Logics with Common Weak Completions

MAURICIO OSORIO GALINDO, *DECFI, Universidad de las Américas, Puebla.*
E-mail: mauricioj.osorio@udlap.mx

JUAN ANTONIO NAVARRO PÉREZ, *School of Computer Sciences, The University of Manchester, Manchester M13 9PL, UK.*

JOSÉ R. ARRAZOLA RAMÍREZ and VERÓNICA BORJA MACÍAS, *Mathematics Department, Benemérita Universidad Autónoma de Puebla, C.P. 72000, Mexico.*

Abstract

We introduce the notion of X-stable models parametrized by a given logic X. Such notion is based on a construction that we call weak completions: a set of atoms M is an X-stable model of a theory T if M is a model of T, in the sense of classical logic, and the weak completion of T (namely $T \cup \neg\widetilde{M}$) can prove, in the sense given by logic X, every atom in the set M. We prove that, for normal logic programs, the result obtained by these weak completions is invariant with respect to a large family of logics. Two kinds of logics are mainly considered: paraconsistent logics and normal modal logics. For modal logics we use a translation proposed by Gelfond that identifies $\neg\Box a$ with $\neg a$. As a consequence we prove that several semantics (recently introduced) for non-monotonic reasoning (NMR) are equivalent for normal programs. In addition, we show that such semantics can be characterized by a fixed-point operator. Also, as a side effect, we provide new results for the stable model semantics.

Keywords: Multivalued logic, paraconsistent logic, non-monotonic reasoning.

1 Introduction

The research community has long recognized the study of non-monotonic reasoning (NMR) as a promising approach to model features of commonsense reasoning. On the other hand, monotonic logics have been successfully applied as a basic building block in the formalization of non-monotonic reasoning. This article follows a line of research that analyses how several kinds of logics—such as multivalued, paraconsistent and modal logics—can be fitted into this approach.

The idea of using modal logics to formalize NMR can be traced back to McDermott and Doyle [20]. Subsequently, McDermott [19] attempted to define non-monotonic logics based on the standard T, S4 and S5 logics. But he observed that, unfortunately, the non-monotonic version of S5 collapses to ordinary logic S5. Moore [23] suggested the use of autoepistemic logic (AEL) as an alternative formalization of NMR to avoid the problems encountered with standard modal logics. Moore explains that the real problem with non-monotonic 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' [23]. Gelfond [11] also showed that the perfect models of stratified logic programs can be characterized in terms of extensions

Published online 27 August 2006 doi:10.1093/logcom/exl013

of the corresponding autoepistemic theory. His characterization is based on the interpretation of *not a* as $\neg\Box$a. In fact, Baral [3] explains that the definition of stable models by Gelfond and Lifschitz [12] 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 [31] 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 [3]. 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 [18], where it is possible to find a detailed discussion of the development of the field.

The term *grounded* in non-monotonic logics refers to the idea of enabling the agent to make only assumptions that are 'grounded' in the world's knowledge. According to Donini *et al*. [8] the notion of groundedness was actually introduced by Konolige [16]. It is worth mentioning 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'' [8]. Donini *et al*. [8] renewed interest in non-monotonic S5 (and other normal modal logics) by studying their grounded versions. They showed, in particular, that grounded non-monotonic S5 does not collapse with S5. We continue this line of research [29], 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 non-monotonic modal logics between T and S5 are equivalent [29]. Furthermore, we also show that these logics are equivalent to a non-monotonic logic that we constructed using the well-known $\mathcal{FOUR}$ bilattice [29]. We called this semantics 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 [29].

In this article, we show that for normal programs, GNM-S5 can be characterized by a fixed-point semantics similar to the definition of the stable model semantics.

Pearce [30] 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 study, we showed how to express G_3 in Łukasiewicz's 3-valued logic (L_3) [27]. 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 [27]. Based on weak completions over G'_3, we introduced a new semantics. We prove in this article 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 [1]. We prove in this

article 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 non-monotonic semantics that are closer to the direction of classical logic. Our article, in fact, presents some more results around the topics already mentioned with the aim to understand non-monotonicity in further detail. Although the original motivation started with modal logics, in this article we do not have to deal too much with them. This is because, following one of our results in Osorio *et al*. [29], it is possible to understand them via $\mathcal{FOUR}$ (of course, for *K*-basic formulas only). However, we refer the interested reader to the work of Goldblatt [15] for a comprehensive introduction to modal logics.

Overall, our results help to establish interesting connections between different non-monotonic formalisms, and suggest the possibility of a general and unifying approach to the study of NMR.

The structure of our article is as follows. Section 2 describes the general background of the study including a brief review of some logic systems that will be used throughout the article. In Section 3, we present the logics G_3 and G'_3 as particular cases of the 3-valued Łukasiewicz's logic. We also construct, in terms of the $\mathcal{FOUR}$ bilattice and an auxiliary modal logic $\mathcal{M}$-$\mathcal{FOUR}$, a multivalued paraconsistent logic that we call $\mathcal{P}$-$\mathcal{FOUR}$. These logics will prove to be very useful in later sections toward our main contributions. The section finishes with a brief summary of all the logics presented and some of the relations that hold between each other.

In Section 4, we first introduce some classes of logic programs and then define the semantics of stable models. We further give some notions about semantics defined in terms of weak completions (Definition 4.3). We describe in particular the semantics built on top of the logics G'_3 and $\mathcal{P}$-$\mathcal{FOUR}$ introduced earlier, including a short summary of our previous results about modal logics and $\mathcal{M}$-$\mathcal{FOUR}$ [29]. We then introduce the notion of pstable models based on a fixed-point operator (Definition 4.5). At the end of the section a few examples of the semantics introduced are shown and the relations between them are made explicit: we prove that every stable model is a pstable model, and that every pstable model is a minimal model.

In Section 5, we focus the discussion of the study toward proving the invariance of weak completions for the main logics considered in the paper. We start by introducing some inferences rules for normal programs that can be used to prove atoms. We also introduce the concept of elementary logics as a tool for proving our principal result, Theorem 5.1, which states that the weak completions of a large fragment of modal and paraconsistent logics are actually equivalent to the pstable model semantics. Finally, in Section 6 we present the conclusions of the article and some open problems.

This paper is a revised and extended version of an unpublished work written by the first author of the current paper.

2 Background

In this section, we first introduce the syntax of logic formulas considered in this article. Then, we present a few basic definitions of how logics can be built to interpret the meaning of such formulas in order to, finally, give a brief introduction to several of the logics that are relevant for the results of our later sections.

2.1 Syntax of formulas

We consider a formal (propositional) language built from: an enumerable set $\mathcal{L}$ of elements called *atoms* (denoted a, b, c, ...); the binary connectives $\wedge$ (*conjunction*), $\vee$ (*disjunction*) and $\rightarrow$ (*implication*); and the unary connective $\neg$ (*negation*). Formulas (denoted A, B, C, ...) are constructed as usual by combining these basic connectives together. We also use $A \leftrightarrow B$ to abbreviate $(A \rightarrow B) \wedge (B \rightarrow A)$ and, following the tradition in logic programing, $A \leftarrow B$ as an alternate way of writing $B \rightarrow A$. A *theory* is just a set of formulas and, in this article, we only consider finite theories. Moreover, if T is a theory, we use the notation $\mathcal{L}_T$ to stand for the set of atoms that occur in the theory T.

Note, however, that the logic connectives that we have just introduced have no meaning by themselves. 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 possible 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 subscript might be dropped if the relevant logic is clear by context, or if the formula is actually parametrized by an unspecified logic and is intended to be evaluated under several different semantics.

2.2 Logic systems

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 $A \rightarrow B$ are in the logic, then so also is B) 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 than or equal* to a logic Y if $X \subseteq Y$, similarly we say that X is *stronger than or equal* to Y if $Y \subseteq X$.

Given a set of atoms M and when a theory, or logic program, T is clear by context we use the symbol $\widetilde{M}$ to denote the set complement $\mathcal{L}_T \setminus M$. 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 specify the meaning of logic formulas or, in other words, 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 Section 2.3.

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 article as explained in Section 2.3.1, 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 further 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 of T if the induced interpretation evaluates T to true). Some of these notations are not standard in literature, they follow from our previous work [25, 28].

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 for a logic is by the use of truth values and interpretations. Multivalued logics generalize the idea of 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 $\mathcal{D}$, 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 $\mathcal{D}$ according to the particular definition of the logic.

An *interpretation* is a function $I\colon \mathcal{L} \rightarrow \mathcal{D}$ 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 $\mathcal{D}$, and then evaluating the resulting expression in terms of the connectives of the logic (which are defined over $\mathcal{D}$). A formula is said to be a *tautology* if, for every possible interpretation, the formula evaluates to a designated value. The simplest example of a multivalued logic is classical logic where: $\mathcal{D} = \{0, 1\}$, 1 is the unique designated value, and connectives are defined through the usual basic truth tables. Further examples will be given later in Section 2.3.

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 $x \in \mathcal{D}$, if there is a designated value $y \in \mathcal{D}$ 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 article, in a slight abuse of notation, we would not distinguish between these two situations writing $\vdash$ and using the term *theorems* in both cases.

2.3 Brief survey of logics

In this subsection, we will briefly introduce several logics that will be relevant for our purposes in this article. We will present a Hilbert style definition for most of them, together with a few notes and examples.

2.3.1 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 these axioms somewhat constraint the meaning of the $\rightarrow$, $\wedge$ and $\vee$ connectives to match our usual intuition. Positive logic however, as its name 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. 21). The theorem holds, in particular, for all the logics stronger than Pos summarized later in Figure 2.

2.3.2 The C_ω logic

The C_ω logic, the weakest paraconsistent logic due to daCosta [7], 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 consider in this article 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.

An anonymous referee suggested a comparison of C_ω with a logic F defined in [6], we also believe that it would be interesting to see how this logic fits into the proposed framework and leave this topic for our future work.

2.3.3 The Pac logic

The Pac logic was extensively studied and axiomatized by Avron [1], 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 truth 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 $\mathcal{D} = \{0, 1, 2\}$ 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 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 [1, first prop. in Section 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).

2.3.4 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 one to do proofs by contradiction but in some limited way; other constructions such as the law of excluded middle (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.

2.3.5 The logic of Here and There

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

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

Table 1. Truth tables of connectives in Pac

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

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

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 $\mathcal{D} = \{0, 1, \ldots, i-1\}$ 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
- $x \rightarrow y = i - 1$ if $x \leq y$ and y otherwise.

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

2.3.6 Classical logic

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

CL1 $(\neg a \rightarrow a) \rightarrow a$

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.

2.3.7 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' 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 have to first 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 $\mathcal{D} = \{0, 1, 2\}$, 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 $\bot$ and $\rightarrow$ as follows:

$$\begin{array}{cc} \sim A := A \rightarrow \bot & \top := \sim\bot \\ A \vee B := (A \rightarrow B) \rightarrow B & A \wedge B := \sim(\sim A \vee \sim B) \\ \Box A := \sim(A \rightarrow \sim A) & \Diamond A := \sim A \rightarrow A \end{array}$$

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 article. The truth tables of most connectives are shown in Table 2, the conjunction and disjunction connectives (not shown) coincide with the min and max functions respectively. Minari [22] studies a syntactic characterization of the modal content of $Ł_3$, and checks the behaviour of modal operators against some of the relevant modal principles. Minari also studied $Ł_3$'s axiomatization [22] as well as its relation with modal logics, particularly with S5.

3 Definition of new logics

In this section, we define a couple of multivalued logics, namely G'_3 and $\mathcal{P}$-$\mathcal{FOUR}$, that will be very useful to define the semantics of logic programs in later sections. These two logics are

Table 2. Truth tables of connectives in $Ł_3$

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

based on other well-known logic formalisms, namely Łukasiewicz's $Ł_3$ logic and Belnap's $\mathcal{FOUR}$ bilattice, and provide the basis to lay out the main contributions of this article.

3.1 Defining G_3 *and* G'_3 *via* $Ł_3$

Our main motivation in studying Łukasiewicz's $Ł_3$ logic is the fact that we found it possible 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.

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 3 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 2.3.5.

In Carnielli and Macros [5], G'_3 is introduced only to prove that $a \vee (a \rightarrow b)$ is not a theorem of C_ω. In particular, none of **Pac1**, **Pac2** or **Pac5** are theorems in G'_3. Nevertheless, axioms **Pac3** and **Pac4**, definitely are theorems in G'_3. It will be shown in Section 3.3 that the set of theorems of G'_3 is a superset of those in 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 the definition of $\neg G'_3$, with respect to the original modal language of $Ł_3$, is based on the translation proposed by Gelfond [11]. In later sections, we will apply the same translation to different modal logics.

Table 3. Truth tables of connectives in G_3 and G'_3

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

3.2 Defining a paraconsistent logic via $\mathcal{FOUR}$

This research work was started by the first authors of this article [24], and further developed in subsequent publications [29]. In these previous publications, a study of ground non-monotonic 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 $\mathcal{FOUR}$.

Belnap [4] introduced a logic for dealing in an useful way with inconsistent and incomplete information. This logic is based on a structure called $\mathcal{FOUR}$ 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 1.

(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 $\neg_{tr}$, and the meet and join operators denoted, respectively, as $\wedge_{tr}$ and $\vee_{tr}$.
(2) 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 [14] proposed algebraic structures called *bilattices* that generalize Belnap's $\mathcal{FOUR}$; his motivation for introducing bilattices was to provide a uniform approach for a diversity of applications in computer science. The logical role that $\mathcal{FOUR}$ has among bilattices is very similar to the one of two-valued algebra among Boolean Algebras [2].

We now define a semantics for modal propositional theories, which we call *modal* $\mathcal{FOUR}$ or just $\mathcal{M}$-$\mathcal{FOUR}$ for short, where the primitive modal connectives are given by:

$$\begin{aligned} \bot &:= \neg_{tr}\neg_{kn} a \wedge_{kn} a \qquad \text{for some atom } a \\ \Box A &:= A \wedge_{kn} \neg_{tr} A \\ A \rightarrow B &:= \neg_{tr}\neg_{kn} A \vee_{kn} B \end{aligned}$$

It is easy to verify that, under these previous definitions, $\bot = 0$ and the other two connectives have the truth tables shown in Table 4. Recall that there is no 'typical' implication connective in $\mathcal{FOUR}$, ours is an abbreviation in terms of other $\mathcal{FOUR}$ connectives. The *native negation* connective of $\mathcal{M}$-$\mathcal{FOUR}$ 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. $\wedge_{\mathcal{M}\text{-}\mathcal{FOUR}}$ is $\wedge_{kn}$.

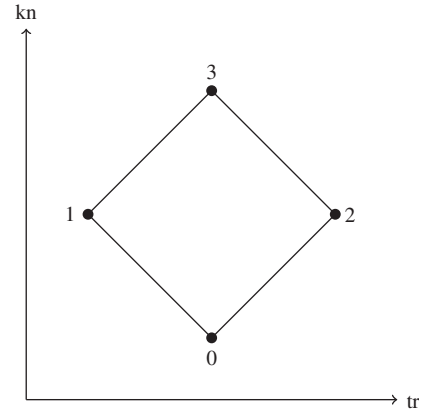

FIGURE 1. $\mathcal{FOUR}$ identified with 0, 1, 2 and 3

The native negation connective of $\mathcal{M}$-$\mathcal{FOUR}$ is used to construct the negation connective $\neg A := \sim \Box A$ using, one more time, the translation proposed by Gelfond [11]. The truth tables of both negation connectives are also given in Table 4. Particularly in later sections, 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 $\mathcal{M}$-$\mathcal{FOUR}$ for reasoning is to choose its designated values. As the tetravalent modal algebras (TMAs) do [10], 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 $\mathcal{P}$-$\mathcal{FOUR}$. Later, in Section 4.2.2, this logic is further utilized to define semantics of logic programs. $\mathcal{P}$-$\mathcal{FOUR}$, as well as G'_3 introduced earlier, shows some *paraconsistent* like behaviour 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, $\mathcal{P}$-$\mathcal{FOUR}$, G'_3) and call them *paraconsistent* logics. Some properties and relations of these logics are given in the next section.

3.3 Relations between logics

In this final subsection, 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 3.1

The relations between logics pictured in Figure 2 hold. An arrow between two logics denotes proper inclusion of the logics, while the absence of a path between them denotes incomparability.

Table 4. Truth tables of connectives in $\mathcal{M}$-$\mathcal{FOUR}$ and $\mathcal{P}$-$\mathcal{FOUR}$

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

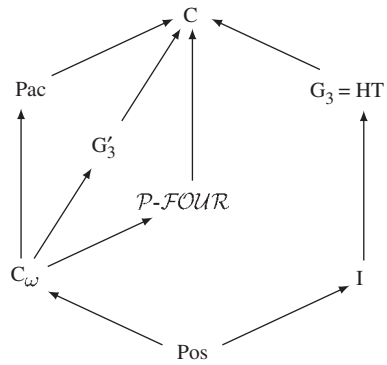

Figure 2. Relations between the logics considered

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.
- C_ω *is weaker than* Pac, G'_3 *and* $\mathcal{P}$-$\mathcal{FOUR}$. 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, $\mathcal{P}$-$\mathcal{FOUR}$ *and* Pac *are incomparable*. The formula $\neg a \wedge \neg\neg a \rightarrow b$ is a theorem in both G'_3 and $\mathcal{P}$-$\mathcal{FOUR}$ but not in Pac; while the formula $a \rightarrow \neg\neg a$ is a theorem in Pac but neither in G'_3 nor $\mathcal{P}$-$\mathcal{FOUR}$. On the other hand, the axiom **Pac1** is valid in $\mathcal{FOUR}$ and not in G'_3; while **Pac3** is valid in G'_3 and not in $\mathcal{FOUR}$. As a side effect this shows that C_ω is strictly weaker than all three Pac, G'_3 and $\mathcal{P}$-$\mathcal{FOUR}$.
- I *is strictly weaker than* G_3. It is a well-known result [see e.g. 21]. 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 $\mathcal{P}$-$\mathcal{FOUR}$, 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 than classical logic*. The previous item already gave examples of non-theorems for each logic which do are theorems in classical logic.

□

4 Semantics of 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 article 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 article, the conjunction connective is commutative; we will sometimes abbreviate an arbitrary normal clause by the expression

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

where $\mathcal{B}^+$ and $\mathcal{B}^-$ are supposed to be sets containing, respectively, the positive and negative atoms that occur in the *body* of the clause, i.e. $\mathcal{B}^+ = \{b_1, \ldots, b_n\}$ and $\mathcal{B}^- = \{b_{n+1}, \ldots, b_{n+m}\}$. In general, we might write in the body of a normal clause any conjunction of literals and/or sets of literals, e.g. $a \leftarrow b \wedge \mathcal{B}$, which is simply intended to mean the conjunction of all such literals. The main focus of the results in this article 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 NMR 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 cannot be empty.

Given a class of logic programs $\mathcal{C}$, a *semantic operator* is a function that assigns, to each program $P \in \mathcal{C}$, some subset of the power set of $\mathcal{L}_P$. 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 semantics of minimal models [17].

Definition 4.1
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. MM(P) is the set of minimal models of P.

4.1 The stable model semantics

The original definition of the stable model semantics for normal logic programs was given by Gelfond and Lifschitz [12]. A generalization for the class of disjunctive programs, given later also by Gelfond and Lifschitz [13], is the following:

DEFINITION 4.2
Let P be a disjunctive 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.

Note that P^M does not contain negation anymore. Then M is said to be a *stable model* of P if M is a minimal model of the reduced program P^M. We use Stable to denote the semantic operator of stable models.

While most of the content of this article deals with normal programs, the previous definition is given for the more general case of disjunctive programs. The only significative difference is that, for normal programs, the reduct P^M will always have exactly one minimal model [17], and M is a stable model if this minimal model happens to be M. The purpose of giving here the more general definition is 2-fold: first to be able to set our work within context with related work and, second, to be able to discuss possible extensions and generalizations for future work.

The following is a well-known result originally given by Pearce [30], 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 Section 2.2.

THEOREM 4.1
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 [e.g. 12]. We also point out that the definition of $\Vdash$ is a bit different from the one used in our previous work [25, 27, 28]. 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 4.1 both this definition and the one presented in Section 2.2 provide the same result. But this is no longer the case for the logics G'_3 and Pac studied in this article, so that the definition given in Section 2.2 must be used from now on.

4.2 The X-stable semantics

Theorem 4.1 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. 28], but are not considered in this article.

DEFINITION 4.3
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 $P \cup \neg\widetilde{M} \Vdash_X M$.

4.2.1 The G_3-stable semantics

Of particular interest to us is the G'_3-stable semantics,[2] which is the result of using the logic G'_3 in Definition 4.3. We have to emphasize that the reinterpretation of $\Vdash$ is relevant. Our original definition of G'_3-stable models [27] 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 Section 2.2 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 [12], this follows by Theorem 4.1 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 3. All this together suggests that Łukasiewicz's 3-valued logic provides a good framework for studying and defining NMR systems.

EXAMPLE 4.1
Consider the following logic program P:

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

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

4.2.2 The $\mathcal{P}$-$\mathcal{FOUR}$-stable semantics

The very same Definition 4.3 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 definition that is given in terms of the logic $\mathcal{P}$-$\mathcal{FOUR}$ of Section 3.2. This material was originally introduced by Osorio *et al*. [24, 26], but an extended version of such work is also available [29].

Most of our results in previous work were given for a large class of $\mathcal{M}$-$\mathcal{FOUR}$ formulas (i.e. formulas including modal connectives and native negation) that were called K-basic [29]. For our current purposes, in terms of the $\mathcal{P}$-$\mathcal{FOUR}$ fragment obtained after applying Gelfond's translation [11], 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 Goldblat [15] for a comprehensive introduction to modal logics.

THEOREM 4.2
All semantics of X-stable models, induced by any modal logic $\mathrm{KT} \subseteq X \subseteq \mathrm{S5}$*, as well as the* $\mathcal{P}$-$\mathcal{FOUR}$*-stable model semantics, are equivalent for the class of formulas in negation normal form.*

[2] Originally called $Ł_3$-WFS [27].

Note that, in order to apply Definition 4.3 to a modal logic X we first have to apply Gelfond's translation [11]. 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 [29] we also had a proof of the following very useful and important result.

COROLLARY 4.1. [29]
Let P be a normal logic program and let x be an atom. $P \vdash_C x$ if and only if $P \vdash_{\mathcal{P}\text{-}\mathcal{FOUR}} x$.

In view of the new results and logics considered in this article, this result will be further generalized in Section 5.1 to yield Proposition 5.1.

4.3 The pstable model semantics

The following definition is very similar to the well-known reduction due to Gelfond and Lifschitz [12] used in the definition of stable models in Section 4.1.

DEFINITION 4.4
Let P be a normal program and M a set of atoms. We define

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

EXAMPLE 4.2
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:

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

Also, following a similar approach to Gelfond and Lifschitz [12], the previous reduction is used to provide a semantics of logic programs using a fixed-point operator in terms of classical logic.

DEFINITION 4.5
Let P be a normal program, and M a set of atoms. We say that M is a *pstable model* of P if $RED(P, M) \Vdash_C M$. We use PStable to denote the semantic operator of pstable models.

The main result of Section 5 will show that, in fact, both the G'_3-stable and the $\mathcal{P}$-$\mathcal{FOUR}$-stable model semantics (as well as semantics based on 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 4.5, and the X-stable semantics in general, provide a solid base to study and propose new semantics for logic programs.

4.4 Examples

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

Example 4.3. The following are examples of normal programs together with their stable, pstable and minimal models as defined by Definitions 4.2, 4.5 and 4.1, respectively.

- $P_1 : a \leftarrow \neg a.$
  $\text{Stable}(P_1) = \{\}$
  $\text{PStable}(P_1) = \{\{a\}\}$
  $\text{MM}(P_1) = \{\{a\}\}$
- $P_2 : a \leftarrow \neg b.$
  $\text{Stable}(P_2) = \{\{a\}\}$
  $\text{PStable}(P_2) = \{\{a\}\}$
  $\text{MM}(P_2) = \{\{a\}, \{b\}\}$
- $P_3 : a \leftarrow \neg b.$
  $b \leftarrow \neg a.$
  $\text{Stable}(P_3) = \{\{a\}, \{b\}\}$
  $\text{PStable}(P_3) = \{\{a\}, \{b\}\}$
  $\text{MM}(P_3) = \{\{a\}, \{b\}\}$
- $P_4 : a \leftarrow \neg b.$
  $a \leftarrow \neg b.$
  $b \leftarrow \neg a.$
  $\text{Stable}(P_4) = \{\}$
  $\text{PStable}(P_4) = \{\{a, b\}\}$
  $\text{MM}(P_4) = \{\{a, b\}\}$
- $P_5 : a \leftarrow \neg b.$
  $b \leftarrow \neg c.$
  $c \leftarrow \neg a.$
  $\text{Stable}(P_5) = \{\}$
  $\text{PStable}(P_5) = \{\}$
  $\text{MM}(P_5) = \{\{a, b\}, \{a, c\}, \{b, c\}\}$

Note that P_5 does not have pstable models.

4.5 Relating minimal, stable and pstable models

Now we turn our attention again 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 [28] characterization of minimal models in terms of provability in classical logic was given as follows:

LEMMA 4.1 [28]
Let P be any theory, and M a set of atoms. $P \cup \neg\widetilde{M} \Vdash_C M$ 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 programs to provide the following theorem as a straightforward consequence of this lemma.

THEOREM 4.3
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 4.1, 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 4.3.

DEFINITION 4.6
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 4.4
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 4.2, respectively). Since $DEF(RED(P, M))$ is a definite program, then by Theorem 6.2 in [17], we have that $M = \{x \in \mathcal{L}_P \mid DEF(RED(P, M)) \vdash_C x\}$. 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 pstable model of P. □

Again, note that the converse of Theorem 4.4 is false. The counterexample is the first program in Example 4.3.

5 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 5.1 we present a natural deduction system for normal programs, then in Section 5.2 we introduce several definitions that will be later used in Section 5.3 to prove one of the main contributions of this article.

Theorem 5.1 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.

5.1 Classical deductions for normal programs

In this section, we briefly summarize some of our early work [29], where a natural deduction system that can be used to prove atoms, with respect to classical logic, in the context of normal logic programs was developed. Our main interest is Proposition 5.1 that will be used in later sections.

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:

$$\frac{A \vee c \quad B \vee \neg c}{A \vee B}\,\text{Resolution}$$

A formula F is derived in classical logic by a theory T, i.e. $T \vdash_C F$, if and only if the conjunctive normal form of $T \cup \{\neg F\}$ derives the empty formula (failure or contradiction). When we try to study resolution techniques, restricted to normal logic programs, then inference rules such as G, C and R appear.

DEFINITION 5.1 [29]
We define the following inference rules for normal logic programs,

$$\frac{a \leftarrow \mathcal{B}_1 \wedge d \quad d \leftarrow \mathcal{B}_2}{a \leftarrow \mathcal{B}_1 \wedge \mathcal{B}_2}\mathrm{G} \qquad \frac{a \leftarrow \mathcal{B}_1 \wedge c \quad a \leftarrow \mathcal{B}_2 \wedge \neg c}{a \leftarrow \mathcal{B}_1 \wedge \mathcal{B}_2}\mathrm{C}$$

$$\frac{a \leftarrow \mathcal{B}_1 \wedge c \quad b \leftarrow \mathcal{B}_2 \wedge \neg c}{a \leftarrow \neg b \wedge \mathcal{B}_1 \wedge \mathcal{B}_2}\mathrm{R}$$

where a, b, c and d are atoms, and the symbols $\mathcal{B}_1$, $\mathcal{B}_2$ represent arbitrary sets of literals. We furthermore introduce the notation $T \vdash_{[[R]]} F$, where T is a theory, F a formula, and R a list of names of inference rules; to denote the fact that the formula F can be derived from the formulas in T by applying the inference rules listed in R.

LEMMA 5.1 [29]
Let P be a normal program and let $a \leftarrow \mathcal{B}$ be a normal clause. It follows that

$$P \vdash_C a \leftarrow \mathcal{B} \quad \textit{if and only if} \quad P \cup T \vdash_{[[\mathrm{GCR}]]} a \leftarrow \mathcal{B},$$

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

LEMMA 5.2 [29]
Let P be a normal program and let a *be an atom. If $P \vdash_{[[\mathrm{GCR}]]} a$ then $P \vdash_{[[\mathrm{GC}]]} a$.*

If we consider only single atoms, Lemma 5.1 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 $T = \{a \leftarrow a \mid a \in \mathcal{L}_P\}$ and using the set of inference rules [GCR]. Lemma 5.2 allows us to reduce the set of rules required to [GC] only. This motivates the following result.

PROPOSITION 5.1
Let P be a normal program and let a be an atom. $P \vdash_C a$ if and only if $P \vdash_{C_\omega} a$.

PROOF. From Lemmas 5.1 and 5.2, if $P \vdash_C a$ then $P \cup T \vdash_{[[\mathrm{GC}]]} a$, where the set $T = \{a \rightarrow a \mid a \in \mathcal{L}_P\}$. 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 $P \vdash_{C_\omega} a$.

The converse follows immediately since C_ω is weaker than C (Theorem 3.1). □

COROLLARY 5.1
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, $P \vdash_{C_\omega} a$ and, by Proposition 5.1, $P \vdash_C a$ 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. □

5.2 Elementary multivalued logics

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

DEFINITION 5.2
A multivalued logic E is *elementary* if there are three special elements $\mathbf{0}$, $\mathbf{1}$ and $\mathbf{t}$ in its domain that satisfy the following properties:

- $\mathbf{t}$ is a designated value, and $\mathbf{0}$ is not.[3]
- 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 C_ω and C, i.e. $\mathrm{C}_\omega \subseteq E \subseteq \mathrm{C}$.

Note that G'_3, Pac and $\mathcal{P}$-$\mathcal{FOUR}$ are all elementary logics. The following Lemmas 5.3 and 5.4 apply, in particular, to these three elementary logics. Later, in Section 5.3, we present our main results that apply more generally to logics between C_ω and any elementary logic.

DEFINITION 5.3
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 5.3
Let P be a normal program and let M be a set of atoms such that every atom 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) $I'(P) \in \{\mathbf{1}, \mathbf{t}\}$.

PROOF. Define I' as follows:

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

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 $I(p \leftarrow \mathcal{B}^+ \wedge \neg\mathcal{B}^-) = \mathbf{t}$ for every rule $p \leftarrow \mathcal{B}^+ \wedge \neg\mathcal{B}^-$ in P. We now have two cases:

(1) $I(\mathcal{B}^+ \wedge \mathcal{B}^-) = \mathbf{0}$. The proof further splits in the following two cases:
   (a) There is an atom $x \in \mathcal{B}^+$ 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 $x \in \mathcal{B}^-$ with $I(\neg x) = \mathbf{0}$ or, equivalently, $I(x) = \mathbf{t}$. By hypothesis, and since $x \in \mathcal{B}^-$, it must be the case that $x \in M$; and therefore $I'(x) = \mathbf{t}$, yielding $I'(\neg x) = \mathbf{0}$.

[3] 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 $\mathbf{1}$.

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

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

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 5.4
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 $P \cup \widetilde{M} \nvdash_E M$. Take an atom $x \in M$ such that $RED(P, M) \nvdash_E x$. It follows, since $C_\omega \subseteq E \subseteq C$ and applying Corollary 5.1, that $RED(P, M) \nvdash_C x$. 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 5.3 there is an interpretation J' such that $J'(x) = 0$, $J'(y) \in \{\mathbf{1}, \mathbf{t}\}$ 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 $J'(P \cup \neg\widetilde{M} \rightarrow x)$ is not a designated value (since either $\mathbf{1} \rightarrow \mathbf{0}$ is not a designated value, or $\mathbf{t} \rightarrow \mathbf{0}$ evaluates $\mathbf{0}$ which is also not designated) so that, finally, $P \cup \neg\widetilde{M} \nvdash_E x$. □

The converse of Lemma 5.4 is also true, but it will follow as a simple corollary of the more general Lemma 5.5.

5.3 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 article. 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 5.1.

LEMMA 5.5
Let P be a normal program, let M be a set of atoms, and let E be an elementary logic. Furthermore, let X be an any logic with $C_\omega \subseteq X \subseteq E$. 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 5.1, $RED(P, M) \vdash_{C_\omega} M$. 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 5.4 it then follows that $RED(P, M) \vdash_E M$ and finally, by Corollary 5.1, $RED(P, M) \vdash_X M$. □

LEMMA 5.6
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 5.7
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 5.5 and 5.6, $RED(P, M) \Vdash_X M$ iff, by Corollary 5.1, $RED(P, M) \Vdash_C M$ iff, by definition, M is a pstable model of P.

THEOREM 5.1
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* $C_\omega \subseteq X \subseteq \mathcal{P}\text{-}\mathcal{FOUR}$, *or*
- X *is any modal logic* $T \subseteq X \subseteq S5$.

Similarly, let Y be any of such logics. Given a normal program P and a set of atoms M it follows:

(a) *M is a X-stable model of P iff M is a pstable model of P,*
(b) *M is a X-stable model of P iff M is a Y-stable model of P (invariance).*

PROOF. Item (a) follows from previous results: the proof for the modal logics between T and S5 follows by Theorem 4.2; any of the other logics satisfy the conditions of Lemma 5.7. Item (b) is a direct consequence of (a).

6 Conclusions

In this article we study the notions of weak completions within the class of normal programs using different logics. 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 (Section 3). Particularly, we present a semantics defined over general theories that, within the class of normal programs, is invariant among the large collection of paraconsistent logics studied previously: the PStable semantics (our main result, Theorem 5.1). 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. 9] as an inference back end.

Observe that not all of our results can be easily carried forward to the class of disjunctive logic programs. It is easy to verify that part (b) of Theorem 5.1 will hold for disjunctive programs, the problem with other results such as Theorem 4.4 is first to obtain a sensible

generalization of pstable models for disjunctive programs. An ingenuous extension of the reduction used in Definition 4.5 that does not modify the head of disjunctive clauses seems to have some difficulties. A simple program such as $a \vee b$, for instance, would not have any pstable models, while it does have stable models. We feel that more research needs to be carried out in order to obtain a suitable generalization and leave this topic for our future work.

References

[1] A. Avron. Natural 3-valued logics – characterization and proof theory. *The Journal of Symbolic Logic*, **56**, 276–294, 1991.

[2] A. Avron. On the expressive power of three-valued and four-valued languages. *Journal of Logic and Computation*, **9**, 977–994, 1999.

[3] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge, 2003.

[4] N. D. Belnap. A useful four-valued logic. In *Modern Uses of Multiple-Valued Logics*, J. M. Dunn and G. Epstein, eds, pp. 8–37. D. Reidel, Dordrecht, 1977.

[5] 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)*, *Lecture Notes in Pure and Applied Mathematics*, No. 228, pp. 1–94. Marcel Dekker, Inc., New York, 2002.

[6] G. Corsi. Weak logics with strict implication. *Zeitschr. Tift für Mathematisc he Logik und Grundlagen der Mathematik*, **33**, 389–406, 1987.

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

[8] F. M. Donini, D. Nardi, and R. Rosati. Ground nonmonotonic modal logics. *Journal of Logic and Computation*, **7**, 523–548, 1997.

[9] N. Eén and N. Sörensson. MiniSat — a SAT solver with confict-clause minimization. Poster presented at the SAT 2005 Competition, 2005.

[10] J. M. Font and M. Rius. An abstract logic approach to tetravalent modal logics. *Journal of Symbolic Logic*, **65**, 481–518, 2000.

[11] M. Gelfond. On stratified auto-epistemic theories. In *Proceedings of AAAI*, pp. 207–211. Morgan Kaufman, Los Altos, CA, 1987.

[12] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, eds, *Proceedings of the 5th Conference on Logic Programming*, pp. 1070–1080. MIT Press, Cambridge, MA, 1988.

[13] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, **9**, 365–385, 1991.

[14] M. Ginsberg. Multivalued logics. *Computational Intelligence*, **4**, 265–316, 1988.

[15] R. Goldblatt. *Logics of Time and Computation*. CSLI Lecture Notes No. 7, Stanford, 2nd edition, 1992.

[16] K. Konolige. On the relation between default and autoepistemic logic. In M. L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*, pp. 195–226. Morgan Kaufmann, Los Altos, CA, 1987.

[17] J. W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, 2nd edition, 1987.

[18] W. Marek and M. Truszczyński. *Nonmonotonic Logics; Context-Dependent Reasoning*. Springer, Berlin, 1st edition, 1993.

[19] D. McDermott. Nonmonotonic logic II: Nonmonotonic modal theories. *ACM Transactions on Computer Systems*, **29**, 33–57, 1982.
[20] D. McDermott and J. Doyle. Non-monotonic logic I. *Artificial Intelligence*, **13**, 41–72, 1980.
[21] E. Mendelson. *Introduction to Mathematical Logic*. Wadsworth, Belmont, CA, 3rd edition, 1987.
[22] P. Minari. A note on Łukasiewicz's three-valued logic. *Annali del Dipartimento di Filosofia dell'Università di Firenze*, pp. 163–190, 2003.
[23] R. C. Moore. Autoepistemic logic. In *Non-Standard Logics for Automated Reasoning*, P. Smets, E. H. Mamdani, D. Dubois, and H. Prade, eds, Academic Press, London, 1988.
[24] M. Osorio and J. A. Navarro. Modal logic $S5_2$ and FOUR (abstract). In *Proceedings of the 2003 Annual Meeting of the Association for Symbolic Logic*, Chicago, June 2003.
[25] M. Osorio, J. A. Navarro, and J. Arrazola. A logical approach for A-Prolog. In R. de Queiroz, L. C. Pereira, and E. H. Haeusler, eds, In *Proceedings of the 9th Workshop on Logic, Language, Information and Computation (WoLLIC)*, Electronic Notes in Theoretical Computer Science, Volume 67, pp. 265–275, Rio de Janeiro, Brazil, Amsterdam, 2002. Elsevier Science Publishers, Amsterdam.
[26] M. Osorio, V. Borja, and J. Arrazola. Closing the gap between the stable semantics and extensions of WFS. In *Proceedings of the Mexican International Conference on Artificial Intelligence*, Lecture Notes in Computer Science, No. 2972, pp. 202–211. Springer, Berlin, 2004.
[27] M. Osorio, V. Borja, and J. Arrazola. Three valued logic of Łukasiewicz for modeling semantics of logic programs. In *Proceedings of IBERAMIA*, Lecture Notes in Computer Science, No. 3315, pp. 343–352. Springer, Berlin, 2004.
[28] M. Osorio, J. A. Navarro, and J. Arrazola. Applications of intuitionistic logic in answer set programming. *Theory and Practice of Logic Programming*, **4**, 325–354, 2004.
[29] M. Osorio, J. A. Navarro, J. Arrazola, and V. Borja. Ground nonmonotonic modal logic S5: New results. *Journal of Logic and Computation*, **15**, 787–813, 2005.
[30] D. Pearce. Stable inference as intuitionistic validity. *Logic Programming*, **38**, 79–91, 1999.
[31] G. Schwarz. Autoepistemic logic of knowledge. In *Proceedings of LPNMR*, pp. 260–274, LNAI, 1991.

Received 4 January 2006