UNIVERSIDAD DE LAS AMÉRICAS – PUEBLA

Escuela de Ingeniería

Departamento de Ingeniería en Sistemas Computacionales

SEMANTICS FOR NONMONOTONIC REASONING: A LOGICAL APPROACH

Tesis profesional presentada por

Juan Antonio Navarro Pérez

como requisito parcial
para obtener el grado de Maestría en Ciencias con Especialidad en Ingeniería en Sistemas Computacionales.

Dirigida por Dr. Mauricio Osorio Galindo y aceptada por el Departamento de Ingeniería en Sistemas Computacionales.

Sta. Catarina Mártir, Puebla, Primavera de 2007

UNIVERSIDAD DE LAS AMÉRICAS – PUEBLA

Escuela de Ingeniería

SEMANTICS FOR NONMONOTONIC REASONING: A LOGICAL APPROACH

Tesis profesional presentada por

Juan Antonio Navarro Pérez

como requisito parcial
para obtener el grado de Maestría en Ciencias con Especialidad en Ingeniería en Sistemas Computacionales.

Jurado calificador

Presidente
Dr. David Sol

Vocal y Director
Dr. Mauricio Osorio Galindo

Secretario
Dr. Daniel Vallejo

FORMA DE LIBERACIÓN FINAL

Tesis profesional sustentada por Juan Antonio Navarro Pérez como requisito parcial para obtener el título de Maestría en Ciencias con Especialidad en Ingeniería en Sistemas Computacionales.

Aceptada por el Departamento de Ingeniería en Sistemas Computacionales.

Presidente Dr. David Sol	Vocal y Director Dr. Mauricio Osorio Galindo
Secretario Dr. Daniel Vallejo	Jefe del Departamento Dr. David Sol

9 de diciembre de 2005

CONTENTS

List of Tables

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Men and women have been passionate about logic probably since the very early beginning of modern human civilizations. Historians date the origin of logic as a discipline back to the 4th century BC, particularly in the regions of China, India and Greece (Gabbay and Woods, 2004a). The main motivation of such logic pioneers was the study of *correct argumentation* or, in other words, how to distinguish the good from the bad arguments while explaining or discussing about any particular topic. Perhaps most broadly known are the works and theories of Aristotle, who developed the concept of syllogisms and the practice of dialectics: a search of the *truth* by an exchange of logical arguments.

It was later in the 17th century when, most prominently by philosophers such as René Descartes and Gottfried W. von Leibniz, the foundations of logic as a framework for the study of *mathematics*, and of *science* in general, were first stated. An early idea of how an *alphabet of human thought* could be used to *systematically derive any natural truth* is found, in fact, in the works of Leibniz (Peterson, 1966). In the context of mathematics, but still of course with strong repercussions in philosophy, it was in the early 20th century when these ideas were first realized and, pioneered by logicians such as Gottlob Frege and Bertrand Russell, the development of the system of *mathematical logic* began (Gabbay and Woods, 2004b).

The aim was to show that all of mathematics, at least, could be formalized in terms

of logic: every theorem in mathematics could be systematically, i.e. mechanically, derived from a set of few axioms and inference rules. Frege, for instance, proposed a formalization of arithmetic in terms of logic concepts in his *Grundgesetze der Arithmetik* (Foundations of Arithmetic). Unfortunately Russell discovered a flaw within Frege's axioms for arithmetic: The logic was inconsistent (Hendricks et al., 2000). Not only formulas such as $2+2=4$ were true in the logic, but any other nonsense such as $2+2=5$ or $6\times 9=42$ were also true. A completely useless system.

The idea of a formalization of all mathematics (at least in some sense) finally collapsed when the renowned Hilbert's Program, proposed by the German mathematician David Hilbert in 1920, catastrophically failed when being faced to Gödel's second incompleteness theorem shown in 1931. The theorem by Kurt Gödel, a logician and philosopher of mathematics born in Brünn in Moravia, Austria-Hungary (now Brno in the Czech Republic), not only showed that it is impossible to formalize all of mathematics, but even that —for a relatively small fragment of it— it is also impossible to prove its consistency (Hendricks et al., 2000). For example, the axiomatization of arithmetic given by Giuseppe Peano has nowadays replaced the failed system of Frege, but no one can assure us that a day will never come in which some *Russell* happens to prove that Peano's arithmetic is actually inconsistent. Despite the apparent negative nature of such results, they have enormously helped the scientific and research community to better understand the power and limitations of logic.

It was also in the mid of the 20th century when, most notably in the works of Alan Mathison Turing, the first clear and concrete ideas on how to mechanically, e.g. automatically, prove mathematical theorems were exposed. The dream of having *thinking machines* —capable of reasoning and solving problems— was of course not new at that time, but it was certainly Alan Turing who pointed in the right direction towards a plausible realization of it. His famous proof about the *undecidability* of the *Entscheidungsproblem* (the halting problem) served to clarify, analog to what the Gödel's theorem did in logic, the limits of what a machine *can do*, and thus laying out the foundations of the *theory of computation*.

The idea of having machines that are able to *understand* and *solve* our problems is, of course, highly appealing and fascinating. Following to the discoveries of Alan Turing many researchers, including Turing himself, rushed to predict that —if somehow human knowledge could be expressed within some mathematical notation— the creation of a machine that could *reason*, i.e. artificial intelligence, would be implementable in the near future. This turned out to be much more difficult than expected, mainly because of the complexity involved in human reasoning, but gave rise to the now active research areas of *knowledge representation* and *logic programming*.

1.1 Nonstandard logics

The standard classical logic (in either it's propositional, first or higher order versions) is the most common *logic* that has been widely applied and studied in mathematics. Most of the discussion in the previous introductory text is, for instance, concerned with standard logics only. But as the formal study of logic evolved, many scientists and logicians found that the standard version was not quite appropriate for their needs. Why would someone want to do *logic* in a different way?

Perhaps the first attempts to modify the *logic* were done because of apparent "paradoxes" that arise from the fact that the *implication* connective of classical logic does not always match the intuition of conditionals that we usually express in our everyday life. Perhaps the most famous example being (some variant of):

> If the moon is made of green cheese *then* Elvis never died.

This sentence seems counterintuitive to most people since, whatever material the moon is made of, it clearly has not much to do with whether Elvis is still alive or not. The sentence is however *true* in classical logic because a *false* premise can imply *everything*, even nonsense. To solve this kind of problems, logicians and philosophers have battled and argued trying to come up with a more sensible definition of a *strict* implication that does matches our use of

conditionals in everyday life (Schechter, 2005).

A more practical reason, but still with strong philosophical implications, of why one would like to change the definition of *logic*, is simply because one wants to *reason* about different things. Classical logic was devised as a model to study the *truth* of statements. As proof theory evolved the primary concern shifted from deciding whether a sentence is *true* or *false*, to decide whether we can actually *construct a proof* (in our logic system) of the fact that the sentence is either *true* or *false*. Intuitionistic logic was proposed to this end, by Luitzen Egbertus Jan Brouwer in 1920 and was followed by a whole school of constructionist mathematicians, claiming that mathematics should be built upon such logic (Schechter, 2005). A typical classical claim such as "everything is either true or false" has to be reinterpreted, in our new context, as something like: "for every formula, we can prove (in our logic system) that it is either true or false"; which is no longer obviously true (moreover, it is false in general as a consequence of Gödel's results).

Other logics have been proposed trying to *reason* about many other kinds of things that were not possible to express, at least not in a clear or natural way, within classical logic. The most common examples deal with reasoning about expressions that involve notions such as: *necessarily*, *possibly*, *always*, *eventually*, *must*, *can*, ... A family of *modal logics* was introduced by Clarence Irving Lewis that extended the syntax of logic formulas with new connectives that attempt to model such kinds of notions. Recently, with a strong influence of computer science applications, modal logics have been extended with even more *fancy* connectives that are helpful to describe the execution of a program. This gave rise to the *temporal logics* where it is possible to *reason* about properties such as: "if a program requests access to some resource (e.g. printer, memory, etc.) the operating system will *eventually* make the resource available", or "the program will continue to run *until* the task is finished or is canceled by the user". *Description logics* follow a similar approach, but they are mainly intended to reason about "concepts", "terms" and the relations between them within a knowledge base some particular application domain (e.g. the domain of medical symptoms and diseases, or a

biological classification of species).

In another direction, proponents of the so called *paraconsistent logics* do not like the fact that when some contradiction is found in the logic then an *explosion* phenomenon occur, just as it did to Frege's theory, and *every* formula (including nonsense!) becomes valid in the logic, thus making it trivial and uninteresting. Paraconsistent logics do try to allow, so to speak, *local inconsistencies* that do not cause the explosion problem. Although the idea might seem a bit esoteric at first glance, its study has been actually driven by many modern and relevant applications. In a knowledge database, for example, one does not want to loose all the deductive power of the database just because of a mistakenly entered piece of data. It would clearly be a very desirable feature, if the logic is not only able to handle such inconsistencies, but also to detect them and even automatically fix them. These are the motivations of currently active research areas such as *belief revision and updates* (Carnielli and Coniglio, 2002).

To finalize this brief survey of nonstandard logics, and to get closer to the motivation of the research work presented in this thesis, we have to mention the case of *nonmonotonic logics*. Classical logic is monotonic in the sense that, when you prove a theorem it remains proved forever. It is never the case that, just by adding more information to a given theory, you can end up loosing some of the theorems that you proved yesterday. This is, however, not always the case in the kind of reasoning that we humans do in our everyday lives. Suppose that we have the following, very plausible, set of information:

> Birds usually fly.
>
> Tiki is a bird.

Any sensible person, if given this information and asked to tell whether Tiki can fly, would probably answer: "Yes. Tiki can fly."[1] However, if we further extend our knowledge with the following facts:

[1] Unless trying to be very cautious, skeptic, and wondering why someone would ask such silly questions about flying birds with funny names.

Tiki is a penguin.

Penguins can not fly.

Then we would have to retract our conclusions and, regretfully, accept that Tiki can not fly. Although this example might seem a bit artificial, we humans are constantly faced with situations in which these are the only means of reasoning possible: we often have to come up with conclusions, or even make important decisions, based in some set of incomplete, and often imprecise, knowledge. Nonmonotonic logics address such situations trying to study the meaning of the word "usually" in the example. This gave rise to logic formalisms such as *default reasoning* and the introduction of the concept of *default negation*: "We will assume that any bird can fly, unless we somehow *know that it doesn't*."

Our main object of study, *the semantic of stable models for logic programs*, lies precisely within the context of nonmonotonic reasoning. And, as the reader will see throughout this thesis, is very much related with many of the other nonstandard logics briefly discussed in this section. One of the main motivations that have driven this trend of research is, of course, trying to better model and understand the notion of *commonsense reasoning*. But applications, as we have already seen scattered through this review, can actually be found in very diverse areas such as knowledge management and representation, deductive databases, planning and expert systems, logistics, formal verification, general problem solving, ...

1.2 Thesis structure and contributions

The work presented here is the result of the summed effort of several people working together at the research group in logic and logic programming leaded by Prof. Mauricio Javier Osorio Galindo at Universidad de las Américas, Puebla in México. Most of the survey material on modal logics and the $\mathscr{FOUR}$ bilattice in Chapters 4 and 5 was originally published by Borja (2004) as part of her thesis submitted to Benemérita Universidad Autónoma de Puebla.

The work had later developed into a publication by Osorio, Navarro, Arrazola, and Borja

(2005) in the Journal of Logic and Computation, whose main contribution was the proposal of a new semantic for logic programs, now presented in Chapter 8, originally motivated by the modal logic S5. My main contribution in that paper, and therefore in this thesis as well, was the development of the material in Chapters 6 and 7 where the tools of proof theory and natural deduction, necessary for our main result, were introduced.

New material in this thesis comprises Chapter 3, a brief summary of several logics used throughout this thesis, and Chapter 9, where the results of Osorio et al. (2005) are generalized to consider an even broader class of logics. Significant revisions were made to Chapters 5 and 8, and a few additions at the end of Chapter 7. Most of this new material is also currently being prepared as an independent article for submission to a journal.

The structure of this thesis takes then the following form.

- Formulas are, by themselves, not much more than a string of symbols that are put one after another by following some syntax rules. Chapter 2 describes the general syntax of the logic formulas that we consider in this thesis. In particular, the syntax of modal logic formulas, as well as several classes of logic programs are introduced.

- After the shape of the formulas that we use has been introduced, we can start to talk about their meaning. Chapter 3 briefly surveys several different logics that are relevant to the theory that we develop. The survey includes, essentially, some of the main representatives of paraconsistent and intermediate (intuitionistic) logics. In order to understand the definitions of such logics, a brief discussion of proof and model theory is given, but this will be revisited with more detail in later chapters.

- In this and the following chapters we explore different ways of attaching meaning to logical formulas. Here, in Chapter 4, we begin by introducing the semantics of modal logics in terms of abstract mathematical structures called *frames*. These frames implicitly encode different possible *realizations of the world* and thus allow to describe propositions that are *necessarily* true (true in all of them) and *possibly* true (true in

some of them). One of our main results here is a formula that —intuitively— serves to count the number of "worlds" that are available to the logic.

- Another way of defining the semantics of a logic, perhaps the one that most of us are more familiar to, is by means of truth values and truth tables. Chapter 5 defines in more detail the nature of multivalued logics, which generalize the standard classical interpretations by allowing them to assign more that two truth values. In this chapter we construct two new paraconsistent logics and show some of the relations that they have with respect to the logics introduced earlier in Chapter 3.

- In the previous two chapters we assigned meaning to formulas by using *models* or *interpretations* of the objects that the formulas somehow represent. Chapter 6 introduces the dual concept of proof theory where the notions of *axioms* and *inference rules* are used to describe, in a very systematic way, how the *proofs* of theorems are to be constructed. The main result of this section is an axiomatization of the family of modal logics introduced in Chapter 4.

- Natural deduction is also a proof system that instead of placing an emphasis on axioms, as the Hilbert style systems introduced in the previous chapter do, most of the *deduction* is done through inference rules. We use the tools of Natural deduction to provide, in Chapter 7, a very small system that can be used to establish all the possible atomic deductions of normal logic programs. This result is crucial and serves to link all the theory from previous the chapters to the applications in the following two.

- At last, in Chapter 8 lies the core of all our motivations to study the theory of earlier chapters: The use of logic to define nonmonotonic reasoning systems and their application to logic programming. This chapter explores a couple of ideas of how nonstandard, but still monotonic, logics can be used to produce nonmonotonic inference systems. The chapter ends showing the relation between the different approaches considered and a brief discussion of their properties.

- Chapter 9 makes use of all the methodologies and techniques that had been developed in the previous chapters in order to prove one of the main results presented in this thesis. The result states that a large class of monotonic logics provide —including several paraconsistent, intermediate and modal logics— actually end up defining the same semantic, at least for normal logic programs, when one of the mechanisms introduced in the previous Chapter 8 is used.

- Finally, Chapter 10 closes this thesis with general comments and conclusions. The relevance of the results contained in this work are briefly discussed, and some lines for future research are outlined.

1.3 Technical details and motivation

The idea of using modal logics to formalize nonmonotonic reasoning (NMR) can be traced back to McDermott and Doyle (1980). Subsequently McDermott (1982) attempted to define nonmonotonic logics based on the standard T, S4 and S5 logics. But he observed that, unfortunately, the nonmonotonic version of S5 collapses to ordinary logic S5. Moore (1988) suggested the use of autoepistemic logic (AEL) as an alternative formalization of nonmonotonic reasoning to avoid the problems encountered with standard modal logics. Moore explains that the real problem with nonmonotonic S5 is not the S5 schema, but the adoption of $\Box A \rightarrow A$ as an axiom in the logic. He argues that "the S5 schema merely makes explicit the consequences of adopting $\Box A \rightarrow A$ as a premise schema that are implicit in the logic's natural semantics" (Moore, 1988). Gelfond (1987) also showed that the perfect models of stratified logic programs can be characterized in terms of extensions of the corresponding autoepistemic theory. His characterization is based on the interpretation of $not\, a$ as $\neg\Box a$. In fact, Baral (2003) explains that the definition of stable models by Gelfond and Lifschitz (1988) was inspired by this transformation. Having in mind McDermott and Doyle's work, this idea can be interpreted as bounding introspection to objective (non modal) formulas. Schwarz

(1991) proved later the equivalence of AEL with the logic KD45 and, more recently, Lifschitz was able to characterize the stable semantics of disjunctive programs in terms of AEL via Gelfond's translation (Baral, 2003). Autoepistemic logic gained a lot of interest (well deserved) but, at the same time, the approach based on modal logics was almost abandoned. We suggest the reader to review the work of Marek and Truszczyński (1993), where it is possible to find a detailed discussion of the development of the field.

The term *grounded* in nonmonotonic logics refers to the idea of enabling the agent to make only assumptions that are 'grounded' in the world's knowledge. According to Donini, Nardi, and Rosati (1997) the notion of groundedness was actually introduced by Konolige (1987). It is worth to mention that groundedness has a rather intuitive motivation: "it corresponds to discarding the reasoning based on epistemic assumptions, which would enable, for example, to conclude that something is true in the world just by assuming to know it" (Donini et al., 1997). Donini, Nardi, and Rosati renewed interest in nonmonotonic S5 (and other normal modal logics) by studying their grounded versions. They showed, in particular, that grounded nonmonotonic S5 does not collapse with S5. We continue this line of research (Osorio et al., 2005), but restricted to what we called K-basic formulas (sentences with modalities applied only to literals).

Our approach is based on Gelfond's original interpretation and the experience on stable model semantics that shows how it suffices to apply modalities to literals, instead of arbitrary complex formulas, in order to express interesting problems. With our restricted syntax, we recently showed that all ground nonmonotonic modal logics between T and S5 are equivalent (Osorio et al., 2005). Furthermore, we also show that these logics are equivalent to a nonmonotonic logic that we constructed using the well known $\mathscr{FOUR}$ bilattice (Osorio et al., 2005). We called this semantic GNM-S5 as a reminder of its origin in the logic S5. We also proved that GNM-S5 has the properties of classicality and extended cut (Osorio et al., 2005).

Pearce (1999) presented a characterization of the stable model semantics in terms of a collection of logics. He proved that a formula is "entailed by a disjunctive program in

the stable model semantics if and only if it belongs to every intuitionistically complete and consistent extension of the program formed by adding only negated atoms". Moreover, he also showed that in place of intuitionistic logic, any proper intermediate logic can be used. The strongest intermediate logic is Gödel's 3-valued logic G_3. We call weak completion the construction used by Pearce.

In another recent paper we showed how to express G_3 in Łukasiewicz's 3-valued logic ($Ł_3$) (Osorio et al., 2004b). Since G_3 can also be used to express the stable model semantics, we found a new characterization of stable models based on $Ł_3$.

We also introduced a very similar 3-valued logic based in the $Ł_3$ logic that we called G'_3 (Osorio et al., 2004b). Based on weak completions over G'_3, we introduced a new semantics. Part of the work presented in this thesis shows that such semantics is also equivalent to GNM-S5. Since G'_3 is related to paraconsistent logics we decided to study a very large fragment of such logics. We consider all logics stronger or equal to C_ω, the weakest paraconsistent logic, and weaker or equal to Pac, a well known maximal paraconsistent logic studied by Avron (1991). In this thesis it is proved that the weak completions of all these logics are equivalent to each other for normal programs and that, moreover, they are equivalent to GNM-S5. Hence, a large fragment of logics that include many well known modal and paraconsistent logics have this invariant property. Finally, as a direct consequence of our discussion, we show that this invariant property can be expressed as a fixed-point in a very similar way to the definition of the stable semantics. Therefore, we claim that GNM-S5 is a good candidate for defining nonmonotonic semantics that are closer to the direction of classical logic. Some more results around these topics are also presented, with the aim to understand nonmonotonicity in further detail.

CHAPTER 2

SYNTAX OF LOGIC FORMULAS

In this chapter we introduce the syntax of propositional modal logic formulas, which are the central object of study in this thesis. Logic formulas, by themselves, are not much more than a bunch of symbols that are put together by following a simple set of rules. These formulas will be later *filled in* with meaning by using different semantics, which are the main topic of the following chapters. One could imagine the syntax as being the *body* of our object of study, while the semantics is its *soul*. This chapter is, therefore, mainly concerned with the *anatomy* of propositional modal logic formulas.

2.1 Syntax of modal logic

Formally, the language of *propositional modal logic* consists of:

- An enumerable set $\mathscr{L}$ of elements that are called *atomic formulas* or simply *atoms* and will be usually represented with lowercase letters: a, b, c, $\ldots$;
- the binary connectives $\wedge$ (*conjunction*), $\vee$ (*disjunction*) and $\rightarrow$ (*implication*);
- the unary connectives $\neg$ (*negation*), $\Box$ (*modal necessitation*) and $\Diamond$ (*modal possibility*);
- the 0-place connectives $\bot$ (*falsity*) and $\top$ (*truth*);

- and parenthesis that are used as auxiliary punctuation symbols: $($, $)$.

The language is called *propositional*, because it does not allow the use of variables or quantification, it only contains basic atomic propositions. Moreover, it is also *modal*, because it does include the modal connectives $\Box$ and $\Diamond$. Formulas (which will be denoted with uppercase letters A, B, C, $\ldots$) are constructed as usual by combining these basic connectives together. Parenthesis might be dropped when there is no possible source of confusion. The usual practice is that unary connectives bind the strongest, followed by conjunction and disjunction at the same level, and finally implication that binds the weakest. As an example, in the formula

$$((\neg(a \vee b) \wedge c) \rightarrow (\Box d \vee (e \wedge \neg f))) \, ,$$

parenthesis might be dropped to yield

$$\neg(a \vee b) \wedge c \rightarrow \Box d \vee (e \wedge \neg f) \, .$$

Depending on the semantical approach, which are the subject of later chapters, only a few of these connectives are actually considered to be *primitive*, while the others are introduced as syntactical abbreviations of the primitive ones. In the context of modal logics it is standard to accept, for instance, only $\{\rightarrow, \Box, \bot\}$ as primitive connectives and introduce the rest of them according to the following definitions:

$$\begin{aligned} \neg A &:= A \rightarrow \bot \, , & \top &:= \neg\bot \, , \\ A \vee B &:= \neg A \rightarrow B \, , & A \wedge B &:= \neg(\neg A \vee \neg B) \, , \\ \Diamond A &:= \neg\Box\neg A \, . \end{aligned}$$

Another standard defined connective, which is rarely taken as primitive, is the biconditional $A \leftrightarrow B := (A \rightarrow B) \wedge (B \rightarrow A)$. Moreover we may also use, following the tradition in logic programing, $A \leftarrow B$ as an alternate way of writing the formula $B \rightarrow A$.

Note however that, even at this point, logic connectives have absolutely no meaning by themselves; formulas are still not more than a string of symbols put together following some simple rules. The names of the connectives (e.g. 'disjunction' or 'negation') do state some intuition of what the symbol is *intended* to mean, but their *formal meaning* can only be given through a formally defined semantics. Even more confusingly, different semantics might assign different meanings to the same symbol. To avoid such ambiguities we sometimes use subscripts to distinguish such connectives, e.g. the symbols $\wedge_X$ and $\wedge_Y$ are two different conjunction connectives as defined by the logics X and Y respectively.

The connectives $\Box$ and $\Diamond$ are usually known as *modal connectives* and they may have different intuitive interpretations. For instance $\Box A$ could be read as "It is necessarily true that A", "It will always be true that A", "It ought to be that A", or "It is known that A". The possible readings of the formula $\Diamond A$ depend on the one selected for $\Box A$: "It is possible that A is true", "It is sometimes true that A", "It is allowed that A", ...

A *theory* T is just a set of formulas and its *signature*, denoted $\mathscr{L}_T$, is the set of atoms that occur in it. Given a theory T we also define the negated theory $\neg T = \{\neg A \mid A \in T\}$. A *literal* is either an atom a (a positive literal) or a negated atom $\neg a$ (a negative literal); literals will be denoted by lowercase letters close to: $\ldots, x, y, z$. Given a literal x we also define its complement as $\bar{x}$, i.e. $\bar{a} = \neg a$ and $\overline{\neg a} = a$.

We find particularly useful the class of formulas where the scope of the primitive connective $\Box$ is restricted to literals. One of the contributions of this paper is to study the properties of this fragment of modal formulas that we refer as K-basic.

Definition 2.1. The class of *K-basic formulas* is the minimal set X satisfying:

- $\bot \in X$.
- If x is a literal then $x, \Box x \in X$
- If $A, B \in X$ then $(A \to B) \in X$.

It is easy to observe that, introducing other standard connectives as their usual abbreviations, $\neg A$, $A \vee B$ and $A \wedge B$ are also K-basic formulas provided that both A and B are K-basic. For atomic p, $\Diamond p$ is also a K-basic formula.

2.2 Syntax of logic programs

A logic program is, in general, a propositional theory without modal connectives. Moreover, we restrict our attention to finite programs that have, as a consequence, finite signatures. Formally we consider the set $\{\wedge, \vee, \rightarrow, \bot\}$ as primitive connectives while $\neg$ and $\top$ are introduced as the usual abbreviations.

While we present a very general definition of logic program there are several classes of programs that had been of particular interest in the development of the theory and practice of logic programming. A *disjunctive program*, for instance, is a set of formulas of the form

$$h_1 \vee \cdots \vee h_k \leftarrow b_1 \wedge \cdots \wedge b_n \wedge \neg b_{n+1} \wedge \cdots \wedge \neg b_{n+m} \ ,$$

where each h_i and b_j is an atom in $\mathscr{L}$. A rule of this form with $k = 1$ is known as a *normal rule*. A *normal program* is one containing only normal rules. Moreover, if $n = m = 0$ then the right hand side of the formula is interpreted as a single $\top$ and the rule is known as a *fact*; while $k = 0$ means to have $\bot$ in the left hand side of the rule that is known as a *constraint*.

Another class of programs interesting in our context is the class of *basic formulas* that is defined as the class of formulas where the scope of negation is restricted to single atoms. This kind of formulas correspond, as commonly known in literature, to formulas in *negation normal form*. Formally it corresponds to the minimal set X satisfying:

- If $p \in \mathscr{L}$ then $p, \neg p \in X$.
- If $F, G \in X$ then $(F \wedge G), (F \vee G), (F \rightarrow G) \in X$.

A *basic program* is a program that contains only basic formulas. Note that a disjunctive

program without constraints is, in particular, a basic program. Basic programs are closely related to K-basic theories defined in previous section, following the translation of modal formulas given by Gelfond (1987).

CHAPTER 3

BRIEF SURVEY OF LOGICS

After the shape of the formulas that we use has been introduced, we can start to talk about their meaning, their 'soul' or 'personality'. This chapter presents several different logics that are relevant to the theory developed in this thesis, including some of the main representatives of paraconsistent and intermediate logics. In order to understand the definitions of such logics, a brief discussion of proof and model theory is given, but this will be revisited with more detail in following chapters.

We consider a *logic* simply as a set of formulas that, moreover, satisfies the following two properties: (i) is closed under modus ponens (i.e. if A and $A \to B$ are in the logic, then also B is) and (ii) is closed under substitution (i.e. if a formula A is in the logic, then any other formula obtained by replacing all occurrences of an atom b in A with another formula B is still in the logic). The elements of a logic are called *theorems* and the notation $\vdash_X A$ is used to state that the formula A is a theorem of X (i.e. $A \in X$). We say that a logic X is *weaker or equal* than a logic Y if $X \subseteq Y$, similarly we say that X is *stronger or equal* than Y if $Y \subseteq X$.

When a theory, or equivalently a logic program, T is clear by context we use the symbol $\widetilde{M}$ to denote the complement $\mathscr{L}_T \setminus M$. Moreover, given a theory T we define the negation of the theory $\neg T$ as the set $\{\neg F \mid F \in T\}$.

Hilbert style proof systems There are many different approaches that have been used to define the semantics of logic formulas or, in other works, several ways to define logics. In Hilbert style proof systems, also known as axiomatic systems, a logic is specified by giving a set of axioms (which is usually assumed to be closed by substitution). This set of axioms specifies, so to speak, the 'kernel' of the logic. The actual logic is obtained when this 'kernel' is closed with respect to the inference rule of modus ponens. Examples of Hilbert style definitions will be given in later sections.

The notation $\vdash_X F$ for provability of a logic formula F in the logic X is usually extended within Hilbert style systems, given a theory T, using $T \vdash_X F$ to denote the fact that the formula F can be derived from the axioms of the logic and the formulas contained in T by a sequence of applications of modus ponens. The well known result of the *deduction theorem*, which is valid in the logics considered in this paper as explained in section 3.1, gives an alternate interpretation to this notation. A formula F is a logical consequence of T, i.e. $T \vdash_X F$, if and only if $\vdash_X (F_1 \wedge \cdots \wedge F_n) \rightarrow F$ for some formulas $F_i \in T$.

We furthermore extend this notation, for any pair of theories T and U, using $T \vdash_X U$ to state the fact that $T \vdash_X F$ for every formula $F \in U$. If M is a set of atoms we also write $T \Vdash_X M$ when: $T \vdash_X M$ and M is a classical 2-valued model of T (i.e. atoms in M are set to true, and atoms not in M to false; the set of atoms is a classical model if the induced interpretation evaluates the formula to true). Some of these notations are not standard in literature, they follow from our previous work (Osorio et al., 2002, 2004c).

Recall that, in all these definitions, the logic connectives are parameterized by some underlying logic, e.g. the expression $\vdash_X (F_1 \wedge \cdots \wedge F_n) \rightarrow F$ actually stands for $\vdash_X (F_1 \wedge_X \cdots \wedge_X F_n) \rightarrow_X F$.

Multivalued logics An alternative way to define the semantics of a logic formula is trough truth values and interpretations. Multivalued logics generalize the idea of using truth tables that are used to determine the validity of formulas in classical logic. The core of a multivalued logic is its *domain* of values $\mathscr{D}$, where some of such values are special and identified as

designated. Logic connectives (e.g. $\wedge$, $\vee$, $\rightarrow$, $\neg$) are then introduced as operators over $\mathscr{D}$ according to the particular definition of the logic.

An *interpretation* is a function $I\colon \mathscr{L} \rightarrow \mathscr{D}$ that maps atoms to elements in the domain. The application of I is then extended to arbitrary formulas by mapping first the atoms to values in $\mathscr{D}$, and then evaluating the resulting expression in terms of the connectives of the logic (which are defined over $\mathscr{D}$). A formula is said to be a *tautology* if, for every possible interpretation, the formula evaluates to a designated value. The most simple example of a multivalued logic is classical logic where: $\mathscr{D} = \{0,1\}$, 1 is the unique designated value, and connectives are defined through the usual basic truth tables. Further examples will be given in the following sections.

Note that in a multivalued logic, so that it can truly be a *logic*, the implication connective has to satisfy the following property: for every value $x \in \mathscr{D}$, if there is a designated value $y \in \mathscr{D}$ such that $y \rightarrow x$ is designated, then x must also be a designated value. This restriction enforces the validity of modus ponens in the logic. The inference rule of substitution holds without further conditions because of the functional nature of interpretations and how they are evaluated.

Here we also have to clarify that, in the context of multivalued logics, we have to interpret the notation $T \vdash_X F$ using the result of the deduction theorem as its definition. In other words, $T \vdash_X F$ is defined for a multivalued logic as $\vdash_X (F_1 \wedge \cdots \wedge F_n) \rightarrow F$ for some formulas $F_i \in T$.

It is customary to write $\models$ and talk about *tautologies* in the context of model theory (e.g. multivalued logics), while the symbol $\vdash$ is used when proving *theorems* in the framework of proof theory (e.g. Hilbert style systems). In this paper, in a slight abuse of notation, we won't distinguish between these two situations writing $\vdash$ and using the term *theorems* in both cases.

3.1 Positive logic

Positive Logic, Pos, is defined by the following set of axioms:

Pos1 $a \rightarrow (b \rightarrow a)$

Pos2 $(a \rightarrow (b \rightarrow c)) \rightarrow ((a \rightarrow b) \rightarrow (a \rightarrow c))$

Pos3 $a \wedge b \rightarrow a$

Pos4 $a \wedge b \rightarrow b$

Pos5 $a \rightarrow (b \rightarrow (a \wedge b))$

Pos6 $a \rightarrow (a \vee b)$

Pos7 $b \rightarrow (a \vee b)$

Pos8 $(a \rightarrow c) \rightarrow ((b \rightarrow c) \rightarrow (a \vee b \rightarrow c))$

Note that this axioms somewhat constraint the meaning of the $\rightarrow$, $\wedge$ and $\vee$ connectives to match our usual intuition. Positive logic however, as it names suggests, does not contain formulas with negation.

It is a well known result that in any logic satisfying axioms **Pos1** and **Pos2**, and with *modus ponens* as its unique inference rule, the *deduction theorem* holds (see e.g. Mendelson, 1987). This theorem holds, in particular, for other logics stronger than positive logic that are also considered in this paper.

3.2 The C_ω logic

The C_ω logic, the weakest paraconsistent logic due to daCosta (1963), is defined as positive logic plus the following two axioms:

Cw1 $a \vee \neg a$

Cw2 $\neg\neg a \rightarrow a$

Note that $a \vee \neg a$ is a theorem of C_ω (it is an axiom of the logic), while the formula $(\neg a \wedge a) \rightarrow b$ is not. This non-theorem shows one of the motivations of paraconsistent logics: they do allow, so to speak, 'local inconsistencies' (global inconsistencies are disallowed as usual). All the paraconsistent logics that we will consider in this paper share the same property. It follows that results such as the contrapositive of implication, i.e. $(a \rightarrow b) \rightarrow (\neg b \rightarrow \neg a)$, are no longer valid in paraconsistent logics.

3.3 The Pac logic

The Pac logic was extensively studied and axiomatized by Avron (1991), who also proved that it is a maximal paraconsistent logic. Pac can be obtained from C_ω by adding the following set of axioms:

Pac1 $((a \rightarrow b) \rightarrow a) \rightarrow a$

Pac2 $a \rightarrow \neg\neg a$

Pac3 $\neg(a \vee b) \leftrightarrow (\neg a \wedge \neg b)$

Pac4 $\neg(a \wedge b) \leftrightarrow (\neg a \vee \neg b)$

Pac5 $\neg(a \rightarrow b) \leftrightarrow (a \wedge \neg b)$

Pac introduces De Morgan's laws explicitly as axioms, it also allows to cancel out two negations in a row, and allows the implication connective to be expressed in terms of disjunction. All of these properties were not true for C_ω.

Perhaps the simplest way of generating a paraconsistent logic is to use a multivalued logic with more than two values, the logic can be made paraconsistent by allowing the valuation of both a formula and its negation to be designated. An alternative definition of the semantics of Pac is therefore through a 3-valued logic with truth values in the domain

x	$\neg x$
0	2
1	1
2	0

$\rightarrow$	0	1	2
0	2	2	2
1	0	1	2
2	0	1	2

Table 3.1: Truth tables of connectives in Pac.

$\mathscr{D} = \{0,1,2\}$ where both 1 and 2 are designated.[1] The evaluation function of logic connectives is then defined as follows: $x \wedge y = \min(x,y)$; $x \vee y = \max(x,y)$; and the $\neg$ and $\rightarrow$ connectives are defined according to the truth tables given in Table 3.1.

The reader can verify, for instance, that the formula $a \wedge \neg a$ evaluates to the designated value 1, when a is mapped also to 1 through a Pac interpretation. Following the results of Avron (see 1991, first prop. in sec. 3.2.1) theorems in Pac are a subset of those in classical logic. Moreover, two important fragments of the logic coincide with the corresponding classical ones, these are the $\{\neg,\wedge,\vee\}$-fragment and the positive one (i.e. the $\{\wedge,\vee,\rightarrow\}$-fragment).

3.4 Intuitionistic logic

Intuitionistic logic, I, is defined as positive logic plus the following axioms:

Int1 $(a \rightarrow b) \rightarrow ((a \rightarrow \neg b) \rightarrow \neg a)$

Int2 $\neg a \rightarrow (a \rightarrow b)$

These two axioms model the role of negation in intuitionistic logic. They allow to do proofs by contradiction but in some limited way; other constructions such as proof by cases (e.g. $a \vee \neg a$) are not valid in intuitionistic logic. Note that this is the opposite situation to C_ω: in intuitionistic logic $(\neg a \wedge a) \rightarrow b$ is a theorem, but $a \vee \neg a$ is not. Intermediate logics, located between intuitionistic and classical logics, also have this same property.

[1] These values are usually denoted in literature by F, $\perp$ and T respectively. In order to simplify notation we use 0, 1 and 2 instead.

3.5 The logic of Here and There

The logic of Here and There, HT, is obtained from intuitionistic logic by adding the following axiom:

G3 $(\neg b \to a) \to (((a \to b) \to a) \to a)$

This logic is actually equivalent to the well known Gödel's 3-valued logic G_3. Gödel defined, in fact, a family of multivalued logics G_i with truth values over the domain $\mathscr{D} = \{0, 1, \ldots, i-1\}$ and with $i-1$ as the unique designated value. Logic connectives are defined as:

- $\bot = 0$, $x \wedge y = \min(x, y)$, $x \vee y = \max(x, y)$, and
- $x \leftarrow y = i - 1$ if $x \leq y$ and y otherwise.

In this paper we are, however, mainly interested in G_3 only.

3.6 Classical logic

Classical logic, C, is obtained from intuitionistic logic by adding the following axiom:

CL1 $(\neg a \to a) \to a$

This axiom enables any sort of proofs by contradiction, and thus gives to the negation connective its full deduction power. Classical logic, of course, coincides with the standard 'truth table' logic of two values. Note that G_2 is, precisely, classical logic.

3.7 Łukasiewicz's 3-valued logic

The polish logician and philosopher Jan Łukasiewicz began to create systems of multivalued logics in 1920. He developed, in particular, a system with a third value to denote "possible"

x	$\sim x$	$\Box x$	$\Diamond x$
0	2	0	0
1	1	0	2
2	0	2	2

$\rightarrow$	0	1	2
0	2	2	2
1	1	2	2
2	0	1	2

Table 3.2: Truth tables of connectives in Ł$_3$.

that could be used to express the modalities "it is necessary that" and "it is possible that". To construct this logic, denoted Ł$_3$, we first have to modify the syntax of our formulas to allow, as primitive connectives, only: the 0-place connective $\bot$ (*failure*) and the 2-place connective $\rightarrow$ (*implication*). These connectives operate over a domain $\mathscr{D} = \{0,1,2\}$, with 2 as the unique designated value, and are defined as follows:

- $\bot = 0$,
- $x \rightarrow y = \min(2, 2 - x + y)$.

Other connectives in Ł$_3$ are introduced in terms of $\neg$ and $\rightarrow$ as follows:

$$\sim A := A \rightarrow \bot \qquad \top := \sim\bot$$

$$A \vee B := (A \rightarrow B) \rightarrow B \qquad A \wedge B := \sim(\sim A \vee \sim B)$$

$$\Box A := \sim(A \rightarrow \sim A) \qquad \Diamond A := \sim A \rightarrow A$$

We use the symbol $\sim$, and call it the *native negation* of Ł$_3$, in order to distinguish it from other negation connectives that will be introduced later in this paper. The truth tables of most connectives are shown in Table 3.2, the conjunction and disjunction connectives (not shown) coincide with the min and max functions respectively. Minari (2003) studies a syntactic characterization of the modal content of Ł$_3$, and checks the behavior of modal operators against some of the relevant modal principles. Minari also studied Ł$_3$'s axiomatization (see Minari, 2003) as well as its relation with modal logics, particularly with S5.

CHAPTER 4

SEMANTICS USING FRAMES

This section presents some of the basic notions of frames that are commonly used to give semantics for modal logics. We review some of the basic results from Goldblatt (1992) and introduce a modal formula that can be used to characterize the number of points in a given frame. This section also presents that the K-basic fragment of modal formulas is invariant under many classes of reflexive frames.

A *frame* is a pair $\mathscr{F} = (S,R)$, where S is a non-empty set, and R is a binary relation on S. A *model* is a triple $\mathscr{M} = (S,R,V)$ where $V\colon \mathscr{L} \to 2^S$ is a function that assigns to each atomic formula $p \in \mathscr{L}$ a subset $V(p)$ of S. Informally speaking $V(p)$ can be thought as the set of points where the atomic formula p is "true".

For a modal formula A, the relation "A is true in the point s of the model $\mathscr{M}$", denoted $\mathscr{M} \models_s A$, is defined recursively as follows:

$$\mathscr{M} \not\models_s \bot\,.$$

$$\mathscr{M} \models_s p \qquad \text{if } s \in V(p)$$

$$\mathscr{M} \models_s A \to B \qquad \text{if } \mathscr{M} \models_s A \text{ implies } \mathscr{M} \models_s B\,.$$

$$\mathscr{M} \models_s \Box A \qquad \text{if } sRt \text{ implies } \mathscr{M} \models_t A,\ \forall t \in S\,.$$

Schemata		Frame Property	
T	$\Box A \to A$	Reflexive	$\forall s(sRs)$
B	$A \to \Box\Diamond A$	Symmetric	$\forall s\forall t(sRt \to tRs)$
D	$\Box A \to \Diamond A$	Serial	$\forall s\exists t(sRt)$
4	$\Box A \to \Box\Box A$	Transitive	$\forall s\forall t\forall u(sRt \wedge tRu \to sRu)$
5	$\Diamond A \to \Box\Diamond A$	Euclidean	$\forall s\forall t\forall u(sRt \wedge sRu \to tRu)$

Table 4.1: Modal schemata and their corresponding frame properties.

It is easy to verify that $\mathscr{M} \models_s \Diamond A$ holds if $\exists t \in S$ such that sRt and $\mathscr{M} \models_t A$.

We say that a formula A is *true in model* $\mathscr{M}$, denoted $\mathscr{M} \models A$, if it is true in all points of $s \in S$. And a formula A is *true in the frame* $\mathscr{F}$, denoted $\mathscr{F} \models A$, if it is true for all possible models $\mathscr{M}$ based on the frame $\mathscr{F}$. Moreover, if $\mathscr{C}$ is a class of frames we say A *is true in* $\mathscr{C}$, denoted $\mathscr{C} \models A$, if it is true for every frame $\mathscr{F} \in \mathscr{C}$.

In these notations that we have just presented, we may often use, instead of the single formula A, a collection of formulas. The notation is used to specify that all formulas in the collection are true in the corresponding point, model, frame or class of frames.

4.1 Frame properties and schemata

A *schema* is a collection of formulas that share a common syntactic form, i.e. the schema $\Box A \to A$ represents the collection $\{\Box B \to B \mid B \text{ is a formula}\}$. It is a common procedure in modal logics to characterize some properties on frames with respect to the validity of some particular schema in the frame. The following well known theorem formally states the relation between some common schemata and their corresponding frame properties.

Theorem 4.1. *Let $\mathscr{F} = (S,R)$ be a frame. Each one of the schemata in Table 4.1 is true in the frame $\mathscr{F}$ if and only if $\mathscr{F}$ satisfies the corresponding property. (Theorems 1.12 and 1.13 in Goldblatt (1992), pages 12 and 13).*

4.2 The counting schemata

A frame is universal if its corresponding relation is universal, i.e. $R = S \times S$. We will use $\mathscr{U}_n$ to denote the universal frame with n different points, i.e. with the set $S = \{0, 1, \ldots, n-1\}$. Universal frames are important since they relate to provability in the modal logic S5 and, as we will see in this paper, they can be useful for logic programming applications. We present now some schemata that can be used to characterize the cardinality of the set S in universal frames.

Definition 4.1. The *counting schema* $\mathbf{F_n}$ is defined, for every integer n, as:

$$\mathbf{F_n} := \bigwedge_{i=1}^{n} \Diamond\Big(A_i \wedge \bigwedge_{j=1}^{i-1} \neg A_j\Big) \rightarrow \Box \bigvee_{i=1}^{n} A_i \ .$$

Some particular instances are the schemata:

$$\mathbf{F_1} = \Diamond A \rightarrow \Box A \ ,$$

$$\mathbf{F_2} = \Diamond A \wedge \Diamond(B \wedge \neg A) \rightarrow \Box(A \vee B) \ ,$$

$$\mathbf{F_3} = \Diamond A \wedge \Diamond(B \wedge \neg A) \wedge \Diamond(C \wedge \neg A \wedge \neg B) \rightarrow \Box(A \vee B \vee C) \ .$$

Proposition 4.1. *Let $\mathscr{F} = (S,R)$ be an universal frame. $\mathscr{F} \models \mathbf{F_n}$ if and only if the frame satisfies $|S| \leq n$.*

Proof. We will show first that having a frame with $|S| \leq n$ implies validity of the formula $\mathbf{F_n}$. Let $\mathscr{M}$ be an arbitrary model based on $\mathscr{F}$ and suppose that the formula $\bigwedge_{i=1}^{n} \Diamond(A_i \wedge \bigwedge_{j=1}^{i-1} \neg A_j)$ is valid in $\mathscr{M}$. From this it follows that there are n different points $s_i \in S$ —for $i = 1, \ldots, n$— with $\mathscr{M} \models_{s_i} A_i$ and $\mathscr{M} \not\models_{s_i} A_j$ for $j < i$. Since $|S| \leq n$, we must have in fact $S = \{s_i\}_{i=1}^{n}$. Since A_i is true in the point s_i we may conclude that $\Box \bigvee_{i=1}^{n} A_i$ is true in the model $\mathscr{M}$.

For the other implication we follow the contrapositive. If $|S| > n$, there are at least $n+1$

distinct points $s_i \in S$ for $i = 1, \ldots, n+1$. Take n different atomic formulas a_i with $i = 1, \ldots, n$ and let $\mathscr{M} = (S, R, V)$ be a model based on $\mathscr{F}$ with the valuation $V(a_i) = \{s_i\}$ for every a_i in the set. It is then clear that the formula $\bigwedge_{i=1}^{n} \Diamond(a_i \wedge \bigwedge_{j=1}^{i-1} \neg a_j)$ is true in $\mathscr{M}$ while $\Box \bigvee_{i=1}^{n} a_i$ is not, since $\bigvee_{i=1}^{n} a_i$ is not true in s_{n+1}. □

Proposition 4.2. *Given two positive numbers $i < j$ then*

$$\{A \mid \mathscr{U}_j \models A\} \subset \{A \mid \mathscr{U}_i \models A\} \ .$$

Proof. We will first show that $\mathscr{U}_j \models A$ implies $\mathscr{U}_i \models A$. Let $\mathscr{M}_i$ be a model based on $\mathscr{U}_i$ with some valuation function V_i. Then define the model $\mathscr{M}_j$ based on $\mathscr{U}_j$ with a valuation $V_j \colon \mathscr{L} \to 2^{\{0,1,\ldots,j-1\}}$ as follows:

$$V_j(p) = \begin{cases} V_i(p) \cup \{i, \ldots, j-1\} & \text{if } i-1 \in V_i(p), \\ V_i(p) & \text{otherwise.} \end{cases}$$

It is simple to show, since the relation is universal, that for $0 \leq k < i$ and every formula B:

$$\mathscr{M}_i \models_k B \quad \text{iff} \quad \mathscr{M}_j \models_k B \ .$$

Since $\mathscr{U}_j \models A$ we know that, in particular, A is true at all points of the model $\mathscr{M}_j$ that we constructed. Previous note makes it clear that A is also true at all points in the model $\mathscr{M}_i$.

To prove that the subset relation is proper just note that, by Proposition 4.1, the schema $\mathbf{F_i}$ is true in $\mathscr{U}_i$ but some instance of it is not true in $\mathscr{U}_j$. □

Corollary 4.1. *A formula is true in the class of universal frames with $|S| \leq n$ if and only if it is true in the frame $\mathscr{U}_n$.*

Proof. If a formula is true in all universal frames with $|S| \leq n$ it is true, in particular, in the frame $\mathscr{U}_n$. Conversely if a formula is true in the frame $\mathscr{U}_n$, by Proposition 4.2, it is true in all universal frames with $|S| \leq n$. □

4.3 K-equivalent frames

In this section we have one of the main building blocks for our principal results. It states that, with respect to K-basic formulas, many classes of frames (that share the property of being reflexive) are equivalent.

If $\mathscr{M} = (S,R,V)$ is a model based on a reflexive frame and s is a fixed point in S we can construct a model $\mathscr{M}'$ based on the universal frame with two points $\mathscr{U}_2$ such that a K-basic formula is true in the original model if and only if it is true in the prime model (see Lemma A.1).

We say that two classes of frames $\mathscr{C}_1$ and $\mathscr{C}_2$ are K-equivalent if, for every K-basic formula A, it holds that $\mathscr{C}_1 \models A$ iff $\mathscr{C}_2 \models A$. Then we have that all the frames in the class of reflexive frames are K-equivalent to the frame $\mathscr{U}_2$ (see Proposition A.1).

The previous facts are quite useful to determine K-equivalence as we see in the following theorem.

Theorem 4.2. *The following classes of frames are K-equivalent:*

- *The class of all reflexive frames.*
- *The class of all reflexive and transitive frames.*
- *The class of all symmetric, reflexive and transitive frames.*
- *The class of all universal frames.*
- *The class of all universal frames with* $|S| \leq 2$.
- *The class of all universal frames with* $|S| \leq n$.

Proof. All the equivalences follow immediately from Lemma A.1 and Proposition A.1 since $\mathscr{U}_2$ is a frame in all of these classes. □

The previous result is of great significance in order to show that the logics S4, S5, and some others we will construct agree in the class of K-basic formulas. We remark that the class of K-basic formulas is useful in computer science. Gelfond (1987), in fact, uses a subclass of K-basic formulas to characterize stable models using AEL.

CHAPTER 5

SEMANTICS USING TRUTH VALUES

In the context of semantics based on truth values our work takes advantage of the expressive power of the $\mathscr{FOUR}$ system to interpret modal connectives. Belnap (1977) originally introduced the four-valued logic $\mathscr{FOUR}$ trying to deal in a useful way with inconsistent and incomplete information. This structure was further investigated by Ginsberg (1988) who proposed bilattices that generalize $\mathscr{FOUR}$, and Fitting (1991) showed how this system is useful to provide semantics for logic programs.

In this section we introduce the $\mathscr{FOUR}$ structure and then we use it to define a particular truth valuation for modal formulas. We also see how $\mathscr{FOUR}$ is expressive enough to emulate other logics such as the Gödel's G_4. Finally we discover some relations between the universal frame $\mathscr{U}_2$ and the $\mathscr{FOUR}$ system.

5.1 Defining G_3 and G_3' via $Ł_3$

Our main motivation to study Łukasiewicz's $Ł_3$ logic is the fact that we found it able to express the semantics of other logics such as Gödel's G_3 logic. Moreover, we found it useful to introduce another logic, which we called G_3', that will play a central role in the results of later sections to define semantics of logic programs.

x	$\neg_{G_3} x$	$\neg_{G'_3} x$
0	2	2
1	0	2
2	0	0

$\rightarrow$	0	1	2
0	2	2	2
1	0	2	2
2	0	1	2

Table 5.1: Truth tables of connectives in G_3 and G'_3.

We first define the connectives $\rightarrow$ and $\neg$ of the G_3 and G'_3 logics as follows (connectives that are not subscripted correspond to $Ł_3$):

$$\neg_{G_3} x := \Box \sim x$$

$$x \rightarrow_{G_3} y := (x \rightarrow y) \wedge \neg_{G_3} \neg_{G_3} (\neg_{G_3} \neg_{G_3} x \rightarrow y)$$

$$\neg_{G'_3} x := \sim \Box x$$

$$x \rightarrow_{G'_3} y := x \rightarrow_{G_3} y$$

Table 5.1 shows the truth tables of these connectives for the G_3 and G'_3 logics. The reader can easily verify that the definitions just given do reproduce the values shown in the tables. Conjunction and disjunction are defined, just as in all other logics considered, as the min and max functions respectively. Furthermore, the reader can verify, that this definition of G_3 coincides with the one given in section 3.5.

In Carnielli and Marcos (2002), G'_3 is introduced only to prove that $a \vee (a \rightarrow b)$ is not a theorem of C_ω. In particular neither **Pac1**, **Pac2** nor **Pac5** are theorems in G'_3. Nevertheless, axioms **Pac3** and **Pac4** do are theorems in G'_3. It will be shown in section 5.5 that G'_3 is sound with respect to C_ω, and it is a well known fact that G_3 is a superset of I. This shows a very interesting relation between the G_3 and G'_3 logics, in particular a nice feature of the $Ł_3$ logic: only with a slight change in the definition of the negation connective one can toggle between defining an intermediate or a paraconsistent logic.

It is quite obvious but still interesting to observe that in G'_3 one can still express the G_3 logic, since $\neg_{G_3} a = a \rightarrow_{G'_3} (\neg_{G'_3} a \wedge \neg_{G'_3} \neg_{G'_3} a)$. An important point to observe is that he definition of $\neg_{G'_3}$, with respect to the original modal language of $Ł_3$, is based on the translation

proposed by Gelfond (1987). In later sections we will apply the same translation to different modal logics.

5.2 Defining a paraconsistent logic via $\mathscr{FOUR}$

This research work was started in (Osorio and Navarro, 2003), and further developed in subsequent publications (Osorio et al., 2005). In these previous publications a study of ground nonmonotonic modal logics between T and S5 was carried out. It was shown that, for a specific class of modal formulas, such logics are equivalent to a particular construction carried out in $\mathscr{FOUR}$.

Belnap (1977) introduced a logic for dealing in an useful way with inconsistent and incomplete information. This logic is based on a structure called $\mathscr{FOUR}$ with four truth values $\{0,1,2,3\}$. This values are usually identified with the symbols $\{\bot, f, t, \top\}$ that provide an intuition of the meaning of the four truth values: the classical t and f, $\bot$ that intuitively denotes lack of information (no knowledge), and $\top$ that indicates inconsistency ("over" knowledge). We will use, however, numbers instead of these symbols in order to keep the notation simple. This values have two different natural orderings shown in Figure 5.1.

- Measuring the truth: The minimal element is 1, the maximal element is 2 and the values 0, 3 are incomparable. Here we have the inverse involution $\neg_{tr}$, the meet and join operators denoted respectively as $\wedge_{tr}$ and $\vee_{tr}$.

- Reflecting differences in the amount of knowledge or information: The minimal element is 0, the maximal element is 3 and the values 1, 2 are incomparable. Here we have inverse involution $\neg_{kn}$, the meet and join operators denoted respectively as $\wedge_{kn}$ and $\vee_{kn}$.

Ginsberg (1988) proposed algebraic structures called *bilattices* that generalize Belnap's $\mathscr{FOUR}$; his motivation for introducing bilattices was to provide a uniform approach for

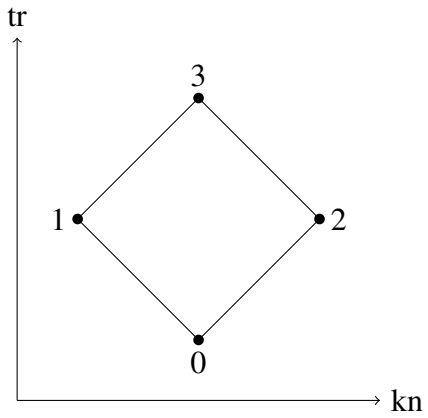

Figure 5.1: $\mathscr{FOUR}$ identified with 0, 1, 2 and 3.

a diversity of applications in computer science. The logical role that $\mathscr{FOUR}$ has among bilattices is very similar to the one of two-valued algebra among Boolean Algebras (Avron, 1999).

We define now a semantic for modal propositional theories, called *modal* $\mathscr{FOUR}$ or just $\mathscr{M}$-$\mathscr{FOUR}$ for short, where the primitive modal connectives are given by:

$$\bot := \neg_{tr}\neg_{kn} a \wedge_{kn} a \qquad \text{for some atom } a$$

$$\Box A := A \wedge_{kn} \neg_{tr} A$$

$$A \rightarrow B := \neg_{tr}\neg_{kn} A \vee_{kn} B$$

It is easy to verify that, under these previous definitions, $\bot = 0$ and the other two connectives have the truth tables shown in Table 5.2. Recall that there is no "typical" implication connective in $\mathscr{FOUR}$, ours is an abbreviation in terms of other $\mathscr{FOUR}$ connectives. The *native negation* connective of $\mathscr{M}$-$\mathscr{FOUR}$ is defined as $\sim A := A \rightarrow \bot$, while conjunction and disjunction are defined to match their corresponding symbols in the knowledge ordering, e.g. $\wedge_{\mathscr{M}\text{-}\mathscr{FOUR}}$ is $\wedge_{kn}$.

The native negation connective of $\mathscr{M}$-$\mathscr{FOUR}$ is used to construct the negation connective $\neg A := \sim\Box A$ using, one more time, the translation proposed by Gelfond (1987). The truth tables of both negation connectives are also given in Table 5.2. Particularly in later sections,

A	$\Box A$
0	0
1	0
2	0
3	3

$\rightarrow$	0	1	2	3
0	3	3	3	3
1	2	3	2	3
2	1	1	3	3
3	0	1	2	3

x	$\sim x$	$\neg x$
0	3	3
1	2	3
2	1	3
3	0	0

Table 5.2: Truth tables of connectives in $\mathscr{M}$-$\mathscr{FOUR}$ and $\mathscr{P}$-$\mathscr{FOUR}$

when we use a modal logic to define the semantics of logic programs, the negation connective *always* means the connective defined as $\neg A := \sim\Box A$ where the symbol $\sim$ represents the native negation of the modal logic.

The final step required in order to use $\mathscr{M}$-$\mathscr{FOUR}$ for reasoning is to choose its designated values. As the Tetravalent Modal Algebras (TMAs) do (Font and Rius, 2000), we let the largest element 3 to be our unique designated value.

The resulting $\{\wedge, \vee, \rightarrow, \neg\}$-fragment of the logic that we have just constructed is what we call $\mathscr{P}$-$\mathscr{FOUR}$. Later, in subsection 8.3.2, this logic is further utilized to define semantics of logic programs. $\mathscr{P}$-$\mathscr{FOUR}$, as well as G_3' introduced earlier, shows some *paraconsistent like* behavior since, in particular, the formula $(a \wedge \neg a) \rightarrow b$ is *not* one of its theorems. Because of this we will informally group all these logics together (C_ω, Pac, $\mathscr{P}$-$\mathscr{FOUR}$, G_3') and call them *paraconsistent* logics. Some properties and relations of these logics are given in the next section.

5.3 Defining G_4 via $\mathscr{FOUR}$

Using the expressive power of the $\mathscr{FOUR}$ bilattice we show in this section how the connectives of the multivalued logic G_4 can also be constructed. First we define a congruence operator $\dagger$ that, in the case of logic can be interpreted as an identity or equivalence connective. The idea of this new connective is taken from Font and Rius (2000) and is defined as[1]:

$$A \dagger B := \big(\neg\Box(A \vee B) \vee \Box(A \wedge B)\big) \wedge \big(\Box\neg(A \vee B) \vee \neg\Box\neg(A \wedge B)\big) .$$

[1] In this and the following definition the symbols $\vee$ and $\wedge$ denote the knowledge ordering, i.e. $\vee_{kn}$ and $\wedge_{kn}$ respectively.

$\vee_{G_4}$	0	1	2	3
0	0	1	2	3
1	1	1	2	3
2	2	2	2	3
3	3	3	3	3

$\wedge_{G_4}$	0	1	2	3
0	0	0	0	0
1	0	1	1	1
2	0	1	2	2
3	0	1	2	3

Table 5.3: Truth tables of $\vee$ and $\wedge$ in logic G_4

Now we are able to express the implication connective of G_4 in terms of $\mathscr{FOUR}$ using the following definition.

$$A \rightarrow_{G_4} B := \Box(A \rightarrow B) \vee (\Diamond B \wedge (A \rightarrow B)) \vee \big(\big(A \dagger (\neg A \wedge_{tr} A)\big) \wedge \big(B \dagger \neg_{tr}(\neg B \wedge_{tr} B)\big)\big)$$

And it is easy to verify that $\rightarrow_{G_4}$ has the following truth table.

$\rightarrow_{G_4}$	0	1	2	3
0	3	3	3	3
1	0	3	3	3
2	0	1	3	3
3	0	1	2	3

Moreover the connective $\neg_{G_4} A$ can be defined, as usual, as an abbreviation of the formula $A \rightarrow_{G_4} \bot$. Recall, however, that in this Gödel logics the usual definitions for the conjunction and disjunction do not hold. In fact the disjunction connective $\vee_{G_4}$, that behaves just like the max function on numbers, can be alternatively defined as:

$$A \vee_{G_4} B := (A \vee_{tr} B) \vee_{kn} (A \wedge_{kn} \neg_{G_4} B) \vee_{kn} (\neg_{G_4} A \wedge_{kn} B) \vee_{kn} \Box A \vee_{kn} \Box B \,.$$

Finally the conjunction connective $\wedge_{G_4}$ can be obtained using the original $\mathscr{FOUR}$ negation: $A \wedge_{G_4} B := \neg(\neg A \vee_{G_4} \neg B)$. It is easy to verify that the defined connectives have the truth tables shown in Table 5.3.

5.4 The frame $\mathscr{U}_2$ and $\mathscr{FOUR}$

This brief section presents one of our first basic results that relates the semantics based on the frame $\mathscr{U}_2$ and the truth values of $\mathscr{FOUR}$.

Proposition 5.1. *For any formula A,*

$$\mathscr{U}_2 \models A \qquad \textit{iff} \qquad \models_{\mathscr{FOUR}} A\,.$$

Proof. From Lemma B.1 it follows immediately that $\mathscr{M} \models A$ iff $I(A) = 3$. For one of the implications, given $I\colon \mathscr{L} \to \{0,1,2,3\}$ we can define $\mathscr{M}^I$ using the valuation $V^I(p) = g(I(p))$ for every atom p. Completely analogously for the other implication, given a model $\mathscr{M}$ based on $\mathscr{U}_2$ with a valuation V, we can define the $\mathscr{FOUR}$ valuation $I^{\mathscr{M}}(p) = g^{-1}(V(p))$ for every atom p. □

5.5 Relations between logics

In this final section we will explicitly state many of the relations between the logics that have been discussed so far. These relations are formally given by the following theorem. Please note that we are comparing the $\{\wedge, \vee, \rightarrow, \neg\}$-fragments of these logics; formulas containing modal connectives, native negation ($\sim$) or other auxiliary connectives are *not* considered when comparing two logics.

Theorem 5.1. *The relations between logics pictured in Figure 5.2 hold. An arrow between two logics denotes proper inclusion of the logics, while the absence of a path between them denotes incomparability.*

Proof. The proof is broken up in several items.

- Pos *is strictly weaker than* C_ω *and* I. Follows trivially by the axiomatic definitions of these logics.

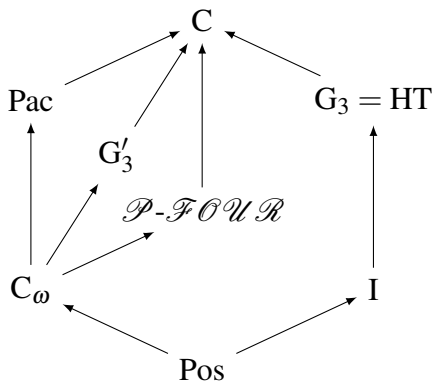

Figure 5.2: Relations between the logics considered

- C_ω *is weaker than* Pac, G'_3 *and* $\mathscr{P}$-$\mathscr{FOUR}$. This follows easily recalling that modus ponens preserves tautologies, and from the fact that each axiom of C_ω is a tautology in all these three logics.

- G'_3, $\mathscr{P}$-$\mathscr{FOUR}$ *and* Pac *are incomparable.* The formula $\neg a \wedge \neg\neg a \rightarrow b$ is a theorem in both G'_3 and $\mathscr{P}$-$\mathscr{FOUR}$ but not in Pac; while the formula $a \rightarrow \neg\neg a$ is a theorem in Pac but neither in G'_3 nor $\mathscr{P}$-$\mathscr{FOUR}$. On the other hand the axiom **Pac1** is valid in $\mathscr{FOUR}$ and not in G'_3; while **Pac3** is valid in G'_3 and not in $\mathscr{FOUR}$. As a side effect this shows that C_ω is strictly weaker than all three Pac, G'_3 and $\mathscr{P}$-$\mathscr{FOUR}$.

- I *is strictly weaker that* G_3. It is a well known result (see e.g. Mendelson, 1987). The hierarchy of G_i logics lie, in fact, between I and G_3, G_2 is classical logic.

- *Paraconsistent and intermediate logics are incomparable.* In all paraconsistent logics, namely C_ω, Pac, G'_3 and $\mathscr{P}$-$\mathscr{FOUR}$, the formula $a \vee \neg a$ is a theorem while $(\neg a \wedge a) \rightarrow b$ is not. In all intermediate logics, namely I and G_3, the formula $(\neg a \wedge a) \rightarrow b$ *is* a theorem, but $a \vee \neg a$ is not.

- *All other logics are strictly weaker classical logic.* The previous item already gave examples of non-theorems for each logic which do are theorems in classical logic.

□

CHAPTER 6

PROOF THEORY

In this section we will review the proof theory of modal logics. In particular we define a new logic that is based on S5 adding $\mathbf{F_n}$ as a new axiom scheme. Moreover we show that this logic is sound and complete with respect to the semantic in universal frames with n points. All the results in previous sections are helpful here to produce interesting results about the relation of several logics in the K-basic fragment, as well as applications of $\mathscr{FOUR}$.

A *normal logic* can be defined in terms of Hilbert type proof systems as a logic that contains the following axiom schemata:

A1 $A \to (B \to A)$

A2 $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$

A3 $(\neg B \to \neg A) \to ((\neg B \to A) \to B)$

K $\Box(A \to B) \to (\Box A \to \Box B)$

And is closed under the following inference rules rules:

- *Modus Ponens*: If the pair of formulas A and $A \to B$ are provable then B is provable.
- *Necessitation*: If the formula A is provable then $\Box A$ is provable.

Other axiom schemata, like those presented in Table 4.1, can be added to produce different normal logics. It has become customary to use the notation $\mathrm{K}\Sigma_1 \cdots \Sigma_n$ to refer the smallest

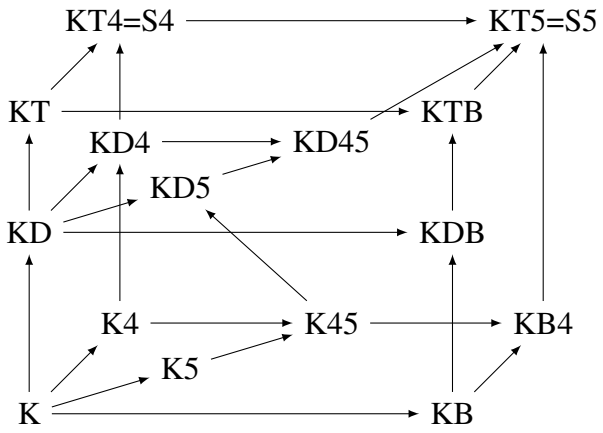

Figure 6.1: Modal logics based on axioms **T**, **B**, **D**, **4** and **5**.

normal logic that contains, in addition to axioms A1–A3, the axiom schemata **K**, $\Sigma_1, \ldots, \Sigma_n$. Figure 6.1 shows several normal logics, in particular S4 and S5 created by Lewis, according to this notation.

We use the notation $\vdash_\Lambda A$ to denote, for a formula A and a logic Λ, that A is provable in the axiomatic logic Λ. Formulas satisfying $\vdash_\Lambda A$ are also known as the theorems of Λ.

Given a set of formulas Γ we also say that a formula A is Λ-deducible from the set Γ, denoted $\Gamma \vdash_\Lambda A$, if there exist $B_0, \ldots, B_{n-1} \in \Gamma$ such that

$$\vdash_\Lambda B_0 \to (B_1 \to (\cdots \to (B_{n-1} \to A) \ldots)) .$$

We write $\Gamma \nvdash_\Lambda A$ when A is not Λ-deducible from Γ. We also say that Γ is Λ-consistent if $\Gamma \nvdash_\Lambda \bot$; and Γ is Λ-maximal if Γ is Λ-consistent and, for every formula A, either $A \in \Gamma$ or $\neg A \in \Gamma$. Given a set of formulas Δ we also use $\Gamma \vdash_\Lambda \Delta$ to denote that every formula in Δ is Λ-deducible from the set Γ. Finally we use the symbol $\Gamma \Vdash_\Lambda \Delta$ to denote that Γ is Λ-consistent and $\Gamma \vdash_\Lambda \Delta$.

6.1 Canonical models

In this section we will introduce a pair of generated models, together with some of their properties, that will be helpful to prove our completeness results. It is common to find this kind of models in completeness proofs for modal logics as in Goldblatt (1992).

Definition 6.1. Let $\mathscr{M} = (S,R,V)$ be a model and let t be a point in S. The *submodel of* $\mathscr{M}$ *generated by* t is defined as

$$\mathscr{M}^t = (S^t, R^t, V^t) \,,$$

where

$$S^t = \{u \in S \mid tR^*u\} \,, \qquad R^t = R \cap (S^t \times S^t) \,, \qquad V^t(p) = V(p) \cap S^t \,,$$

and R^* is the reflexive and transitive closure of R.

Lemma 6.1 (Submodel Lemma)**.** *For any formula A and $u \in S^t$,*

$$\mathscr{M}^t \models_u A \quad \textit{iff} \quad \mathscr{M} \models_u A \,.$$

Definition 6.2 (Canonical Model)**.** The *canonical model* of a consistent normal logic Λ is the structure defined as

$$\mathscr{M}^\Lambda = (S^\Lambda, R^\Lambda, V^\Lambda) \,,$$

where

$$\begin{aligned} &S^\Lambda = \{s \subseteq \mathrm{Fma}(\mathscr{L}) \mid s \text{ is } \Lambda\text{-maximal}\} \,, \\ &sR^\Lambda t \quad \text{iff} \quad \{A \in \mathrm{Fma}(\mathscr{L}) \mid \Box A \in s\} \subseteq t \,, \\ &V^\Lambda(p) = \Big\{s \in S^\Lambda \mid p \in s\Big\} \,. \end{aligned}$$

The symbol $\mathrm{Fma}(\mathscr{L})$ denotes the set of all formulas with atoms in $\mathscr{L}$.

Lemma 6.2 (Truth Lemma)**.** *For any formula A,*

$$\mathscr{M}^{\Lambda} \models A \qquad \textit{iff} \qquad \vdash_{\Lambda} A\,.$$

Theorem 6.1. *If a normal logic* Λ *contains any one of the schemata in Table 4.1, then* R^{Λ} *satisfies the corresponding property (Theorem in Goldblatt (1992)).*

6.2 The logic $\mathrm{S5}_n$

Scroggs (1951) proved that S5 could be characterized by an infinite multivalued logic and moreover that every proper normal extension of S5 can be characterized by a finite multivalued logic. We continue Scroggs's work by studying these extensions in further detail.

The logic $\mathrm{S5}_n$ is obtained adding the axiom schema $\mathbf{F_n}$ to the axioms of S5, this corresponds to the logic $\mathrm{KT4BF}_n$. It is a well known result that the logic S5 is determined by the class of universal frames (Theorem 3.10 in Goldblatt (1992), page 29). We will now show a similar result for the $\mathrm{S5}_n$ logics.

Theorem 6.2. $\mathrm{S5}_n$ *is determined by the class of universal frames with* $|S| \leq n$.

Proof. Soundness is easy. All universal frames satisfy, by Theorem 4.1, the schemata **T**, **4** and **B**. Moreover, since $|S| \leq n$, Proposition 4.1 shows that $\mathbf{F_n}$ is valid in the frame.

For completeness, suppose $\nvdash_{S5_n} A$. By Lemma 6.2, A is false at some point t in the canonical model $\mathscr{M}^{\mathrm{S5}_2}$. Let $\mathscr{M}^t$ be the submodel of $\mathscr{M}^{\mathrm{S5}_n}$ generated by t. By the Submodel Lemma, A is also false at t in $\mathscr{M}^t$. Since $\mathrm{S5}_n$ contains the schemata **T**, **4** and **B** the canonical relation R^{S5_n} is, by Theorem 6.1, an equivalence relation. This shows that $(R^{\mathrm{S5}_n})^* = R^{\mathrm{S5}_n}$ and, in particular, the generated relation R^t is universal. It would suffice to show that $|S^t| \leq n$.

Suppose, by contradiction, that $|S^t| > n$ so that there are $n+1$ different points $s_i \in S$ for $i = 1, \ldots, n+1$. For every $1 \leq i < k \leq n+1$, let $B_{i,k}$ be a formula such that $B_{i,k} \in s_i$ and $B_{i,k} \notin s_k$ (it is easy to see that such formulas must exist). Let $A_i = \bigwedge_{k=i+1}^{n+1} B_{i,k}$ for $1 \leq i \leq n$.

$$\begin{array}{lccccc}
A_1 = & B_{1,2} \wedge & B_{1,3} \wedge \cdots \wedge & B_{1,n+1} & \in s_1 \\
A_2 = & & B_{2,3} \wedge \cdots \wedge & B_{2,n+1} & \in s_2 \\
\vdots & & \ddots & \vdots & \vdots \\
A_n = & & & B_{n,n+1} & \in c_n \\
& \not\in s_2 & \not\in s_3 \quad \cdots & \not\in s_{n+1} &
\end{array}$$

Figure 6.2: Construction of formulas for completeness theorem

It is clear by construction, see Figure 6.2 for an illustration, that $A_i \in s_i$ and $j < i$ implies $\neg A_j \in s_i$. In particular $\Diamond(A_i \wedge \bigwedge_{j=1}^{i-1} \neg A_j)$ is true in $\mathscr{M}^t$. We also know that $\mathbf{F_2}$ is true in the model $\mathscr{M}^t$, follows from Truth and Submodel Lemmas. So that, by logical consequence, $\Box \bigvee_{i=1}^{n} A_i$ should be true in $\mathscr{M}^t$. But, by construction, $A_i \notin s_{n+1}$ for $1 \leq i \leq n$ and $\Box \bigvee_{i=1}^{n} A_i$ is not true in the submodel $\mathscr{M}^t$. □

If we use $X \subset Y$ to denote the proper inclusion of the set of theorems for two given logics X and Y then we have the following result.

Theorem 6.3. $S5 \subset \cdots \subset S5_{i+1} \subset S5_i \subset \cdots \subset S5_3 \subset S5_2 \subset S5_1$.

Proof. The fact that S5 is a lower bound to all the logics $S5_i$ comes from proof theory, since logics $S5_i$ have one additional axiom with respect to the axioms of the logic S5.

The fact that $S5_{i+1} \subset S5_i$ follows by Theorem 6.2 and Proposition 4.2. Moreover this shows that $S5 \neq S5_i$, for all natural numbers i, since there is another logic, namely $S5_{i+1}$, that also contains S5 but is properly contained in $S5_i$. □

The particular case of the logic $S5_2$ is also quite interesting and serves to connect our previous results in $\mathscr{FOUR}$ and the frame $\mathscr{U}_2$ with proof theory.

Corollary 6.1. $S5_2$ *is determined by the class of universal frames with* $|S| \leq 2$.

Proof. Follows by Theorem 6.2. □

Theorem 6.4. *For any formula A,*

$$\vdash_{S5_2} A \qquad \textit{iff} \qquad \models_{\mathscr{FOUR}} A\,.$$

Proof. We know $\vdash_{S5_2} A$ iff, by Corollary 6.1, A is true in the class of universal frames with $|S| \leq 2$ iff, by Corollary 4.1, A is true in the frame $\mathscr{F}_2$ iff, by Proposition 5.1, $\models_{\mathscr{FOUR}} A$. □

Also, with respect to K-basic formulas, we have the following result. Similar as we did with classes of frames, we say that two theories X and Y are K-equivalent if, for every K-basic formula F, $\vdash_X F$ iff $\vdash_Y F$.

Theorem 6.5. *The following logics are K-equivalent:* KT, S4, S5, $S5_2$, $S5_n$.

Proof. Follows by Theorem 4.2, Corollary 6.1, and some other well know results of soundness and completeness, i.e. logic S4 is sound and complete with respect to reflexive and transitive frames. □

6.3 Alternative presentation of $S5_n$

It is just important to mention that there are other possible representations for the axiomatic logics $S5_n$. The axiom $\mathbf{F_n}$ can be alternatively replaced by the following axiom schemata

$$\mathbf{F'_n} := \bigwedge_{i=1}^{n} \Diamond A_i \wedge \bigwedge_{1\leq i<j\leq n} \neg\Diamond(A_i \wedge A_j) \rightarrow \bigvee_{i=1}^{n} A_i\,,$$

as already sketched in Osorio and Navarro (2003); Osorio et al. (2004a).

It is easy to prove that any of the two axiom schemata produce the same logic $S5_n$ since, in particular letting $A_i = B_i \wedge \bigwedge_{j=1}^{i-1} \neg B_j$ in the axiom scheme $\mathbf{F'_n}$ produces, after simplifications, an instance of the axiom scheme $\mathbf{F_n}$. On the other hand $\vdash_{S5} F_n \rightarrow F'_n$ where F_n and F'_n are instances of each axiom schemata constructed with the same subformulas A_i.

CHAPTER 7

NATURAL DEDUCTION SYSTEMS

A *natural deduction system* is another mathematical structure useful to study and define provability relations. Here we will propose a set of inference rules that will be useful to provide a natural deduction system for normal logic programs. The main result obtained is Proposition 7.2 and it will be applied in the next section.

7.1 Motivation in resolution

It is a well known result that resolution is sound and complete with respect to classical logic. For propositional formulas in conjunctive normal form (CNF) resolution is based on just one inference rule namely $\frac{\alpha \vee c \quad \beta \vee \neg c}{\alpha \vee \beta}$ Res and a formula F is derived in classical logic by a theory T if and only if the CNF version of $T \cup \{(\neg F)\}$ derives the empty formula (i.e. contradiction). The following inference rules appear when we try to study resolution techniques applied to normal programs.

Definition 7.1. We define the following inference rules for normal logic programs,

$$\frac{x \leftarrow \alpha \wedge d \qquad d \leftarrow \beta}{x \leftarrow \alpha \wedge \beta}\ \mathrm{G} \qquad \frac{x \leftarrow \alpha \wedge c \qquad x \leftarrow \beta \wedge \neg c}{x \leftarrow \alpha \wedge \beta}\ \mathrm{Ca}$$

$$\frac{x \leftarrow \alpha \wedge c \qquad y \leftarrow \beta \wedge \neg c}{x \leftarrow \neg y \wedge \alpha \wedge \beta}\,\mathsf{R}$$

where x, y, c and d are atoms, and the formulas α, β represent an arbitrary conjunction of literals.

The notation $\frac{\alpha \quad \beta}{\gamma}\,\mathsf{x}$ in previous definition denotes that the formula γ is a logic consequence of the pair of formulas α, β. Moreover, the application of an inference rule is annotated with a letter on the right (G, Ca or R as just defined) that indicates the name of the inference rule used. We usually use X to denote the application of an arbitrary inference rule.

Definition 7.2. Given a set $\mathscr{S}$ of inference rules, a *proof* $P \vdash_{[\![\mathscr{S}]\!]} \alpha$ is defined recursively as follows:

- If $\alpha \in P$ then $P \vdash_{[\![\mathscr{S}]\!]} \alpha$.
- If $P \vdash_{[\![\mathscr{S}]\!]} \beta$, $P \vdash_{[\![\mathscr{S}]\!]} \gamma$ and $\frac{\beta \quad \gamma}{\alpha}\,\mathsf{x}$ with some rule $\mathsf{X} \in \mathscr{S}$ then $P \vdash_{[\![\mathscr{S}]\!]} \alpha$.

Definition 7.3. Given two sets of inference rules $\mathscr{S}$ and $\mathscr{T}$, a *proof* $P \vdash_{[\![\mathscr{S}-\mathscr{T}]\!]} \alpha$ is defined recursively as follows:

- If $P \vdash_{[\![\mathscr{S}]\!]} \alpha$ then $P \vdash_{[\![\mathscr{S}-\mathscr{T}]\!]} \alpha$.
- If $P \vdash_{[\![\mathscr{S}-\mathscr{T}]\!]} \beta$, $P \vdash_{[\![\mathscr{S}-\mathscr{T}]\!]} \gamma$ and $\frac{\beta \quad \gamma}{\alpha}\,\mathsf{x}$ with some rule $\mathsf{X} \in \mathscr{T}$ then $P \vdash_{[\![\mathscr{S}-\mathscr{T}]\!]} \alpha$.

Since these proofs have a tree like structure, we will sometimes call them proof trees. However, we are really thinking in Hilbert-style proofs with no axioms but only hypothesis and inferences rules.

7.2 A system for normal programs

The main goal of this section is to prove that the third rule R is not required to infer a single atom from a normal logic program, the formal statement of this result given in Proposition 7.2. Thus the system GCa is enough to obtain, using resolution techniques, all the

atomic inferences of a given normal logic program. We sincerely don't know if this result is already known or, more likely, can follow as a direct consequence of some particular theorem available from the theory of resolution. Moreover the proof presented here is not very 'neat' but valid and formal anyway. Nevertheless, the result is of importance for the next section.

The sketch of the proof is as follows: first we prove that the rule G can always be moved to the beginning of the proof tree so that it does not have a strong interaction with the other pair of rules Ca and R. Then we show how, in a proof using the system CaR, it is possible to remove applications of the rule R one by one. There are some minor details that we have to handle with care but they are addressed appropriately in the development of the proof.

In order to make this section easier to read we avoid to place here some preliminary results needed in the following proof, but the reader can find it in Appendix D.

Proposition 7.1. *Let P be a normal program and let α be a normal formula. It follows that*

$$P \vdash_C \alpha \quad \textit{if and only if} \quad P \cup T \vdash_{[\![GCaR]\!]} \alpha \,,$$

where $T = \{a \leftarrow a \mid a \in \mathscr{L}_P\}$.

Proof. It is well known that classical logic is sound and complete with respect to resolution. In fact $P \vdash_C F$ if and only if $P \cup \{\neg F\}$ yields $\bot$ (the empty formula) using resolution. For the case of an atomic formula a it is easy to see that this is equivalent to the program P deriving a by the use of resolution. It is a simple exercise to verify that, with respect to the syntax of normal formulas, resolution takes the form of the inference rules G, Ca and R. There is the need, in fact, of a fourth syntactic transformation rule namely:

$$\frac{x \leftarrow \neg x \wedge \alpha}{x \leftarrow \alpha}$$

But, having the set of basic tautologies T (namely, formulas of the form $x \leftarrow x$, for every atom x) this rule follows as a particular case of the application of Ca. □

Proposition 7.2. *Let P be a normal program and a an atom. If $P \vdash_{[\![GCaR]\!]} a$ then $P \vdash_{[\![GCa]\!]} a$.*

Proof. Using Lemma D.2 we obtain that $P \vdash_{[\![G-CaR]\!]} a$. Let $Q = \{\delta \mid P \vdash_{[\![G]\!]} \delta\}$, so that $Q \vdash_{[\![CaR]\!]} a$. Assume that we have a proof for $Q \vdash_{[\![CaR]\!]} a$ that uses the rule R as few times as possible. We claim that, in fact, the rule R is not required. Assume that the proof does require some applications of the rule R. Let R be a set such that $Q \vdash_{[\![CaR]\!]} R$ and there are a pair of formulas $\alpha, \beta \in R$ such that $\frac{\alpha \quad \beta}{\gamma}$ R and $R, \gamma \vdash_{[\![Ca]\!]} a$. Lemma D.5 shows that $R \vdash_{[\![GCa]\!]} a$. This shows that there is a proof of $Q \vdash_{[\![GCaR]\!]}$ with less occurrences of the rule R in the proof tree. But again, using Lemma D.2, $Q \vdash_{[\![G-CaR]\!]} a$ with the same number of uses of the rule R in the proof. But, since Q is already closed for the rule G, it is possible to show that $Q \vdash_{[\![CaR]\!]} a$ with less applications of the rule R arising contradiction. □

If we consider only single atoms, Proposition 7.1 states that a normal program can prove an atom (with respect to classical logic) if and only if the atom can be derived by the same program extended with the theory $T = \{a \leftarrow a \mid a \in \mathscr{L}_P\}$ and using the set of inference rules [[GCR]]. Proposition 7.2 allows us to reduce the set of rules required to [[GC]] only. This motivates the following result.

Proposition 7.3. *Let P be a normal program and a an atom. $P \vdash_{\mathrm{C}} a$ iff $P \vdash_{\mathrm{C}_\omega} a$.*

Proof. From Propositions 7.1 and 7.2, if $P \vdash_{\mathrm{C}} a$ then $P \cup T \vdash_{[\![\mathsf{GC}]\!]} a$, where the set $T = \{a \rightarrow a \mid a \in \mathscr{L}_P\}$. Now one can easily verify that: (i) $a \rightarrow a$ is a theorem of C_ω (follows by axioms **Pos1**–**Pos2**), (ii) rule G is valid in C_ω (follows by axioms **Pos1**–**Pos4**), and (iii) rule C is valid in C_ω (follows by axioms **Pos1**–**Pos4**, **Pos8** and **Cw1**). Therefore it follows that $P \vdash_{\mathrm{C}_\omega} a$.

The converse follows immediately since C_ω is weaker than C (Theorem 5.1). □

Corollary 7.1. *Let P be a normal program and let a be an atom. Let X, Y be any two logics $\mathrm{C}_\omega \subseteq X, Y \subseteq \mathrm{C}$. $P \vdash_X a$ if and only if $P \vdash_Y a$.*

Proof. If $P \vdash_X a$ then, since X is stronger than C_ω, $P \vdash_{\mathrm{C}_\omega} a$ and, by Proposition 7.3, $P \vdash_{\mathrm{C}} a$ so that, since C is stronger than Y, $P \vdash_Y a$. Since X and Y are symmetrical in the statement of the corollary, there is nothing left to prove. □

CHAPTER 8

NONMONOTONIC REASONING AND LOGIC PROGRAMMING

In this final section all the results and constructions previously developed find their conclusion. First we will provide a general definition of "semantics" for logic programs, in particular we briefly describe the popular semantics of answer sets and WFS. Then we present GNM-S5, a proposal for a new semantic whose behavior is closer to the notion of well behaved semantics as defined by Dix (1995a,b). We also prove several properties of this new semantic that serve to justify the previous claim. We would like to stress that the notion of "well behaved semantics" served us only as motivation. The reader do not really need to accept or know about this topic in order to follow the rest of our paper. However, the interested reader is invited to consult the works of Dix (1995a,b) for more information on this subject.

8.1 Semantics for logic programs

For our purposes a *logic program* is nothing more that a theory, i.e. a set of formulas. A *class of logic programs* is, in turn, a set of logic programs that groups together programs that satisfy certain property or syntactic limitation. Most of the results presented in this paper are

concerned, for example, with the class of *normal logic programs*. In a normal logic program all formulas, also known as *clauses*, have the form

$$a \leftarrow b_1 \wedge \cdots b_n \wedge \neg b_{n+1} \wedge \cdots \wedge \neg b_{n+m},$$

where a and all b_i are atoms, $n \geq 0$, and $m \geq 0$. Since, in all logics considered in this paper, the conjunction connective is commutative; we will sometimes abbreviate an arbitrary normal clause by the expression

$$a \leftarrow \mathscr{B}^+ \wedge \neg \mathscr{B}^-,$$

where $\mathscr{B}^+$ and $\mathscr{B}^-$ are supposed to be sets containing, respectively, the positive and negative atoms that occur in the *body* of the clause: $\mathscr{B}^+ = \{b_1, \ldots, b_n\}$ and $\mathscr{B}^- = \{b_{n+1}, \ldots, b_{n+m}\}$. In general we might write in the body of a normal clause any conjunction of literals and/or sets of literals, e.g. $a \leftarrow b \wedge \mathscr{B}$, which is simply intended to mean the conjunction of all such literals. The main focus of many of the results in this thesis is the class of normal logic programs, which is both syntactically very simple but also quite expressive. This class of programs, since it has been found useful in many interesting and relevant applications, has been studied extensively in the logic programming and nonmonotonic reasoning domains.

The class of *disjunctive logic programs* generalizes normal programs by allowing the use of a disjunction in the *head* of the clause. More precisely, clauses in disjunctive programs have the form

$$a_1 \vee \cdots \vee a_k \leftarrow b_1 \wedge \cdots b_n \wedge \neg b_{n+1} \wedge \cdots \wedge \neg b_{n+m},$$

with $k \geq 1$. Note that the head of a disjunctive clause can not be empty.

Given a class of logic programs $\mathscr{C}$, a *semantic operator* is a function that assigns, to each program $P \in \mathscr{C}$, some subset of the power set of $\mathscr{L}_P$. These sets of atoms, which are called the *semantic models* of the program P, are usually some "preferred" subset of the classical (two-valued) models of P.

The most simple example of a semantic operator is the one of *classical models*, which

merely maps each program to its standard two-valued models. Another more interesting example is the semantic of minimal models (see Lloyd, 1987).

Definition 8.1. A set of atoms M is a *minimal model* of P if M is a classical model of P and is minimal (with respect to set inclusion) among the other classical models of P. We use MM to denote the semantic operator of minimal models, i.e. $\mathrm{MM}(P)$ is the set of minimal models of P.

There is another kind of semantics that assign one single intended model to each logic program. Formally a *one-model semantic operator* is a function S that assigns to each program P, in some given class C, a subset of literals from P. This models are usually consistent (i.e. there is no atom a such that both a and $\neg a$ appear in the model) or total (i.e. all the literals relevant to P). Moreover, observe that this models have a three valued nature: an atom can appear either positive or negative in the model (to denote a "true" or "false" fact respectively) or not appear at all (denoting a third "undetermined" state).

Given a scenarios semantic function S it is also common to define the *skeptical semantics* S^{sk} for each program P as follows:

$$S^{sk}(P) = \bigcap_{M \in S(P)} (M \cup \neg(\mathscr{L}_P \setminus M)) \, .$$

Observe that the skeptical version of a scenarios semantics corresponds to a one-model semantics.

Given the one-model semantic functions S_1 and S_2, we define: $S_1 \leq S_2$ if for every program P is true that $S_1(P) \subseteq S_2(P)$. The relations $S_1 = S_2$ and $S_1 < S_2$ are defined as usual. We say that one semantic operator is stronger than other according to this ordering.

8.2 The stable model semantics

The original definition of the stable model semantics was given by Gelfond and Lifschitz (1988) and is the following:

Definition 8.2. Let P be a normal program. For any set M of atoms from P, let P^M be the program obtained from P by deleting

(i) each clause that has a negative literal $\neg b$ in its body with $b \in M$, and

(ii) all negative literals in the bodies of the remaining clauses.

Clearly, P^M contains no negation, so that P^M has a unique minimal model (Lloyd, 1987). If this minimal model coincides with M then we say that M is a *stable model* of P. We use Stable to denote the semantic operator of stable models.

The following is a well known result originally given by Pearce (see 1999), which characterizes the stable models of propositional theories in terms of intermediate logics, i.e. any logic between G_3 and I inclusive. Recall the notation introduced in Chapter 3.

Theorem 8.1. *(Pearce, 1999) Let P be a disjunctive program, M a set of atoms, and X an arbitrary intermediate logic. M is a stable model of P iff $P \cup \neg\widetilde{M} \Vdash_X M$.*

Note that our negation symbol $\neg$ corresponds to the default negation symbol *not* commonly used in logic programming Gelfond and Lifschitz (e.g. 1988). We also point out that, the definition of $\Vdash$ is a bit different from the one used in our previous work (Osorio et al., 2004b, 2002, 2004c). We have previously interpreted the notation $T \Vdash_X M$ as $T \nvdash_X \bot$ (i.e. T is consistent) and $T \vdash_X M$. In the context of Theorem 8.1 both this definition and the one presented in Chapter 3 provide the same result. But this is no longer the case for the logics G'_3 and Pac studied in this paper, so that the definition given in Chapter 3 must be used from now on.

8.3 The X-stable semantics

Theorem 8.1 already suggests a natural generalization of stable models for any arbitrary logic X. We call the construct $P \cup \neg\widetilde{M}$ a *weak completion* of the program P (with respect to the set of atoms M). Strong completions, of the form $P \cup \neg\widetilde{M} \cup \neg\neg M$, provide an alternative generalization of the stable model semantics (see e.g. Osorio et al., 2004c), but are not considered in this paper.

Definition 8.3. Let P be any theory and X any logic. Also let M be a set of atoms. M is a X-stable model of P if $P \cup \neg\widetilde{M} \Vdash_X M$.

8.3.1 The G_3-stable semantics

Of particular interest to us is the G'_3-stable semantics,[1] which is the result of using the logic G'_3 in Definition 8.3. We have to emphasize that the reinterpretation of $\Vdash$ is relevant. Our original definition of G'_3-stable models (Osorio et al., 2004b) gives unwanted results: a very simple program such as $P = \{a\}$ has the undesirable $\emptyset$ model. Such unexpected model appears because $(a \wedge \neg a) \rightarrow \bot$ is not a tautology in G'_3. With the new definition of $\Vdash$ given in Chapter 3 we avoided such problem.

It is also important to note that G_3-stable models coincide exactly with the stable model semantics defined by Gelfond and Lifschitz (1988), this follows by Theorem 8.1 and the fact that G_3 is an intermediate logic. Moreover, the only difference between the two semantics, G_3- and G'_3-stable, is the definition of the negation connective. This small difference in the definition of $\neg$ has been already made explicit in Table 5.1. All this together suggests that Łukasiewicz's 3-valued logic provides a good framework for studying and defining nonmonotonic reasoning systems.

Example 8.1. Consider the following logic program P:

[1] Originally called $Ł_3$-WFS (Osorio et al., 2004b)

$b \leftarrow \neg a.$

$a \leftarrow \neg b.$

$p \leftarrow \neg a.$

$p \leftarrow \neg p.$

It is easy to verify that this program has two G'_3-stable models, which are $\{a, p\}$ and $\{b, p\}$.

8.3.2 The $\mathscr{P}$-$\mathscr{FOUR}$-stable semantics

The very same Definition 8.3 invites to experiment with different kinds of logics and classes of programs to construct semantics of logic programs. In this section we briefly review one such definitions that is given in terms of the logic $\mathscr{P}$-$\mathscr{FOUR}$ of section 5.4. This material was originally introduced by Osorio et al. (2004a), but an extended version of such work is also available (Osorio et al., 2005).

Most of our results in previous work were given for a large class of $\mathscr{M}$-$\mathscr{FOUR}$ formulas (i.e. formulas including modal connectives and native negation) that were called *K*-basic (Osorio et al., 2005). For our current purposes, in terms of the $\mathscr{P}$-$\mathscr{FOUR}$ fragment obtained after applying Gelfond's translation (Gelfond, 1987), such results are relevant to formulas in *negation normal form*; i.e. the scope of the negation connective $\neg$ is restricted to single atoms. Note, in particular, that normal logic programs are in negation normal form. We refer the reader to the work of Goldblatt (1992) for a comprehensive introduction to modal logics.

Theorem 8.2. *(Osorio et al., 2005) All semantics of X-stable models, induced by any modal logic* KT $\subseteq X \subseteq$ S5*, as well as the* $\mathscr{P}$-$\mathscr{FOUR}$*-stable model semantics, are equivalent for the class formulas in negation normal form.*

Do not forget that, in order to apply Definition 8.3 to a modal logic X we first have to apply Gelfond's translation (Gelfond, 1987). This basically means that the negation $\neg a$ actually represents $\sim\Box a$ where $\sim$ is the *native negation* connective of the modal logic. In the

same publication (Osorio et al., 2005) we also had a proof of the following very useful and important result.

Corollary 8.1. *(Osorio et al., 2005) Let P be a normal logic program and let x be an atom. $P \vdash_{\mathrm{C}} x$ if and only if $P \vdash_{\mathscr{P}\text{-}\mathscr{FOUR}} x$.*

In view of the new results and logics considered in this paper, this result will be further generalized in section 7.2 to yield Proposition 7.3.

8.4 The pstable model semantics

The following definition is similar but not identical to the well known reduction due to Gelfond and Lifschitz (1988).

Definition 8.4. Let P be normal program and M a set of atoms. We define

$$RED(P,M) := \left\{ a \leftarrow \mathscr{B}^+ \wedge \neg(\mathscr{B}^- \cap M) \mid a \leftarrow \mathscr{B}^+, \neg\mathscr{B}^- \in P \right\}.$$

Example 8.2. Take the following logic program P (already considered):

$b \leftarrow \neg a.$

$a \leftarrow \neg b.$

$p \leftarrow \neg a.$

$p \leftarrow \neg p.$

Given $M = \{a, p\}$, it follows that $RED(P,M)$ is the program:

$b \leftarrow \neg a.$

$a.$

$p \leftarrow \neg a.$

$p \leftarrow \neg p.$

The following definition provides semantics of logic programs using a fixed-point operator very similar to the original of Gelfond and Lifschitz (1988) and in terms of classical logic.

Definition 8.5. Let P be a normal program, and M a set of atoms. We say that M is a *pstable model* of P if $RED(P,M) \Vdash_{\mathrm{C}} M$. We use PStable to denote the semantic operator of pstable models.

The main result of Chapter 9 will show that, in fact, both the G'_3- and the $\mathscr{P}$-$\mathscr{FOUR}$-stable model semantics (as well as semantics based in many other paraconsistent and modal logics) coincide, for the class of normal programs, with the pstable models just defined. All this seems to provide further evidence that Definition 8.5, and the X-stable semantics in general, provide a solid base to study and propose new semantics for logic programs.

8.5 Well Founded Semantics

The well-founded semantics (WFS) is another paradigm that developed in parallel to the answer set semantics (van Gelder et al., 1991). The main difference between AS and WFS is that in the definition of the former a *guess* is made and then a particular (2-valued) model is constructed and used to justify the guess or to reject it. However, in the definition of WFS, more and more atoms are declared to be true (or false): once a decision has been drawn, it will never be rejected. WFS is defined as a one-model semantics and is based on a 3-valued intended model.

There are some cases, however, where the well-founded semantics does not give the expected results. As the following example from Apt and Bol (1994) shows, the program $P = \{p \leftarrow \neg q, q \leftarrow \neg p, r \leftarrow p, r \leftarrow q\}$ has two stable models. The first model satisfies p while the second satisfies q. Apt and Bol also mention that the formula $p \vee q$ should be, in a reasonable definition for a semantic operator, considered valid and thus set the value the atom r to true. The WFS operator leaves, contrary to this intuition, the atom r as undefined.

Several extensions of WFS have been proposed, trying to overcome this kind of arguments against the semantic, that integrate some additional mechanisms on top of the original definition, e.g. EWFS (Dix, 1991), GWFS (Baral et al., 1990), and WFS$^+$ (Brewka et al., 1997). Dix (1995b) noticed, however, that many of these new semantics have more serious shortcomings than the original WFS and hence he defined a set of principles on which all semantics should be checked against. Such notions helped Dix to propose the concept of *well behaved semantics*.

8.6 The GNM-S5 semantics

Several attempts have been made to model the notion of nonmonotonic reasoning using modal logics. McDermott and Doyle introduced nonmonotonic versions of modal logics by the use of expansions. Given a modal theory T an *X-expansion* of T, based on the modal X, is a theory E that satisfies the equation

$$E = \mathrm{Cn}_X(T \cup \{\neg\Box F \mid F \notin E\}) ;$$

where $\mathrm{Cn}_X(\Gamma)$ denotes the consequence closure of the theory Γ with respect to the particular logic X, i.e. $\mathrm{Cn}_X(\Gamma) = \{F \mid \Gamma \vdash_X F\}$. Depending on the approach some particular X-expansion or the intersection of all X-expansions is considered as the set of nonmonotonic consequences of T.

This is an useful method to produce, for instance, a nonmonotonic version of the logic S4. Authors expected, however, S5 to be a better approach to model the notions of logic programming. Very disappointing was that, as McDermott was able to prove, the nonmonotonic version of S5 is equivalent to the plain monotonic logic S5.

Observe that McDermott's expansions are constructed adding formulas to the original theory. Another approach could be to consider only simple formulas of the form $\neg\Box a$, with a an atom. In fact, Gelfond (1987) was able to characterize the answer set semantics of

stratified normal programs using AEL with this idea and the following translation: A normal clause

$$a_0 \leftarrow a_1 \wedge \cdots \wedge a_n \wedge \neg a_{n+1} \wedge \cdots \wedge \neg a_{n+m} \ ,$$

is translated to the modal formula

$$a_0 \leftarrow a_1 \wedge \cdots \wedge a_n \wedge \neg\Box a_{n+1} \wedge \cdots \wedge \neg\Box a_{n+m} \ .$$

Generalizing this idea we propose the following translation of basic into K-basic formulas.

Definition 8.6. We define the translation $^\circ$ for basic formulas into K-basic modal formulas as follows:

$$\begin{aligned} p^\circ &= p \ , & (F \wedge G)^\circ &= F^\circ \wedge G^\circ \ , \\ (\neg p)^\circ &= \neg\Box p \ , & (F \vee G)^\circ &= F^\circ \vee G^\circ \ , \\ & & (F \to G)^\circ &= F^\circ \to G^\circ \ . \end{aligned}$$

Then we are able to propose a framework to define semantics for basic logic programs in terms of modal logics.

Definition 8.7. Given a modal logic Λ we define the GNM_Λ semantics for the class of basic programs as follows: A set of atoms $M \subseteq \mathscr{L}_P$ is an intended model of P, if the following equation is satisfied:

$$M = \mathrm{ACn}_X(P^\circ \cup \{\neg\Box a \mid a \in \mathscr{L}_P \setminus M\}) \ ;$$

where $\mathrm{ACn}_X(\Gamma)$ denotes the atomic consequence closure of the theory Γ with respect to logic X, i.e. $\mathrm{ACn}_X(\Gamma) = \{a \in \mathscr{L}_\Gamma \mid \Gamma \vdash_X a\}$.

By our previous Theorem 6.5 it follows that any of the modal logics KT, S4, S5, $\mathrm{S5}_n$ and $\mathrm{S5}_2$ produce the same semantic operator when used as in the previous definition. Moreover, by Theorem 6.4, we can use the $\mathscr{FOUR}$ truth values to validate this condition for any of

this logics. We will therefore omit the subscript Λ when it is K-equivalent to, say, logic S5. For historical reasons we call it GNM-S5 semantic.

Note that other modal logics without the reflexive frame property, logic K for instance, can produce different semantics as the following example shows.

Example 8.3. Take the following logic program P:

$$b \leftarrow \neg a \,.$$
$$a \leftarrow \neg b \,.$$
$$a \leftarrow \neg p \,.$$
$$p \leftarrow \neg p \,.$$

It is easy to verify that GNM-S5$(P) = \{\{a, p\}, \{b, p\}\}$, while $\mathrm{GNM_K}(P) = \emptyset$.

There are some counterexamples of the well behavior for several WFS extensions (such as GWFS and EWFS) that do not apply for GNM-S5. In particular GNM-S5 is different from GWFS, EWFS, WFS$^+$, and the revised stable semantics. The following examples serve to prove this affirmation.

Example 8.4. Consider the EWFS semantics that only has an skeptical semantics. The CUT rule and the following program P, all of them taken from Dix (1995a):

$$a \leftarrow \neg a \,.$$
$$b \leftarrow \neg x \wedge a \,.$$
$$y \leftarrow \neg b \,.$$
$$z \leftarrow \neg y \,.$$

Here EWFS$(P) = \{a, b, \neg x\}$, however EWFS$(P \cup \{b\}) = \{a, b, z, \neg x, \neg y\}$. This example shows that the EWFS semantics does not satisfy the CUT rule. But GNM-S5$(P) =$ GNM-S5$(P \cup \{b\}) = \{a, b, z, \neg x, \neg y\}$. In particular the semantics GNM-S5 is different from EWFS. Here of course GNM-S5 refers to its skeptical version.

Example 8.5. Consider the following two program examples taken from Dix (1995a):

$p \leftarrow \neg b$.

$b \leftarrow c$.

$c \leftarrow p \wedge \neg a$.

$a \leftarrow \neg b$.

and

$p \leftarrow \neg b$.

$b \leftarrow p \wedge \neg a$.

$a \leftarrow \neg b$.

One may expect the same semantics of both programs, at least with respect to the common language. However, GWFS infers p in the first program, but it does not in the second. GNM-S5 gives the same answer in both programs which consists in deriving only $\neg c$, under the skeptical version. Hence, GNM-S5 is different to GWFS.

Example 8.6. Consider the program P, taken from Dix et al. (2001):

$a \leftarrow \neg b$.

$b \leftarrow \neg a$.

$x \leftarrow \neg a$.

$x \leftarrow \neg b$.

Note that GNM-S5$(P) = \{\{a,x\},\{b,x\}\}$. Hence, its skeptical version gives GNM-S5$(P) = \{x\}$. This shows that GNM-S5 is different from WFS$^+$, since WFS$^+$ does not infers x.

Example 8.7. Consider this last program P, taken from Pereira and Pinto (2004):

$a \leftarrow \neg b$.

$x \leftarrow \neg y$.

$b \leftarrow \neg a$.

$y \leftarrow \neg x$.

$x \leftarrow a \wedge \neg c$.

$z \leftarrow x \wedge \neg z$.

Note that GNM-S5$(P) = \{\{a,c,x,z\},\{a,c,y\},\{b,x,z\},\{b,y\}\}$. This shows that GNM-S5 is different from the revised stable semantics by Pereira and Pinto. The difference is that they exclude $\{a,c,x,z\}$ from the solution. However, they consider the interesting choice to include it. In order to do it, they also propose the notion of combination revised stable models (CRSM) that adds again $\{a,c,x,z\}$ to the solution. We do not know at this point if both semantics GNM-S5 and CRSM agree. CRSM is presented in a very different setting than GNM-S5 and is defined just for normal programs.

In the following definition we restrict our attention to skeptical semantics. We say that a semantic S satisfies the extended Cut principle if for every program P and pair of literals a and b it holds that: $a \in S(P)$ and $b \in S(P \cup \{a\})$ implies that $b \in S(P)$ (see Dix, 1995a).

We have the informal claim that the particular interpretation that Dix gives to the notion of well behaved semantics has a small defect and in order to correct it we propose to reject to interpret $P \cup M$ as the reduct operation P^M (for a definition on the reduct operator see Dix, 1995b). Note that the notion P^M is a syntactic transformation, not required when $P \cup M$ has a logical meaning.

Take for instance, the following program $P = \{a \leftarrow b, b \leftarrow \neg a\}$ also presented by Dix (1995b). This example is used to show that WFS$^+$ does not satisfy the Extended Cut principle. While WFS$^+(P) = \{a, \neg b\}$, WFS$^+(P \cup \{\neg b\}) = \{\neg a, \neg b\}$. But the set $\{\neg a, \neg b\}$ is neither a 2-valued model, nor a 3-valued model of the logic program $P \cup \{\neg b\}$. This happens because WFS$^+$ does not have a "logical" definition for programs with constraints (negated formulas). Hence, Dix has to interpret $P \cup \{\neg b\}$ as $P^{\{\neg b\}} = \{a \leftarrow b\}$.

GNM-S5 is defined for any basic propositional theory and, in particular, allows the use of constraints. In this particular example, with the same logic program P, we get GNM-S5$(P) =$ GNM-S5$(P \cup \{\neg b\}) = \{a, \neg b\}$. In fact, thanks to the logical approach of GNM-S5 is straightforward to prove that:

Proposition 8.1. GNM-S5 *satisfies Extended Cut.*

We propose to reconsider Dix's work on well behaved semantics, towards a direction of

making it more general and logical based.

Classicality One feature in nonmonotonic reasoning that has been recognized as important by many authors is that semantics should have a closed behavior with respect to classical logic (Denecker et al., 2001; Dix, 1995a,b; Dix et al., 2001; Schlipf, 1992). Classicality, a condition that shows part of this expected behavior, basically states that atomic classical consequences of a program P can be added to the logic program without modifying its semantics. The results in Chapter 7 serve to prove that GNM-S5 satisfies supraclassicality.

Proposition 8.2. *Let P be a normal program and a an atom. If $P \vdash_C a$ then $P \models_{\mathscr{FOUR}} a$.*

Proof. Suppose that $P \vdash_C a$, by Propositions 7.1 and 7.2 we may conclude that $P \cup T \vdash_{[\![GCa]\!]} a$. But again, it is easy to observe that the inference rules G and Ca are valid in Modal $\mathscr{FOUR}$; moreover the set T is not needed since it contains basic tautologies already part of Modal $\mathscr{FOUR}$. So, finally $P \models_{\mathscr{FOUR}} a$. □

Theorem 8.3 (Classicality)**.** *Let P be a normal program and let a be an atom. If $P \vdash_C a$ then $a \in$ GNM-S5(P). Moreover Supraclassicality also holds, namely that* GNM-S5$(P) =$ GNM-S5$(P \cup \{a\})$.

Proof. By Proposition 8.2 we know that $P \models_{\mathscr{FOUR}} a$ and, therefore, it follows that P and $P \cup \{a\}$ are equivalent with respect to $\mathscr{FOUR}$. In particular both programs are interchangeable in the definition of GNM-S5 with respect to Modal $\mathscr{FOUR}$. □

8.7 Examples

To conclude with this section we give some example programs together with their semantics.

Example 8.8. The following are examples of normal programs together with their stable, pstable and minimal models as defined by Definitions 8.2, 8.5 and 8.1 respectively.

- P_1: $a \leftarrow \neg a.$
  $\text{Stable}(P_1) = \{\}$
  $\text{PStable}(P_1) = \{\{a\}\}$
  $\text{MM}(P_1) = \{\{a\}\}$

- P_2: $a \leftarrow \neg b.$
  $\text{Stable}(P_2) = \{\{a\}\}$
  $\text{PStable}(P_2) = \{\{a\}\}$
  $\text{MM}(P_2) = \{\{a\},\{b\}\}$

- P_3: $a \leftarrow \neg b.$
  $b \leftarrow \neg a.$
  $\text{Stable}(P_3) = \{\{a\},\{b\}\}$
  $\text{PStable}(P_3) = \{\{a\},\{b\}\}$
  $\text{MM}(P_3) = \{\{a\},\{b\}\}$

- P_4: $a \leftarrow \neg b.$
  $a \leftarrow b.$
  $b \leftarrow a.$
  $\text{Stable}(P_4) = \{\}$
  $\text{PStable}(P_4) = \{\{a,b\}\}$
  $\text{MM}(P_4) = \{\{a,b\}\}$

- P_5: $a \leftarrow \neg b.$
  $b \leftarrow \neg c.$
  $c \leftarrow \neg a.$
  $\text{Stable}(P_5) = \{\}$
  $\text{PStable}(P_5) = \{\}$
  $\text{MM}(P_5) = \{\{a,b\},\{a,c\},\{b,c\}\}$

Note that P_5 does not have pstable models.

8.8 Relating minimal, stable and pstable models

Now we turn again our attention to an interesting topic, the semantics of minimal models, that is also related to the X-stable semantics that we have proposed. In one of our previous works (Osorio et al., 2004c) characterization of minimal models in terms of provability in classical logic was given as follows:

Lemma 8.1. *(Osorio et al., 2004c) Let P be any theory, and M a set of atoms.* $P \cup \neg\widetilde{M} \Vdash_{\text{C}} M$ *if and only if M is a minimal model of P.*

Note that the proof of this lemma is done for any arbitrary theory. For our current purposes, however, we can restrict ourselves to the class of normal program to provide the following theorem as a straightforward consequence of this lemma.

Theorem 8.4. *Let P be a normal program, and M a set of atoms. If M is a pstable model of P then M is a minimal model of P.*

Proof. Let M be a pstable model of P. Then, by definition, $RED(P,M) \Vdash_C M$. Hence $RED(P,M) \cup \neg\widetilde{M} \Vdash_C M$. Thus $P \cup \neg\widetilde{M} \Vdash_C M$ and, by Lemma 8.1, M is a minimal model of P. □

An important remark is that, even if every X-stable model is minimal, the converse does not hold. This is shown by the second program in Example 8.8.

Definition 8.8. A normal clause is called *definite* if it does not contain negation. Given a program P, we will denote by $DEF(P)$ the set formed by all definite clauses in P.

Theorem 8.5. *Let P be a normal program and M a set of atoms. If M is a stable model of P then M is a pstable model of P.*

Proof. M is a stable model of P iff, by definition, M is a minimal model of the program $DEF(RED(P,M))$ (note that DEF deletes rules and RED negative literals as in conditions (i) and (ii) of Definition 8.2 respectively). Since $DEF(RED(P,M))$ is a definite program, then by Theorem 6.2 in (Lloyd, 1987), we have that $M = \{x \in \mathscr{L}_P | DEF(RED(P,M)) \vdash_C x\}$. Thus:

(a) M is a model of $RED(P)$

(b) $RED(P,M) \vdash_C M$ by monotonicity, since $DEF(RED(P,M))$ is a subset of $RED(P,M)$.

So $RED(P,M) \Vdash M$. Hence M is a X is a pstable model of P. □

Again, note that the converse of Theorem 8.5 is false. The counterexample is the first program in Example 8.8

CHAPTER 9

INVARIANCE OF WEAK COMPLETIONS

The lesson that we have learned from the stable model semantics is that weak completions are interesting to study. In this section we show that, for normal logic programs, the weak completions of a large class of interesting logics are equivalent. First in section 9.1 we introduce several definitions that will be later used in section 9.2 to prove one of the main contributions of this paper.

Theorem 9.1 at the end of this section is the main contribution of the paper, since it gives us a large class of logics in which semantics based on weak completions agree for normal programs.

9.1 Elementary multivalued logics

The following definition extracts some of the abstract similarities between several of the multivalued logics presented in this paper, and that will enable to carry out the proofs of later results.

Definition 9.1. A multivalued logic E is *elementary* if it there are three special elements **0**, **1** and **t** in its domain that satisfy the following properties:

- $\mathbf{t}$ is a designated value, and $\mathbf{0}$ is not.[1]
- The value assigned to $\mathbf{1} \rightarrow \mathbf{0}$ is not a designated value.
- The fragment $\{\mathbf{0}, \mathbf{t}\}$ coincides with classical logic (for $\wedge, \rightarrow, \neg$).
- The fragment $\{\mathbf{1}, \mathbf{t}\}$ is closed with respect to the connectives $\wedge$ and $\rightarrow$.
- The values assigned to the negation of $\mathbf{0}$ and $\mathbf{1}$ lie in the set $\{\mathbf{1}, \mathbf{t}\}$.
- The logic lies between the C_ω and C logics, i.e. $\mathrm{C}_\omega \subseteq E \subseteq \mathrm{C}$.

Note that G'_3, Pac and $\mathscr{P}$-$\mathscr{FOUR}$ are all elementary logics. The following Lemmas 9.1, and 9.2 apply, in particular, to these three elementary logics. Later, in section 9.2, we present our main results that apply more generally to logics between C_ω and any elementary logic.

Definition 9.2. Given an elementary logic E, an interpretation I is said to be *definite* if it maps atoms only to the elements $\mathbf{0}$ or $\mathbf{t}$.

Lemma 9.1. *Let P be a normal program and let M be a set of atoms such that every atoms that appears in a negated literal in P is also contained in M. Also let E be an elementary multivalued logic. If there is a definite interpretation I such that $I(P)$ is designated (w.r.t. E), then there is an interpretation I' such that*

1. $I'(x) = I(x)$ *for every* $x \in M$,
2. *if* $I(x) = \mathbf{0}$ *then* $I'(x) = \mathbf{0}$,
3. $I'(\neg x) \in \{\mathbf{1}, \mathbf{t}\}$ *for every* $x \notin M$,
4. $I'(P) \in \{\mathbf{1}, \mathbf{t}\}$.

[1] Note that there might be more designated and non-designated values; these are just the minimum requirements. In particular the definition does not impose any restriction on the designated status of $\mathbf{1}$.

Proof. Define I' as follows:

$$I'(x) = \begin{cases} I(x) & \text{if } x \in M, \\ \mathbf{0} & \text{if } x \notin M \text{ and } I(x) = \mathbf{0}, \\ \mathbf{1} & \text{otherwise.} \end{cases}$$

Items 1 and 2 follow immediately by construction of I'. To prove item 3 take $x \notin M$, then it follows that $I'(x) \in \{\mathbf{0}, \mathbf{1}\}$ and, by definition of elementary logics, $I'(\neg x) \in \{\mathbf{1}, \mathbf{t}\}$.

Now, to prove item 4, since I is definite and $I(P)$ is designated, it must be the case that $I(P) = \mathbf{t}$. Since, again by the definition of elementary logics, the definite fragment of E coincides with classical logic, we have that $I(p \leftarrow \mathscr{B}^+ \wedge \neg\mathscr{B}^-) = \mathbf{t}$ for every rule $p \leftarrow \mathscr{B}^+ \wedge \neg\mathscr{B}^-$ in P. We now have two cases:

1. $I(\mathscr{B}^+ \wedge \neg\mathscr{B}^-) = \mathbf{0}$. The proof furthermore splits in the following two cases:

   (a) There is an atom $x \in \mathscr{B}^+$ with $I(x) = \mathbf{0}$. By item 2 of the statement of the lemma, it follows that also $I'(x) = \mathbf{0}$.

   (b) There is an atom $x \in \mathscr{B}^-$ with $I(\neg x) = \mathbf{0}$ or, equivalently, $I(x) = \mathbf{t}$. By hypothesis, and since $x \in \mathscr{B}^-$, it must be the case that $x \in M$; and therefore $I'(x) = \mathbf{t}$, yielding $I'(\neg x) = \mathbf{0}$.

   In either case we have shown that $I'(\mathscr{B}^+ \wedge \neg\mathscr{B}^-) = \mathbf{0}$ and $I(p \leftarrow \mathscr{B}^+ \wedge \neg\mathscr{B}^-) = \mathbf{t}$.

2. $I(p) = \mathbf{t}$, $I(\mathscr{B}^+) = \mathbf{t}$ and $I(\neg\mathscr{B}^-) = \mathbf{t}$. Following a similar argument to the one used in 1.(b), it can be shown that $I'(\neg\mathscr{B}^-) = \mathbf{t}$. Then, by definition of I' and since I is definite, it must be the case that both $I'(p) \in \{\mathbf{1}, \mathbf{t}\}$ and $I'(\mathscr{B}^+) \in \{\mathbf{1}, \mathbf{t}\}$. Finally, by closure of elementary logics in the fragment $\{\mathbf{1}, \mathbf{t}\}$, we have that also $I'(p \leftarrow \mathscr{B}^+ \wedge \neg\mathscr{B}^-) \in \{\mathbf{1}, \mathbf{t}\}$.

We have shown that every rule in the program P must evaluate to either $\mathbf{1}$ or $\mathbf{t}$. Using the closure of the $\{\mathbf{1}, \mathbf{t}\}$ fragment for a final time, we can conclude that $I'(P) \in \{\mathbf{1}, \mathbf{t}\}$. □

Lemma 9.2. *Let P be a normal program, let M be a set of atoms, and let E be an elementary logic. If $P \cup \neg\widetilde{M} \vdash_E M$ then $RED(P,M) \vdash_E M$.*

Proof. We will prove the contrapositive of the statement, i.e. $RED(P,M) \nvdash_E M$ implies $P \cup \neg\widetilde{M} \nvdash_E M$. Take an atom $x \in M$ such that $RED(P,M) \nvdash_E x$. It follows, since $\mathrm{C}_\omega \subseteq E \subseteq \mathrm{C}$ and applying Corollary 7.1, that $RED(P,M) \nvdash_{\mathrm{C}} x$. Then, there must be a classical (two-valued) interpretation I such that $I(RED(P,M)) = 1$ and $I(x) = 0$. The interpretation I can then be lifted to the domain of E to obtain an interpretation J with $J(RED(P,M)) = \mathbf{t}$ and $J(x) = \mathbf{0}$. By Lemma 9.1 there is an interpretation J' such that $J'(x) = \mathbf{0}$, $J'(y) \in \{\mathbf{1},\mathbf{t}\}$ for every $y \notin M$, and $J'(RED(P,M)) \in \{\mathbf{1},\mathbf{t}\}$. From the last two results it is easy to verify that also $J'(P \cup \neg\widetilde{M}) \in \{\mathbf{1},\mathbf{t}\}$. Finally, from the definition of elementary logics, we have that $J'(P \cup \neg\widetilde{M} \to x)$ is not a designated value (since either $\mathbf{1} \to \mathbf{0}$ is not a designated value, or $\mathbf{t} \to \mathbf{0}$ evaluates $\mathbf{0}$ which is also not designated) so that, finally, $P \cup \neg\widetilde{M} \nvdash_E x$. □

The converse of Lemma 9.2 is also true, but it will follow as a simple corollary of the more general Lemma 9.3.

9.2 Relating X-stable and pstable models

In the previous subsection we have encapsulated several general properties shared by all the logics that we consider. Having these tools, we are now ready to present the main contribution of this paper. It states that, for a very large class of logics, the corresponding X-stable semantics (where X is any of our logics) are all equivalent to each other at least up to normal programs. Moreover we show that such X-Stable models can be expressed by a simple fixed-point operator and using classical logic. The precise statement is given in Theorem 9.1.

Lemma 9.3. *Let P be a normal program, let M be a set of atoms, and let E be an elementary logic. Furthermore, let X be an any logic with $\mathrm{C}_\omega \subseteq X \subseteq E$. Then we have that $RED(P,M) \vdash_X M$ iff $P \cup \neg\widetilde{M} \vdash_X M$.*

Proof. We will first show that $RED(P,M) \vdash_X M$ implies $P \cup \neg\widetilde{M} \vdash_X M$. Since $RED(P,M) \vdash_X M$ and by Corollary 7.1, $RED(P,M) \vdash_{C_\omega} M$. Also note that any formula in $RED(P,M)$ can be derived in C_ω from the set $P \cup \neg\widetilde{M}$ (follows by axioms **Pos1**–**Pos2**), i.e. $P \cup \neg\widetilde{M} \vdash_{C_\omega} RED(P,M)$. Therefore it easily follows, by transitivity of the *proves* relation that $P \cup \neg\widetilde{M} \vdash_{C_\omega} M$. Since X is stronger than C_ω, we can finally conclude that $P \cup \neg\widetilde{M} \vdash_X M$.

For the other implication we start with $P \cup \neg\widetilde{M} \vdash_X M$. Since E is stronger than X then we also have that $P \cup \neg\widetilde{M} \vdash_E M$. By Lemma 9.2 it then follows that $RED(P,M) \vdash_E M$ and finally, by Corollary 7.1, $RED(P,M) \vdash_X M$. □

Lemma 9.4. *Let P be a normal program and let M be a set of atoms. M is a classical model of $P \cup \neg\widetilde{M}$ iff it is a classical model of $RED(P,M)$.*

Proof. Straightforward. □

Lemma 9.5. *Let P be a normal program, let M be a set of atoms, and let E be an elementary logic. Furthermore, let X be any logic with $C_\omega \subseteq X \subseteq E$. M is an X-stable model of P iff M is a pstable model of P.*

Proof. M is a X-stable model of P iff, by definition, $P \cup \neg\widetilde{M} \Vdash_X M$ iff, by Lemmas 9.3 and 9.4, $RED(P,M) \Vdash_X M$ iff, by Corollary 7.1, $RED(P,M) \Vdash_C M$ iff, by definition, M is a pstable model of P. □

Theorem 9.1. *Let P be a normal program, let M be a set of atoms, and let X be a logic such that*

- *X is any logic $C_\omega \subseteq X \subseteq$ Pac, or*
- *X is any logic $C_\omega \subseteq X \subseteq G'_3$, or*
- *X is any logic $C_\omega \subseteq X \subseteq \mathscr{P}$-$\mathscr{FOUR}$, or*
- *X is any modal logic $T \subseteq X \subseteq S5$.*

Then M is a X-stable model of P iff M is a pstable model of P.

Proof. The proof for the modal logics between T and S5 follow by Theorem 8.2. Any of the other logics satisfy the conditions of Lemma 9.5. □

CHAPTER 10

CONCLUSIONS

In this thesis we were able to show how the extensive available background of mathematical logic in general and modal logics in particular is quite useful to study and better understand logic programming and nonmonotonic reasoning. We also note how a very general framework, based on the notion of weak completions within the class of normal programs and parametrized with different logics, can be used to model nonmonotonic systems.

The coarse existing knowledge in the scope of the mathematical logic in general, modal and multivalued logics in particular, allows us to study and better understand logic programming. We find some properties shared by a large family of different paraconsistent logics stronger than C_ω, some results about expressiveness of a given logic in terms of another, and finally the relation between these logics (Chapter 5). Particularly we present a semantic defined over general theories which, within the class of normal programs, is invariant among the large collection of paraconsistent logics studied previously: the PStable semantics (our main result, Theorem 9.1). Studying the properties of this semantics we find out that it is between the stable and minimal models, (i.e. every stable model is a pstable model and every pstable model is a minimal model as well). It is also important to note that, since pstable models are defined in using a simple syntactical reduction and in terms of classical logic, it is quite straightforward to build simple prototypes to compute pstable models using a modern

classical satisfiability solver (e.g. Eén and Sörensson, 2005) as an inference back end.

Observe that we can not generalize our results to disjunctive programs such that all of our major results hold together, namely Theorems 8.4, 8.5, and 9.1. It is easy to verify that Theorems 8.4 and 9.1 will hold for disjunctive programs. The problem will occur with Theorem 8.5. The counter example is the simple program: $a \vee b$. This program does not have pstable models, but it does have stable models. Hence, dealing with disjunctive logic programs is an open topic for future research.

BIBLIOGRAPHY

Krzysztof R. Apt and Roland Bol. Logic programming and negation: A survey. *Journal of Logic Programming*, 19/20:9–71, 1994.

Arnon Avron. Natural 3-valued logics – characterization and proof theory. *The Journal of Symbolic Logic*, 56(1):276–294, 1991.

Arnon Avron. On the expressive power of three-valued and four-valued languages. *Journal of Logic and Computation*, 9(6):977–994, 1999.

Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge, 2003.

Chitta Baral, Jorge Lobo, and Jack Minker. Generalized Well-founded Semantics for Logic Programs. In M. E. Stickel, editor, *10th International Conference on Automated Deduction*, LNAI 449, pages 102–116, Berlin, July 1990. Springer.

N. D. Belnap. A useful four-valued logic. In J. Michael Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logics*, pages 8–37. D. Reidel, 1977.

Veronica Borja. Lógicas Super-S5 y $\mathscr{FOUR}$. Thesis, Benemérita Universidad Autónoma de Puebla, February 2004.

Gerd Brewka, Jürgen Dix, and Kurt Konolige. *Nonmonotonic Reasoning: An Overview*. CSLI Lecture Notes 73. CSLI Publications, Stanford, CA, 1997.

W. A. Carnielli and J. Marcos. A taxonomy of **C**-Systems. In *Paraconsistency: The Logical Way to the Inconsistent, Proceedings of the Second World Congress on Paraconsistency (WCP 2000)*, number 228 in Lecture Notes in Pure and Applied Mathematics, pages 1–94. Marcel Dekker, Inc., 2002.

Walter A. Carnielli and Marcelo E. Coniglio. *Paraconsistency*. Marcel Dekker, April 2002. ISBN 0-824-70805-9.

N. C. A. daCosta. *On the theory of inconsistent formal systems (in Portuguese)*. PhD thesis, Curitiva:Editora UFPR, Brazil, 1963.

Marc Denecker, Nikolay Pelov, and Maurice Bruynooghe. Ultimate well-founded and stable semantics for logic programs with aggregates. In *Logic Programming. 17th International Conference, ICLP 2001*, number 2237 in Lecture Notes in Computer Science, pages 212–226. Springer, December 2001.

Jürgen Dix. Cumulativity and Rationality in Semantics of Normal Logic Programs. In J. Dix, K. P. Jantke, and P. H. Schmitt, editors, *Proceedings of the first Workshop on Nonmonotonic and Inductive Logic 1990 in Karlsruhe*, LNCS 543, pages 13–37, Berlin, October 1991. Springer.

Jürgen Dix. A classification theory of semantics of normal logic programs: I. strong properties. *Fundamental Informaticae*, 22(3):227–255, 1995a.

Jürgen Dix. A classification theory of semantics of normal logic programs: II. weak properties. *Fundamental Informaticae*, 22(3):257–288, 1995b.

Jürgen Dix, Mauricio Osorio, and Claudia Zepeda. A general theory of confluent rewriting systems for logic programming and its applications. *Annals of Pure and Applied Logic*, 108(1–3):153–188, 2001.

Francesco M. Donini, Daniele Nardi, and Riccardo Rosati. Ground nonmonotonic modal logics. *Logic and Computation*, 7(4), 1997.

Niklas Eén and Niklas Sörensson. MINISAT — a SAT solver with conflict-clause minimization. Poster presented at the SAT 2005 Competition, 2005.

Melvin C. Fitting. Bilattices and the semantics of logic programming. *Journal of Logic Programming*, 11:91–116, 1991.

Josep Maria Font and Miquel Rius. An absract logic approach to tetravalent modal logics. *Journal of Symbolic Logic*, 65(2):481–518, 2000.

Dov Gabbay and John Woods, editors. *Handbook of the history of logic, volume 1: Greek, Indian and Arabic logic*. Elsevier, 2004a. ISBN 0-444-50466-4.

Dov Gabbay and John Woods, editors. *Handbook of the history of logic, volume 3: The Rise of Modern Logic I: Leibniz to Frege*. Elsevier, 2004b. ISBN 0-444-51611-5.

Michael Gelfond. On stratified auto-epistemic theories. In *Proceedings of AAAI*, pages 207–211. Morgan Kaufman, 1987.

Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

Matthew Ginsberg. Multivalued logics. *Computational Intelligence*, 4(3), 1988.

Robert Goldblatt. *Logics of Time and Computation*. Number 7 in CSLI Lecture Notes. Stanford, second edition, 1992.

Vincent F. Hendricks, Stig Andur Pedersen, and Klaus Frovin Jorgensen, editors. *Proof Theory: History and Philosophical Significance*. Springer, September 2000. ISBN 0-792-36544-5.

Kurt Konolige. On the relation between default and autoepistemic logic. In M. L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*, pages 195–226. Kaufmann, Los Altos, CA, 1987.

John W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, second edition, 1987.

Wiktor Marek and Mirek Truszczyński. *Nonmonotonic Logics; Context-Dependent Reasoning*. Springer, Berlin, first edition, 1993.

Drew McDermott. Nonmonotonic logic II: Nonmonotonic modal theories. *ACM Transactions on Computer Systems*, 29:33–57, 1982.

Drew McDermott and Jon Doyle. Non-monotonic logic I. *Artificial Intelligence*, 13:41–72, 1980.

Elliott Mendelson. *Introduction to Mathematical Logic*. Wadsworth, Belmont, CA, third edition, 1987.

Pierluigi Minari. A note on Łukasiewicz's three-valued logic. *Annali del Dipartimento di Filosofia dell'Università di Firenze*, pages 163–190, 2003.

R. C. Moore. Autoepistemic logic. In P. Smets, E. H. Mamdani, D. Dubois, and H. Prade, editors, *Non-Standard Logics for Automated Reasoning*. Academic Press, 1988.

Mauricio Osorio and Juan Antonio Navarro. Modal logic $S5_2$ and FOUR (abstract). In *2003 Annual Meeting of the Association for Symbolic Logic*, Chicago, June 2003.

Mauricio Osorio, Juan Antonio Navarro, and José Arrazola. A logical approach for A-Prolog. In Ruy de Queiroz, Luiz Carlos Pereira, and Edward Hermann Haeusler, editors, *9th Workshop on Logic, Language, Information and Computation (WoLLIC)*, volume 67 of *Electronic Notes in Theoretical Computer Science*, pages 265–275, Rio de Janeiro, Brazil, 2002. Elsevier Science Publishers.

Mauricio Osorio, Verónica Borja, and José Arrazola. Closing the gap between the stable semantics and extensions of WFS. In *Mexican International Conference on Artificial Intelligence*, number 2972 in Lecture Notes in Computer Science, pages 202–211. Springer, 2004a.

Mauricio Osorio, Verónica Borja, and José Arrazola. Three valued logic of Łukasiewicz for modeling semantics of logic programs. In *Proceedings of IBERAMIA*, number 3315 in Lecture Notes in Computer Science, pages 343–352. Springer, 2004b.

Mauricio Osorio, Juan Antonio Navarro, and José Arrazola. Applications of intuitionistic logic in answer set programming. *Theory and Practice of Logic Programming*, 4(3):325–354, 2004c.

Mauricio Osorio, Juan Antonio Navarro, José Arrazola, and Verónica Borja. Ground non-monotonic modal logic S5: New results. *Journal of Logic and Computation*, 15(5):787–813, 2005.

David Pearce. Stable inference as intuitionistic validity. *Logic Programming*, 38:79–91, 1999.

Luís Moniz Pereira and Alexandre Miguel Pinto. Revised stable models - a new semantics for logic programs. In *Convegno Italiano di Logica Computazionale (CILC'04)*, Parma, Italy, July 2004.

G. Peterson. *G. W. Leibniz, Logical Papers*. Clarendon Press, Oxford, 1966.

Eric Schechter. *Classical and Nonclassical Logics: An Introduction to the Mathematics of Propositions*. Princeton University Press, 2005.

John S. Schlipf. Formalizing a logic for logic programming. *Annals of Mathematics and Artificial Intelligence*, 5(2–4):279–302, 1992.

Grigori Schwarz. Autoepistemic logic of knowledge. In *Proceedings of LPNMR*, pages 260–274. LNAI, 1991.

Schiller Joe Scroggs. Extensions of the Lewis system S5. *Journal of Symbolic Logic*, 16(2): 112–120, 1951.

Allen van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38:620–650, 1991.

APPENDIX A

K-BASIC FORMULAS IN REFLEXIVE FRAMES

Definition A.1. Let $\mathscr{M} = (S,R,V)$ be a model based on a reflexive frame $\mathscr{F}$, and let s be a fixed point in S. We define $\mathscr{M}'$ as the model based on $\mathscr{U}_2$, the universal frame with two points, with the valuation $V' \colon \mathscr{L} \to 2^{\{0,1\}}$ defined as follows:

$$V'(p) = \begin{cases} \{0,1\} & \text{if } s \in V(p) \text{ and } T \subseteq V(p) \\ \{0\} & \text{if } s \in V(p) \text{ and } T \not\subseteq V(p) \\ \{1\} & \text{if } s \notin V(p) \text{ and } T \cap V(p) \neq \emptyset \\ \emptyset & \text{if } s \notin V(p) \text{ and } T \cap V(p) = \emptyset \end{cases}$$

where $T = \{t \mid sRt \text{ and } s \neq t\}$.

Lemma A.1. *Let $\mathscr{M} = (S,R,V)$ be a model based on a reflexive frame $\mathscr{F}$, and let s be a fixed point in S. If A is a K-basic formula then*

$$\mathscr{M} \models_s A \text{ if and only if } \mathscr{M}' \models_0 A\,.$$

Proof. The proof is by structural induction over the construction of the K-basic formula A:

- $A = \bot$. The result follows trivially since neither $\mathscr{M} \models_s \bot$ nor $\mathscr{M}' \models_0 \bot$.

- $A = p$, for atomic p. It follows immediately by the definition of V' since

$$\mathscr{M} \models_s p \text{ iff } s \in V(p) \text{ iff } 0 \in V'(p) \text{ iff } \mathscr{M}' \models_0 p \,.$$

- $A = \neg p$, for atomic p. Analogously

$$\mathscr{M} \models_s \neg p \text{ iff } s \notin V(p) \text{ iff } 0 \notin V'(p) \text{ iff } \mathscr{M}' \models_0 \neg p \,.$$

- $A = \Box p$, for atomic p.

$$\begin{aligned}
\mathscr{M} \models_s \Box p &\text{ iff } (\forall t, sRt \text{ implies } \mathscr{M} \models_t p) && \text{by definition} \\
&\text{iff } \{t \mid sRt\} \subseteq V(p) \\
&\text{iff } V'(p) = \{0,1\} && \text{by definition of } V' \\
&\text{iff } \mathscr{M}' \models_0 p \text{ and } \mathscr{M}' \models_1 p \\
&\text{iff } \mathscr{M}' \models_0 \Box p \,.
\end{aligned}$$

- $A = \Box\neg p$, for atomic p.

$$\begin{aligned}
\mathscr{M} \models_s \Box\neg p &\text{ iff } (\forall t, sRt \text{ implies } t \notin V(p)) && \text{by definition} \\
&\text{iff } \{t \mid sRt\} \cap V(p) = \emptyset \\
&\text{iff } V'(p) = \emptyset && \text{by definition of } V' \\
&\text{iff } \mathscr{M}' \models_0 \neg p \text{ and } \mathscr{M}' \models_1 \neg p \\
&\text{iff } \mathscr{M}' \models_0 \Box\neg p \,.
\end{aligned}$$

- $A = B \to C$, for some K-basic formulas B, C.

$$\begin{aligned} \mathscr{M} \models_s B \to C \text{ iff } (\mathscr{M} \models_s B \text{ implies } \mathscr{M} \models_s C) && \text{by definition} \\ \text{iff } (\mathscr{M}_s \models_0 B \text{ implies } \mathscr{M}_s \models_0 C) && \text{by induction} \\ \text{iff } \mathscr{M}_s \models_0 B \to C && \text{by definition.} \end{aligned}$$

□

Proposition A.1. *Let $\mathscr{C}$ be a class of frames such that every frame in $\mathscr{C}$ is reflexive and the frame $\mathscr{U}_2 \in \mathscr{C}$. Then, for every K-basic formula A:*

$$\mathscr{C} \models A \quad \textit{iff} \quad \mathscr{U}_2 \models A\,.$$

Proof. The "only if" part is trivial since $\mathscr{U}_2 \in \mathscr{C}$. For the "if" part we will prove the contrapositive, i.e. $\mathscr{C} \not\models A$ implies $\mathscr{U}_2 \not\models A$. Assume $\mathscr{C} \not\models A$, then there is a frame $\mathscr{F} = (S,R) \in \mathscr{C}$, a model $\mathscr{M}$ based on $\mathscr{F}$, and a point $s \in S$ such that $\mathscr{M} \not\models_s A$. Let $\mathscr{M}'$ be constructed as in Definition A.1 so that, by Lemma A.1, $\mathscr{M}' \not\models_0 A$. Since the model $\mathscr{M}'$ is based on $\mathscr{U}_2$, this shows that $\mathscr{U}_2 \not\models A$. □

APPENDIX B

THE FRAME $\mathscr{U}_2$ AND $\mathscr{FOUR}$

Definition B.1. Let $g\colon \{0,1,2,3\} \to 2^{\{0,1\}}$ be the mapping:

$$g(0) = \emptyset \qquad g(2) = \{1\}$$
$$g(1) = \{0\} \qquad g(3) = \{0,1\}\ .$$

Lemma B.1. *Let $I\colon \mathscr{L} \to \{0,1,2,3\}$ be a $\mathscr{FOUR}$ valuation. The domain of the valuation is extended to arbitrary modal formulas as defined in section 5.2. Also let $\mathscr{M}$ be an arbitrary model for the frame $\mathscr{U}_2$ with some valuation V. If it turns out that $V(p) = g(I(p))$ for every propositional formula p then, for every formula A,*

$$\mathscr{M} \models_s A \qquad \textit{iff} \qquad s \in g(I(A))\ .$$

Proof. The proof is by induction on the length of the formula A, there are three cases:

- $A = p$, for atomic p. The is trivial since $\mathscr{M} \models_s p$ iff $s \in V(p) = g(I(p))$.
- $A = B \to C$. To simplify writing the proof let $W(F) = \{s \mid \mathscr{M} \models_s F\}$ for any formula F. Observe that the lemma's statement is equivalent to show that $W(A) = g(I(A))$. Also note that $g(I(\neg_{tr}\neg_{kn}F)) = S \setminus g(I(F))$ and $g(I(F \vee_{kn} G)) = g(I(F)) \cup g(I(G))$, where

$S = \{0,1\}$, and F, G are any pair of formulas. From this it follows:

$$\begin{aligned}
W(A \to B) &= \{s \mid \mathscr{M} \models_s B \to C\} \\
&= \{s \mid \mathscr{M} \models_s B \text{ implies } \mathscr{M} \models_s C\} && \text{by definition} \\
&= \{s \mid s \in W(B) \text{ implies } s \in W(C)\} \\
&= (S \setminus W(B)) \cup W(C) && \text{by simple set theory} \\
&= (S \setminus g(I(B))) \cup g(I(C)) && \text{by induction} \\
&= g(I(\neg_{tr}\neg_{kn} B \vee_{kn} C)) && \text{by previous notes} \\
&= g(I(B \to C)) && \text{by definition.}
\end{aligned}$$

- $A = \Box B$. The proof is as follows:

$$\begin{aligned}
\mathscr{M} \models_s \Box B \quad &\text{iff} \quad \forall t \in \{0,1\}, \mathscr{M} \models_t B && \text{by definition in } \mathscr{F}_2 \\
&\text{iff} \quad \forall t \in \{0,1\}, t \in g(I(B)) && \text{by induction} \\
&\text{iff} \quad g(I(B)) = \{0,1\} \\
&\text{iff} \quad g(I(\Box B)) = \{0,1\} && \text{by definition} \\
&\text{iff} \quad s \in g(I(\Box B)) \,.
\end{aligned}$$

For the last step observe that $s \in g(I(\Box B))$ implies that the set $g(I(\Box B))$ is not empty, and the only possible case is $g(I(\Box B)) = \{0,1\}$. The other implication is trivial.

□

APPENDIX C

INFERENCE RULE TRANSITIONS

Left side transitions

Rule Ca

$$\dfrac{\dfrac{x \leftarrow \alpha \wedge d \wedge c \qquad x \leftarrow \beta \wedge \neg c}{x \leftarrow \alpha \wedge \beta \wedge d}\,\text{Ca} \qquad d \leftarrow \gamma}{x \leftarrow \alpha \wedge \beta \wedge \gamma}\,\text{G}$$

$$\dfrac{\dfrac{x \leftarrow \alpha \wedge d \wedge c \qquad d \leftarrow \gamma}{x \leftarrow \alpha \wedge \gamma \wedge c}\,\text{G} \qquad x \leftarrow \beta \wedge \neg c}{x \leftarrow \alpha \wedge \beta \wedge \gamma}\,\text{Ca}$$

Rule R

$$\dfrac{\dfrac{x \leftarrow \alpha \wedge d \wedge c \qquad y \leftarrow \beta \wedge \neg c}{x \leftarrow \neg y \wedge \alpha \wedge \beta \wedge d}\,\text{R} \qquad d \leftarrow \gamma}{x \leftarrow \neg y \wedge \alpha \wedge \beta \wedge \gamma}\,\text{G}$$

$$\dfrac{\dfrac{x \leftarrow \alpha \wedge d \wedge c \qquad d \leftarrow \gamma}{x \leftarrow \alpha \wedge \gamma \wedge c}\,\text{G} \qquad y \leftarrow \beta \wedge \neg c}{x \leftarrow \neg y \wedge \alpha \wedge \beta \wedge \gamma}\,\text{R}$$

Right side transitions

Rule C

$$\dfrac{x \leftarrow \alpha \wedge d \qquad \dfrac{d \leftarrow \beta \wedge c \qquad d \leftarrow \gamma \wedge \neg c}{d \leftarrow \beta \wedge \gamma}\,\mathrm{Ca}}{x \leftarrow \alpha \wedge \beta \wedge \gamma}\,\mathrm{G}$$

$$\dfrac{\dfrac{x \leftarrow \alpha \wedge d \qquad d \leftarrow \beta \wedge c}{x \leftarrow \alpha \wedge \beta \wedge c}\,\mathrm{G} \qquad \dfrac{x \leftarrow \alpha \wedge d \qquad d \leftarrow \gamma \wedge \neg c}{x \leftarrow \alpha \wedge \gamma \wedge \neg c}\,\mathrm{G}}{x \leftarrow \alpha \wedge \beta \wedge \gamma}\,\mathrm{Ca}$$

Rule R

$$\dfrac{x \leftarrow \alpha \wedge d \qquad \dfrac{d \leftarrow \beta \wedge c \qquad y \leftarrow \gamma \wedge \neg c}{d \leftarrow \neg y \wedge \beta \wedge \gamma}\,\mathrm{R}}{x \leftarrow \neg y \wedge \alpha \wedge \beta \wedge \gamma}\,\mathrm{G}$$

$$\dfrac{\dfrac{x \leftarrow \alpha \wedge d \qquad d \leftarrow \beta \wedge c}{x \leftarrow \alpha \wedge \beta \wedge c}\,\mathrm{G} \qquad y \leftarrow \gamma \wedge \neg c}{x \leftarrow \neg y \wedge \alpha \wedge \beta \wedge \gamma}\,\mathrm{R}$$

APPENDIX D

PRELIMINARY RESULTS FOR DEDUCTIONS

Lemma D.1. *If* $P \vdash_{[\![CaR]\!]} \alpha$, $P \vdash_{[\![CaR]\!]} \beta$ *and* $\frac{\alpha \quad \beta}{\gamma}\,\mathsf{G}$ *then* $P \vdash_{[\![G-CaR]\!]} \gamma$. *Moreover it is possible to ensure that the number of times that* R *is used in* $P \vdash_{[\![G-CaR]\!]} \gamma$ *is the sum of the number of times that it is used in* $P \vdash_{[\![CaR]\!]} \alpha$ *and* $P \vdash_{[\![CaR]\!]} \beta$.

Proof. The proof is by induction on $n = s + t$ where s and t are the sizes of the proofs for α and β respectively. If $n = 0$ then both $\alpha \in P$ and $\beta \in P$, it follows immediately that $P \vdash_{[\![G]\!]} \gamma$ and, therefore, also $P \vdash_{[\![G-CaR]\!]} \gamma$.

Assume now that $n > 0$, then one of the two proofs if not empty. Let us suppose that the size of the proof $P \vdash_{[\![CaR]\!]} \alpha$ is non-zero, the other case is analogous. Then we have that $P \vdash_{[\![CaR]\!]} \delta$ and $P \vdash_{[\![CaR]\!]} \phi$ for a pair of clauses such that $\frac{\delta \quad \phi}{\alpha}\,\mathsf{Ca\ or\ R}$. The situation is illustrated for clarity.

$$\dfrac{\dfrac{\delta \qquad \phi}{\alpha}\,\text{Ca or R} \qquad \beta}{\gamma}\,\text{G}$$

When using left side transitions, see Appendix C, it is possible to construct a formula α' such that $\frac{\delta\ (\text{or}\ \phi) \quad \beta}{\alpha'}\,\mathsf{G}$. Applying the inductive hypothesis we obtain that $P \vdash_{[\![G-CaR]\!]} \alpha'$. Left side

transitions also show how $\frac{\alpha' \quad \phi \text{ (or } \delta)}{\gamma}$ Ca or R so that we may conclude $P \vdash_{[\![G-CaR]\!]} \gamma$.

If the size of the proof $P \vdash_{[\![CaR]\!]} \alpha$ is zero then the size of the proof $P \vdash_{[\![CaR]\!]} \beta$ must be non-zero. In this case we can follow a similar argument but using the right side transitions. Note that the right side transition for rule Ca shows how to provide two formulas, say α' and α'', with shorter proofs in which induction can be applied. Then γ is derived from α' and α'' and the same argument already provided follows. We can also see at Appendix C that in every case the number of times that R is used is preserved. In the application of rule Ca, one may think that the number of times that R is used could increase since the proof of $x \leftarrow \alpha \wedge d$ is required twice. However, is just the conclusion that we need to reuse and hence the number of times that R is used can be really preserved. □

Lemma D.2. *If $P \vdash_{[\![GCaR]\!]} \alpha$ then $P \vdash_{[\![G-CaR]\!]} \alpha$. Moreover it is possible to ensure that* R *is used the same number of times in $P \vdash_{[\![G-CaR]\!]} \alpha$ as in $P \vdash_{[\![GCaR]\!]} \alpha$.*

Proof. The proof is by induction on n, the size of the proof for α. If $n = 0$ then the result is immediate since $\alpha \in P$ and $P \vdash_{[\![G-CaR]\!]} \alpha$.

Assume now that $n > 0$. Then $P \vdash_{[\![GCaR]\!]} \beta$ and $P \vdash_{[\![GCaR]\!]} \gamma$ for a pair of clauses such that $\frac{\beta \quad \gamma}{\alpha}$ G, Ca or R. We can use the inductive hypothesis to obtain $P \vdash_{[\![G-CaR]\!]} \beta$ and $P \vdash_{[\![G-CaR]\!]} \gamma$ using R the same number of times as their corresponding cases. If the actual rule used is either Ca or R we can easily conclude since, by definition, $P \vdash_{[\![G-CaR]\!]} \alpha$.

On the other case, if the rule used is G, let $Q = \{\delta \mid P \vdash_{[\![G]\!]} \delta\}$. Observe that $Q \vdash_{[\![CaR]\!]} \beta$, $Q \vdash_{[\![CaR]\!]} \gamma$ and, using Lemma D.1, we obtain that $Q \vdash_{[\![G-CaR]\!]} \alpha$. It follows then, by the construction of Q, that $P \vdash_{[\![G-CaR]\!]} \alpha$. It is easy to see that the property about the use of R is also preserved. □

Lemma D.3. *Let P be a normal program and let P' be a program such that $(a \leftarrow A) \in P$ implies $(a \leftarrow A') \in P'$ for some $A' \subseteq A$. If $P \vdash_{[\![Ca]\!]} a \leftarrow A$ then $P' \vdash_{[\![Ca]\!]} a \leftarrow A'$ for some $A' \subseteq A$.*

Proof. The proof is by induction over n, the length of the proof of $P \vdash_{[\![Ca]\!]} a \leftarrow A$. If $n = 0$ then $(a \leftarrow A) \in P$ and, by hypothesis, there is a clause $(a \leftarrow A') \in P'$ with $A' \subseteq A$, it follows

that $P' \vdash_{[\![Ca]\!]} a \leftarrow A'$.

If $n > 0$ then we must have $A = B \cup C$ for a pair of sets B, C such that $P \vdash_{[\![Ca]\!]} a \leftarrow B \wedge y$ and $P \vdash_{[\![Ca]\!]} a \leftarrow C \wedge \neg y$. Using the inductive hypothesis we obtain $P' \vdash_{[\![Ca]\!]} a \leftarrow B'$ and $P' \vdash_{[\![Ca]\!]} a \leftarrow C'$ for some sets $B' \subseteq B \cup \{y\}$ and $C' \subseteq C \cup \{\neg y\}$ respectively. If it is the case that $y \notin B'$ (or $\neg y \notin C'$) we are done by taking $A' = B' \subseteq B \subseteq A$ (or $A' = C'$). If it is not the case then let $B'' = B' \setminus \{y\}$ and $C'' = C' \setminus \{\neg y\}$, we can use the Ca rule on previous premises to obtain $P' \vdash_{[\![Ca]\!]} B'' \cup C''$. In this final case letting $A' = B'' \cup C''$ we solve the lemma. □

For the following lemma we remind the reader about the notion $\bar{x}$ defined earlier in the paper.

Lemma D.4. *If $P, (a \leftarrow \bar{x}) \vdash_{[\![Ca]\!]} a \leftarrow A$ with $\bar{x} \notin A$ then $P \vdash_{[\![Ca]\!]} a \leftarrow A'$ with $A' \subseteq A \cup \{x\}$.*

Proof. The proof is by induction over n, the length of the proof. If $n = 0$ then, since $\bar{x} \notin A$, we obtain that $(a \leftarrow A) \in P$. It follows immediately that $P \vdash_{[\![Ca]\!]} a \leftarrow A$ and $A \subseteq A \cup \{x\}$.

If $n > 0$ then we must have $A = B \cup C$ for a pair of sets B, C such that $P, (a \leftarrow \bar{x}) \vdash_{[\![Ca]\!]} a \leftarrow B \wedge y$ and $P, (a \leftarrow \bar{x}) \vdash_{[\![Ca]\!]} a \leftarrow C \wedge \neg y$. Recall that $\bar{x} \notin A = B \cup C$ and, therefore, we have both $\bar{x} \notin B$ and $\bar{x} \notin C$. If $x = y$ we can apply the inductive hypothesis to the first subproof obtain $P \vdash_{[\![Ca]\!]} a \leftarrow B'$ for some $B' \subseteq B \cup \{x\} \subseteq A \cup \{x\}$ and, letting $A' = B'$, find a solution to the lemma conditions. A similar argument follows in case $x = \neg y$ using induction on the second subproof.

Otherwise, neither $x = y$ nor $x = \neg y$, we can apply the inductive hypothesis to both subproofs obtaining $P \vdash_{[\![Ca]\!]} a \leftarrow B'$ and $P \vdash_{[\![Ca]\!]} a \leftarrow C'$ for some sets $B' \subseteq B \cup \{x, y\}$ and $C' \subseteq C \cup \{x, \neg y\}$ respectively. If it is the case that $y \notin B'$ (or $\neg y \notin C'$) we are done by taking, again, $A' = B'$ (or $A' = C'$). If it is not the case then let $B'' = B' \setminus \{y\}$ and $C'' = C' \setminus \{\neg y\}$, we can use the Ca rule on previous premises to obtain $P \vdash_{[\![Ca]\!]} B'' \cup C''$. In this final case letting $A' = B'' \cup C''$ we solve the lemma. □

Corollary D.1. *If $P, (a \leftarrow \bar{x} \wedge B) \vdash_{[\![Ca]\!]} a$ then either $P \vdash_{[\![Ca]\!]} a$ or $P \vdash_{[\![Ca]\!]} a \leftarrow x$.*

Proof. Using Lemma D.3 we get that $P, (a \leftarrow \bar{x}) \vdash_{[\![Ca]\!]} a$ and then, from Lemma D.4, $P \vdash_{[\![Ca]\!]} a \leftarrow X$ for some $X \subseteq \{x\}$. So that either $X = \emptyset$ or $X = \{x\}$. □

Lemma D.5. *If* $\frac{\alpha \quad \beta}{\gamma}$ R *and* $P, \alpha, \gamma \vdash_{[\![Ca]\!]} a$ *then* $P, \alpha, \beta \vdash_{[\![GCa]\!]} a$.

Proof. Observe that, since only Ca rules are allowed in the proof of a then both γ and α must have the atom a in the head. It follows that α must have the form $a \leftarrow A \wedge c$, while $\beta = b \leftarrow B \wedge \neg c$ so that $\gamma = a \leftarrow \neg b \wedge A \wedge B$. Also define the formula γ' as $a \leftarrow A \wedge B$.

Using Corollary D.1 we get that either $P, \alpha \vdash_{[\![Ca]\!]} a$, in which case we are done, or $P, \alpha \vdash_{[\![Ca]\!]} a \leftarrow b$. In the second case observe that

$$\dfrac{(\alpha)\ a \leftarrow A, c \qquad \dfrac{a \leftarrow b \qquad (\beta)\ b \leftarrow B, \neg c}{a \leftarrow B, \neg c}\ \mathrm{G}}{(\gamma')\ a \leftarrow A, B}\ \mathrm{Ca}$$

so we have $P, \alpha, \beta \vdash_{[\![GCa]\!]} \gamma'$. But recall that, using Lemma D.3 on the original lemma's hypothesis, also $P, \alpha, \gamma' \vdash_{[\![Ca]\!]} a$. This way we can finally conclude that $P, \alpha, \beta \vdash_{[\![GCa]\!]} a$. □