IL*, $\hat{}$ acts as if it takes an expression α and forms from it another expression denoting a function on indices which gives as

value for any index the extension of α at that index. Taking into account Dowty's example, the previous proposition exposes a view of the $\hat{}$ operator which is extremely confusing. The result of using it on an expression α just takes whatever the object denoted by α is (the intension of α) and constructs a constant function on indices. The operator $\hat{}$ should be seen *just* as a constant function constructor. On the other hand Gallin says that $\check{}$ acts as a kind of hidden 'functional application' operation on indices; it takes the function denoted by α and 'applies' it to whatever index $\check{}\alpha$ is being evaluated at. We also think that this is an extremely confusing view of that operator's use. What it does, except in the general case where α denotes a constant function on indices, is to destroy the information contained on the function of type $\mathcal{I} \rightarrow \mathcal{I} \rightarrow \mathcal{D}$ by creating a completely new function of type $\mathcal{I} \rightarrow \mathcal{D}$. If all expressions are 'protected' by the $\hat{}$ operator there is no danger of using the operator. This is the explanation for the fact that $\check{}\hat{}\alpha$ is always equivalent to α. It also explains why the other equivalence is not valid except when α denotes a constant function on indices, not just a constant of *IL*. It is straightforward to prove that if β denotes a constant function on indices, e.g. $\beta = \hat{}\alpha$, or that for all indices i, $u[\![\beta]\!]i$ is a constant, then the other equivalence always holds, $\hat{}\check{}\beta = \beta$. As we will see in the following, it is not so strange that these operators are so puzzling when used in the treatment of intentionality.

2.4.3 The Double Formalization of Intentionality in *PTQ*

Failure of Compositionality in NL

Montague recognized that the naive composition of extensions fails to capture the intensional aspects present in natural language. In general if F_n is a syntactic function or structural operation and α, β and γ are natural language phrases, the second equality does not hold in general, because there is no such function S_n :

$$\alpha = F_n(\beta, \gamma)$$
$$* \quad ext(\alpha) = S_n(ext(\beta), ext(\gamma))$$

For example if the syntactic operation F_n corresponds to simple concatenation or F_6 in *PTQ*, and S_n corresponds to functional application we would write,

$$\alpha ::= \beta\,\gamma$$

* $ext(\alpha) = ext(\beta)\{ext(\gamma)\}$

The classical examples involve replacement of coextensional phrases in compound expressions involving necessity, 'Necessarily the morning star is the evening star', and compound expressions involving time, in a context where Jones is a US senator, 'colleague of Jones' is coextensional with 'US senator' but we do not have that 'former colleague of Jones' means the same as 'former US senator'.

Compositionality in IL

According to Gallin[1975], Frege did not abandon the functionality principle for extensions, but rather held that the extension of an expression depends on the syntactic context in which it occurs. In principle one could eliminate the difficulties by introducing, for each expression ζ, a new expression $\hat{}\zeta$, called the concept of ζ, whose extension is the intension of ζ. For this language then we would have that,

$$\alpha = F_n(\beta, \gamma)$$
$$ext(\alpha) = S_n(ext(\hat{}\beta), ext(\hat{}\gamma))$$

Montague's *IL* is such a language. In Montague's notation we would write,

$$\alpha = f_n(\beta, \gamma)$$
$$\alpha^{U,g,i} = s_n(\hat{}\beta\,^{U,g,i}, \hat{}\gamma\,^{U,g,i}) \qquad \text{or}$$
$$\alpha^{U,g,i} = s_n(\beta\,^{U,g}, \gamma\,^{U,g})$$

For rule 4 in the definition of *IL* in *PTQ* we would write,

$$\alpha ::= \beta(\gamma)$$
$$\alpha^{U,g,i} = \beta\,^{U,g,i}(\gamma\,^{U,g,i})$$

However this notation does not clearly expose the proposal, which Gallin attributes to Carnap, that the extension of an expression is really a function on two variables, the expression and the state of affairs or context of use. For closed expressions, which contain no free variables, we should write in our notation:

$$ext(\alpha, i) = S_n(ext(\beta, i), ext(\gamma, i))$$

and for rules 4, 9 and 10,

$$\alpha ::= \beta(\gamma)$$
$$ext(\alpha, i) = ext(\beta, i)\ \{ ext(\gamma, i) \}$$

$$\alpha ::= \hat{}\gamma$$
$$ext(\alpha, i) = \lambda j.\ ext(\gamma, j)$$

$$\alpha ::= \check{}\gamma$$
$$ext(\alpha, i) = ext(\gamma, i)\ (i)$$

We must stress that these definitions are for expressions of *IL*. This is supposed to be an extensional language, i.e. such that the extension of any expression at any index is either given or compositionally evaluated. This compositionality tells us that each expression of the language has one object standing for meaning. As the equations above show the use of the cap operator corresponds to constructing a constant function on indices which value is itself a function, $\lambda j.\ ext(\gamma, j)$, the meaning of γ. So this operator just constructs a constant function on indices. If we could formalize 'information' we certainly would say that it does not create information. The extensional operator on the other hand would clearly be destroying information, except for the case where γ meant a constant function on indices. This can be seen as a starting point to give an intuitive explanation for Dowty's example. If k was a constant function, not just a constant of *IL* which is not interpreted as a constant function, the equivalences would hold. As it is in *IL*, to take the extension of an *IL* constant corresponds to losing information on the meaning of that constant. But as Dowty *et al* also show, that never happens in Montague Semantics. Before showing why, we observe that Dowty's result is telling us that Montague is not using all the expressive power of *IL*.

Translation from NL into IL

After defining *IL*, Montague translates English phrases into *IL* expressions using a translation function t. The *meaning* of English is then defined as,

$$meaning\ \alpha\ i\ = ext\,(t\,(\alpha), i)$$

or

$$meaning\ \alpha\ = \lambda i.\ ext\,(t\,(\alpha\,),\ i)$$

In particular for all compound expressions in which the oblique contexts could arise Montague made his system to look like,

$$\alpha ::= \beta(\gamma)$$
$$t(\alpha) = t(\beta)\,(\hat{}\,t(\gamma))$$

$$\begin{aligned} meaning\ \alpha\ i\ &= ext\,(t\,(\alpha\,),\ i) \\ &= ext(t\,(\beta\,)\,(\hat{}\,t\,(\gamma)\,),\ i\,) \\ &= ext(t\,(\beta\,),\ i\,)\,\{\ ext(\hat{}\,t\,(\gamma)\,),\ i\,)\,\} \\ &= ext(t\,(\beta\,),\ i\,)\,\{\ \lambda j.\ ext(t\,(\gamma)\,,\ j\,)\,\} \\ meaning\ \alpha\ &= \lambda i.\ ext(t\,(\beta\,),\ i\,)\,\{\ \lambda j.\ ext(t\,(\gamma)\,,\ j\,)\,\} \end{aligned}$$

Compositionality Restored in NL

The last equations became easier to write if we use intension, *int*, instead of extension *ext*, where as we have seen in section 2.3.4, *(int (e)) i = ext (e,i)* :

$$\begin{aligned} meaning\ \alpha\ &= \lambda i.\ (\{int\,(t\,(\beta\,))\ \ i\ \}\ \{\ \lambda j.\ int(t\,(\gamma))\,j\,\}\ ) \\ &= \lambda i.\ \{int(t\,(\beta\,))\,i\,\}\ \{int\,(t\,(\gamma))\,\} \end{aligned}$$

It is easy to show, by taking a 'curried' variant of the right hand side, that this corresponds through an isomorphic operation to restoring compositionality of intensions for natural language: Assume both,

$meaning\ \alpha : \mathcal{I} \to \Xi$ and $int\,(t\,(\gamma)) : \Psi$; then,
$int(t\,(\beta\,)) : \mathcal{I} \to (\Psi \to \Xi)$; but as we also have that,
$\mathcal{I} \to (\Psi \to \Xi) \equiv \Psi \to (\mathcal{I} \to \Xi)$.

Therefore we should be able to conclude that the meaning of α , *meaning* α, is isomorphic to the meaning of $\lambda i.\ int_cur(t(\beta))\{int\,(t(\gamma))\,\}i$. With meanings seen as intensions 'curried' in the way above if necessary, we would conclude that,

$$meaning\ \alpha\ = meaning\ \beta\ \{meaning\ \gamma\}$$

At this stage we believe that Montague could have achieved this result directly if he had started by using the following definitions,

$$\alpha_{IL} = f_n(\beta_{IL}, \gamma_{IL})$$
$$\alpha_{IL}{}^{U,g} = s_n(\beta_{IL}{}^{U,g}, \gamma_{IL}{}^{U,g})$$

or,

$$\alpha_{IL} ::= \beta_{IL}(\gamma_{IL})$$
$$int(\alpha_{IL}, i) = int(\beta_{IL})\{int(\gamma_{IL})\}$$
$$t(\alpha) = t(\beta)(t(\gamma))$$

and with,

$$meaning\ \alpha \quad = int(t(\beta))\{int(t(\gamma))\}$$

A very similar thing happens with the treatment of necessity but at a syncategorematic level. At this stage we are almost tempted to think that Montague in fact did not really use the cap-cup operators for his semantics of English. If he had written clause 4 in the definition of the interpretation of *IL* using only one type of notation, he surely would have realized that that definition together with his definition of the translation would be achieving the equivalent of composing intensions directly in English or in *IL*. In chapter three we will prove that we can avoid the use the operators while we still have for any English expression the same meaning induced in *PTQ* by the use of *IL*.

2.4.4 Excessive Expressiveness of *IL*

The observations in Dowty *et al*[1981] which were used to justify the use of the cap-cup operators are equivalent to say that *IL* has a subset of expressions which is not required to formalize Montague's English. In the following chapter we will show that we do not need the cap-cup operators to achieve the same interpretation for Montague's English. This result may as well seem puzzling, given that we are used to think that intensional verbs need the cap-cup operators 'expressive power' to be formalized. We could have shown, but that would involve a rather long proof, that the meaning postulates involving the distinction between intensional and extensional verbs are just constraining the constant interpretation of those verbs to

be extensional on the subject or object argument. This same constraint can be written using a normal extensional logic without the operators, directly on the definition of the function. We will not follow this as we will be proposing a new approach by the end of the next chapter.

Dowty's result also reminds us that it is not sufficient to show that *IL* is a high-order logic to show that Montague's English is equivalent to a high order logic. *IL* can be a high-order logic and still we may be using in translating English only its first-order subset. To show that Montague's English is equivalent to a logic where we quantify on predicates we would need to show that in its translation we are using that possibility of *IL*, and exactly where are expressions of *IL* being introduced which are not first-order.

2.5 CONCLUSIONS

The use of denotational semantics brings into Montague semantics an unified notational description. This helps in exposing the counter-intuitive use of the cap-cup operators, and provides insights on how they are used in the meaning postulates. We observe as well that meaning postulates introduce in *PTQ* high-order quantification. We conclude that those operators have undesirable properties and that their use in the formalization of intentionality seems unnecessary. The main conclusion of this chapter is that we should use neither the cap-cup operators nor meaning postulates.

Denotational semantics also provides the required compositional and computational basis to express a semantics of natural language on which Montague's treatment can be expressed. Its methodology requires that the presentation of the interpretation be proved to be effectively computable.

> By *method* is to be understood the manner in which a certain object is to be thoroughly known. It must be derived from the nature of the science itself, and accordingly being a necessary order of thought determined thereby, it cannot be altered. *Treatment* signifies only the manner of communicating one's thoughts to others, in order to make a doctrine intelligible.
>
> In Logic I do not ask *what* the understanding knows, and *how much* it can know, or *how far* its knowledge reaches; for that would be self cognition in respect of its material use, and therefore belongs to metaphysics. In Logic, the question is only: *How will the understanding know itself* ?
>
> Kant[1800], pp. 10 & 4

CHAPTER 3

A SIMPLIFIED TREATMENT OF MONTAGUE'S ENGLISH

3.1 INTRODUCTION

Following our previous observations, the first part of this chapter will introduce a simplified semantical treatment of Montague's English. This results mainly from eliminating from the intensional language the cap-cup operators. We will show that their use in the intensional logic is unnecessary in having a compositional semantics for English isomorphic to the one in *PTQ*. The type structure of the intermediate language and its denotations and the translation functions are much simplified. The simplification we will propose brings the intermediate logical notation closer to a simply typed Modal and Temporal Logic. Although we still allow higher order quantification, it will not be used in translating English. The advantage of using such logics is obvious as they provide languages 'closer' to natural language and

their computational characterization is simplified. For instance, theorem proving is better understood and tractable [1], which does not happen with high-order logics.

The definition of *M-NL* in the previous chapter uses basically the same intensional logic as *PTQ*. As we have seen we can visualize the definition of *M-NL* with the following diagram.

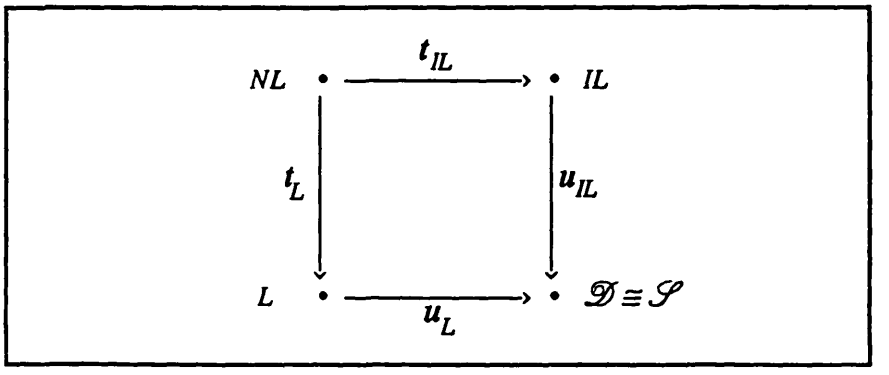
$$M\text{-}NL_{\perp} \xrightarrow[\text{translation}]{t} IL_{\perp} \xrightarrow[\text{interpretation}]{u} \mathscr{S}_{\perp}$$

The simplification we will introduce in sections 3.2 through 3.4 to Montague's definition concerns mainly the intermediate language. We will define an intensional language, without the cap-cup operators, language *L*, and with a simplified translation t_L and interpretation u_L we will similarly provide a characterization for *M-NL*:

$$M\text{-}NL_{\perp} \xrightarrow[\text{translation}]{t} L_{\perp} \xrightarrow[\text{interpretation}]{u} \mathscr{D}_{\perp}$$

We will then show that with the given translation and interpretation functions, t_L and u_L, for any expression *exp* of *M-NL* we will have an isomorphic semantics:

$$\mathscr{D} \cong \mathscr{S}$$
$$u_L[\![\, t_L\{\, exp \,\}]\!] \cong u_{IL}[\![\, t_{IL}\{\, exp \,\}]\!]$$



[1] For recent engineering of theorem provers for modal and temporal logics see de la Quintana[1988].

In the last part of this chapter, section 3.5, we will briefly discuss a possible theory for learning, implementing and extending computable grammars for natural language. We do this by imposing computational restrictions on Montague's program, such as restrictions resulting from accepting only computable syntax and semantics, definable with effectively computable functions. One of the conclusions of imposing such restrictions is that meaning postulates not only can be avoided but should be so, therefore eliminating high-order quantification. In that section we also defend the claim that Turner's [1983a] proposal of using domain theory as an underlying mathematical framework should be more explored in natural language analysis. We believe that its precise computational approach to partiality of information offers a constructive framework where we can discuss problems of learnability and information growth as proposed by Popper.

3.2 THE SIMPLIFIED INTENSIONAL LANGUAGE

We introduce the language *L* which will replace *IL* in the intermediate representation for *M-NL*. Unlike *IL*, it does not have in its syntax the cup-cap operators. Syntactically, it is the language of a Modal and Temporal Logic, with constants of different types and the simply typed operations of lambda abstraction and application. We first introduce its alphabet and syntax. The basic or primitive domains required for its interpretation are the same as the ones for *IL*. We will then define the domains of denotations and the interpretation function. With this semantic function we define the notions of intension and extension as different forms of the same object, the interpretation. We cannot express the meaning postulates in this intermediate language. We observe that this is intuitively valid. We show that postulates are used by Montague as a parallel device to help in the definition of the interpretation, as constraint on the meaning of the constants of the language. Those constraints should be explicit in the definition of the constants.

3.2.1 Syntax of the Intensional Language L

Types

Definition 3.2.1.1 The set $\mathbb{T}$ of *types* of Intensional Logic L is defined as follows:

1. Basic categories e and t are types.
2. If α and β are types so is $(\alpha \rightarrow \beta)$.

Symbols

Definition 3.2.1.2 We will define eight different kind of symbols in the alphabet of the language:

1. Connectives: $\wedge$ $\vee$ $\sim$ $\leftrightarrow$ $\rightarrow$ $=$
2. Quantifiers: $\exists$ $\forall$
3. Modal operators: $\Diamond$ $\Box$
4. Temporal operators: $\boldsymbol{F}$ $\boldsymbol{P}$
5. Punctuation marks: . ( )
6. Abstractor: λ
7. Constants $\mathbb{C}_\alpha$ of type α: $C_{\alpha 1}$ $C_{\alpha 2}$ $C_{\alpha 3}$ ... $C_{\alpha i}$ ...
   The set of all constants C of L is $\mathbb{C} = \Sigma_\alpha \mathbb{C}_\alpha$.
8. Variables $\mathbb{V}_\alpha$ of type α: $h_{\alpha 1}$ $h_{\alpha 2}$ $h_{\alpha 3}$ ... $h_{\alpha i}$ ...
   The set of all variables h of L is $\mathbb{V} = \Sigma_\alpha \mathbb{V}_\alpha$.

Expressions

Definition 3.2.1.3 The set $\mathbb{E}_\alpha$ of well-formed expressions of type α in the language L is defined by the rules:

1. Every constant $C_{\alpha i}$ and every variable $h_{\alpha i}$ is in $\mathbb{E}_\alpha$.
2. If $e_{(\beta \rightarrow \alpha)}$ is of type $(\beta \rightarrow \alpha)$ and e_β is of type β, then $e_{(\beta \rightarrow \alpha)}(e_\beta)$ is in $\mathbb{E}_\alpha$.
3. If α can be analysed as $(\beta \rightarrow \rho)$ then $\lambda h_\beta . e_\rho$ is in $\mathbb{E}_\alpha$.
4. If α is t, e_t, e_{t1} and e_{t2} are any expressions of type t, $e_{\pi 1}$ and $e_{\pi 2}$ are any expressions of the same type π, then the following are in $\mathbb{E}_\alpha$:

$e_{\pi 1} = e_{\pi 2}$	$\sim e_t$
$\Diamond e_t$	$\Box e_t$
$F\, e_t$	$P\, e_t$
$\exists h_\alpha \cdot e_t$	$\forall h_\alpha \cdot e_t$
$e_{t1} \wedge e_{t2}$	$e_{t1} \vee e_{t2}$
$e_{t1} \rightarrow e_{t2}$	$e_{t1} \leftrightarrow e_{t2}$

The set of all expressions e of L is $\mathbb{E} = \Sigma_\alpha \mathbb{E}_\alpha$.

3.2.2 Semantics for Language *L*

Up to this point we have only defined syntactic objects, the well formed strings of symbols or expressions. To satisfy Frege's requirement, we have to guarantee that every expression we introduce will have secured a referent. If we are given the primitive referents for our constants, and if we accept the principle of compositionality, then every syntactic rule, as in definition 3.2.1.3, will define expressions whose referents can be constructed from the referents of its constituents. The definition of this construction has the form of a semantic rule. This corresponds to Frege's requirement that *A logically perfect language (Begriffsschrift) should satisfy the conditions, that every expression grammatically well constructed* [...] *out of signs already introduced shall in fact designate an object, and that no new sign shall be introduced* [...] *without being secured a reference* ([1892], pp. 70).

In Natural Language every phrase may be seen as referring to some object, e.g., an entity, unambiguously. However, as we have seen this is not enough for compositionality to hold in a theory of meaning. With different usages expressions can have different objects as referents. This enlarged notion of meaning is usually called the intension or sense of an expression. It can take into account the place, time and world, or situation, in which the expression is used. If we let the meaning or denotation of an expression became this sense, instead of a primitive object, we can have a compositional theory of meaning.

For our formalized language this tells us that each expression should denote a function from contexts of use to primitive objects, for example entities. So if $\mathscr{E}$ is the domain of primitive possible objects referred to by an expression in L of type e, then we should define its denotation as a function from possible contexts of use $\mathscr{I}$ to $\mathscr{E}$. Below we introduce the domains and functions required to construct the semantics for L.

Meanings

Definition 3.2.2.1 The domains $\mathscr{D}_\alpha$ of *possible denotations* of type α are constructed as follows,

1. If $\alpha = e$, $\mathscr{D}_\alpha = \mathscr{I} \rightarrow \mathscr{E}$
2. If $\alpha = t$, $\mathscr{D}_\alpha = \mathscr{I} \rightarrow \mathscr{B}$
3. For all α such that $\alpha=(\rho\rightarrow\beta)$, $\mathscr{D}_\alpha = \mathscr{D}_\rho \rightarrow \mathscr{D}_\beta$

and the domain of all possible *denotations* m is: $\mathscr{D} = \Sigma_\alpha \mathscr{D}_\alpha$.

Definition 3.2.2.2 Each domain of referred *objects*, $\mathscr{O}_\alpha$, is defined implicitly as a solution to the domain equation:

$$\mathscr{D}_\alpha = \mathscr{I} \rightarrow \mathscr{O}_\alpha \qquad \text{with} \quad \mathscr{O}_e = \mathscr{E} \text{ and } \mathscr{O}_t = \mathscr{B},$$

and the domain of all *objects* o is: $\mathscr{O} = \Sigma_\alpha \mathscr{O}_\alpha$.

If α is e or t, the domain $\mathscr{O}_\alpha$ is explicitly identified. But for complex types it is not so. For example $\mathscr{O}_{(e\rightarrow t)}$ is a solution to the equation,

$$\begin{aligned} \mathscr{D}_{(e\rightarrow t)} &= \mathscr{I} \rightarrow \mathscr{O}_{(e\rightarrow t)} \\ (\mathscr{I}\rightarrow\mathscr{E}) \rightarrow (\mathscr{I}\rightarrow\mathscr{B}) &= \mathscr{I} \rightarrow \mathscr{O}_{(e\rightarrow t)} \\ \text{e.g: } (\mathscr{I}\rightarrow\mathscr{E}) \rightarrow \mathscr{B} &\cong \mathscr{O}_{(e\rightarrow t)} \end{aligned}$$

Proposition 3.2.2.3 For each type α the expression $\mathscr{O}_\alpha \cong \mathscr{M}_{\tau(\alpha)}$ is a solution of the equation in definition 3.2.2.2, with $\mathscr{M}_\alpha$ defined as in 2.3.3.4, and where $\tau: \mathbb{T}_L \rightarrow \mathbb{T}_{IL}$ is defined as follows (compare with the function defined in 2.3.5.1):

1. $\tau(e) = e$
2. $\tau(t) = t$
3. $\tau(\alpha->\beta) = (s \rightarrow \tau(\alpha)) \rightarrow \tau(\beta)$ and

The proof of this proposition is similar to the proof of proposition 3.4.1, which will be presented and proved in section 3.4. We just need to observe that τ_L can be regarded as taking an expression describing a category of English and producing the same expression but this time being the expression which describes a type of intermediate language *L*. Then the translation function τ just defined is nothing but τ_{IL}. Then proposition 3.4.1 tells us that for any category *c* of English, or type *c* of *L*, $\mathcal{S}_{\tau_{IL}(c)} \cong \mathcal{D}_c$, or from the definition that $\mathcal{I} \rightarrow \mathcal{M}_{\tau_{IL}(c)} \cong \mathcal{D}_c$. So, from definition 3.2.2.2, we conclude that $\mathcal{O}_\alpha \cong \mathcal{M}_{\tau(\alpha)}$.

In *PTQ* Montague defines the object sets explicitly, because of his introduction in *IL* of the cap-cup operators which try to capture in the syntax of the intermediate language the notions of extension and intension. But then the translation function on categories of English will have to be defined accordingly to give the correct type in *IL* for an object denotation.

Definition 3.2.2.4 For each type α, g_α is a *variable-assignment* function defined over the domain $\mathbb{V}_\alpha$. The function g is the element of the domain $\mathcal{V}$ which sums all g_α[2],

$$g_\alpha : \mathcal{V}_\alpha = \mathbb{V}_\alpha \rightarrow \mathcal{D}_\alpha, \qquad g : \mathcal{V} = \mathbb{V} \rightarrow \mathcal{D}$$

Definition 3.2.2.5 For each type α, f_α is the given *constant-interpretation* function f defined over domain $\mathbb{C}_\alpha$. The function f is the element of the domain $\mathcal{C}$ which sums all f_α,

$$f_\alpha : \mathcal{C}_\alpha = \mathbb{C}_\alpha \rightarrow \mathcal{D}_\alpha, \qquad f : \mathcal{C} = \mathbb{C} \rightarrow \mathcal{D}$$

[2] See footnote 6 on chapter two.

Semantic Interpretation

Definition 3.2.2.6 The semantic *interpretation* function u_α, which gives the value in domain $\mathscr{D}_\alpha$ of an expression e_α of $\mathbb{E}_\alpha$, given g of $\mathscr{V}$,

$$u_\alpha : \mathbb{E}_\alpha \rightarrow [\mathscr{V} \rightarrow \mathscr{D}_\alpha]$$

is defined recursively for each type α, following the syntactic definition 3.2.1.3, of language L:

1. If $e_\alpha \in \mathbb{C}_\alpha$ then $u_\alpha[\![e_\alpha]\!]g = f_\alpha[\![e_\alpha]\!]$
   If $e_\alpha \in \mathbb{V}_\alpha$ then $u_\alpha[\![e_\alpha]\!]g = g_\alpha[\![e_\alpha]\!]$

2. If $e_{(\beta\rightarrow\alpha)}(e_\beta)$ is in $\mathbb{E}_\alpha$ then,
   $u_\alpha[\![e_{(\beta\rightarrow\alpha)}(e_\beta)]\!]g = u_{(\beta\rightarrow\alpha)}[\![e_{(\beta\rightarrow\alpha)}]\!]g\ (u_\beta[\![e_\beta]\!]g)$

3. If α is $(\beta\rightarrow\rho)$ and $\lambda v_\beta.\ e_\rho$ is in $\mathbb{E}_\alpha$ then,
   $u_\alpha[\![\lambda v_\beta.\ e_\rho]\!]g = \lambda d_\beta.\ u_\rho[\![e_\rho]\!]g_{[d \leftarrow v]}$

4. If α is t, e_t, e_{t1} and e_{t2} are in $\mathbb{E}_\alpha$, and $e_{\pi1}$, $e_{\pi2}$ are any expressions of the same type π then,

$u_\alpha[\![e_{\pi1} = e_{\pi2}]\!]g\ i$ = **cond** ($u_\pi[\![e_{\pi1}]\!]g\ i$ **equal** $u_\pi[\![e_{\pi2}]\!]g\ i$, t, f)

$u_\alpha[\![\sim e_t]\!]g\ i$ = **cond** ($u_\alpha[\![e_t]\!]g\ i$ **equival** f, t, f)

$u_\alpha[\![\Diamond e_t]\!]g\ i$ = **cond** (**exists** i'. $r(i, i')$ **and** $u_\alpha[\![e_t]\!]g\ i'$, t, f)

$u_\alpha[\![\Box e_t]\!]g\ i$ = **cond** (**forall** i'. $r(i, i')$ **implies** $u_\alpha[\![e_t]\!]g\ i$, t, f)

$u_\alpha[\![\mathbf{F}\ e_t]\!]g\ i$ = **cond** (**exists** i'. $l(i, i')$ **and** $u_\alpha[\![e_t]\!]g\ i$, t, f)

$u_\alpha[\![\mathbf{P}\ e_t]\!]g\ i$ = **cond** (**exists** i'. $l(i', i)$ **and** $u_\alpha[\![e_t]\!]g\ i$, t, f)

$u_\alpha[\![\exists\ v_\alpha.\ e_t]\!]g\ i$ = **cond** (**exists** d_α. $u_\alpha[\![e_t]\!]g_{[d \leftarrow v]}\ i$, t, f)

$u_\alpha[\![\forall v_\alpha.\ e_t]\!]g\ i$ = **cond** (**forall** d_α. $u_\alpha[\![e_t]\!]g_{[d \leftarrow v]}\ i$, t, f)

$u_\alpha[\![e_{t1} \wedge e_{t2}]\!]g\ i$ = **cond** ($u_\alpha[\![e_{t1}]\!]g\ i$ **and** $u_\alpha[\![e_{t2}]\!]g\ i$, t, f)

$u_\alpha[\![e_{t1} \vee e_{t2}]\!]g\ i$ = **cond** ($u_\alpha[\![e_{t1}]\!]g\ i$ **or** $u_\alpha[\![e_{t2}]\!]g\ i$, t, f)

$u_\alpha[\![e_{t1} \rightarrow e_{t2}]\!]g\ i$ = **cond** ($u_\alpha[\![e_{t1}]\!]g\ i$ **implies** $u_\alpha[\![e_{t2}]\!]g\ i$, t, f)

$u_\alpha[\![e_{t1} \leftrightarrow e_{t2}]\!]g\ i$ = **cond** ($u_\alpha[\![e_{t1}]\!]g\ i$ **equival** $u_\alpha[\![e_{t2}]\!]g\ i$, t, f)

The function u is the element which sums all u_α,

$$u: \mathbb{E} \rightarrow \mathscr{V} \rightarrow \mathscr{D}$$

or

$$u: \mathbb{E} \rightarrow \mathscr{V} \rightarrow \mathscr{I} \rightarrow \mathscr{O}$$

3.2.3 About the Semantic Function

One of the motivations to write the definition of the semantic function using the functions ***cond***, ***and***, ***or***, ***implies***, ***equival***, ***exists***, ***forall*** and ***equal*** was to stress that the only semantic functions we can 'program' are the computable ones. However, once this requirement is clear we can accept a more usual notation for writing the definitions, and so as an abbreviation we would write, for instance,

$$u[\![\Box\, e_t]\!]g\, i \qquad = \forall i'.\, r(i, i') \rightarrow u[\![e_t]\!]g\, i$$

Semantic Extension and Intension

Informally, the meaning of an expression given without its context of use is a function from all possible contexts to the possible meanings in use. Usually we refer to the first meaning as the intension of an expression, and to the second as the extension of an expression in a particular context. Asking for the extension of an expression without giving a context is the same as asking for its intension. As we have seen in chapter two, the semantic function u allows for an elegant formalization of the concepts of intension and extension.

Definition 3.2.3.1 The *intension* of an expression e, $int: \mathbb{E} \rightarrow \mathscr{D}$, is defined as:

$$int(e) = u[\![e]\!]g\,.$$

Definition 3.2.3.2 The *extension* of an expression e on context i, $ext: [\mathbb{E} \times \mathscr{I}] \rightarrow \mathscr{O}$, is defined as:

$$ext(e, i) = u[\![e]\!]g\, i\,.$$

Therefore the extension of e on i is the same as the intension of e evaluated at i

$$ext(e, i) = int(e)\, i$$

It follows from the definitions that the functions *int* and *ext* are related by the equation,

$$int = curry\,(ext)\,.$$

Meaning Postulates

As we have shown in section 2.3.4 on *Logical Equivalence and Meaning Postulates*, accepting the first meaning postulate of *PTQ* is equivalent to constrain the *constant-interpretation* function on constants of *IL* of type *e*, such as *j*, to be constant on indices. We have also claimed that the other postulates are equivalent to constraints on the *constant-interpretation* function for the other constants of *IL*. When we replace *IL* we are unable to express the restrictions directly in the new intermediate intensional language which translates *M-NL* English.

As we have seen in 2.4.2, we should think of the cap-cup operators in the following way. The cap operator constructs a constant function on indices, and the cup operator 'destroys' a function except in the case of applying it to a variable x, in which case it 'means' the variable assignment itself,

$u[\![\hat{}\,f]\!]\,g = \lambda i.u[\![f]\!]\,g$ and we can prove that $\forall i.\, u[\![\hat{}\,f]\!]\,g\,i = u[\![f]\!]\,g$.
$u[\![\check{}\,x]\!]\,g = g[\![x]\!]$ and we can prove that $\forall i.\, u[\![\check{}\,x]\!]\,g\,i = g[\![x]\!]\,i$.

It is clear to us, from what we have seen in 2.4.2, that the operators are used in the meaning postulates of *IL* to talk about functions, constant functions, or about elements of the semantic domains. In this new intensional language we cannot speak about constant functions or constant meanings, so we will need to express the postulates directly as our intuitions indicate, directly as a constraint on the *constant-interpretation.* We will be using the extensional language which allows us to speak about indices, the language of our semantic domains.

We say that this is closer to our intuitions because it corresponds to translating directly our English description (and Montague's description) of what is, let us say, a proper noun or a common noun. We claim that Montague's meaning postulates

can only be understood after mastering the use of the cap-cup operators to speak about interpretation functions. We will describe what we have just said using as examples meaning postulates 1, 2 and 3 in *PTQ*. For the others a similar, but rather lengthy proof can be given of what we have just said. We do not think however that meaning postulates are the correct way to go about restricting possible interpretations. We will come back to this in the last section, 3.5, of this chapter, where we explain our view and propose how to live without them.

Examples

Meaning Postulate 1

$\exists u_e.\, \Box[u_e = c_e]$

The Truth of 1 guarantees that proper nouns will be 'logically' determinate according to the interpretations under consideration, that is, will have extensions invariant with respect to possible worlds and moments of time.

Montague[1970d] (PTQ)

Our constraint corresponds to an almost direct translation of Montague's English sentence, saying that $f[\![c]\!]$ is a constant function on indices,

$$\exists\, u.\ \forall\, i.\, f[\![c]\!]\ i = u \qquad\qquad u\colon \mathcal{E}$$

Meaning Postulate 2

$\forall x_{s\to e}.\, \Box[horse'(x) \to \exists u_e.\, x = \hat{}u]$

In view of 2 'ordinary' common nouns (for example horse) will denote sets of *constant* individual concepts (for example the set of constant functions on worlds and moments of time having horses as their values; from an intuitive viewpoint, this is no different from the set of horses). It would be unacceptable to impose this condition on such 'extraordinary' common nouns as price and temperature.

Montague[1970d] (PTQ)

Our constraint says that if $f[\![c]\!]\ x\ i$ is true on indices i, for individual concepts x then those individual concepts are constant functions on indices j,

$$\forall x.\ \forall i.\, [\, f[\![horse']\!]\ x\, i \to \exists u.\, \forall j.\ x\, j = u\,] \qquad\qquad x\colon \mathcal{I}\to\mathcal{E}$$

Meaning Postulate 3

$$\exists m_{s\to e\to t}.\, \forall x_{s\to e}.\, \Box[walk'(x) \leftrightarrow \check{}m(\check{}x)]$$

The truth of 3 is the natural requirement of *extensionality* for intransitive verbs.

Montague[1970d] (PTQ)

We would say that,

$$\exists m.\, \forall x.\, \forall i.\, [\, f[\![walk']\!]\ x\, i \leftrightarrow m\, i\, (x\, i)\,] \qquad m\colon \mathcal{I}\to\mathcal{E}\to\mathcal{B}$$

The other meaning postulates correspond basically to express the constraints on f using the cap-cup operators, with box meaning truth in all indices. Once we understand how the operators are used, basically, to speak about functions and in particular constant functions, the constraints on the interpretations (on the *constant-interpretation*) can be re-phrased using *IL* and meaning postulates, but not before that. It is also clearer that we would need a lot more meaning postulates to constraint f for a 'possible interpretation'. We would need to restrict the interpretations not only to impose 'equality' or synonymy but also 'inequality' or antonymy. We would need for instance to guarantee that the interpretations of ***fish*** is not the same as of ***woman***, and of ***man*** is not as of ***horse***.

Surviving without Meaning Postulates

Some of the meaning postulates can be avoided if, as Dowty[1979] points out, we use a direct complex translation from *M-NL* into *IL* (pp. 197). However we do not think that to eliminate some meaning postulates will be enough. We could ask ourselves what happens if we decide to change the use of ***necessarily*** from true in any world to true in accessible worlds. Should we assert as meaning postulates the same expressions Montague used, or maintain the restrictions on the *constant-interpretation* function? There is also an interesting observation to be made that there is no sentence of Montague's English which translates into any meaning postulate. If English is a formal language, and if Montague Grammar is to be a general theory of language we cannot accept meaning postulates which cannot be stated in English. Otherwise the program is also telling us that there are truths which cannot be expressed in any language.

The last remark forces us to reject the possibility of using meaning postulates to constrain possible interpretations, as it leads into a contradiction in the objectives of Montague Grammar. The way out of the problem accepts both compositionality and the formal reduction of semantics. And we still believe, as Montague did, that syntax should just be a prelude to semantics. The main difference we see, for instance in a formal characterization of semantics, is that a definition which tries to model our understanding of meaning should take into account all the information about meaning that is known, and that such information must be made explicit in the semantic definitions. We hope to make this last sentence more clear in section 3.5 of this chapter.

Intuitively, we want to say that if we have learnt that there is a difference between say 'extraordinary' and 'ordinary' common nouns or between intensional and extensional verbs, that classification not only can but must be explicit in my syntactic or semantic rules. Once the distinction is made, the semantic interpretation can be directly constructed to take account of the difference in use. Once again the main distinction we see between syntax and semantics is that the former is not context of use sensitive while the latter is, however this being an unprecise distinction.

3.3 THE SIMPLIFIED TRANSLATION

The translation function from *M-NL* into language *L* is very similar to the one in *PTQ*. The translation is between syntactic domains preserving the type structure, i.e. objects of the some category are translated into objects of the same type. We remark that language *L* has no type constructor s, and no cap-cup operators. So our definition of the translation on categories is much simplified. The translation on phrases of *M-NL* deals basically with providing constants of *L* as translations for the lexicon (except for pronouns), of translating determiners using expressions with first-order quantifiers, and respecting the type structure with functional application.

After the definition we give some examples comparing how a phrase of *M-NL* is translated into *IL* and how it is translated into *L*.

Translation on Syntactic Categories

Definition 3.3.1 The function from syntactic categories of *M-NL* English to types of language *L*, τ_L: $CAT \rightarrow \mathbb{T}_L$ is defined as follows:

1. $\tau(e) = e$
2. $\tau(t) = t$
3. $\tau(a\text{->}b) = \tau(a) \rightarrow \tau(b)$ and
   $\tau(a\text{=>}b) = \tau(a) \rightarrow \tau(b)$, for all categories a and b.

Translation on Phrases

Definition 3.3.2 The following translation rules from the domain of well-formed phrases P of *M-NL* to the domain of well-formed expressions $\mathbb{E}$ of language *L*, define function t_L,

$$t_L: P \rightarrow \mathbb{E}.$$

We will be using the following names in this definition,

j	is C_{e1},	x	is h_{e1},	p	is h_{t1},
m	is C_{e2},	y	is h_{e3},		
b	is C_{e3},			P	is $h_{(e\rightarrow t)1}$,
n	is C_{e4},	x_n	is h_{e2n},	Z	is $h_{((e\rightarrow t)\rightarrow t)1}$.

Basic Rules:

1.1. For all n:

$$t\{he_n\} = \lambda P.\, P(x_n) \qquad = he_{Ln} \qquad = h_{\tau(T)n} \quad \in \mathbb{V}_{\tau(T)}.$$

1.2. For every other *basic expression* b_a of *M-NL*, $t\{b_a\}$ is a given constant $C_{\tau(a)}$ of *L*. For simplification of use, if ***bexp*** is the identifier used as b_a, then *bexp* is to be the identifier used for the translation in *L*, $t\{bexp\}$=*bexp*; e.g., the following constants have the form shown:

$$t\{be\} = \lambda Z.\ \lambda x.\ Z(\lambda y.\ x=y) \quad = be_L \quad = C_{\tau(TV)\,6} \quad \in \mathbb{C}_{\tau(TV)},$$
$$t\{necessarily\} = \lambda p.\ \Box p \quad = necessarily_L \quad = C_{\tau(SA)\,1} \quad \in \mathbb{C}_{\tau(SA)},$$
$$t\{John\} = \lambda P.\ P(j) \quad = John_L \quad = C_{\tau(T)\,1} \quad \in \mathbb{C}_{\tau(T)},$$
$$t\{Mary\} = \lambda P.\ P(m) \quad = Mary_L \quad = C_{\tau(T)\,2} \quad \in \mathbb{C}_{\tau(T)},$$
$$t\{Bill\} = \lambda P.\ P(b) \quad = Bill_L \quad = C_{\tau(T)\,3} \quad \in \mathbb{C}_{\tau(T)},$$
$$t\{ninety\} = \lambda P.\ P(n) \quad = ninety_L \quad = C_{\tau(T)\,4} \quad \in \mathbb{C}_{\tau(T)}.$$

2. If $\alpha \in P_{CN}$ then,

$$t\{every\ \alpha\} = \lambda P.\ \forall x.\ (\ t\{\alpha\}(x) \rightarrow P(x)\ ),$$
$$t\{the\ \alpha\} = \lambda P.\ \exists y.\ (\forall x.\ (\ t\{\alpha\}(x) \leftrightarrow x=y\ ) \wedge P(y)\ ),$$
$$t\{F_2(\alpha)\} = \lambda P.\ \exists x.\ (\ t\{\alpha\}(x) \wedge P(x)\ ).$$

3. If $\alpha \in P_{CN}$ and $\beta \in P_t$ then,

$$t\{F_{3,n}(\alpha,\beta)\} = \lambda x_n.\ (\ t\{\alpha\}(x_n) \wedge t\{\beta\}\ ).$$

Rules of functional application:

4. If $\alpha \in P_T$ and $\beta \in P_{IV}$ then,

$$t\{F_4(\alpha,\beta)\} = t\{\alpha\}(t\{\beta\}).$$

5. If $\alpha \in P_{TV}$ and $\beta \in P_T$ then,

$$t\{F_5(\alpha,\beta)\} = t\{\alpha\}(t\{\beta\}).$$

6. If $\alpha \in P_{IAVP}$ and $\beta \in P_T$ then,

$$t\{F_4(\alpha,\beta)\} = t\{\alpha\}(t\{\beta\}).$$

7. If $\alpha \in P_{SV}$ and $\beta \in P_t$ then,

$$t\{F_6(\alpha,\beta)\} = t\{\alpha\}(t\{\beta\}).$$

8. If $\alpha \in P_{TIV}$ and $\beta \in P_{IV}$ then,

$$t\{F_6(\alpha,\beta)\} = t\{\alpha\}(t\{\beta\}).$$

9. If $\alpha \in P_{SN}$ and $\beta \in P_t$ then,

$$t\{F_6(\alpha,\beta)\} = t\{\alpha\}(t\{\beta\}).$$

10. If $\alpha \in P_{IAV}$ and $\beta \in P_{IV}$ then,

$$t\{F_7(\alpha,\beta)\} = t\{\alpha\}(t\{\beta\}).$$

Rules of conjunction and disjunction:

11. If $\phi, \psi \in P_t$ then,

$$t\{\phi \textit{ and } \psi\} = t\{\phi\} \wedge t\{\psi\} ,$$
$$t\{\phi \textit{ or } \psi\} = t\{\phi\} \vee t\{\psi\} .$$

12. If $\phi, \psi \in P_{IV}$ then,
$$t\{\phi \textit{ and } \psi\} = \lambda x.\ (t\{\phi\}(x) \wedge t\{\psi\}(x)) ,$$
$$t\{\phi \textit{ or } \psi\} = \lambda x.\ (t\{\phi\}(x) \vee t\{\psi\}(x)) .$$
13. If $\phi, \psi \in P_T$ then,
$$t\{\phi \textit{ or } \psi\} = \lambda P.\ (t\{\phi\}(P) \vee t\{\psi\}(P)) .$$

Rules of quantification:

14. If $\alpha \in P_T$ and $\phi \in P_t$ then,
$$t\{F_{10,n}(\alpha,\phi)\} = t\{\alpha\}(\lambda x_n.\ t\{\phi\}) .$$
15. If $\alpha \in P_T$ and $\zeta \in P_{CN}$ then,
$$t\{F_{10,n}(\alpha,\zeta)\} = \lambda y.\ t\{\alpha\}(\lambda x_n.\ t\{\zeta\}(y)) .$$
16. If $\alpha \in P_T$ and $\delta \in P_{IV}$ then,
$$t\{F_{10,n}(\alpha,\delta)\} = \lambda y.\ t\{\alpha\}(\lambda x_n.\ t\{\delta\}(y)) .$$

Rules of tense and sign:

17. If $\alpha \in P_T$ and $\delta \in P_{IV}$ then,
$$t\{F_{11}(\alpha,\delta)\} = \sim(t\{\alpha\}(t\{\delta\})) ,$$
$$t\{F_{12}(\alpha,\delta)\} = F(t\{\alpha\}(t\{\delta\})) ,$$
$$t\{F_{13}(\alpha,\delta)\} = \sim F(t\{\alpha\}(t\{\delta\})) ,$$
$$t\{F_{14}(\alpha,\delta)\} = P(t\{\alpha\}(t\{\delta\})) ,$$
$$t\{F_{15}(\alpha,\delta)\} = \sim P(t\{\alpha\}(t\{\delta\})) .$$

Informally, the difference between the translation t_L, that we define here, and the translation function t_{IL}, in chapter two, results from two syntactic operations in the presentation. All cap-cup operators are deleted, and all variables suffer the type change that results from not having in the language the type constructor s which corresponds to indices.

Example 3.3.3 Translation of the *M-NL* expression of category sentence, ***John walks***, into *IL* and *L*:

$t_{IL}\{John\} = John_{IL} = \lambda P.\ \check{}P(\hat{}j)$, $\quad P \in \mathbb{V}_{IL\,(s\to((s\to e)\to t))}$ and $j \in \mathbb{C}_{IL\,e}$

$t_{IL}\{walk\} = walk_{IL}$, $\quad walk_{IL} \in \mathbb{C}_{IL\,((s\to e)\to t)}$

$t_{IL}\{John\ walks\}$

$t\{John\}(\hat{}t\{walks\}) = John_{IL}(\hat{}walk_{IL})$

$\lambda P.\ \check{}P(\hat{}j)(\hat{}walk_{IL})$

$\check{}\hat{}walk_{IL}(\hat{}j)$

$walk_{IL}(\hat{}j)$.

$t_L\{John\} = John_L = \lambda P.\ P(j)$, $\quad P \in \mathbb{V}_{L\,(e\to t)}$ and $j \in \mathbb{C}_{L\,e}$

$t_L\{walk\} = walk_L$, $\quad walk_L \in \mathbb{C}_{L\,(e\to t)}$

$t_L\{John\ walks\}$

$t\{John\}(t\{walks\}) = John_L(walk_L)$

$\lambda P.\ P(j)(walk_L)$

$walk_L(j)$.

Example 3.3.4 We can observe how the induced interpretation reduction process works out for a simple expression of category *Term* such as ***John***, with *IL* and *L*:

$t_{IL}\{John\} = \lambda P.\ \check{}P(\hat{}j)$, $\quad P \in \mathbb{V}_{IL\,\alpha}$, $\alpha = s\to((s\to e)\to t)$

$u[\![t\{John\}]\!]g\,i$

$u[\![\lambda P.\ \check{}P(\hat{}j)]\!]g\,i$

$\lambda m.\ u[\![\check{}P(\hat{}j)]\!]g_{[m\leftarrow P]}\,i$

$\lambda m.\ u[\![\check{}P]\!]g_{[m\leftarrow P]}\,i\,(u[\![\hat{}j]\!]g_{[m\leftarrow P]}\,i)$

$\lambda m.\ u[\![P]\!]g_{[m\leftarrow P]}\,i\,i(\lambda i.\ u[\![j]\!]g_{[m\leftarrow P]}\,i)$

$\lambda m.\ g_{[m\leftarrow P]}[\![P]\!]\,i\,(\lambda i.\ f[\![j]\!]\,i)$

$\lambda m.\ m\,i(f[\![j]\!])$, $\quad m \in \mathcal{M}_\alpha = \mathcal{I}\to((\mathcal{I}\to\mathcal{E})\to\mathcal{B})$

$t_L\{John\} = \lambda P.\ P(j)$, $\quad P \in \mathbb{V}_{L\,\beta}$, $\beta = e\to t$

$u[\![t\{John\}]\!]g\,i$

$u[\![\lambda P.\ P(j)]\!]g\,i$

$\lambda d.\ u[\![P(j)]\!]g_{[d\leftarrow P]}\,i$

$$\lambda d.\, u[\![P]\!]g_{[d \leftarrow P]}\,(u[\![j]\!]g_{[d \leftarrow P]})\,i$$
$$\lambda d.\, g_{[d \leftarrow P]}\,[\![P]\!]\,(f[\![j]\!])\,i$$
$$\lambda d.\, d(f[\![j]\!])\;i, \qquad\qquad d \in \mathcal{D}_\beta = (\mathcal{I} \rightarrow \mathcal{B}) \rightarrow (\mathcal{I} \rightarrow \mathcal{B})$$

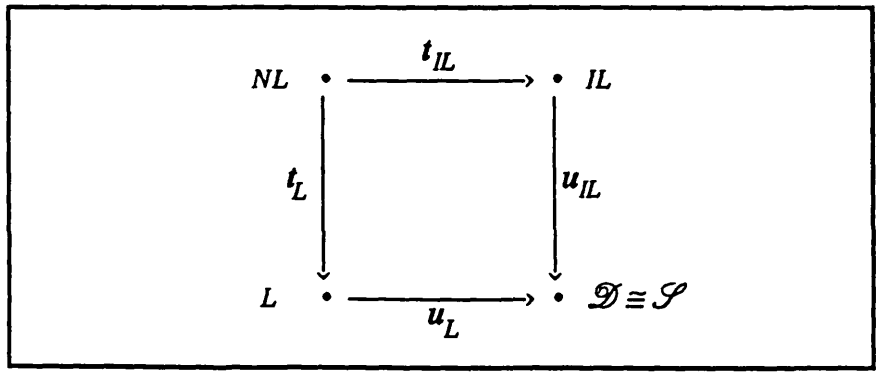
Is the Intensional Translation of English High-Order ?

There is an interesting observation to be made at this stage. With this function English phrases are translated using only first order quantifiers on variables of type entity. Although the intensional language *L* in principle allows expressions with quantification over variables of any type, the translations of *M-NL* phrases do not use such expressions. The same thing happens in *PTQ* with the translations. But there, as we have seen, meaning postulates use that 'facility' of expression.

3.4 ISOMORPHIC SEMANTICS FOR ENGLISH

In the presentation of chapter two an expression of *M-NL* is translated into an expression of *IL* which is then interpreted with a defined semantic function. The interpretation of an expression of any syntactic category is characterized by its intension or by its extension in a given context. In this section we show that the domains of denotations of *IL*-translations or *L*-translations, of phrases in the same category, are isomorphic. Moreover, if our constant and variable interpretation functions are correctly constructed there is an isomorphism between the interpretation of any phrase, using the intensional languages *IL* or *L*.

$$\mathcal{D} \cong \mathcal{I}$$
$$u_L[\![\, t_L\{\, exp\,\}]\!] \cong u_{IL}[\![\, t_{IL}\{\, exp\,\}]\!]$$



Proposition 3.4.1 For any category c of IL, the domain defined in 2.3.3.5 of intensions or senses $\mathcal{S}_{\tau_{IL}(c)}$ of expressions of type c, is isomorphic to the domain defined in 3.2.2.1 of denotations $\mathcal{D}_{\tau_L(c)}$ of expressions of the same type c:

$$\mathcal{S}_{\tau_{IL}(c)} \cong \mathcal{D}_{\tau_L(c)},$$

where c: CAT as defined in 2.3.1.1, $\tau_{IL}(c)$: $\mathbb{T}_{IL}$ as defined in 2.3.5.1 and $\tau_L(c)$: $\mathbb{T}_L$ as defined in 3.2.1. The proof is based on induction over the translation on syntactical categories.

We will reproduce here a summary of some definitions from the previous chapter which are involved in the proof.

(Definition 2.3.5.1) The function from syntactic categories of English to types of language IL, τ_{IL}: $CAT \rightarrow \mathbb{T}_{IL}$ is defined as follows:

1. $\tau(e) = e$
2. $\tau(t) = t$
3. $\tau(a\text{->}b) = (s \rightarrow \tau(a)) \rightarrow \tau(b)$ and
   $\tau(a\text{=>}b) = (s \rightarrow \tau(a)) \rightarrow \tau(b)$, for all categories a and b.

(Definition 2.3.3.4) The domains $\mathcal{M}_\alpha$ of *possible denotations* of type α are constructed as:

1. If $\alpha = e$, $\mathcal{M}_\alpha = \mathcal{E}$
2. If $\alpha = t$, $\mathcal{M}_\alpha = \mathcal{B}$
3. For all α such that $\alpha=(\rho\rightarrow\beta)$, $\mathcal{M}_\alpha = \mathcal{M}_\rho \rightarrow \mathcal{M}_\beta$
4. For all α such that $\alpha=(s\rightarrow\beta)$, $\mathcal{M}_\alpha = \mathcal{I} \rightarrow \mathcal{M}_\beta$

(Definition 2.3.3.5) The domains $\mathcal{S}_\alpha$ of *senses* of type α are defined as:

$$\mathcal{S}_\alpha = \mathcal{I} \rightarrow \mathcal{M}_\alpha$$

(Definition 2.3.4.1) The semantic *interpretation* function $u_{IL\,\alpha}$, which gives the value in domain $\mathcal{S}_\alpha$ of an expression $e_{IL\alpha}$ of $\mathbb{E}_{IL\alpha}$, given g of $\mathcal{V}_{IL}$,

$$u_{IL\alpha}: \mathbb{E}_{IL\alpha} \rightarrow \mathscr{V}_{IL} \rightarrow \mathscr{S}_{\alpha}$$

is defined recursively for each type α of language *IL* [...].

Proposition 3.4.2 According to definitions 2.3.5.1 and 3.3.1, the expressions of *M-NL* of category *c* will be translated into the following type of expressions in the intermediate languages *IL* and *L*:

1. if $c = e$,	$e_{IL\,\tau(c)}$: $\mathbb{E}_{IL\,e}$	$e_{L\,\tau(c)}$: $\mathbb{E}_{L\,e}$
2. if $c = t$,	$e_{IL\,\tau(c)}$: $\mathbb{E}_{IL\,t}$	$e_{L\,\tau(c)}$: $\mathbb{E}_{L\,t}$
3. if $c = (a\text{->}b)$ or $c = (a\text{=>}b)$,	$e_{IL\,\tau(c)}$: $\mathbb{E}_{IL\,(s\rightarrow\tau(a))\rightarrow\tau(b)}$	$e_{L\,\tau(c)}$: $\mathbb{E}_{L\,\tau(a)\rightarrow\tau(b)}$

where $\mathbb{E}_{IL\,\alpha}$ is the domain of well-formed expressions of type α in *IL*, and $\mathbb{E}_{L\,\alpha}$ is the domain of well-formed expressions of *L*. The proposition follows from the definitions.

In order to prove that $\mathscr{S}$ and $\mathscr{D}$ are isomorphic, we show that for an arbitrary well-formed expression exp_c of category *c* in *M-NL*, s and d belong to isomorphic domains, where:

$s = u_{IL}[\![t_{IL}\{exp_c\}]\!]g_{IL}$ and s is in $\mathscr{S}_{\tau(c)} = \mathscr{I}\rightarrow\mathscr{M}_{\tau(c)}$

$d = u_L[\![t_L\{exp_c\}]\!]g_L$ and d is in $\mathscr{D}_{\tau(c)}$

We have two main cases:

Case 1: c is a basic category either *e* or *t*. In these cases it follows from the definitions 2.3.3.4, 2.3.3.5 and 3.2.2.1 that $\mathscr{S}_{\tau(c)} = \mathscr{I}\rightarrow\mathscr{M}_{\tau(c)} = \mathscr{D}_{\tau(c)}$. This is the basic case.

Case 2: c is not a basic category and has either of the forms *(a ->b)* or *(a =>b)*. This is the induction step.

$$\begin{aligned}\mathscr{S}_{\tau(c)} &= \mathscr{I}\rightarrow((\mathscr{I}\rightarrow\mathscr{M}_{\tau(a)})\rightarrow\mathscr{M}_{\tau(b)})\\ &\cong(\mathscr{I}\times(\mathscr{I}\rightarrow\mathscr{M}_{\tau(a)}))\rightarrow\mathscr{M}_{\tau(b)})\\ &\cong((\mathscr{I}\rightarrow\mathscr{M}_{\tau(a)})\times\mathscr{I})\rightarrow\mathscr{M}_{\tau(b)})\end{aligned}$$

$$\cong (\mathcal{I} \rightarrow \mathcal{M}_{\tau(a)}) \rightarrow (\mathcal{I} \rightarrow \mathcal{M}_{\tau(b)})$$
$$= \mathcal{S}_{\tau(a)} \rightarrow \mathcal{S}_{\tau(b)}$$
$$\mathcal{D}_{\tau(c)} = \mathcal{D}_{\tau(a)} \rightarrow \mathcal{D}_{\tau(b)} .$$

If a and b are basic categories e or t, given case 1 we conclude that $\mathcal{S}_{\tau(c)}$ and $\mathcal{D}_{\tau(c)}$ are isomorphic domains. If a and b are still not basic categories we use again the induction step. Therefore:

$$\mathcal{S}_{\tau_{IL}(c)} \cong \mathcal{D}_{\tau_L(c)} .$$

Proposition 3.4.3 If δ is the sense of the induced interpretation of an expression exp_c of category c, after translation t_{IL} into IL under interpretation u and variable-assignment g,

$$\delta = u_{IL}[\![t_{IL}\{exp_c\}]\!]g_{IL}$$

and if α is the denotation of the translation t_L of the same exp_c into L,

$$\alpha = u_L[\![t_L\{exp_c\}]\!]g_L$$

then for any exp_c, δ is isomorphic to α,

$$\delta \cong \alpha ,$$

and therefore,

$$u_{IL}[\![t_{IL}\{exp_c\}]\!]g_{IL}i \cong u_L[\![t_L\{exp_c\}]\!]g_L i .$$

The following definitions, 3.4.4 to 3.4.6, can be used to support the proof of the previous proposition, and would be used in its formal proof, which is outlined latter on. If, as we have proved, the domains of denotations are isomorphic, we know that there is a one to one correspondence between elements. The functions below are used to make explicit that isomorphism.

Definition 3.4.4 Any variable $h_{\pi n}$ and constant $C_{\pi n}$ of IL is replaced in L by a variable $h_{\rho n}$ and constant $C_{\rho n}$ such that, $\rho=k(\pi)$, where,

$$k: \mathbb{T}_{IL} \rightarrow \mathbb{T}_{L}$$

1. $k(e) = e$
2. $k(t) = t$
3. for all types $s \rightarrow \alpha$, $k(s \rightarrow \alpha) = k(\alpha)$
4. for all types $\alpha \rightarrow \beta$, and α not s, $k(\alpha \rightarrow \beta) = k(\alpha) \rightarrow k(\beta)$.

e.g. $k(s \rightarrow ((s \rightarrow e) \rightarrow t)) = k((s \rightarrow e) \rightarrow t))$

$= k(s \rightarrow e) \rightarrow k(t)$

$= k(e) \rightarrow k(t)$

$= e \rightarrow t$

Definition 3.4.5 The variable-assignment g_L can be defined from g_{IL} in 2.3.3.6,

$$g_L[\![h_{k(\pi)n}]\!] = g_{IL}[\![h_{\pi n}]\!]$$

e.g. $g_L[\![P_L]\!] = g_{IL}[\![P_{IL}]\!]$, where P_L is $h_{(e \rightarrow t)1}$ and P_{IL} is $h_{s \rightarrow ((s \rightarrow e) \rightarrow t)1}$.

Definition 3.4.6 The function f_L can be defined from f_{IL} in 2.3.3.7,

$$f_L[\![C_{k(\pi)n}]\!] = f_{IL}[\![C_{\pi n}]\!]$$

e.g. $f_L[\![j_L]\!] = f_{IL}[\![j_{IL}]\!]$, where j_L is $C_{L_{e1}}$ and j_{IL} is $C_{IL_{e1}}$.

To show that δ and α are different forms of the same object we will do as follows. We have constructed translation t_L to be like t_{IL}, except that all intensional operators are deleted and all variables and constants are replaced as was just defined. With this observation, and the way in which f and g have been defined, proposition 3.4.3 is, we believe, quite obvious. In order to be more formal we should give a proof using mathematical induction on categories for every translation and interpretation rule. Given the number of such rules the proof is very long. We would conclude therefore that,

$$u_{IL}[\![t_{IL}\{exp_c\}]\!]g_{IL}i \equiv u_L[\![t_L\{exp_c\}]\!]g_L i.$$

Example 3.4.7 The induced interpretation of expression ***John*** of *M-NL* in example 3.3.4 via *IL* or *L* is the same. If we have $f_L[\![j_L]\!] = f_{IL}[\![j_{IL}]\!] = c_e$: $\mathcal{I} \rightarrow \mathcal{E}$,

$$u_{IL}[\![t_{IL}\{John\}]\!]g_{IL}\, i = \lambda m.\, m\, i(f_{IL}[\![j_{IL}]\!]) = \lambda m.\, m\, i\, c_e$$

$$u_L[\![t_L\{John\}]\!]g_L\, i = \lambda d.\, d(f_L[\![j_L]\!])\, i = \lambda d.\, d\, c_e\, i$$

but m is in $\mathcal{M}_\alpha = \mathcal{I} \rightarrow ((\mathcal{I} \rightarrow \mathcal{E}) \rightarrow \mathcal{B})$

and d is in $\mathcal{D}_\beta = (\mathcal{I} \rightarrow \mathcal{E}) \rightarrow (\mathcal{I} \rightarrow \mathcal{B})$, so,

$$\lambda m.\, m\, i\, c_e \equiv \lambda d.\, d\, c_e\, i$$

3.5 COMPUTABLE GRAMMARS WITHOUT MEANING POSTULATES

After having shown how we can have a simplified treatment of *PTQ*, defined in a compositional and computational manner, we would also like to proceed in the direction of a even more simple and more flexible treatment. By flexible we mean that alterations should be incorporated in the treatment with minimal changes. Intuitively we should just need to add new definitions to our grammar in order to extend the 'amount' of natural language under formal treatment.

The main objective and importance of this section is to show how to treat a fragment of intensional English without using meaning postulates at all, as we have been proposing, and therefore to provide interpretations for Montague's intensional English using only first order quantification. This can be seen as another simplification to Montague's system, and also to the system proposed in the previous sections. Having the characterization of words defined in different parts of the system is a violation of basic principles. Principles of economy and logical engineering would require a uniquely localized definition.

The second objective is to illustrate some basic characteristics of a general method for constructing grammars, which we believe corresponds to the flexible human learning process. Because of its flexibility, this method has merits on its own,

which as we will propose in chapter five should be further investigated. However it also justifies the elimination of Montague's meaning postulates, as it introduces the definition of a word's meaning in just one place. We could have eliminated the postulates from Montague's system directly, but we believe that such procedure although justifiable from basic principles, would still need a cognitive explanation. We believe that the method we exemplify, being cognitively plausible justifies not only the elimination of the postulates but also indicates the correct place for a formal definition of the meaning of words.

In order to achieve these objectives we will define a computable grammar for a subset of extensional English and then extend it to a grammar for intensional English. Unlike PTQ, we will avoid the use of an intermediate language and translation by defining a direct semantic interpretation. Although the notation for the semantic system follows closely the one used throughout this thesis, we hope the reader will forgive us for simplifying the presentation of the syntactic system in the lines already mentioned in section 3.2.1 on *Example from LR-Grammars* and following Pitt[1989]. We believe that this new simplified notation is amply justified as for instance we can present a syntactic definition, as is 2.3.1.4, in a much more concentrated form. And a direct interpretation eliminates a translation definition like 2.3.5.2. As we will define two versions of English, we believe the simplification and compactation is essential. We have not used the notation for syntax throughout for the simple reason that this thesis is more concerned with the semantic system and assumes that the reader is familiar with Montague Grammar.

Overview

We start by discussing the principles on which Montague's framework is based, and the properties required in the syntax and semantics of a computable theory of grammar. We then introduce a very simplified version of English, which we will call approximation zero or ^{0}*English*, with an extensional semantics. Then we will introduce an improved version of English, approximation one or ^{1}*English*, with an intensional semantics. We show and we claim without proof that in this version of English, Montague's Meaning Postulates are unnecessary mainly due to our simple

and uniquely localized definition for the meaning of the words in the lexicon. This definition satisfies the natural requirements motivating Montague's Meaning Postulates. Therefore we will take a further step to conclude that quantification in our semantic language should only be first-order.

3.5.1 Some Basic Characteristics of Montague's Framework

Compositionality, Logic and Postulates

Two of the principles underlying Montague's treatment of natural language are the principle of compositionality and the principle that formal logic can be used to provide the characterizations of meaning. Although they are used together, it is very important to recognize that they are different principles. Different combinations of those principles can be accepted or rejected with, if nothing else, basic pragmatic reasons. For instance, some theories of understanding and characterization of natural language may accept some form of compositionality but would reject formal logic, for instance connectionist theories. Some schools would only accept some systems of formal logic, for instance first order logic in PROLOG logic programming. Some schools seem to reject explicit compositionality while still accepting different forms of formal logic. It is common understanding that Montague accepted both compositionality and high-order logic. In general, we believe that the use of high-order logics in the treatment of natural language is not intuitively correct as pointed out in chapter one.

In *PTQ* Montague translates English into a subset of Intensional Logic expressions. He then defines the compositional meaning of that logic, given an interpretation. In fact this compositional definition is the definition of that interpretation. As we pointed out, it seems to be less correct to quantify on the interpretation as a whole in the definition of logical equivalence (see definition 2.3.4.5, and section 3.2.3 on *Meaning Postulates*). We can however quantify on the free variables in the definition of that interpretation. In *PTQ* those free variables are for instance the meanings of words given by the constant interpretation function on constants of the logic. As we have shown before for some meaning postulates, they can be seen to

define, as constraints, the primitive meanings of words. Therefore from a methodological stand they should not be introduced at a later stage, but rather where words are defined. Then, if there are no free variables left in the definition of the interpretation, we cannot quantify it. As we will see in chapter four, we believe Situation Semantics has taken this correct approach in the definition of strong equivalence.

Limited Changeability

One of the major problems with Montague semantics has to do with changeability. Given its structure of types, whenever we want to account for new phenomena in natural language, it is most likely that most or all of the previous assigned types to expressions will have to be changed, as Gabbay[1988] pointed out. One possible semantic solution which we can think of as an evolution still in Montague's framework was proposed by Partee[1988]. Partee proposes mechanisms for type shifting and discusses some of the problems involved in finding natural solutions for type shifting, which could be used to characterize formally the flexibility and versatility of natural language. In our example of constructing intensional English we use a mechanism similar to Partee's type-shifting.

In general, we would like to have a method of defining a formal grammar for natural language, its syntax and semantics, which could be improved just by adding new definitions to the current ones. We believe that it is possible to do this in a framework which is still related to Montague's. As our second objective, this general method is illustrated by improving the definition of extensional English to arrive at intensional English.

3.5.2 Properties Required for a Computable Grammar

In this section we indicate some of the properties a computable syntax and semantics for natural language should have. As we have said we divide grammar

into syntax and semantics, and we would require both to be represented as programs which terminate, or as effectivelly computable functions [3].

Computable Syntax

According to Montague[1970c], the importance of syntax is due to its role as a preliminary to semantics. If we accept the proposed division between syntax and semantics coming from the difference between the study of the forms and of the uses of languages, syntax studies whatever is not dependent on contexts of uses. We would say that for the context of use in which our formalization is being built, syntactic definitions deal with context insensitive objects, 'forms' (see chpater one). This is not a discussion on whether syntax is context-free or not, as in Chomsky's hierarchy of grammars.

Computer Science has developed many different formalisms to define formal languages. The construction of computable natural language approximations should use the most interesting of those formalisms. For instance, we can take advantage of the expressive power of attribute-value grammars and the existence of efficient parser generators for such formalisms. Moreover by restricting the syntax to the subset of context-free grammars known as *LR*, we can guarantee parsing strings in linear time and memory space, i.e. proportional only to the length of the string (for unambiguous grammars). The use of such restriction also guarantees termination, which, as we have seen, should be a property proved for implementations of grammars. Attribute-value grammar formalisms, parser generation and computational efficiency are discussed in Pitt[1989] [4].

---

[3] This identification of grammar with computable functions for syntax and semantics explains our name 'computable grammar'.

[4] As we have seen our definition for syntax only allows it to be context insensitive. If we identify the expressions 'context insensitive' and 'context-free', we seem to be saying that natural language syntactic descriptions should be context-free. This is not the appropriate place for such discussion, however we just notice its admissibility due to the fact that the valid strings of any language can be described using an over generating context-free grammar. Cunha and Pitt[1989] take this discussion further.

Computable Semantics

From basic principles we will accept that semantic functions which characterize the use of natural language should be restricted to computable ones. As we are using denotational semantics we will require that the functions used in the definitions should be provably monotonic and continuous. We could argue from basic principles that any real interpretation function should be proved to terminate on any argument (see chapter one, section 1.3.3 on *Formal and Connectionist Systems*).

If we accept the importance of syntax, a formal semantics for natural language has to do the job of characterizing contexts of use and the uses of the syntactic forms. To be formal it has to provide both a formal definition for contexts of use and for the semantic function. This requires a formal language were contexts of use are explicit. This is the case, for example, of the constructively typed language we have been using to define the semantics of the intensional language *L*.

We believe that under this view of syntax and semantics many of the arguments on meaning became circular or non-problems. For instance the argument on whether the semantics of a declarative sentence is a truth-value. For us, *sentence* is a syntactic question applying to strings of characters, *declarative* is a question of use, a semantic question. A declarative sentence is thus one which is used to affirm or deny and which can be judged as being true or false. So, 'the semantics of a declarative sentence is a truth-value' would be an analytic proposition, i.e. containing its own truth. It would not be of any use to ask if we can syntactically characterize the declarative sentences of natural language. To be declarative is not a direct question of form, but of use, a semantical question. In a computational environment and given a context of use, we should be able to say, or compute, whether a declarative sentence is true, false or perhaps that we do not known.

The difference between the semantic types of declarative and indicative sentences we referred to in chapter one, section 1.3.4 on *Types and Types of Sentences* is, we claim, just one example of sentence typing. If we want to be precise, the form of the sentence does not give me a definite answer to whether a sentence is interrogative,

imperative or declarative, unless I make an explicit formal convention beforehand. Such classifications must in general be given by the use of the sentence. As Winograd observes,

> The syntactic form of an utterance is not correlated in a one-to-one way with its use as a *speech act*. The sentence *You're going with him?* if said with a rising intonation pattern, is a question, even though its form is declarative. The sentence *What was I gonna do?* is used most often not as a question (as its form would indicate), but as a *rhetorical question* — a statement conveying my feeling that I had no alternative to the action I chose.
>
> Winograd[1983], pp. 295

Although we claim that the meanings of say a declarative sentence and an indicative sentence are different we are not claiming that they are unrelated. The separation into the two types of use is made in order to reflect and make precise different aspects of the overall possible uses of the sentence.

We believe this discussion on types of sentences is justified here, because below we will have to make a choice on types for sentences and other expressions.

3.5.3 Computable Grammars to Approximate English

In this section we will define two approximations for a grammar of English. We begin by defining an extensional approximation of English and then by extending it to an approximation to intensional English. This improved aproximation is defined by adding syntactic and semantic rules to the extensional approximation, and solving type equations. In both cases sentences will be used declaratively, like in Montague's approximation to English, and therefore the type equations will have as constraint that the meaning of a sentence is of type boolean, for the extensional case, and of type intensions on booleans for the intensional case.

Instead of using meaning postulates after defining the intensional interpretation, the definition of meanings for the lexicon of the intensional approximation will guarantee that *the natural requirements* behind Montague's postulates are satisfied. We will claim this without proof, as it seems to us much simpler than to prove that Montague's meaning postulates do correspond to those requirements.

The initial approximation, we will call *⁰English*, is an extensional English defined with a context free syntax and a compositionally defined interpretation [5]. In our notation all elements, in the syntax or semantics, used in formalizing this version of English will be preceded by the superscript 0. As we will see, the equations for the compositional interpretation define a set of type equations with free variables, which we can identify with the usual extensional solution, when for instance declarative sentences denote boolean values.

The last approximation, which we will call *¹English*, is an intensional English defined by adding three syntactic rules and the corresponding semantic definitions, and by solving the type equations for an intensional semantics, where for instance sentences denote boolean functions on indices. The meaning of a word in this approximation is based on its meaning in *⁰English*. We claim that by doing this in the form shown satisfies the natural requirements for the meaning postulates.

About the Notation

The notation for the semantic system is similar to the one in the rest of this thesis, but for instance we will use simple brackets '(' and ')' instead of '$\mathbb{(}$' and '$\mathbb{)}$'. However, as we said in the beginning of this section, we wil use a simplified and compact notation for the syntax. Basically we describe the syntax using standard notations of denotational semantics. For instance, while in Montague's system the set of expressions of some category x is written P_x, here we will just write X, say, instead of P_{det} we will just write *Det*, and instead of writing as in definition 2.3.1.4, where for instance determiners are introduced as syncategorematic elements:

[...]

2. If $\alpha \in P_{CN}$ then $F_0(\alpha), F_1(\alpha), F_2(\alpha) \in P_T$, where:

   $F_0(\alpha) = \textit{every}\ \alpha$,

   $F_1(\alpha) = \textit{the}\ \alpha$,

   $F_2(\alpha) = \textit{a}\ \alpha$ [...],

[5] If to the syntactic rules corresponds a program, they correspond as well to a function on strings. The function which 'corresponds' to the context free syntax will be the basis on which the interpretation is compositionally defined.

we will write for the definition of determiners, rule 1 below, and for rule 2 above, rule 6 below, where the underscore corresponds to concatenation with a blank character:

1. ^0Det ::= ***every*** / ***the*** / ***a*** .

[...]

6. ^0Np ::= $^0Det_^0Cn$.

Approximation ^{0}English

To introduce the first approximation to English, we define the language's alphabet with the set ^0Char and the syntactic categories with the set ^0Catg. Then we informally define a set of phrase markers, which we call ^0PhrMk [6].

^0Char = {_,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,x,y,w,z}

^0Catg = {det, verb, pn, cn, pron, np, vp, sent}

^0Strg ::= ^0Char / ^0Char ^0Strg

For instance, the set ^0Strg just defined has as elements ***a, the, john_loves_mary*** and ***fgtyfyg***. We will only define informally the set of phrase markers ^0PhrMk as being the set of possible parse trees defined by the context-free syntax and based on the sets ^0Strg and ^0Catg. For instance, ***(((a,det),(man,cn)),np)***, ***((walks,verb),vp)***, and ***(a,det)*** are elements of the set of phrase markers. The category on the top of the tree names the pharse marker, and so the previous examples are respectively a *np*, a *vp* and a *det* phrase marker.

Syntactic Rules

1. ^0Det ::= ***every*** / ***the*** /

a .

[6] The syntactic rules can be seen as defining a syntactic function, ^0f, which for instance has as arguments elements of the strings of the language and returns elements of the set of phrase markers. We could use this syntactic function as the support for the isomorphism between syntax and semantics.

2.	^0Verb	::=	***walk***	/ ***believe***	/
			love	/ ***seek***	.
3.	^0Pn	::=	***john***	/ ***mary***	.
4.	^0Cn	::=	***man***	/ ***woman***	/
			unicorn	.	
5.	^0Pron	::=	***he***	/ ***she***	.
6.	^0Np	::=	^0Pn	/ ^0Pron	/
			$^0Det_^0Cn$	.	
7.	^0Vp	::=	^0Verb	/ $^0Verb_^0Np$	.
8.	^0Sent	::=	$^0Np_^0Vp$	.	

As we have accepted compositionality, there must be a one to one correspondence between syntactic and semantic rules in the interpretation we will define below. The interpretation function ^0u has as arguments phrase markers and values in some domain of values ^0Values. If the phrase marker corresponds to a sentence, then the value of the interpretation of that phrase marker is a boolean.

$$^0u : {}^0PhrMk \rightarrow {}^0Values$$

Semantic Rules

The interpretation of the lexicon is given in the first 5 rules. We can read the vertical bar as *or*. For instance: rule 1 says that the meaning of a *det* phrase marker is either given by the meaning of ***a***, of ***every*** and of ***the***; rule 6 says that the meaning of a *np* phrase marker is given either by the meaning of a *pn* or a *pron* phrase marker or by the composition of the meaning of its constituent phrase markers in the indicated way.

1.	$^0u(^0Det)$	:≈	$^0u(a)$	/ $^0u(every)$	/
			$^0u(the)$	.	
2.	$^0u(^0Verb)$	:≈	$^0u(walk)$	/ $^0u(believe)$	/

		$^0u(love)$	/ $^0u(seek)$	.
3. $^0u(^0Pn)$	$\coloneqq$	$^0u(john)$	/ $^0u(mary)$	.
4. $^0u(^0Cn)$	$\coloneqq$	$^0u(man)$	/ $^0u(woman)$	/
		$^0u(unicorn)$	.	
5. $^0u(^0Pron)$	$\coloneqq$	$^0u(he)$	/ $^0u(she)$	.
6. $^0u(^0Np)$	$\coloneqq$	$^0u(^0Pn)$	/ $^0u(^0Pron)$	/
		$^0u(^0Det)\ (^0u(^0Cn))$	.	
7. $^0u(^0Vp)$	$\coloneqq$	$^0u(^0Verb)$	/ $\lambda x.\ ^0u(^0Verb)\ (x,\ ^0u(^0Np))$	.
8. $^0u(^0Sent)$	$\coloneqq$	$^0u(^0Vp)\ (^0u(^0Np))$	.	

The definitions seem to be the obvious ones. As a first approximation we have choosen for the composition functions in rules 6, 7 and 8, the ones which will provide an initial approximation to a functional use of English without quantification.

Semantic Equations

This choice of composition introduced above defines a system of equations for the use-values of the expressions. If D_{catg} is the domain of values for the meaning of *catg* phrase markers, and D_{strg} is the domain of values for the meaning of strings, we will require, for instance for semantic rule 1, the equation:

eq1. $D_{det} = D_a + D_{every} + D_{the}$.

It is not difficult to verify that the following equations, on domain variables $\mathscr{X}$, $\mathscr{Y}$ and $\mathscr{Z}$, provides a solution for the grammar above [7].

eq1. $^0u(^0Det)$: $\quad ^0D_{det} = \mathscr{Y} \to \mathscr{X}$

eq2. $^0u(^0Verb)$: $\quad ^0D_{verb} = \mathscr{X} \to \mathscr{Z} + \mathscr{X} \times \mathscr{X} \to \mathscr{Z}$

[7] It would be interesting to investigate how to use automatic processes to find a solution. For instance could polymorphic type checking, used in some programming languages, see for instance Damas[1984], be used here? Given an arbitrary untyped grammar of the language, such as the previous one, we would like to generate automatically the type equations and solve them for each type of semantics.

eq3. ${}^0u({}^0Pn)$: ${}^0D_{pn} = \mathscr{X}$

eq4. ${}^0u({}^0Cn)$: ${}^0D_{cn} = \mathscr{Y}$

eq5. ${}^0u({}^0Pron)$: ${}^0D_{pron} = \mathscr{X}$

eq6. ${}^0u({}^0Np)$: ${}^0D_{np} = \mathscr{X}$

eq7. ${}^0u({}^0Vp)$: ${}^0D_{vp} = \mathscr{X} \rightarrow \mathscr{Z}$

eq8. ${}^0u({}^0Sent)$: ${}^0D_{sent} = \mathscr{Z}$

To have an extensional English, we just need to think of $\mathscr{Z}$ as the domain $\mathscr{B}$ of booleans. The domain $\mathscr{X}$ is usually referred to as the domain $\mathscr{E}$ of entities or individuals. The form of domains D_{verb} and D_{vp} reflects the treatment of the use of intransitive and transitive verbs. Domain D_{cn} is usually associated with $\mathscr{E} \rightarrow \mathscr{B}$. In extensional English, we adopt therefore the following solution:

$$\mathscr{X} = \mathscr{E} \qquad \mathscr{Y} = \mathscr{E} \rightarrow \mathscr{B} \qquad \mathscr{Z} = \mathscr{B} \quad .$$

Examples

We will write for instance ${}^0u(john)$, but more precisely we should write between the brackets the phrase marker associated with ***john*** instead of ***john***. The meaning for the lexicon is given.

${}^0u(john) = {}^0john : \mathscr{E}$

${}^0u(mary) = {}^0mary : \mathscr{E}$

${}^0u(walk) = {}^0walk : \mathscr{E} \rightarrow \mathscr{B}$

${}^0u(love) = {}^0love : \mathscr{E} \times \mathscr{E} \rightarrow \mathscr{B}$

${}^0u(seek) = {}^0seek : \mathscr{E} \times \mathscr{E} \rightarrow \mathscr{B}$

${}^0u(believe) = {}^0believe : \mathscr{E} \times \mathscr{E} \rightarrow \mathscr{B}$

${}^0u(man) = {}^0man : \mathscr{E} \rightarrow \mathscr{B}$

${}^0u(woman) = {}^0woman : \mathscr{E} \rightarrow \mathscr{B}$

${}^0u(unicorn) = {}^0unicorn : \mathscr{E} \rightarrow \mathscr{B}$

${}^0u(a) = {}^0a : (\mathscr{E} \rightarrow \mathscr{B}) \rightarrow \mathscr{B}$

${}^0u(john_walk) = {}^0walk({}^0john)$

${}^0u(john_love_mary) = {}^0love({}^0john, {}^0mary)$

${}^0u(john_seek_mary) = {}^0seek({}^0john, {}^0mary)$

$^0u(\textit{john_seek_a_unicorn}) = {}^0seek({}^0john, {}^0a({}^0unicorn))$

$^0u(\textit{john_believe_mary}) = {}^0believe({}^0john, {}^0mary)$

Comments

We could have made explicit in the syntactic definitions the difference between transitive and intransitive verbs. What we have done is equivalent as it is assumed that some verbs will have use-values of type $\mathscr{X} \rightarrow D_{sent}$ and others of type $\mathscr{X} \times \mathscr{X} \rightarrow D_{sent}$.

To classify a full version of first-order English we would need to add different semantic composition functions instead of the ones used in rules 7 and 8, for example as Montague so successfully did, and solve a new system of equations to find the correct denotations. We believe we would not need to change the other definitions. As Montague has shown, the process of reducing natural language using formal logic is not trivial and needs to be defined. Using a more complex set of semantic composition functions and assuming the use of full first order logic, we should be able to get a better i-aproximation to first-order extensional English. For the expression *john_seek_a_unicorn* we should write with an extensional reading,

$$^iu(\textit{john_seek_a_unicorn}) = \forall x.\ ({}^0unicorn(x) \wedge {}^0seek({}^0john, x))\ .$$

Approximation ^{1}English

In the following approximation to English, called ^{1}*English*, we will account for intensional phenomena of the same kind covered by Montague in PTQ. We claim that our definition for ^1u is 'logically possible' satisfying the motivations for Montague's meaning postulates 1-7. Proper nouns are 'logically determinate' (postulate 1), all our common nouns are 'ordinary' (postulate 2), and intensionality and extensionality with respect to subject or object position is respected depending on the type of the verb (postulates 3-7). We will assume that ^{1}*English* is exactly as ^{0}*English* except for the following rules which are added to the previous ones.

Extra Syntactic Rules

7. ^1Vp ::= [...] / $^1Verb_^1Sent$.
8. ^1Sent ::= [...] / $^1Adj_^1Sent$.
9. ^1Adj ::= [...] / ***necessarily*** .

Extra Semantic Rules

7. $^1u(^1Vp)$:= [...] / $\lambda x.\ ^1u(^1Verb)\ (x,\ ^1u(^1Sent))$.
8. $^1u(^1Sent)$:= [...] / $^1u(^1Adj)\ (^1u(^1Sent))$.
9. $^1u(^1Adj)$:= [...] / $^1u(necessarily)$.

Extra Semantic Equations

Using both sets of rules, the previous ones and the ones of extensional English, we propose the following solution, which can be used to characterize intensional aspects of English.

eq1. $^1D_{det} = (\mathcal{E} \rightarrow \mathcal{B}) \rightarrow (\mathcal{I} \rightarrow \mathcal{E})$
eq2. $^1D_{verb} = (\mathcal{E} \times (\mathcal{I} \rightarrow \mathcal{E})) \rightarrow (\mathcal{I} \rightarrow \mathcal{B}) + (\mathcal{E} \times (\mathcal{I} \rightarrow \mathcal{B})) \rightarrow (\mathcal{I} \rightarrow \mathcal{B})$
eq3. $^1D_{pn} = \mathcal{E}$
eq4. $^1D_{cn} = \mathcal{E} \rightarrow \mathcal{B}$
eq5. $^1D_{pron} = \mathcal{E}$
eq6. $^1D_{np} = \mathcal{I} \rightarrow \mathcal{E}$
eq7. $^1D_{vp} = \mathcal{E} \rightarrow (\mathcal{I} \rightarrow \mathcal{B})$
eq8. $^1D_{sent} = \mathcal{I} \rightarrow \mathcal{B}$
eq9. $^1D_{adj} = (\mathcal{I} \rightarrow \mathcal{B}) \rightarrow (\mathcal{I} \rightarrow \mathcal{B})$

We need to provide the meaning of intensional words in this new approximation. But instead of defining all of them, we should use as much as possible the meanings already used in the extensional approximation. We will also need to redefine some of the meanings of extensional words when they are used in intensional positions, for which it is required a type different from the initially defined. In order to do the later we were influenced by Partee's [1988] type-shifting functors, which were proposed to solve a different problem. In our particular case

we will only have to construct new objects which denote intensional constant functions on the meaning of the initial. This constant functor is intuitively natural, as it does not provide more 'information'[8]. In order to do what we have just described informally, we provide new meanings for some words we want to treat intensionally:

$$
\begin{array}{lll}
{}^{1}u(seek) & = {}^{1}seek & : \ (\mathscr{E} \times (\mathscr{I} \to \mathscr{E})) \to (\mathscr{I} \to \mathscr{B}) \\
{}^{1}u(believe) & = {}^{1}believe & : \ (\mathscr{E} \times (\mathscr{I} \to \mathscr{B})) \to (\mathscr{I} \to \mathscr{B}) \\
{}^{1}u(unicorn) & = {}^{1}unicorn & : \ \mathscr{E} \to \mathscr{B} \\
{}^{1}u(a) & = {}^{1}a & : \ (\mathscr{E} \to \mathscr{B}) \to (\mathscr{I} \to \mathscr{E}) \\
{}^{1}u(necessarily) & = {}^{1}necessarily & : \ (\mathscr{I} \to \mathscr{B}) \to (\mathscr{I} \to \mathscr{E})
\end{array}
$$

Whenever the words defined in *${}^{1}English$* are to be used in the intensional approximation we define, using what we call a *constant functor*:

$$
\begin{array}{ll}
{}^{1}u(john) & = \lambda i.\, {}^{0}u(john) \\
{}^{1}u(mary) & = \lambda i.\, {}^{0}u(mary) \\
{}^{1}u(walk) & = \lambda i.\, {}^{0}u(walk) \\
{}^{1}u(love) & = \lambda i.\, {}^{0}u(love) \\
{}^{1}u(man) & = \lambda i.\, {}^{0}u(man) \\
{}^{1}u(woman) & = \lambda i.\, {}^{0}u(woman)
\end{array}
$$

Comment on Meaning Postulates

Given our analysis in section 3.2.3 on *Meaning Postulates*, the choice of domains in equations eq1-eq9 above, and of this constant functor will result in the meaning postulates being unnecessary. In this intensional treatment we already have that proper nouns are 'logically determinate', as its denotations are constant with respect to context of use, common nouns are 'ordinary' as they denote constant individual concepts on contexts of use, and intransitive verbs are 'extensional' by definition in eq2. It also happens that all verbs are by definition extensional with respect to

[8] We use the word 'natural' in Partee's [1988] sense: if we are asked which is the 'real' number which best corresponds to an integer, the answer is: the 'integer' itself; the naturalness of converting reals into integers is much less intuitive, as Partee points out, except when the 'real' numbers correspond to integers.

subject position and extensional verbs are extensional with respect to object position as well, as we take their initial approximation which meaning is given in eq2.

Moreover, we have for this Intensional English that the meaning of words is defined in this unique place. As we have seen in the beginning of this section, the use of Meaning Postulates in Montague's systems corresponds to an undesirable double placed definition.

Examples

One of the objectives of constructing *^{1}English* was to provide an example that it is possible to avoid Montague's meaning postulates in a compositional semantics. We will now provide some examples of how to derive the meaning of the extensional and intensional sentences for the two approximations to English . Usually we should prefer the better approximation. Whenever this cannot be used, elements from version one can be used. For instance given its type, *^{1}seek* cannot be applied to *^{1}john*, so we use *^{0}john* instead.

u(john_seek_a_unicorn)	can be:	
^{1}u(john_seek_a_unicorn)	= ***^{1}seek(^{0}john,^{1}a(^{1}unicorn))***	or
^{0}u(john_seek_a_unicorn)	= ***^{0}seek(^{0}john,^{0}a(^{0}unicorn))***	(as before) [9]

u(john_love_mary)	can be:	
^{1}u(john_love_mary)	= ***^{1}love(^{0}john,^{1}mary)***	or
^{0}u(john_love_mary)	= ***^{0}love(^{0}john,^{0}mary)***	(as before)

u(john_seek_mary)	can be:	
^{1}u(john_seek_mary)	= ***^{1}seek(^{0}john,^{1}mary)***	or
^{0}u(john_seek_mary)	= ***^{0}seek(^{0}john,^{0}mary)***	(as before)

[9] If in *PTQ* we regard the star in the notation δ_* as a functor, it converts an object of some intensional type into an extensional relation. For a verb which is extensional such as ***love*** or ***find*** there is no problem, as the high-order object is used as a type-shifted version of its extensional use. But for a verb like ***seek*** its use corresponds to a loss in the information content which characterizes the intensional meaning function of the translated constant.

$u(necessarily_john_love_mary)$ can be:

$^1u(necessarily_john_love_mary) = {}^1necessarily({}^1love({}^0john, {}^1mary))$ but

$^0u(necessarily_john_love_mary)$ is undefined

$u(mary_believe_john_love_mary)$ can be:

$^1u(mary_believe_john_love_mary) = {}^1believe({}^0mary, {}^1love({}^0john, {}^1mary))$

$^0u(mary_believe_john_love_mary)$ is undefined

We do not attempt to define extraordinary common nouns such as *temperature* and *price*, although we believe it would be perfectly possible to do so in this same way, so common nouns only denote constant individual concepts on contexts of use.

As we have said before, this compositional treatment of intentionality for English is, if nothing else, an illustration of how we think English grammar should be presented without the technical device of meaning postulates. We think that although being a simple example, it tells us that intentionality and quantification in *PTQ* could be treated compositionally in the same way, without meaning postulates. Doing that, we would observe that *PTQ*'s English was compositionally translated into a formal system, a logic, where there is no quantification on variables other than of primitive type.

3.6 CONCLUSION

The main conclusion of this chapter is that Montague's treatment of English can be simplified without any loss of clarity or explanatory power by dropping the cap-cup operators from the intermediate language, and removing the meaning postulates as the last section illustrated. The most important result of this simplification is that higher order quantification would be removed from Montague's treatment of English.

We make no claim for the alternative approach to computable grammar introduced in section 3.5, because it clearly requires more work, as we will refer to in chapter

five. As we have said in the introduction, its main immediate objective was of showing how to eliminate the meaning postulates. Mediately important, it also shows how to be flexible in defining better aproximations for natural language.

> It seems paradoxical to us that we should make such a medley, mixing physical states and states of consciousness up together in a *single* report: "He suffered great torments and tossed about restlessly". It is quite usual; so why do we find it paradoxical? Because we want to say that the sentence deals with both tangibles and intangibles at once.— But does it worry you if I say: "These three struts give the building stability"? Are three and stability tangible? — Look at the sentence as an instrument, and at its sense as its employment.
>
> Wittgenstein[1953], §421, pp. 126[e]

> What I suggest is that *it is always undesirable to make an effort to increase precision for its own sake — especially linguistic precision — since this usually leads to loss of clarity* [...], *one should never try to be more precise than the problem situation demands.*
>
> Popper[1974], pp. 24

CHAPTER 4

TOWARDS COMPUTABLE DENOTATIONS FOR SITUATION SEMANTICS

4.1 INTRODUCTION

Situation semantics was developed to become a viable alternative to a propositional possible-worlds semantics for natural language, such as Montague's. Barwise[1981] expresses the view that possible-worlds semantics rests on dubious metaphysical and epistemological grounds, and refers to logical omniscience as a major defect which forces him to develop situation semantics. This semantics is to be regarded as a new model-theoretic approach to the analysis of natural language. In particular, Barwise uses a puzzle of logical equivalence involving perception

verbs to show that in the propositional possible-worlds account we cannot get the right logical laws for perception, as well as for belief, doubt and knowledge, because of logical omniscience. The conclusion is that possible-worlds semantics offers an unacceptable framework for a mathematical semantics of natural language.

Objectives

According to Bull and Segerberg[1984] 'possible-worlds semantics is the dominating kind of model-theoretic semantics, perhaps even, if we take advantage of the vagueness of this term and stretch it a little, the only kind' (pp. 3). We believe that it should be possible to capture the model-theoretic analysis of language undertaken by situation semantics without rejecting possible-worlds semantics and Montague's general framework. The first reason for that is because we share Barwise's views on the role of semantics in the study of natural language:

> Model-theoretic Semantics is an attempt to use mathematics as a vantage point from which to focus on the most fascinating aspect of the magic prism, namely, how it is possible that sounds and marks can sometimes mean something. Surely we will never completely solve the mystery of man's capacity for symbol mongering, but the study of semantics can systematize our implicit understanding of human language and make clear where some of the mysteries lie.
>
> Barwise[1981], pp. 369-372

In the second place, we believe that studying language's meaning with the notions of possible worlds, which comes at least from Leibniz, has proved to be very fruitful. Finally we wish to proceed because Montague's framework offers a powerful, and perhaps unique general point of reference for linguistics and both theoretical and computational logic. We would hope to use Barwise and Perry's useful results, particularly on the logics of perception, without having to make a major departure say from modal logic theory and from the Montagovian compositional approach to the semantics of natural language.

One of the major problems in using the results of situation semantics is that it still suffers from one kind of omniscience. In fact non-normal or non-standard modal treatments of propositional attitudes, see for instance Konolige[1986] particularly

chapter six, seem unsatisfactory because of their complexity (Dieu[1989]). We believe that to solve this problem we need to be more realistic in the construction of possible semantic interpretations, and in the definition of boolean valued interpretations. This realistic position is in contrast with Hintikka's [1975] idealistic formal views on propositional attitudes, which has been a major influence in the formalization of such sentences. The basic objective of the last section of this chapter will be to define the basic properties which possible real interpretations for propositional attitudes and perception should satisfy.

Our interpretation and presentation of situation semantics has much to owe to Landman's [1986] formal interpretation of situation theory using partial objects. However we are more interested in our formalization on the computability of the interpretation, and that is why partial objects, mainly functions are important to us.

> As I have stated before, in my opinion the reason why partial objects are interesting in semantics is, that you can take extensions of them into account, and — most importantly — that you can recognize patterns in these extensions, and constraints on these patterns. On an 'ecological' theory of meaning, organisms pick up meaning, or information by getting attuned to patterns.
>
> Landman[1983], pp. 14

In many respects, the work of Muskens[1988] claiming that it is possible to extend Montague's *PTQ* using Landman's analysis to cover the results of situation semantics, supports our objective of keeping within Montague's general framework. The first part of our work of formalization differs from both of these mainly because we use a theory of partiality as a theory of finite representation, with a view of providing computable characterizations for the necessary domains and functions [1]. The problem of omniscience is not an object of explicit concern either for Landman or Muskens.

Overview

This chapter provides a denotational treatment of natural language mainly inspired in

[1] For instance, our domains satisfy Plotkin's [1981] computability requirement, by not having a top element.

the one outlined by Barwise & Perry's [1983] 'Situations and Attitudes', or just *S&A*. Our treatment has been influenced by Landman[1986], Cooper[1987] and Moennich and Bealer[1988]. We then show that Barwise[1981] and Hintikka's [1975] criticism of possible-worlds semantics applies to situation semantics in respect of omniscience of the latter. The chapter finishes with some observations on the characteristics required by a formal treatment of perception sentences. We believe that some of these characteristics are common to the formal logic of propositional attitudes.

We introduce a denotational or computational treatment of natural language based on situation semantics. We will use a formal system, called *SSL*, which can be regarded as an alternative formal presentation of the theory of situations in *S&A*. Instead of defining a general system, we concentrate on formalizing the one required to construct the semantics of *Determiner-Free Aliass*, a language similar to the one defined in appendix to chapter six of *S&A*, which we call *DFA-NL*. A very Montagovian approach is taken to achieve this objective. We provide a syntactic definition for the language (section 4.2), we introduce the required theory of situations *SSL* (section 4.3), and we use it to express a compositional semantics for the language (section 4.4):

$$DFA\text{-}NL \xrightarrow[\text{interpretation}]{u} SSL$$

For instance, if $\alpha \in P_S$, i.e., α is a sentence of the language, then,

$$u[\![\alpha]\!]: SIT \rightarrow SIT \rightarrow \mathscr{B}$$

$$u[\![\alpha]\!]\, u\, e: \mathscr{B}$$

where *SIT* is the domain of situations, u is the situation where the sentence is uttered, and e is the situation being referred to by the utterance of the sentence. In order to characterize situation semantics, we think of *S&A*'s notations as abbreviations for typed lambda calculus functional expressions, and constructively interpreted using the definitions of chapter two.

In the last section (4.5) and once the semantics of *DFA-NL* has been formalized using our version of situation semantics, we extend the language and the system in order to cover attitude and perception sentences which involve sentence embedding. We compare it with possible-worlds semantics, and we conclude that while situation semantics is immune from boolean logical omniscience it can still suffer in general from omniscience. We then chose to investigate the importance of omniscience in a formalization using sentences involving perception verbs. We believe that the study of knowledge and belief has been less objectively conducted than it should be expected in a formal study, due to confusions between the real use of those words and a perfect ideal use. Perception offers us a much more objective area to work with. We offer a tentative formalization for perception, and we outline some properties a real interpretation for perception, and we believe for propositional attitudes, should satisfy.

4.2 BARWISE & PERRY'S ENGLISH

This definition of the syntactic system for *Determiner-Free Aliass* or *DFA-NL* follows closely the presentation in *S&A*. However, our style is very close to that used for *M-NL* in chapter 2. We do this to enable a more direct comparison between the semantics of *DFA-NL* and the semantics of *M-NL* as in chapter 3. The grammar of *DFA-NL* presented in *S&A* is originally better structured, as the rules are already presented as to be codified by an *LR-grammar* [2]. We give a set of syntactic categories and basic expressions for each syntactic category. Classically the basic expressions constitute the lexicon of the language, although, as we will see, in *S&A* there are basic expressions which are not in the lexicon. The non-basic expressions of the language, phrases and sentences, are then defined recursively.

Definition 4.2.1 The set *CAT* of *syntactic categories* of Barwise & Perry's Formal English, *DFA-NL*, is defined as follows:

1. Basic syntactic categories is the set $\{\mathbb{V}, \mathbb{N}, \mathbb{W}, \mathbb{I}, \mathbb{R}^1, \mathbb{R}^2, NAM\}$.

[2] See *Example from LR-Grammars*, section 2.3.1.

2. Syntactic categories is the set $CAT = \{RS, IT, TM, LRP, PP, NP, S\}$.

Definition 4.2.2 The following names will be used for the syntactic categories defined before:

$\mathbb{V}$ or *variables.*
$\mathbb{N}$ or *present tense markers.*
$\mathbb{W}$ or *past tense markers.*
$\mathbb{I}$ or *indeterminates.*
$\mathbb{R}^1$ or *unary relation symbols.*
$\mathbb{R}^2$ or *binary relation symbols.*
NAM or *names.*
RS or *relation symbols.*
IT or *individual terms.*
TM or *tense markers.*
LRP or *located relation phrases.*
PP or *property phrases (common nouns and verb phrases)*
NP or *noun phrases.*
S or *sentences.*

Definition 4.2.3 The sets B_a of *basic expressions* b_a of category a are given. For instance the set B_V is the set of basic expressions of the syntactic category of variables.

$B_V = \{h_1, h_2, h_3, \ldots\}$.
$B_N = \{n_1, n_2, n_3, \ldots\}$.
$B_W = \{w_1, w_2, w_3, \ldots\}$.
$B_I = \{sbj, obj, loc, bol\}$.
$B_R1 = \{running, walking, barking, r^1, s^1, \ldots\}$.
$B_R2 = \{eating, touching, biting, r^2, s^2, \ldots\}$.
$B_{NAM} = \{John, Mary, Bill, Jackie, \ldots\}$.
$B_{RS} = B_R1 + B_R2$.
$B_{IT} = \{I\} + B_{NAM} + B_V$.
$B_{TM} = B_N + B_W$.

The sets of *basic expressions* of other categories are empty. A *basic expression b* is a member of $B = \Sigma_a B_a$.

The elements in the sets B_{RS}, B_{IT} and B_{TM} are called by Barwise & Perry the *lexicon* of the language *DFA-NL*.

Definition 4.2.4 The sets P_a of *phrases* p_a of category a are introduced by the rules:

Basic rule:

1. For all *syntactic categories a*, any member of B_a is in P_a .

Sentence rules:

2.1. If $\alpha \in P_{NP}$ and $\beta \in P_{PP}$ then $F_1(\alpha,\beta) \in P_S$, where
$F_1(\alpha,\beta) = (\alpha\ \beta)$.

2.2. If $\phi, \psi \in P_S$ then $F_2(\phi,\psi) \in P_S$, where
$F_2(\phi,\psi) = (\phi\ \textbf{and}\ \psi)$.

2.3. If $\phi, \psi \in P_S$ then $F_3(\phi,\psi) \in P_S$, where
$F_3(\phi,\psi) = (\phi\ \textbf{or}\ \psi)$.

Property phrase rules:

3.1. If $\alpha \in P_{LRP}$ then $F_4(\alpha) \in P_{PP}$, where
$F_4(\alpha) = (\alpha)$.

3.2. If $\alpha \in P_{LRP}$ and $\beta \in P_{NP}$ then $F_1(\alpha,\beta) \in P_{PP}$.

3.3. If $\alpha \in P_{PP}$ and $\beta \in P_{PP}$ then $F_2(\alpha,\beta) \in P_{PP}$.

Noun phrase rules:

4.1. If $\alpha \in P_{IT}$ then $F_4(\alpha) \in P_{NP}$.

4.2. If $\alpha \in P_{NP}$ and $\beta_i \in P_V$ then $F_5(\alpha,\beta_i) \in P_{PP}$, where
$F_5(\alpha,\beta_i) = (\alpha_i)$.

Located relation phrase rule:

5. If $\alpha \in P_{RS}$ and $\beta \in P_{TM}$ then $F_6(\alpha,\beta) \in P_{LRP}$ and $F_7(\alpha,\beta) \in P_{LRP}$, where
   $F_6(\alpha,\beta) = (\alpha_\beta)$,
   $F_7(\alpha,\beta) = (\textit{not}\ \alpha_\beta)$.

A *meaningful* or *well-formed expression p* of *DFA-NL* is a member of $P = \Sigma_a\ P_a$.

For instance the following is a sentence, an element of P_S, of the language *DFA-NL*.

$$(\textit{Jackie biting}_{w_1}\ \textit{Mary}\ )\ \textit{and}\ (h_1\ \textit{walking}_{w_1}\ )$$

It corresponds to a more natural language like ***Jackie was biting Mary and she was walking***, with w_1 being a past tense marker and h_1 a variable standing for ***she***.

4.3 A Formal System for Situation Semantics: *SSL*

There are two possible ways we can use the mathematical theory outlined in *S&A*. As we have seen, we think of the mathematical language of Situation Semantics as that of a formal system. However, in the form in which it was originaly presented we do not find obvious its relation to computability. For instance, the relations between Scott's typed lambda system and recursion theory or computability theory are well known, through the use of Domain Theory as in Scott[1981]. In the previous chapters we used it to define the intermediate logic systems. If Situation Semantics' relation to recursion theory was as clear to us as that of lambda calculus, we could use it to express the interpretation of *DFA-NL*. Other works, such as Landman[1986] or Muskens[1988] can be seen as working towards this objective, although not as explicitly. What we propose to do in this section is to regard Situation Semantics' notation as an abbreviation for typed lambda calculus expressions. We could as well have chosen to define a set of expressions for Situation Semantics, introduce its syntax, and then provide an interpretation for that syntactic language. The former method was preferred as it keeps this presentation simple to follow, once the abbreviations are understood.

The present system, called *SSL* for *situation semantics language*, is a simple but non-trivial presentation of the basic theory of situation semantics as we understand it. We start by introducing the primitive and non-primitive semantic domains together with basic given interpretation functions. At the same time we also introduce some of the abbreviations with their use. We provide some of the definitions necessary to characterize the expressions of *DFA-NL*, which will be done in section 4.4.3, and also to describe the semantics of attitudes, in section 4.5.1. In order to do this some of the definitions have been simplified or not explicitly given. For instance we will not define *Event-types* and *Roles*, as they were defined in *S&A*. On the other hand we bring some elements into proper domains, such as *constraints* which are introduced as objects which will allow for descriptions of situations which can themselves describe specific constraints between situations.

4.3.1 Primitive Domains

Definition 4.3.1.1 The domain *POL* of *polarities* i is defined as,

$$POL = \{no, yes\} .$$

Definition 4.3.1.2 The domain $\mathcal{E}$ of *entities* or *individuals* e. For later comparison we identify it with the domain of entities defined in chapter 2.

$$e.g:\ \mathcal{E} = \{John, Mary, John's\ leg, \ldots\}$$

Definition 4.3.1.3 The domain $\mathcal{L}$ of *locations* l.

$\mathcal{L} = \mathcal{P} \times \mathcal{T}$ where,

$\mathcal{P}$ is the domain of *spatial locations*.

$\mathcal{T}$ is the domain of *temporal locations*. It is identified with the one in 2.3.3.2.

As in section 2.3.3 the domain of temporal locations is defined together with partial ordering relations. *S&A* uses the following ordering relations, intuitively given as elements of domain $\mathcal{L} \times \mathcal{L} \rightarrow \mathcal{B}$. Using an infix notation we would write:

$l < l'$: l *wholly temporally precedes* l'

$\ell \circ \ell'$: ℓ *temporally overlaps* ℓ'

$\ell @ \ell'$: ℓ *spatially overlaps* ℓ'

We will name ℓ_u, or universal location, the effectively given location which for any other location ℓ: $\ell_u \circ \ell$ and $\ell_u @ \ell$.

Definition 4.3.1.4 The domain $\mathcal{R}$ of *relations* ♦. This domain is a primitive in *S&A*.

$\mathcal{R} = \Sigma_n \mathcal{R}_n$ where,

$\mathcal{R}_0$ is the domain of *situational states.*

$\mathcal{R}_1$ is the domain of *properties.*

$\mathcal{R}_2$ is the domain of *binary relations.*

...

$\mathcal{R}_i$ is the domain of *i-ary relations.*

Having domains of relational elements as primitives is one of the characteristics of the semantical approach of Property Theory as can be seen for instance in Moennich and Bealer[1988] and Turner[1988a]. In order to relate a relation with objects we need a function, to provide its value i.e., which objects stand in that relation. This function must be given, and we define its type later in 4.3.2.5.

4.3.2 Non Primitive Basic Domains

The non primitive basic domains are the domains of Objects, Facts and Situations. An object can be seen as an entity such as *Natassia Kinski* or *Margaret Thatcher*; a fact is a simple description of an aspect of the real or imaginary world, such as the fact that *Natassia Kinski* is *Margaret Thatcher*; a situation is a partial collection of facts, which can be a view of some world, either possible or not. The notion of regarding the world as collections of facts goes back at least to Wittgenstein and is to be found for instance in Russell[1918].

Definition 4.3.2.1 The domain $\mathcal{O}_n$ of *objects.* Any element of $\mathcal{E}$ is in this domain, but there are other elements which can be counted as objects. For the moment we

will use the variable $\mathscr{X}$ for that domain.

$\mathscr{Obj}_n = \Pi_n \mathscr{Obj}$ (e.g.: $\mathscr{Obj}_2 = \mathscr{Obj} \times \mathscr{Obj}$) where,
$\mathscr{Obj} = \mathscr{C} + \mathscr{X}$

The domain $\mathscr{X}$ will be defined in 4.3.2.13 as a constraint on situations [3]. This choice for the definition of objects will allow a situation to express that a simple relation on entities holds or not at some location and also to express that a situation also involves a complex constraint on situations at a certain location. We hope this will became clearer later on.

Definition 4.3.2.2 The domain $\mathscr{C}$ of *constituent sequences* is defined as,

$\mathscr{C} = \Sigma_n \mathscr{C}_n$ (n≥0) where,
$\mathscr{C}_0 = \mathscr{R}_0$
$\mathscr{C}_n = \mathscr{R}_n \times \mathscr{Obj}_n$

We can regard constituent sequences as unpolarized facts about the world, like for instance *Green snow* or *John loves Mary*.

Situations

A situation is intuitively a set of polarized and located facts about the world. For instance, a situation could be in our universe the fact *Green snow* polarized positively together with in some location in Kent the fact *John loves Mary* polarized positively. Obviously, depending on the valuation function we are using, a situation may have different properties, for instance, it may not be actual. Formally we define situations as follows.

Definition 4.3.2.3 The domain of *situations* e is defined as,*POL*

$$SIT = (\mathscr{L} \times \mathscr{C} \times POL) \to \mathscr{B} \equiv \mathscr{L} \to \mathscr{C} \to POL \to \mathscr{B} .$$

[3] As we may notice, we are introducing a set of recursive domain equations: later on, in the definition of $\mathscr{X}$ a domain based on $\mathscr{Obj}$ will be used. Domain theory guarantees the existence of non-trivial solutions and we believe that Peter Aczel's theory of hypersets offers an equivalent mathematical framework for the solution of this kind of equations (see for instance Barwise and Etchmendy[1987]).

Notation and Abbreviations

We will refer now to a type of notation used in the following presentation of situation semantics and how it can be seen as an abbreviation of typed lambda expressions. Basically, we abbreviate a boolean function by its characteristic set. When a computational description is required we should transform this representation into a typed lambda expression. We will then use the infix operators + and - for addition and subtraction of characteristic sets in the obvious way and without presenting here their formal definition. These operations will be used to define operations on the semantic features in section 4.4.3.1.

In the following examples the first type of notation is the one introduced in chapter two, the second corresponds to the characteristic set of a total boolean function, and the third is the notation used in *S&A*.

Example A

1. $e = \lambda x.\ \mathbf{cond}(x=(l, (r_2, o_0, o_1), yes), \not t,$
   $\mathbf{cond}(x=(l, (s_2, o_0, o_1), yes), \not t, \not f))$

2. $e = \{ (l, (r_2, o_0, o_1), yes),\ (l, (s_2, o_0, o_1), yes) \}$

3. $e := at\ l:\ r_2, o_0, o_1 ; yes$
   $s_2, o_0, o_1 ; yes$

Example B

1. $e = \lambda x.\ \mathbf{cond}(x=(l, c_1, yes), \not t,$
   $\mathbf{cond}(x=(l, c_2, no), \not t, \not f))$

2. $e = \{ (l, c_1, yes),\ (l, c_2, no) \}$

3. $e := at\ l:\ c_1 ; yes$
   $c_2 ; no$

In the cases where the notation 2 or 3 is ambiguous, we should use notation 1.

Properties of Situations

In order to define the notion of a *structure of situations*, the notion in situation semantics equivalent to the classic model-theoretic structure, several concepts are required. Given that the relation domains are formed of primitive elements, as we said before, we need to be able to compute the values that relations have for objects in locations. For this we must assume a given *valuation* function. This function is necessary to computationally analyse situations and is the basis to compute the properties or classification of situations as being *actual, factual* or *coherent.* These classifications are necessary for instance to define in 4.4.3.5 the boolean meaning of a statement in a given situation, for a given structure of situations.

Definition 4.3.2.5 The valuation function val_n: $(\mathcal{R}_n \times Obj_n \times \mathcal{L}) \rightarrow \mathcal{B}$ is given for all the located facts.

Definition 4.3.2.6 *The function* $in : SIT \times (\mathcal{L} \times \mathcal{C} \times POL) \rightarrow \mathcal{B}$ is defined as,

$$in(e, s) = \textbf{cond}(e(s) = t,\ t,\ f)$$

For instance $in(e, (l, (s_2, o_0, o_1), yes)) = t$. This function just explicitly tells us if a certain polarized and located fact is present in a situation.

Definition 4.3.2.7 The function $actual : SIT \rightarrow \mathcal{B}$ is defined as,

$$actual\,(e) = \textbf{cond}(val(r_n, o_n, l) \quad \textbf{equival} \quad in(e, (l, (r_n, o_n), yes)) \textbf{ and } val(r_n, o_n, l)=f \textbf{ equival } in(e, (l, (r_n, o_n), no)), t, f).$$

Definition 4.3.2.8 The function $factual : SIT \rightarrow \mathcal{B}$ is defined as,

$$factual\,(e) = \textbf{cond}(in(e, (l, (r_n, o_n), yes)) \textbf{ implies } val(r_n, o_n, l) \textbf{ and } in(e, (l, (r_n, o_n), no)) \textbf{ implies } val(r_n, o_n, l)=f, t, f).$$

Definition 4.3.2.9 The function $coherent : SIT \rightarrow \mathcal{B}$ is defined as,

coherent (e) = **cond(** *in(e, (l, (r$_n$, o$_n$), yes))* **xor** *in(e, (l, (r$_n$, o$_n$), no))* **and**
in(e, (l, (same, o, o'), yes)) **implies** *o=o', t, f)* **and**
in(e, (l, (same, o, o), no)) = *f, t, f)*.

Discourse Situations and Connections

Discourse Situations and Connections are special types of situations. We will not define them here but we assume that given a situation it is possible to determine whether it falls in one of these two types. This classification depends on the structure of the situation, i.e. basically on which relations are in the facts of the situation. Associated with these domains there is a set of functions which analyse the situation, for instance the temporal location of some of its facts, and are indicated below. Discourse situations and connections play an important role in the definition of the interpretation of the language.

Definition 4.3.2.10 The domain of *discourse situations* D_{SIT} is a domain of elements *d: SIT* $\rightarrow \mathcal{B}$ which satisfy the conditions of being a discourse situation as in pp. 55 of *S&A*.

E.g, *d*:= *at* l_d: *speaking, Joe; yes*
addressing, Joe, Jonny; yes
saying, Joe, ***Jackie is biting Molly;*** *yes*

The functions *speaker* : $D_{SIT} \rightarrow \mathcal{E}$ and *loc* : $D_{SIT} \rightarrow \mathcal{L}$ are given. The function *speaker* identifies that role in the discourse situation, for instance *speaker(d)* is *Joe*. The function *loc* identifies the location of the discourse, *loc(d)* is l_d.

Definition 4.3.2.11 The domain of *connections* C_{SIT} is a domain of elements *c: SIT* $\rightarrow \mathcal{B}$ which satisfy the conditions of being a referring situation as in pp. 124 of *S&A*.

E.g, *c*:= *at* l_1: *speaking, Joe; yes*
addressing, Joe, Jonny; yes
saying, Joe, Jackie ; yes

referring to, Joe, Jackie , Jackie; yes

The following functions are given, *loc:* $C_{SIT} \times B_{TM} \rightarrow \mathscr{B}$, *entity:* $C_{SIT} \times B_V \rightarrow \mathscr{E}$ and *extens:* $C_{SIT} \times C_{SIT} \rightarrow \mathscr{B}$. Given a connection and a tense marker consistent with it, for instance if they are both past, the function *loc* identifies the location of the situation, otherwise it is undefined. Given a variable, a pronoun, the function *entity* provides an individual as its reference. The function *extens* is true when the second connection is contained in the first, i.e., it is its extension.

Constraints and Structures of Situations

The following definitions are required to formalize the notions of meaning of *sentences*, *utterances* and *statements* using situation semantics. The word constraint has different uses, as is explained at footnote 3 on chapter 6 of *S&A*. If we see a constraint as a relation between situations, we will write definition 4.3.2.12. A constraint is then a computable element of that domain which corresponds to some computable expression or program. If we want to use situation semantics to actually write constraints, and constraints are to be situations definable like *e:= at l: involves, e',e" ; yes* , then situations *e'* and *e"* should also be viewed as objects in domain $\mathscr{O}$. Our choice however is to have the definition of a constraint so we can write situations such as *e:= at l: involves, c; yes* , where *involves* is a given unary relation and *c* is a constraint. Or better, the representation of a constraint, which can be a rather 'long' object. For us the important thing is that constraints are objects as in 4.3.2.12. Typed lambda expressions can be written to 'program' constraints and they can be used in the descriptions of situations, which could then also be called constraints. This proposal is to be seen at slight variation from Barwise & Perry's.

Definition 4.3.2.12 The domain of *constraints* is defined as,

$$CO = SIT \times SIT \rightarrow \mathscr{B} .$$

Definition 4.3.2.13 The domain $\mathscr{X}$ introduced in 4.3.2.3 is now defined as being isomorphic to CO .

$\mathscr{X} = CO$.

Definition 4.3.2.14 The domain of *collections of situations* is defined as $CS_{SIT} = SIT \rightarrow \mathscr{B}$

E.g.: $CS = \{ \mathscr{m}, \{ e, \{(l, y, i), (l, (r, a), 1)\}, e'', ...\}, ... \}$

We will just reproduce Barwise & Perry's definition for what is a structure of situations. This definition will be used when defining the boolean meaning of a statement. We should be able to translate it into our formalism. For our purpose however, it is only necessary that it is defined in some intuitively correct way, and in this respect we completely rely on *S&A*.

Definition 4.3.2.15 The domain of *structures of situations* S_{SIT} is a domain of elements $m: SIT \rightarrow \mathscr{B}$ which satisfy the conditions:

1. There is a non empty m_0 such that $m_0 \subseteq m$.
2. Every $e \in m_0$ is *coherent*.
3. If $e \in m_0$ and $e' \subseteq e$ then $e' \in m$.
4. If $m' \subseteq m$ then it exists $e \in m_0$ such that for every $e' \in m'$, e' is a *part of* e.
5. If c is any *factual constraint* in m then m respects c.

This concludes our presentation of the domains, functions and notation which we will require as defined to construct the meaning or interpretation of *DFA-NL*. An implementation of this system would of course need all the functions and domains we have assumed to be represented in some computable form.

4.4 THE INTERPRETATION OF *DFA-NL* WITH *SSL*

After defining the syntax of the language *DFA-NL* and a theory of situations with enough primitives, we can now construct the interpretation that corresponds to our mathematical understanding of our use of that language.

$$DFA\text{-}NL \xrightarrow[\text{interpretation}]{u} SSL$$

The interpretation function is defined to provide a compositional semantics for the language. For instance if $\alpha \in P_S$, i.e., if α is a sentence of the language,

$$u[\![\alpha]\!]: SIT \rightarrow SIT \rightarrow \mathscr{B}$$

$$u[\![\alpha]\!]\, u\, e : \mathscr{B}$$

where *SIT* is the domain of situations, u is the situation where the sentence is uttered, and e is the situation being referred to by the utterance of the sentence. An and utterance of an indicative sentence is called a statement and the interpretation u is then used to define the notions of weak and strong equivalence between statements and of boolean meaning of a statement.

4.4.1 Interpretation of Lexicon

Definition 4.4.1.1 The *lexical interpretation* function f is defined over the elements of the basic domains which constitute the lexicon of *DFA-NL*.

$$f: \quad (B_R1 \rightarrow \mathscr{R}_1) + (B_R2 \rightarrow \mathscr{R}_2) + (B_{NAM} \rightarrow \mathscr{E}) .$$

4.4.2 Semantic Features and Settings

In order to take account of semantic issues such as anaphora, tense and the subject and direct object functions, situation semantics introduces a set of semantic features for each expression of the language. For instance *Scat* is going to provide indeterminates ***sbj***, ***obj***, ***loc*** and ***bol*** which will store relevant information respectively on subject, direct object, location and semantic boolean value possibly associated with an expression of *DFA-NL*. In order to formalize this mechanism we will define a function setting as a function on each expression of *DFA-NL*. For each expression and depending on which set of indeterminates are relevant (this set depends on the syntactic category of the expression), a set of semantic values is given to the expression.

Definition 4.4.2.1 The following *semantic feature* functions are defined on the elements of P with the following types:

1. $Scat: \quad P \rightarrow B_I \rightarrow \mathcal{B}$
2. $Var: \quad P \rightarrow B_V \rightarrow \mathcal{B}$
3. $Tense: P \rightarrow B_{RS} \rightarrow \mathcal{B}$
4. $AF: \quad P \rightarrow B_V \rightarrow \mathcal{B}$

E.g., $Scat\ (\alpha) = \{sbj, loc, bol\}$.

The *semantic feature* functions will be given for all expressions in B_{RS} , B_{IT} , B_{TM} , in the basic rule of definition 4.4.3.1.

Definition 4.4.2.2 The function *setting* σ is defined on the elements of P. Given an element p of P:

$$Sett = P \rightarrow B_I \rightarrow \mathcal{E} + \mathcal{L} + \mathcal{B}$$

E.g., $\sigma(touching) = \begin{bmatrix} sbj & john \\ obj & mary \\ loc & \ell \\ bol & \ell \end{bmatrix}$

The function *extens* : $Sett \times Sett \rightarrow \mathcal{B}$ is defined in the obvious way, if the second setting is included in the first, then *extens* will be true.

4.4.3 Semantics of *DFA-NL*

Compositional Interpretation

The following definition is based on the definition of the meaning of *Determiner-Free Aliass* as is given in pp. 299 of *S&A*. It only uses functions defined or referred to previously [4].

Definition 4.4.3.1 The *semantic interpretation function* u is defined on the elements

[4] The presentation of this language should be seen as falling into the computational paradigm which was presented in chapter 3, section 3.5.

of P following the syntactic definition 4.2.4 of language *DFA-NL*,

$$u : P \rightarrow D_{SIT} \rightarrow C_{SIT} \rightarrow Sett \rightarrow SIT \rightarrow \mathscr{B}$$

$$u[\![p]\!]\, d\, c\, \sigma\, e : \mathscr{B}$$

Basic rule:

1.1.1. If $\alpha \in P_{R^1}$:

$$Scat\ (\alpha) = \{sbj, loc, bol\}$$

$$Var\ (\alpha) = \{\ \}$$

$$Tense\ (\alpha) = \{\ \}$$

$$AF\ (\alpha) = \{\ \}$$

$$u[\![\alpha]\!]\, d\, c\, \sigma\, e = in(e, (f[\![\alpha]\!], \sigma[\![\alpha]\!](sbj), \sigma[\![\alpha]\!](bol))\ )$$

1.1.2. If $\alpha \in P_{R^2}$:

$$Scat\ (\alpha) = \{sbj, obj, loc, bol\}$$

$$Var\ (\alpha) = \{\ \}$$

$$Tense\ (\alpha) = \{\ \}$$

$$AF\ (\alpha) = \{\ \}$$

$$u[\![\alpha]\!]\, d\, c\, \sigma\, e = in(e, (f[\![\alpha]\!], \sigma[\![\alpha]\!](sbj), \sigma[\![\alpha]\!](obj), \sigma[\![\alpha]\!](bol))\ )$$

1.2. If $\alpha \in P_{IT}$:

$$Scat\ (\alpha) = \{sbj\}\ .$$

$$Tense\ (\alpha) = \{\ \}$$

$$AF\ (\alpha) = \{\ \}$$

1.2.1. If $\alpha \in \{I\}$:

$$Var\ (\alpha) = \{\ \}$$

$$u[\![\alpha]\!]\, d\, c\, \sigma\, e = cond(\sigma[\![\alpha]\!](sbj) = speaker(d),\ t, f\ ).$$

1.2.2. If $\alpha \in P_{NAM}$:

$$Var\ (\alpha) = \{\ \}$$

$$u[\![\alpha]\!]\, d\, c\, \sigma\, e = cond(\sigma[\![\alpha]\!](sbj) = f[\![\alpha]\!],\ t, f\ ).$$

1.2.3. If $\alpha \in P_V$:

$$Var\ (\alpha) = \{\alpha\}$$

$$u[\![\alpha]\!]\, d\, c\, \sigma\, e = cond(\sigma[\![\alpha]\!](sbj) = entity(c, \alpha),\ t, f\ ).$$

1.3. If $\alpha \in P_{TM}$:

$$Scat(\alpha) = \{loc\}$$
$$Var(\alpha) = \{\}$$
$$Tense(\alpha) = \{\alpha\}$$
$$AF(\alpha) = \{\}$$

1.3.1. If $\alpha \in P_N$:

$$u[\![\alpha]\!]\, d\, c\, \sigma\, e = cond(\ell = loc(c, \alpha) \text{ and } \ell \circ loc(d),\ t, f).$$
$$\text{where } \ell = \sigma[\![\alpha]\!](loc)$$

1.3.2. If $\alpha \in P_W$:

$$u[\![\alpha]\!]\, d\, c\, \sigma\, e = cond(\ell = loc(c, \alpha) \text{ and } \ell < loc(d),\ t, f).$$
$$\text{where } \ell = \sigma[\![\alpha]\!](loc)$$

Sentence rules:

2.1. If $\alpha \in P_{NP}$ and $\beta \in P_{PP}$ then $p = F_1(\alpha,\beta) \in P_S$, and,

$$Scat(p) = \{\}$$
$$Var(p) = Var(\alpha) + (\, Var(\beta) - AF(\alpha)\,)^5$$
$$Tense(p) = Tense(\alpha) + Tense(\beta)$$
$$AF(p) = AF(\alpha) + AF(\beta)$$
$$u[\![p]\!]\, d\, c\, \sigma\, e = cond(\text{exists } \sigma'.\ extens(\sigma,\sigma') \text{ and } u[\![\alpha]\!]\, d\, c'\, \sigma'\, e \text{ and}$$
$$u[\![\beta]\!]\, d\, c'\, \sigma'\, e\ ,\ t, f\ ).$$

2.2. If $\phi, \psi \in P_S$ then $p = F_2(\phi,\psi) \in P_S$, and,

$$Scat(p) = Scat(\alpha) + Scat(\beta)$$
$$Var(p) = Var(\alpha) + (\, Var(\beta) - AF(\alpha)\,)$$
$$Tense(p) = Tense(\alpha) + Tense(\beta)$$
$$AF(p) = AF(\alpha) + AF(\beta)$$
$$u[\![p]\!]\, d\, c\, \sigma\, e = cond(\text{exists } c'.\ extens(c,c') \text{ and } u[\![\alpha]\!]\, d\, c'\, \sigma\, e \text{ and}$$
$$u[\![\beta]\!]\, d\, c'\, \sigma\, e\ ,\ t, f\ ).$$

[5] See in section 4.3.2 on *Notation and Abbreviations* the definition of operators + and - .

2.3. If $\phi, \psi \in P_S$ then $p=F_3(\phi,\psi)\in P_S$, and,

$Scat\ (p) = Scat\ (\alpha) + Scat\ (\beta)$

$Var\ (p) = Var\ (\alpha) + (\ Var\ (\beta) - AF\ (\alpha)\ )$

$Tense\ (p) = Tense\ (\alpha) + Tense\ (\beta)$

$AF\ (p) = AF\ (\alpha) + AF\ (\beta)$

$u[\![p]\!]\ d\ c\ \sigma\ e = \mathbf{cond}(\mathbf{exists}\ c'\ .\ extens(c,c')\ \mathbf{and}\ u[\![\alpha]\!]\ d\ c'\ \sigma\ e\ \mathbf{or}\ u[\![\beta]\!]\ d\ c'\ \sigma\ e\ ,\ t, f\ ).$

Property phrase rules:

3.1. If $\alpha \in P_{LRP}$ then $p=F_4(\alpha)\in P_{PP}$, and,

$Scat\ (p) = Scat\ (\alpha)$

$Var\ (p) = Var\ (\alpha)$

$Tense\ (p) = Tense\ (\alpha)$

$AF\ (p) = AF\ (\alpha)$

$u[\![p]\!]\ d\ c\ \sigma\ e = \mathbf{cond}(Scat\ (\alpha) = \{sbj\}, u[\![\alpha]\!]\ d\ c\ \sigma\ e\ ,\ \perp\ ).$

3.2. If $\alpha \in P_{LRP}$ and $\beta \in P_{NP}$ then $p=F_1(\alpha,\beta)\in P_{PP}$ and,

$Scat\ (p) = \{sbj\}$

$Var\ (p) = Var\ (\beta)$

$Tense\ (p) = Tense\ (\beta)$

$AF\ (p) = AF\ (\beta)$

$u[\![p]\!]\ d\ c\ \sigma\ e = \mathbf{cond}(obj \in Scat(\alpha)\ \mathbf{and\ exists}\ \sigma', \sigma''.\ extens(\sigma,\sigma')\ \mathbf{and}\ u[\![\alpha]\!]\ d\ c\ \sigma'\ e\ \mathbf{and}\ u[\![\beta]\!]\ d\ c\ \sigma''\ e\ \mathbf{and}\ \sigma'[\![\alpha]\!]\ (sbj) = \sigma''[\![\beta]\!]\ (obj),\ t, f\ ).$

3.3. If $\alpha \in P_{PP}$ and $\beta \in P_{PP}$ then $p=F_2(\alpha,\beta)\in P_{PP}$ and,

$Scat\ (p) = \{sbj\}$

$Var\ (p) = Var\ (\alpha) + (\ Var\ (\beta) - AF\ (\alpha)\ )$

$Tense\ (p) = Tense\ (\beta)$

$AF\ (p) = AF\ (\alpha) + AF\ (\beta)$

$u[\![p]\!]\ d\ c\ \sigma\ e = \mathbf{cond}(\mathbf{exists}\ c'.\ extens(c,c')\ \mathbf{and}\ u[\![\alpha]\!]\ d\ c'\ \sigma\ e\ \mathbf{and}\ u[\![\beta]\!]\ d\ c'\ \sigma\ e\ ,\ t, f\ ).$

Noun phrase rules:

4.1. If $\alpha \in P_{IT}$ then $p = F_4(\alpha) \in P_{NP}$ *and*

$Scat\ (p) = Scat\ (\alpha)$

$Var\ (p) = Var\ (\alpha)$

$Tense\ (p) = Tense\ (\alpha)$

$AF\ (p) = AF\ (\alpha)$

$u[\![p]\!]\ d\ c\ \sigma\ e = u[\![\alpha]\!]\ d\ c\ \sigma\ e$

4.2. If $\alpha \in P_{NP}$ and $\beta_i \in P_V$ then $p = F_5(\alpha,\beta) \in P_{PP}$, and,

$Scat\ (p) = Scat\ (\alpha)$

$Var\ (p) = Var\ (\alpha)$

$Tense\ (p) = Tense\ (\alpha)$

$AF\ (p) = AF\ (\alpha) + \beta_i$

$u[\![p]\!]\ d\ c\ \sigma\ e = cond(u[\![\alpha]\!]\ d\ c\ \sigma\ e\ \textbf{and}\ entity(c, p) = \sigma[\![\alpha]\!](sbj),\ t, f).$

Located relation phrase rule:

5. If $\alpha \in P_{RS}$ and $\beta \in P_{TM}$ then $p = F_6(\alpha,\beta) \in P_{LRP}$ and $q = F_7(\alpha,\beta) \in P_{LRP}$, and

$Scat\ (p) = Scat\ (q) = Scat\ (\alpha) - \{loc,bol\}$

$Var\ (p) = Var\ (q) = \{\ \}$

$Tense\ (p) = Tense\ (q) = Tense\ (\alpha)$

$AF\ (p) = AF\ (q) = \{\ \}$

$u[\![p]\!]\ d\ c\ \sigma\ e = cond(\textbf{exists}\ \sigma', \sigma'.\ extens(\sigma,\sigma')\ \textbf{and}\ \sigma[\![\alpha]\!](bol) = t$
$\textbf{and}\ u[\![\alpha]\!]\ d\ c'\sigma'e\ \textbf{and}\ u[\![\beta]\!]\ d\ c'\sigma'e\ ,\ t, f).$

$u[\![q]\!]\ d\ c\ \sigma\ e = cond(\textbf{exists}\ \sigma', \sigma'.\ extens(\sigma,\sigma')\ \textbf{and}\ \sigma[\![\alpha]\!](bol) = f$
$\textbf{and}\ u[\![\alpha]\!]\ d\ c'\sigma'e\ \textbf{and}\ u[\![\beta]\!]\ d\ c'\sigma'e\ ,\ t, f).$

We can simplify the meaning function for sentences, elements of P_S , as follows:

$$u : P_S \rightarrow D_{SIT} \rightarrow C_{SIT} \rightarrow SIT \rightarrow \mathscr{B}$$

$$u[\![p]\!]\ d\ c\ e : \mathscr{B}$$

This is justified because settings are not free in the definition of sentence meaning.

We can further simplify the meaning function for sentences, had we not distinguished the domains of Discourse situations and Connections. If the restrictions were built directly into the definition of meaning we could have defined u as follows, where u is the *uttering situation*, which provides the information of a discourse situation and a connection:

$$u : P_S \rightarrow SIT \rightarrow SIT \rightarrow \mathcal{B}$$

$$u[\![p]\!]\, u\, e : \mathcal{B}$$

Intuitively this is justified because a situation can contain a discourse situation and a connection. If it does not the interpretation would be undefined.

As we can see the domain $SIT \rightarrow SIT \rightarrow \mathcal{B}$ is the domain of constraints, the same that was defined in 4.3.2.12. This is the reason why in situation semantics the meaning of a sentence is said to be a *constraint on an utterance of a sentence.*

Definition 4.4.3.4 Two sentences α and β are *semantically equivalent* if they have the same interpretation for any uttering and referred situation, i.e. if and only if $u[\![\alpha]\!]\, u\, e = u[\![\beta]\!]\, u\, e$.

Weak and Strong Equivalence

The notions of equivalence are defined in *S&A* only for statements. The reasons for this will be explained later. Intuitively a statement is the utterance of a sentence. The interpretation of a statement in *S&A* is the collection of the referred situations by the sentence in the uttering situation. Given the interpretation of the language, the interpretation of a statement is determined. Strong equivalence is then defined between statements, not sentences. We make the observation that it does not use a quantification over possible interpretations, as the notion of logically equivalent sentences in *PTQ*. The interpretation works very much like a given constant which determines the meaning of the expressions in the language.

Definition 4.4.3.2 The domain of *statements* $\Phi = (p,d,c)$ is defined as $STAT = P_S \times D_{SIT} \times C_{SIT}$.

Definition 4.4.3.3 The *interpretation* u of a *statement* $\Phi = (p,d,c)$ is defined as follows using the semantic interpretation u of definition 4.4.3.1. We use the same letter for both functions, but the type of argument makes clear which one we are using.

$$u : STAT \rightarrow SIT \rightarrow \mathcal{B}$$

$$u[\![\Phi]\!]\, e = u[\![p]\!]\, d\, c\, e$$

Definition 4.4.3.4 Two statements Φ and Ψ are *strongly equivalent* if they have the same interpretation. By definition we write $\Phi \equiv \Psi$ *iff* $u[\![\Phi]\!] = u[\![\Psi]\!]$.

In order to define a notion of weak equivalence, *S&A* defines the boolean meaning of a statement. This definition uses a structure of situations. According to Barwise & Perry this substitutes the traditional definition of a model-theoretic structure.

Definition 4.4.3.5 The *boolean value* b of a statement Φ given a *structure of situations* m is defined as,

$$b : STAT \rightarrow S_{SIT} \rightarrow \mathcal{B}$$

$$b\,[\![\Phi]\!]\, m = cond(\textbf{exists}\ e.\ m(e) = t\ \textbf{and}\ u[\![\Phi]\!]e = t\ \textbf{and}\ actual(e)\ ,\ t, f).$$

Definition 4.4.3.6 Two statements Φ and Ψ are *weakly equivalent* if they are true in the same structures of situations. By definition, we will write $\Phi \leftrightarrow \Psi$ *iff* $b[\![\Phi]\!]\, m = b[\![\Psi]\!]\, m$.

The definition of boolean value (boolean meaning) as we can see, is not syntactically compositional, in the sense that it does not depend on the boolean value of the immediate constituents of the sentence, part of the statement. Intuitively, the utterance of an indicative sentence, a statement, is true in some model m (structure of situations) when we can find in that model a referred situation e which satisfies the constraint the statement imposes ($u[\![\Phi]\!]e$) and e is judged to be *actual.* This definition uses a verificationist theory of meaning for boolean value. In some situations an alternative definition based on a falsifiability theory might be

better suited.

4.5 About Omniscience and Perception

Compositional Semantics

The treatments of natural language we have been using, inspired by Montague and in Barwise & Perry, share the property that the meaning of expressions is mathematically defined with a compositional semantics. However the different concepts of what meaning is, are reflected on the form of the semantical objects. Montague takes the Tarskian view that *the basic aim of semantics is to characterize the notion of a true sentence and of entailment* while for Barwise & Perry, as we have seen, the aim is to characterize *constraints on utterances of sentences* [6].

Omniscience

In Montague Semantics, *MS*, the concept of logical equivalence is defined between sentences. As Hintikka[1975] has originally shown, its extension to propositional attitudes, like belief or knowledge, and the acceptance of inference rules or logical equivalences introducing new information indirectly brings logical omniscience into the semantical treatment. In Barwise and Perry semantics, or just *B&PS*, semantic equivalence concepts are defined directly only for statements, the utterances of sentences. Because the boolean value of statements is not compositionally defined, as we will show, no logical omniscience as such (logical as boolean) is ever observed. However, given that both treatments respect semantic compositionality, and because both treatments allow syntactic embedding of sentences, we would expect that in general both would suffer from some form or another of omniscience. As we will see, it may or may not be present in a particular formalization of *B&PS*, depending on the properties of the semantic interpretation function. If it is not explicitly excluded, or if we do not prove it omniscience free, we cannot be certain.

[6] If we accept the discussion of section 3.5.2, chapter 3, on *Computable Semantics*, the basic aim of semantics depends on the type of sentence being analysed. Montague Semantics' basic aim is to characterize declarative sentences, while situation semantics' is to characterize indicative sentences.

Since Hintikka's paper there have been several other attempts to deal with omniscience, in particular the kind originating from the propositional attitudes of belief and knowledge. Rantala[1982], Levesque[1984], Fagin and Halpern[1985], Konolige[1986] and Vardi[1986] have introduced several formal systems based on non-normal modal logics [7] with partial success. As was pointed out by Dieu[1989], in all of these approaches the attempt to avoid omniscience has resulted in having the syntactic and semantic levels so mixed that it is usually difficult to find their correct balance to provide an intuitively acceptable formalization. But possibly worst, as Dieu claims, is that they still suffer from omniscience at the reasoning level. Having discovered again potential omniscience in the approach of Situation Semantics, we believe that we need a more fundamental discussion on the causes of omniscience. On this respect we think that Barwise[1981] should have taken further his doubts on logical equivalence [8]. After reading Hintikka[1975] we also believe that more than ever the notions of *formality* and *logic* must be well understood if we want to make computational sense out of the discussions on omniscience.

For sentences involving perception, as we will see in section 4.5.3, we need formal logics of the kind used for instance for belief, i.e., of the non-normal kind referred to above. As we have seen, even if for propositional attitudes it was possible to live with omniscience, that is completely unacceptable for perception. This is the main reason why we mainly use perception sentences to discuss omniscience, and Barwise and Hintikka's views on the subject.

Overview

In this section we extend *B&PS* to take account of propositional attitudes and sentences involving perception and we claim that it is not in general free of omniscience, one of the main problems it was supposed to solve and which was claimed to be its main technical advantage over possible-world semantics. However, in some way which we will make precise, *B&PS* is much better prepared

[7] For non-normal modal logics see Hughes and Cresswell [1968], for instance the modal systems *S1*, *S2* and *S3*, which do not have the rule of necessitation.

[8] [...] After all, logical equivalence *is* logical equivalence, isn't it ? But is it ?' (Barwise[1981], pp. 388).

to face it. This, we believe, results first from its decision to use equivalence defined on statements and not on sentences, as Barwise[1981] asserted, but mainly from the decision of not quantifying on the mathematically defined interpretation of the language. This interpretation should be regarded as a given, non-quantifiable constant. If we want our treatment to be completely free of omniscience we would need to prove a strong condition on the interpretation function of the language, which is equivalent to a global restriction on inference rules to non-informative ones.

We then analyse Barwise's [1981] puzzle of logical equivalence. This puzzle of perception was the example Barwise used to reject possible-worlds semantics and a modal logic treatment of propositional attitudes and perception. We conclude that a semantics for perception can use a notion of possible worlds, by providing an example, and that Hintikka's decision to use the name of impossible worlds was misleading. We propose some basic characteristics that a formalization of perception should have. Given these basic characteristics, we investigate Hintikka's [1975] arguments, and we mainly conclude that the rejection of possible-worlds semantics and modal logic is not technically justifiable for a logic of perception. We trace that rejection to a philosophical decision by Hintikka of not accepting a positivistic analysis of what *formality* is all about. Our technical conclusions on omniscience in situation semantics increase our belief that we require exactly that criticized positivistic analysis, for treatments of perception and also for propositional attitudes.

4.5.1 Formal Semantics, Synonymy and Omniscience

Intuitively, sentences in a natural language can have the same meaning or can be used in the same way, even when the constituent words are different or ordered differently. This happens for instance when we change a word for a synonym [9] in a

[9] **Synonym. 1.** Strictly, a word having the same sense as another (in the same language); but more usu., either of any two or more words (in the same language) having the same general sense, but possessing each of them meanings which are not shared by the other or others, or having different shades of meaning appropriate to different contexts [...]. **2.** By extension: A name or expression which involves or implies a meaning properly or literally expressed by some other [...]. *The Shorter OED on*

sentence. We also hope that syntactically different sentences can be used to express the same idea, because they have the same meaning. We also expect that different expressions can be used together consistently or without contradictions. When providing a formal semantics for a language it is expected that notions of equivalence are defined to formalize those intuitive relations. Equivalence notions are also used as the basis to provide formal inference mechanisms. As Barwise and Perry say, *with the right semantics, the inferences take care of themselves* (*S&A*, pp. 183). In *MS* two sentences can have the same extension and have therefore the same extensional meaning or truth value, for a given context of use or index and logical interpretation. They can as well have the same meaning in any context of use and under a fixed logical interpretation. Finally they can be logically equivalent, if under any logically possible interpretation [10] and in any context of use they have the same extensional meaning. This is the definition for logically equivalent sentences used in *PTQ* and also used by Gallin[1975] to provide an axiomatization for a version of *IL*. In *B&PS* two sentences have the same meaning if their interpretation builds the same constraint on situations. But in *B&PS* there are other definitions that correspond to equality of meaning, that of statements, and which are supposed to model inference as being situated. Two statements are weakly equivalent, given a structure of situations, if they have the same boolean value and two statements are strongly equivalent if they have the same interpretation (4.4.3.4). A statement is defined as a sentence together with its context of use called the uttering situation.

Compositionality and Semantic Omniscience

The language *DFA-NL* which we defined in the previous sections of this chapter

---

Historical Principles, Third Edition 1944, Vol II, © OUP 1973, (1986 reprint), pp.1279 - 2672. Formally, the notion of strict synonymity is treated as equality of meaning; for instance as logical equivalence for sentences. The notion of non-strict synonymity seems to be captured for instance in S&A's notion of strong equivalence for statements, *utterances of sentences*. We believe it could be extended to utterances of words. In this case we would require that the interpretation of synonyms must be somehow different with different contexts, apart from words which are, or are imposed to be, strict synonyms. The interpretation u of the language for two different words is equal, exactly when (iff) they strictly are synonyms (when those two words have always the same meaning). If I allow strict synonymity between expressions in my formalization then, as we will see, I will have to accept all the consequences of that decision. In S&A the notion of synonymy is never explicitly referred to.

[10] An interpretation is logical if it respects an accepted constant meaning for the logical symbols.

does not treat attitude or perception sentences. In order to do so we would have to extend its syntax and semantics. We assume that this is possible. We also observe that situation semantics is guided by compositionality as in pp. 130 of *S&A*. We also note that attitude and perception sentences, either epistemically active or neutral, embed other sentences. *Epistemically neutral reports embed sentences, just as other attitudes do* (*S&A*, pp 180).

If we accept that sentences embed other sentences, our syntactic definition of the language will include a rule of the following form, where the domain P_S is the domain of syntactically defined sentences of the language, and P_X corresponds to some sentence taking syntactic category:

If $\alpha \in P_X$ and $\beta \in P_S$ then $\gamma = F_X(\alpha,\beta) \in P_S$, where
$F_X(\alpha,\beta) = \alpha\,\beta$.

If we accept compositionality of meaning, whatever the meaning objects may be, our semantic definition will include a rule of the following form, where u is the interpretation function being defined and S_x is the semantic composition function.

$$u[\![\gamma]\!] = S_x(u[\![\alpha]\!], u[\![\beta]\!]) .$$

It is clear that if in our semantic definition of u we have two sentences β_1 and β_2 for which $u[\![\beta_1]\!] = u[\![\beta_2]\!]$ we will have semantic omniscience [11]. So we conclude that any such compositional treatment with sentence embedding in general is bound to suffer from omniscience. The same argument is valid for syntactic expressions other than sentence. It may seem that the argument could be rejected on the grounds that equality between functions cannot be decided in general. This argument applies to both *MS* and *B&PS*, but it is obvious that we are not after the general case as the existence of some rule for which we can prove the equality is sufficient to show omniscience.

[11] In fact we need in general a less strong property on u , for example that $u[\![\beta_1]\!]$ *implies* $u[\![\beta_2]\!]$, where the type of *implies* depends on the type of the interpretation.

Example with Sentences

The syntactic system of *MS* defines a category of expressions called sentences. The treatment of propositional attitude and perception verbs is included when we allow a syntactic rule which combines a sentence as the object of such verb. Let us call this rule syntactic rule *n1* and let F_{n1} be the syntactic function that composes a Term-Perception Verb phrase, an element of P_{TPV}, such as ***John believes that***, with a sentence:

$SENT_{1.1}$ = ***John believes that Natassia was kissing Peter***
$SENT_1$ = ***Natassia was kissing Peter***

Syntactic Rule *n1*:
If $\alpha \in P_{TPV}$ and $\beta \in P_S$ then $F_{n1}(\alpha,\beta) \in P_S$, where
$F_{n1}(\alpha,\beta) = \alpha\ \beta$.

$SENT_{1.1}$ = F_{n1} (***John believes that*** , $SENT_1$)

Given that *MS* defines a compositional logical interpretation u, the interpretation of $SENT_{1.1}$ could be as follows, where S_{n1} is the semantic function corresponding to the syntactic rule *n1*,

Semantic Rule *n1*:
$u[\![\gamma]\!]$ = S_{n1} ($u[\![\alpha]\!], u[\![\beta]\!]$)

$u\ [\![SENT_{1.1}]\!]$ = S_{n1} ($u[\![$***John believes that***$]\!], u\ [\![SENT_1]\!]$)

A similar thing happens in *B&PS* for the syntactic and semantic definition for such expressions. The difference has to do with the types of the interpretation functions,

MS : $u[\![sent]\!] = \mathcal{I} \rightarrow \mathcal{B}$
B&PS : $u[\![sent]\!] = SIT \rightarrow SIT \rightarrow \mathcal{B}$

In *MS*, if we replace ***Natassia was kissing Peter*** by another logically equivalent sentence $SENT_2$, i.e., one which has the same truth value in any index, John has to

believe $SENT_2$ also. In *B&PS*, if we replace ***Natassia was kissing Peter*** by another semantically equivalent sentence like $SENT_2$, i.e., one that constrains situations in the same way, John also has to believe $SENT_2$. Because the compositional interpretations are defined in different ways, the sentences which in each case have to be believed are different, but unless we restrict the interpretations, the omniscience is present.

Example with Statements

This second example is an attempt to clarify what happens when we use statements instead of sentences. *S&A* claims that inference is based on statements, the utterance of indicative sentences (pp. 139 and followings of *S&A*). We will see how this affects omniscience.

As we have seen the boolean value of a statement is not defined compositionally. Therefore inferences arising from boolean equality or weak equivalence do not fall under the previous discussion, and so we may say that there is no logical omniscience if logical is identified with boolean. Strong equivalence however is compositionally defined, and therefore it seems to fall under the same conclusion.

Let us analyse the previous example, with $u_{1.1}$ being a situation which characterizes the utterance of $SENT_{1.1}$.

$$SENT_{1.1} = \textbf{\textit{John believes that Natassia was kissing Peter}}$$
$$SENT_1 = \textbf{\textit{Natassia was kissing Peter}}$$
$$STAT_{1.1} = (\textbf{\textit{John believes that Natassia was kissing Peter}}, u_{1.1})$$

In the grammar we will have as before,

$$SENT_{1.1} = F_{n1}(\textbf{\textit{John believes that}}, SENT_1)$$
$$u\,[\![SENT_{1.1}]\!] = S_{n1}(u[\![\textbf{\textit{John believes that}}]\!], u\,[\![SENT_1]\!])$$

And the interpretation of this statement is,

$$u\,[\![STAT_{1.1}]\!] = u\,[\![SENT_{1.1}]\!]\,u_{1.1}$$

Now suppose we have also

$$SENT_2 = \textit{Natassia was kissing Peter and John was walking or John was not walking}$$

This sentence in *MS* is logically equivalent to ***Natassia was kissing Peter.*** The question we ask in *B&PS* is what is the interpretation of the statement which results from uttering in the same situation the sentence $SENT_{1.2}$, i.e., $STAT_{1.2}$.

$$SENT_{1.2} = \textit{John believes that } SENT_2$$
$$STAT_{1.2} = (\, SENT_2 \,, u_{1.1} \,)$$

$$u\,[\![SENT_{1.1}]\!] = S_{nl}\,(u[\![\textit{John believes that}]\!],\, u[\![SENT_1]\!])$$
$$u\,[\![SENT_{1.2}]\!] = S_{nl}\,(u[\![\textit{John believes that}]\!],\, u[\![SENT_2]\!])$$

$$u\,[\![STAT_{1.1}]\!] = u\,[\![SENT_{1.1}]\!]\,u_{1.1} = S_{nl}\,(u[\![\textit{John believes that}]\!],\, u[\![SENT_1]\!])\,u_{1.1}$$

$$u\,[\![STAT_{1.2}]\!] = u\,[\![SENT_{1.2}]\!]\,u_{1.1} = S_{nl}\,(u[\![\textit{John believes that}]\!],\, u[\![SENT_2]\!])\,u_{1.1}$$

So if we want to avoid two statements being strongly equivalent we need to prove that for our definition of the interpretation $u[\![SENT_1]\!]$ and $u[\![SENT_2]\!]$ are different functions in general, i.e., are different constraints on situations in order for the meaning of the embedding sentence to be able to be also different. In this case $u[\![SENT_1]\!]$ should be different from $u[\![SENT_2]\!]$. The main conclusion is very simple: if we want to avoid omniscience in general, we must not quantify u, exactly as Barwise does, in the definition of equivalence. In the limit where our system involves attitudes, perception and other more explicitly syntactical sentence taking phrases, we need to be sure that for any two syntactically different sentences there will be a different semantic value. We hope this will became clear in the last section.

4.5.2 Barwise's Puzzle of Logical Equivalence

Barwise[1981] makes use of a puzzle as an example of the deficiencies with

possible worlds semantics. This puzzle raises questions about the technical use of worlds in the semantics of classical modal logic when formalizing the language of perception. The problems that Barwise finds leads him into Situation Semantics. As we believe in the general approach of compositional semantics and model-theory for modal logic as being particularly intuitive, we think that it should be able to explain Barwise's solutions and incorporate them. We start by summarizing the puzzle and the conclusions. Then we investigate the expressive power of the formal language used, first as a classical logic with extensional boolean meanings and second with intensional meanings on worlds. Finally we discuss the notion of logical equivalence and whether the puzzle creates a good case against the use of possible world semantics.

The Puzzle of 'Seeing'

> Can a man's death be attributed to a breakdown in the laws of modal logic? Brown is accused of murdering his mortal enemy Fred Smith. At Brown's trial, Smith's wife, Mary, testifies as follows: "Brown and I happened to enter the room at the same time, by different doors. Fred, facing my door, saw me enter. I saw Brown enter, but Fred did not see Brown enter." Brown who is pleading self-defence, needs to refute the claim that Fred did not see him enter the room. So he calls in an expert witness, a famous modal logician κ who "proves" that, by the very laws of logic, the situation could not possibly be as Mary Smith described! How?
>
> Barwise[1981], pp 381

The solution given by the expert witness uses the following translations into a formal logic:

m: ***Mary Smith***

f: ***Fred Smith***

b: ***Brown***

F(x): ***x entering by the door more or less in front of Fred***

B(x): ***x entering by the other door***

The testimony of Mary would then be formalized as:

(1) *m saw B(b)*

(2) *f saw F(m)*

(3) $\sim$ *(f saw B(b))*

It is assumed that the following principles (A) to (E) are accepted for a logic of perception.

(A) if *a sees* ϕ then ϕ

(B) if ϕ then $\sim$ *(f sees* $\sim\phi$*)*

(C) if *f sees* (ϕ *or* ψ) then *f sees* ϕ *or f sees* ψ

(D) if *f sees* (ϕ *and* ψ) then *f sees* ϕ *and f sees* ψ

According to Barwise the modal logician must also accept the following principle (E), for ϕ and ψ *logically equivalent*.

(E) if *f sees* ϕ then *f sees* ψ

It is also accepted that

(4) *F(m)* is *logically equivalent* to *(F(m) and B(b)) or (F(m) and* $\sim$ *B(b))* [12]

Then the expert witness is able to deduce from Mary's testimony that *f saw B(b)* and therefore arrived at a contradiction, that *f saw B(b)* and $\sim$ *(f saw B(b))*. From (1) and (A) deduce *B(b)*. From *B(b)* and (B) deduce $\sim$ *(f saw* $\sim$ *B(b))*. Whence, from (4), (C) and (D) we derive *f saw B(b)*, which contradicts Mary's testimony.

The conclusion Barwise takes is that *the old problem of logical omniscience* is responsible for the puzzle. According to Barwise, the modal logician is committed to the claim that if you know, believe or see ϕ, you must also know, believe or see any ψ logically equivalent to ϕ. Barwise cannot accept this view, particularly for perception verbs, and he takes from here his rejection of possible-worlds semantics and modal logic in the treatment of attitudes and perception. As we have said Hintikka[1975] provides technical support and mainly the philosophical support for

[12] We could reject this particular equivalence, as arising from strict implication. The use of some system which avoids the so-called "paradoxes of strict implication" could stop this particular equivalence (see for instance in Hughes and Cresswell [1968] system E). However, the argument rests on any logical equivalence, and any formal system should provide some equivalences.

Barwise's decision.

Before discussing Hintikka's analysis, we will look at the problem of choosing a logic to solve the problem. We investigate whether a formal logic has enough expressive power, i.e., whether its semantics is sufficiently fine-grained to represent the problem as we understand it [13]. In other words, we investigate which type of meanings satisfy compositionality in an intuitively acceptable way. We then provide some basic principles for a functional or compositional formal treatment of visual perception inspired in Hintikka's modal treatment of knowledge and belief and in particular for Barwise & Perry's 'primary ***sees that*** perception reports' (pp. 193 of *S&A*). As it happens with modal logics, what we can expect to be formally derived depends on the choice of axioms and inference rules. The analysis in *S&A* is very rich on this subject by providing what we believe are essentially axioms for different formal (modal and logical) treatments of attitudes and perception (pp. 195 of *S&A*).

Logic and the Choice of Formal Logic

We might be tempted to say that the overall meaning of a sentence such as ***Robert sees that something is happening***, does depend not on the meaning of that ***something is happening*** but on the external facts around Robert, or in other words that ***sees*** is not at all compositional. However, we need to have computational agents which are going to use a potentially infinity number of surface forms involving perception verbs. This forces us to find some compositional mode of representation and suitable meaning objects, if for nothing else than a question of economy.

We believe that an expert's treatment of the problem must start by deciding which logic to use. The logic used by Barwise is truth-functional classical logic, and so sentences have boolean meanings, i.e., true or false. It is trivial to show that with an extensional semantics we cannot model the following four sentences of natural language, because ***seeing*** is not intuitively truth-functional.

[13] See Dieu[1989] and Levesque[1984] for a discussion on fine and coarse-grained semantics.

s1: ***Robert sees Charles kissing Natassia***

s2: ***Robert sees Charles not kissing Natassia***

s3: ***Robert does not see Charles kissing Natassia***

s4: ***Robert does not see Charles not kissing Natassia***

They would be formalized as follows, where *j* stands for ***Robert*** and *p* for ***Charles Kissing Natassia***:

s1: j *sees* p

s2: j *sees* $\sim p$

s3: $\sim (j \text{ sees } p)$

s4: $\sim (j \text{ sees } \sim p)$

j: ***Robert***

p: ***Charles Kissing Natassia***

With an extensional semantics, with *p* taking only a true or a false value this is obviously not enough. We can provide a perfectly possible case where *s3* and *s4* are both true.

However, as Hintikka[1967] initially showed for attitudes, if we use with this formal language an intensional semantics on contexts of use, we could provide interpretations of ***sees*** which would increase the language's expressive power and allow a compositional semantics to be defined. In this case, ***sees*** would not be applied to an object with boolean meaning. This is consistent with our intuitions, as we do not see truth values. Still using the previous example, we could have the following four perfectly possible and different worlds, or contexts of use.

w1. Charles is Kissing Natassia.
John is not there so he does not see it.

w2. Charles and Natassia are at home and not kissing.
John is not there so he does not see it.

w3. Charles is Kissing Natassia.
John is there so he can see it and he sees it.

w4. Charles and Natassia are at home and not kissing.
John is there so he can see it and he sees it.

The interpretation of the six sentences could be for example as follows for each world.

	p	$\sim p$	$s1$	$s2$	$s3$	$s4$
w1.	t	f	f	f	t	t
w2.	f	t	f	f	t	t
w3.	t	f	t	f	f	t
w4.	t	f	f	t	t	f

With such worlds, we could define an intensional semantics with the meanings of p and j *sees* as functions on intensions of booleans as follows.

$u[\![p]\!] : \quad W \to \mathscr{B}$ for example,

$$u[\![p]\!] = \begin{bmatrix} w1 & t \\ w2 & f \\ w3 & t \\ w4 & t \end{bmatrix}$$

$u[\![j\ sees]\!] : \quad W \to (W \to \mathscr{B}) \to \mathscr{B}$ for example,

$$u[\![j\ sees]\!] = \begin{bmatrix} w1 & A \\ w2 & B \\ w3 & C \\ w4 & D \end{bmatrix}$$

where, for economy of space we only show A as an example:

$$A = \begin{bmatrix} \begin{bmatrix} w1 & t \\ w2 & t \\ w3 & t \\ w4 & t \end{bmatrix} & t \\ \cdots & \cdots \\ \begin{bmatrix} w1 & t \\ w2 & f \\ w3 & t \\ w4 & t \end{bmatrix} & f \\ \cdots & \cdots \\ \begin{bmatrix} w1 & f \\ w2 & t \\ w3 & f \\ w4 & f \end{bmatrix} & f \\ \cdots & \cdots \\ \begin{bmatrix} w1 & f \\ w2 & f \\ w3 & f \\ w4 & f \end{bmatrix} & t \end{bmatrix}$$

Then we would have that,

$$u[\![s1]\!] = u[\![j\ sees]\!](u[\![p]\!]) = \begin{bmatrix} w1 & f \\ w2 & f \\ w3 & t \\ w4 & f \end{bmatrix}$$

$$u[\![s2]\!] = u[\![j\ sees]\!](u[\![\sim p]\!]) = \begin{bmatrix} w1 & f \\ w2 & f \\ w3 & f \\ w4 & t \end{bmatrix}$$

The first conclusion we take is that once we provide a formal representation for worlds as for instance contexts of use, the formal language seems to be able to be used with a compositional intensional interpretation. Obviously, we do not claim that one example is sufficient, but it seems to us that a normal world based semantics is just offering a sufficient fine-grained structure for perception. The practical question we are not yet answering is what are these normal worlds. If we accept that perception needs an intuitively natural and normal notion of world or context of use for its semantics, it seems to us completely unnatural to bring over Hintikka's [1975] suggestions on propositional attitudes of using impossible possible worlds. It may be that it is just a question of words, but we believe that it is more than that.

4.5.3 On a Formal Logic for Perception

Objectives

The objective of this section is not as the title might suggest to develop a logic of perception but to discuss the properties of such formalization, to enable the discussion of the following section on omniscience. Our main objective is to expose some problems with Hintikka's treatment of knowledge and belief, of sentences involving the phrases ***knows that*** and ***believes that***, through the study of the less epistemically connotated perception sentences involving the phrase ***sees that***. For a question of simplification of notation we will use the following abbreviations:

$K_a p$ for *a knows that p*

$B_a p$ for *a believes that p*

$S_a p$ for *a sees that p*

Other Propositional Words and Phrases

One of the characteristics of these phrases K_a, B_a and also of a phrase like ***It is necessary that*** is that they are not truth functional on the proposition following them. We will call such phrases propositional phrases. We list in some partial order of decreasing 'logical power', sentences with propositional phrases.

It is necessary that p	*It is necessary that 1+1=2. It is necessary that 7,536,429+8,897,698=16,434,127.*
a knows that p	*Natassia knows that the morning star is the evening star.*
a believes that p	*Natassia believes that the morning star is the evening star.*
a sees that p	*Charles sees that Robert is kissing Natassia.*
It is obvious that p	*It is obvious that 1+1=2. It is obvious that 7,536,429+8,897,698=16,434,127.*

Hintikka's Treatment of Propositional Attitudes

Given the success of Kripke's possible worlds semantics for modal logics involving the formalization of necessity and possibility, Hintikka decided to take the same approach into the propositional attitudes. Hintikka[1967] for instance, uses for their treatment operators interpreted with an intensional semantics. For the knowledge operator he introduces an accessibility relation on epistemically possible worlds, the *epistemic a-alternative relation*, to define the interpretation:

$$u[\![K_a p]\!]w \quad \text{iff} \quad \forall w'.\ epist_alt(a, w, w') \rightarrow u[\![p]\!]w'.$$

In principle this will allow the operator K_a to be intensionally truth-functional. Given that we want such semantics for S_a we may think of doing the same thing. We could

justify this step on a basic identification of ***seeing*** with ***knowing***. For instance Martin-Löf[1983] makes detailed observations on how often we use ***I see that*** instead of saying ***I know that***. This is basically the main justification we have to write the following formula, where *vis_alt* can be read, paraphrasing Hintikka, as the *visual a-alternatives* relation:

$$u[\![S_a p]\!]w \quad \text{iff} \quad \forall w'.\ vis_alt(a, w, w') \rightarrow u[\![p]\!]w' .$$

We believe it is much easier to discuss perception than knowledge. It may be simpler to ask the question whether this later definition makes any sense, and then relate the conclusions to Hintikka's formalization of knowledge. As we have seen omniscience is definitely not acceptable in a logic of perception.

Axiomatization

The main technical problem with the definition above is that it is equivalent to accept a form of the rule of necessitation. We think of a formal system, for instance as in Hughes and Creswell[1968], presented by a set of axioms *AX* and rules of inference *RI* where the turnstile $\vdash$ expresses formal inference. Then, with a logical notion of validity as truth in all possible-worlds, we would have that if we deduce p we need to deduce $S_a p$ as well. If $\vdash p$ means $\forall w.\ u[\![p]\!]w=t$ then from the definition above we also must have that $\forall w.\ u[\![S_a p]\!]w=t$ or $\vdash S_a p$, for short that the following inference rule, similar in form to necessitation, is inescapable: $p \vdash S_a p$.

It is at this point that we will start diverging from Hintikka and in some respect from Barwise. We do not take this as a failure of possible worlds semantics, neither of modal logic, but a problem with the understanding of formality, logic and validity. We know that modal logic is able to cope with this situation. It is well known that we can have modal formal systems for which the rule of necessitation is not valid, for instance the modal systems *S1*, *S2* and *S3* described by Hughes and Creswell[1968]. They also present a Kripkian style semantics for *S2* and *S3*, but fail to find a satisfactory semantics for *S1*. More recently, Dieu[1989] reviews a few formal systems for belief for which satisfactory inference mechanisms can be built but still having unsatisfactory semantics, and which we particularly find

counterintuitive by the use of notions of non-normal or impossible possible worlds.

The main formal difference between such systems and a logic for perception would rest on the choice of axioms and rules of inference. If we take a propositional version of Barwise and Perry formalization of visual perception [14], we would require the following axioms together with the normal propositional calculus:

A1. $S_a p \rightarrow p$

A2. $S_a \sim p \rightarrow \sim S_a p$

A3. $p \rightarrow \sim S_a \sim p$

A4. $S_a (p \text{ and } q) \rightarrow S_a p \text{ and } S_a q$

A5. $S_a (p \text{ or } q) \rightarrow S_a p \text{ or } S_a q$

And for instance we do not want any of the following rules and axioms:

- $p \vdash S_a p$
- $p \rightarrow q \vdash S_a p \rightarrow S_a q$
- $(S_a p \text{ and } S_a (p \rightarrow q)) \rightarrow S_a q$

We are not assuming this to be a complete presentation for the formal logic of *seeing* but we believe it shows that such is quite feasible. It is obvious that this system would not bring about any puzzling results for the example of section 4.5.2.

Semantics

It is not possible to provide a semantics for perception before discussing in general the problem of meaning and validity for propositional phrases, in particular for perception sentences. However before such discussion in the next section, we believe that we need to follow our intuitions of what is that we *see*. We think that we need in the model a set of agents Ag who can see, and for each agent the equivalent of a visual 'information system' which models the states of its visual input. A boolean interpretation for sentences involving seeing would then be given by an agent dependent function that would have as argument the information system

[14] Primary S_a perception reports.

and the proposition for which the question of being seen was asked. One of the characteristics of those functions is that they must be strictly computable, i.e., its implementation must be a program which terminates. If we are interested in declarative sentences $S_a p$ [15], that function needs to have a three value result domain, respectively for the cases when the visual database contains p, contains $\sim p$, and the case where it contains neither $\sim p$ nor p.

4.5.4 On Formal Inference and Interpretation for Perception

Formal Representations and Meaning

Hodges[1977] presents the following example to justify the necessity of an alternative to intensions, as the meaning objects which would satisfy compositionality:

> For example the two following sentences are necessary truths, and hence they have the same intension:
>
> 1+1=2.
>
> 7,536,429+8,897,698=16,434,127. (44.2)
>
> Nevertheless one of the following sentences is true and the other is false:
>
> It is obvious that 1+1=2.
>
> It is obvious that 7,536,429+8,897,698=16,434,127. (44.3)
>
> We see at once that some other feature of the sentences (44.2) besides their intension must be weaving an influence.
>
> Hodges[1977], pp. 252

The formalization of mathematics, for instance integer arithmetic, is concerned with creating a formal system for the automatic, or objective, process of problem solving in integer arithmetic. We would assume that typographical or psychological characteristics are excluded from such formalization, for instance questions involving the font we use to express the arithmetic operators and whether they are known to the readers. We believe therefore that it is not satisfactory to conclude that the example above is a good reason to replace intensions.

---

[15] See the distinction made between for instance declarative and indicative sentences in section 3.5.2, chapter 3.

The real problem is that the mathematical and logical formalization we start with is an objective formalization for mathematics, and the notion of logical equivalence is objectively formalized in such way that it is acceptable as corresponding to our intuitions of mathematical equivalence.

Hintikka's extensions to knowledge and belief of the formalization of mathematical or logical necessity has, we believe, the same fundamental problem Hodges is facing above, which in the end does not eliminate omniscience. We believe that this becames very clear in Hintikka[1975]. There we find two uses of logic. The first one has to do with the definitions of truth-functional classical logical symbols in formal languages, such as mathematical equality and boolean connectives and objective necessity. Given a language with boolean interpretable primitives, truth functional logical connectives and expressions made out of them, such as first order logic, the semantics is given using an interpretation, which is logical if the standard truth functional boolean, logical and objective operations are respected. We have defined the concepts of equivalent sentences and the concept of logically equivalent sentences. The first concept is relative to *some* logical interpretation, the second to *any* logical interpretation. When the logic is extended with modal operators and temporal operators, which are not truth-functional, notions of intensions on logically-possible or objective world and time, accessibility and ordering have to be introduced to preserve truth functionality and to achieve a proper notion of equivalence and logical equivalence [16]. The theorems of such logics depend on the formal axiomatization and inference rules used, which defines the logically equivalent sentences.

The extension of formalization to other non truth functional operators, such as the ones necessary to formalize our use of verbs like *believe* or *see* requires a proper formalization process, which is intuitively valid *a priori*. Such formal deductive systems would need to square with our intuitive judgements of what can be absolutely taken as *believing-ly* or *seeing-ly equivalent* and never the opposite. The

[16] For instance we would not dream of rejecting intensions when extending the formal semantics of necessity to an objective formal semantics of time.

semantics of such formalization would be restricted to possible *believing-ly* or *seeing-ly* interpretations. And here comes the second sense of logic in Hintikka's work, his belief that the use of *logically equivalence* must incorporate *believing-ly* or *seeing-ly equivalences*. Carrying this process for all non truth functional words in the language would provide a notion of absolute logical equivalence.

Hintikka wants an intuitively valid formalization of knowledge and belief but at the same time he wants to keep the formal framework of objective or mathematical logic, in which the following inference rules are accepted:

$$p \rightarrow q \quad \vdash K_a p \rightarrow K_a q$$
$$p \rightarrow q \quad \vdash B_a p \rightarrow B_a q$$

His main reason to keep these rules is attributed explicitly to his rejection of the *positivistic doctrine of the non-informative (tautological) character of logical truths.* We believe that this is exactly Martin-Löf's [1983] claim on formality, and in particular on knowledge.

> There is absolutely no question of a judgement being evident in itself, independently of us and of our cognitive activity. That would be just as absurd as to speak of a judgement as being known, not by somebody, you or me, but in itself. To be evident is to be evident to somebody, as inevitably as to be known is to be known by somebody.
>
> The notion of formal proof [...] has been arrived at by formalistically interpreting what you mean by an immediate inference, by forgetting about the difference between a judgement and a proposition, and, finally, by interpreting the notion of proposition formalistically, that is by replacing it by the notion of formula.
>
> Martin-Löf[1983], pp. 21, 30

However, such a 'positivistic doctrine' of formality does not stop us from formalizing knowledge, belief perception and other propositional aspects of language. Its formal semantics just needs to be intuitively valid, in the sense for instance that it takes into account, if necessary, the correct level of for instance typographical or psychological characteristics. But it is obvious that the necessary consideration of typographical characteristics, for instance in sentences like ***There are five e's on this sentence*** or in Quine's sentences with quotations, requires that

the intensional interpretation should have in those cases an explicit typographical or 'syntactical' semantics [17]. Moreover, and referring back to section 4.5.1, we conclude that in general for a real semantic interpretation (as u) we must always be able to totally distinguish between syntactically or typographically different sentences. The great advantage of Situation Semantics on the subject of omniscience, as we have said, is that we may be able for some language, and if we want, make sure that there is no strict synonymity, while being able to have local or situated reasoning, based on local notions of equivalence.

> [...] Thus deductive inference is, like truth, *objective*, and even *absolute*. Objectivity does not mean, of course, that we can always ascertain whether or not a given statement is true. Nor can we always ascertain whether a given inference is valid. If we agree to use the term "true" only in the objective sense, then there are many statements which we can *prove* to be true; yet *we cannot have a general criterion of truth*. If we had such criterion, we would be omniscient, at least potentially, which we are not. According to the work of Gödel and Tarski, we cannot even have a general criterion of truth for arithmetical statements, although we can of course describe infinite sets of arithmetical statements which are true. In the same way, we may agree to use the term "valid inference" in the objective sense, in which case we can prove of many inferences that they are valid (that is, they unfailingly transmit truth); yet we have no general criterion of validity—not even if we confine ourselves to purely arithmetical statements.
>
> Popper[1974], pp. 143

4.6 CONCLUSIONS

We have defined the syntax and semantics for the language *Determiner-Free Aliass* of Barwise & Perry using a compositional definition in the same style of Montague semantics. The interpretation of a sentence is defined as a constraint on situations and required to be computationally expressed. A situation is a finite collection of facts about the world. The notion of statement, the utterance of an indicative sentence, is defined as a pair of an uttering situation and a sentence. The notion of equality of meaning for statements is modeled or formalized with the concept of strong equivalence between statements. We showed that in general such semantics

[17] We would like to make it clear that we are not proposing a system were we would end up with no capability to find equivalences between expressions of different syntactic form. We are just claiming that at the global level, of a real or total or complete interpretation we must take into account that there are no two objects with different forms which are totally equal, or referring back to footnote 9, that really there are no strict synonyms.

can suffer from the problem of omniscience. However, if we restrict ourselves to real and computable interpretations, where we avoid strict synonymity and quantification of the interpretation on the definition of equivalence, we may avoid that problem. We believe that this solution could be used both for situation semantics and for possible-worlds semantics, or alternatively that this would be used as a distinguishing characteristic for the former semantics. We also commented on Hintikka's formal semantics for propositional attitudes, and we noticed that his attempt to make the use of knowledge mathematically objective, always creates omniscience.

> The Greeks in general attached more importance to deduction as a source of knowledge than modern philosophers do. In this respect, Aristotle was less at fault than Plato; he repeatedly admitted the importance of induction, and he devoted considerable attention to the question: how do we know the first premisses from which deduction must start? [...] [An induction] has less cogency than a deduction, and yields only a probability, not certainty; but on the other hand it gives *new* knowledge, which deduction does not. All the important inferences outside logic and pure mathematics are inductive, not deductive; the only exceptions are law and theology, each of which derives its first principles from an unquestionable text, viz. the statute books or the scriptures.
>
> Russell[1946], pp. 209

> 'How do we jump from an observation statement to a *good* theory?' [...] the answer is: by jumping first to *any* theory and then testing it, to find whether it is good or not; i.e. by repeatedly applying the critical method, eliminating many bad theories, and inventing many new ones. Not everybody is able to do this; but there is no other way.
>
> Popper[1953], pp. 55

CHAPTER 5

CONCLUSIONS

Using the framework of denotational semantics we have modeled two of the major paradigms of mathematical study of natural language. This offers a computationally meaningful analysis of natural language, which we believe is required if we want to have a constructive framework where different semantical approaches can be compared.

We have shown how the computational methodology of denotational semantics forced us to simplify the definitions of meaning in Montague semantics and

Situation Semantics. This constructive methodology, which is a characteristic of the main schools of analytic philosophy, requires a study based on the finite properties of objects and therefore of language meaning. It is also consistent with the scientific method in cognitive studies which demands a study of language based on the principle that human cognitive capacity needs a finite basis.

5.1 ON FORMAL ANALYSIS OF NATURAL LANGUAGE

Montague's Operators and the Postulates

Our constructive study forces an understanding of the cap-cup operators in terms of its functional or computational properties. This understanding allows a re-evaluation of their role and imposes a simplification in the treatment of *PTQ*, achieved through their elimination and alteration of the type structure and translation functions. Our interpretation of the operators forces the meaning postulates to be formally expressed in very direct relation with their natural English motivation. However, we also conclude that the simplification can be extended by not using meaning postulates at all. A direct advantage of their elimination is that in the treatment we avoid using higher order quantification. Although there are fundamental reasons, such as economy, discussed in chapter three, and realism, mainly discussed in chapter four, which make the decision of quantification on logical interpretations to be looked at suspiciously, avoiding higher order quantification is computationally advisable. Moreover computing with higher order quantification is not usually acceptable for modeling human cognitive processes.

Computability of Grammar

In chapter three we propose the notion of computable grammars to define natural language approximations, based on computational restrictions to the theory of Montague's *UG*. Such grammars define a syntactic and semantic approximation, as two computable functions which capture our intuitions of a certain form and usage of natural language. We have used such notion of grammar to define two formal approximations to Montague's intensional English, satisfying the requirements

which motivated his meaning postulates. However, unlike Montague's formalization, our formal approximation would not to require higher-order quantification. The definition in chapter four of *ALIASS* again falls into the category of such computable grammars, although we have chosen to present it in Montague's style of notation.

Propositional Attitudes, Perception and Formal Inference

In chapter four we have shown that Situation Semantics suffers in general from omniscience, when *ALIASS* is extended to cover propositional attitudes. Our positivistic analysis of attitudes and perception makes us believe that we can use formal modal logics and possible worlds semantics in their formal logical treatment. However we have to reject Hintikka's view which admits inference rules deriving new propositional information. This is equivalent to constrain a compositional semantics to real interpretations, which cannot be quantified and which we have to guarantee to be omniscient free.

Language Use instead of Meaning and Grammar

We used arguments mainly based on Montague and Wittgenstein to propose that the study of Natural Language or of Grammar can be divided into Syntax and Semantics. Syntax as the study of the forms, and semantics as the study of their uses. We believe that the word 'use' should replace 'meaning' because it has finitistic rather than infinitistic connotations. This conclusion is based also on Hintikka's [1976] analysis of Wittgenstein's *PI* theory of use and his non-positivistic view on the finitism of the later. This division is used to define computable aproximations to a 'complete' grammar of natural language.

With this understanding of Grammar, we are supporting the view that everything that affects language is part of the study of grammar. Accepting the view of analytic philosophy, that all problems are in some way problems of language, grammar inevitably will have to face any problem which can be expressed using language.

5.2 SUMMARY OF RESULTS

Our critical analysis of treatments of natural language has produced some hitherto unperceived facts. Amongst these are the role of the cap cup operators in Montague semantics, and its relation with the higher order quantification introduced by the meaning postulates. It has also uncovered the possibility of omniscience in Situation Semantics, and helped in defining a strategy to avoid it.

The overall use of the functional notation in the formal treatment of natural language has proved to be extremely helpful. It allowed a clarification on the double formalization used by Montague in his compositional treatment of intentionality. For this clarification it has also contributed the explicit naming of interpretation functions, instead of the use of square parentheses on their arguments. We believe that it was the overall notational clarity which led us into the elimination of the cap-cup operators and also to the elimination of meaning postulates and with them the higher order quantification from the formal treatment of natural language proposed by Montague.

The compositional presentation of Situation Semantics' analysis of natural language and the observation of the conditions in which it suffers from omniscience are, we believe new. Our critical analysis of the treatment of propositional attitudes brings us to the conclusion that we need realistic interpretations, mainly computable functions which cannot be quantified, for modelling natural language in an intuitively acceptable way.

We believe that our definition and use of a uniform functional description of computational grammar, to model approximations to natural language, can be used to bring into the formal logical analysis of natural language new alternative ways of discussing and explaining language learnability. Using such models, we believe that any logical formal approximation, to be cognitively acceptable, should always be computationally characterized as a grammar or a program which terminates. Therefore our semantic interpretation functions, for instance, should be guaranteed to terminate by proving that such happens.

5.3 RECOMMENDATIONS FOR FURTHER STUDY

Deduction, Induction and Quantification

The view that logical inferences can be not only deductive but also inductive, as pointed out by Russell in the citation above, was justified by the acceptance of an independent logical principle of induction, which would solve Hume's problem of induction. This view has been extensively criticized by Popper, see for instance Poper[1953], on the basis that scientific induction does not follow from logical necessity, and should not be referred to as a form of inference. A scientific thesis, a universal law, is a 'bold' conjecture, which can only be proved to be false, and never be inferred. As we have seen, our formalization of quantification is an attempt to formally reduce this analysis by using a domain with a truth value less defined than a false value. This treatment needs more investigation, for instance, with respect to the basic boolean functions and to the compositional semantics of propositional attitudes. We should ask for instance when are we using such domains, and more precisely what happens with the semantics of embedded sentences.

Another of the questions posed by such domains concerns the form of the logic being utilized, which might be answered possibly by using the methods proposed in Abramsky's [1987a] 'Domain Theory in Logical Form'. One trial answer in need of justification is that that logic could be the one of the systems of non-monotonic reasoning. Whether the formal derivations of non-monotonic reasoning systems are inferences following from logical necessity, or have the same cogency of Russell's inductions, is also a question which we think needs to be understood, under the light of Popper's claims on induction. In our theory of computable approximations to grammars of natural language, each new approximation is very much a 'bold' conjecture about grammar. One open question is whether non-monotonic logic could be used to get new approximations, or whether the process is outside the application of such systems. The main problem can be stated as how far is the logic of non-monotonic systems, the logic of scientific discovery.

Logical Grammatic Approximations and Connectionism

One of the methodological advantages of our treatment of computable grammars, as finite approximations, is that it requires the use of effective computational procedures, and effective support tools. For instance when writing a syntactic definition we should be using a formalism which makes the generation of efficient parsers simple and automatic. Although this treatment needs a much improved presentation, in particular we would like to investigate how to solve automatically the type equations arising from the system of equations of semantic definitions.

We would also like to investigate the grammatic formalization of propositional attitudes and perception at the same time as investigating how those forms of language are treated by connectionist machines. For instance, by asking the question of what do we need to do to model attitude and perception verbs with a connectionist machine, we might learn how to write a grammatic approximation for expressions of propositional and perception attitudes. This method may as well clarify some of the questions which are open on the semantics of non-normal modal logics we referred to in chapter four.

Applications to other Syntactic and Semantic Objects

As in computer programs there are expressions such as declarations and commands which are combined to form larger syntactic expressions and semantic functions, in natural language sentences such as declarative or interrogative are combined in texts to form larger syntactic groups such as paragraphs and chapters, with roles varying from introductions and context definition to explanations, examples and conclusions. We all have learnt, more or less informally, how to organize our writings and how to analyse our readings at this higher level. We believe that formal processing of texts, other than sentences requires the development of formal grammars, for instance computable ones, for syntactic and semantic objects of which sentences, in declarative, interrogative, indicative and imperative use are just a few building blocks.

REFERENCES

ABRAMSKY, SAMSON

1987a **Domain Theory in Logical Form.** Dept. of Computing, Imperial College, University of London (October 1987), 76 pp. Submitted to *Annals of Pure and Applied Logic.*

1987b **Semantics of Programming Languages.** Lecture Notes for the FAIT. Dept. of Computing, Imperial College, University of London (1987).

ACZEL, PETER

1980 **Frege Structures and the Notions of Proposition, Truth and Set.** In: *The Kleene Symposium.* J. Barwise, H. J. Keisler and K. Kunen, eds., North-Holland (1980), pp. 31-59.

ALLEN, JAMES F.

1984 **Towards a General Theory of Action and Time.** In: *Artificial Intelligence*, Vol. 23 (1984), pp. 123-154.

1987 **Natural Language Understanding.** Benjamin Cummings, Menlo Park, California (1987), xv+574 pp.

ALSTON, WILLIAM P.

1967 **Meaning.** In: *Encyclopedia of Philosophy.* Vol. 5. P. Edwards, ed., MacMillan, New York (1967), pp. 233-241.

AUGUSTINE, ST.

397 **Confessiones.** Translated by R. S. Pine-Coffin under the title **Confessions.** Penguin Books, Harmondsworth (1986), 346 pp.

AUSTIN, J. L.

1950 **Truth.** In: *Philosophical Papers*, J. O. Urmson and G. J. Warnock, eds., Oxford (1979), pp. 117-133.

BARENDREGT, H.P.

1981 **The Lambda Calculus: Its Syntax and Semantics.** North-Holland Oxford (1981), xi+615 pp.

BARWISE, JON

1981 **Scenes and Other Situations.** In: *The Journal of Philosophy.* Vol. 78, No.7 (July 1981), pp. 369-397.

1984 **Situations, Sets and the Axiom of Foundation.** In: *Logic Colloquium'84.* J. B. Paris, A. J. Wilkie and G. M. Wilmers, eds., North-Holland (1984), pp. 21-36.

1985 **A Model for the Treatment of Anaphora in Situation Semantics.** Center for the Study of Language and Information. Stanford University. Informal Notes No. IN-CSLI-85-1 (May 1985), 24 pp.

1986 **Unburdening the Language of Thought.** In: *Mind and Language.* Center for the Study of Language and Information. Stanford University (1986), pp. 82-96 .

BARWISE, JON AND COOPER, R.

1981 **Generalized Quantifiers and Natural Language.** In: *Linguistics and Philosophy.* Vol. 4, D. Reidel, Dordrecht, Holland (1981), pp. 159-219.

BARWISE, JON AND ETCHMENDY, JOHN

1987 **The Liar. An Essay in Truth and Circularity.** Oxford University Press (1987), xii+185 pp.

BARWISE, JON AND PERRY, JOHN

1983 **Situations and Attitudes.** MIT Press, Cambridge, Mass. (1986), xxii+352 pp.

VAN BENTHEM, JOHAN

1984 **Correspondence Theory.** In: *Handbook of Philosophical Logic.* Vol. 2, Dov Gabbay and F. Guenthner, eds., D. Reidel, Dordrecht, Holland (1984), pp. 167-247.

BOOLE, GEORGE

1854 **An Investigation of the Laws of Thought, on which are Founded the Mathematical Theories of Logic and Probabilities.** Walton and Maberly, London (1854), iv+424.

BULL, ROBERT AND SEGERBERG, KRISTER

1984 **Basic Modal Logic.** In: *Handbook of Philosophical Logic.* Vol. 2, Dov Gabbay and F. Guenthner, eds., D. Reidel, Dordrecht, Holland (1984), pp. 1-88.

CARNAP, RUDOLF

1946 **Introduction to Semantics.**

CHANG, CHIN-LIANG AND LEE, RICHARD CHAR-TUNG

1973 **Symbolic Logic and Mechanical Theorem Proving.** Academic Press(1973).

CHOMSKY, NOAM

1957 **Syntactic Structures.** Mouton, s'Gravenham (1957).

1965 **Aspects of the Theory of Syntax.** MIT Press, Cambridge, Mass. (1970), vii+251 pp.

CHURCH, ALONZO

1936 **Mathematical Logic.** Princeton University Press (1936).

1940 **A Formulation of the Simple Theory of Types.** In: *Journal of Symbolic Logic.* Vol. 5, Association for Symbolic Logic.(1940), pp. 56-68.

1941 **The Calculi of Lambda-Conversion.** Princeton University Press.(1941).

COLMERAUER, A.

1976 **Metamorphosis Grammars.** In: *Natural Language Communication with Computers, Lecture Notes in Computer Science,* Vol. 63, L. Bolc., ed., Springer-Verlag, Berlin (1978), pp. 133-189.

COOPER, ROBIN

1984 **Sentence Negation in Situation Semantics.** University of Wisconsin—Madison. Presented at the *Twentieth Regional Meeting of the Chicago Linguistic Society* (1984), 8 pp.

1987 **Introduction to Situation Semantics.** Draft of Chapters 1, 2 and 3. University of Edinburgh (June 1987), 48+10+22 pp.

CUNHA, JOÃO AND CUNNINGHAM, R. J.

1987 **Some Comments on Montague's Treatment of English.** Dept. of Computing, Imperial College, University of London (July 1987), 19 pp. Presented at the *Conference on Logic and Linguistics, Stanford, July 87.*

1988 **The Intensional Operators on Montague's Intensional Semantics.** Dept. of Computing, Imperial College, University of London (July 1987), 19 pp.

CUNHA, JOÃO AND GERA, MICHAEL

198? **Wittgenstein Meaning as Use and PDP Models for Learning.** Dept. of Computing, Imperial College, University of London (198?). *Forthcoming.*

CUNHA, JOÃO AND PITT, JEREMY V

1989 **Using Context-Free Grammars for Natural Language Analysis.** Dept. of Computing, Imperial College, University of London (1989). *Forthcoming.*

CUTLAND, NIGEL J.

1980 **Computability. An Introduction to Recursive Function Theory.** Cambridge University Press (1980), x+251 pp.

DAMAS, LUIS M. M.

1984 **Type Assignment in Programming Languages.** PhD Thesis, Report CST-33-85, Department of Computer Science, University of Edinburgh (1985), 134 pp.

DAVIDSON, D.

1964 **Theories of Meaning and Learnable Languages.** In: *Logic, Methodology and Philosophy of Science (Proceedings of the 1964 International Congress for)* Jerusalem, August 1964, Yehoshua Bar-Hillel, ed., North-Holland, Amsterdam (1965), pp. 383-394.

1967 **Truth and Meaning.** In: *Synthese*, Vol 17, D. Reidel (1967), pp. 304-323.

DEVITT, MICHAEL AND STERELNY, KIM

1987 **Language and Reality. An Introduction to the Philosophy of Language.** Basil Blackwell, Oxford (1987), xii+274 pp.

DIEU, HENDRIK

1989 **On Belief And Omniscience.** Research Report, Dept. of Computing, Imperial College, University of London (March 1989), 72 pp.

DOWTY, DAVID R.

1979 **Word Meaning and Montague Grammar.** The Semantics of Verbs and Times in Generative Semantics and Montague's *PTQ*: D. Reidel, Dordrecht, Holland (1979), xi+415 pp.

DOWTY, DAVID R., WALL, ROBERT E. AND PETERS, STANLEY

1981 **Introduction to Montague Semantics.** In: *Studies in Linguistics and Philosophy*, Vol 11. D. Reidel, Dordrecht, Holland (1981), xi+313 pp.

DUMMETT, MICHAEL

1977 **Elements of Intuitionism.** Clarendon, Oxford University Press (1985), xii+467 pp.

ETCHEMENDY, JOHN

1988 **Models, Semantics and Logical Truth.** In: *Linguistics and Philosophy*, Vol 11, No. 1, Kluwer Academic Publishers (February 1988), pp. 91-106.

FAGIN, R. AND HALPERN, J. Y.

1985 **Belief, Awareness, and Limited Reasoning.** In: *Proceedings of the Nineth International Joint Conference on Artificial Intelligence*, Los Angeles (August 1985), pp. 491-501.

FITTING, MELVIN

1983 **Proof Methods for Modal and Intuitionistic Logics.** D. Reidel (1983), viii+556 pp.

FODOR, J. A.

1986 **Information and Association.** In: *Nôtre Dame Journal of Formal Logic.* Vol 27, (1986), pp. 307-323.

FONG, S., PITT, JEREMY V. AND CUNNINGHAM, R. J.

1984 **Parsing English with Traditional Grammar.** Dept. of Computing, ICST, London (December 1984), 29 pp.

FORTUNE, STEVEN, LEIVANT, DANIEL AND O'DONNELL, MICHAEL

1984 **The Expressiveness of Simple and Second-Order Type Structures.** Journal of the ACM, Vol 30, No.1 (January 1983), pp. 151-185.

FREGE, GOTTLOB

1879 **Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens.** Translated by P. Geach under the title **Begriffsschrift, a Formalized Language of Pure Thought modeled upon the Language of Arithmetic.** In: *Translations from the Philosophical Writings of Gottlob Frege.* P. Geach and M. Black, Oxford (1952), pp. 1-20.

1892 **Über Sinn und Bedeutung.** In: *Zeitschrift für Philosophie und Philosophische Kritik* 100:25-50 (1892). Translated by M. Black under the title **On Sense and Reference.** In: *Translations from the Philosophical Writings of Gottlob Frege.* P. Geach and M. Black, Oxford (1952), pp. 56-78.

GABBAY, DOV M. AND ROHRER, CHRISTIAN

1982 **Non-Monotonic Reasoning – Concessive Clauses and Tensed Conditionals in English.** 1st Draft, Research Report, University of Stuttgart (1982), 18 pp.

GABBAY, DOV M.

1982 **Intuitionistic Basis for Non-Monotonic Logic.** Sixth Conference in Automated Deduction, New York. In: *Lecture Notes in Computer Science*, Vol. 138, Springer-Verlag (1982), pp. 260-273.

1985 **Temporal Logic and Computer Science.** Lecture Notes for the FAIT. Dept. of Computing, Imperial College, University of London (1985).

1987 **Modal and Temporal Logic Programming.** In: *Temporal Logics and their Applications.* Academic Press (1987), pp. 197-237.

1988 **Changeability in Montague Semantics.** Dept. of Computing, Imperial College, London (1988). *Personal Communication.*

GALLIN, DANIEL

1975 **Intensional and Higher-Order Modal Logic.** North-Holland (1975), vii+148 pp.

GERA, MICHAEL

1988 **Neural Networks.** Dept. of Computing, Imperial College, London (1988). *Personal Communication.*

GÖDEL, KURT

1931 **On Formally Undecidable Propositions of *Principia Mathematica* and Related Systems.** In: *From Frege to Gödel. A Source Book in Mathematical Logic,1879-1931.* Jean van Heijenoort, ed., Harvard University Press, Cambridge, Mass. (1967), pp. 596-617.

GONÇALVES, MARIA JOÃO MOREIRA

1988 **Routines in Learning.** In: *International Summer School in Developmental Language Disorders*, National Hospitals of Speech Sciences, London (1-5 August 1988). *Personal Communication.*

GORDON, MICHAEL J. C.

1979 **The Denotational Description of Programming Languages.** Springer-Verlag (1979), viii+159 pp.

GRICE, H. P.

1975 **Logic and Conversation.** In: *Syntax and Semantics.* Vol 3. P. Cole and J. Morgan, Academic Press, New York (1975), pp. 41-58.

HALLIDAY, M. A. K.

1961 **Categories of the Theory of Grammar.** Word 17 (1961), pp. 241-292.

HALLIDAY, M. A. K. AND MARTIN, J. R. (EDS.)

1981 **Readings in Systemic Linguistics.** Batsford Academic & Educational Ltd, London (1981), 361 pp.

HALVORSEN, PER-KRISTIAN AND LADUSAW, WILLIAM

1979 **Montague's 'Universal Grammar': an Introduction for the Linguist.** In: *Linguistics and Philosophy.* Vol. 3, D. Reidel, Dordrecht, Holland (1979), pp. 185-223.

HARRISON, MICHAEL A.

1978 **Introduction to Formal Language Theory.** Addison-Wesley (1978), xiv+594 pp.

VAN HEIJENOORT, JEAN

1967 **From Frege to Gödel. A Source Book in Mathematical Logic,1879-1931.** Harvard University Press, Cambridge, Mass. (1967), viii+664 pp.

HINDLEY, J. ROGER AND SELDIN, JONATHAN P.

1986 **Introduction to Combinators and λ-Calculus.** Cambridge University Press, Cambridge (1986), iv+360 pp.

HINTIKKA, JAAKKO

1964 **Towards a Theory of Inductive Generalization.** In: *Logic, Methodology and Philosophy of Science (Proceedings of the 1964 International Congress for)* Jerusalem, August 1964, Yehoshua Bar-Hillel, ed., North-Holland, Amsterdam (1965), pp. 274-288.

1967 **Individuals, Possible-Worlds and Epistemic Logic.** In: *Noûs*. Vol 1, (1967), pp. 33-62.

1975 **Impossible Possible Worlds Vindicated.** In: *Game-Theoretical Semantics*. E. Saarinen, ed., D. Reidel, Dordrecht, Holland (1978), pp. 367-379.

1976 **Language-Games.** In: *Game-Theoretical Semantics*. E. Saarinen, ed., D. Reidel, Dordrecht, Holland (1978), pp. 1-26.

HODGES, WILFRID

1977 **Logic.** Penguin Books (1988), 331 pp.

HOFSTADTER, DOUGLAS R.

1979 **Gödel, Escher, Bach: an Eternal Golden Braid.** Basic Books, New York (1979), xxi+777 pp.

HOLLOWAY, ANGELA

1987 **Experiments in English Word Formation.** MSc Thesis. Dept. of Computing, Imperial College, University of London (1987).

HUGHES, G. E. AND CRESSWELL, M. J.

1968 **An Introduction to Modal Logic.** Methuen, London (1974), x+386 pp.

JOWSEY, H. EINAR.

1986 **Montague Grammar and First-Order Logic.** In: *Edinburgh Working Papers in Cognitive Science*. Centre for Cognitive Science, University of Edinburgh (1986), pp. 143-194.

KANT, IMMANUEL

1800 **Introduction to Logic.** Translated by T. K. Abbott from the Introduction in Kant's **Logik**, pp. 170-265. Vision Press Limited, London (1963), 100 pp.

KAY, MARTIN

1985 **Parsing in Functional Unification Grammar.** In: *Natural Language Parsing*, David R. Dowty, Lauri Karttunen and Arnold M. Zwicky, eds., Cambridge University Press (1985), pp. 251-278.

KEENAN, EDWARD L. AND FALTZ, LEONARD M.

1983 **Boolean Semantics for Natural Language.** In: *Studies in Linguistics and Philosophy*, Vol 23. D. Reidel, Holland (1983), xii+340 pp.

KENNY, ANTHONY

1973 **Wittgenstein.** Penguin Books, Harmondsworth (1986), x+240 pp.

KENT, STUART J. H.

1987 **A Quirkian View of Word Formation in the English Language.** Dept. of Computing, Imperial College, London (1987), 16 pp.

KONOLIGE, KURT

1986 **A Deduction Model for Belief.** Pitman, London and Morgan Kaufmann, California (1986), 165 pp.

KUHN, STEVEN T.

1988 **Tense, Aspect, Time and Events.** In: *Handbook of Philosophical Logic.* Vol. 4, Dov Gabbay and F. Guenthner, eds., D. Reidel, Dordrecht, Holland (1989), pp. 513-554.

KUHN, THOMAS S.

1962 **The Structure of Scientific Revolutions.** Second edition, enlarged. University of Chicago Press (1970), xii+210 pp.

LAKATOS, IMRE

1976 **Proofs and Refutations: the Logic of Mathematical Discovery.** Cambridge University Press (1983), xii+174 pp.

LALOE, FRANCK

1986 **Les Surprenantes Préditions de la Mécanique Quantique.** In: *La Recherche*, No. 182 (November 1986), pp. 1358-1386.

LANDMAN, FRED

1986 **Towards a Theory of Information. The Status of Partial Objects in Semantics.** In: *Groningen-Amsterdam Studies in Semantics.* Vol 6. Foris Publications, Dordrecht, Holland (1986), xiv+228 pp.

LEAR, JONATHAN

1980 **Aristotle and Logical Theory.** Cambridge University Press (1985), xi+123 pp.

LEHMAN, MANNY

1988 **Software's Time-Bombs.** In: *The Guardian* (June 23 1988) pp. 27.

LEVESQUE, HECTOR J.

1984 **A Logic of Implicit and Explicit Belief.** In: *Proceedings of the National Conference on Artificial Intelligence* (1984), pp. 198-202.

LEWIS, DAVID

1986 **On the Plurality of Worlds.** Basil Blackwell, Oxford (1986), ix+276 pp.

MALMGREN, HELGE

1971 **Intentionality and Knowledge. Studies in the Philosophy of G. E. Moore and Ludwig Wittgenstein.** Goteborgs Universitet, Sweden (1971), iv+104 pp.

MANNO, AMBROGIO GIACOMO

1969 **Filosofia del Linguaggio. Atomismo Logico, Neopositivismo, Filosofia Analitica.** Libreria Scientifica Editrice, Napoli, Italy (Settembre 1969), ix+276 pp.

MARQUEZ, GABRIEL GARCIA

1967 **Cien Años de Soledad.** Translated by Gregory Rabassa under the title **One Hundred Years of Solitude.** Harper & Row Publishers, London (1970), 336 pp.

MARTIN-LÖF, PER

1983 **On the Meaning of the Logical Constants and the Justification of the Logical Laws.** Lecture notes to the meeting *Teoria della Dimostrazione e Filosofia della Logica*, Siena (1983). University of Stockholm, Sweden (1983), 79 pp.

MILLER, DAVID W. (ED.)

1983 **A Pocket Popper.** Fontana Press, London (1987). 479 pp.

MILNE, ROBERT AND STRACHEY, CHRISTOPHER

1976 **A Theory of Programming Language Semantics.** (parts a and b). Chapman and Hall, London (1976).

MINSKY, MARVIN AND PAPERT, S.

1969 **Perceptrons.** MIT Press, Cambridge, Mass. (1969).

MOENNICH, UWE AND BEALER, GEORGE

1988 **Property Theories.** In: *Handbook of Philosophical Logic.* Vol. 4, Dov Gabbay and F. Guenthner, eds., D. Reidel, Dordrecht, Holland (1989), pp. 133-252.

MONTAGUE, RICHARD

1968 **Pragmatics.** In: *Formal Philosophy. Selected Papers of Richard Montague.* Richmond H. Thomason, ed., Yale University Press, New Haven and London (1974), pp. 95-118.

1970a **Pragmatics and Intensional Logic.** In: *Formal Philosophy. Selected Papers of Richard Montague.* Richmond H. Thomason, ed., Yale University Press, New Haven and London (1974), pp. 119-147.

1970b **English as a Formal Language.** In: *Formal Philosophy. Selected Papers of Richard Montague.* Richmond H. Thomason, ed., Yale University Press, New Haven and London (1974), pp. 189-221.

1970c **Universal Grammar.** In: *Formal Philosophy. Selected Papers of Richard Montague.* Richmond H. Thomason, ed., Yale University Press, New Haven and London (1974), pp. 222-246.

1970d **The Proper Treatment of Quantification in Ordinary English.** In: *Formal Philosophy. Selected Papers of Richard Montague.* Richmond H. Thomason, ed., Yale University Press, New Haven and London (1974), pp. 247-270.

MORRIS, C.

1938 **Foundations of the Theory of Signs.** In: *International Encyclopedia of Unified Science*, Otto Neurath, ed., Vol 1, No 2, University of Chicago Press (1955), pp. 77-137.

MUSKENS, REINHARD

1988 **Going Partial in Montague Semantics.** Instituut voor Tall, Logica en Informatie (Institute for Language, Logic and Information), Department of Philosophy, University of Amsterdam (March 1988). 41 pp.

PARTEE, BARBARA H.

1982 **Compositionality.** In: *Varieties of Formal Semantics.* Fred Landman and Franck Veltman, eds., Foris Publications, Dordrecht, Holland (1984), pp. 281-311.

1988 **Syntactic Categories and Semantic Types.** University of Massachussets, Amherst, USA (1988). 10 pp. Presented at the *Workshop on Computational Linguistics and Formal Semantics, Lugano, Switzerland, August 1988.*

PARTEE, BARBARA H., TER MEULEN, ALICE AND WALL, ROBERT

1987 **Mathematical Methods in Linguistics.** In: *Studies in Linguistics and Philosophy.* Vol 30. Kluwer Academic, Dordrecht, Holland (1987), 372 pp.

PEREIRA, FERNANDO AND WARREN, DAVID H. D.

1978 **Definite Clause Grammars Compared with Augmented Transition Networks.** Research Report, Department of AI, University of Edinburgh (1978). *Journal of AI*, Vol. 13 (1980)

PERRY, JOHN

1987 **From Worlds to Situations.** Center for the Study of Language and Information. Stanford University. Informal Notes No. CSLI-87-73 (January 1987), 21 pp.

PIATTELLI-PALMARINI , MASSIMO

1980 **Language and Learning. The Debate between Jean Piaget and Noam Chomsky.** Routledge and Kegan Paul, London (1980).

PINTO, ANTONIO VAZ

1971 **Introdução ao Tractatus Logico-Philosophicus de Ludwig Wittgenstein.** Publicações da Faculdade de Filosofia, Universidade de Braga, Portugal (1982), 169 pp.

PITT, JEREMY V.

1987 **Allen's Action and Time and Quirk's Traditional Semantics for Verb Phrases.** Dept. of Computing, Imperial College, University of London (1987): *Personal Communication.*

1989 **A Computational Framework for the Formal Analysis of Natural Language Grammars.** PhD Dissertation, Dept. of Computing, Imperial College, University of London (1989). *Forthcoming.*

PITT, JEREMY V. AND CUNNINGHAM, R. J.

1986 **LLARMA is more logical than YACC: A Prolog Parser Generator for Natural Language Analysis.** Dept. of Computing, Imperial College, London (November 1986), 17 pp.

PLOTKIN, GORDON D.

1981 **Post-Graduate Lecture Notes in Advanced Domain Theory.** Dept. of Computer Science, University of Edinburgh (1981).

POPPER, KARL RAIMUND

1935 **Logik der Forschung.** Translated under the title **The Logic of Scientific Discovery.** Hutchinson, London (1968), 480 pp.

1945 **The Open Socity and its Enemies.** Routledge & Kegan Paul, London (1966), Vol. 1 **Plato,** xi+361 pp., Vol. 2 **Hegel and Marx,** v+420 pp.

1953 **Science: Conjectures and Refutations.** In *Conjectures and Refutations. The growth of Scientific Knowledge,* Routledge & Kegan Paul, London (1972), pp. 33-65.

1958 **Back to the Presocratics.** In *Conjectures and Refutations. The growth of Scientific Knowledge,* Routledge & Kegan Paul, London (1972), pp. 136-165.

1960 **Truth, Rationality and the Growth of Scientific Knowledge.** In *Conjectures and Refutations. The growth of Scientific Knowledge,* Routledge & Kegan Paul, London (1972), pp. 215-250.

1972 **Objective Knowledge.** 2nd edition, Oxford University Press, London (1979).

1974 **Unended Quest. An Intellectual Autobiography.** First published as *Autobiography of Karl Popper* in *The Philosophy of Karl Popper,* in *The Library of Living Philosophers,* Paul Arthur Schlipp (ed.), Open Copurt Publishing Co. Illinois (1974). Flamingo revised edition, with Postscript and updated bibliography, Fontana Paperbacks, Collins Publishing Group (1986).

DE QUEIROZ, RUY JOSÉ GUERRA BARRETO

1988 **Meaning as Use, not Content: an Alternative View on the Proof-Theoretic Account of Meaning.** Dept. of Computing, Imperial College, University of London (1988).

1989 **Proof Theory and Computer Programming. The Logical Foundation of Computation.** PhD Dissertation. Dept. of Computing, Imperial College, University of London (1989). *Forthcoming.*

DE LA QUINTANA, PABLO

1988 **Automated reasoning for Modal Logics: A Natural Deduction Based Approach.** PhD Dissertation. Dept. of Computing, Imperial College, University of London (October 1989).

QUIRK, RANDOLPH, GREENBAUM, SIDNEY, LEECH, GEOFFREY AND SVARTVIK, JAN

1985 **A Comprehensive Grammar of the English Language.** Longman, London(1985), x+1779 pp.

RANTALA, V.

1982 **Impossible Worlds Semantics and Logical Omniscience.** In: *Acta Philosophica Fennica*, Vol. 35 (1982), pp. 106-115.

RESCHER, NICHOLAS AND URQUHART, ALASDAIR

1971 **Temporal Logic.** Springer-Verlag, New York (1971), xviii+273 pp.

ROBINSON, JOHN ALAN

1979 **Logic: Form and Function. The Mechanization of Deductive Reasoning.** University Press, Edinburgh (1979), vi+311 pp.

ROUNDS, W. C. AND KASPER, R.

1986 **A Complete Logical Calculus for Record Structures Representing Linguistic Information.** In: *Symposium on Logic in Computer Science*, IEEE Computer Society (1986).

RUMELHART, DAVID E. AND MCCLELLAND, JAMES L.

1986 **PDP Models and General Issues in Cognitive Science.** In: *Parallel Distributed Processing. Explorations in the Microstructure of Cognition Vol 1: Foundations.* David E. Rumelhart, James L. McClelland and the PDP Research Group, eds., MIT Press, Cambridge, Mass. (1987), pp. 111-146.

RUSSELL, BERTRAND

1908 **Mathematical Logic as Based on the Theory of Types.** In: *Logic and Knowledge: Essays, 1901-1950.* Robert Charles Marsh, ed., George Allen & Unwin, London (1956), pp. 57-102.

1918 **The Philosophy of Logical Atomism.** In: *Logic and Knowledge: Essays, 1901-1950.* Robert Charles Marsh, ed., George Allen & Unwin, London (1956), pp. 175-281.

1924 **Logical Atomism.** In: *Logic and Knowledge: Essays, 1901-1950.* Robert Charles Marsh, ed., George Allen & Unwin, London (1956), pp. 321-343.

1946 **A History of Western Philosophy.** George Allen & Unwin, London (1987), 842 pp.

1950 **Logical Positivism.** In: *Logic and Knowledge: Essays, 1901-1950.* Robert Charles Marsh, ed., George Allen & Unwin, London (1956), pp. 365-382.

SALMON, NATHAN

1988 **Reference and Information Content: Names and Descriptions.** In: *Handbook of Philosophical Logic.* Vol. 4, Dov Gabbay and F. Guenthner, eds., D. Reidel, Dordrecht, Holland (1989), pp. 409-462.

SCHMIDT, DAVID A.

1986 **Denotational Semantics - a Methodology for Language Development.** Allyn & Bacon Inc., Boston (1986), xiii+331 pp.

SCOTT, DANA S.

1976 **Data Types as Lattices.** In: *SIAM Journal on Computing.* Vol. 5, No 3 (September 1976), pp. 522-587.

1977 **Logic and Programming Languages.** In: *Communications of the ACM.* Vol. 20, No 9 (September 1977), pp. 634-641.

1981 **Lectures on a Mathematical Theory of Computation.** *Lecture Notes in Mathematics,* Oxford University Computing Laboratory, Technical Monograph PRG-19 (May 1981), 148 pp.

1982 **Domains for Denotational Semantics.** I.C.A.L.P. (1982), pp. 577-613.

SHIEBER, B.

1985 **Evidence against the Context-Freeness of Natural Language.** In *Linguistics and Philosophy*, Vol. 8 (1985), pp. 333-343.

SHOHAM, YOAV

1986 **Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence.** PhD Thesis, Yale University, Computer Science Department. Published by MIT Press, Cambridge, Mass. (1988), xvi+200 pp.

SMITH, M.

1989 **Continous and Monotonic Quantifiers.** Dept. of Computing, Imperial College, London (1989). *Personal Communication.*

STOY, J. E.

1977 **Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory.** MIT Press, Cambridge, Mass. (1981), xxx+414 pp.

TARSKI, ALFRED

1931 **The Concept of Truth in Formalized Languages.** In: *Logic, Semantics, Metamathematics*, Oxford (1956), pp. 152-278.

THOMASON, RICHMOND H.

1974 **Introduction.** In: *Formal Philosophy. Selected Papers of Richard Montague;* Thomason, Richmond H., ed., Yale University Press, New Haven and London(1974), pp. 1-69.

TURNER, RAYMOND

1983a **Montague Semantics, Nominalization and Scott's Domains.** In: *Linguistics and Philosophy.* Vol. 6 , D. Reidel (1983), pp. 259-288.

1983b **Nominalization and Scott's Domains II.** Department of Computer Science, University of Essex, UK (1983?). 39 pp.

1987 **A Theory of Properties.** In: *The Journal of Symbolic Logic*, Vol 52, (1988). pp. 455-472.

1988a **Properties, Propositions and Semantic Theory.** Department of Computer Science, University of Essex, UK (1988). 15 pp. Presented at the *Workshop on Computational Linguistics and Formal Semantics, Lugano, Switzerland, August 1988.*

1988b **Quantification and Scott's Domains.** Department of Computer Science, University of Essex, UK (1988). *Personal Communication.*

VARDI, M. Y.

1986 **On Epistemic Logic and Logical Omniscience.** In: *Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge*, J. Y. Halpern, ed. (1986), pp. 293-305.

VISSER, ALBERT

1988 **Semantics and The Liar Paradox.** In: *Handbook of Philosophical Logic.* Vol. 4, Dov Gabbay and F. Guenthner, eds., D. Reidel, Dordrecht, Holland (1989), pp. 617-706.

WESTERSTÄHL, DAG

1988 **Quantifiers in Formal and Natural Languages.** In: *Handbook of Philosophical Logic.* Vol. 4, Dov Gabbay and F. Guenthner, eds., D. Reidel, Dordrecht, Holland (1989), pp. 1-131.

WINOGRAD, TERRY

1972 **Understanding Natural Language.** Academic Press, New York (1972).

1983 **Language as a Cognitive Process. Vol 1: Syntax.** Addison-Wesley (1983), xiv+640 pp.

WHITEHEAD, ALFRED NORTH AND RUSSELL, BERTRAND

1910 **Principia Mathematica.** To *56. Cambridge University Press (1970), xlvi+410 pp.

WITTGENSTEIN, LUDWIG

1921 **Logisch-Philosophische Abhandlung.** Translated by C. K. Ogden under the title **Tractatus Logico-Philosophicus.** Routledge & Kegan Paul, London and New York (1986), 207 pp.

1953 **Philosophische Untersuchungen.** Translated by G. E. M. Anscombe under the title **Philosophical Investigations.** Basil Blackwell, Oxford (1974), viii+250 pp.

1969 **Über Gewissheit.** Translated by Denis Paul and G. E. M. Anscombe under the title **On Certainty.** Basil Blackwell, Oxford (1988), 105+90e pp.