On Compromising Updates in Labelled Databases

Imperial College Research Report DoC 96/1

Fátima C. C. Dargam
e-mail: fccd@doc.ic.ac.uk
Department of Computing
Imperial College of Science, Technology and Medicine
180 Queens Gate, London SW7 2BZ
United Kingdom

25 March 1996

Abstract

This paper presents a logical system, CIU_{LDS}, as a labelled realization to our approach of *Compromising Interfering Updates*. Basically, this approach proposes a method for handling logically conflicting inputs into knowledge bases, via restricting their consequences. The main idea is to update the database with as many consistent consequences of the inputs as possible, in the case that the inputs themselves are not allowed to be kept in it. And in the case that a revision applies, the idea is to keep as many as possible of the consistent consequences of the retracted sentences as a compromise.

Our approach caters for the specific case where compromised solutions for revising knowledge bases apply, when conflicts involving updates occur. In comparison with approaches that require preference between conflicting inputs, or that avoid them by cancelling them out completely, our approach fits as an alternative which provides more informative results, and is directed to some specific applications. Hence, instead of preventing updates to be performed, when they introduce inconsistency to the system, our approach proposes to generate the consequences of the conflicting inputs, and to get rid of the inconsistency, via a minimal number of retractions of those consequences. We expect the resulting database to be consistent w.r.t. the integrity constraints, and to retain a *safe-maximal* subset of the consistent non-supported consequences. This reconciliation of conflicting inputs follows some specified postulates for compromised revision.

CIU_{LDS} is based on the Labelled Deductive Systems framework (LDS). This framework deals with labelled formulae as its basic units of information. By labelling the formulae, we are provided with a way of including in the labels extra information to the system. The main motivation for adopting LDS as the underlying framework of this formalization was to take advantage of its labelling facility, to control the derivation process of the compromised consequences. We embed in the labelling propagation conditions, which act on the inference rules, part of the control mechanism for the compromised approach. This control mechanism helps the update operations to perform the reconciliation of conflicting inputs. The update operations invoke a compromised revision on the labelled database, whenever conflicts arise.

In this paper, we present briefly our main motivations and we discuss the general issue of conflict resolution and theory revision. We introduce the basic specification of our approach CIU, for the case of database updates, describing the adopted policies for reconciling conflicting updates under a compromised reasoning perspective. We introduce the CIU_{LDS} system, by decribing informally its main features and definitions. In CIU_{LDS}, we propose a specific revision method which applies some compromising criteria for achieving the revised database. Finally, we summarize the system's main properties.

1 Introduction

Resolving conflicting updates in dynamic databases, or conflicting actions in planning applications, for instance, are frequent and critically important problems of real applications. Such problems require the revision of theories and knowledge bases. As pointed out by Winslett in [Win-90], it is not realistic to aim for a generic approach in those cases, since theory revision is fundamentally dependent on application-specific mechanisms, principles and heuristics. The approach we propose in this paper, caters for the specific case where compromised solutions apply when conflicts occur. It supports a compromised way of handling conflicting updates for revising databases.

Our approach is mainly suitable for applications which allow for compromised solutions, i.e. solutions which present the closest result with relation to the expected one. Some of the applications which can benefit from our approach, are in the area of *design processes*. Here, one builds up the goal state of a particular task, via performances of intermediary updates. This procedure allows for compromised results of updates when conflicts arise.

In more practical terms, consider the situation where DB is a database and A an input. Assume that A is inconsistent with DB . Current belief revision/update approaches will keep A and mantain consistency by selecting some element from DB to form a revised database, usually denoted as $DB * A$. There is a lot of research in this area, both theorectical, e.g.: the AGM theory of belief revision[1], and algorithmic research, e.g.: Reason Maintenance Systems[2]. Our aim is to offer an alternative approach, which is flexible enough to keep more data in DB, in the case of conflicts. We view the above situation as a *conflict* between two inputs (DB and A) into an empty database, and we tackle the problem of reconciling these inputs. Under our approach, the conflicting input A is kept in DB only in the case that A generates inconsistency to DB indirectly[3], in which case a revision also applies in order to restore consistency. However, in the case that A is not allowed to be kept in DB, its consistent consequences, w.r.t. the existing data of DB, are added to the database under the compromised policy of our approach. This way, instead of preventing updates to be performed, when they introduce inconsistency to the system, our approach reconciles the conflicting inputs by compromising on their consequences. We propose to generate the consequences of the conflicting inputs, and get rid of the inconsistency, via a minimal number of retraction of those consequences. We expect the resulting database to be consistent w.r.t. the integrity constraints, and to retain a *safe maximal subset* of the consistent consequences of those updates. This reconciliation of conflicting inputs follows some specified postulates for compromised revision.[4]

1.1 Motivations

As pointed out by Galliers [Gal-90], in most of the existing AI research work, conflicts either simply never arise, or are alternatively avoided when they do arise. However, in a constantly changing and unpredictable environment, inconsistencies within the system are most of the times inevitable, and conflict situations do arise. The central interest in [Gal-90] is to solve conflicts in cooperative multiagent systems, by facing their positive aspects. They claim that achievement of cooperation from conflict, among formalized agents may involve the decision of a mutually preferred compromised solution, and/or persuasion to the validity of another position.

Gabbay and Hunter, in [GaHu-91,93], also support that inconsistency should be faced and formalized. They urge for a revision on the way inconsistency is currently being handled in formal logical systems, as opposed to the way it is handled by humans. They claim that there is a need for the development of a framework, in which inconsistency can be viewed in a context-dependent way. As a signal for external and/or internal actions, and not necessarily as a bad element which induces the whole system to collapse.

[1] The AGM theory was first introduced in [AlMa-82,85,86][AGM-85], and since then gained many followers who apply and modify that theory in various ways, see for instance [Mak-85],[Neb-89] [RaFo-89],[JaPa-90],[Neb-90],[Rot-91,92],[KaMe-92] [BoGo-93],[Mak-93],[Sri-93] and [FrLe-94].

[2] Reason Maintenance Systems were initiated in [Doy-79] based on justifications, and in [Kle-86a] based on assumptions. More recent research work following this line have also emerged. Some of them are found in [Elk-90],[GiMa-90],[PiCu-89],[RoPi-91],[Salw-91] and [WaCh-91].

[3] By *indirectly* here, we mean that A alone does not violate any of DB's integrity constraints.

[4] In [Dar-96c,96d], the compromised revision is defined under a belief revision perspective and some postulates for finite bases with integrity constraints are introduced, as guidelines for the compromised revision function.

They argue that dealing with inconsistencies is not necessarily a job for restoring consistency, but rather for supplying rules which state how to act in the case of inconsistencies.

We strongly endorse the viewpoints of Galliers and Gabbay & Hunter. Based on the same grounds, we investigate an approach which handles conflicts that introduce inconsistency into a system, and puts forward a compromised reasoning way for dealing with conflicting updates and actions, instead of simply avoiding them.

As in [Gal-90], we propose to solve conflicts by facing their positive aspects. We do so, by reconciling the conflicting updates with the underlying knowledge base, and getting as many of their consequences as possible.

We support the point in [GaHu-91,93], that inconsistency (caused by conflicts) should be faced and that we should supply mechanisms for handling situations when they arise. In the current work, we approach such situations by allowing some consequences of the conflicting updates to remain in the database. However, by reconciling the conflicting inputs, we also restore consistency, which does not conform to their view of keeping inconsistency in the system and supplying the appropriate mechanisms to handle it.

Our main motivation in pursuing this approach, comes from the premise that by reconciling conflicting updates with a knowledge base, our approach provides more informative results. In comparison with approaches that require preference between conflicting inputs, or that simply avoid them by cancelling them out completely, our proposal of compromised revision allows more information to be kept in a theory base. Following our compromised revision approach to conflicts, one will possibly not get all of what he/she originally wanted[5], in the case of conflict. Instead, he/she will get extra data, leading to the direction of the original goal. This is because most of the extra data are related to the goal's consistent consequences.

The results we get with our approach suit the needs of particular application areas, e.g. design processes; resource allocation; and decision making. As an application example, let us consider a research organization which has the task of deciding the allocation of funds among projects. We assume that it is necessary for the projects to discriminate all the expenses required for each of their phases, allowing for the option of satisfying only partially those phases (compromised solutions). We assume also that the decision makers are not supposed to favour any project in particular. So, if funds are not sufficient to support all the projects' requirements, our approach would be appropriate to be applied in the process of funds allocation. In the sense that it would allow for as many of all the projects' phases as possible, considering the constraints involved in the process.

1.2 Our Approach to Handling Conflicting Inputs

Our approach proposes to reconcile conflicting inputs with respect to the underlying theory, and establishes some policies for dealing with the problem of inconsistency caused by them. The way we approach the problem of conflicting inputs differs from the other existing approaches, in the sense that we allow for a special process of performance of the conflicting updates. A process of *reconciliation of conflicting inputs*, which considers restrictions of the effects of those inputs by compromising on their consequences. We refer to our approach as CIU, meaning *Compromising Interfering Updates.*

By conflicting, or interfering, updates, we mean either simultaneous updates which interfere with each other, generating inconsistencies as part of their combined effects, or updates which are inconsistent to be performed because they conflict with the given database or scenario representation, by violating some of their constraints. Below, we present two examples which illustrate the intuitive notion of our approach with relation to database updates.

Example 1.1 *Let us consider the database of formulae as shown below:*

$$\begin{array}{ll} (1) & A \rightarrow B \\ (2) & A \wedge C \rightarrow \bot \\ (3) & A \end{array}$$

[5]The idea of this revision approach conforms with the meaning of the word *compromise*. Quoting from the Oxford's Dictionary: *"Compromise" is a settlement of a dispute which each side gives up something it has asked for, and neither gets all it asked for.*

If we want to update this database with the formula (4) C , then, by applying a TMS-*like approach [Doy-79], for instance, we would force C in, by removing A and all the consequences derived from A , in order to keep consistency, as shown below.*

$$\begin{array}{ll} (1) & A \rightarrow B \\ (2) & A \wedge C \rightarrow \bot \\ (4) & C \end{array}$$

In the way we approach this problem, we would also end up with either A or C , but not both. However, we want to be able to keep all the consistent derived consequences of the conflicting update. In this case, we would be able to have B as well in the resulting database, as shown below.

$$\begin{array}{ll} (1) & A \rightarrow B \\ (2) & A \wedge C \rightarrow \bot \\ (4) & C \\ (5) & B \end{array}$$

•

The example above could be interpreted with the following meanings for the sentences A, B and C: A = "Executive Class Passenger"; B = "Extra baggage allowance"; C = "Economic Class Passenger". Then, we would have that the database update above represents a situation, in which an executive class passenger for some reason has to be moved to the economic class, in a particular flight. However, even in the the economic class, the originally executive class passenger still keeps his/her right of having extra baggage allowance.

CIU can be described as a module of a reasoning system, which is invoked whenever we have conflicting updates, w.r.t. databases and to their sets of integrity constraints. We assume that we have a database module D which is subjected to a module of integrity constraints I . The integrity constraints are assumed to be protected against any update modification, and they restrict the possible transactions on D. The database can be, for instance, a declarative representation of a scenario, in terms of the facts that hold in the current state. A finite set of updates, to be performed on the database, is given as input to the system. We assume that the updates executing module only effectively performs the updates in the case that they modify D without violating any integrity constraint. Otherwise, CIU module is invoked in order to perform the compromised version of the set of updates. In the end, the compromised updated database is supposed to be consistent and to satisfy the constraints in module I. The peculiar characteristic that CIU has in dealing with updates is that, instead of preventing an update to be performed when inconsistency arises[6], CIU proceeds and generates the consequences of the conflicting updates.

When CIU is invoked, it instigates the compromised performance of the conflicting updates, by firstly generating all their consequences/derivations. Later it takes care of restoring consistency in the database. In order to restore consistency, a special revision procedure takes place. It is based on the minimal elimination of the formulae involved in the generation of inconsistency, and guided by the compromised reasoning policies of the approach.

1.2.1 Different Kinds of Conflicting Inputs

Conflicting inputs can be of various kinds. For instance, we can have simultaneous updates which interfere with each other, generating inconsistencies as part of their combined effects, or updates which are inconsistent to be performed because they conflict with the given database, by violating some of its constraints. We can also have the case in which updates are individually consistent to be applied, but if performed in parallel they interfere with each other.[7]

[6] This would make the approach equivalent to many existing ones which do not allow for updates to be performed if they are not consistent with the theory.

[7] A similar motivating approach was pursued by Cholvy [Cho-93] in the context of multi-sourced information environment. Cholvy treats the problem of consistency of information provided by different sources, considering the case that the global set of information is inconsistent even if each separate source is consistent. Notice, however, that in this work we propose to deal with inconsistency generated by conflicting updates within the same system, while Cholvy treats the inconsistent information which is due to the combination of different data/knowledge bases. A further analogy between the two approaches requires, at least, a re-definition of the basic conflicting entities, in order to cater for the representation of information sources.

Below, we state clearly all the different kinds of conflicting inputs that we are considering, and we describe how we propose to handle them. We consider that A and B are formulae of the language being considered. We consider classical logic as the underlying logic, including the usual connectives. A database DB is such that $DB = \Delta \bigcup P_\Delta$, where Δ denotes set of formulae which compose the body of the database, and P_Δ denotes the set of integrity constraints which rules Δ.

(a) Conflicting inputs within the update, or within the transaction, e.g. Update $= \{A, \neg A\}$. In this case, the updates are rejected, since one logically cancels the other. However, if within a transaction T we have the following sequence of inputs $T = \{A, \neg A, B\}$, the subset $\{A, \neg A\}$ is removed from T and the remaining inputs in the transaction are still performed. In this case, $T = \{B\}$.

(b) Inputs which conflict directly with some of the integrity constraints which rule the database, e.g. $DB = \Delta \bigcup P_\Delta$, $P_\Delta = \{A \rightarrow \bot\}$, and Update $= \{A\}$. In this case, the input is not allowed to be inserted in the database. However, we allow the consistent consequences of the input to be inserted in DB, with particular status of non-supported data.

(c) Inputs which conflict indirectly with some of the integrity constraints which rule the database, e.g. $DB = \Delta \bigcup P_\Delta$, $P_\Delta = \{A \wedge C \rightarrow \bot\}$, $\Delta = \{C\}$, and Update $= \{A\}$. In this case, the input is inserted in the database and a revision procedure takes place in order to restore consistency and allow the database to accomplish the new update. The revision presents special properties which preserves the consistent consequences of all the retracted formulae from DB.

(d) Inputs which contradicts existing data of the database, e.g. $DB = \Delta \bigcup P_\Delta$, $\Delta = \{A\}$, and Update $= \{\neg A\}$. In this case, the input is inserted in the database and a revision on DB takes place, just as described above.

1.2.2 Multiple Updates Case

In the case of a transaction, which involves a set of single updates, if we have "n" conflicting updates w.r.t. the integrity constraints, the resulting compromised updated database might contain at most "$n-1$" of those updates. Transactions have their consistency initially checked with relation to the three conditions described below. Assume that a transaction $T = \{U_1, U_2, \cdots, U_n\}$, composed of n updates, is to be performed to DB, and I is the set of integrity constraints which rules DB.[8]

1. For any U_i and any U_j in a transaction $T = \{U_1, \cdots, U_n\}$, where $1 \leq i \leq n$; $1 \leq j \leq n$; and $i \neq j$, if U_i expresses a formula which is the complement of the formula expressed by U_j, say A and $\neg A$ [9], then the set $\{U_i, U_j\}$ is retracted from the transaction T.

2. For any U_i in a transaction $T = \{U_1, \cdots, U_n\}$, where $1 \leq i \leq n$, if U_i violates an integrity constraint in I, $\{U_i\} \bigcup I \vdash \bot$, then the update U_i is rejected, however its consistent consequences are allowed to remain as non-supported consequences in the database.

3. For any U_i and U_j in a transaction $T = \{U_1, \cdots, U_n\}$, where $1 \leq i \leq n$; $1 \leq j \leq n$; and $i \neq j$, such that U_i and U_j are not complementary updates in T, and $\{U_i\} \bigcup I \nvdash \bot$ anf $\{U_j\} \bigcup I \nvdash \bot$, if $\{U_i, U_j\} \bigcup I \vdash \bot$, then a choice is made between U_i and U_j, according to meta-level information concerning, the relevance of the updates within the transaction[10]. In this case, the transaction is then reduced to $T - \{U_k\}$, where k is either i or j, depending on this meta-level based choice, however the consistent consequences of U_k are allowed to remain as non-supported consequences in the database.

In the cases described above, when consistency of the updates is not obtained initially, the updates which cause inconsistency are not supposed to be performed, since they are removed from the transaction, as

[8] The formulae considered here are propositional sentences from the system of propositional classical logic.

[9] This would represent adding and deleting the formula A in the same transaction.

[10] This meta-level information is totally context-dependent. We could, for instance, have a total ordering among the updates in T, so that each U_i would be less preferable than U_{i+1}. We will not discuss details about this meta-level based choice in this paper.

described. However, the transaction is not cancelled due to the fact that some of its updates failed the initial consistency checking phase[11].

Condition 1 above, ensures that complementary information is cancelled prior to the database transaction performance, in order to avoid redundant update execution.

Condition 2 puts forwards that an update U_i which violates directly an integrity constraint of the database cannot be performed, however, under our compromised approach, its consistent consequences can be kept in DB. For instance, if $T = \{A, B, C\}$, $DB = \{\ \}$ and $I = \{C \rightarrow \bot\}$, the transaction would be reduced to $\{A, B\}$, since $\{C\} \bigcup I \vdash \bot$. Eliminating the update that violates the integrity constraint from the transaction, and allowing the other updates in T to be performed is, most of the times, an intuitive procedure which conforms with the compromised philosophy of our approach. Consider the case that A expresses that worker $W1$ gets a raise of 10% on his salary; B expresses that worker $W2$ gets a raise of 30%; and C expresses that worker $W3$ gets a raise of 50% on his salary. Assume that their company has restricted raises of 50% or higher on workers' salaries. Then, update C would not be performed and would have to be negotiated later. However, this would not stop updates A and B from being performed.

Condition 3 caters for the case when two updates are individually consistent to be performed, but together they violate the set of integrity constraints which rules the database. In the case of two conflicting updates within the same transaction, our approach allows for their consistent consequences to be kept in the database.

2 *CIU* formalized under the *LDS* Framework

Labelled Deductive System (LDS) is a logical framework, introduced by Gabbay in [Gab-94], for the representation of various existing logical systems and their interactions. LDS basically arose in response to conceptual pressure from various application areas and their needs. The LDS framework deals with labelled formulae as its basic units of information, where the labels can be of arbitrary form, belonging to a given *labelling algebra*. LDS's derivation rules act on the labels as well as on the formulae. These rules include some prescribed ways, given by the labelling algebra, to propagate the labels. The handling of labelled formulae is a very important feature of LDS. The extra power given by the labelling algebras allows standard proof systems to be extended with non-standard features. Hence, the LDS formalism provides a rich syntactic characterization for a proof-theorectical presentation, which combines the information of the labels with the formulae. This way, the proof systems are able to cover a wider operating scope, without modifying their structure.

By labelling the formulae, we are provided with a way of including in the labels extra information to the system. The original motivation was to be able to code control information in the labels, such as dependencies within a proof, and controlling flags for a derivation process. Nevertheless, also structural database information, and network information, among other meta-level pieces of information, can be incorporated explicitly into the object language via the labels. Moreover, labelling may also be used to facilitate truth maintenance and conflict resolution.

We propose here to build a Labelled Deductive System to formalize our approach. We present a logical system CIU_{LDS} as a labelled realization to our specified system CIU. Following the lines of [Gab-94], CIU_{LDS} is defined by the triple $\langle \mathcal{A}, \mathcal{L}_{CIU}, M_{CIU} \rangle$, where $\mathcal{L}_{CIU}$ denotes the system's language, $\mathcal{A}$ denotes an algebra of labels, and M_{CIU} denotes a discipline for labelling formulae of the logic, and propagating the labels within to the system's deduction mechanisms. One of the main motivations for adopting LDS as the underlying framework of this formalization was to take advantage of its labelling facility, to control the derivation process of the compromised consequences. Actually, we embed in the labelling propagation conditions, which act on the inference rules, part of the control mechanism for the compromised approach. This control mechanism helps the defined update operations to perform the reconciliation of conflicting inputs. The update operations invoke a compromised revision on the labelled database, whenever conflicts arise.

[11] Most of the database-update approaches in the literature do not conform with this viewpoint, since they adopt a style denoted sometimes as *all-or-none* updates performance, in the case of inconsistency or integrity constraint violation within a transaction, e.g. [MBM-95].

Before introducing the formal definitions of CIU_{LDS}, we discuss briefly the components and the mechanisms used in this formalization, relating them to their formal definitions as well as to the philosophy of our approach.

2.1 CIU_{LDS} Briefly Described

2.2 The Language

The system's basic units of information are labelled formulae, denoted as declarative units and written as $\gamma : \alpha$, where γ is a label and α is a logical formula. The intended meaning of $\gamma : \alpha$ is that γ indicates the nature of the formula α w.r.t. its data status in the database. In order to supply a proper syntax for these labelled formulae, the system's language $\mathcal{L}_{CIU}$ is defined as composed of a propositional logical language $\mathcal{L}$, and a distinct language for the labels $\mathcal{L}_\gamma$. The logical language $\mathcal{L}$ provides propositional well formed formulae (wff), (the α part of a declarative unit), whereas the labelling language $\mathcal{L}_\gamma$ caters for the representation of the labels. $\mathcal{L}_\gamma$ is mainly based on finite lists of typed constants symbols, used to qualify the label nature, and on binary function symbols which are used to define how labels propagate in relation to the derivation rule being applied. Definitions 2.1, 2.2, 2.3, 2.4, 2.5 and 2.8 state formally the language $\mathcal{L}$, the wff's of $\mathcal{L}$, and the languages $\mathcal{L}_\gamma$ and $\mathcal{L}_{CIU}$, respectively. Definitions 2.17, 2.18, 2.19, 2.20, 2.21 and 2.22 state how the functions of $\mathcal{L}_\gamma$ are defined w.r.t. the label types.

2.2.1 The Labelled Database

We consider a labelled database, denoted as $\mathcal{D}$, to be the tuple $\mathcal{D} = \langle \Delta_\mathcal{D}, \preccurlyeq \rangle$, where $\Delta_\mathcal{D}$ is a set of declarative units, and $\preccurlyeq$ is an ordering on the declarative units of $\Delta_\mathcal{D}$.

Notice that the ordering relation $\preccurlyeq$ is not part of the language $\mathcal{L}_{CIU}$, but it is used to compare the declarative units of this language on a meta-level. Intuitively, the notion of an ordering, comparing the declarative units of $\Delta_\mathcal{D}$, states the meaning of priority or relevance to those elements. In general, such an ordering is guided by the requirements of the database application. Among many different meanings that it can bring to $\Delta_\mathcal{D}$, we can cite that it can express the novelty of the declarative units in the database, or some sort of degree of importance, for instance. Here, we leave the interpretation of the ordering $\preccurlyeq$ open, since we do not specify which database application our formalization is dealing with. However, in the course of the formal definitions, we will assume some basic properties on $\preccurlyeq$, restricting it to our formalization requirements.

The intended meaning of using labelled formulae in our database representation is to provide the structural information of the database, by using the labels to express the nature of each of the formulae available in and from the database. Borrowing the conceptual presentation of deductive databases, we distinguish in $\mathcal{D}$ the explicit facts from the rules. In deductive databases' terms, we distinguish between the extensional and the intensional components of our labelled database $\mathcal{D}$. The extensional data refer to the formulae stored as explicit facts, and the intensional data refer to the deductive rules, in our case the clausal formulae, and the derivable data. Our approach, however, requires that we introduce two more formulae identifications to our database. One which refers to the protected data, and another which addresses the non-supported consequences generated by the compromised solutions of our revision policy. Hence, the possible types of data in our labelled database $\mathcal{D}$ are the following:

- Extensional data, used to represent explicitly stored facts;
- Intensional data, used to represent rules, (clausal formulae), and derivable facts;
- Protected data, used to represent data which cannot be modified or removed from the database, for instance, integrity constraints; and
- Non-supported data, used to qualify consequences whose set of premises is not available from the database.

This last sort of data will eventually appear after a *compromised revision* takes place on the database.

By handling declarative units, we are able to distinguish by the labels all those different data. We then define that a label γ can be of the type 'E', 'I', 'P', or 'N'. These types qualify, respectively, the possible declarative units listed above.

In this formalization we allow for the empty database, where $\Delta_{\mathcal{D}} = \emptyset$, and we consider that a single declarative unit in $\Delta_{\mathcal{D}}$ composes a singleton database $\mathcal{D}$. Definitions 2.9, 2.10, 2.11, 2.12 and 2.13 state formally the concept of declarative units and their types, and Definition 2.14 formalizes the notion of a CIU_{LDS} labelled database.

Example 2.1 *Let us consider the database $\mathcal{D}_1$, where the set $\Delta_{\mathcal{D}_1}$ of declarative units is as shown below:*

$$\begin{array}{ll} (1) & \mathrm{I}_1 : A \rightarrow B \\ (2) & \mathrm{I}_2 : C \rightarrow D \\ (3) & \mathrm{P}_1 : A \wedge C \rightarrow \bot \\ (4) & \mathrm{E}_1 : A \end{array}$$

We have that (1) and (2) are the intensional part of the database, (3) represents the protected part of the database, which serves as an integrity constraint; and (4) is the only explicit fact of this database. There is no non-supported data represented in this database.

•

2.2.2 The Inputs

The inputs of the system are in the form of update requests, which invoke an update function. This update function, formally denoted as $U(\mathcal{D}, \sigma, \delta) \Rightarrow \mathcal{D}'$, takes into account the two labelled databases $\mathcal{D}$ and $\mathcal{D}'$, before and after the update, respectively, and the arguments σ and δ. σ is the type of update to be performed, and δ is the data involved in the update. The σ argument can take one of the constant values of the set $\{\mathrm{U}_+, \mathrm{U}_-\}$, where ' U_+ ' implies that the update requests an addition of the argument δ , and ' U_- ' implies that the update operation requested is a deletion. The δ argument denotes a declarative unit $\gamma : \alpha$.

In this section, we first introduce the LDS formalization of the approach for the case that the update requests involve single declarative units. Then, we present the notion of transactions to cater for the case of a request involving a sequence of updates.

Example 2.2

Let us consider the database $\mathcal{D}_1$ of example 2.1. An update of the form $U(\mathcal{D}_1, \mathrm{U}_+, \mathrm{E}_2 : C)$ requests that the declarative unit $\mathrm{E}_2 : C$ be added to $\mathcal{D}_1$.

•

Basically, the allowed updates in CIU_{LDS} w.r.t. declarative units, are the ones involving addition or deletion of declarative units which are either extensional data or intensional data. Atomic formulae of $\mathcal{L}$, can be used in the update only as explicit facts. The intensional data involved in the updates are not supposed to be atomic formulae of $\mathcal{L}$, since these forms of intensional data represent the derivable data from the database. However, derivable formulae are allowed to be added to the database, when they are introduced as explicit facts, using extensional labels. These restrictions avoid the manipulation of non-supported data, and of protected data in the updates, as expected. Non-supported data can only be derived or added to the database by the system, as a compromised solution. And protected formulae cannot be modified by means of updates to the database. Example 2.3 illustrates some allowed updates in CIU_{LDS}.

Example 2.3

According to the database $\mathcal{D}_1$ of example 2.1, the declarative unit $\mathrm{I}_3 : B$ *is not allowed to be involved in an update request, however it would be allowed if it were given the form of an explicit fact:* $\mathrm{E}_3 : B$. *Hence,* $U(\mathcal{D}_1, \mathrm{U}_+, \mathrm{E}_3 : B)$ *is considered as a valid update request, as well as* $U(\mathcal{D}_1, \mathrm{U}_-, \mathrm{E}_1 : A)$, *and* $U(\mathcal{D}_1, \mathrm{U}_-, \mathrm{I}_2 : C \rightarrow D)$. *On the other hand,* $U(\mathcal{D}_1, \mathrm{U}_+, \mathrm{P}_2 : E \rightarrow \bot)$, *and* $U(\mathcal{D}_1, \mathrm{U}_+, \mathrm{N}_1 : D)$ *would not be valid update requests.*

•

The update function invokes the update operations of conditional addition, $\uplus$, and conditional retraction, Ξ, of data to/from the database, depending on the argument σ. These update operations are defined taking into account the compromised philosophy of our approach, and they also invoke the revision function, whenever it is needed. See Definitions 2.38 and 2.46. Definition 2.37 states formally the concept of updates, w.r.t. declarative units. And Definition 2.51 caters for the formalization of the transaction notion.

2.2.3 The Notion of Integrity

We use a notion of database integrity in order to extend the classical notion of getting consistency. By defining integrity constraints on the database, we can customize the notion of consistency w.r.t. the needs of the application area. In general, integrity constraints are formulae intended to be inconsistent. In this formalization, we have restricted the integrity constraints to be declarative units of the form, $\mathrm{P} : \bigwedge_{i=1}^{n} A_i \rightarrow \bot$, where each A_i is a proposition or its negation. So, these integrity constraints extend the notion of classical consistency, such that if $\gamma : \bigwedge_{i=1}^{n} A_i$ can be derived in the labelled database $\mathcal{D}$, for any label γ, then $\mathcal{D}$ is inconsistent.

Our approach requires that integrity checkings be carried out everytime the database suffers a modification. Given that initially a CIU_{LDS} database is assumed to be consistent, we have that whenever the database becomes inconsistent, the integrity constraint violation is due to the update being performed on the database. However, under the compromised philosophy of the approach, an integrity constraint violation does not have the same restrictive weight that it usually has in conventional databases. We do not view the update input which causes integrity constraint violation as a totally banned input option. Actually, we allow its consistent consequences to be added to the database. Proof theoretically, a CIU_{LDS} database is consistent w.r.t. to the set of integrity constraints I, if it does not derive $\bot$ via the application of the inference rules. Semantically speaking, given a database, the problem of proving its consistency w.r.t. the set of integrity constraints I, is solved by proving that all the declarative units $\gamma : \bigwedge_{i=1}^{n} A_i$ which are premises of the ones in I, are not satisfiable in all the models of the database.

In this work, we distinguish between two kinds of integrity constraint violation, referred to as *direct* and *indirect*. We say that an input directly violates an integrity constraint, if we can derive bottom from the set of integrity constraints, when that input is added to it, in the set-theoretical sense. And we say that an input indirectly violates an integrity constraint, if it does not comply with the previous case, and if it causes the database to become inconsistent w.r.t. the set of integrity constraints, when it is added to it. Example 2.4 illustrates this notion. Definitions 2.12 and 2.29 state formally the notion of integrity constraints and the notion of a consistent CIU_{LDS} database, respectively.

Example 2.4

Let us consider the database $\mathcal{D}_1$ *of example 2.1, and the update request* $U(\mathcal{D}_1, \mathrm{U}_+, \mathrm{E}_2 : C)$ *of example 2.2. If the declarative unit* $\mathrm{E}_2 : C$ *is added to* $\mathcal{D}_1$ *as requested, the integrity constraint* $\mathrm{P}_1 : A \wedge C \rightarrow \bot$ *will be violated. Hence, the declarative unit* $\mathrm{I}_4 : \bot$ *will be derivable, since the declarative unit* $\mathrm{E}_1 : A$ *is in* $\mathcal{D}_1$ *, and the updated* $\mathcal{D}_1$ *will become inconsistent. This indicates that the resulting database needs to be revised. In this case, we say that the input* $\mathrm{E}_2 : C$ *violates indirectly some integrity constraints in* $\mathcal{D}_1$.

•

2.2.4 The Derivation Mechanisms

The proof system defined for CIU_{LDS} plays an important part in the reconciling notion of our approach. It is given by a set of inference rules; some labelling conditions, which have to be satisfied by the inference rules in order to define the labelling propagation in the derivable declarative units; the notion of proof of a declarative unit; and the notion of the system's consequence relation.

In CIU_{LDS} the notion of consequence is stated as a binary relation between a database and a declarative unit, denoted as $\mathcal{D} \vdash_{CIU_{LDS}} \gamma : \alpha$, (or $\mathcal{D} \vdash \gamma : \alpha$ for short). The intended meaning is to determine via the derivation mechanisms, if a given declarative unit $\gamma : \alpha$ is derivable from a labelled database $\mathcal{D}$. That is, if we can exhibit a proof of $\gamma : \alpha$, denoted as $\rho[\gamma : \alpha]$, from $\mathcal{D}$. Definitions 2.28 and 2.27 state formally the notions of consequence and proof, respectively.

This formalization considers a convenient subset of the set of inference rules relative to each connective defined in the language, presented in the natural deduction style. In the course of their formal definitions, more details are given about them. We also justify why some of the connectives do not present correspondent inference rules in CIU_{LDS}. Definition 2.26 states the general notion of inference rules in CIU_{LDS}, and Definitions 2.30, 2.31, 2.32 2.33, 2.34, 2.35, and 2.36 state formally the inference rules defined for the system w.r.t. the labelling conditions.

The concept of labelling propagation conditions is used to monitor the application of inference rules and the basic update operations. We denote a labelling condition with $\varphi_{\mathrm{x}}(\gamma_1, \gamma_2)$, where the subscript $_{\mathrm{x}}$ identifies the inference rule or the database operation that it stands for, and γ_1 and γ_2 are the labels which should be combined to satisfy the condition φ_{x}. In most cases, the combination of the labels γ_1 and γ_2 is given by the $\mathcal{L}_\gamma$ function $©_{\mathrm{x}}$. So, unless differently specified, we say that the labelling condition $\varphi_{\mathrm{x}}(\gamma_1, \gamma_2)$ holds if $\gamma_1 \, ©_{\mathrm{x}} \, \gamma_2$ defines a label of $\mathcal{L}_\gamma$. Then, the algebra of labels is simply defined as the set of all the labelling conditions of the form $\varphi_{\mathrm{x}}(\gamma_1, \gamma_2)$, which hold in our system. Definitions 2.23 and 2.24 formally present the notions of CIU_{LDS} labelling conditions, and the algebra of labels, respectively.

Remark 2.1

Alternatively, we could also have defined a sort of resource labelling system, which would , for instance, annotate on the labels of the formulae the proof-steps taken or the labels of the premises used to obtain them. In such a system, one could define the constraints on the way the formulae were derived. That is, one could say that a labelled formula $\gamma : \alpha$ would only be accepted as derivable if the label γ did not involve any label of type P*, addressing protected formulae, for instance, or any other particular formula type. It would also be possible to recognize non-supported consequences as derived formulae which carried labels in γ, whose wff are available from the database any longer. Such a formalization would be more general than the one presented in this section. However, it would also need a more elaborated algebra of labels, in order to cater for the labelling propagation and to control the derivation mechanisms employed.*

•

2.2.5 The Reconciling Notion

In CIU_{LDS}, the notion of reconciling conflicting updates with the underlying database in follows the compromised reasoning policies for updates described previously. This notion is mainly represented by the compromised revision function, which we denote here as $\odot$, and by the compromised contraction function, denoted by the operator Ξ.

When we get an input request $U(\mathcal{D}, \mathrm{u}_+, \gamma : \alpha)$, the basic operation of conditional inclusion of a declarative unit into a labelled database is invoked. This function is denoted by the operation $\biguplus$. We define that $\biguplus(\mathcal{D}, \gamma : \alpha) = \mathcal{D}'$, also written as $\mathcal{D} \biguplus \gamma : \alpha = \mathcal{D}'$, such that:

$$\mathcal{D}' = \begin{cases} \mathcal{D} \bigcup \{\gamma : \alpha\} & \text{if } \Delta_{\mathcal{D}}, \gamma : \alpha \nvdash \bot; \\ \mathcal{D} \odot \gamma : \alpha & \text{otherwise.} \end{cases}$$

By $\mathcal{D} \bigcup \{\gamma : \alpha\}$, we mean that the declarative unit $\gamma : \alpha$ is added to $\Delta_{\mathcal{D}}$, such that it takes the highest position in the ordering $\preccurlyeq$.

In the case that a revision applies within the operation $\biguplus$, we have the two following possibilities:

1. $I, \gamma : \alpha \vdash \bot$. This means that the declarative unit violates the integrity constraints in $\mathcal{D}$ directly. In this case, $\gamma : \alpha$ is not allowed to be inserted in $\Delta_{\mathcal{D}}$. However, the compromised revision function takes care of including in the revised database, all the consistent consequences of $\gamma : \alpha$ w.r.t. $\mathcal{D}$, as a compromised solution.

2. $\Delta_{\mathcal{D}}, \gamma : \alpha \vdash \bot$ when $I, \gamma : \alpha \nvdash \bot$. In this case, a revision applies in order to accomodate $\gamma : \alpha$ in the resulting database and to preserve consistency. As defined in the policies of our approach, the compromised revision allows for the consistent consequences of retracted declarative units to remain available in the resulting database. The ordering $\preccurlyeq$ on $\Delta_{\mathcal{D}}$ facilitates the process of choosing which declarative unit to discard, in order to regain consistency.

When the input request is $U(\mathcal{D}, \mathrm{U}_-, \gamma : \alpha)$, the operation Ξ is invoked. Ξ retracts a declarative unit from a labelled database in a compromised way. That is, the compromised contraction function, written as $\Xi(\mathcal{D}, \gamma : \alpha) = \mathcal{D}'$ or $\mathcal{D}' = \mathcal{D}\ \Xi\ \gamma : \alpha$, retracts the existing declarative unit $\gamma : \alpha$ from $\Delta_{\mathcal{D}}$, and inserts to it the consequences of $\gamma : \alpha$ w.r.t. $\mathcal{D}$ as a compromised solution, provided that $\mathcal{D}' \nvdash \gamma : \alpha$. In this case, a revision is not needed, because there is no chance that $\mathcal{D}\ \Xi\ \gamma : \alpha \vdash \bot$, since we always consider that $\mathcal{D}$ is initially consistent.

The compromised retraction is a very straightforward operation. Basically, if the declarative unit to be retracted is present in the database, it deletes it and preserves its consistent consequences as non-supported consequences. Otherwise, no operation is performed on the database.

The operations $\biguplus$ for data inclusion, and Ξ for data retraction, can be described as algorithms which are defined on top of the notion of the consequence relation. That is, the definition of these operations make use of the proof procedure which presents the consequence relation $\vdash$.

The consequences which those operations add to the database, without support from their premises, carry the non-supported label type and are subject to continuous checking by the proof system, as the database is further modified. We can say that the derivation mechanism of our system is sensible to the presence of the non-supported declarative units. By this we mean that it applies some restrictions, in the case that a non-supported declarative unit is involved in a derivation process. Further in this section, we make this point clearer.

Below, we give more details and examples about the conditional additional operation of declarative units, and the compromised revision.[12]

- Given a labelled database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, and an update request $U(\mathcal{D}, \mathrm{U}_+, \gamma_1 : \alpha)$, consider that $I, \gamma_1 : \alpha \nvdash \bot$. If $\gamma_2 : \alpha \in \Delta_{\mathcal{D}}$, then the resulting database will have the declarative unit $\gamma_1 : \alpha$ replacing $\gamma_2 : \alpha$, and for all other declarative units x in $\Delta_{\mathcal{D}}$, $x \preccurlyeq \gamma_1 : \alpha$.

Example 2.5

Let us consider a labelled database $\mathcal{D}_2$, with the following set $\Delta_{\mathcal{D}_2}$:

$$
\begin{array}{ll}
(1) & \mathrm{I}_1 : A \rightarrow B \\
(2) & \mathrm{I}_2 : C \rightarrow D \\
(3) & \mathrm{P}_1 : A \wedge C \rightarrow \bot \\
(4) & \mathrm{E}_1 : A \\
(5) & \mathrm{N}_1 : D
\end{array}
$$

and the update request $U(\mathcal{D}_2, \mathrm{U}_+, \mathrm{E}_2 : D)$. So, according to our defined notion of updates, the non-supported declarative unit $\mathrm{N}_1 : D$ is replaced by the declarative unit $\mathrm{E}_2 : D$.

[12] In most of the examples in this section, the ordering $\preccurlyeq$ is omitted on purpose, since it does not take a central role.

- Given an update request $U(\mathcal{D},\ \mathrm{U}_+,\ \gamma_1 : \alpha)$, such that $I,\ \gamma_1 : \alpha \nvdash \bot$. If $\gamma_2 : \neg\alpha \in \Delta_{\mathcal{D}}$, the resulting database will have the declarative unit $\gamma_1 : \alpha$ added to it, and the declarative unit $\gamma_2 : \neg\alpha$ retracted from it, via our compromised notion of retraction.

Example 2.6

Let us consider a labelled database $\mathcal{D}_3$, with the following set $\Delta_{\mathcal{D}_3}$:

$$\begin{array}{ll} (1) & \mathrm{E}_1 : A \\ (2) & \mathrm{E}_2 : \neg B \\ (3) & \mathrm{I}_1 : \neg B \rightarrow C \end{array}$$

and the update request $U(\mathcal{D}_3,\ \mathrm{U}_+,\ \mathrm{E}_3 : B)$. In this case, the declarative unit $\mathrm{E}_2 : \neg B$ is replaced by the $\mathrm{E}_3 : B$, and the declarative unit $\mathrm{N}_1 : C$ is added to the revised database due to the compromised retraction of (2).

- Given an update request $U(\mathcal{D},\ \mathrm{U}_+,\ \gamma : \alpha)$, if $I,\ \gamma : \alpha \nvdash \bot$ and $\Delta_{\mathcal{D}},\ \gamma : \alpha \vdash \bot$, then the database needs revision to accomodate the insertion of the declarative unit $\gamma : \alpha$.

Example 2.7

Let us consider the labelled database $\mathcal{D}_1$ of example 2.1, and the update request $U(\mathcal{D}_1,\ \mathrm{U}_+,\ \mathrm{E}_2 : C)$. When the declarative unit $\mathrm{E}_2 : C$ is added to $\Delta_{\mathcal{D}_1}$ as requested, there is an integrity constraint violation. This invokes the compromised revision function. Since the incoming data has priority over the existing data in the database, we will end up retracting $\mathrm{E}_1 : A$ in order to be able to incorporate $\mathrm{E}_2 : C$ into $\mathcal{D}_1$. Hence, the resulting database after the compromised revision is the following:

$$\begin{array}{ll} (1) & \mathrm{I}_1 : A \rightarrow B \\ (2) & \mathrm{I}_2 : C \rightarrow D \\ (3) & \mathrm{P}_1 : A \wedge C \rightarrow \bot \\ (4) & \mathrm{E}_2 : C \\ (5) & \mathrm{N}_1 : B \end{array}$$

It is important to notice in the example above, that the declarative unit (5) would not be derivable from the resulting database, had the update operation been a conventional one, without any embedded reconciling revision notion for conflicting updates.

Definitions 2.38 and 2.46 state formally the notions of conditional addition, and compromised retraction of a declarative unit to/from a CIU_{LDS} database, repectively. The revision procedure invoked by those operations is presented in section 2.6.4.

In section 2.7 we present some properties of the system CIU_{LDS}, concerning its consequence relation, the basic update operations, the revision function, and other general features of the system.

2.3 CIU_{LDS} Basic Definitions

We present in this section the basic definitions concerning the CIU_{LDS} formalization .

Definition 2.1 (Language $\mathcal{L}$)

$\mathcal{L}$ is a propositional logical language composed of:

- *an alphabet which is expressed by a countable*[13] *number of propositional letters, $A,\ B,\ C,\ D, \cdots$ with or without subscript, including $\top$ and $\bot$;*[14]

[13] By countable we mean finite or enumerable.

[14] We consider $\top$ and $\bot$ as distinguished propositions of $\mathcal{L}$, meaning true and false, respectively.

- *the logical connectives* $\neg$, $\wedge$, *and* $\rightarrow$;
- *the punctuation symbols* ' ('; ') '; *and* ' , '.

□

Definition 2.2 (Well formed formulae of $\mathcal{L}$)

Given a propositional logical language $\mathcal{L}$, *the wff of* $\mathcal{L}$ *are obtained as follows:*

- *If* α *is a propositional letter, then* α *is a wff;*
- *If* α *is a wff, then* $\neg\alpha$ *is a wff;*
- *If* α_1 *and* α_2 *are wffs, then* $\alpha_1 \wedge \alpha_2$ *is a wff;*
- *If* α_1 *and* α_2 *are propositional letters or their negation, and* β *is a propositional letter, then* $\alpha_1 \wedge \alpha_2 \rightarrow \beta$ *is a wff.*
- *The only wff of* $\mathcal{L}$ *are those obtained by finite applications of the items above.*

□

Definition 2.3 (Literal Formula)

Let α *denote an atomic proposition*[15] *of the logical language* $\mathcal{L}$. *A* literal formula, *or just literal for short, is a wff of* $\mathcal{L}$ *defined as* α *or* $\neg\alpha$.

□

Definition 2.4 (Clausal Formula)

Given the logical language $\mathcal{L}$, *a clausal formula*[16] *is any wff of the form* $\bigwedge_{i=1}^{n} \alpha_i \rightarrow \beta$, *where each* α_i *is a literal of* $\mathcal{L}$, *and* β *is an atomic proposition of* $\mathcal{L}$, *including* $\bot$.

□

Definition 2.5 (Language $\mathcal{L}_\gamma$)

Let the language of the labels, denoted by $\mathcal{L}_\gamma$, *be composed of:*

- *a finite set of types* T, *where* $T = \{\mathrm{E}, \mathrm{I}, \mathrm{P}, \mathrm{N}\}$, *in which each type is a list of constants as follows:*
  - $\mathrm{E} = \mathrm{E}_1, \mathrm{E}_2, \cdots, \mathrm{E}_k$;
  - $\mathrm{I} = \mathrm{I}_1, \mathrm{I}_2, \cdots, \mathrm{I}_l$;
  - $\mathrm{P} = \mathrm{P}_1, \mathrm{P}_2, \cdots, \mathrm{P}_m$;
  - $\mathrm{N} = \mathrm{N}_1, \mathrm{N}_2, \cdots, \mathrm{N}_n$;
- *the binary function symbols* '$©_{\cup}$', '$©_{\Xi}$', '$©_{\wedge \mathrm{I}}$', '$©_{\rightarrow \mathrm{I}}$', '$©_{\rightarrow \mathrm{E}}$', '$©_{\bot \mathrm{I}}$';
- *the punctuation symbols* '('; ')'; '$\langle$'; '$\rangle$'; *and* ','.

□

Definition 2.6 (Terms of $\mathcal{L}_\gamma$)

Given a labelling language $\mathcal{L}_\gamma$, *the terms of* $\mathcal{L}_\gamma$, *denoted as* t_γ, *are obtained as follows:*

- *If* $t_\gamma \in \mathrm{Z}$ *where* Z *is a type in* T, *then* t_γ *is a term of* $\mathcal{L}_\gamma$;

[15] We sometimes refer to propositional letters as atomic propositions or atomic formulae.
[16] Also called Horn clause.

- *If* $t_{\gamma 1}$ *and* $t_{\gamma 2}$ *are terms of* $\mathcal{L}_\gamma$, *then* $t_{\gamma 1}$ $©_X$ $t_{\gamma 2}$ *is also a term of* $\mathcal{L}_\gamma$, *where* $X \in \{\uplus, \Xi, \text{CR}, \wedge\text{I}, \rightarrow \text{I}, \rightarrow \text{E}, \perp\text{I}\}$;
- *The only terms of* $\mathcal{L}_\gamma$ *are those obtained by the items above.*

□

Definition 2.7 (Labels)

Given a labelling language $\mathcal{L}_\gamma$, *a label* γ *is a term of* $\mathcal{L}_\gamma$, *such that either* $\gamma \in Z$ *where* $Z \in T$, *or* γ *is a term* t_γ *which results from the application of a function* $©_X$, *where* $X \in \{\uplus, \Xi, \text{CR}, \wedge\text{I}, \rightarrow \text{I}, \rightarrow \text{E}, \perp\text{I}\}$, *such that* t_γ *is a defined value in* $©_X$, *that is* t_γ *belongs to a type in* T.

□

Remark 2.2

The labels of $\mathcal{L}_\gamma$ *compose a subset of the set of terms of* $\mathcal{L}_\gamma$, *since not every term of the type* $t_{\gamma 1}$ $©_X$ $t_{\gamma 2}$, *returns a label as a result.*

•

Definition 2.8 (CIU_{LDS} Language $\mathcal{L}_{CIU}$)

Given a language of labels $\mathcal{L}_\gamma$ *and a logical language* $\mathcal{L}$, *the* CIU_{LDS} *language, denoted as* $\mathcal{L}_{CIU}$, *is defined as the ordered pair:* $\langle \mathcal{L}_\gamma, \mathcal{L} \rangle$.

□

Definition 2.9 (Declarative Unit)

Given the language $\mathcal{L}_{CIU}$, *a* declarative unit *is a labelled formula of the form* $\gamma : \alpha$, *where* γ *is a label of the language* $\mathcal{L}_\gamma$, *and* α *is either a literal formula, or a conjunction of literals, or a clausal formula of* $\mathcal{L}$.

□

Remark 2.3

In the declarative unit defined above, the γ *label type will be chosen, according to the formula* α *it is qualifying. The label types are either* E, I, P, *or* N, *as defined in* $\mathcal{L}_\gamma$. *A label of type '* E *' is supposed to qualify an explicitly stored formula in the database. A label of type '* I *' qualifies derived and clausal formulae. A label of type '* P *' is supposed to qualify a protected formula, and a label of type '* N *' qualifies a non-supported formula in the database. This notion of non-supported formulae qualifies the formulae which are no longer derivable, after compromised updates have taken place in the database.*

•

We define below the different types of declarative units which are considered in our database.

Definition 2.10 (Extensional Data)

Given the language $\mathcal{L}_{CIU}$, extensional data *are declarative units of the form* $\gamma : \alpha$, *where* $\gamma \in \text{E}$,[17] *and the formula* α *is either a literal formula, or a conjunction of literal formulae.*

□

Definition 2.11 (Intensional Data)

Given the language $\mathcal{L}_{CIU}$, intensional data *are declarative units of the form* $\gamma : \alpha$, *where* $\gamma \in \text{I}$, *and the formula* α *is either a literal formula, a conjunction of literal formulae, or a clausal formula whose consequence is different from* $\perp$.

[17] We will use the symbol $\in$ to denote both set-membership and list-membership.

□

Remark 2.4

In the above definition of intensional data, if the formula α is a literal, or a conjunction of literals, it means that the declarative unit $\gamma : \alpha$ can be derived [18] *by the system, from the given database.*

•

Definition 2.12 (Protected Formulae - Integrity Constraints)

Given the language $\mathcal{L}_{CIU}$, an integrity constraint [19] *is defined as a declarative unit of the form $\gamma : \alpha$, where $\gamma \in$ P , and α is a clausal formula of the form: $\bigwedge_{i=1}^{n} \beta_i \rightarrow \bot$, where each β_i is a literal formula of $\mathcal{L}$.*

□

Definition 2.13 (Non-supported Formulae)

Given the language $\mathcal{L}_{CIU}$, a non-supported formula *is a declarative unit of the form $\gamma : \alpha$, where $\gamma \in$ N , and α is either an atomic formula, or a conjunction of atomic formula of $\mathcal{L}$.*

□

The extensional declarative units qualify explicitly stored data in the database. The intensional declarative units qualify implicitly stored data in the database, i.e., rules of the database presented in the clausal form, or derivable formulae. The non-supported declarative units qualify the compromised consequences of our approach, which are then stored explicitly under the N label type. These declarative units result from the CIU_{LDS} reconciling revision notion, when a conflicting update occurs. In general, the "non-supported" status of those compromised consequences can be changed into an "intensional-data" or "extensional-data" status, via an update performance or via the application of the inference rules, under some given conditions.

Definition 2.14 (CIU_{LDS} Database)

A CIU_{LDS} database, *also called labelled database, denoted as $\mathcal{D}$, is the tuple $\langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, where $\Delta_{\mathcal{D}}$ is a finite set of declarative units $\Delta_{\mathcal{D}} = \{\gamma_1 : \alpha_1, \gamma_2 : \alpha_2, \cdots, \gamma_n : \alpha_n\}$, and $\preccurlyeq$ is an ordering on the declarative units of $\Delta_{\mathcal{D}}$. The declarative units in $\Delta_{\mathcal{D}}$ can be extensional data; intensional data; integrity constraints; and non-supported formulae.*

□

Remark 2.5

Concerning the set of declarative units in the initially described database, we assume that different labels will always refer to different wff's.

•

Below we define the ordering $\preccurlyeq$, as well as the special conditions which it applies to some declarative units in $\Delta_{\mathcal{D}}$.

Definition 2.15 (The Ordering $\preccurlyeq$)

Given a set $\Delta_{\mathcal{D}} = \{\gamma_1 : \alpha_1, \gamma_2 : \alpha_2, \cdots, \gamma_n : \alpha_n\}$, let $\preccurlyeq$ be a pre-order on $\Delta_{\mathcal{D}}$, such that the following conditions are satisfied:

- $\forall\ \gamma_i : \alpha_i \in \Delta_{\mathcal{D}},\ \gamma_i : \alpha_i \preccurlyeq \gamma_i : \alpha_i$.

[18] The formal notion of derivability for CIU_{LDS} is stated in the definition 2.28, further in this section.

[19] So far, we are considering only integrity constraints as protected formulae. However, we can think of extending this notion to other formulae as well, depending on the application's requirements.

- $\forall\ \gamma_i : \alpha_i,\ \gamma_j : \alpha_j,\ \gamma_k : \alpha_k \in \Delta_{\mathcal{D}}$, *if* $\gamma_i : \alpha_i \preccurlyeq \gamma_j : \alpha_j$ *and* $\gamma_j : \alpha_j \preccurlyeq \gamma_k : \alpha_k$, *then* $\gamma_i : \alpha_i \preccurlyeq \gamma_k : \alpha_k$.
- $\forall\ \gamma_i : \alpha_i,\ \gamma_j : \alpha_j \in \Delta_{\mathcal{D}}$, *if* $\gamma_i \in \mathrm{P}$ *and* $\gamma_j \in \mathrm{P}$, *then* $\gamma_i : \alpha_i \preccurlyeq \gamma_j : \alpha_j$ *and* $\gamma_j : \alpha_j \preccurlyeq \gamma_i : \alpha_i$, *also written as* $\gamma_i : \alpha_i \ \ \gamma_j : \alpha_j$.
- $\forall\ \gamma_i : \alpha_i,\ \gamma_j : \alpha_j \in \Delta_{\mathcal{D}}$, *if* $\gamma_i \in \mathrm{P}$ *and* $\gamma_j \notin \mathrm{P}$, *then* $\gamma_i : \alpha_i$ *is not comparable to* $\gamma_j : \alpha_j$ *via* $\preccurlyeq$.

□

The two first conditions in the definition above, represent respectively the properties of reflexivity and transitivity of $\preccurlyeq$, since it is a pre-order.

The third condition states the natural concept that the protected data in $\Delta_{\mathcal{D}}$ are equivalent in the ordering. This assumption is justified by the fact that protected data are not supposed to be modified or retracted from $\Delta_{\mathcal{D}}$. Hence, there is no need to have them under an ordering, since we will never have to choose any single declarative unit among them to be retracted from $\Delta_{\mathcal{D}}$.

The last condition places the set of protected formulae in $\Delta_{\mathcal{D}}$ as a distinguished non-related one.

Below, we define the way that the ordering $\preccurlyeq$ propagates to newly inserted declarative units in $\Delta_{\mathcal{D}}$. This propagation notion is based on the consequence relation of the system, which is defined in Section 2.4.

Definition 2.16 (Propagation of the Ordering $\preccurlyeq$)

Given a set $\Delta_{\mathcal{D}}$ *ordered by* $\preccurlyeq$, *assume that the declarative unit* $\gamma_i : \alpha_i$ *is inserted to* $\Delta_{\mathcal{D}}$. *The ordering of* $\Delta_{\mathcal{D}} \bigcup \gamma_i : \alpha_i$ *is then obtained satisfying one of the following conditions:*

- *If* $Z \vdash \gamma_i : \alpha_i$, *for any set* $Z \subseteq \Delta_{\mathcal{D}}$, *such that* Z *is minimal w.r.t.* $\subseteq$, *then* $\gamma_j : \alpha_j \preccurlyeq \gamma_i : \alpha_i$, $\forall \gamma_j : \alpha_j \in Z$.
- *If* $\Delta_{\mathcal{D}} \nvdash \gamma_i : \alpha_i$, *then* $\gamma_j : \alpha_j \preccurlyeq \gamma_i : \alpha_i$, $\forall \gamma_j : \alpha_j \in \Delta_{\mathcal{D}}$.

□

The first condition in the definition above, states the propagation of the ordering $\preccurlyeq$ on the consequences of $\Delta_{\mathcal{D}}$. As a natural dominance dependency, we define that a derived declarative unit has a higher position in the ordering than any of the declarative units involved in its derivation.

The second condition states the ordering of the expanded set $\Delta_{\mathcal{D}}$, by a new declarative unit which is not a consequence of $\Delta_{\mathcal{D}}$. In this case, we assume that the new declarative unit gets the highest priority in the ordering.

Below we define the functions '$©_{\uplus}$', '$©_{\Xi}$', '$©_{\wedge \mathrm{I}}$', '$©_{\rightarrow \mathrm{I}}$', '$©_{\rightarrow \mathrm{E}}$', and '$©_{\perp \mathrm{I}}$', based on different combinations of label types given in $\mathcal{L}_{\gamma}$. The functions ' $©_{\mathrm{X}}$ ', where $\mathrm{X} \in \{\uplus, \Xi, \wedge \mathrm{I}, \rightarrow \mathrm{I}, \rightarrow \mathrm{E}, \perp \mathrm{I}\}$, combine two label types of $\mathcal{L}_{\gamma}$ and return another label type as result, in the case that the combination succeeds. Given two labels γ_1 and γ_2, $\gamma_1\ ©_{\mathrm{X}}\ \gamma_2$ returns another label γ_3 as a result, if γ_1 can be combined to γ_2 under the operation or inference rule X. If we assume that L is the set of all terms given by $\mathcal{L}_{\gamma}$, then the binary functions ' $©_{\mathrm{X}}$ ', are defined from $L \times L$ to L. These functions are required by the labelling conditions, in the definition of the update operations and of some CIU_{LDS} inference rules. Figure 1 illustrates the results obtained by the $©_{\mathrm{X}}$ functions.

Definition 2.17 ($©_{\uplus}$)

Given the function symbol $©_{\uplus}$, *and the labels* γ_1, *and* γ_2 *of the language of labels* $\mathcal{L}_{\gamma}$, *we assume that* $\gamma_1\ ©_{\uplus}\ \gamma_2$, *is given as follows:*

- *If* $\gamma_1 \in \mathrm{E}$ *and* $\gamma_2 \in \mathrm{E}$, *then* $\gamma_1\ ©_{\uplus}\ \gamma_2 \in \mathrm{E}$;

- *If* $\gamma_1 \in$ I *and* $\gamma_2 \in$ I, *then* $\gamma_1 \textcircled{c}_{\uplus} \gamma_2 \in$ I;
- *If* $\gamma_1 \in$ E *and* $\gamma_2 \in$ N, *then* $\gamma_1 \textcircled{c}_{\uplus} \gamma_2 \in$ E;
- *For all the other cases not specified above,* $\gamma_1 \textcircled{c}_{\uplus} \gamma_2$ *is not defined.*[20]

□

γ_1	γ_2	$\gamma_1 \textcircled{c}_{\uplus} \gamma_2$	$\gamma_1 \textcircled{c}_{\Xi} \gamma_2$	$\gamma_1 \textcircled{c}_{\wedge_I} \gamma_2$	$\gamma_1 \textcircled{c}_{\rightarrow_I} \gamma_2$	$\gamma_1 \textcircled{c}_{\rightarrow_E} \gamma_2$	$\gamma_1 \textcircled{c}_{\perp_I} \gamma_2$
E	E	E	-	E	I	-	I
E	I	-	N	I	I	I	I
E	P	-	-	-	-	I	-
E	N	E	N	N	I	-	I
I	E	-	-	I	I	-	I
I	I	I	-	I	I	I	I
I	P	P	-	-	-	I	-
I	N	-	-	N	I	-	I
P	E	-	-	-	-	-	-
P	I	P	-	-	-	-	-
P	P	P	-	-	-	-	-
P	N	-	-	-	-	-	-
N	E	E	-	N	I	-	I
N	I	-	N	N	I	N	I
N	P	-	-	-	-	I	-
N	N	N	N	N	I	-	I

Figure 1: Labelling functions.

Definition 2.18 ($\textcircled{c}_{\Xi}$)

Given the function symbol $\textcircled{c}_{\Xi}$, *and the labels* γ_1, *and* γ_2 *of the language of labels* $\mathcal{L}_{\gamma}$, $\gamma_1 \textcircled{c}_{\Xi} \gamma_2$, *is given as follows:*

- *If* $\gamma_1 \in$ E *and* $\gamma_2 \in$ I, *then* $\gamma_1 \textcircled{c}_{\Xi} \gamma_2 \in$ N;
- *If* $\gamma_1 \in$ E *and* $\gamma_2 \in$ N, *then* $\gamma_1 \textcircled{c}_{\Xi} \gamma_2 \in$ N;
- *If* $\gamma_1 \in$ I *and* $\gamma_2 \in$ I, *then* $\gamma_1 \textcircled{c}_{\Xi} \gamma_2 \in$ N;

[20] Formally speaking, in the case that $\gamma_1 \textcircled{c}_{\chi} \gamma_2$ is not defined, we would have to return a dummy symbol as a non-defined result. However, in order to avoid cumbersome notation, we do not include such symbol in the definitions of the $\textcircled{c}_{\chi}$ labelling functions.

- *If* $\gamma_1 \in \mathrm{N}$ *and* $\gamma_2 \in \mathrm{I}$, *then* $\gamma_1 \textcircled{c}_{\Xi} \gamma_2 \in \mathrm{N}$;
- *If* $\gamma_1 \in \mathrm{N}$ *and* $\gamma_2 \in \mathrm{N}$, *then* $\gamma_1 \textcircled{c}_{\Xi} \gamma_2 \in \mathrm{N}$;
- *For all the other cases not specified above,* $\gamma_1 \textcircled{c}_{\Xi} \gamma_2$ *is not defined.*

□

Remark 2.6

The function $\textcircled{c}_{\Xi}$ *always returns a label of type* N *for those cases defined.*

•

Definition 2.19 ($\textcircled{c}_{\wedge \mathrm{I}}$)

Given the function symbol $\textcircled{c}_{\wedge \mathrm{I}}$, *and the labels* γ_1, *and* γ_2 *of the language of labels* $\mathcal{L}_\gamma$, *we assume that* $\gamma_1 \textcircled{c}_{\wedge \mathrm{I}} \gamma_2$, *is given as follows:*

- $\gamma_1 \textcircled{c}_{\wedge \mathrm{I}} \gamma_2 = \gamma_2 \textcircled{c}_{\wedge \mathrm{I}} \gamma_1$;
- *If* γ_1 *and* γ_2 *are of the same label type* Z, *where* $\mathrm{Z} \in \{\mathrm{E}, \mathrm{I}, \mathrm{N}\}$, *then* $\gamma_1 \textcircled{c}_{\wedge \mathrm{I}} \gamma_2 \in \mathrm{Z}$;
- *If* $\gamma_1 \in \mathrm{N}$, *then* $\gamma_1 \textcircled{c}_{\wedge \mathrm{I}} \gamma_2 \in \mathrm{N}$, *where* $\gamma_2 \in \{\mathrm{E}, \mathrm{I}, \mathrm{N}\}$;
- *If* $\gamma_1 \in \mathrm{E}$ *and* $\gamma_2 \in \mathrm{I}$, *then* $\gamma_1 \textcircled{c}_{\wedge \mathrm{I}} \gamma_2 \in \mathrm{I}$;
- *For all the other cases not specified above,* $\gamma_1 \textcircled{c}_{\wedge \mathrm{I}} \gamma_2$ *is not defined.*

□

Definition 2.20 ($\textcircled{c}_{\rightarrow \mathrm{I}}$)

Given the function symbol $\textcircled{c}_{\rightarrow \mathrm{I}}$, *and the labels* γ_1, *and* γ_2 *of the language of labels* $\mathcal{L}_\gamma$, *we assume that* $\gamma_1 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2$, *is given as follows:*

- $\gamma_1 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2 = \gamma_2 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_1$;
- *If* $\gamma_1 \in \mathrm{E}$ *and* $\gamma_2 \in \mathrm{E}$, *then* $\gamma_1 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2 \in \mathrm{I}$;
- *If* $\gamma_1 \in \mathrm{E}$ *and* $\gamma_2 \in \mathrm{I}$, *then* $\gamma_1 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2 \in \mathrm{I}$;
- *If* $\gamma_1 \in \mathrm{E}$ *and* $\gamma_2 \in \mathrm{N}$, *then* $\gamma_1 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2 \in \mathrm{I}$;
- *If* $\gamma_1 \in \mathrm{I}$ *and* $\gamma_2 \in \mathrm{I}$, *then* $\gamma_1 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2 \in \mathrm{I}$;
- *If* $\gamma_1 \in \mathrm{I}$ *and* $\gamma_2 \in \mathrm{N}$, *then* $\gamma_1 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2 \in \mathrm{I}$;
- *If* $\gamma_1 \in \mathrm{N}$ *and* $\gamma_2 \in \mathrm{N}$, *then* $\gamma_1 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2 \in \mathrm{I}$;
- *For all the other cases not specified above,* $\gamma_1 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2$ *is not defined.*

□

Remark 2.7

The function $\textcircled{c}_{\rightarrow \mathrm{I}}$ *is commutative w.r.t. the label types, and it always returns a label of type* I *for those cases defined.*

•

Definition 2.21 ($\textcircled{c}_{\rightarrow \mathrm{E}}$)

Given the function symbol $\textcircled{c}_{\rightarrow \mathrm{E}}$, *and the labels* γ_1, *and* γ_2 *of the language of labels* $\mathcal{L}_\gamma$, *we assume that* $\gamma_1 \textcircled{c}_{\rightarrow \mathrm{E}} \gamma_2$, *is given as follows:*

- *If* $\gamma_1 \in \mathrm{E}$ *and* $\gamma_2 \in \mathrm{I}$, *then* $\gamma_1 \textcircled{c}_{\rightarrow \mathrm{E}} \gamma_2 \in \mathrm{I}$;

- *If* $\gamma_1 \in$ E *and* $\gamma_2 \in$ P, *then* $\gamma_1 \copyright_{\rightarrow E} \gamma_2 \in$ I;
- *If* $\gamma_1 \in$ I *and* $\gamma_2 \in$ P, *then* $\gamma_1 \copyright_{\rightarrow E} \gamma_2 \in$ I;
- *If* $\gamma_1 \in$ I *and* $\gamma_2 \in$ I, *then* $\gamma_1 \copyright_{\rightarrow E} \gamma_2 \in$ I;
- *If* $\gamma_1 \in$ N *and* $\gamma_2 \in$ I, *then* $\gamma_1 \copyright_{\rightarrow E} \gamma_2 \in$ N;
- *If* $\gamma_1 \in$ N *and* $\gamma_2 \in$ P, *then* $\gamma_1 \copyright_{\rightarrow E} \gamma_2 \in$ I;
- *For all the other cases not specified above,* $\gamma_1 \copyright_{\rightarrow E} \gamma_2$ *is not defined.*

□

Definition 2.22 ($\copyright_{\perp I}$)

Given the function symbol $\copyright_{\perp I}$, *and the labels* γ_1, *and* γ_2 *of the language of labels* $\mathcal{L}_\gamma$, *we assume that* $\gamma_1 \copyright_{\perp I} \gamma_2$, *is given as follows:*

- $\gamma_1 \copyright_{\perp I} \gamma_2 = \gamma_2 \copyright_{\perp I} \gamma_1$;
- *If* $\gamma_1 \in$ I, *and* $\gamma_2 \in$ E∪I∪N, *then* $\gamma_1 \copyright_{\perp I} \gamma_2 \in$ I;
- *If* $\gamma_1 \in$ E *and* $\gamma_2 \in$ E, *then* $\gamma_1 \copyright_{\perp I} \gamma_2 \in$ I;
- *If* $\gamma_1 \in$ E *and* $\gamma_2 \in$ N, *then* $\gamma_1 \copyright_{\perp I} \gamma_2 \in$ I;
- *If* $\gamma_1 \in$ N *and* $\gamma_2 \in$ N, *then* $\gamma_1 \copyright_{\perp I} \gamma_2 \in$ I;
- *For all the other cases not specified above,* $\gamma_1 \copyright_{\perp I} \gamma_2$ *is not defined.*

□

Remark 2.8

The function $\copyright_{\perp I}$ *is commutative w.r.t. the label types, and it always returns a label of type* I *for those cases defined.*

•

In CIU_{LDS}, inference rules as well as update operations on the database depend on some defined labelling propagation conditions, which compose the algebra of labels. We introduce below the labelling propagation conditions for each inference rule and update operation, which will be defined further in this section.

Notation 2.1

We denote as IR_{CIU} *the set of all the inference rules of* CIU_{LDS}, *such that* $IR_{CIU} = \{$CR, ∧I, ∧E, → I, → E, ¬E, ⊥I*These abbreviations refer to the different inference rules defined in the system. Namely,* CR *represents conditional reflexivity;* ∧I *represents* ∧ *introduction;* ∧E *represents* ∧ *elimination;* → I *represents* → *introduction;* → E *represents* → *elimination;* ¬E *represents* ¬ *elimination; and* ⊥I *represents* ⊥ *introduction.*[21] *We denote as* UP *the set of the update operations of* CIU_{LDS}, *such that* $UP = \{\uplus, \Xi\}$.[22]

•

Definition 2.23 (CIU_{LDS} **Labelling Conditions**)

Given any labels γ_1 *and* γ_2 *of the language* $\mathcal{L}_\gamma$, *let* $\varphi_X(\gamma_1, \gamma_2)$ *be the labelling propagation conditions for* CIU_{LDS}, *where* $X \in IR_{CIU} \bigcup UP$, *such that* $\varphi_X(\gamma_1, \gamma_2)$ *satisfies the following:*

- *For* $X \in \{\uplus, \Xi,$ ∧I, → I, → E, ⊥I$\}$, $\varphi_X(\gamma_1, \gamma_2)$ *holds if* $\gamma_1 \copyright_X \gamma_2$ *returns a label* γ_3, *as in Definition 2.7.*

[21] See Definitions 2.23, 2.31, 2.32, 2.33, 2.34, 2.35, and 2.36.
[22] See Section 2.6.

- *For* $\mathrm{X} = \mathrm{CR}$, $\varphi_{\mathrm{CR}}(\gamma_1, \gamma_2)$ *holds if* $\gamma_1 \in \mathrm{E}$, *and* $\gamma_2 \in \mathrm{I}$;
- *For* $\mathrm{X} = \wedge\mathrm{E}$, $\varphi_{\wedge\mathrm{E}}(\gamma_1, \gamma_2)$ *holds if* $\gamma_1 \in \mathrm{EUIUN}$ *and* γ_2 *belongs to the same label type as* γ_1;
- *For* $\mathrm{X} = \neg\mathrm{E}$, $\varphi_{\neg\mathrm{E}}(\gamma_1, \gamma_2)$ *holds if* γ_1 *and* γ_2 *belong to the same label type* Z, *where* $\mathrm{Z} \in \mathrm{T}$ *in* $\mathcal{L}_\gamma$;
- *For any other case not specified above,* φ_{X} *does not hold.*

□

Definition 2.24 (Algebra of Labels $\mathcal{A}$)

Given the language of labels $\mathcal{L}_\gamma$, *and the labelling conditions* $\varphi_{\mathrm{X}}(\gamma_1, \gamma_2)$, *where* $\mathrm{X} \in IR_{CIU} \bigcup UP$, *let* $\mathcal{A}$ *be the* algebra of labels *in* CIU_{LDS}, *such that* $\mathcal{A}$ *is the following set:* $\mathcal{A} = \{\varphi_{\mathrm{X}}(\gamma_1, \gamma_2) \mid \varphi_{\mathrm{X}}(\gamma_1, \gamma_2)$ *holds*$\}$.

□

Definition 2.25 (CIU_{LDS} Labelled Deductive System)

A Labelled Deductive System CIU_{LDS} *is a tuple* $\langle \mathcal{A}, \mathcal{L}_{CIU}, M_{CIU} \rangle$, *where* $\mathcal{A}$ *is the algebra of labels and* $\mathcal{L}_{CIU}$ *is the system language, and* M_{CIU} *represents the possible deduction and change mechanisms of the system.* M_{CIU} *includes the set of all inference rules and the set of update operations of* CIU_{LDS}.[23]

□

2.3.1 Discussions

In this formalization, the introduction of labels allows to represent explicitly the defined types of the formulae that we can deal with in the database. The labels also control the derivation process via the labelling conditions of the algebra of labels. The labelling functions $©_{\mathrm{X}}$ are tightly related to the derivation mechanisms represented by the inference rules, and to the expected result types of the update operations. They map the possible labels of the inputs for the inference rules and update functions, to their corresponding output label within the resulting declarative unit. The ordering $\preccurlyeq$ on the set of declarative units, allows the system to state the application's priority or relevance on the data being represented. Moreover, $\preccurlyeq$ also facilitates the process of choosing one particular declarative unit among the conflicting ones, when revising the labelled database for a compromised solution. In the next section, we present the derivation mechanism of the system.

[23] See Sections 2.5 and 2.6.

CIU_{LDS} Basic Definitions:

$\mathcal{L}_{CIU}$ is the CIU_{LDS} language, such that $\mathcal{L}_{CIU} = \langle \mathcal{L}_\gamma, \mathcal{L} \rangle$, where $\mathcal{L}_\gamma$ is the language of the labels, and $\mathcal{L}$ is a propositional logical language.

$\gamma : \alpha$ is a *declarative unit* ,
where γ is a label of $\mathcal{L}_\gamma$, and α is either a literal, a conjunction of literals, or a clausal formula of $\mathcal{L}$.

$\mathcal{D} = \langle \Delta_\mathcal{D}, \preccurlyeq \rangle$, is a CIU_{LDS} *database*, where
$\Delta_\mathcal{D} = \{\gamma_1 : \alpha_1, \gamma_2 : \alpha_2, \cdots, \gamma_n : \alpha_n\}$, in which the declarative units can be extensional data; intensional data; integrity constraints; and non-supported formulae; and $\preccurlyeq$ is an ordering on $\Delta_\mathcal{D}$.

Labelling Conditions $\varphi_{\mathrm{X}}(\gamma_1, \gamma_2)$:
For $\mathrm{X} \in \{\uplus, \Xi, \wedge\mathrm{I}, \rightarrow\mathrm{I}, \rightarrow\mathrm{E}, \neg\mathrm{I}, \perp\mathrm{I}\}$, $\varphi_{\mathrm{X}}(\gamma_1, \gamma_2)$ holds if $\gamma_1 \copyright_{\mathrm{X}} \gamma_2$ returns a label γ_3;
For $\mathrm{X} = \mathrm{CR}$, $\varphi_{\mathrm{CR}}(\gamma_1, \gamma_2)$ holds if $\gamma_1 \in \mathrm{E}$, and $\gamma_2 \in \mathrm{I}$;
For $\mathrm{X} = \wedge\mathrm{E}$, $\varphi_{\wedge\mathrm{E}}(\gamma_1, \gamma_2)$ holds if $\gamma_1 \in \mathrm{E} \cup \mathrm{I} \cup \mathrm{N}$ and γ_2 belongs to the same label type as γ_1; For $\mathrm{X} = \neg\mathrm{E}$, $\varphi_{\neg\mathrm{E}}(\gamma_1, \gamma_2)$ holds if γ_1 and γ_2 belong to the same label type Z, where $\mathrm{Z} \in \mathrm{T}$ in $\mathcal{L}_\gamma$; and
For any other case not specified above, φ_{X} does not hold.

Algebra of Labels:

$$\mathcal{A} = \{\varphi_{\mathrm{X}}(\gamma_1, \gamma_2) \mid \varphi_{\mathrm{X}}(\gamma_1, \gamma_2) \text{ holds}\}.$$

A Labelled Deductive System CIU_{LDS}:

$$CIU_{LDS} = \langle \mathcal{A}, \mathcal{L}_{CIU}, M_{CIU} \rangle,$$

where M_{CIU} includes the set of all inference rules and the set of update operations of CIU_{LDS}.

Figure 2: Summary of the CIU_{LDS} Basic Definitions.

2.4 CIU_{LDS} Proof System

The Proof System of CIU_{LDS} is defined taking into account the following definitions:

- The language $\mathcal{L}_{CIU}$, and a labelled database $\mathcal{D}$;
- The set of inference rules;
- The algebra of labels which contains the labelling conditions that are satisfied, in order to define the labelling propagation in the derivable declarative units;
- The notion of proof of a declarative unit; and
- The system's consequence relation w.r.t. a database $\mathcal{D}$ and a particular declarative unit.

For all the definitions to follow, we assume that a CIU_{LDS} is given by $\langle \mathcal{A}, \mathcal{L}_{CIU}, M_{CIU} \rangle$, and that $\mathcal{D}$ is a CIU_{LDS} database.

In CIU_{LDS}, the inference rules are applied to declarative units. Hence, given a database $\mathcal{D}$, an inference rule generates a declarative unit from a set of declarative units. In general terms, an inference rule can be defined as follows.

Notation 2.2

We denote an inference rule as IR, *such that* $\text{IR} \in IR_{CIU}$, *where* $IR_{CIU} = \{\text{CR}, \wedge\text{I}, \wedge\text{E}, \rightarrow\text{I}, \rightarrow\text{E}, \neg\text{E}, \perp\text{I}\}$.

•

Definition 2.26 (CIU_{LDS} Inference Rules)

Given a database $\mathcal{D}$, *an inference rule* IR *is defined, in the general case, as a tuple* $\langle A_{\text{IR}}, \varphi_{\text{IR}}, C_{\text{IR}} \rangle$, *where:*

- A_{IR} *indicates a set of declarative units, in or derived by* $\mathcal{D}$, *used as antecedents, or premises, of the rule;*
- φ_{IR} *denotes the labelling condition which needs to be satisfied by the application of the* IR *inference rule;*
- C_{IR} *represents the declarative unit derived from* A_{IR} *via the inference rule* IR, *provided that* φ_{IR} *holds.*

□

Below, we introduce the notions of a proof in CIU_{LDS}, and of the consequence relation $\vdash_{CIU_{LDS}}$, which determines for an arbitrary database $\mathcal{D}$ and for an arbitrary declarative unit $\gamma : \alpha$, whether $\mathcal{D} \vdash_{CIU_{LDS}} \gamma : \alpha$ holds or not.

Definition 2.27 (Proof)

Given a database $\mathcal{D}$, *and a declarative unit* $\gamma : \alpha$, *a proof of* $\gamma : \alpha$ *from* $\mathcal{D}$, *written* $\rho[\gamma : \alpha]$, *is a pair* $\langle P_\rho, k \rangle$, *where* P_ρ *is a finite sequence of the pairs (or sub-derivations)* A/C, $P_\rho = \{A_1/C_1, A_2/C_2, \cdots, A_n/C_n\}$, *where* $n > 0$, *and each* A_i, *for* $1 \leq i \leq n$, *is a set of declarative units used as premises by a* CIU_{LDS} *inference rule* IR, *in order to reach the consequent* C_i *which is a single declarative unit. The declarative units in* A_i *are either in* $\Delta_{\mathcal{D}}$, *or are derived from* $\Delta_{\mathcal{D}}$, *in which case they include previous consequences* C_i *of* P_ρ. *And* k *is a mapping from the set* $\{1, \cdots, n\}$, *to the set of inference rules* IR_{CIU}, *such that for each* i, *where* $1 \leq i \leq n$, $k(i) = \text{IR}$, *for any* $\text{IR} \in IR_{CIU}$, *and* $A_i/C_i = A_{\text{IR}}/C_{\text{IR}}$.

□

Definition 2.28 (Consequence Relation $\vdash_{CIU_{LDS}}$)

Given a database $\mathcal{D}$, and an arbitrary declarative unit $\gamma : \alpha$, we say that $\gamma : \alpha$ is a consequence of $\mathcal{D}$, denoted by $\mathcal{D} \vdash_{CIU_{LDS}} \gamma : \alpha$, (or $\mathcal{D} \vdash \gamma : \alpha$ for short), if there exists a proof $\rho[\gamma : \alpha]$ from $\mathcal{D}$.

□

Below, we state the notion of a consistent CIU_{LDS} database.

Definition 2.29 (Consistent CIU_{LDS} Database)

Given a database $\mathcal{D}$, we say that $\mathcal{D}$ is consistent *if and only if it does not derive $\bot$,[24], i.e. $\mathcal{D} \nvdash \gamma : \bot$, for any label γ .*

□

2.5 CIU_{LDS} Inference Rules

We define the inference rules for CIU_{LDS}, based on the labelling propagation conditions of the algebra of labels.

Definition 2.30 (Conditional Reflexivity (CR))

Given a database $\mathcal{D}$, for any declarative unit $\gamma : \alpha$, such that $\gamma : \alpha \in \Delta_{\mathcal{D}}$, the inference rule CR *is the tuple $\langle A_{\text{CR}}, \varphi_{\text{CR}}, C_{\text{CR}} \rangle$, where $A_{\text{CR}} = \{ \gamma : \alpha \}$, and $C_{\text{CR}} = \gamma : \alpha$, provided that $\varphi_{\text{CR}}(\gamma, \gamma_i) \in \mathcal{A}$ for any label γ_i, such that $\mathcal{D}' \vdash \gamma_i : \alpha$, where $\mathcal{D}' = \mathcal{D} - \{\gamma : \alpha\}$.[25]*

□

Remark 2.9 *The reflexivity rule is conditioned to φ_{CR}, only in the case that declarative units have different versions within the database.[26] Otherwise, the labelling condition φ_{CR} does not play any relevant role within the* CR *inference rule.*

•

Definition 2.31 ($\wedge$ Introduction ($\wedge$I))

Given a database $\mathcal{D}$, for any pair of declarative units $\gamma_1 : \alpha_1$ and $\gamma_2 : \alpha_2$, where α_1 and α_2 are either literals or conjunction of literals, if there exists a declarative unit $\gamma_3 : \alpha_1 \wedge \alpha_2$, where $\gamma_3 = \gamma_1 \copyright_{\wedge \text{I}} \gamma_2$ and $\varphi_{\wedge \text{I}}(\gamma_1, \gamma_2) \in \mathcal{A}$, then the inference rule $\wedge$I is the tuple $\langle A_{\wedge \text{I}}, \varphi_{\wedge \text{I}}, C_{\wedge \text{I}} \rangle$, where $A_{\wedge \text{I}} = \{ \gamma_1 : \alpha_1, \gamma_2 : \alpha_2 \}$; and $C_{\wedge \text{I}} = \gamma_3 : \alpha_1 \wedge \alpha_2$. Hence, given that $\varphi_{\wedge \text{I}}(\gamma_1, \gamma_2)$ holds:

$$\frac{\gamma_1 : \alpha_1 \qquad \gamma_2 : \alpha_2}{\gamma_3 : \alpha_1 \wedge \alpha_2}$$

□

Definition 2.32 ($\wedge$ Elimination ($\wedge$E))

[24] See remark 2.10 for the cases when $\mathcal{D} \vdash \gamma : \bot$ in CIU_{LDS} .

[25] For simplicity of notation, we consider that α can be taken directly from the declarative unit it belongs, in order to be checked against another formula of $\mathcal{L}$. Formally, this is done via a function f, which ranges from the set of declarative units to the set of wff's of $\mathcal{L}$. f associates with each declarative unit a wff of $\mathcal{L}$, such that: $\forall \gamma : \alpha$, $f(\gamma : \alpha) = \alpha$.

[26] We say that a declarative unit $\gamma : \alpha$ has a different version within a database $\mathcal{D}$, if there exists another declarative unit $\gamma_i : \alpha_i$ either in $\Delta_{\mathcal{D}}$ or derivable from it, where γ and γ_i are not of the same type, and $\alpha_i = \alpha$.

Given a database $\mathcal{D}$, for any declarative unit of the form $\gamma_1 : \alpha_1 \wedge \alpha_2$, the inference rule $\wedge$E is the tuple $\langle A_{\wedge E}, \varphi_{\wedge E}, C_{\wedge E} \rangle$, where $A_{\wedge E} = \{ \gamma_1 : \alpha_1 \wedge \alpha_2 \}$; $\varphi_{\wedge E}(\gamma_1, \gamma_2)$ is the labelling condition; and $C_{\wedge E} = \gamma_2 : \alpha_1$ or $C_{\wedge E} = \gamma_2 : \alpha_2$, where $\varphi_{\wedge E}(\gamma_1, \gamma_2) \in \mathcal{A}$.[27]

$$\frac{\gamma_1 : \alpha_1 \wedge \alpha_2}{\gamma_2 : \alpha_1} \qquad \frac{\gamma_1 : \alpha_1 \wedge \alpha_2}{\gamma_2 : \alpha_2}$$

□

Definition 2.33 ($\rightarrow$ Introduction ($\rightarrow$ I))

Given a database $\mathcal{D}$, for any pair of declarative units $\gamma_1 : \alpha_1$ and $\gamma_2 : \alpha_2$, where α_1 is either a literal or a conjunction of literals and α_2 is a propositional letter, such that $\mathcal{D}, \gamma_1 : \alpha_1 \vdash \gamma_2 : \alpha_2$,[28] *if there exists a declarative unit $\gamma_3 : \alpha_1 \rightarrow \alpha_2$, where $\gamma_3 : \alpha_1 \wedge \alpha_2$, where $\gamma_3 = \gamma_1 \copyright_{\rightarrow I} \gamma_2$ and $\varphi_{\rightarrow I}(\gamma_1, \gamma_2) \in \mathcal{A}$, then the inference rule $\rightarrow$ I is the tuple $\langle A_{\rightarrow I}, \varphi_{\rightarrow I}, C_{\rightarrow I} \rangle$, where $A_{\rightarrow I} = \{ \gamma_1 : \alpha_1, \gamma_2 : \alpha_2 \}$; and $C_{\rightarrow I} = \gamma_3 : \alpha_1 \rightarrow \alpha_2$. Hence, provided that: $\varphi_{\rightarrow I}(\gamma_1, \gamma_2)$ holds:*

$$\frac{\mathcal{D}, \gamma_1 : \alpha_1 \vdash \gamma_2 : \alpha_2}{\gamma_3 : \alpha_1 \rightarrow \alpha_2}$$

□

Definition 2.34 ($\rightarrow$ Elimination ($\rightarrow$ E))

Given a database $\mathcal{D}$, for any pair of declarative units $\gamma_1 : \alpha_1$ and $\gamma_2 : \alpha_1 \rightarrow \alpha_2$, where α_1 is either a literal or a conjunction of literals, if there exists a declarative unit $\gamma_3 : \alpha_2$, where $\gamma_3 = \gamma_1 \copyright_{\rightarrow E} \gamma_2$ and $\varphi_{\rightarrow E}(\gamma_1, \gamma_2) \in \mathcal{A}$, then the inference rule $\rightarrow$ E is the tuple $\langle A_{\rightarrow E}, \varphi_{\rightarrow E}, C_{\rightarrow E} \rangle$, where $A_{\rightarrow E} = \{ \gamma_1 : \alpha_1, \gamma_2 : \alpha_1 \rightarrow \alpha_2 \}$; and $C_{\rightarrow E} = \gamma_3 : \alpha_2$. Hence, given that $\varphi_{\rightarrow E}(\gamma_1, \gamma_2)$ holds:

$$\frac{\gamma_1 : \alpha_1 \qquad \gamma_2 : \alpha_1 \rightarrow \alpha_2}{\gamma_3 : \alpha_2}$$

□

Definition 2.35 ($\neg$ Elimination ($\neg$E))

Given a database $\mathcal{D}$, for any declarative unit $\gamma_1 : \alpha_1$, where $\alpha_1 = \neg\neg\alpha$ and α is a propositional letter, if there exists a declarative unit $\gamma_2 : \alpha$, where $\varphi_{\neg E}(\gamma_1, \gamma_2) \in \mathcal{A}$, then the inference rule $\neg$E is the tuple $\langle A_{\neg E}, \varphi_{\neg E}, C_{\neg E} \rangle$, where $A_{\neg E} = \{ \gamma_1 : \alpha_1 \}$; and $C_{\neg E} = \gamma_2 : \alpha$. Hence, provided that $\varphi_{\neg E}(\gamma_1, \gamma_2)$ holds:

$$\frac{\gamma_1 : \neg\neg\alpha}{\gamma_2 : \alpha}$$

□

Definition 2.36 ($\perp$ Introduction ($\perp$I))

[27] By Definition 2.23, γ_2 is of the same label type as γ_1, such that $\gamma_1 \in$ EUIUN.

[28] By $\mathcal{D}, \gamma_1 : \alpha_1$ we mean that the declarative unit $\gamma_1 : \alpha_1$ is temporarily added to $\Delta_{\mathcal{D}}$ as an assumption.

Given a database $\mathcal{D}$, for any pair of declarative units $\gamma_1 : \alpha_1$ and $\gamma_2 : \alpha_2$, where $\alpha_1 = \neg\alpha_2$ and α_2 is a propositional letter, if there exists a declarative unit $\gamma_3 : \bot$, where $\gamma_3 = \gamma_1 \copyright_{\bot I} \gamma_2$, and $\varphi_{\bot I}(\gamma_1, \gamma_2) \in \mathcal{A}$, then the inference rule $\bot$I is the tuple $\langle A_{\bot I}, \varphi_{\bot I}, C_{\bot I} \rangle$, where $A_{\bot I} = \{ \gamma_1 : \alpha_1, \gamma_2 : \alpha_2 \}$; and $C_{\bot I} = \gamma_3 : \bot$. Hence, given that $\varphi_{\bot I}(\gamma_1, \gamma_2)$ holds

$$\frac{\gamma_1 : \neg\alpha_2 \qquad \gamma_2 : \alpha_2}{\gamma_3 : \bot}$$

□

Provided that the respective labelling conditions hold, the meanings of the inference rules above are the following:

The rule ($\wedge$I) says that given two provable declarative units $\gamma_1 : \alpha_1$ and $\gamma_2 : \alpha_2$, we can also prove a declarative unit $\gamma_3 : \alpha_1 \wedge \alpha_2$.

Conversely, the rule ($\wedge$E) says that given $\gamma_1 : \alpha_1 \wedge \alpha_2$ provable, we can prove $\gamma_2 : \alpha_1$ and $\gamma_2 : \alpha_2$.

The rule ($\rightarrow$ I) says that if we can prove $\gamma_2 : \alpha_2$ from a database $\mathcal{D}$, by assuming $\gamma_1 : \alpha_1$ as hypotheses, then we can prove $\gamma_3 : \alpha_1 \rightarrow \alpha_2$ from $\mathcal{D}$, and $\gamma_3 : \alpha_1 \rightarrow \alpha_2$ does not depend on the hypotheses $\gamma_1 : \alpha_1$. In a more simple description, we say that $\gamma_3 : \alpha_1 \rightarrow \alpha_2$ holds if by assuming $\gamma_1 : \alpha_1$, we can prove $\gamma_2 : \alpha_2$. This rule admits a sort of sequent presentation in its antecedent part.

The rule ($\rightarrow$ E) says that if we have $\gamma_1 : \alpha_1$ and $\gamma_2 : \alpha_1 \rightarrow \alpha_2$ provable, then we can prove $\gamma_3 : \alpha_2$. As shown in [Dum-77], this rule is essentially the *Modus Ponens* rule.

The rule ($\neg$E), also referred to as the double negation elimination rule, allows us to deduce $\gamma_2 : \alpha$ given that $\gamma_1 : \neg\neg\alpha$ is provable.[29]

The rule ($\bot$I) says that if we can prove a contradiction, say $\gamma_1 : \alpha$ and $\gamma_1 : \neg\alpha$, then can prove inconsistency from our database. The introduction rule for $\bot$ is defined in the literature differently. In [Pra-65] and in [RySa-92], they refer to it as the $\neg$ elimination rule.

Remark 2.10

We say that $\mathcal{D} \vdash \gamma : \bot$, if one of the following cases holds:

- *There exists a declarative unit $\gamma_1 : \alpha$, such that $\mathcal{D} \vdash \gamma_1 : \alpha$, and there also exists a declarative unit $\gamma_2 : \alpha \rightarrow \bot$, such that $\mathcal{D} \vdash \gamma_2 : \alpha \rightarrow \bot$. In this case, the $\rightarrow$ E inference rule is applied and γ is given by $\gamma_1 \copyright_{\rightarrow E} \gamma_2$, provided that $\varphi_{\rightarrow E}(\gamma_1, \gamma_2)$ holds.*
- *There exists a declarative unit $\gamma_1 : \alpha \wedge \neg\alpha$, such that $\mathcal{D} \vdash \gamma_1 : \alpha \wedge \neg\alpha$. In this case, the $\bot$I inference rule is applied and γ is given by the labelling condition $\varphi_{\bot I}$.*

•

[29] Since we do not include the $\neg$ Introduction rule in our system, the rule ($\neg$E) is not very likely to be applied in a proof.

CIU_{LDS} Proof System Definitions:

Proof: $\rho[\gamma : \alpha] = \langle P_\rho, k \rangle$,
where $P_\rho = \{A_1/C_1, A_2/C_2, \cdots, A_n/C_n\}$, for $n > 0$, and k is a mapping from $\{1, \cdots, n\}$, to the set IR_{CIU}, such that for each i $k(i) = \text{IR}$, for any $\text{IR} \in IR_{CIU}$, and $A_i/C_i = A_{\text{IR}}/C_{\text{IR}}$.

$\mathcal{D} \vdash \gamma : \alpha$, if there exists a proof $\rho[\gamma : \alpha]$ from $\mathcal{D}$.

CIU_{LDS} Inference Rules:

Conditional Reflexivity: $\langle A_{\text{CR}}, \varphi_{\text{CR}}, C_{\text{CR}} \rangle$,
where $A_{\text{CR}} = \{ \gamma : \alpha \}$, $C_{\text{CR}} = \gamma : \alpha$, and $\varphi_{\text{CR}}(\gamma, \gamma_i) \in \mathcal{A}$, for any γ_i, such that $\mathcal{D}' \vdash \gamma_i : \alpha$, where $\mathcal{D}' = \mathcal{D} - \{\gamma : \alpha\}$.

$\wedge$ Introduction: $\langle A_{\wedge\text{I}}, \varphi_{\wedge\text{I}}, C_{\wedge\text{I}} \rangle$, where $\varphi_{\wedge\text{I}}(\gamma_1, \gamma_2) \in \mathcal{A}$,

$$\frac{\gamma_1 : \alpha_1 \qquad \gamma_2 : \alpha_2}{\gamma_3 : \alpha_1 \wedge \alpha_2}$$

$\wedge$ Elimination: $\langle A_{\wedge\text{E}}, \varphi_{\wedge\text{E}}, C_{\wedge\text{E}} \rangle$, where $\varphi_{\wedge\text{E}}(\gamma_1, \gamma_2) \in \mathcal{A}$,

$$\frac{\gamma_1 : \alpha_1 \wedge \alpha_2}{\gamma_2 : \alpha_1} \qquad \frac{\gamma_1 : \alpha_1 \wedge \alpha_2}{\gamma_2 : \alpha_2}$$

$\rightarrow$ Introduction: $\langle A_{\rightarrow\text{I}}, \varphi_{\rightarrow\text{I}}, C_{\rightarrow\text{I}} \rangle$, where $\varphi_{\rightarrow\text{I}}(\gamma_1, \gamma_2) \in \mathcal{A}$,

$$\frac{\mathcal{D}, \gamma_1 : \alpha_1 \vdash \gamma_2 : \alpha_2}{\gamma_3 : \alpha_1 \rightarrow \alpha_2}$$

$\rightarrow$ Elimination: $\langle A_{\rightarrow\text{E}}, \varphi_{\rightarrow\text{E}}, C_{\rightarrow\text{E}} \rangle$, where $\varphi_{\rightarrow\text{E}}(\gamma_1, \gamma_2) \in \mathcal{A}$,

$$\frac{\gamma_1 : \alpha_1 \qquad \gamma_2 : \alpha_1 \rightarrow \alpha_2}{\gamma_3 : \alpha_2}$$

$\neg$ Elimination: $\langle A_{\neg\text{E}}, \varphi_{\neg\text{E}}, C_{\neg\text{E}} \rangle$, where $\varphi_{\neg\text{E}}(\gamma_1, \gamma_2) \in \mathcal{A}$,

$$\frac{\gamma_1 : \neg\neg\alpha}{\gamma_2 : \alpha}$$

$\perp$ Introduction: $\langle A_{\perp\text{I}}, \varphi_{\perp\text{I}}, C_{\perp\text{I}} \rangle$, where $\varphi_{\perp\text{I}}(\gamma_1, \gamma_2) \in \mathcal{A}$,

$$\frac{\gamma_1 : \neg\alpha_1 \qquad \gamma_2 : \alpha_1}{\gamma_3 : \perp}$$

Figure 3: Summary of the CIU_{LDS} Proof System Definitions.

2.5.1 Discussions

In this section, we have introduced the proof system's basic definitions. The inference rules for the logical connectives were based on the natural deduction presentation style for propositional logic, following [Pra-65]. Some differences, however, apply in our presentation. We use labelled formulae and we define some conditions for applying the rules. The introduction of labelling conditions controls the application of the inference rules, and the labelling propagation on derived and updated declarative units. Hence, it guides the derivation mechanism specifically to the requirements of our compromised approach. We have introduced a reflexivity inference rule in the proof system. Due to this rule, and to the way that a proof is defined in our system, we always have that if a declarative unit $\gamma : \alpha$ can be proved from $\mathcal{D}$, the proof $\rho[\gamma : \alpha]$ applies at least one inference rule. And if it is the case that $\rho[\gamma : \alpha]$ applies only one inference rule, this rule is the conditional reflexivity one. Unlike the typical natural deduction presentation, which supplies two types of rules - introduction and elimination - for each operation, we have omitted on purpose two usual inference rules from our proof system. The $\neg$ *Introduction* rule was not included, due to the fact that we do not want to generate negated wff of $\mathcal{L}$, when integrity constraints were involved in the derivation. Moreover, the derivation of a formula $\gamma_1 : \neg\alpha$ everytime $\gamma_2 : \alpha$ together with $\mathcal{D}$ generate inconsistency, would interfere with the revision process of CIU_{LDS}. The *Ex Contradictione Quodlibet* (ECQ) inference rule[30], which says that everything can be derived from inconsistency, was also omitted. Mainly because we do not want our system to collapse in the presence of inconsistency. It would be unnatural to abandon a database once we discover that it is inconsistent. Instead, we propose to take inconsistency as a signal for revision on the database. In the next section, we define how the updates are performed in CIU_{LDS}, and how they invoke the compromised revision mechanism.

2.6 Updates and Revision in CIU_{LDS}

Update requests are the inputs of our system. Basically, the updates in CIU_{LDS} involve addition or retraction of data to or from a labelled database $\mathcal{D}$. The retraction is performed in the compromised way, by allowing the consequences of the retracted declarative units to be kept in $\Delta_{\mathcal{D}}$ as non-supported data. And the addition of data is conditioned to the way it conflicts with $\mathcal{D}$, if it is the case. This operation may invoke revision, when a compromised solution applies.

Updates are defined as single declarative units operations. A *single update* deals with addition or retraction of declarative units to or from $\mathcal{D}$. In this section, we define first the operations for a *single update*, and then we present the notion of *transactions*, which can involve a sequence of different updates. Revision is viewed as a function invoked by the update function, whenever a compromised solution applies. This is further detailed in the subsection 2.6.4.

2.6.1 Single Updates

Single updates in a labelled database $\mathcal{D}$, involve addition or retraction of declarative units which are either extensional data or intensional data. The intensional data involved in the updates are supposed to be clausal formulae of $\mathcal{L}$ in their non-labelled part. That is, non-clausal formulae can only be involved in updates as extensional data. These restrictions avoid the manipulation of non-supported data, and of protected data in the updates, as expected. Non-supported data can only be derived by CIU_{LDS}, whereas protected formulae cannot be modified by means of updates to the database. [31]

[30] Sometimes also referred to as $\perp$ *Elimination*. See [RySa-92].

[31] We are not concerned here in dealing with the modification of the protected data of a database in CIU_{LDS}. For our current interests, we consider that the initial database has got its protected data well defined, and that they remain the same after all update operations.

A single update in CIU_{LDS}, is formally defined as a function, denoted as $Up(\mathcal{D},\ \sigma,\ \delta) \Rightarrow \mathcal{D}'$, where $\mathcal{D}$ and $\mathcal{D}'$ are the labelled databases, before and after the update, the argument σ is the type of update to be performed, and δ is the declarative unit $\gamma : \alpha$ involved in the update. The σ argument can take one of the constant values of the set $\{\mathrm{U}_+,\ \mathrm{U}_-\}$. "$\sigma = \mathrm{U}_+$" implies that the update operation is addition, and "$\sigma = \mathrm{U}_-$" implies that the update operation is retraction. In both cases, the δ argument of the update function can be either extensional or intensional data, where α is a clausal formula, in this last data type.

Definition 2.37 (Single Update Function)

Assume that $\mathcal{D}_{CIU}$ and $\mathcal{D}_U$ are the set of databases and the set of declarative units of CIU_{LDS} system, respectively. Let σ be a type of updates given by the set $\{\mathrm{U}_+,\ \mathrm{U}_-\}$. Given that $\gamma : \alpha$ is a declarative unit and $\mathcal{D}$ a CIU_{LDS} database, let the update function $Up : \mathcal{D}_{CIU} \times \{\mathrm{U}_+,\ \mathrm{U}_-\} \times \mathcal{D}_U \Rightarrow \mathcal{D}_{CIU}$ be defined as follows:

- $Up(\mathcal{D},\ \mathrm{U}_+,\ \gamma : \alpha) = \mathcal{D} \uplus \gamma : \alpha.$
- $Up(\mathcal{D},\ \mathrm{U}_-,\ \gamma : \alpha) = \mathcal{D} \,\Xi\, \gamma : \alpha.$

Where $\mathcal{D} \uplus \gamma : \alpha$ and $\mathcal{D} \,\Xi\, \gamma : \alpha$ are the basic update operations of conditional inclusion and compromised retraction.[32]

□

2.6.2 Basic Update Operations

We define as CIU_{LDS} *basic database update operations*, the notion of conditional inclusion of a declarative unit in a labelled database, [33] denoted by $\uplus$, and the notion of compromised retraction of a declarative unit from a CIU_{LDS} database, denoted by Ξ.

Next, we introduce some assumptions for the definition of $\mathcal{D} \uplus \gamma : \alpha$. For both operations, $\uplus$ and Ξ, we assume that the given database $\mathcal{D}$ is initially consistent. Also in both operations, $\gamma : \alpha$ can only be of types $\mathrm{E} : \alpha$ or $\mathrm{I} : \alpha$, such that for $\mathrm{I} : \alpha$, α is a clausal formula of $\mathcal{L}$.

2.6.3 Basic assumptions for $\mathcal{D} \uplus \gamma : \alpha$

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, if the declarative unit $\gamma : \alpha$ is consistent with $\Delta_{\mathcal{D}}$, then $\gamma : \alpha$ is simply added to $\Delta_{\mathcal{D}}$ in the set-theoretical sense. The ordering on $\Delta_{\mathcal{D}} \bigcup \{\gamma : \alpha\}$ is given by the propagation of $\preccurlyeq$, as in Definition 2.16.

When $\gamma : \alpha$ is inconsistent with $\Delta_{\mathcal{D}}$, then the conditional inclusion of $\gamma : \alpha$ to $\mathcal{D}$ is responsibility of the revision function $\odot$.[34] In this case, two possibilities may apply: (We denote as I the set of integrity constraints in $\Delta_{\mathcal{D}}$.)

- The declarative unit violates directly some integrity constraints in $\Delta_{\mathcal{D}}$, i.e. $I,\ \gamma : \alpha \vdash \gamma' : \alpha$ for some $\gamma' \in \mathcal{L}_\gamma$. In this case, the inclusion of this declarative unit is not allowed. However, its consistent consequences w.r.t. $\mathcal{D}$ are added to $\Delta_{\mathcal{D}}$, as non-supported consequences.
- The declarative unit does not violate directly any integrity constraint of the database. In this case, $\gamma : \alpha$ is added to $\Delta_{\mathcal{D}}$, and some old declarative units are retracted from $\Delta_{\mathcal{D}}$, via compromised retraction, in order to maintain consistency in the resulting database.

The resulting database from $\mathcal{D} \uplus \gamma : \alpha$ is always consistent.

---

[32] See Definitions 2.38 and 2.46.

[33] This notion of conditional inclusion is motivated by the notion of structured addition of data in structured databases, in [Gab-94].

[34] See Definition 2.50.

Definition 2.38 ($\mathcal{D} \uplus \gamma : \alpha$)

Given a consistent database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and a declarative unit* $\gamma : \alpha$, *let the operation* $\uplus$ *denote the conditional addition of a declarative unit in* $\mathcal{D}$. *Such that* $\mathcal{D} \uplus \gamma : \alpha = \mathcal{D}'$. *Assume that* c_1 *is the following condition* $c_1 = \Delta_{\mathcal{D}}, \gamma : \alpha \nvdash \gamma' : \bot$, *for any label* $\gamma' \in \mathcal{L}_{\gamma}$. *Then, the new database* $\mathcal{D}'$ *is obtained as follows:*

$$\mathcal{D}' = \begin{cases} \langle (\Delta_{\mathcal{D}} - X_{\mathcal{D}}) \cup \{\gamma : \alpha\}, \preccurlyeq \rangle & \text{if } c_1 \text{ is satisfied ;} \\ \mathcal{D} \odot \gamma : \alpha & \text{otherwise.} \end{cases}$$

Where $\mathcal{D} \odot \gamma : \alpha$ *denotes the revision of* $\mathcal{D}$ *by* $\gamma : \alpha$. $(\Delta_{\mathcal{D}} - X_{\mathcal{D}}) \cup \{\gamma : \alpha\}$ *is ordered by* $\preccurlyeq$, *according to Definition 2.16. And* $X_{\mathcal{D}} \subseteq \Delta_{\mathcal{D}}$, *which is given as follows:*

$$X_{\mathcal{D}} = \{\gamma_l : \alpha \mid \gamma_l : \alpha \in \Delta_{\mathcal{D}}, \text{ and } \varphi_{\Theta}(\gamma, \gamma_l) \in \mathcal{A}\}.$$

□

Remark 2.11

In Definition 2.38, the retraction of the set $X_{\mathcal{D}}$ *from* $\Delta_{\mathcal{D}}$ *caters for the case that there exists a declarative unit* $\gamma_l : \alpha$ *in* $\Delta_{\mathcal{D}}$, *prior to the inclusion of* $\gamma : \alpha$. *Hence, the declarative unit* $\gamma_l : \alpha$ *is removed from* $\Delta_{\mathcal{D}}$, *provided also that the labelling condition* $\varphi_{\Theta}(\gamma, \gamma_l)$ *holds. This prevents the resulting database from having different versions of declarative units. That is, the same wff of* $\mathcal{L}$ *under different labels. Example 2.5 in section 2.1, ilustrates such situation.*

•

Next, we define the notion of compromised retraction of a declarative unit from a CIU_{LDS} database.

Basic assumptions for $\mathcal{D} \Xi \gamma : \alpha$

The notion of conditional retraction of a declarative unit from a database, allows for compromised consequences of retracted declarative units to be added to the database, carrying a non-supported label type.

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, when a declarative unit $\gamma : \alpha \in \Delta_{\mathcal{D}}$ is requested to be retracted from $\Delta_{\mathcal{D}}$ in CIU_{LDS}, we allow the set of consequences of $\gamma : \alpha$ w.r.t. $\mathcal{D}$, denoted as $Con(\gamma : \alpha)$, to be added to the resulting database, under the condition that the consequences in $Con(\gamma : \alpha)$ together with the set $(\Delta_{\mathcal{D}} - \{\gamma : \alpha\})$ do not derive $\gamma_l : \alpha$, for any label γ_l.

In order to guarantee the condition above, we have to restrict the addition of some of the elements of the set of all the consequences of $\gamma : \alpha$ w.r.t. $\mathcal{D}$. This implies a choice problem among those consequences. To solve this problem, we adopt the following strategy:

1. We generate the set $Con(\gamma : \alpha)$ as the set of all consequences of $\gamma : \alpha$ w.r.t. $\mathcal{D}$.
2. If $Con(\gamma : \alpha) \bigcup (\Delta_{\mathcal{D}} - \{\gamma : \alpha\}) \nvdash \gamma_l : \alpha$, for any for any label γ_l, then all the elements of $Con(\gamma : \alpha)$ are added to $\Delta_{\mathcal{D}} - \{\gamma : \alpha\}$, to compose the resulting database.
3. If $Con(\gamma : \alpha) \bigcup (\Delta_{\mathcal{D}} - \{\gamma : \alpha\}) \vdash \gamma_l : \alpha$, for some $\gamma_l \in \mathcal{L}_{\gamma}$, then we get the *safe-maximal subset* of $Con(\gamma : \alpha)$ that fails to derive $\gamma_l : \alpha$. So, all the elements of the *safe-maximal subset* of $Con(\gamma : \alpha)$ are added to $\Delta_{\mathcal{D}} - \{\gamma : \alpha\}$, to compose the resulting database.

As introduced in [Dar-96c], the notion of a *safe-maximal subset*[35] of an ordered set X, w.r.t. a condition c which involves X in its premise, is denoted as $Smax(X)_c$. $Smax(X)_c$ is defined with the aim that when it substitutes X in the condition c, c succeeds. The set $Smax(X)_c$ is obtained considering the ordering $\preccurlyeq$, the set-inclusion property of minimality, and some auxiliary sets denoted as $Fail(X)_c$,

[35] More details about the safe-maximality notion for this formalization is given in section 2.6.4.

$min(Fail(X)_c)$, $Min(Fail(X)_c)$, and $RMin(Fail(X)_c)$. The set $Fail(X)_c$ contains all the minimal subsets of X w.r.t. $\subseteq$, such that when they substitute X in the condition c, c fails. The set $min(Fail(X)_c)$ contains the subsets of minimal elements w.r.t. $\preccurlyeq$, of each set belonging to $Fail(X)_c$. The set $Min(Fail(X)_c)$ contains all the elements of each set belonging to $min(Fail(X)_c)$. And the set $RMin(Fail(X)_c)$ is a refined construction of the set $Min(\mathcal{Y})$, so that fewer elements of the original ordered set are removed from it.

Below, we give the general definitions for the auxiliary sets $Fail(X)_c$, $min(Fail(X)_c)$, $Min(Fail(X)_c)$, $RMin(Fail(X)_c)$ and $Smax(X)_c$.

Definition 2.39 ($Fail(X)_c$)

Given an ordered set X, w.r.t. $\preccurlyeq$, and a condition c which involves X in its premise, let $Fail(X)_c$ be the following set:

$$Fail(X)_c = \begin{cases} \emptyset & \text{if } c \text{ is satisfied;} \\ \{S \mid S \subseteq X; & \\ \text{such that if } c(X/S),\ c \text{ fails;} & \\ \text{and } S \text{ is minimal w.r.t. } \subseteq\} & \text{otherwise.} \end{cases}$$

Where $c(X/S)$ denotes the fact that the set S substitutes the set X in condition c. And each set S_i of $Fail(X)_c$, for $i = 1, \cdots, n$, is ordered by $\preccurlyeq$ as a subset of X.

□

Definition 2.40 ($min(Y)$)

Given a set Y such that $Y = \{S_1, S_2, \cdots, S_n\}$, where each S_i, for $i = 1, \cdots, n$, is ordered w.r.t. $\preccurlyeq$. Let $min(Y)$ be the following set:

$$min(Y) = \begin{cases} \emptyset & \text{if } Y = \emptyset; \\ \{min(S_i) \mid min(S_i) \subseteq S_i; & \\ \text{and } \forall x \in min(S_i),\ x \text{ is minimal w.r.t} \preccurlyeq \text{ in } S_i\} & \text{otherwise.} \end{cases}$$

□

Definition 2.41 ($Min(Y)$)

Given a set Y such that $Y = \{S_1, S_2, \cdots, S_n\}$, where each S_i, for $i = 1, \cdots, n$, is ordered w.r.t. $\preccurlyeq$, assume that $min(Y)$, is such that $min(Y) = \{min(S_1), min(S_2), \cdots, min(S_n)\}$. Let $Min(Y)$ be the following set:

$$Min(Y) = \{x \mid x \in min(S_i),\ \forall min(S_i) \in min(Y)\}.$$

□

Definition 2.42 ($RMin(Y)$)

Given a set Y such that $Y = \{S_1, S_2, \cdots, S_n\}$, where each S_i, for $i = 1, \cdots, n$, is ordered w.r.t. the partial order $\preccurlyeq$. Assume that $min(Y)$, is such that $min(Y) = \{min(S_1), min(S_2), \cdots, min(S_n)\}$. Let $RMin(Y)$ be the following set:

$$RMin(Y) = min(Y)^1 \cup (Min(Y) - M^*) \cup CM.$$

Where: $M^* = \{y \mid y \in min^x,\ \forall min^x \in M\}$; $M = \{min^x \mid \forall min^x \in Min(Y)\}$; and $min^x = \{S_x \mid S_x = min(S_i),\ \forall min(S_i) \in min(Y),\ s.t.\ x \in min(S_i),\ and\ x \in Min(Y)\}$.

And $CM = \{x \mid x \in Min(Y),\ such\ that\ min^x \in M\}$.

□

Remark 2.12

In Definition 2.42, the set min^x contains the set-elements of $min(Y)$, which have the element x in common, for an element x in $Min(Y)$. The set M contains all the sets min^x, for all x in $Min(Y)$. The set M^* is the union of all the sets min^x in M. the set CM contains the common elements x of the sets min^x in M.

•

Definition 2.43 ($Smax(X)_c$)

Given an ordered set X, w.r.t. $\preccurlyeq$, a condition c which involves X in its premise, and the set $Min(Fail(X)_c)$, let $Smax(X)_c$ be a subset of X, obtained as follows:

$$Smax(X)_c = \begin{cases} X & \text{if } c \text{ is satisfied;} \\ X - RMin(Fail(X)_c) & \text{otherwise.} \end{cases}$$

The set $Smax(X)_c$ is ordered by $\preccurlyeq$ as a subset of X.

□

We define below the set of consequences of a declarative unit $\gamma : \alpha$ w.r.t. a database $\mathcal{D}$, where $\gamma : \alpha \in \mathcal{D}$.

Definition 2.44 ($Con(\gamma : \alpha)$)

Given a consistent database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, and a declarative unit $\gamma : \alpha$, such that $\gamma : \alpha \in \mathcal{D}$, let the set $Con(\gamma : \alpha)$ be defined as follows:

$$Con(\gamma : \alpha) = \{\gamma_i : \alpha_i \mid \Delta_{\mathcal{D}} \vdash \gamma_i : \alpha_i, \text{ and } (\Delta_{\mathcal{D}} - \{\gamma : \alpha\}) \nvdash \gamma_i : \alpha_i\}.$$

□

Remark 2.13

Notice that since the idea is to add the declarative units in $Con(\gamma : \alpha)$ to the resulting database contracted by $\gamma : \alpha$, these declarative units have to carry the label type N, *of non-supported consequences. So, we have to re-label the declarative units in $Con(\gamma : \alpha)$, before we get the safe-maximal subset out of it. The definition below caters for this re-labelling on $Con(\gamma : \alpha)$.*

•

Definition 2.45 ($CCon(\gamma : \alpha)$)

Given a set $Con(\gamma : \alpha)$, let the set $CCon(\gamma : \alpha)$, denoting the compromised version of $Con(\gamma : \alpha)$, be defined as follows:

$$CCon(\gamma : \alpha) = \{\gamma_i' : \alpha_i' \mid \forall\, \gamma_i : \alpha_i \in Con(\gamma : \alpha), \\ \gamma_i' = \gamma \copyright_{\Xi} \gamma_i, \text{ such that } \varphi_{\Xi}(\gamma, \gamma_i) \in \mathcal{A}, \text{ and } \alpha_i' = \alpha_i\}.$$

□

This way, we guarantee that all the declarative units $\gamma_i' : \alpha_i'$ in $CCon(\gamma : \alpha)$, carry the label type N, of non-supported consequences.

Remark 2.14 ($Smax(CCon(\gamma : \alpha))_{c_2}$)

From Definitions 2.43 and 2.45, we have that the safe maximal subset of $CCon(\gamma : \alpha)$, *relative to the condition* $c_2 = CCon(\gamma : \alpha) \bigcup (\Delta_{\mathcal{D}} - \{\gamma : \alpha\}) \nvdash \gamma' : \alpha$, *for any* $\gamma' \in \mathcal{L}_{\gamma}$, *is given as follows:*

$$Smax(CCon(\gamma : \alpha))_{c_2} = \begin{cases} CCon(\gamma : \alpha) \\ \quad \textit{if } c_2 \textit{ is satisfied;} \\ \\ CCon(\gamma : \alpha) - RMin(Fail(CCon(\gamma : \alpha))_{c_2}) \\ \quad \textit{otherwise.} \end{cases}$$

•

Now, we can define the compromised retraction operation.

Definition 2.46 ($\mathcal{D} \; \Xi \; \gamma : \alpha$)

Given a consistent database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and a declarative unit* $\gamma : \alpha$, *let the operation "* Ξ *" denote the compromised retraction of a declarative unit from* $\mathcal{D}$. *Such that* $\mathcal{D} \; \Xi \; \gamma : \alpha = \mathcal{D}'$, *where the new database* $\mathcal{D}'$ *is obtained as follows:*

$$\mathcal{D}' = \begin{cases} \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle & \textit{if } \gamma : \alpha \notin \Delta_{\mathcal{D}}; \\ \\ \langle (\Delta_{\mathcal{D}} \bigcup Smax(CCon(\gamma : \alpha))_{c_2}) - \{\gamma : \alpha\}, \preccurlyeq \rangle & \textit{otherwise.} \end{cases}$$

Where $(\Delta_{\mathcal{D}} \bigcup Smax(CCon(\gamma : \alpha))_{c_2})$ *is ordered by* $\preccurlyeq$, *according to Definition 2.16.*

□

Summary - Single Updates

We have defined as single updates for CIU_{LDS}, the update requests which involve addition and retraction of declarative units to and from a labelled database $\mathcal{D}$. We have restricted the declarative units in the single updates, to be either extensional data or intensional data which are present in $\mathcal{D}$. These restrictions avoid that the updates handle protected data, as usually expected, and also non-supported data. We have assumed that non-supported data can only be derived by the system, and can not be involved in update operations. The basic update operations which are invoked by the single updates, carry the reconciling flavour of our compromised approach to conflicting inputs. The operation of conditional inclusion $\biguplus$, invokes the compromised revision function $\odot$, when a compromised solution for the update applies. The compromised retraction operation Ξ, already embeds in its definition the mechanism for allowing consequences of retracted declarative units to be added to the database as non-supported data. This operation uses the notion of safe-maximality, when a choice is needed among the compromised consequences. In the next section, we define the compromised revision mechanism, which is invoked by the update function.

2.6.4 Revision in CIU_{LDS}

Revision plays a central role in our system. It is basically via the revision mechanism, that the compromised philosophy of our approach is introduced in CIU_{LDS}. The compromised revision adopted here, frees the database from inconsistency and allows some consequences from conflicting updates, or from retracted declarative units, to be kept in the resulting database.

The compromised revision mechanism distinguishes between the protected formulae and the other formulae in a CIU_{LDS} database.

Notation 2.3

CIU_{LDS} Single Updates Definitions:

Single Update Function: $Up : \mathcal{D}_{CIU} \times \{\mathtt{U}_+, \mathtt{U}_-\} \times \mathcal{D}_U \Rightarrow \mathcal{D}_{CIU}$:

- $Up(\mathcal{D}, \mathtt{U}_+, \gamma : \alpha) = \mathcal{D} \uplus \gamma : \alpha.$
- $Up(\mathcal{D}, \mathtt{U}_-, \gamma : \alpha) = \mathcal{D} \,\Xi\, \gamma : \alpha.$

Given $\mathcal{D} = \langle \Delta_\mathcal{D}, \preccurlyeq \rangle$, $\gamma : \alpha$, and the condition $c_1 = \Delta_\mathcal{D}, \gamma : \alpha \nvdash \gamma' : \bot$, for any label $\gamma' \in \mathcal{L}_\gamma$, $\mathcal{D} \uplus \gamma : \alpha = \mathcal{D}'$, where:

$$\mathcal{D}' = \begin{cases} \langle (\Delta_\mathcal{D} - X_\mathcal{D}) \cup \{\gamma : \alpha\}, \preccurlyeq \rangle & \text{if } c_1 \text{ is satisfied ;} \\ \mathcal{D} \odot \gamma : \alpha & \text{otherwise.} \end{cases}$$

$$X_\mathcal{D} = \{\gamma_l : \alpha \mid \gamma_l : \alpha \in \Delta_\mathcal{D}, \text{ and } \varphi_\uplus(\gamma, \gamma_l) \in \mathcal{A}\}.$$

Given that X is ordered by $\preccurlyeq$, and a condition c which involves X,

$$Fail(X)_c = \begin{cases} \emptyset & \text{if } c \text{ is satisfied;} \\ \{S \mid S \subseteq X; \\ \text{such that if } c(X/S), \ c \text{ fails;} \\ \text{and } S \text{ is minimal w.r.t. } \subseteq\} & \text{otherwise.} \end{cases}$$

$$Smax(X)_c = \begin{cases} X & \text{if } c \text{ is satisfied;} \\ X - RMin(Fail(X)_c) & \text{otherwise.} \end{cases}$$

Given that $\gamma : \alpha \in \mathcal{D}$,

$Con(\gamma : \alpha) = \{\gamma_i : \alpha_i \mid \Delta_\mathcal{D} \vdash \gamma_i : \alpha_i, \text{ and } (\Delta_\mathcal{D} - \{\gamma : \alpha\}) \nvdash \gamma_i : \alpha_i\}.$

$CCon(\gamma : \alpha) = \{\gamma'_i : \alpha'_i \mid \forall\, \gamma_i : \alpha_i \in Con(\gamma : \alpha),$
$\gamma'_i = \gamma \copyright_\Xi \gamma_i, \text{ such that } \varphi_\Xi(\gamma, \gamma_i) \in \mathcal{A}, \text{ and } \alpha'_i = \alpha_i\}.$

Given $c_2 = CCon(\gamma : \alpha) \bigcup (\Delta_\mathcal{D} - \{\gamma : \alpha\}) \nvdash \gamma' : \alpha$, for any $\gamma' \in \mathcal{L}_\gamma$,

$$Smax(CCon(\gamma : \alpha))_{c_2} = \begin{cases} CCon(\gamma : \alpha), \\ \text{if } c_2 \text{ is satisfied;} \\ CCon(\gamma : \alpha) - RMin(Fail(CCon(\gamma : \alpha))_{c_2}), \\ \text{otherwise.} \end{cases}$$

$\mathcal{D} \,\Xi\, \gamma : \alpha = \mathcal{D}'$, where:

$$\mathcal{D}' = \begin{cases} \langle \Delta_\mathcal{D}, \preccurlyeq \rangle, \\ \text{if } \gamma : \alpha \notin \Delta_\mathcal{D}; \\ \langle (\Delta_\mathcal{D} \bigcup Smax(CCon(\gamma : \alpha))_{c_2}) - \{\gamma : \alpha\}, \preccurlyeq \rangle, \\ \text{otherwise.} \end{cases}$$

Figure 4: Summary of the CIU_{LDS} Single Updates Definitions.

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *we denote as* I *the finite set of declarative units of the type* $\mathrm{P} : \alpha$, *which represent the integrity constraints ranging over* $\mathcal{D}$, *such that* $I \subseteq \mathcal{D}$.

$$I = \{\gamma_i : \alpha_i \mid \forall \gamma_i : \alpha_i \in \Delta_{\mathcal{D}}, \textit{ such that } \gamma_i \in \mathrm{P}\}.$$

•

The revision in CIU_{LDS} is based on the compromised revision ® for finite bases with integrity constraints, defined in [Dar-96c]. Like in [Dar-96c], here we also adopt the notion of *safe-maximality* as an impartial solution to choose among minimal elements of an ordered set. This notion restricts the notion of maximal subsets relative to certain conditions. One of the advantages in adopting this notion is that we get uniqueness in the result.

Notation 2.4

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and a declarative unit* $\gamma : \alpha$, *we denote as* $\mathcal{D} \odot \gamma : \alpha$ *the result of revision* $\mathcal{D}$ *by* $\gamma : \alpha$, *with the compromised revision* $\odot$.

•

Compromised Revision Steps

In CIU_{LDS}, given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, and a declarative unit $\gamma : \alpha$, such that $\gamma : \alpha \in \Delta_{\mathcal{D}}$, a revision is invoked by the update $Up(\mathcal{D}, \mathrm{U}_+, \gamma : \alpha)$, when one of the conditions below holds:

1. If $I, \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$;
2. If $I, \gamma : \alpha \nvdash \gamma' : \bot$, and $\Delta_{\mathcal{D}}, \gamma : \alpha \vdash \gamma' : \bot$ for $\gamma' \in \mathcal{L}_\gamma$.

When Condition (1) applies, we take the following revision steps on $\mathcal{D}$:

- If α contradicts a tautology of the underlying logical system, then we reject the update request, and the result of the compromised revision is the original database, such that $\mathcal{D} \odot \gamma : \alpha = \mathcal{D}$.
- Otherwise, we do not add $\gamma : \alpha$ to $\Delta_{\mathcal{D}}$, but we generate a set of consequences of $\gamma : \alpha$ with respect to $\mathcal{D}$, and we add to $\Delta_{\mathcal{D}}$ a safe-maximal subset of this set of consequences, relative to the condition that the consequences in it together with $\Delta_{\mathcal{D}}$ are not inconsistent and do not derive $\gamma' : \alpha$, for any label γ' of $\mathcal{L}_\gamma$.

When Condition (2) applies, we add the input $\gamma : \alpha$ to $\Delta_{\mathcal{D}}$, and we reject from $\Delta_{\mathcal{D}}$ some old declarative units, which are not protected, to regain consistency. We adopt the safe-maximality notion whenever we have to choose among a set of declarative units to be retracted from the ordered database, for not satisfying a particular condition imposed to it.[36] We also take into account the inclusion of the consistent consequences of the retracted sentences, as non-supported data. This is done via the following steps:

- First we get the minimal set[37] of minimal elements[38], say $R_{\gamma:\alpha}$, to be retracted from $(\Delta_{\mathcal{D}} - I)$, such that the resulting database is consistent with the inclusion of $\gamma : \alpha$.
- Then, we make the compromised retraction of the set $R_{\gamma:\alpha}$ from $\Delta_{\mathcal{D}}$, with the condition that the consistent consequences of $R_{\gamma:\alpha}$ to be kept in the resulting database, should not contribute to the derivation of any element in $R_{\gamma:\alpha}$, and should not introduce inconsistency either.

[36] See Definition 2.43.
[37] With relation to $\subseteq$.
[38] With relation to $\preccurlyeq$.

- Finally, we get the resulting database, such that it is consistent. It has the same structure as $\mathcal{D}$, and it contains $\gamma : \alpha$, and the selected consequences of $R_{\gamma:\alpha}$.

In the subsections to follow, we introduce the basic assumptions and the auxiliary definitions, for the two cases of revision described above.

Definitions for the case that $I, \gamma : \alpha \vdash \gamma' : \bot$

For this case, we have to define the set of consistent consequences of the declarative unit $\gamma : \alpha$ w.r.t. $\mathcal{D}$, that will be inserted to the resulting database, as a revision compromise.

We will denote as $CI(\gamma : \alpha)$, the set of all consequences of the requested input $\gamma : \alpha$ w.r.t. $\mathcal{D}$. Our goal is to define $CI(\gamma : \alpha)$ such that $\gamma : \alpha \notin CI(\gamma : \alpha)$, and $CI(\gamma : \alpha) \bigcup \Delta_{\mathcal{D}}$ is consistent and do not derive $\gamma : \alpha$. When defining $CI(\gamma : \alpha)$, we have to consider the following:

- $(\Delta_{\mathcal{D}} - I) \bigcup \{\gamma : \alpha\}$ can also be inconsistent when $I \bigcup \{\gamma : \alpha\} \vdash \gamma' : \bot$.
- When selecting a subset of $CI(\gamma : \alpha)$, in the case that $CI(\gamma : \alpha) \bigcup \Delta_{\mathcal{D}} \vdash \gamma' : \bot$, if a maximal subset of $CI(\gamma : \alpha)$ is considered to avoid inconsistency, in order to cater for the minimal change revision notion, it might not be necessarily unique.

Taking these problems into account, we adopt the notion of *safe maximal* subset, as in Definition 2.43, and we proceed in the following way:

- First we get a safe-maximal subset of $(\Delta_{\mathcal{D}} - I)$, w.r.t. condition c_3, where $c_3 = (\Delta_{\mathcal{D}} - I) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, for any label γ'. We use the auxiliary sets $Fail(\Delta_{\mathcal{D}} - I)_{c_3}$ and $RMin(Fail(\Delta_{\mathcal{D}} - I)_{c_3})$ to create $Smax(\Delta_{\mathcal{D}} - I)_{c_3}$, as stated in Definitions 2.39, 2.42, and 2.43. We then prove that $Smax(\Delta_{\mathcal{D}} - I)_{c_3} \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$ for any label γ'. (See Remarks 2.15, 2.16, and 2.17; as well as Propositions 2.1 and 2.2).
- Then we define $CI(\gamma : \alpha)$ in relation to $Smax(\Delta_{\mathcal{D}} - I)_{c_3}$. This step allows us to eliminate the possibility of dealing with an inconsistent set, in the case that condition c_3 is not satisfied. (See Definition 2.47).
- From $CI(\gamma : \alpha)$, we get a safe-maximal subset, w.r.t. the condition $c_4 = CI(\gamma : \alpha) \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \bot$, for any label γ'. The auxiliary sets $Fail(CI(\gamma : \alpha))_{c_4}$ and $RMin(Fail(CI(\gamma : \alpha))_{c_4})$ are used to create $Smax(CI(\gamma : \alpha))_{c_4}$. And we prove that $Smax(CI(\gamma : \alpha))_{c_4} \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \bot$, for any label γ'. (See Remarks 2.18 and 2.19; and Proposition 2.3).
- Finally, we get a safe-maximal subset of $Smax(CI(\gamma : \alpha))_{c_4}$, w.r.t. condition c_5, where $c_5 = Smax(CI(\gamma : \alpha))_{c_4} \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \alpha$, for any label γ'. And it is proved that $Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5} \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \alpha$, for any label γ'. (See Remarks 2.20, and 2.21, as well as Proposition 2.4).

We state below the remarks, definitions and propositions cited above. The proofs of the propositions stated in this section appear in the appendix of this paper.

Remark 2.15 ($Fail(\Delta_{\mathcal{D}} - I)_{c_3}$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *a declarative unit* $\gamma : \alpha$, *and the condition* $c_3 = (\Delta_{\mathcal{D}} - I) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, *for any label* γ', *by Definition 2.39, we have that* $Fail(\Delta_{\mathcal{D}} - I)_{c_3}$ *is the following set:*

$$Fail(\Delta_{\mathcal{D}} - I)_{c_3} = \begin{cases} \emptyset, \\ \textit{if } c_3 \textit{ is satisfied;} \\ \\ \{S \mid S \subseteq (\Delta_{\mathcal{D}} - I); \\ \textit{such that } S \cup \{\gamma : \alpha\} \vdash \gamma' : \bot, \\ \textit{and } S \textit{ is minimal w.r.t. } \subseteq\}, \\ \textit{otherwise.} \end{cases}$$

Each set S_i *of* $Fail(\Delta_{\mathcal{D}} - I)_{c_3}$, *for* $i = 1, \cdots, l$, *is ordered by* $\preccurlyeq$ *as a subset of* $\Delta_{\mathcal{D}}$.

•

Proposition 2.1

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and a declarative unit* $\gamma : \alpha$, *such that* $(\Delta_{\mathcal{D}} - I) \bigcup \{\gamma : \alpha\} \vdash \gamma' : \bot$, *for some label* $\gamma' \in \mathcal{L}_\gamma$, *and it is not the case that* $\neg\alpha$ *is a tautology. Then, given the set* $Fail(\Delta_{\mathcal{D}} - I)_{c_3} = \{S_1, S_2, \cdots, S_j\}$, *it is sufficient to retract one element from each* S_i, *such that* $((\Delta_{\mathcal{D}} - I) - \mathcal{S}(Fail(\Delta_{\mathcal{D}} - I)_{c_3})) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, *for any label* $\gamma' \in \mathcal{L}_\gamma$, *where* $\mathcal{S}(Fail(\Delta_{\mathcal{D}} - I)_{c_3}) = \{\gamma_i : \alpha_i \mid \forall S_i \in Fail(\Delta_{\mathcal{D}} - I)_{c_3}, \exists \gamma^* : \alpha^* \in S_i$, *such that* $\gamma_i, \gamma^* \in \mathtt{z}$, *and* $\alpha_i = \alpha^*\}$, *where* $\mathtt{z}$ *is a label type in* $\mathcal{L}_\gamma$.

Remark 2.16 ($RMin(Fail(\Delta_{\mathcal{D}} - I)_{c_3})$)

Applying Definition 2.40, for $Y = Fail(\Delta_{\mathcal{D}} - I)_{c_3}$, *we obtain the set* $min(Fail(\Delta_{\mathcal{D}} - I)_{c_3})$. *By Definitions 2.41 and 2.42, we get the sets* $Min(Fail(\Delta_{\mathcal{D}} - I)_{c_3})$ *and* $RMin(Fail(\Delta_{\mathcal{D}} - I)_{c_3})$.

•

Remark 2.17 ($Smax(\Delta_{\mathcal{D}} - I)_{c_3}$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *a declarative unit* $\gamma : \alpha$, *and the condition* $c_3 = (\Delta_{\mathcal{D}} - I) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, *for any label* γ', *by Definition 2.43, we have that* $Smax(\Delta_{\mathcal{D}} - I)_{c_3}$ *is the following set:*

$$Smax(\Delta_{\mathcal{D}} - I)_{c_3} = \begin{cases} (\Delta_{\mathcal{D}} - I) & \text{if } c_3 \text{ is satisfied;} \\ (\Delta_{\mathcal{D}} - I) - RMin(Fail(\Delta_{\mathcal{D}} - I)_{c_3}) & \text{otherwise.} \end{cases}$$

$Smax(\Delta_{\mathcal{D}} - I)_{c_3}$ *is ordered by* $\preccurlyeq$ *as a subset of* $\Delta_{\mathcal{D}}$.

•

Proposition 2.2

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *a declarative unit* $\gamma : \alpha$, *such that* $(\Delta_{\mathcal{D}} - I) \bigcup \{\gamma : \alpha\} \vdash \gamma' : \bot$, *for some* $\gamma' \in \mathcal{L}_\gamma$, *and it is not the case that* $\neg\alpha$ *is a tautology,* $Smax(\Delta_{\mathcal{D}} - I)_{c_3}, \gamma : \alpha \nvdash \gamma' : \bot$, *for any* $\gamma' \in \mathcal{L}_\gamma$.

Now we define the set of consequences of the input $\gamma : \alpha$, $CI(\gamma : \alpha)$, w.r.t. $Smax(\Delta_{\mathcal{D}} - I)_{c_3}$.

Definition 2.47 ($CI(\gamma : \alpha)$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and a declarative unit* $\gamma : \alpha$, *such that it is not the case that* $\neg\alpha$ *is a tautology, let* $CI(\gamma : \alpha)$ *be the set of consequences of* $\gamma : \alpha$, *w.r.t.* $\Delta_{\mathcal{D}}$, *such that:*

$$\begin{aligned} CI(\gamma : \alpha) = \; & \{\gamma^* : \alpha^* \mid Smax(\Delta_{\mathcal{D}} - I)_{c_3} \nvdash \gamma^* : \alpha^*, \text{ and} \\ & Smax(\Delta_{\mathcal{D}} - I)_{c_3} \cup \{\gamma : \alpha\} \vdash \gamma^* : \alpha^*; \\ & \alpha^* \neq \alpha, \text{ and } Smax(\Delta_{\mathcal{D}} - I)_{c_3} \cup \{\gamma^* : \alpha^*\} \nvdash \gamma : \alpha\}, \end{aligned}$$

where $CI(\gamma : \alpha)$ *is finite, maximal w.r.t.* $\subseteq$, *and is ordered by* $\preccurlyeq$, *according to Definition 2.16.*

□

We now get the safe-maximal subset $Smax(CI(\gamma : \alpha))_{c_4}$ relative to the condition $c_4 = CI(\gamma : \alpha) \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \bot$.

Remark 2.18 ($Fail(CI(\gamma : \alpha))_{c_4}$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *a declarative unit* $\gamma : \alpha$, *such that* $CI(\gamma : \alpha)$ *is non-empty, and the condition* $c_4 = CI(\gamma : \alpha) \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \bot$, *for any label* $\gamma' \in \mathcal{L}_\gamma$, *by Definition 2.39 we have that* $Fail(CI(\gamma : \alpha))_{c_4}$ *is the following set:*

$$Fail(CI(\gamma:\alpha))_{c_4} = \begin{cases} \emptyset & \text{if } c_4 \text{ is satisfied;} \\ \{S \mid S \subseteq CI(\gamma:\alpha); & \\ \text{such that } S \cup \Delta_{\mathcal{D}} \vdash \gamma' : \bot, \text{and} & \\ S \text{ is minimal w.r.t. } \subseteq\} & \text{otherwise.} \end{cases}$$

Each set S_i *of* $Fail(CI(\gamma:\alpha))_{c_4}$, *for* $i = 1, \cdots, m$, *is ordered by* $\preccurlyeq$ *as a subset of* $CI(\gamma:\alpha)$.

•

Remark 2.19 ($Smax(CI(\gamma:\alpha))_{c_4}$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *a declarative unit* $\gamma:\alpha$, *such that* $CI(\gamma:\alpha)$ *is non-empty, and the condition* $c_4 = CI(\gamma:\alpha) \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \bot$, *for any label* $\gamma' \in \mathcal{L}_\gamma$, *by Definition 2.43 we have that* $Smax(CI(\gamma:\alpha))_{c_4}$ *is the following set:*

$$Smax(CI(\gamma:\alpha))_{c_4} = \begin{cases} CI(\gamma:\alpha) & \text{if } c_4 \text{ is satisfied;} \\ CI(\gamma:\alpha) - RMin(Fail(CI(\gamma:\alpha))_{c_4}) & \text{otherwise.} \end{cases}$$

$Smax(CI(\gamma:\alpha))_{c_4}$ *is ordered by* $\preccurlyeq$ *as a subset of* $CI(\gamma:\alpha)$.

•

Proposition 2.3

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *a declarative unit* $\gamma:\alpha$, *such that* $CI(\gamma:\alpha)$ *is non-empty,* $\Delta_{\mathcal{D}}, \gamma:\alpha \vdash \gamma' : \bot$ *for some label* $\gamma' \in \mathcal{L}_\gamma$, *and it is not the case that* α *is a tautology,* $Smax(CI(\gamma:\alpha))_{c_4} \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \bot$, *for any label* $\gamma' \in \mathcal{L}_\gamma$.

We still have to guarantee the condition $c_5 = Smax(CI(\gamma:\alpha))_{c_4} \cup \Delta_{\mathcal{D}} \nvdash \gamma' : \alpha$. So, we get the safe-maximal subset $Smax(Smax(CI(\gamma:\alpha))_{c_4})_{c_5}$.

Remark 2.20 ($Fail(Smax(CI(\gamma:\alpha))_{c_4})_{c_5}$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *a declarative unit* $\gamma:\alpha$, *such that* $CI(\gamma:\alpha)$ *is non-empty, and the condition* $c_5 = Smax(CI(\gamma:\alpha))_{c_4} \cup \Delta_{\mathcal{D}} \nvdash \gamma' : \alpha$, *for any* $\gamma' \in \mathcal{L}_\gamma$. *By Definition 2.39, we have that* $Fail(Smax(CI(\gamma:\alpha))_{c_4})_{c_5}$ *is the following set:*

$$Fail(Smax(CI(\gamma:\alpha))_{c_4})_{c_5} = \begin{cases} \emptyset & \text{if } c_5 \text{ is satisfied;} \\ \{S \mid S \subseteq Smax(CI(\gamma:\alpha))_{c_4}; & \\ \text{such that } S \cup \Delta_{\mathcal{D}} \vdash \gamma' : \alpha, & \\ \text{and } S \text{ is minimal w.r.t. } \subseteq\} & \text{otherwise.} \end{cases}$$

Each set S_i *of* $Fail(Smax(CI(\gamma:\alpha))_{c_4})_{c_5}$, *for* $i = 1, \cdots, n$, *is ordered by* $\preccurlyeq$ *as a subset of* $CI(\gamma:\alpha)$.

•

Remark 2.21 ($Smax(Smax(CI(\gamma:\alpha))_{c_4})_{c_5}$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *a declarative unit* $\gamma:\alpha$, *such that* $CI(\gamma:\alpha)$ *is non-empty, and the condition* $c_5 = Smax(CI(\gamma:\alpha))_{c_4} \cup \Delta_{\mathcal{D}} \nvdash \gamma' : \alpha$, *for any* $\gamma' \in \mathcal{L}_\gamma$. *By Definition 2.43, we have that* $Smax(Smax(CI(\gamma:\alpha))_{c_4})_{c_5}$ *is the following set:*

$$Smax(Smax(CI(\gamma:\alpha))_{c_4})_{c_5} = \begin{cases} Smax(CI(\gamma:\alpha))_{c_4}, & \\ \text{if } c_5 \text{ is satisfied;} & \\ Smax(CI(\gamma:\alpha))_{c_4} - & \\ RMin(Fail(Smax(CI(\gamma:\alpha))_{c_4})_{c_5}), & \\ \text{otherwise.} & \end{cases}$$

$Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$ is ordered by $\preccurlyeq$, as a subset of $CI(\gamma : \alpha)$.

•

Proposition 2.4

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, a declarative unit $\gamma : \alpha$, such that $CI(\gamma : \alpha)$ is non-empty, and it is not the case that α is a tautology, $Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5} \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \alpha$, for any $\gamma' \in \mathcal{L}_\gamma$.

Definitions for the case that $I, \gamma : \alpha \vdash \gamma' : \bot$:

Given $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, and $c_3 = (\Delta_{\mathcal{D}} - I) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, where $\gamma' \in \mathcal{L}_\gamma$, and $I = \{\gamma_i : \alpha_i \mid \forall \gamma_i : \alpha_i \in Delta_{\mathcal{D}}$, s. t. $\gamma_i \in$ P$\}$.

$$Fail(\Delta_{\mathcal{D}} - I)_{c_3} = \begin{cases} \emptyset \\ \text{if } c_3 \text{ is satisfied;} \\ \{S \mid S \subseteq (\Delta_{\mathcal{D}} - I); \\ \text{s. t. } S \cup \{\gamma : \alpha\} \vdash \gamma' : \bot, \\ \text{and } S \text{ is minimal w.r.t. } \subseteq\} \\ \text{otherwise.} \end{cases}$$

$$Smax(\Delta_{\mathcal{D}} - I)_{c_3} = \begin{cases} (\Delta_{\mathcal{D}} - I), \\ \text{if } c_3 \text{ is satisfied;} \\ (\Delta_{\mathcal{D}} - I) - RMin(Fail(\Delta_{\mathcal{D}} - I)_{c_3}), \quad \text{otherwise.} \end{cases}$$

$$CI(\gamma : \alpha) = \{\gamma^* : \alpha^* \mid Smax(\Delta_{\mathcal{D}} - I)_{c_3} \nvdash \gamma^* : \alpha^* \text{ and } Smax(\Delta_{\mathcal{D}} - I)_{c_3} \cup \{\gamma : \alpha\} \vdash \gamma^* : \alpha^*; \alpha^* \neq \alpha, \text{ and } Smax(\Delta_{\mathcal{D}} - I)_{c_3} \cup \{\gamma^* : \alpha^*\} \nvdash \gamma : \alpha\},$$

Given $c_4 = CI(\gamma : \alpha) \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \bot$, for any label $\gamma' \in \mathcal{L}_\gamma$,

$$Fail(CI(\gamma : \alpha))_{c_4} = \begin{cases} \emptyset & \text{if } c_4 \text{ is satisfied;} \\ \{S \mid S \subseteq CI(\gamma : \alpha); & \\ \text{such that } S \cup \Delta_{\mathcal{D}} \vdash \gamma' : \bot, & \\ \text{and } S \text{ is minimal w.r.t. } \subseteq\} & \text{otherwise.} \end{cases}$$

$$Smax(CI(\gamma : \alpha))_{c_4} = \begin{cases} CI(\gamma : \alpha), \\ \text{if } c_4 \text{ is satisfied;} \\ CI(\gamma : \alpha) - RMin(Fail(CI(\gamma : \alpha))_{c_4}), \\ \text{otherwise.} \end{cases}$$

Given $c_5 = Smax(CI(\gamma : \alpha))_{c_4} \cup \Delta_{\mathcal{D}} \nvdash \gamma' : \alpha$, for any $\gamma' \in \mathcal{L}_\gamma$,

$$Fail(Smax(CI(\gamma : \alpha))_{c_4})_{c_5} = \begin{cases} \emptyset \\ \text{if } c_5 \text{ is satisfied;} \\ \{S \mid S \subseteq Smax(CI(\gamma : \alpha))_{c_4}; \\ \text{such that } S \cup \Delta_{\mathcal{D}} \vdash \gamma' : \alpha, \\ \text{and } S \text{ is minimal w.r.t. } \subseteq\} \\ \text{otherwise.} \end{cases}$$

$$Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5} = \begin{cases} Smax(CI(\gamma : \alpha))_{c_4}, \\ \text{if } c_5 \text{ is satisfied;} \\ Smax(CI(\gamma : \alpha))_{c_4} - \\ RMin(Fail(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}), \\ \text{otherwise.} \end{cases}$$

Figure 5: Summary of Auxiliary Definitions for Revision - Case (2).

2.6.5 Definitions for the case that $I, \gamma : \alpha \nvdash \gamma' : \bot$ and $\Delta_{\mathcal{D}}, \gamma : \alpha \vdash \gamma' : \bot$

In this case, we first have to define the minimal subset, w.r.t. $\subseteq$, of $\Delta_{\mathcal{D}}$ that should be retracted from it, in order to allow the introduction of $\gamma : \alpha$ and keep consistency. We refer to this set as $R_{\gamma:\alpha}$. Then, we have to define the set of consequences of $R_{\gamma:\alpha}$ to be added to the resulting revised database as a compromise.

Notation 2.5

We denote as $\Delta_{\mathcal{D}-I}$ the subset of $\Delta_{\mathcal{D}}$, which excludes the set of integrity constraints I. $\Delta_{\mathcal{D}-I} = \Delta_{\mathcal{D}} - I$.

•

Our aim is to define $R_{\gamma:\alpha}$, such that $R_{\gamma:\alpha} \subseteq \Delta_{\mathcal{D}-I}$, and the following conditions are satisfied: $(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R_{\gamma:\alpha}$; $(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}) \bigcup \{\gamma : \alpha\} \bigcup I \nvdash \gamma' : \bot$, for any $\gamma' \in \mathcal{L}_\gamma$; and $R_{\gamma:\alpha}$ is minimal w.r.t. $\subseteq$. However, we will have to face choice problems, among the declarative units of $\Delta_{\mathcal{D}-I}$, in order to build up the set $R_{\gamma:\alpha}$. In order to get a version of $R_{\gamma:\alpha}$ which accomplishes the conditions above, we proceed in the following way:

- First we get the set $R_{\gamma:\alpha}$, which is a subset of $\Delta_{\mathcal{D}-I}$, relative to the condition $c_6 = \Delta_{\mathcal{D}} \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, for any $\gamma' \in \mathcal{L}_\gamma$. (See Definition 2.48).
- Then we define the safe-maximal subset of $(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})$, denoted as $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$, relative to the condition $c_7 = (\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R_{\gamma:\alpha}$. We then prove that $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$ does not derive any declarative unit of $R_{\gamma:\alpha}$. (See Remarks 2.22 and 2.23, and Proposition 2.5).
- From $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$, we define the safe-maximal subset $Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$, in order to guarantee that $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} \bigcup \{\gamma : \alpha\} \bigcup I$ is consistent. (See Remarks 2.24 and 2.25, and Proposition 2.6).
- Finally, we have that $R^*_{\gamma:\alpha} = \Delta_{\mathcal{D}-I} - Smax(Smax(\Delta_{\mathcal{D}} - R_{\gamma:\alpha})_{c_7})_{c_8}$, is the extended version of $R_{\gamma:\alpha}$, for $R_{\gamma:\alpha} \subseteq R^*_{\gamma:\alpha}$, which accomplishes the conditions cited above. Hence, $R^*_{\gamma:\alpha}$ is effectively the set of declarative units that we need to retract from $\Delta_{\mathcal{D}}$, in order to restore consistency when an input $\gamma : \alpha$ conflicts with $\mathcal{D}$ and $I, \gamma : \alpha \nvdash \gamma' : \bot$.

Concerning the set of consequences of $R^*_{\gamma:\alpha}$, denoted as $CR(R^*_{\gamma:\alpha})$, which should be included in the resulting revised database as a compromise, we define in the following way:

- First we get the maximal set $CR(R^*_{\gamma:\alpha})$, of consequences of $R^*_{\gamma:\alpha}$ w.r.t. $\mathcal{D}$. (See Definition 2.49).
- Then we define the safe-maximal subset of $CR(R^*_{\gamma:\alpha})$, $Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$, which guarantees that $Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \bigcup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$. We also show that $Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \bigcup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, for any label $\gamma' \in \mathcal{L}_\gamma$. (See Remarks 2.26 and 2.27, and Propositions 2.7 and 2.8).

Definition 2.48 ($R_{\gamma:\alpha}$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, an input $\gamma : \alpha$, such that it is not the case that $\neg\alpha$ is a tautology, and the condition $c_6 = \Delta_{\mathcal{D}} \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, for any $\gamma' \in \mathcal{L}_\gamma$. Let $R_{\gamma:\alpha}$ be the following set:

$$R_{\gamma:\alpha} = RMin(F(\Delta_{\mathcal{D}-I})_{c_6}).$$

Where $RMin(F(\Delta_{\mathcal{D}-I})_{c_6})$ is given according to Definition 2.42, and the set $F(\Delta_{\mathcal{D}-I})_{c_6}$ is such that:

$$F(\Delta_{\mathcal{D}-I})_{c_6} = \begin{cases} \emptyset & \text{if } c_6 \text{ is satisfied;} \\ \{S |\ S \subseteq \Delta_{\mathcal{D}-I}\ ;\ S \cup \{\gamma : \alpha\} \vdash \gamma' : \bot; \\ \ \text{and } S \text{ is minimal w.r.t. } \subseteq\} & \text{otherwise.} \end{cases}$$

Where each set S_i *of* $F(\Delta_{\mathcal{D}-I})_{c_6}$, *for* $i = 1, 2, \cdots, n$, *is ordered by* $\preccurlyeq$, *as a subset of* $\Delta_{\mathcal{D}}$.

□

Now we need to guarantee that $(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*,\ \forall \gamma^* : \alpha^* \in R_{\gamma:\alpha}$. In order to fulfill this condition, we apply the notion of safe-maximal subset on $(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})$. First we get the set $Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$, relative to the condition $c_7 = (\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*$. Then we get the safe maximal subset $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$.

Remark 2.22 ($Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and an input* $\gamma : \alpha$, *such that the set* $R_{\gamma:\alpha} \neq \emptyset$, *and the condition* $c_7 = (\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*,\ \forall \gamma^* : \alpha^* \in R_{\gamma:\alpha}$. *According to Definition 2.39,* $Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$ *is the following set:*

$$Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} = \begin{cases} \emptyset & \text{if } c_7 \text{ is satisfied;} \\ \{S |\ S \subseteq (\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}); \\ \exists \gamma^* : \alpha^* \in R_{\gamma:\alpha},\ s.\ t.\ S \vdash \gamma^* : \alpha^*; \\ \text{and } S \text{ is minimal w.r.t. } \subseteq\} & \text{otherwise.} \end{cases}$$

Where each set S_i *of* $Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$, *for* $i = 1, 2, \cdots, n$, *is ordered by* $\preccurlyeq$, *as a subset of* $\Delta_{\mathcal{D}}$.

•

Remark 2.23 ($Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *a declarative unit* $\gamma : \alpha$, *such that* $R_{\gamma:\alpha}$ *is non-empty, and the condition* $c_7 = (\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*,\ \forall \gamma^* : \alpha^* \in R_{\gamma:\alpha}$. *By Definition 2.43, we have that* $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$ *is the following set:*

$$Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} = \begin{cases} (\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}), \\ \ \text{if } c_7 \text{ is satisfied;} \\ (\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}) - RMin(Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}), \\ \ \text{otherwise.} \end{cases}$$

$Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$ *is ordered by* $\preccurlyeq$, *as a subset of* $\Delta_{\mathcal{D}}$.

•

Proposition 2.5

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *for any set* $R_{\gamma:\alpha} \subseteq \Delta_{\mathcal{D}-I}$, *we have that* $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} \nvdash \gamma^* : \alpha^*,\ \forall \gamma^* : \alpha^* \in R_{\gamma:\alpha}$.

We still need to guarantee that $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} \bigcup \{\gamma : \alpha\} \bigcup I \nvdash \gamma^* : \bot$, for any $\gamma^* \in \mathcal{L}_\gamma$. Hence, we get a safe-maximal subset of $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$ which guarantees this condition.

Remark 2.24 ($Fail(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *an input* $\gamma : \alpha$, *such that the set* $R_{\gamma:\alpha} \neq \emptyset$, *and the condition* $c_8 = Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} \bigcup \{\gamma : \alpha\} \bigcup I \nvdash \gamma^* : \bot$, *for any* $\gamma^* \in \mathcal{L}_\gamma$. *According to Definition 2.39,* $Fail(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$ *is the following set:*

$$Fail(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8} = \begin{cases} \emptyset, \\ \textit{if } c_8 \textit{ is satisfied;} \\ \\ \{S |\ S \subseteq Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}; \\ \textit{s. t. } S \bigcup \{\gamma:\alpha\} \nvdash \gamma^* : \bot; \\ \textit{and } S \textit{ is minimal w.r.t. } \subseteq\}, \\ \textit{otherwise.} \end{cases}$$

Where each set S_i *of* $Fail(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$, *for* $i = 1, 2, \cdots, n$, *is ordered by* $\preccurlyeq$, *as a subset of* $\Delta_{\mathcal{D}}$.

•

Remark 2.25 ($Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *a declarative unit* $\gamma : \alpha$, *such that* $R_{\gamma:\alpha}$ *is non-empty, and the condition* $c_8 = Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} \bigcup \{\gamma:\alpha\} \bigcup I \nvdash \gamma^* : \bot$, *for any* $\gamma^* \in \mathcal{L}_\gamma$. *By Definition 2.43, we have that* $Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$ *is the following set:*

$$Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8} = \begin{cases} Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}, \\ \textit{if } c_8 \textit{ is satisfied;} \\ \\ Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} - \\ RMin(Fail(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}), \\ \textit{otherwise.} \end{cases}$$

$Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$ *is ordered by* $\preccurlyeq$, *as a subset of* $\Delta_{\mathcal{D}}$.

•

Proposition 2.6

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and a declarative unit* $\gamma : \alpha$, *such that* $R_{\gamma:\alpha}$ *is non-empty,* $Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8} \bigcup \{\gamma:\alpha\} \bigcup I \nvdash \gamma^* : \bot$, *for any* $\gamma^* \in \mathcal{L}_\gamma$.

Given that $R^*_{\gamma:\alpha} = \Delta_{\mathcal{D}-I} - Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$, and that $R^*_{\gamma:\alpha}$ is effectively the set of declarative units that we have to retract from $\Delta_{\mathcal{D}}$, when an input $\gamma : \alpha$ conflicts with $\mathcal{D}$, we have already guaranteed that retracting $R^*_{\gamma:\alpha}$ from $\Delta_{\mathcal{D}}$ will not make any of its elements derivable from the resulting set. However, the contraction for compromised revision allows the consistent consequences of the of declarative units to be retracted, to become available in the resulting revised database. Hence, we also have to cater for introducing the consequences of the elements of $R^*_{\gamma:\alpha}$ w.r.t. $\mathcal{D}$, provided that they do not conflict with the database $\langle (Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8} \cup \{\gamma:\alpha\}), \preccurlyeq \rangle$.

Definition 2.49 below caters for the set of consequences of the elements of $R^*_{\gamma:\alpha}$. We call such a set $CR(R^*_{\gamma:\alpha})$.

Definition 2.49 ($CR(R^*_{\gamma:\alpha})$)

Given a knowledge base $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$ *and a set* $R^*_{\gamma:\alpha} \in \Delta_{\mathcal{D}}$, *such that* $R^*_{\gamma:\alpha}$ *should be retracted from* $\Delta_{\mathcal{D}}$, *such so that it is guaranteed that* $\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha} \nvdash \gamma^* : \alpha^*$, $\forall\, \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$. *Let* $CR(R^*_{\gamma:\alpha})$ *be the set of consequences of the elements in* $R^*_{\gamma:\alpha}$ *w.r.t.* $\mathcal{D}$, *considering* $\vdash$, *such that:*

$$\begin{aligned} CR(R^*_{\gamma:\alpha}) = \ & \{\gamma' : \alpha' \mid \Delta_{\mathcal{D}} \vdash \gamma' : \alpha' \textit{ and } \Delta_{\mathcal{D}} - R^*_{\gamma:\alpha} \nvdash \gamma' : \alpha', \\ & \textit{for } \gamma' \in \mathcal{L}_\gamma; \textit{ and } \forall \gamma^* : \alpha^* \in R^*_{\gamma:\alpha},\ \alpha' \neq \alpha^*, \textit{ and} \\ & \Delta_{\mathcal{D}} - R^*_{\gamma:\alpha} \cup \{\gamma' : \alpha'\} \nvdash \gamma^* : \alpha^*\}, \end{aligned}$$

where $CR(R^*_{\gamma:\alpha})$ *is maximal w.r.t.* $\subseteq$, *and is ordered by* $\preccurlyeq$, *according to Definition 2.16.*

□

Our objective is to include the set $CR(R^*_{\gamma:\alpha})$ in $\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}$, without being able to derive any element of $R^*_{\gamma:\alpha}$ from the resulting base. But with Definition 2.49, this is not yet possible. We still need to specify

a subset of $CR(R^*_{\gamma:\alpha})$, that satisfies this condition. However, we do not need to check for consistency of the base $\langle(\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \cup CR(R^*_{\gamma:\alpha}),\ \preccurlyeq\rangle$, since the original base, $D = \langle\Delta_{\mathcal{D}},\ \preccurlyeq\rangle$, is assumed to be consistent and as $R^*_{\gamma:\alpha} \subseteq \Delta_{\mathcal{D}}$, $CR(R^*_{\gamma:\alpha}) \subseteq Cn(\Delta_{\mathcal{D}})$, and $Cn(\Delta_{\mathcal{D}})$ is consistent.

We then state formally the subset of $CR(R^*_{\gamma:\alpha})$, by obtaining its safe-maximal subset, relative to condition $c_9 = CR(R^*_{\gamma:\alpha}) \bigcup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*, \forall\, \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$.

First, we obtain the set $Fail(CR(R^*_{\gamma:\alpha}))_{c_9}$, which contains all the minimal subsets of $CR(R^*_{\gamma:\alpha})$ that contribute to the failure of the condition c_9.

Remark 2.26 ($Fail(CR(R^*_{\gamma:\alpha}))_{c_9}$)

Given a knowledge base $\mathcal{D} = \langle\Delta_{\mathcal{D}},\ \preccurlyeq\rangle$, a set $R^*_{\gamma:\alpha} \in \Delta_{\mathcal{D}}$, such that $CR(R^*_{\gamma:\alpha})$ is non-empty, and the condition $c_9 = CR(R^*_{\gamma:\alpha}) \bigcup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*, \forall\, \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$. By Definition 2.39, we have that $Fail(CR(R^*_{\gamma:\alpha}))_{c_9}$ is the following set:

$$Fail(CR(R^*_{\gamma:\alpha}))_{c_9} = \begin{cases} \emptyset & \text{if } c_9 \text{ is satisfied;} \\ \{S \mid S \subseteq CR(R^*_{\gamma:\alpha}); & \\ \text{such that } \exists \gamma^* : \alpha^* \in CR(R^*_{\gamma:\alpha}), & \\ S \cup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \vdash \gamma^* : \alpha^*, \text{and} & \\ S \text{ is minimal w.r.t. } \subseteq\} & \text{otherwise.} \end{cases}$$

Each set S_i of $Fail(CR(R^*_{\gamma:\alpha}))_{c_9}$, for $i = 1, \cdots, k$, is ordered by $\preccurlyeq$ as a subset of $CR(R^*_{\gamma:\alpha})$.

•

Now we build up the set $Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$ using both $CR(R^*_{\gamma:\alpha})$ and $RMin(Fail(CR(R^*_{\gamma:\alpha}))_{c_9})$.

Remark 2.27 ($Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$)

Given a knowledge base $\mathcal{D} = \langle\Delta_{\mathcal{D}},\ \preccurlyeq\rangle$, a set $R^*_{\gamma:\alpha} \in \Delta_{\mathcal{D}}$, such that $CR(R^*_{\gamma:\alpha})$ is non-empty, and the condition $c_9 = CR(R^*_{\gamma:\alpha}) \bigcup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*, \forall \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$. By Definition 2.43, we have that $Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$ is the following set:

$$Smax(CR(R^*_{\gamma:\alpha}))_{c_9} = \begin{cases} CR(R^*_{\gamma:\alpha}) & \\ \quad \text{if } c_9 \text{ is satisfied;} & \\ CR(R^*_{\gamma:\alpha}) - RMin(Fail(CR(R^*_{\gamma:\alpha}))_{c_9}) & \\ \quad \text{otherwise.} \end{cases}$$

$Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$ is ordered by $\preccurlyeq$, as a subset of $CR(R^*_{\gamma:\alpha})$.

•

Proposition 2.7

Given a knowledge base $\mathcal{D} = \langle\Delta_{\mathcal{D}},\ \preccurlyeq\rangle$, for any set $R^*_{\gamma:\alpha} \subseteq \Delta_{\mathcal{D}}$, such that $CR(R^*_{\gamma:\alpha})$ is non-empty, $Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \cup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*,\ \forall \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$.

The proposition below guarantees that the safe maximal subset of the consequences of $R^*_{\gamma:\alpha}$, $Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$, when added to $(\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \bigcup \{\gamma : \alpha\}$, does not generate inconsistency.

Proposition 2.8

Given a database $\mathcal{D} = \langle\Delta_{\mathcal{D}},\ \preccurlyeq\rangle$, and an input $\gamma : \alpha$, such that it is not the case that $\neg\alpha$ is a tautology. If I, $\gamma : \alpha \nvdash \gamma' : \bot$ and $\Delta_{\mathcal{D}}$, $\gamma : \alpha \vdash \gamma' : \bot$, for any set $R_{\gamma:\alpha}$, as in Definition 2.48, such that $CR(R^*_{\gamma:\alpha})$ is non-empty, $Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \bigcup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, for any label $\gamma' \in \mathcal{L}_\gamma$.

The Compromised Revision Function for CIU_{LDS}

Definitions for the case that $I, \gamma : \alpha \not\vdash \gamma' : \bot$ and $\Delta_{\mathcal{D}}, \gamma : \alpha \vdash \gamma' : \bot$:

Given $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, and $c_6 = \Delta_{\mathcal{D}} \bigcup \{\gamma : \alpha\} \not\vdash \gamma' : \bot$, for $\gamma' \in \mathcal{L}_\gamma$,

$$R_{\gamma:\alpha} = RMin(F(\Delta_{\mathcal{D}-I})_{c_6}).$$

$$F(\Delta_{\mathcal{D}-I})_{c_6} = \begin{cases} \emptyset & \text{if } c_6 \text{ is satisfied;} \\ \{S \mid S \subseteq \Delta_{\mathcal{D}-I}; & \\ S \cup \{\gamma : \alpha\} \vdash \gamma' : \bot; & \\ \text{and } S \text{ is minimal w.r.t. } \subseteq\} & \text{otherwise.} \end{cases}$$

Given $c_7 = (\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}) \not\vdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R_{\gamma:\alpha}$,

$$Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} = \begin{cases} \emptyset & \text{if } c_7 \text{ is satisfied;} \\ \{S \mid S \subseteq (\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}); & \\ \exists \gamma^* : \alpha^* \in R_{\gamma:\alpha}, & \\ \text{such that } S \vdash \gamma^* : \alpha^*; & \\ \text{and } S \text{ is minimal w.r.t. } \subseteq\} & \text{otherwise.} \end{cases}$$

$$Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} =$$
$$\begin{cases} (\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}), & \\ \text{if } c_7 \text{ is satisfied;} & \\ (\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}) - RMin(Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}), & \\ \text{otherwise.} & \end{cases}$$

Given $c_8 = Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} \bigcup \{\gamma : \alpha\} \bigcup I \not\vdash \gamma^* : \bot$, for any $\gamma^* \in \mathcal{L}_\gamma$,

$$Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8} =$$
$$\begin{cases} Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}, & \\ \text{if } c_8 \text{ is satisfied;} & \\ Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} - & \\ RMin(Fail(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}), & \\ \text{otherwise.} & \end{cases}$$

Given $R^*_{\gamma:\alpha} = \Delta_{\mathcal{D}-I} - Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$,

$$CR(R^*_{\gamma:\alpha}) = \{\gamma' : \alpha' \mid \Delta_{\mathcal{D}} \vdash \gamma' : \alpha' \text{ and } \Delta_{\mathcal{D}} - R^*_{\gamma:\alpha} \not\vdash \gamma' : \alpha',$$
$$\text{for } \gamma' \in \mathcal{L}_\gamma; \text{ and } \forall \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}, \alpha' \neq \alpha^*, \text{ and}$$
$$\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha} \bigcup \{\gamma' : \alpha'\} \not\vdash \gamma^* : \alpha^*\}$$

Given $c_9 = CR(R^*_{\gamma:\alpha}) \bigcup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \not\vdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$,

$$Smax(CR(R^*_{\gamma:\alpha}))_{c_9} = \begin{cases} CR(R^*_{\gamma:\alpha}), & \\ \text{if } c_9 \text{ is satisfied;} & \\ CR(R^*_{\gamma:\alpha}) - & \\ RMin(Fail(CR(R^*_{\gamma:\alpha}))_{c_9}), & \\ \text{otherwise.} & \end{cases}$$

Figure 6: Summary of Auxiliary Definitions for Revision - Case (3).

The definition of the compromised revision function $\odot$ is based on the three possible cases below.

- If the database augmented by the requested input $\gamma : \alpha$ is inconsistent because α contradicts a tautology of the logical system, then we make the database consistent by rejecting the input. $\mathcal{D} \odot \gamma : \alpha = \mathcal{D}$.
- If the database is inconsistent because $I,\ \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, and it is not the case that $\neg\alpha$ is a tautology. Then, we regain consistency by rejecting $\gamma : \alpha$, but allowing its consistent consequences to be added to the database. In this case, $\mathcal{D} \odot \gamma : \alpha = \langle((\Delta_\mathcal{D} \bigcup Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}),\ \preccurlyeq\rangle$.
- If the database is inconsistent because $\Delta_\mathcal{D},\ \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, when $I,\ \gamma : \alpha \nvdash \gamma' : \bot$, then we regain consistency by keeping $\gamma : \alpha$ in the database and rejecting from it a safe-minimal number of declarative units that are not protected. We allow the inclusion of the consistent consequences of the retracted sentences, as a compromise. In this case, $\mathcal{D} \odot \gamma : \alpha = \langle(((\Delta_\mathcal{D} - R^*_{\gamma:\alpha}) \bigcup Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \bigcup \{\gamma : \alpha\}),\ \preccurlyeq\rangle$.

We formalize now the definition of the compromised revision function considering the steps described above.

Definition 2.50 (Compromised Revision Function)

Given a database $\mathcal{D} = \langle\Delta_\mathcal{D},\ \preccurlyeq\rangle$ and an input $\gamma : \alpha$, let the operation $\odot$ denote the compromised revision of $\mathcal{D}$ by $\gamma : \alpha$, such that the result $\mathcal{D} \odot \gamma : \alpha$ is a new database with the same structure of $\mathcal{D}$. We denote by $\Delta_{\mathcal{D} \odot \gamma:\alpha}$ the resulting set of declarative units of $\mathcal{D} \odot \gamma : \alpha$, such that $\mathcal{D} \odot \gamma : \alpha = \langle\Delta_{\mathcal{D}\odot\gamma:\alpha},\ \preccurlyeq\rangle$. And $\Delta_{\mathcal{D}\odot\gamma:\alpha}$ is such that one of the following conditions holds:

(Case 1) *$I,\ \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, and $\neg\alpha$ is a tautology, then $\Delta_{\mathcal{D}\odot\gamma:\alpha} = \Delta_\mathcal{D}$.*

(Case 2) *$I,\ \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, and it is not the case that $\neg\alpha$ is a tautology, then $\Delta_{\mathcal{D}\odot\gamma:\alpha} = \Delta_\mathcal{D} \bigcup Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$.*

(Case 3) *If $\Delta_\mathcal{D}, \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, and $I, \gamma : \alpha \nvdash \gamma' : \bot$, then $\Delta_{\mathcal{D}\odot\gamma:\alpha} = (\Delta_\mathcal{D} - R^*_{\gamma:\alpha}) \bigcup Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \bigcup \{\gamma : \alpha\}$.*

□

The proposition below states that all the consistent consequences of an input $\gamma : \alpha$, which violates some integrity constraints of a database $\mathcal{D}$, are available in $\mathcal{D} \odot \gamma : \alpha$.

Proposition 2.9

Given a base $\mathcal{D} = \langle\Delta_\mathcal{D},\ \preccurlyeq\rangle$, for any input $\gamma : \alpha$, such that $\neg\alpha$ is not a tautology, and $CI(\gamma : \alpha)$ is non-empty, if $I,\ \gamma : \alpha \vdash \gamma' : \bot$, for any label $\gamma' \in \mathcal{L}_\gamma$, then $\forall\gamma^* : \alpha^* \in Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$, $\mathcal{D} \nvdash \gamma^* : \alpha^*$ and $\mathcal{D} \odot \gamma : \alpha \vdash \gamma^* : \alpha^*$.

The proposition to follow states that all the consistent consequences of the retracted declarative units from $\Delta_\mathcal{D}$ w.r.t. an input $\gamma : \alpha$, in order to achieve $\mathcal{D} \odot \gamma : \alpha$, are derivable from $\mathcal{D} \odot \gamma : \alpha$.

Proposition 2.10

Given a base $\mathcal{D} = \langle\Delta_\mathcal{D},\ \preccurlyeq\rangle$, for any input $\gamma : \alpha$, such that $\neg\alpha$ is not a tautology, if $\Delta_\mathcal{D}, \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, and $I, \gamma : \alpha \nvdash \gamma' : \bot$, then $\forall\gamma^* : \alpha^* \in Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$, $\mathcal{D} \vdash \gamma^* : \alpha^*$ and also $\mathcal{D} \odot \gamma : \alpha \vdash \gamma^* : \alpha^*$.

The two propositions to follow, state important conditions concerning the declarative units in $\mathcal{D} \odot \gamma : \alpha$.

Proposition 2.11

Given a base $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *for any input* $\gamma : \alpha$, *such that* $\neg\alpha$ *is not a tautology, if* $\Delta_{\mathcal{D}}, \gamma : \alpha \vdash \gamma' : \bot$, *for some label* $\gamma' \in \mathcal{L}_{\gamma}$, *and* $I, \gamma : \alpha \nvdash \gamma' : \bot$, *then* $\forall \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$, $\mathcal{D} \odot \gamma : \alpha \nvdash \gamma^* : \alpha^*$.

Proposition 2.12

Given a base $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *for any input* $\gamma : \alpha$, *such that* $\neg\alpha$ *is not a tautology, if a declarative unit* $\gamma^* : \alpha^* \in \mathcal{D} \odot \gamma : \alpha$, *then either* $\gamma^* : \alpha^* \in \Delta_{\mathcal{D}}$ *and* $\gamma^* : \alpha^* \notin R^*_{\gamma:\alpha}$; *or* $\gamma^* : \alpha^* \notin \Delta_{\mathcal{D}}$ *and either* $\gamma^* : \alpha^* \in Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$, *or* $\gamma^* : \alpha^* = \gamma : \alpha$, *or* $\gamma^* : \alpha^* \in Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$.

Summary - Revision in CIU_{LDS}

We have introduced the revision $\odot$ in CIU_{LDS}, based on the compromised revision ® for finite bases with integrity constraints defined in [Dar-96c]. Since in CIU_{LDS} the revision function is part of the update function, some cases of the revision ® were embedded by the update definition and were not defined within the revision $\odot$. So, $\odot$ is basically devoted to free the database from inconsistency by allowing some consequences from conflicting updates, or from retracted declarative units, to be kept in the resulting database as a compromise. We have adopted the notion of safe-maximality introduced in [Dar-96c], as an impartial solution to choose among the minimal declarative units of the labelled database, considering the ordering $\preccurlyeq$. Basically, the safe-maximality mechanism can be described as a procedure which is applied whenever a condition c needs to be satisfied for defining a subset out of an ordered set X. So, first we get the set $Fail(X)_c$, which contains all the minimal subsets of X which fail to accomplish condition c. Then, we get the set $Smax(X)_c$, which is a subset of X which satisfies c, subtracted by the set $RMin(Fail(X)_c)$. This last set contains at least one minimal element, w.r.t. the ordering on X, of each minimal sets in $Fail(X)_c$. In [Dar-96c], we have also allowed for user interaction when defining some of the safe-maximal subsets, so that we would achieve minimal change in the original set. This approach could also be incorporated in the safe-maximality notion used in CIU_{LDS}, if required by the user. Some properties of the revision function $\odot$ in CIU_{LDS}, were presented in this section. In Section 2.7.2, we also prove consistency and a compromised version of persistence for the revision $\odot$. In the next section, we extend the notion of single updates to to the notion of transactions, which involves a sequence of updates.

2.6.6 Transactions

Basically, a transaction in CIU_{LDS} is a sequence of updates to be performed on a given labelled database.

Definition 2.51 below, presents formally the notion of updates transactions in CIU_{LDS}.

Definition 2.51 (CIU_{LDS} Transactions)

Let $\mathcal{D}_{CIU}$ *be the set of* CIU_{LDS} *databases and let* $\mathcal{D}$ *and* $\mathcal{D}'$ *be two* CIU_{LDS} *databases. Let* $UP = Up_1, \cdots, Up_n$ *be a sequence of update functions where for each* $1 \leq i \leq n$ $Up_i(\mathcal{D}_i, \sigma_i, \delta_i) \rightarrow \mathcal{D}'_i$ *is such that* $\mathcal{D}_i = \mathcal{D}'_{i-1}$, *for* $i = 1$ $\mathcal{D}_1 = \mathcal{D}$, *and for* $i = n$ $\mathcal{D}'_n = \mathcal{D}'$. *Let* UP_{seq} *be the set of all the sequences of updates of this form. A* CIU_{LDS} *Transaction is a function* $Trans : \mathcal{D}_{CIU} \times UP_{seq} \rightarrow \mathcal{D}_{CIU}$, *such that* $Trans(\mathcal{D}, UP) = \mathcal{D}'$.

□

A transaction, then incorporates the results of various single updates in sequence. Within each update request Up_i of a transaction, compromised revision $\odot$ may apply. Since, by the definition of single updates, the resulting updated databases are consistent, then we also have that the result of a transaction is a consistent labelled database.

In Section 2.7.2, we show the consistency property for a CIU_{LDS} transaction, and also the property of database structural properties preservation.

Definition for Compromised Revision in CIU_{LDS}:

- Compromised Revision Function:

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$ and an input $\gamma : \alpha$,

$$\mathcal{D} \odot \gamma : \alpha = \langle \Delta_{\mathcal{D} \odot \gamma:\alpha}, \preccurlyeq \rangle$$

where $\Delta_{\mathcal{D} \odot \gamma:\alpha}$ is such that one of the following conditions holds:

(Case 1) $I, \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, and $\neg\alpha$ is a tautology, then $\Delta_{\mathcal{D} \odot \gamma:\alpha} = \Delta_{\mathcal{D}}$.

(Case 2) $I, \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, and it is not the case that $\neg\alpha$ is a tautology, then $\Delta_{\mathcal{D} \odot \gamma:\alpha} = \Delta_{\mathcal{D}} \bigcup Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$.

(Case 3) If $\Delta_{\mathcal{D}}, \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, and $I, \gamma : \alpha \nvdash \gamma' : \bot$, then $\Delta_{\mathcal{D} \odot \gamma:\alpha} = (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \bigcup Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \bigcup \{\gamma : \alpha\}$.

Figure 7: Compromised Revision in CIU_{LDS}.

2.7 Properties of the System CIU_{LDS}

In this section we show some properties of our system, concerning the consequence relation $\vdash_{CIU_{LDS}}$, the compromised revision, the updates, the basic update operations, and the transactions of CIU_{LDS}.

2.7.1 Properties of the CIU_{LDS} Consequence Relation

As already expected for a consequence relation which does not follow the classical standards, the CIU_{LDS} consequence relation relaxes both reflexivity and monotonicity conditions. We consider, however, a restricted version of reflexivity, and a weaker monotonicity condition, to be satisfied by $\vdash_{CIU_{LDS}}$.[39]

The Restricted Reflexivity Notion

In CIU_{LDS}, thi snotion means that if a declarative unit is present in the database, and it can be proved by the application of the conditional reflexivity inference rule, then we say that it is a consequence of the system.

Proposition 2.13 (Restricted Reflexivity) *Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$ and a declarative unit $\gamma : \alpha$, such that $\gamma : \alpha \in \Delta_{\mathcal{D}}$, then we say that $\mathcal{D} \vdash_{CIU_{LDS}} \gamma : \alpha$, provided that $\gamma : \alpha$ is derivable from $\mathcal{D}$ via the Conditional Reflexivity inference rule,* (CR).

Proof:

By Definition 2.30, $\gamma : \alpha \in C_{\text{CR}}$. Therefore $\langle \gamma : \alpha, \varphi_{\text{CR}}, \gamma : \alpha \rangle$ defines the inference rule CR. Then, by hypotheses φ_{CR} holds, and by Definition 2.27, we have that a proof of $\gamma : \alpha$ is given by $\rho[\gamma : \alpha] = \langle \{A_1/C_1\}, k \rangle$, where $k(1) = \text{CR}$, $A_1 = \{\gamma : \alpha\}$ and $C_1 = \gamma : \alpha$ Hence, by Definition 2.28, $\mathcal{D} \vdash \gamma : \alpha$.

■

Remark 2.28

The consequence relation $\vdash_{CIU_{LDS}}$ satisfies the property of restricted reflexivity, as long as it satisfies the application of the conditional reflexivity inference rule CR. *That is, provided that the condition applied to the* CR *inference rule holds, when required.*

•

The Restricted Monotonicity[40] Notion

The Restricted Monotonicity notion basically guarantees that the addition of a derivable sentence to a knowledge base does not cause any harm to further inferences of the base.

In our system, $\vdash_{CIU_{LDS}}$ satisfies this property as shown below.

Proposition 2.14 (Restricted Monotonicity) *Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, and the declarative units $\gamma_1 : \alpha$ and $\gamma_2 : \beta$, we say that the consequence relation $\vdash$ satisfies the property of Restricted Monotonicity, if the following rule is satisfied:*

[39] We will also refer to $\vdash_{CIU_{LDS}}$ as $\vdash$, for short, whenever its meaning is not compromised.

[40] We refer to this property as Restricted Monotonicity, following Gabbay's definition, in [Gab-94], concerning the basic axioms for a consequence relation of non-monotonic systems.

$$\frac{\mathcal{D} \vdash \gamma_1 : \alpha \quad \mathcal{D} \vdash \gamma_2 : \beta}{\mathcal{D},\ \gamma_1 : \alpha \vdash \gamma_2 : \beta}$$

Proof:

By Definition 2.27, if $\mathcal{D} \vdash \gamma_1 : \alpha$ there is a proof $\rho[\gamma_1 : \alpha]$ from $\mathcal{D}$. The same applies to $\gamma_2 : \beta$. So, given $\rho[\gamma_1 : \alpha]$ and $\rho[\gamma_2 : \alpha]$, and that $\mathcal{D}$ is assumed to be consistent, then for any $\gamma_1 : \alpha$ and $\gamma_2 : \beta$ such that it is not the case that $\vdash \neg\alpha$ and $\vdash \neg\beta$, it is guaranteed that we can still show a proof $\rho[\gamma_2 : \alpha]$ from $\mathcal{D} \bigcup \{\gamma_1 : \alpha\}$, since $\mathcal{D} \bigcup \{\gamma_1 : \alpha\}$ does not imply any other changes on $\mathcal{D}$. Hence by Definition 2.28, we have that $\mathcal{D},\ \gamma_1 : \alpha \vdash \gamma_2 : \beta$, as we wanted to prove.

■

The Restricted Monotonicity property also applies to CIU_{LDS}, when we consider the conditional inclusion operation $\biguplus$ instead of set inclusion for adding the derivable sentence to the base. If a declarative unit is a consequence of the database, then if we add it to the database via the conditional inclusion operation, restricted to some labelling conditions, this should not interfere with further inferences of the system.

Proposition 2.15 (Restricted Monotonicity - $\biguplus$ version)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, and the declarative units $\gamma_1 : \alpha$ and $\gamma_2 : \beta$, we say that the consequence relation $\vdash$ satisfies the property of Restricted Monotonicity ($\biguplus$ version), if the following rule is satisfied:

$$\frac{\mathcal{D} \vdash \gamma_1 : \alpha \quad \mathcal{D} \vdash \gamma_2 : \beta}{\mathcal{D} \uplus \gamma_1' : \alpha \vdash \gamma_2' : \beta}$$

where $\mathcal{D} \biguplus \gamma_1' : \alpha$ is the conditional addition of the declarative unit $\gamma_1' : \alpha$ to the database $\mathcal{D}$ [41]*, such that $\gamma_1 \notin$ P, $\gamma_1' \in$ E $\cup$ I such that $\varphi_{\Cup}(\gamma_1', \gamma_1) \in \mathcal{A}$, where γ_1' is obtained as follows:*
If $\gamma_1 \in$ I and α is a clausal formula, then $\gamma_1' \in$ I.
If $\gamma_1 \in$ I and α is not a clausal formula, then $\gamma_1' \in$ E.
If $\gamma_1 \in$ E, then $\gamma_1' \in$ E.
If $\gamma_1 \in$ N, then $\gamma_1 \in$ E.
And γ_2' is restricted to the following condition:
If $\gamma_1 : \alpha = \gamma_2 : \beta$, then $\gamma_2' = \gamma_1'$. Otherwise, $\gamma_2' = \gamma_2$.

Proof:

By the Definition 2.38, since $\mathcal{D} \vdash \gamma_1 : \alpha$ and $\mathcal{D}$ is assumed to be consistent, we have that $\mathcal{D} \biguplus \gamma_1' : \alpha$ is consistent and is given by $\mathcal{D} \biguplus \gamma_1' : \alpha = \langle (\Delta_{\mathcal{D}} - X_{\mathcal{D}}) \cup \{\gamma_1' : \alpha\}, \preccurlyeq \rangle$, where $X_{\mathcal{D}} = \{\ \gamma_I : \alpha \mid \gamma_I : \alpha \in \Delta_{\mathcal{D}}$, and $\varphi_{\Cup}(\gamma_1', \gamma_I) \in \mathcal{A}\}$. By definition, $\varphi_{\Cup}(\gamma_1, \gamma_I)$ only holds for some cases that $\gamma_1' \in$ E $\cup$ I, in accordance with the proposition condition. Also, since $\mathcal{D} \biguplus \gamma_1' : \alpha$ does not revise the database, we have that $\forall \gamma_2 : \beta$, such that $\mathcal{D} \vdash \gamma_2 : \beta$, if $\gamma_2 : \beta \neq \gamma_1 : \alpha$, then $\mathcal{D} \biguplus \gamma_1' : \alpha \vdash \gamma_2 : \beta$. Otherwise, due to the fact that $\gamma_1' : \alpha$ is added to $\mathcal{D}$, following the conditions above, and also to the changes implied by the retraction of $X_{\mathcal{D}}$ from $\mathcal{D}$, we have that $\mathcal{D} \biguplus \gamma_1' : \alpha \vdash \gamma_2' : \beta$, where $\gamma_2' = \gamma_1'$. Hence, when all the stated conditions apply, it is guaranteed that the consequence relation $\vdash_{CIU_{LDS}}$ satisfies the property of restricted monotonicity.

■

[41] See Definition 2.38.

The Deduction Property Notion

We say that a consequence relation has the Deduction Property, if the implication operator of the language is such that, for all sentences A, B and sets of sentences Δ, $\Delta \vdash A \rightarrow B$ iff $\Delta, A \vdash B$. This property relates the meta-level implication of the consequence relation with the with the object-level one. For this reason, it cannot be enforced on the consequence relation by means of the system's inference rules.

CIU_{LDS} satisfies the deduction property, under some labelling restrictions, as shown below.

Proposition 2.16 (Deduction Property)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and the declarative units* $\gamma_1 : \alpha \rightarrow \beta$ *and* $\gamma_2 : \beta$, *we say that the consequence relation* $\vdash_{CIU_{LDS}}$ *satisfies the Deduction Property, if we have the following:*

$$\mathcal{D} \vdash \gamma_3 : \alpha \rightarrow \beta \quad \textit{iff} \quad \mathcal{D}, \gamma_1 : \alpha \vdash \gamma_2 : \beta$$

where γ_3 *is given by* $\gamma_1 \copyright_{\rightarrow \mathrm{I}} \gamma_2$, γ_2 *is given by the labelling conditions of the inference rules applied in the derivation of* $\gamma_2 : \beta$, *and* α *is not a clausal formula.*[42]

Proof:

We prove the if and only if of the proposition statement separately, in the cases (a) and (b) below.

(a) If $\mathcal{D}, \gamma_1 : \alpha \vdash \gamma_2 : \beta$, then $\mathcal{D} \vdash \gamma_3 : \alpha \rightarrow \beta$

By Definition 2.33, by applying $\rightarrow$ introduction to $\mathcal{D}, \gamma_1 : \alpha \vdash \gamma_2 : \beta$, we get that $\mathcal{D} \vdash \gamma_3 : \alpha \rightarrow \beta$, where $\gamma_3 = \gamma_1 \copyright_{\rightarrow \mathrm{I}} \gamma_2$, provided that $\varphi_{\rightarrow \mathrm{I}}(\gamma_1, \gamma_2) \in \mathcal{A}$.

(b) If $\mathcal{D} \vdash \gamma_3 : \alpha \rightarrow \beta$, then $\mathcal{D}, \gamma_1 : \alpha \vdash \gamma_2 : \beta$

By Definition 2.34, if we assume $\gamma_1 : \alpha$, by applying $\rightarrow$ elimination to $\mathcal{D} \vdash \gamma_3 : \alpha \rightarrow \beta$ and $\gamma_1 : \alpha$, we get $\gamma' : \beta$ such that $\gamma' = \gamma_1 \copyright_{\rightarrow \mathrm{E}} \gamma_3$, provided that $\varphi_{\rightarrow \mathrm{E}}(\gamma_1, \gamma_3) \in \mathcal{A}$. If we then consider that $\gamma_2 = \gamma'$, we show that $\mathcal{D}, \gamma_1 : \alpha \vdash \gamma_2 : \beta$.

From (a) and (b), we prove that $\mathcal{D} \vdash \gamma_3 : \alpha \rightarrow \beta$ iff $\mathcal{D}, \gamma_1 : \alpha \vdash \gamma_2 : \beta$. This way, the consequence relation $\vdash_{CIU_{LDS}}$ proves to satisfy this restricted version of the deduction theorem.

■

CIU_{LDS} also satisfies a restricted version of the Deduction Theorem, which justifies the introduction of the logical connective $\rightarrow$, by taking into account the conditional inclusion operation, and some labelling restrictions as shown below.

Proposition 2.17 (Deduction Property - $\biguplus$ version)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and the declarative units* $\gamma_1 : \alpha \rightarrow \beta$ *and* $\gamma_2 : \beta$, *we say that the consequence relation* $\vdash_{CIU_{LDS}}$ *satisfies the Deduction Property (* $\biguplus$ *version), if we have the following:*

[42] α is constrained from being a clausal formula, because the system does not support embedded implications.

$$\mathcal{D} \vdash \gamma_3 : \alpha \rightarrow \beta \quad \textit{iff} \quad \mathcal{D} \uplus \gamma_1 : \alpha \vdash \gamma_2 : \beta$$

where γ_3 *is given by* $\gamma_1 \copyright_{\rightarrow \mathrm{I}} \gamma_2$, $\gamma_1 \in \mathrm{E} \cup \mathrm{I}$, $\gamma_2 \notin \mathrm{N}$,[43] *and* α *is not a clausal formula.*

Proof:

We prove the if and only if of the proposition statement separately, in the cases (a) and (b) below.

(a) If $\mathcal{D} \uplus \gamma_1 : \alpha \vdash \gamma_2 : \beta$, then $\mathcal{D} \vdash \gamma_3 : \alpha \rightarrow \beta$

By hypotheses and by Definition 2.38, we have that $\gamma_1 \in \mathrm{E}$ since α is not a clausal formula. This hypotheses also allows the application of the $(\rightarrow \mathrm{I})$ inference rule. Since by hypotheses, $\gamma_2 \notin \mathrm{N}$, then even if the operation $\mathcal{D} \biguplus \gamma_1 : \alpha$ revises the database, the application of the $(\rightarrow \mathrm{I})$ inference rule is restricted to the non-compromised case. That is, the derived declarative unit $\gamma_3 : \alpha \rightarrow \beta$ does not depend on any non-supported compromised consequence, introduced by the revision function. Hence, in the case that $\gamma_1 : \alpha \in \mathcal{D} \uplus \gamma_1 : \alpha$, by hypotheses and by Definition 2.27, we have that a proof of $\gamma_3 : \alpha \rightarrow \beta$, $\rho[\gamma_3 : \alpha \rightarrow \beta] = \langle \{A_1/C_1, \cdots, A_n/C_n\}, k \rangle$, where $k(1) = (\rightarrow \mathrm{CR})$, $A_1 = \{\gamma_1 : \alpha\}$ and $C_1 = \gamma_1 : \alpha$; for some $k(n-1)$, $\gamma_1 : \beta \in A_{n-1}$ and $C_{n-1} = \gamma_2 : \beta$; and $k(n) = (\rightarrow \mathrm{I})$, $A_n = \{\gamma_1 : \alpha, \gamma_2 : \beta\}$ and $C_n = \gamma_3 : \alpha \rightarrow \beta$, where γ_3 is given by $\gamma_1 \copyright_{\rightarrow \mathrm{I}} \gamma_2$. Hence, by Definition 2.28, $\mathcal{D} \vdash \gamma_3 : \alpha \rightarrow \beta$. In the case that $\gamma_1 : \alpha \notin \mathcal{D} \uplus \gamma_1 : \alpha$, that would mean that if $\mathcal{D} \uplus \gamma_1 : \alpha \vdash \gamma_2 : \beta$, $\gamma_2 \in \mathrm{N}$ and this case is constrained by the proposition statement.

(b) If $\mathcal{D} \vdash \gamma_3 : \alpha \rightarrow \beta$, then $\mathcal{D} \uplus \gamma_1 : \alpha \vdash \gamma_2 : \beta$

By Definition 2.34, if we assume $\gamma_1 : \alpha$, by applying $\rightarrow$ elimination, we get $\gamma' : \beta$, where $\gamma' = \gamma_1 \copyright_{\rightarrow \mathrm{E}} \gamma_3$, provided that $\varphi_{\rightarrow \mathrm{E}}(\gamma_1, \gamma_3) \in \mathcal{A}$. We can consider that $\gamma_2 = \gamma'$, if we restrict that $\gamma_2 \notin \mathrm{N}$. So, we have that $\mathcal{D} \vdash \gamma_2 : \beta$. By Restricted Monotonicity, Proposition 2.15, from $\mathcal{D} \vdash \gamma_2 : \beta$ and the assumption that $\mathcal{D} \vdash \gamma_1 : \alpha$, we show that $\mathcal{D} \uplus \gamma_1 : \alpha \vdash \gamma_2 : \beta$.

From (a) and (b), we prove that $\mathcal{D} \vdash \gamma_3 : \alpha \rightarrow \beta$ iff $\mathcal{D} \uplus \gamma_1 : \alpha \vdash \gamma_2 : \beta$, considering that the restrictions required apply.

■

Remark 2.29 *It is important to notice that in the deduction property stated in Proposition 2.16,* $\gamma_1 : \alpha$, *in* $\mathcal{D}$, $\gamma_1 : \alpha \vdash \gamma_2 : \beta$, *can introduce inconsistency to the database. However,* $\gamma_2 : \beta$ *can still be derived, and so can* $\gamma_3 : \alpha \rightarrow \beta$, *since we show that the non-explosiveness property holds (Proposition 2.18). In the deduction property stated in Proposition 2.17, the same does not happen because we guarantee that* $\mathcal{D} \uplus \gamma_1 : \alpha \vdash$ *is consistent.*

•

The Notion of Strong Transitivity

In general, the full version of Transitivity, or Cut, is such that for all sentences A, B and sets of sentences Δ and Γ, if $\Delta \vdash A$ and Γ, $A \vdash B$, then Γ, $\Delta \vdash B$. Strong Transitivity[44] states the following variation of the cut rule: if $\Delta \vdash A$ and Δ, $A \vdash B$, then $\Delta \vdash B$.

[43] In this case, γ_2 is constrained from being of type N, because this would allow for the case in which $\gamma_1 : \alpha \notin \mathcal{D} \uplus \gamma_1 : \alpha$.

[44] Also referred to as Unitary Cut in [Gab-94].

CIU_{LDS} satisfies Strong Transitivity, provided that some labelling restrictions apply, as shown below.

Proposition 2.18 (Strong Transitivity)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and the declarative units* $\gamma_1 : \alpha$ *and* $\gamma_2 : \beta$, *such that* $\gamma_1 \notin$ P, *we say that* $\vdash_{CIU_{LDS}}$ *satisfies the property of Strong Transitivity, if the following rule is satisfied:*

$$\frac{\mathcal{D} \vdash \gamma_1 : \alpha \qquad \mathcal{D}, \gamma_1 : \alpha \vdash \gamma_2 : \beta}{\mathcal{D} \vdash \gamma_2^{'} : \beta}$$

where $\gamma_2^{'}$ *is given by the labelling conditions of the inference rules used for the derivation of* $\gamma_2^{'} : \beta$.

Proof:

By Definition 2.33, ($\rightarrow$ introduction), from $\mathcal{D}, \gamma_1 : \alpha \vdash \gamma_2 : \beta$, we get $\mathcal{D} \vdash \gamma_1 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2 : \alpha \rightarrow \beta$. By applying Definition 2.34, ($\rightarrow$ elimination), to $\mathcal{D} \vdash \gamma_1 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2 : \alpha \rightarrow \beta$ and $\mathcal{D} \vdash \gamma_1 : \alpha$, we get $\mathcal{D} \vdash \gamma_1 \textcircled{c}_{\rightarrow \mathrm{E}} (\gamma_1 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2) : \beta$. Hence, we have that $\mathcal{D} \vdash \gamma_2^{'} : \beta$, for $\gamma_2^{'} = \gamma_1 \textcircled{c}_{\rightarrow \mathrm{E}} (\gamma_1 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2)$, given that $\varphi_{\rightarrow \mathrm{E}}(\gamma_1, \gamma_1 \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2) \in \mathcal{A}$.

■

Strong Transitivity also holds for the case that we consider $\gamma_1 : \alpha$ added to $\mathcal{D}$, via the conditional addition operation. That is, when we consider $\mathcal{D} \uplus \gamma_1 : \alpha$, provided that some labelling restrictions apply, as shown below.

Proposition 2.19 (Strong Transitivity - $\biguplus$ version)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and the declarative units* $\gamma_1 : \alpha$ *and* $\gamma_2 : \beta$, *such that* $\gamma_1 \notin$ P, *we say that* $\vdash_{CIU_{LDS}}$ *satisfies the property of Strong Transitivity (* $\biguplus$ *version), if the following rule is satisfied:*

$$\frac{\mathcal{D} \vdash \gamma_1 : \alpha \qquad \mathcal{D} \uplus \gamma_1^{'} : \alpha \vdash \gamma_2 : \beta}{\mathcal{D} \vdash \gamma_2^{'} : \beta}$$

where $\gamma_2^{'}$ *is given by the labelling conditions of the inference rules used for the derivation of* $\gamma_2^{'} : \beta$, *and* $\gamma_1^{'} \in$ E ∪ I, *such that:*
If $\gamma_1 \in$ I *and* α *is a clausal formula, then* $\gamma_1^{'} \in$ I.
If $\gamma_1 \in$ I *and* α *is not a clausal formula, then* $\gamma_1^{'} \in$ E.
If $\gamma_1 \in$ E ∪ N, *then* $\gamma_1^{'} \in$ E.

Proof:

By hypotheses and by Definition 2.38, since $\mathcal{D} \vdash \gamma_1 : \alpha$ and $\mathcal{D}$ is assumed to be consistent, then $\mathcal{D} \biguplus \gamma_1^{'} : \alpha$ is consistent and is given by $\mathcal{D} \biguplus \gamma_1^{'} : \alpha = \langle (\Delta_{\mathcal{D}} - X_{\mathcal{D}}) \cup \{\gamma_1^{'} : \alpha\}, \preccurlyeq \rangle$, where $X_{\mathcal{D}} = \{ \gamma_l : \alpha \mid \gamma_l : \alpha \in \Delta_{\mathcal{D}}$, and $\varphi_{\uplus}(\gamma_1^{'}, \gamma_l) \in \mathcal{A} \}$. By definition, we have that $\varphi_{\uplus}(\gamma_1^{'}, \gamma_l)$ only holds for some cases that $\gamma_1^{'} \in$ E ∪ I, in accordance with the proposition condition. By the Deductive Property, Proposition 2.17, from $\mathcal{D} \uplus \gamma_1^{'} : \alpha \vdash \gamma_2 : \beta$, we have that $\mathcal{D} \vdash \gamma_3 : \alpha \rightarrow \beta$, where $\gamma_3 = \gamma_1^{'} \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2$, provided that $\varphi_{\rightarrow \mathrm{I}}(\gamma_1^{'}, \gamma_2) \in \mathcal{A}$. So, by Definition 2.34, if we $\rightarrow$ elimination to $\gamma_1 : \alpha$ and $\gamma_1^{'} \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2 : \alpha \rightarrow \beta$, we get $\gamma^{'} : \beta$, where $\gamma^{'} = \gamma_1 \textcircled{c}_{\rightarrow \mathrm{E}} (\gamma_1^{'} \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2)$, provided that $\varphi_{\rightarrow \mathrm{E}}(\gamma_1, \gamma_1^{'} \textcircled{c}_{\rightarrow \mathrm{I}} \gamma_2) \in \mathcal{A}$. We can consider then that $\gamma_2 = \gamma^{'}$, since no labelling restrictions is applied to $\gamma_2 : \beta$. Hence, we show that $\mathcal{D} \vdash \gamma_2^{'} : \beta$.

■

The Notion of Non-explosiveness

The Non-explosiveness is a notion considered by some pragmatic formalisms, when dealing with contradictory information and conflicting data in general. In traditional logical approaches, the system is forced to collapse when inconsistency is detected. As pointed out in [Wag-94],

> *"It seems to be an unnatural overreaction to abandon a knowledge base once it is discovered to be inconsistent. Rather, one should accomodate it by means of a logic which continues to function plausibly under inconsistency."*

CIU_{LDS} supports this viewpoint, and we show below the Non-explosiveness property for $\vdash_{CIU_{LDS}}$.

Proposition 2.20 (Non-explosiveness)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and a declarative unit* $\gamma : \alpha$ *such that* α *is not a tautology, we say that the consequence relation* $\vdash_{CIU_{LDS}}$ *satisfies Non-explosiveness, if when* $\mathcal{D}, \gamma : \alpha \vdash \gamma^* : \bot$, *we do not have that* $\mathcal{D}, \gamma : \alpha \vdash \gamma' : \alpha'$, *for any* $\gamma' : \alpha' \in \mathcal{L}_{CIU}$.

Proof:

By hypotheses, $\mathcal{D}, \gamma : \alpha \vdash \gamma^* : \bot$, so we have that $\mathcal{D} \nvdash \gamma : \alpha$ since, by initial assumption, $\mathcal{D} \vdash \gamma^* : \bot$, for any label γ^*. If we assume that $\mathcal{D} \nvdash \gamma' : \alpha'$, for a certain $\gamma' : \alpha' \in \mathcal{L}_{CIU}$, and also that $\{\gamma : \alpha\} \nvdash \gamma' : \alpha'$, then if by assuming that $\mathcal{D}, \gamma : \alpha \vdash \gamma' : \alpha'$ we reach a contradiction, we can show by *reductio ad absurdum* that $\mathcal{D}, \gamma : \alpha \nvdash \gamma' : \alpha'$. This suffices to prove non-explosiveness. So, by Proposition 2.14, we get that $\mathcal{D}, \gamma : \alpha \vdash \gamma' : \alpha'$, if we have that $\mathcal{D} \vdash \gamma : \alpha$ and $\mathcal{D} \vdash \gamma' : \alpha'$. By assumption, it is not the case that $\mathcal{D} \vdash \gamma' : \alpha'$, and by the proposition hypotheses, we can assure that $\mathcal{D} \nvdash \gamma : \alpha$. Since we do not manage to show that $\mathcal{D}, \gamma : \alpha \vdash \gamma' : \alpha'$, we can conclude then that $\mathcal{D}, \gamma : \alpha \nvdash \gamma' : \alpha'$, as we wanted to prove.

■

2.7.2 Properties of the Compromised Revision $\odot$

Database Structural Properties Preservation

We say that the revision function $\odot$ preserves the structural properties of a given initial database $\mathcal{D}$, if by revising $\mathcal{D}$ by an input $\gamma : \alpha$, the resulting database $\mathcal{D} \odot \gamma : \alpha$ presents the same structural organization of the initial one.

Proposition 2.21 below guarantees that the structural properties of a CIU_{LDS} database are kept via performances of the compromised revision $\odot$.

Proposition 2.21

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *for any input* $\gamma : \alpha$ *following the definition of* CIU_{LDS} *declarative units,* $\mathcal{D} \odot \gamma : \alpha$ *is another database with the same structure as* $\mathcal{D}$, $\mathcal{D} \odot \gamma : \alpha = \langle \Delta_{\mathcal{D} \odot \gamma : \alpha}, \preccurlyeq \rangle$.

Proof:

By Definition 2.50, for all the three cases that defines the revision $\odot$, $\mathcal{D} \odot \gamma : \alpha$ is a new database with the same structure of $\mathcal{D}$. $\mathcal{D} \odot \gamma : \alpha = \langle \Delta_{\mathcal{D} \odot \gamma:\alpha}, \preccurlyeq \rangle$.
In case 1, $\Delta_{\mathcal{D} \odot \gamma:\alpha} = \Delta_{\mathcal{D}}$.
In case 2, $\Delta_{\mathcal{D} \odot \gamma:\alpha} = \Delta_{\mathcal{D}} \bigcup Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$. By Remark 2.21, $Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$ is a set of declarative units also ordered by $\preccurlyeq$.
In case 3, $\Delta_{\mathcal{D} \odot \gamma:\alpha} = (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \bigcup Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \bigcup \{\gamma : \alpha\}$, where $R^*_{\gamma:\alpha}$ is a subset of $\Delta_{\mathcal{D}}$, $Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$ is a set of declarative units ordered by $\preccurlyeq$, and $\gamma : \alpha$ is added to the database with the highest position in the ordering $\preccurlyeq$.
Hence, for all the three cases which defines the revision $\odot$, $\mathcal{D} \odot \gamma : \alpha$ preserves the structural properties of the initial database $\mathcal{D}$.

■

Consistency

In order to show that the revision function $\odot$ is consistent, we need to prove that for any initial consistent database $\mathcal{D}$, and and any input $\gamma : \alpha$, if we revise $\mathcal{D}$ by the input, the resulting database $\mathcal{D} \odot \gamma : \alpha$ is consistent.

Below, we state the consistency theorem of the compromised revision $\mathcal{D} \odot \gamma : \alpha$.

Theorem 2.1 (Consistency of $\odot$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, for any declarative unit $\gamma : \alpha$, $\mathcal{D} \odot \gamma : \alpha \nvdash \gamma' : \bot$, for any label γ' of $\mathcal{L}_\gamma$.

Proof:

By Definition 2.50, we state that $\mathcal{D} \odot \gamma : \alpha = \langle \Delta_{\mathcal{D} \odot \gamma:\alpha}, \preccurlyeq \rangle$, where $\Delta_{\mathcal{D} \odot \gamma:\alpha}$ is obtained conditioned to three cases.

In case 1, we have that if $I, \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, and $\neg\alpha$ is a tautology, then $\Delta_{\mathcal{D} \odot \gamma:\alpha} = \Delta_{\mathcal{D}}$. So, it is guaranteed that $\mathcal{D} \odot \gamma : \alpha \nvdash \gamma' : \bot$, since by assumption $\mathcal{D} \nvdash \gamma' : \bot$, for any label $\gamma' \in \mathcal{L}_\gamma$.

In case 2, if $I, \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, and $\neg\alpha$ is not a tautology, then $\Delta_{\mathcal{D} \odot \gamma:\alpha} = \Delta_{\mathcal{D}} \bigcup Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$. By Proposition 2.3, we have proved that $Smax(CI(\gamma : \alpha))_{c_4} \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \bot$, for any label $\gamma' \in \mathcal{L}_\gamma$. Since by Definition ?? and Remark 2.21, $Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$ is a subset of $Smax(CI(\gamma : \alpha))_{c_4}$, we also have that $Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5} \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \bot$. Hence, we guarantee that in case 2 $\mathcal{D} \nvdash \gamma' : \bot$, for any label $\gamma' \in \mathcal{L}_\gamma$.

In case 3, if $\Delta_{\mathcal{D}}, \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, and $I, \gamma : \alpha \nvdash \gamma' : \bot$, then $\Delta_{\mathcal{D} \odot \gamma:\alpha} = (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \bigcup Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \bigcup \{\gamma : \alpha\}$. By Proposition 2.8, we have proved that $Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \bigcup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$. Hence, it is also guaranteed for case 3 that $\mathcal{D} \nvdash \gamma' : \bot$, for any label $\gamma' \in \mathcal{L}_\gamma$.

■

Persistence

As shown in [Dar-96c], compromised revision satisfies a restricted version of the persistence property, which we called *Compromised Persistence.*

Basically, the persistence notion states that as much of the original base should survive a revision as possible. Hence, by revising a database $\mathcal{D}$ with a sentence $\gamma : \alpha$ and then retracting $\gamma : \alpha$, we should be able to derive from the resulting database, all the consequences of $\mathcal{D}$ that do not directly contradict $\gamma : \alpha$. The compromised version of this persistence notion, considers that we should be able to derive from the resulting base all the consequences that do not do not directly contradict $\gamma : \alpha$, and also that do not violate integrity contraints in $\mathcal{D}$.

In CIU_{LDS}, the notion of compromised persistence involves both revision and the update operation, in the case of compromised contraction.

If we consider compare Definition ?? of revision ®, with Definition 2.50 of revision $\odot$, we notice that revision $\odot$ only caters for cases 1,3 and 4 of revision ®. Case 2 of ® is dealt directly by the update function in CIU_{LDS}.[45] Also the compromised contraction $\ominus$ defined in [Dar-96c], is implemented by the update $Up(\mathcal{D}, \mathtt{U}_{-}, \gamma : \alpha)$, where the compromised notion of keeping the consistent consequences of the retracted declarative units is embedded in the retraction operation $\mathcal{D} \,\Xi\, \gamma : \alpha$ (Definition 2.46).

Due to the differences cited above, we cannot abstract from the property of compromised persistence proved in [Dar-96c], and assume that it holds for revision $\odot$. Hence, we need to show that the this notion of persistent is also guaranteed in CIU_{LDS}. The proposition below caters for this matter.

Proposition 2.22 (Compromised Persistence in CIU_{LDS})

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *if we revise it by a declarative unit* $\gamma : \alpha$, *then* $\forall\, \gamma' : \alpha'$ *such that* $\mathcal{D} \vdash \gamma' : \alpha'$, $\gamma' : \alpha' \neq \gamma : \alpha$ *and* $\gamma' : \alpha' \notin R^{*}_{\gamma:\alpha}$, $\mathcal{D}' \vdash \gamma' : \alpha'$, *where* $\mathcal{D}' = Up(\mathcal{D} \odot \gamma : \alpha, \mathtt{U}_{-}, \gamma : \alpha)$.

Proof:

By Definition 2.50, we have that in case 1, $\mathcal{D} \odot \gamma : \alpha = \mathcal{D}$. And, by Definitions 2.37 and 2.46, $Up(\mathcal{D} \odot \gamma : \alpha, \mathtt{U}_{-}, \gamma : \alpha) = \mathcal{D}$. So, in this case it is vacuously guaranteed that $\forall\, \gamma' : \alpha'$ such that $\mathcal{D} \vdash \gamma' : \alpha'$, $\mathcal{D}' \vdash \gamma' : \alpha'$. In case 2, $\mathcal{D} \odot \gamma : \alpha = \langle \Delta_{\mathcal{D}} \bigcup Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}, \preccurlyeq \rangle$, where no declarative unit is retracted from $\Delta_{\mathcal{D}}$, and $\gamma : \alpha \notin \mathcal{D} \odot \gamma : \alpha$. So, $Up(\mathcal{D} \odot \gamma : \alpha, \mathtt{U}_{-}, \gamma : \alpha) = \mathcal{D}'$, where $\mathcal{D}' = \mathcal{D} \odot \gamma : \alpha$. And we guarantee that $\forall\, \gamma' : \alpha'$ such that $\mathcal{D} \vdash \gamma' : \alpha'$, $\mathcal{D}' \vdash \gamma' : \alpha'$. In case 3, $\mathcal{D} \odot \gamma : \alpha = \langle \Delta_{\mathcal{D} \odot \gamma:\alpha}, \preccurlyeq \rangle$, where $\Delta_{\mathcal{D} \odot \gamma:\alpha} = (\Delta_{\mathcal{D}} - R^{*}_{\gamma:\alpha}) \bigcup Smax(CR(R^{*}_{\gamma:\alpha}))_{c_9} \bigcup \{\gamma : \alpha\}$. By Definitions 2.37 and 2.46, $Up(\mathcal{D} \odot \gamma : \alpha, \mathtt{U}_{-}, \gamma : \alpha) = \mathcal{D}'$, where $\mathcal{D}' = \langle (\Delta_{\mathcal{D}} \bigcup Smax(CCon(\gamma : \alpha))_{c_2}) - \{\gamma : \alpha\}, \preccurlyeq \rangle$, By definition, the consequences of $\mathcal{D}$ which are not anymore available from $\mathcal{D} \odot \gamma : \alpha$ are the ones in $R^{*}_{\gamma:\alpha}$, where $R^{*}_{\gamma:\alpha} \subseteq \Delta_{\mathcal{D}}$. As proved in Proposition 2.11 $\forall \gamma^{*} : \alpha^{*} \in R^{*}_{\gamma:\alpha}$, $\mathcal{D} \odot \gamma : \alpha \nvdash \gamma^{*} : \alpha^{*}$. So, we have that $\forall\, \gamma' : \alpha'$ such that $\mathcal{D}' \vdash \gamma' : \alpha'$, $\gamma' : \alpha' \notin R^{*}_{\gamma:\alpha}$, since by retracting $\{\gamma : \alpha\}$ from $\mathcal{D} \odot \gamma : \alpha$ the declarative units of the set $Smax(CCon(\gamma : \alpha))_{c_2}$ do not derive any declarative unit of $R^{*}_{\gamma:\alpha}$. By Definitions 2.37 and 2.46, it is guaranteed that if $\gamma : \alpha$ is derivable from $\mathcal{D} \odot \gamma : \alpha$, it is no longer derivable from $\mathcal{D}'$. Hence, for the three cases that defines the revision $\odot$, compromised persistence is satisfied. So, $\forall\, \gamma' : \alpha'$ such that $\mathcal{D} \vdash \gamma' : \alpha'$, $Up(\mathcal{D} \odot \gamma : \alpha, \mathtt{U}_{-}, \gamma : \alpha) \vdash \gamma' : \alpha'$, provided that $\gamma' : \alpha' \neq \gamma : \alpha$ and $\gamma' : \alpha' \notin R^{*}_{\gamma:\alpha}$, as stated in the proposition.

■

2.7.3 Properties of the CIU_{LDS} Updates

Database Structural Properties Preservation

[45] See Definition 2.37.

We say that the update function preserves the structural properties of a given database $\mathcal{D}$, if by updating $\mathcal{D}$ with an input $\gamma : \alpha$, the resulting database $\mathcal{D}' = Up(\mathcal{D}, \sigma, \gamma : \alpha)$, where $\sigma = \{\mathrm{U}_+, \mathrm{U}_-\}$, presents the same structural organization of $\mathcal{D}$.

Proposition 2.23 below guarantees that the structural properties of a CIU_{LDS} database are kept via single updates performances.

Proposition 2.23

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, for any update type σ, $Up(\mathcal{D}, \sigma, \delta)$ yields another database $\mathcal{D}'$ which has also the structure $\langle \Delta_{\mathcal{D}'}, \preccurlyeq \rangle$, given that δ is a labelled formula of the form $\gamma : \alpha$, following the definition of CIU_{LDS} declarative units.

Proof:

We have to prove the statement of the proposition for both cases, when $\sigma = \mathrm{U}_+$ and $\sigma = \mathrm{U}_-$.

[(1) $\sigma = \mathrm{U}_+$] In this case, $Up(\mathcal{D}, \mathrm{U}_+, \gamma : \alpha) = \mathcal{D} \biguplus \gamma : \alpha$. By Definition 2.38, we have that $\mathcal{D} \biguplus \gamma : \alpha = \mathcal{D}'$, where $\mathcal{D}' = \langle (\Delta_{\mathcal{D}} - X_{\mathcal{D}}) \cup \{\gamma : \alpha\}, \preccurlyeq \rangle$, if c_1 is satisfied, and $\mathcal{D}' = \mathcal{D} \odot \gamma : \alpha$, otherwise. Where $c_1 \Rightarrow \Delta_{\mathcal{D}}, \gamma : \alpha \nvdash \gamma' : \bot$, for any label $\gamma' \in \mathcal{L}_\gamma$, and $X_{\mathcal{D}} \subseteq \Delta_{\mathcal{D}}$, such that $X_{\mathcal{D}} = \{\gamma_l : \alpha \mid \gamma_l : \alpha \in \Delta_{\mathcal{D}}, \text{ and } \varphi_{\mathcal{U}}(\gamma, \gamma_l) \in \mathcal{A}\}$. So, in the case that c_1 is satisfied, the resulting database is already guaranteed to be of the same structure as $\mathcal{D}$. Also, by Definition 2.38, all the elements of $\mathcal{D}'$ are well defined declarative units of CIU_{LDS}. When c_1 is not satisfied, $\mathcal{D}'$ is the resulting database from the revision of $\mathcal{D}$ by $\gamma : \alpha$, $\mathcal{D} \odot \gamma : \alpha$. By Proposition 2.21, we guarantee that $\mathcal{D} \odot \gamma : \alpha$ preserves the structural properties of database $\mathcal{D}$.

[(2) $\sigma = \mathrm{U}_-$] In this case, $Up(\mathcal{D}, \mathrm{U}_-, \gamma : \alpha) = \mathcal{D} \Xi \gamma : \alpha$. By Definition 2.46, we have that $\mathcal{D} \Xi \gamma : \alpha = \mathcal{D}'$, where $\mathcal{D}' = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, if $\gamma : \alpha \notin \Delta_{\mathcal{D}}$, and $\mathcal{D}' = \langle (\Delta_{\mathcal{D}} \bigcup Smax(CCon(\gamma : \alpha))_{c_2}) - \{\gamma : \alpha\}, \preccurlyeq \rangle$, otherwise. According to Definition 2.45, $Smax(CCon(\gamma : \alpha))_{c_2}$ contains non-supported declarative units, and is ordered by $\preccurlyeq$. So, Definition 2.46, shows that the resulting database of $\mathcal{D} \Xi \gamma : \alpha$ also preserves the structure of the original database $\mathcal{D}$.

Hence, by cases (1) and (2), we have that the single update function $Up(\mathcal{D}, \sigma, \gamma : \alpha)$ always generates a new database $\mathcal{D}'$, which has the same structural properties of the original database $\mathcal{D}$.

■

Consistency

To show that the update function $Up(\mathcal{D}, \sigma, \delta)$ is consistent, we need to prove that for any initial consistent database $\mathcal{D}$, and any declarative unit $\gamma : \alpha$, if we update $\mathcal{D}$ with $\gamma : \alpha$, the resulting database $\mathcal{D}'$ is always consistent.

Below, we state the consistency theorem for the update function $Up(\mathcal{D}, \sigma, \delta)$.

Theorem 2.2 (Consistency of $Up(\mathcal{D}, \sigma, \delta)$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, for any declarative unit $\gamma : \alpha$, and for any update type σ, where $\sigma = \{\mathrm{U}_+, \mathrm{U}_-\}$, $Up(\mathcal{D}, \sigma, \gamma : \alpha) = \mathcal{D}'$, and $\mathcal{D}' \nvdash \gamma' : \bot$, for any label γ' of $\mathcal{L}_\gamma$.

Proof:

(1) In the case that $\sigma = \mathrm{U}_+$, $Up(\mathcal{D}, \mathrm{U}_+, \gamma : \alpha) = \mathcal{D} \uplus \gamma : \alpha$. By Definition 2.38, we have that $\mathcal{D} \uplus \gamma : \alpha = \mathcal{D}'$, where $\mathcal{D}' = \langle(\Delta_\mathcal{D} - X_\mathcal{D}) \cup \{\gamma : \alpha\}, \preccurlyeq\rangle$, if c_1 is satisfied, where $c_1 = \Delta_\mathcal{D}, \gamma : \alpha \nvdash \gamma' : \bot$ for any label $\gamma' \in \mathcal{L}_\gamma$, and $X_\mathcal{D} \subseteq \Delta_\mathcal{D}$, such that $X_\mathcal{D} = \{\gamma_l : \alpha \mid \gamma_l : \alpha \in \Delta_\mathcal{D}$, and $\varphi_\Downarrow(\gamma, \gamma_l) \in \mathcal{A}\}$. So, in this case, by satisfying c_1 we already guarantee that $\mathcal{D}' \nvdash \gamma' : \bot$. In the case that c_1 is not satisfied, $\mathcal{D}'$ is given by the revision of $\mathcal{D}$ by $\gamma : \alpha$. $\mathcal{D}' = \mathcal{D} \odot \gamma : \alpha$. By Theorem 2.1, we guarantee that $\mathcal{D} \odot \gamma : \alpha \nvdash \gamma' : \bot$.

(2) In the case that $\sigma = \mathrm{U}_-$, $Up(\mathcal{D}, \mathrm{U}_-, \gamma : \alpha) = \mathcal{D} \Xi \gamma : \alpha$. By Definition 2.46, we have that $\mathcal{D} \Xi \gamma : \alpha = \mathcal{D}'$, where $\mathcal{D}' = \langle\Delta_\mathcal{D}, \preccurlyeq\rangle$, if $\gamma : \alpha \notin \Delta_\mathcal{D}$, in which case it is guaranteed that $\mathcal{D}' \nvdash \gamma' : \bot$ since $\mathcal{D}' = \mathcal{D}$ and $\mathcal{D}$ is assumed to be consistent. In the case that $\gamma : \alpha \in \Delta_\mathcal{D}$, $\mathcal{D}' = \langle(\Delta_\mathcal{D} \bigcup Smax(CCon(\gamma : \alpha))_{c_2}) - \{\gamma : \alpha\}, \preccurlyeq\rangle$. According to Remark 2.14, $Smax(CCon(\gamma : \alpha))_{c_2}$ is a subset of $CCon(\gamma : \alpha)$ (Definition 2.45), which is a re-labelled version of $Con(\gamma : \alpha)$. By Definition 2.44, $Con(\gamma : \alpha) = \{\gamma_i : \alpha_i \mid \Delta_\mathcal{D} \vdash \gamma_i : \alpha_i$, and $(\Delta_\mathcal{D} - \{\gamma : \alpha\}) \nvdash \gamma_i : \alpha_i\}$. So, originally the declarative units of $Con(\gamma : \alpha)$ were already derivable from $\mathcal{D}$. Since $\mathcal{D}$ is assumed to be consistent, we can also conclude that in this case $\mathcal{D}' \nvdash \gamma' : \bot$.

Hence, by cases (1) and (2), we show that the single update function $Up(\mathcal{D}, \sigma, \gamma : \alpha)$ always generates a new database $\mathcal{D}'$, which is consistent.

■

2.7.4 The Basic Update Operations $\uplus$ and Ξ

Conditional Addition Operation $\uplus$

The CIU_{LDS} notion of conditional addition of data in a database, does not satisfy associativity. The reason for this is mainly due to the fact that the $\uplus$ operation may invoke revision, and then we cannot guarantee associativity in compromised revision.

■ **$\uplus$ is Non-associative**

Given the $\uplus$ operation of conditional inclusion of a declarative unit into a CIU_{LDS} database, we say that $\uplus$ satisfies associativity iff given a database $\mathcal{D}$, and the declarative units $\gamma_1 : \alpha_1$ and $\gamma_2 : \alpha_2$, the following holds:

$$(\mathcal{D} \uplus \gamma_1 : \alpha_1) \uplus \gamma_2 : \alpha_2 = (\mathcal{D} \uplus \gamma_2 : \alpha_2) \uplus \gamma_1 : \alpha_1$$

However this property does not follow in our system, since the conditional addition of a declarative unit into a database $\mathcal{D}$, may invalidate previously derivable data, and may also cause that some existing data be contracted from the database, via the system's compromised revision.

We can show that associativity does not hold for the operation $\uplus$, via a counter-example.

Example 2.8 (Counter-Example for the Associativity of $\uplus$)

Given a consistent database $\mathcal{D} = \langle\Delta_\mathcal{D}, \preccurlyeq\rangle$, *where* $\Delta_\mathcal{D} = \{\mathrm{E}_1 : A, \mathrm{P}_1 : A \wedge B \rightarrow \bot, \mathrm{I}_1 : A \rightarrow D, \mathrm{I}_2 : B \rightarrow C\}$; *and the declarative units* $\gamma_1 : \alpha_1 = \mathrm{E}_2 : B$, *and* $\gamma_2 : \alpha_2 = \mathrm{E}_3 : A$.

If we add $\gamma_1 : \alpha_1$ *to* $\mathcal{D}$ *and then we add* $\gamma_2 : \alpha_2$ *to the result of the first conditional addition, by Definition 2.38, we have that* $\mathcal{D} \uplus \gamma_1 : \alpha_1 = \langle\Delta'_\mathcal{D}, \preccurlyeq\rangle$, *where* $\Delta'_\mathcal{D} = \{\mathrm{E}_2 : B, \mathrm{P}_1 : A \wedge B \rightarrow \bot, \mathrm{I}_1 : A \rightarrow D, \mathrm{I}_2 : B \rightarrow C, \mathrm{N}_1 : D\}$. *And* $(\mathcal{D} \uplus \gamma_1 : \alpha_1) \uplus \gamma_2 : \alpha_2 = \langle\Delta''_\mathcal{D}, \preccurlyeq\rangle$, *where* $\Delta''_\mathcal{D} = \{\mathrm{E}_3 : A, \mathrm{P}_1 : A \wedge B \rightarrow \bot, \mathrm{I}_1 : A \rightarrow D, \mathrm{I}_2 : B \rightarrow C, \mathrm{N}_2 : C\}$.

If we now do $(\mathcal{D} \uplus \gamma_2 : \alpha_2) \uplus \gamma_1 : \alpha_1$, *we have that* $\mathcal{D} \uplus \gamma_2 : \alpha_2 = \langle \Delta^{*}_{\mathcal{D}}, \preccurlyeq \rangle$, *where* $\Delta^{*}_{\mathcal{D}} = \{E_3 : A,\ P_1 : A \wedge B \rightarrow \bot,\ I_1 : A \rightarrow D,\ I_2 : B \rightarrow C\}$. *And* $(\mathcal{D} \uplus \gamma_2 : \alpha_2) \uplus \gamma_1 : \alpha_1 = \langle \Delta^{**}_{\mathcal{D}}, \preccurlyeq \rangle$, *where* $\Delta^{**}_{\mathcal{D}} = \{E_2 : B,\ P_1 : A \wedge B \rightarrow \bot,\ I_1 : A \rightarrow D,\ I_2 : B \rightarrow C, N_1 : D\}$.

Then, we have that $\Delta^{''}_{\mathcal{D}} \neq \Delta^{**}_{\mathcal{D}}$. *Hence, this shows that the associativity property does not hold for the operation* $\uplus$, *as expected.*

•

Compromised Contraction Operation Ξ

The CIU_{LDS} compromised contraction of data from a database, also does not satisfy associativity.

Given the Ξ operation of compromised contraction of a declarative unit from a CIU_{LDS} database, for any declarative units $\gamma_1 : \alpha_1$ and $\gamma_2 : \alpha_2$, it is not always the case that $(\mathcal{D} \,\Xi\, \gamma_1 : \alpha_1) \,\Xi\, \gamma_2 : \alpha_2 = (\mathcal{D} \,\Xi\, \gamma_2 : \alpha_2) \,\Xi\, \gamma_1 : \alpha_1$.

By Definition 2.46, we have that $\mathcal{D} \,\Xi\, \gamma : \alpha = \mathcal{D}'$, where $\mathcal{D}' = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, if $\gamma : \alpha \notin \Delta_{\mathcal{D}}$, and $\mathcal{D}' = \langle (\Delta_{\mathcal{D}} \bigcup Smax(CCon(\gamma : \alpha))_{c_2}) - \{\gamma : \alpha\}, \preccurlyeq \rangle$, otherwise. Associativity for Ξ only holds in the case that either $\gamma_1 : \alpha_1$ or $\gamma_2 : \alpha_2$ is not in $\Delta_{\mathcal{D}}$, and $\gamma_1 : \alpha_1 \notin Smax(CCon(\gamma_2 : \alpha_2))_{c_2}$ and $\gamma_2 : \alpha_2 \notin Smax(CCon(\gamma_1 : \alpha_1))_{c_2}$. If we consider, for instance, that $\gamma_2 : \alpha_2 \notin \Delta_{\mathcal{D}}$, and $\gamma_2 : \alpha_2 \in Smax(CCon(\gamma_1 : \alpha_1))_{c_2}$, then $(\mathcal{D} \,\Xi\, \gamma_1 : \alpha_1) \,\Xi\, \gamma_2 : \alpha_2 = \langle (\Delta_{\mathcal{D}\Xi\gamma_1:\alpha_1} \bigcup Smax(CCon(\gamma_2 : \alpha_2))_{c_2}) - \{\gamma_2 : \alpha_2\}, \preccurlyeq \rangle$, which is different from $\mathcal{D} \,\Xi\, \gamma_1 : \alpha_1$.

2.7.5 Properties of the CIU_{LDS} Transactions

Database Structural Properties Preservation

Since a transaction is defined as a sequence of updates, and we have already proved that the update function preserves the structural properties of a given database $\mathcal{D}$, then it is trivially guaranteed that a CIU_{LDS} transaction $Trans(\mathcal{D}, UP)$, returns a resulting database $\mathcal{D}'$, which presents the same structural organization of $\mathcal{D}$. Proposition 2.24 below caters for this transaction property.

Proposition 2.24

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *for any sequence of updates* UP, *where* $UP = Up_1, \cdots, Up_n$ *and for each* $1 \leq i \leq n$ $Up_i(\mathcal{D}_i, \sigma_i, \delta_i) = \mathcal{D}'_i$ *such that* $\mathcal{D}_i = \mathcal{D}'_{i-1}$, *a transaction* $Trans(\mathcal{D}, UP)$ *yields another database* $\mathcal{D}'$ *which has also the structure* $\langle \Delta_{\mathcal{D}'}, \preccurlyeq \rangle$, *of the original database* $\mathcal{D}$.

Proof:

By Definition 2.51, the base case of our proof is when $n = 1$. In this case, $UP = Up_1$, $Trans(\mathcal{D}, UP) = Up_1(\mathcal{D}_1, \sigma_1, \delta_1) = \mathcal{D}'_1$, where $\mathcal{D}_1 = \mathcal{D}$ and $\mathcal{D}'_1 = \mathcal{D}'$. By Proposition 2.23, $\mathcal{D}'$ preserves the structural properties of $\mathcal{D}$. In the case that $n > 1$, for each $1 \leq i \leq n$ in $Up_i(\mathcal{D}_i, \sigma_i, \delta_i) = \mathcal{D}'_i$, $\mathcal{D}_i = \mathcal{D}'_{i-1}$. Again, by Proposition 2.23, $\mathcal{D}'_i$ preserves the structural properties of $\mathcal{D}$. By applying the n updates in $Trans(\mathcal{D}, UP)$, we have that $\mathcal{D}'_n$ also preserves the structural properties of $\mathcal{D}$. Hence, we have shown by induction on the number of updates in the transaction, that $Trans(\mathcal{D}, UP)$ always generates a new database $\mathcal{D}'$, which has the same structural properties of the original database $\mathcal{D}$.

■

Consistency

It is also trivial to show that a transaction $Trans(\mathcal{D}, UP)$ is consistent, since we have already shown that each update operation, in its sequence of updates, returns a resulting database $\mathcal{D}'$ which is always consistent.

Below, we state the consistency theorem for the transaction function $Trans(\mathcal{D}, UP)$.

Theorem 2.3 (Consistency of $Trans(\mathcal{D}, UP)$)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, for any sequence of updates UP, where $UP := Up_1, \cdots, Up_n$ and for each $1 \leq i \leq n$ $Up_i(\mathcal{D}_i, \sigma_i, \delta_i) = \mathcal{D}'_i$ such that $\mathcal{D}_i = \mathcal{D}'_{i-1}$, a transaction $Trans(\mathcal{D}, UP)$ yields another database $\mathcal{D}'$ such that $\mathcal{D}' \nvdash \gamma' : \bot$, for any label γ' of $\mathcal{L}_\gamma$.

Proof:

By Definition 2.51, the base case of our proof is when $n = 1$. In this case, $UP = Up_1$, $Trans(\mathcal{D}, UP) = Up_1(\mathcal{D}_1, \sigma_1, \delta_1) = \mathcal{D}'_1$, where $\mathcal{D}_1 = \mathcal{D}$ and $\mathcal{D}'_1 = \mathcal{D}'$. By Proposition ??, we have proved that $\mathcal{D}' \nvdash \gamma' : \bot$, for any label γ' of $\mathcal{L}_\gamma$. In the case that $n > 1$, for each $1 \leq i \leq n$ in $Up_i(\mathcal{D}_i, \sigma_i, \delta_i) = \mathcal{D}'_i$, $\mathcal{D}_i = \mathcal{D}'_{i-1}$. Again, by Proposition ??, $\mathcal{D}'_i \nvdash \gamma' : \bot$, for any label γ' of $\mathcal{L}_\gamma$. By applying the n updates in $Trans(\mathcal{D}, UP)$, we have that $\mathcal{D}'_n$ is also a consistent database. Hence, we have shown by induction on the number of updates in the transaction, that $Trans(\mathcal{D}, UP)$ always generates a new database $\mathcal{D}'$, which is consistent. That is, $Trans(\mathcal{D}, UP) = \mathcal{D}'$, such that $\mathcal{D}' \nvdash \gamma' : \bot$, for any label γ' of $\mathcal{L}_\gamma$.

■

2.7.6 Discussions

In this section, we have shown some important properties of the system CIU_{LDS} w.r.t. the consequence relation, the revision function, the updates and the transactions. In CIU_{LDS}, the consequence relation satisfies particular versions of the properties of restricted reflexivity, restricted monotonicity, the deduction theorem, strong transitivity (unitary cut), and non-explosiveness. Some of those properties follow also for the case that the conditional inclusion operation of the system is considered, in the place of the conventional set-theoretical inclusion. The updates and the transactions defined for CIU_{LDS}, are shown to preserve the structural properties of the given initial labelled database, and to be consistent. The revision operator $\odot$ is defined based on the revision operator ® of [Dar-96c]. Hence, $\odot$ also preserves the structural properties of the input database; is consistent; and satisfies compromised persistence.

2.8 Summary and General Remarks

In this section, we have introduced CIU_{LDS} - a labelled realization of our approach of *Compromising Interfering Updates*, based on the logical framework of Labelled Deductive Systems. In CIU_{LDS}, the labelling conditions applied to the inference rules and to the update operations, control both the derivation and the reconciliation processes of the system. Hence, it guides the derivation mechanism specifically to the requirements of our compromised approach. The labelling functions represented by $©_X$, map the possible labels of the inputs for the inference rules and update functions, to their corresponding output label within the resulting declarative unit. The ordering $\preccurlyeq$ on the set of declarative units is supposed to contribute to the expressive power of the labelled database, and to serve as basis for the construction of the *safe-maximal* consistent subsets which are introduced in [Dar-96c], to provide a unique solution from

the revision process. The inference rules for the logical connectives were based on the natural deduction presentation style for propositional logic, with the main difference that in CIU_{LDS} we use labelled formulae and we define some conditions for applying the rules. In CIU_{LDS}, the update requests involve addition and retraction of declarative units to and from a labelled database $\mathcal{D}$. We have restricted the declarative units in the single updates, to be either extensional data or intensional data which are present in $\mathcal{D}$. These restrictions avoid that the updates handle protected data, as usually expected, and also non-supported data. The basic update operations which are invoked by the single updates, carry the reconciling flavour of our compromised approach to conflicting inputs. The operation of conditional inclusion $\biguplus$, invokes the compromised revision function $\odot$, when a compromised solution for the update applies. The compromised retraction operation Ξ, already embeds in its definition the mechanism for allowing consequences of retracted declarative units to be added to the database as non-supported data. This operation uses the notion of safe-maximality, when a choice is needed among the compromised consequences. We have introduced the revision $\odot$ in CIU_{LDS}, based on the compromised revision ® for finite bases with integrity constraints defined in [Dar-96c]. So, $\odot$ is basically devoted to free the database from inconsistency by allowing some consequences from conflicting updates, or from retracted declarative units, to be kept in the resulting database as a compromise. We have adopted the notion of safe-maximality introduced in [Dar-96c], as an impartial solution to choose among the minimal declarative units of the labelled database, considering the ordering $\preccurlyeq$. Transactions were defined in CIU_{LDS} as a sequence of updates. Hence, it incorporates the results of various single updates in sequence. Within each update request of a transaction, the compromised revision $\odot$ may apply. Since, the resulting updated databases are consistent, then we also have that the result of a transaction is a consistent labelled database.

CIU_{LDS} handles the non-supported consequences which are generated from the compromised revision on the database, without applying specific restrictions to them. However, a more application-oriented implementation of this approach could better explore the handling of non-supported consequences for the application own needs. Within a hypothetical reasoning context, for instance, the non-supported consequences of rejected inputs could represent the consistent consequences of a particular input, say a game move, which would have been supported, had the input not conflicted with the existing constraints. In this case, the labelling control of the non-supported data would have to distinguish among the consequences of different rejected inputs. And the derivation mechanism would also have to take the conditional aspect of those consequences into account.

As a further work for the CIU_{LDS} formalization, we intend to propose a semantics for the system, which grasps the intuitive notion of the reconciliation among conflicting data. As a first thought, we could try to define a semantics for CIU_{LDS}, which would characterize on a state basis the changes of a labelled database $\mathcal{D}$. Each state could be defined as a snapshot of the database, and each step from one state to another would be given by an update operation $\mathcal{D}$. So, we would have a sequence of states representing the databases, initially and after the update operations, up to the current database. In such a semantics, the following concepts would be of relevance: a sequence of database states $\mathcal{S}_{\mathcal{D}}$; an interpretation for the language $\mathcal{L}_{CIU}$, considering the labelling conditions; the notion of a model; the notion of satisfiability of a declarative unit; the notion of satisfiability of a labelled database; and the notion of semantic entailment. Alternatively, we could also think of using the possible worlds semantics to characterize the models of a given labelled database, where the accessibility relation would represent a move from one database to another, via an update. There is still a lot to be investigated in order to define a semantics for CIU_{LDS}. It is our contention to put effort in this direction, so that we can provide a complete updating logical system for our approach.

3 Final Remarks

In this paper, we have introduced our approach of *Compromised Interfering Updates*, and we have presented a a logical system, CIU_{LDS}, as a labelled realization to the approach, under the Labelled Deductive Systems framework (LDS). Some relevant properties were presented, establishing the main results of the system.[46]

[46] Full proofs of the properties cited in this paper are available in [Dar-96b,96d].

Under the compromised philosophy of the approach presented here, some other pieces of work were developed by the author. In [Dar-96a], the problem of dealing with inconsistency after the performance of a database transaction is addressed, within the context of deductive databases. In this formalization, CIU is defined on the basis of the integrity-checking method for deductive databases described in [SaKo-87]. [Dar-96c,96d] presents more details about the idea of compromised reasoning and the compromised revision under a belief revision perspective. A planned further work in this research line, is the investigation of compromised solutions for modelling simultaneous occurrence of actions, where we have to tackle problems which arise when reasoning about possible conflicts and combined effects of these actions. We believe that this area can benefit much from compromising on solutions. We have studied the existing approaches related to these aspects in the context of actions, but so far not much has been developed on it. [Dar-96d] presents a brief comparison with Truth Maintenance Systems. A more detailed study on their differences, advantages and limitations, is also a planned future work.

Acknowledgements

This work was financially supported by the Brazilian Research Council, CNPq, under the grant 202078/90.6. The author is very grateful to Dov Gabbay for his support as a research supervisor, and to Hans Jürgen Ohlbach, for his comments and suggestions on an earlier version of this work.

A Auxiliary Proofs

This appendix presents proofs of some propositions cited in section 2 of this paper.

Proposition A.1 (Proposition 2.1)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and a declarative unit* $\gamma : \alpha$, *such that* $(\Delta_{\mathcal{D}} - I) \bigcup \{\gamma : \alpha\} \vdash \gamma' : \bot$, *for some label* $\gamma' \in \mathcal{L}_\gamma$, *and it is not the case that* $\neg\alpha$ *is a tautology. Then, given the set* $Fail(\Delta_{\mathcal{D}} - I)_{c_3} = \{S_1, S_2, \cdots, S_j\}$, *it is sufficient to retract one element from each* S_i, *such that* $((\Delta_{\mathcal{D}} - I) - \mathcal{S}(Fail(\Delta_{\mathcal{D}} - I)_{c_3})) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, *for any label* $\gamma' \in \mathcal{L}_\gamma$, *where* $\mathcal{S}(Fail(\Delta_{\mathcal{D}} - I)_{c_3}) = \{\gamma_i : \alpha_i \mid \forall S_i \in Fail(\Delta_{\mathcal{D}} - I)_{c_3}, \exists \gamma^* : \alpha^* \in S_i$, *such that* $\gamma_i, \gamma^* \in \mathtt{Z}$, *and* $\alpha_i = \alpha^*\}$, *where* $\mathtt{Z}$ *is a label type in* $\mathcal{L}_\gamma$.

Proof:

By Remark 2.15, each set $S_i \in Fail(\Delta_{\mathcal{D}} - I)_{c_3}$ is minimal w.r.t. $\subseteq$, such that $S_i \bigcup \{\gamma : \alpha\} \vdash \gamma' : \bot$, for $\gamma' \in \mathcal{L}_\gamma$. Then, for any declarative unit $\gamma^* : \alpha^* \in S_i$, $(S_i - \{\gamma^* : \alpha^*\}) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, for $\gamma' \in \mathcal{L}_\gamma$. This is true for all $S_i \in Fail(\Delta_{\mathcal{D}} - I)_{c_3}$. $\mathcal{S}(Fail(\Delta_{\mathcal{D}} - I)_{c_3})$ is a set that contains one element of each $S_i \in Fail(\Delta_{\mathcal{D}} - I)_{c_3}$, for $i = 1, \cdots, l$. Therefore, it is guaranteed that $((\Delta_{\mathcal{D}} - I) - \mathcal{S}(Fail(\Delta_{\mathcal{D}} - I)_{c_3})) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, for any label $\gamma' \in \mathcal{L}_\gamma$.

■

Proposition A.2 (Proposition 2.2)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *a declarative unit* $\gamma : \alpha$, *such that* $(\Delta_{\mathcal{D}} - I) \bigcup \{\gamma : \alpha\} \vdash \gamma' : \bot$, *for some* $\gamma' \in \mathcal{L}_\gamma$, *and it is not the case that* $\neg\alpha$ *is a tautology,* $Smax(\Delta_{\mathcal{D}} - I)_{c_3}$, $\gamma : \alpha \nvdash \gamma' : \bot$, *for any* $\gamma' \in \mathcal{L}_\gamma$.

Proof:

By Remark 2.17, in the case that $Smax(\Delta_{\mathcal{D}} - I)_{c_3} = (\Delta_{\mathcal{D}} - I)$, the statement of the proposition is already guaranteed by condition c_3. In the case that $Smax(\Delta_{\mathcal{D}} - I)_{c_3} = (\Delta_{\mathcal{D}} - I) - Min(Fail(\Delta_{\mathcal{D}} - I)_{c_3})$, we have proved in Proposition 2.1, that $((\Delta_{\mathcal{D}} - I) - \mathcal{S}(Fail(\Delta_{\mathcal{D}} - I)_{c_3})) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, for any label $\gamma' \in \mathcal{L}_\gamma$, where $\mathcal{S}(Fail(\Delta_{\mathcal{D}} - I)_{c_3})$ is any set which includes one element of each set $S_i \in Fail(\Delta_{\mathcal{D}} - I)_{c_3}$. By Definition 2.42, $RMin(Fail(\Delta_{\mathcal{D}} - I)_{c_3})$ contains at least one minimal element w.r.t. $\preccurlyeq$, of each set S_i of $Fail(\Delta_{\mathcal{D}} - I)_{c_3}$. Hence, there exists a set $\mathcal{S}(Fail(\Delta_{\mathcal{D}} - I)_{c_3})$, such that $\mathcal{S}(Fail(\Delta_{\mathcal{D}} - I)_{c_3}) \subseteq RMin(Fail(\Delta_K)_{c_1})$. Then, abstracting from Proposition 2.1, we have that $((\Delta_{\mathcal{D}} - I) - RMin(Fail(\Delta_{\mathcal{D}} - I)_{c_3})) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$. Therefore, also when $Smax(\Delta_{\mathcal{D}} - I)_{c_3} = ((\Delta_{\mathcal{D}} - I) - Min(Fail(\Delta_{\mathcal{D}} - I)_{c_3}))$, it is guaranteed that $Smax(\Delta_{\mathcal{D}} - I)_{c_3} \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, for any label $\gamma' \in \mathcal{L}_\gamma$.

■

Proposition A.3 (Proposition 2.3)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *a declarative unit* $\gamma : \alpha$, *such that* $CI(\gamma : \alpha)$ *is non-empty,* $\Delta_{\mathcal{D}}, \gamma : \alpha \vdash \gamma' : \bot$ *for some label* $\gamma' \in \mathcal{L}_\gamma$, *and it is not the case that* α *is a tautology,* $Smax(CI(\gamma : \alpha))_{c_4} \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \bot$, *for any label* $\gamma' \in \mathcal{L}_\gamma$.

Proof:

By Remark 2.19, in the case that $Smax(CI(\gamma : \alpha))_{c_4} = CI(\gamma : \alpha)$, the statement of the proposition is already guaranteed by condition c_4. In the case that $Smax(CI(\gamma : \alpha))_{c_4} = CI(\gamma : \alpha) - RMin(Fail(CI(\gamma : \alpha))_{c_4})$, we have that by Definition 2.39, each set $S_i \in Fail(CI(\gamma : \alpha))_{c_4}$ is minimal w.r.t. $\subseteq$, such that $S_i \bigcup \Delta_{\mathcal{D}} \vdash \gamma' : \bot$, for some $\gamma' \in \mathcal{L}_\gamma$. Then, for any element $\gamma^* : \alpha^* \in S_i$, $(S_i - \{\gamma^* : \alpha^*\}) \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \bot$. This is true for all $S_i \in Fail(CI(\gamma : \alpha))_{c_4}$. By Definition 2.42, $RMin(Fail(CI(\gamma : \alpha))_{c_4})$ contains at least one minimal element w.r.t. $\preccurlyeq$, of each set S_i of $Fail(CI(\gamma : \alpha))_{c_4}$. Therefore, we have that $(CI(\gamma : \alpha) - RMin(Fail(CI(\gamma : \alpha))_{c_4})) \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \bot$, Hence, also when $Smax(CI(\gamma : \alpha))_{c_4} = CI(\gamma : \alpha) - RMin(Fail(CI(\gamma : \alpha))_{c_4})$, it is guaranteed that $Smax(CI(\gamma : \alpha))_{c_4} \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \bot$, for any $\gamma' \in \mathcal{L}_\gamma$.

■

Proposition A.4 (Proposition 2.4)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *a declarative unit* $\gamma : \alpha$, *such that* $CI(\gamma : \alpha)$ *is non-empty, and it is not the case that* α *is a tautology,* $Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5} \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \alpha$, *for any* $\gamma' \in \mathcal{L}_\gamma$.

Proof:

By Remark 2.21, in the case that $Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5} = Smax(CI(\gamma : \alpha))_{c_4}$, the statement of the proposition is already guaranteed by condition c_5. In the case that $Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5} = Smax(CI(\gamma : \alpha))_{c_4} - RMin(Fail(Smax(CI(\gamma : \alpha))_{c_4})_{c_5})$, we have by Definition 2.39, that each set $S_i \in Fail(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$ is minimal w.r.t. $\subseteq$, such that $S_i \bigcup \Delta_{\mathcal{D}} \vdash \gamma' : \alpha$, for some $\gamma' \in \mathcal{L}_\gamma$. Then, for any element $\gamma^* : \alpha^* \in S_i$, $(S_i - \{\gamma^* : \alpha^*\}) \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \alpha$. This is true for all $S_i \in Fail(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$. By Definition 2.42, $RMin(Fail(Smax(CI(\gamma : \alpha))_{c_4})_{c_5})$ contains at least one minimal element w.r.t. $\preccurlyeq$, of each set S_i of $Fail(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$. Therefore, we have that $Smax(CI(\gamma : \alpha))_{c_4} - RMin(Fail(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}) \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \alpha$. Hence, also when $Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5} = Smax(CI(\gamma : \alpha))_{c_4} - RMin(Fail(Smax(CI(\gamma : \alpha))_{c_4})_{c_5})$, it is guaranteed that $Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5} \bigcup \Delta_{\mathcal{D}} \nvdash \gamma' : \alpha$, for any $\gamma' \in \mathcal{L}_\gamma$.

■

Proposition A.5 (Proposition 2.5)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *for any set* $R_{\gamma:\alpha} \subseteq \Delta_{\mathcal{D}-I}$, *we have that* $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} \nvdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R_{\gamma:\alpha}$.

Proof:

By Remark 2.23, in the case that $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} = (\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})$, the statement of the proposition is already guaranteed by condition c_7. In the case that $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} = (\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}) - RMin(Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})$, we have that by Definition 2.39, each set $S_i \in Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$ is minimal w.r.t. $\subseteq$, satisfying the condition $(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}) \vdash \gamma^* : \alpha^*$, for any $\gamma^* : \alpha^* \in R_{\gamma:\alpha}$. Then, for any element $\gamma' : \alpha' \in S_i$, $(S_i - \{\gamma' : \alpha'\}) \nvdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R_{\gamma:\alpha}$. This holds for all $S_i \in Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$. By Definition 2.40, the set $min(Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})$ contains at least one element of each $S_i \in Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$, *for* i = 1, $\cdots, k$. And by Definition ??, $RMin(Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})$ contains at least one element of each set $min(S_i)$ of $min(Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})$. Hence, also when $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} = (\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}) - RMin(Fail(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})$, it is guaranteed that $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} \nvdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R_{\gamma:\alpha}$.

■

Proposition A.6 (Proposition 2.6)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and a declarative unit* $\gamma : \alpha$, *such that* $R_{\gamma:\alpha}$ *is non-empty,* $Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8} \bigcup \{\gamma : \alpha\} \bigcup I \nvdash \gamma^* : \bot$, *for any* $\gamma^* \in \mathcal{L}_\gamma$.

Proof:

By Remark 2.25, when $Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8} = Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7}$, the statement of the proposition is already guaranteed by condition c_8. In the case that $Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8} = Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} - RMin(Fail(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8})$, we have that by Definition 2.39, each set $S_i \in Fail(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$ is minimal w.r.t. $\subseteq$, satisfying the condition $Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} \vdash \gamma^* : \bot$. Then, for any element $\gamma' : \alpha' \in S_i$, $(S_i - \{\gamma' : \alpha'\}) \nvdash \gamma^* : \bot$. This holds for all $S_i \in Fail(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$. By Definition 2.40, the set $min(Fail(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8})$ contains at least one element of each $S_i \in Fail(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$, *for* i = 1, $\cdots, k$. And by Definition ??, $RMin(Fail(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8})$ contains at least one element of each set $min(S_i)$ of $min(Fail(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8})$. Hence, also when $Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8} = Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7} - RMin(Fail(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8})$, it is guaranteed that $Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8} \bigcup \{\gamma : \alpha\} \bigcup I \nvdash \gamma^* : \bot$, for any $\gamma^* \in \mathcal{L}_\gamma$.

■

Proposition A.7 (Proposition 2.7)

Given a knowledge base $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *for any set* $R^*_{\gamma:\alpha} \subseteq \Delta_{\mathcal{D}}$, *such that* $CR(R^*_{\gamma:\alpha})$ *is non-empty,* $Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \cup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$.

Proof:

By Remark 2.27, in the case that $Smax(CR(R^*_{\gamma:\alpha}))_{c_9} = CR(R^*_{\gamma:\alpha})$, the statement of the proposition is already guaranteed by condition c_9. In the case that $Smax(CR(R^*_{\gamma:\alpha}))_{c_9} = CR(R^*_{\gamma:\alpha}) - RMin(Fail(CR(R^*_{\gamma:\alpha}))_{c_9})$, we have that by Definition 2.39, each set $S_i \in Fail(CR(R^*_{\gamma:\alpha}))_{c_9}$ is minimal w.r.t. $\subseteq$, satisfying the condition $S_i \bigcup(\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \vdash \gamma^* : \alpha^*$, for a $\gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$. Then, for any element $\gamma' : \alpha' \in S_i$, $(S_i - \{\gamma' : \alpha'\}) \bigcup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*$. This holds for all $S_i \in Fail(CR(R^*_{\gamma:\alpha}))_{c_9}$. By Definition 2.40, the set $min(Fail(CR(R^*_{\gamma:\alpha}))_{c_9})$ contains at least one element of each $S_i \in Fail(CR(R^*_{\gamma:\alpha}))_{c_9}$, *for* i = 1, $\cdots, k$. And by Definition 2.42, $RMin(Fail(CR(R^*_{\gamma:\alpha}))_{c_9})$ contains at least one element of each set $min(S_i)$ of $min(Fail(CR(R^*_{\gamma:\alpha}))_{c_9})$. Hence, also when $Smax(CR(R^*_{\gamma:\alpha}))_{c_9} = CR(R^*_{\gamma:\alpha}) - Min(Fail(CR(R^*_{\gamma:\alpha}))_{c_9})$, it is guaranteed that $Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \cup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$.

■

Proposition A.8 (Proposition 2.8)

Given a database $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *and an input* $\gamma : \alpha$, *such that it is not the case that* $\neg\alpha$ *is a tautology. If* I, $\gamma : \alpha \nvdash \gamma' : \bot$ *and* $\Delta_{\mathcal{D}}, \gamma : \alpha \vdash \gamma' : \bot$, *for any set* $R_{\gamma:\alpha}$, *as in Definition 2.48, such that* $CR(R^*_{\gamma:\alpha})$ *is non-empty,* $Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \bigcup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, *for any label* $\gamma' \in \mathcal{L}_\gamma$.

Proof:

By definition, $R^*_{\gamma:\alpha} = \Delta_{\mathcal{D}-I} - Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$. Then, we can also say that $Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8} = \Delta_{\mathcal{D}-I} - R^*_{\gamma:\alpha}$. In Proposition 2.6, we have shown that $Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8} \bigcup \{\gamma : \alpha\} \bigcup I \nvdash \gamma' : \bot$, for any $\gamma' \in \mathcal{L}_\gamma$. That is, $(\Delta_{\mathcal{D}-I} - R^*_{\gamma:\alpha}) \bigcup \{\gamma : \alpha\} \bigcup I \nvdash \gamma' : \bot$. Since $\Delta_{\mathcal{D}-I}$ denotes $(\Delta_{\mathcal{D}} - I)$, $(\Delta_{\mathcal{D}-I} - R^*_{\gamma:\alpha}) \bigcup I$ can be substituted by $(\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha})$. So, by Proposition 2.6, we have that:

(1) $(\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$

By definition, $R^*_{\gamma:\alpha}$ is the set that has to be retracted from $\Delta_{\mathcal{D}}$, to accomplish the inclusion of $\gamma : \alpha$, keeping consistency. Since by Proposition 2.7, we have shown that $Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \cup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$, there is no chance that $\{\gamma : \alpha\}$ conflicts with $Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$. Then, it follows that:

(2) $\mathrm{Smax(CR(R^*_{\gamma:\alpha}))_{c_9}} \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$

Hence, by (1) and (2), it is guaranteed that:
$Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \bigcup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \bigcup \{\gamma : \alpha\} \nvdash \gamma' : \bot$, for any label $\gamma' \in \mathcal{L}_\gamma$.

■

Proposition A.9 (Proposition 2.9)

Given a base $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *for any input* $\gamma : \alpha$, *such that* $\neg\alpha$ *is not a tautology, and* $CI(\gamma : \alpha)$ *is non-empty, if* I, $\gamma : \alpha \vdash \gamma' : \bot$, *for any label* $\gamma' \in \mathcal{L}_\gamma$, *then* $\forall \gamma^* : \alpha^* \in Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$, $\mathcal{D} \nvdash \gamma^* : \alpha^*$ *and* $\mathcal{D} \odot \gamma : \alpha \vdash \gamma^* : \alpha^*$.

Proof:

By Definition 2.47, we guarantee that all the declarative units in $CI(\gamma : \alpha)$ are not originally present or derived from $\mathcal{D}$. By Definition 2.43 and Remark 2.21, we have that $Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5} \subseteq CI(\gamma : \alpha)$. And by Definition 2.50, we state that in the case that I, $\gamma : \alpha \vdash \gamma : \bot$, $\mathcal{D} \odot \gamma : \alpha = \langle \Delta_{\mathcal{D}} \bigcup Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}, \preccurlyeq \rangle$, where $Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$ is part of $\mathcal{D} \odot \gamma : \alpha$. Hence, $\forall \gamma^* : \alpha^* \in Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$, $\mathcal{D} \nvdash \gamma^* : \alpha^*$ and $\mathcal{D} \odot \gamma : \alpha \vdash \gamma^* : \alpha^*$.

■

Proposition A.10 (Proposition 2.10)

Given a base $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *for any input* $\gamma : \alpha$, *such that* $\neg\alpha$ *is not a tautology, if* $\Delta_{\mathcal{D}}, \gamma : \alpha \vdash \gamma' : \bot$, *for some label* $\gamma' \in \mathcal{L}_\gamma$, *and* $I, \gamma : \alpha \nvdash \gamma' : \bot$, *then* $\forall \gamma^* : \alpha^* \in Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$, $\mathcal{D} \vdash \gamma^* : \alpha^*$ *and also* $\mathcal{D} \odot \gamma : \alpha \vdash \gamma^* : \alpha^*$.

Proof:

By Definition 2.43 and Remark 2.27 $Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$ is a safe maximal subset of $CR(R^*_{\gamma:\alpha})$, relative to condition c_9. So, $\forall \gamma^* : \alpha^* \in Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$, $\mathcal{D} \vdash \gamma^* : \alpha^*$, since $CR(R^*_{\gamma:\alpha})$ is a set of consequences of $R^*_{\gamma:\alpha}$ and by definition, $R^*_{\gamma:\alpha} \subseteq \Delta_{\mathcal{D}}$. By Definition 2.50, in the case that $\Delta_{\mathcal{D}}, \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, and $I, \gamma : \alpha \nvdash \gamma' : \bot$, we state that $\mathcal{D} \odot \gamma : \alpha = \langle (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \bigcup Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \bigcup \{\gamma : \alpha\}, \preccurlyeq \rangle$. Hence, $\forall \gamma^* : \alpha^* \in Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$, $\mathcal{D} \odot \gamma : \alpha \vdash \gamma^* : \alpha^*$. Therefore, we guarantee that $\forall \gamma^* : \alpha^* \in Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$, $\mathcal{D} \vdash \gamma^* : \alpha^*$ and $\mathcal{D} \odot \gamma : \alpha \vdash \gamma^* : \alpha^*$.

■

Proposition A.11 (Proposition 2.11)

Given a base $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *for any input* $\gamma : \alpha$, *such that* $\neg\alpha$ *is not a tautology, if* $\Delta_{\mathcal{D}}, \gamma : \alpha \vdash \gamma' : \bot$, *for some label* $\gamma' \in \mathcal{L}_\gamma$, *and* $I, \gamma : \alpha \nvdash \gamma' : \bot$, *then* $\forall \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$, $\mathcal{D} \odot \gamma : \alpha \nvdash \gamma^* : \alpha^*$.

Proof:

By definition, $R^*_{\gamma:\alpha} = \Delta_{\mathcal{D}-I} - Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$ where $Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$ is a subset of $\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}$ and $R_{\gamma:\alpha} \nsubseteq \Delta_{\mathcal{D}-I} - R_{\gamma:\alpha}$. In the case that $\Delta_{\mathcal{D}}, \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, and $I, \gamma : \alpha \nvdash \gamma' : \bot$, we state that $\mathcal{D} \odot \gamma : \alpha = \langle (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \bigcup Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \bigcup \{\gamma : \alpha\}, \preccurlyeq \rangle$. By Definition 2.43 and Remark 2.23, $R_{\gamma:\alpha} \nsubseteq Smax(\Delta_{\mathcal{D}-I} - R^*_{\gamma:\alpha})_{c_7}$ and it is shown in Proposition 2.5 that $Smax(\Delta_{\mathcal{D}-I} - R^*_{\gamma:\alpha})_{c_7} \nvdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R_{\gamma:\alpha}$. By Proposition 2.7 has proved that $Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \cup (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \nvdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$. Since $R_{\gamma:\alpha}$ contains the elements that have to be retracted from $\Delta_{\mathcal{D}}$ in order to accomplish the insertion of $\gamma : \alpha$ and keep consistency, it is then also guaranteed that $\mathcal{D} \odot \gamma : \alpha \nvdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R_\alpha$. Hence, $\mathcal{D} \odot \gamma : \alpha \nvdash \gamma^* : \alpha^*$, $\forall \gamma^* : \alpha^* \in R^*_{\gamma:\alpha}$.

■

Proposition A.12 (Proposition 2.12)

Given a base $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \preccurlyeq \rangle$, *for any input* $\gamma : \alpha$, *such that* $\neg\alpha$ *is not a tautology, if a declarative unit* $\gamma^* : \alpha^* \in \mathcal{D} \odot \gamma : \alpha$, *then either* $\gamma^* : \alpha^* \in \Delta_{\mathcal{D}}$ *and* $\gamma^* : \alpha^* \notin R^*_{\gamma:\alpha}$; *or* $\gamma^* : \alpha^* \notin \Delta_{\mathcal{D}}$ *and either* $\gamma^* : \alpha^* \in Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$, *or* $\gamma^* : \alpha^* = \gamma : \alpha$, *or* $\gamma^* : \alpha^* \in Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$.

Proof:

By Definition 2.50, we state that $\mathcal{D} \odot \gamma : \alpha = \langle \Delta_{\mathcal{D} \odot \gamma:\alpha}, \preccurlyeq \rangle$, where in the case that $I, \gamma : \alpha \vdash \gamma' : \bot$, for some label $\gamma' \in \mathcal{L}_\gamma$, $\Delta_{\mathcal{D} \odot \gamma:\alpha} = \Delta_{\mathcal{D}} \bigcup Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$. This means that $\forall \gamma^* : \alpha^* \in \mathcal{D} \odot \gamma : \alpha$, either $\gamma^* : \alpha^* \in \Delta_{\mathcal{D}}$ or $\gamma^* : \alpha^* \in Smax(Smax(CI(\gamma : \alpha))_{c_4})_{c_5}$. And when $\Delta_{\mathcal{D}}, \gamma : \alpha \vdash \gamma' : \bot$ and $I, \gamma : \alpha \nvdash \gamma' : \bot$, $\Delta_{\mathcal{D} \odot \gamma:\alpha} = (\Delta_{\mathcal{D}} - R^*_{\gamma:\alpha}) \bigcup Smax(CR(R^*_{\gamma:\alpha}))_{c_9} \bigcup \{\gamma : \alpha\}$. By definition, $Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$ is equivalent to $\Delta_{\mathcal{D}-I} - R^*_{\gamma:\alpha}$, since $R^*_{\gamma:\alpha} = \Delta_{\mathcal{D}-I} - Smax(Smax(\Delta_{\mathcal{D}-I} - R_{\gamma:\alpha})_{c_7})_{c_8}$. Then, $\forall \beta \in K \circledR \alpha$, either $\gamma^* : \alpha^* \in \Delta_{\mathcal{D}}$ and $\gamma^* : \alpha^* \notin R^*_{\gamma:\alpha}$, or $\gamma^* : \alpha^* \in Smax(CR(R^*_{\gamma:\alpha}))_{c_9}$. Hence, all the cases of Definition 2.50 satisfy the proposition statement above.

■

References

[AlMa-82] C. Alchourrón, D. Makinson, *"The Logic of Theory Change: Contraction functions and their associated functions"*, Theoria 48, 1982.

[AlMa-85] C. Alchourrón, D. Makinson, *"On the Logic of Theory Change: Safe Contractions"*, Studia Logica, 44, 1985.

[AlMa-86] C. Alchourrón, D. Makinson, *"Maps between some different kinds of contractions functions"*, Studia Logica, 45, 1986.

[AGM-85] C. Alchourrón, P. Gärdenfors, D. Makinson, *"On the Logic of Theory Change: partial meet functions for contraction and revision"*, Journal of Symbolic Logic, 50, 1985.

[BoGo-93] C. Boutilier, M. Goldszmidt, *"Revision by Conditionals Beliefs"*, in Proceedings of the AAAI Conference, 1993.

[Cho-93] L. Cholvy, *"Proving Theorems in a Multi-Source Environment"*, in Proceedings of IJCAI-93, Chambery, vol.1 , pp 66-71, 1993.

[Dar-96a] F.C.C. Dargam, *"Compromised Updates in Deductive Databases"*, Research Report (available by anonymous ftp to *theory.doc.ic.ac.uk*, in the file *papers/Dargam/dar96a.ps*).

[Dar-96c] F.C.C. Dargam, *"A Compromised Characterization to Belief Revision"*, Imperial College Research Report DoC 96/2, (available by anonymous ftp to *theory.doc.ic.ac.uk*, in the file *papers/Dargam/dar96c.ps*).

[Dar-96d] F.C.C. Dargam, *"On Reconciling Conflicting Updates:, A Compromised Revision Approach"*, PhD. Thesis, Department of Computing, Imperial College, to appear.

[DaGa-93] F.C.C. Dargam; D. Gabbay, *"Resolving Conflicting Actions and Updates"*, (extended abstract), in Proceedings of the Compulog Net Meeting on Knowledge Representation and Reasoning CNKRR'93, Lisbon, July 1993.

[Doy-79] J. Doyle, *"A Truth Maintenance System"*, Artificial Intelligence 12, pp 231-272, North Holland Pub. Co., 1979.

[Elk-90] C. Elkan, *"A Rational Reconstruction of Non-monotonic Truth Maintenance Systems"*, Artificial Intelligence 43(2), pp 219-234, North Holland Pub. Co., 1990.

[FrLe-94] M. Freund, D. Lehmann, *"Belief Revision and Rational Inference"* Technical Report, TR94-16, Institute of Computer Science, The Hebrew University of Jerusalem, Israel, 1994.

[Gab-94] Dov Gabbay, *"LDS - Labelled Deductive Systems - Volume I Foundations"*, MPI-I-94-223, Max-Planck-Institut fur Informatik, Saarbrucken, Germany, 1994. (Intermediate draft of a forthcoming book by Oxford University Press).

[GaHu-91] D. Gabbay; A. Hunter, *"Making Inconsistency Respectable - Part 1: A Logical Framework for Inconsistency in Reasoning"*, in Fundamentals of AI Research, LNCS 535, Springer-Verlag, 1991.

[GaHu-93] D. Gabbay; A. Hunter, *"Making Inconsistency Respectable - Part 2: Meta-level handling of inconsistency"*, LNCS 747, Springer-Verlag, 1993.

[Gal-90] J. Galliers, *"The Positive Role of Conflict in Cooperative Multiagent Systems"*, in Decentralized AI, edited by Y. Demazeau and J.P. Muller, North-Holland, pp 33-46, 1990.

[GiMa-90] L. Giordano; A. Martelli, *"An Abductive Characterization of the TMS"*, in LNAI 515, J.P. Martins & M. Reinfrank (Eds.), TMS, ECAI-90 Workshop Proceedings, 1990.

[JaPa-90] P.Jackson, J. Pais, *"Semantic Accounts of Belief Revision"*, in LNAI 515, J.P. Martins & M. Reinfrank (Eds.), TMS, ECAI-90 Workshop Proceedings, 1990.

[KaMe-92] H. Katsuno, A.O. Mendelzon, *"On the Difference between Updating a Knowledge Base and Revising it"*, in Belief Revision, edited by P.Gärdenfors, Cambridge University Press, pp. 183-203, 1992.

[Kle-86a] J.d. Kleer, *"An Assumption-based TMS"*, Artificial Intelligence 28, pp 127-162, North Holland Pub. Co., 1986.

[Mak-85] D. Makinson, *"How to give it up"*, Synthese 62, 1985.

[Mak-93] D. Makinson, *"Five Faces of Minimality"*, Studia Logica, Vol.53, n.3, 1993.

[MBM-95] D. Monteiro; E. Bertino; M. Martelli, *"Transactions and Updates in Deductive Databases"*, Technical Report DOC-95/2, Department of Computing, Imperial College, 1995.

[Neb-89] B. Nebel, *"A Knowledge level Analysis of Belief Revision"*, in Proceedings of the 1st. Conference on Principles of Knowledge Representation and Reasoning, 1989.

[Neb-90] B. Nebel, *"Reasoning and Revision in Hybrid Representation Systems"*, Chapter 6 - Belief Revision, in LNAI 422, 1990.

[PiCu-89] *"A Truth Maintenance System Based on Stable Models"*, S. G. Pimentel; J. L. Cuadrado, in Proceedingsof the North American Conference on Logic Programming NACLP, Cleveland, pp 274-290, 1989.

[RaFo-89] A. Rao, N. Foo, *"Minimal Change and Maximal Coherence: A Basis for Belief Revision and Reasoning about Actions"*, in Proceedings of the 11th International Joint Conference on Artificial Intelligence, pp. 966-971, 1989.

[RoPi-91] *"A Nonmonotonic Assumption-Based TMS using Stable Bases"*, W. Rodi; S. Pimentel, in Proceedings of KR'91, pp 485-495, 1991.

[Rot-91] H. Rott, *"Two Methods of Constructing Contractions and Revisions of Knowledge Systems"*, Jounal of Philosophical Logic, 20, 1991.

[Rot-92] H. Rott, *"On the Logic of Theory Change: more maps between different kinds of contractions functions"* in Belief Revision, edited by P.Gärdenfors, Cambridge University Press, pp. 122-141, 1992.

[SaKo-87] F. Sadri; R. Kowalski, *"A Theorem-Proving Approach to Database Integrity"*, in Foundations of Deductive Databases & Logic Programming, Minker ed., 1987.

[SaIw-91] K. Satoh; N. Iwayama, *"Computing Abduction by using TMS"*, April 1991.

[Sri-93] S. Sripada, *"A Temporal Approach to Belief Revision in Knowledge Bases"*, in Proceedings of the 9th. IEEE Conference on AI for Applications, CAIA-93, Florida, March 1993.

[Wag-94] G. Wagner, *"Vivid Logic"*, Lecture Notes in Artificial Intelligence, LNAI 764, Springer Verlag, 1994.

[WaCh-91] X. Wang; H. Chen, *"On the Semantics of TMS"*, in Proceedings of the IJCAI '91, pp 306-309, 1991.