GLukG logic and its application for Non-Monotonic Reasoning

Mauricio Osorio Galindo,
Universidad de las Américas - Puebla,
Email: osoriomauri@gmail.com

Abstract. We present GLukG, a paraconsistent logic recently introduced. We discuss our motivation as well as interesting properties of this logic. We introduce a non-monotonic semantics based on GLukG.

1 Introduction

It is well known that monotonic logics can be used to define nonmonotonic semantics [8,7,5] and more recently in [4,17,15,16]. It is also well known that nonmonotonic reasoning (NMR) is useful to model intelligent agents. It has recently been shown that GLukG logic can be used to express interesting nonmonotonic semantics [16]. More generally, two major classes of logics that can be successfully used to model nonmonotonic reasoning are (constructive) intermediate logics and paraconsistent logics [17,15,16]. The most well known semantics for NMR is the stable semantics [6]. This semantics provides a fairly general framework for representing and reasoning with incomplete information. There are programs such as P_1:
$a \leftarrow \neg b$.
$a \leftarrow b$.
$b \leftarrow a$:

that do not have stable models. This is not exactly a problem of the stable semantics but sometimes we have applications where we need a behavior closer to classical logic such as in argumentation [11]. The stable semantics has been extended to include default negation in the head of the programs (see for instance [19]) and in this case one can construct inconsistent programs such as P_2:
$\neg a$.
a.
$b \leftarrow a$.

In this case the stable semantics gives no stable models. The sceptical version of the stable semantics infers every formula in this example. This is also the case for most of the traditional logic programming semantics in which inconsistency might spoil the whole program. Most logics (at least classical logic and all constructive intermediate logics) share the theorem $(a \wedge \neg a) \rightarrow b$, meaning that in the presence of an inconsistency $(a \wedge \neg a)$ then one prove anything (such as the

unrelated formula b in this example). Paraconsistent logics reject this principle, the do no not like to be destructive in the presence of inconsistent information. To avoid this problem the stable semantics was generalized by [18] to deal with inconsistent programs. However this approach only allows disjunctive programs. In the approach by [18] it is also not clear what is the proof theory that supports their proposal.

We propose an approach that can be used by any paraconsistent logic stronger or equal to C_ω, the weakest paraconsistent logic introduced by da Costa himself. We select however the paraconsistent logic GLukG because it is 3-valued and is very expressive. GLukG can define the same class of functions as Łukasiewicz 3-valued logic. In GLukG we can express very directly the strongest intermediate logic (also known as the Gödel's 3-valued logic G_3). This is important because G_3 is adequate to express the stable model semantics. However if we use the original C_ω we obtain similar interesting results. Moreover, GLukG can directly express the GNM-S5 semantics [16] for the class of disjunctive programs. Our approach is not restricted to a particular fragment of propositional logics, it is defined for any type of propositional theory.

The structure of our paper is as follows. Section 2 describes the general background of the paper including the definition of GLukG logic. In Section 3 we present our motivation of GLukG logic. This a contribution of this paper, because is the first time that it is explained how and why we came out with GLukG logic. In Section 4 we present an axiomatization of GLukG logic.

In Section 5 we discuss two main properties of GLukG logic that makes it interesting. In Section 6 we present our main contribution, namely the relation of GLukG logic and nonmonotonic reasoning. We also introduce a nonmonotonic semantics that can deal with inconsistent programs, that is our final contribution to this paper. Finally, in Section 7 we present the conclusions of the paper.

We assume that the reader has some familiarity with basic logic such as chapter one in [9].

2 Background

We first introduce the syntax of logic formulas considered in this paper. 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 with help of parentheses. We also use $A \leftrightarrow B$ to abbreviate $(A \rightarrow B) \wedge (B \rightarrow A)$ and $\alpha \leftarrow \beta$ to abbreviate $\beta \rightarrow \alpha$. It

is useful to agree on some conventions to avoid the use of so many parenthesis in writing formulas. This will make the reading of complicated expressions easier. First, we may omit the outer pair of parenthesis of a formula. Second, the connectives are ordered as follows: $\neg, \wedge, \vee, \rightarrow$, and $\leftrightarrow$, and parentheses are eliminated according to the rule that, first, $\neg$ applies to the smallest formula following it, then $\wedge$ is to connect the smallest formulas surrounding it, and so on.

A *theory* is just a set of formulas and, in this paper, 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. A literal l is either an atom or the negation of an atom. A *normal program* is a set of formulas of the form $\alpha \rightarrow x$, where x is an atom and α is a conjunction of literals. When α is empty then formula $\alpha \rightarrow x$ reduces to the atomic formula x. Normal programs are of particular interest in Logic Programming.

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 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 $Y \subseteq X$.

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. Our presentation of our proof system of GLukG logic follows the Hilbert style, see [9]. 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, we use $T \vdash_X F$ 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. 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 $\vdash_X (F_1 \wedge_X \cdots \wedge_X F_n) \rightarrow_X F$.

C_ω logic [3] 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 $(a \rightarrow c) \rightarrow ((b \rightarrow c) \rightarrow (a \vee b \rightarrow c))$
C_ω**1** $a \vee \neg a$
C_ω**2** $\neg\neg a \rightarrow a$

Note that the first 8 axioms somewhat constraint the meaning of the $\rightarrow$, $\wedge$ and $\vee$ connectives to match our usual intuition. 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 [9].

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 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 $\mathcal{D}$, 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 most simple 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. If X is any logic, we write $\models_X \alpha$ to denote that α is a tautology in the logic X. We say that α is a logical consequence of a set of formulas Γ (denoted by $\Gamma \models_X \alpha$) if $\bigwedge \Gamma \rightarrow \alpha$ is a tautology.

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.

3 Motivation on GLukG Logic

One of the most well known semantics for NMR is the stable semantics. A semantics usually associates a set of two-valued models to a given theory. Consider the basic program B:

$a \leftarrow \neg b.$

B has 3 models, namely $M_1 = \{a\}, M_2 = \{b\}$, and $M_3 = \{a, b\}$. Model M_1 represents the function that evaluates atom a to true and atom b to false. Model M_2 represents the function that evaluates atom b to true and atom a to false. Finally,

model M_3 represents the function that evaluates both atoms a, b to true. It turns out that only M_1 is a stable model. Informally such rule reads as 'Normally a holds unless b is truth'. As we said in the introduction there are (contradictory) programs as P_2 that gives no models. This lack of information is useless in many cases. Sometimes consistent programs (in classical logic) as program P_1 also do not give models. Now, the stable semantics can be characterized using intuitionistic logic as follows. We can check each model and if it passes certain test, we consider it a stable model. Take M_1, does it follows that $\neg b, B \vdash_I a$? (where $\vdash_I$ represents the inference relation using intuitionistic logic) The answer is of course 'yes' and so M_1 is a stable model of B. Take now M_2, does it follows that $\neg a, B \vdash_I b$? Well, it turns out that $\neg a, B \vdash_I \neg\neg b$, but it can not infer b, so M_2 is not a stable model. Let us be a little bit more technical.

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 complementary set $\mathcal{L}_T \setminus M$. Moreover, given a theory T, we define the negation of the theory $\neg T$ as the set $\{\neg F \mid F \in T\}$ (the negation symbol is parameterized with respect to some given logic). 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).

Definition 1. *[16] 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 $P \cup \neg\widetilde{M} \Vdash_X M$.*

We call weak completion the construction $P \cup \neg\widetilde{M} \Vdash_X M$ just presented. If we use any constructive intermediate logic, we obtain the same class of X-stable models, namely the well known stable models. Intuitionistic logic is considered the weakest constructive intermediate logic, while Gödel's 3-valued logic G_3 is the strongest constructive intermediate logic.

G_3 can be presented as a multivalued logic. In this form it is defined through a 3-valued logic with truth values in the domain $\mathcal{D} = \{0, 1, 2\}$ where 2 is the designated value. The evaluation function of the 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. We write $\models \alpha$ to denote that the formula α is a tautology, namely that α evaluates to 2 (the designated value) for every valuation. We say that α is a logical consequence of a set of formulas Γ (denoted by $\Gamma \models \alpha$ if $\bigwedge \Gamma \rightarrow \alpha$ is a tautology.

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

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

Table 1. Truth tables of connectives in G_3 .

Hence, We came up with a basic mathematical question. Can we use any other type of monotonic logic different to the constructive intermediate logics such that the following properties hold?

1. The logic should have the following desirable properties. It should satisfy the replacement and deduction theorems. It should be expressive enough. If possible it should be finitely axiomatizable and somehow close to some constructive logic. If possible the logic should be many-valued.
2. Definition 1 should give the same result to our basic program B as the stable semantics. More generally, it is convenient that for every stratified program our intended semantics should agree with the stable semantics.
3. Every stable model of a normal program should be an intended model of our semantics. Every intended model of a normal program in our semantics should be a classical minimal model.
4. The semantics should be expressive enough.
5. The semantics should be close to classical logic, namely if a normal program P infers a given atom x in classical logic then P and $P \cup x$ should have the same intended models.

We considered the above points only for pragmatical reasons. We did not considered at first to be able to handled contradictory programs. The first partial answer was that we could use modal logic $S5$ but representing as long as we use $\neg\Box a$ for the definition of such negation operator, see [14,15]. That is, $\mathbb{Z}$ logic [2] is an interesting solution. However, we discover $\mathbb{Z}$ logic only recently. The use of several logics as well as a four-valued logic to answer our question was originally suggested in [14] and further studied in [15]. By studying the four valued logic introduced in [14] and its relation to S5 we observe that it was interesting to use modal Łukasiewicz 3 valued logic but again using $\neg\Box a$ for the definition of such negation operator. We also noticed that our logics have a paraconsistent flavor and we decided to explore the use of some paraconsistent logics as Pac obtaining interesting results, see [16]. We also made the simple observation that we could use G_3 with a basic change, the negation of 1 was changed from 0 to 2 originating in this way what we called G'_3 logic. Since it was created using some ideas based on Gödel, Łukasiewicz, and Gelfond, we are currently calling it GLukG logic. Using GLukG, we obtain a positive answer to all our questions risen before. GLukG has the following interesting properties.

1. It is stronger than C_ω logic but weaker than classical logic. For normal programs it can prove the same set of atoms as classical logic.
2. It can express other logics such Łukasiewicz 3-valued logic, classical logic, and G_3 logic.
3. It can be characterized as a 3-valued logic.
4. It can express a second negation that corresponds exactly to the negation of G_3 logic.

4 Axiomatization of GLukG

We present a Hilbert-style axiomatization of GLukG simpler than a previous one presented in [12]. Our current axiomatization is essentially the one presented in [13] with a minor simplification in the one axiom. This logic has 3 primitive logical connectives, namely $\rightarrow$, $\wedge$, and $\neg$. We also have several defined connectives.

1. $A \vee B := ((A \rightarrow B) \rightarrow B) \wedge ((B \rightarrow A) \rightarrow A)$.
2. $\sim A := (A \rightarrow (\neg A \wedge \neg\neg A))$.
3. $A \leftrightarrow B := (A \rightarrow B) \wedge (B \rightarrow A)$.

GLukG Logic has all the axioms of C_ω logic plus the following:

E1 $(\neg A \rightarrow \neg B) \leftrightarrow (\neg\neg B \rightarrow \neg\neg A)$
E2 $\neg\neg(A \rightarrow B) \leftrightarrow ((A \rightarrow B) \wedge (\neg\neg A \rightarrow \neg\neg B))$
E3 $\neg\neg(A \wedge B) \leftrightarrow (\neg\neg A \wedge \neg\neg B)$
E4 $((B \wedge \neg B) \rightarrow (\sim\sim A \rightarrow A)$

Note that Classical logic is obtained by adding any of the following list of axioms:

CL1 $A \rightarrow \neg\neg A$
CL2 $A \rightarrow (\neg A \rightarrow B)$
CL3 $(\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B)$

It is always useful to see that some strong inference rules hold in a given logic. In this sense, a simple but useful result is the following.

Theorem 1. *Let Γ and Δ be two set of formulas. Let θ, θ_1, θ_2, α, and ψ be arbitrary formulas. Then the following basic properties hold.*

1. *$\Gamma \vdash \alpha$ implies $\Gamma \cup \Delta \vdash \alpha$ (monotonicity)*
2. *$\Gamma \vdash \alpha$ and $\Delta, \alpha \vdash \psi$ then $\Gamma \cup \Delta \vdash \psi$ (cut)*
3. *$\Gamma, \theta \vdash \alpha$ iff $\Gamma \vdash \theta \rightarrow \alpha$ (deduction theorem)*
4. *$\Gamma \vdash \theta_1 \wedge \theta_2$ iff $\Gamma \vdash \theta_1$ and $\Gamma \vdash \theta_2$ ($\wedge$-rules)*
5. *$\Gamma, \theta \vdash \alpha$ and $\Gamma, \neg\theta \vdash \alpha$ iff $\Gamma \vdash \alpha$ (strong proof by cases)*

It is very important to note that GLukG can also presented as a multivalued logic. Such presentation is given in [16], where GLukG is called G'_3. In this form it is defined through a 3-valued logic with truth values in the domain $\mathcal{D} = \{0, 1, 2\}$ where 2 is the designated value. The evaluation function of the 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 2. We write $\models \alpha$ to denote that the formula α is a tautology, namely that α evaluates to 2 (the designated value) for every valuation. We say that α is a logical consequence of a set of formulas Γ (denoted by $\Gamma \models \alpha$ if $\bigwedge \Gamma \rightarrow \alpha$ is a tautology.

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

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

Table 2. Truth tables of connectives in GLukG.

4.1 Some theorems in GLukG

We present some useful theorems of GLukG logic. We can see for instance that $\sim$ behaves very much as a constructive negation. In fact it behaves exactly as the negation operator of G_3 logic. Note that of course $\sim\sim \alpha \rightarrow \alpha$ is not a theorem. The reader can consult [13] where the proof is given.

Lemma 1. *The following are theorems of* GLukG.

ON $\vdash \neg\beta \leftrightarrow \neg\neg\neg\beta$
EX $\vdash \neg A \rightarrow (\neg\neg A \rightarrow B)$
F1 $\vdash \sim \alpha \rightarrow (\alpha \rightarrow \beta)$
F2 $\vdash (\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow \sim \beta) \rightarrow \sim \alpha)$
F3 $\vdash (\alpha \rightarrow \beta) \rightarrow (\sim \beta \rightarrow \sim \alpha)$
F4 $\vdash (\sim\sim \beta \wedge \sim \theta) \rightarrow \sim (\beta \rightarrow \theta)$
F5 $\vdash \sim\sim \theta \rightarrow \sim\sim (\beta \rightarrow \theta)$
F6 $\vdash \beta \rightarrow (\sim \theta \rightarrow \sim (\beta \rightarrow \theta))$
F7 $(\sim\sim \beta \wedge \sim\sim \theta) \rightarrow \sim\sim (\beta \wedge \theta)$
G1 $\sim \alpha \rightarrow \neg\alpha$
G2 $\vdash \sim \neg\theta \rightarrow \sim\sim \theta.$
G3 $\vdash \neg\neg\theta \rightarrow \sim \neg\theta.$
G4 $\vdash \neg\neg\theta \rightarrow \sim\sim \theta.$
G5 $\vdash \sim \theta \rightarrow \neg\neg \sim \theta.$
G6 $\vdash \neg \sim \theta \rightarrow \sim\sim \theta.$
G7 $\vdash (\sim\sim \beta \wedge \neg\beta) \rightarrow \neg\neg\neg\beta.$
G8 $\vdash \sim \beta \rightarrow \neg\neg\neg\beta.$

4.2 Completeness in GLukG

The strategy of the proof for completeness is to follow the proof of completeness for propositional classical logic given by Kalmar, see [9]. We only sketch the proof but a complete one can be found in [13].

Definition 2. *Given a 3-valuation v of* GLukG*, we define for each formula A an associated formula A_v as follows:*

1. $A_v := \neg\neg A$, *if* $v(A) = 2$.
2. $A_v := \sim\sim A \wedge \neg A$, *if* $v(A) = 1$.
3. $A_v := \sim A$, *if* $v(A) = 0$.

For a set Γ of formulas, we write Γ_v to denote the set of formulas $\{\alpha_v : \alpha \in \Gamma\}$.

Lemma 2. *Given a formula α, whose set of atomic formulas is Δ, the following holds: $\Delta_v \vdash \alpha_v$.*

Proof. The proof is by induction on the size of α.
Base Case: α is an atomic formula, say p. Hence we need to show that $p_v \vdash p_v$, but this is immediately true.
Inductive step: Suppose that α is a non atomic formula. Then, we have 3 cases:

(Case $\neg$) Suppose that α is of the form $\neg\beta$. By inductive hypothesis we know that $\Delta_v \vdash \beta_v$. We need to consider 3 subcases:

$v(\beta) = 2$. Hence $\Delta_v \vdash \neg\neg\beta$. Since $v(\alpha) = 0$, we need to show that $\Delta_v \vdash\sim \alpha$, that is $\Delta_v \vdash\sim \neg\beta$. It suffices to show that $\vdash \neg\neg\beta\rightarrow \sim \neg\beta$. But this can be verified with the help of Theorem 1 and Lemma 1.

$v(\beta) = 1$. Hence $\Delta_v \vdash (\sim\sim \beta \wedge \neg\beta)$. Since $v(\alpha) = 2$, we need to show that $\Delta_v \vdash \neg\neg\alpha$, that is $\Delta_v \vdash \neg\neg\neg\beta$. It suffices to show that $\vdash (\sim\sim \beta \wedge \neg\beta) \rightarrow \neg\neg\neg\beta$. But this can be verified with the help of Theorem 1 and Lemma 1.

$v(\beta) = 0$. Hence $\Delta_v \vdash\sim \beta$. Since $v(\alpha) = 2$, we need to show that $\Delta_v \vdash \neg\neg\alpha$, that is $\Delta_v \vdash \neg\neg\neg\beta$. It suffices to show that $\vdash\sim \beta \rightarrow \neg\neg\neg\beta$. But this can be verified with the help of Theorem 1 and Lemma 1.

(Case $\rightarrow$) Suppose that α is of the form $\beta \rightarrow \theta$. By inductive hypothesis we know that $\Delta_v \vdash \beta_v$, and $\Delta_v \vdash \theta_v$. We need to consider 6 subcases:

$v(\beta) = 0$. Hence $\Delta_v \vdash\sim \beta$. Since $v(\alpha) = 2$, we need to show that $\Delta_v \vdash \neg\neg\alpha$, that is $\Delta_v \vdash \neg\neg(\beta \rightarrow \theta)$. It suffices to show that $\vdash\sim \beta \rightarrow \neg\neg(\beta \rightarrow \theta)$. But this can be verified with the help of Theorem 1 and Lemma 1.

$v(\theta) = 2$. Hence $\Delta_v \vdash \neg\neg\theta$. Since $v(\alpha) = 2$, we need to show that $\Delta_v \vdash \neg\neg\alpha$, that is $\Delta_v \vdash \neg\neg(\beta \rightarrow \theta)$ It suffices to show that $\vdash \neg\neg\theta \rightarrow \neg\neg(\beta \rightarrow \theta)$. But this can be verified with the help of Theorem 1 and Lemma 1.

$v(\beta) = 1, v(\theta) = 0$. Hence $\Delta_v \vdash\sim\sim \beta \wedge \neg\beta$ and $\Delta_v \vdash\sim \theta$. Since $v(\alpha) = 0$, we need to show that $\Delta_v \vdash\sim \alpha$, that is $\Delta_v \vdash\sim (\beta \rightarrow \theta)$. It suffices to show that $\vdash ((\sim\sim \beta \wedge \neg\beta)\wedge \sim \theta)\rightarrow \sim (\beta \rightarrow \theta)$. But this can be verified with the help of Theorem 1 and Lemma 1.

(Rest of cases of $\rightarrow$ and $\wedge$) They carry out similarly and since each case is rather tedious we shall not include it here, however is straightforward, so its details are left to the reader.

Lemma 3. *The following are simple results of* GLukG*, see [13].*

1. $\vdash \neg(A \rightarrow \neg\neg A) \rightarrow \neg \sim A$.

2. $\vdash \neg(A \to \neg\neg A) \to \sim\sim A$.
3. $\vdash \neg(A \to \neg\neg A) \to \neg A$.
4. $\vdash (\neg A \wedge (A \to \neg\neg A)) \to \sim A$.

Lemma 4. *$\Gamma, \neg\neg\alpha \vdash \theta$ and $\Gamma, (\sim\sim \alpha \wedge \neg\alpha) \vdash \theta$ and $\Gamma, \sim \alpha \vdash \theta$ implies $\Gamma \vdash \theta$.*

Proof. Suppose
(a) $\Gamma, \neg\neg\alpha \vdash \theta$
(b) $\Gamma, (\sim\sim \alpha \wedge \neg\alpha) \vdash \theta$
(c) $\Gamma, \sim \alpha \vdash \theta$.
By $\alpha \to \neg\neg\alpha, \alpha \vdash \neg\neg\alpha$ and (a) we get $\Gamma, (\alpha \to \neg\neg\alpha), \alpha \vdash \theta$.
By $\neg(\alpha \to \neg\neg\alpha) \vdash (\sim\sim \alpha \wedge \neg\alpha)$ (Lemma 3) and (b) we obtain $\Gamma, \neg(\alpha \to \neg\neg\alpha) \vdash \theta$.
By $((\alpha \to \neg\neg\alpha), \neg\alpha) \vdash \sim \alpha$ (Lemma 3) and (c) we conclude $\Gamma, (\alpha \to \neg\neg\alpha), \neg\alpha \vdash \theta$.
From $\Gamma, (\alpha \to \neg\neg\alpha), \alpha \vdash \theta$ and $\Gamma, (\alpha \to \neg\neg\alpha), \neg\alpha \vdash \theta$ and basic logical manipulation, we get $\Gamma, (\alpha \to \neg\neg\alpha) \vdash \theta$.
By $\Gamma, (\alpha \to \neg\neg\alpha) \vdash \theta$ and $\Gamma, \neg(\alpha \to \neg\neg\alpha) \vdash \theta$ and basic logical manipulation, we get $\Gamma \vdash \theta$, as desired.

Theorem 2. *(Soundnes and Completeness)*
(a)Every theorem is a tautology (in GLukG*).*
(b) Every tautology i is a theorem (in GLukG*).*

Proof. Case (a) This a straightforward proof by induction in the length of the proof. Of course one need to verify that every axiom is a tautology (this can be done by the help of a computer program) and that the rule of modus ponens preserves tautologies, which is a simple exercise.
Case (b) It follows by Lemma 4, Lemma 2, and a standard basic logical reasoning as in [9]. The key points of the proof are the following:
Suppose that α is a tautology whose set of atomic formulas is Δ. By Lemma 2, $\Delta_v \vdash \alpha_v$ for every 3-valuation v. Hence $\Delta_v \vdash \neg\neg\alpha$. Thus, by $C_\omega 2$ we obtain $\Delta_v \vdash \alpha$. Let a be an atomic formula in Δ, and let $\Gamma := \Delta \setminus \{a\}$. For every 3-valuation v, based on Γ, we have:
$\Gamma_v, \neg\neg a \vdash \alpha$ and
$\Gamma_v, (\sim\sim a \wedge \neg a) \vdash \alpha$ and
$\Gamma_v, \sim a \vdash \alpha$
By Lemma 4, $\Gamma_v \vdash \alpha$. After $|\Delta|$ steps, we finally obtain $\vdash \alpha$.

5 Further Properties of GLukG Logic

In this section we present two main results of GLukG Logic, but first we present some definitions.

Definition 3. *Let T be a (possible infinite)theory. We say that T is contradictory if there exists a formula α such that $T \vdash \alpha$ and $T \vdash \neg\alpha$. Otherwise, we say that T is noncontradictory. We say that T is complete if for every formula α, $T \vdash \alpha$ or $T \vdash \neg\alpha$. A theory T' is said to be an extension of a theory T if for every formula α, $T \vdash \alpha$ then $T' \vdash \alpha$.*

The two main results that help to understand better GLukG are the following. First, that GLukG can express Łukasiewicz 3-valued logic that is a well known expressive logic. Second, one expect that a noncontradictory theory can be extended to a noncontradictory and complete theory. All well known logics satisfy this property. We show that GLukG also satisfies it.

Lemma 5. *[13] In* GLukG *we can express Łukasiewicz 3-valued logic.*

Proof. To express Łukasiewicz 3-valued logic we need to be able to define the corresponding negation and implication connectives. In fact, the original 3-valued system is based on these 2 connectives which are intended to generalize the negation and implication connectives of classical logic, see [20]. Negation of Łukasiewicz 3-valued logic is defined by $\sim A \vee (\sim\sim A \wedge \neg A \wedge A)$ and implication is defined by $(A \rightarrow B) \vee (\sim\sim A \wedge \neg A \wedge A)$. Note that Łukasiewicz 3-valued logic is usually defined with 0, 1/2, 1 values but instead we use 0, 1, 2 respectively.

We now move to prove the second property which is a a new result.

Lemma 6. *Let T be a (possible infinite) noncontradictory theory. Suppose that there exists a formula α such that $T \nvdash \alpha$ and $T \nvdash \neg\alpha$. Then $T \cup \{\alpha\}$ is noncontradictory or $T \cup \{\neg\alpha\}$ is noncontradictory.*

Proof. Suppose (in order to obtain a proof by contradiction) that $T \cup \{\alpha\}$ is contradictory and $T \cup \{\neg\alpha\}$ is contradictory. Then exist formulas β and θ such that $T \cup \{\alpha\} \vdash \beta$, $T \cup \{\alpha\} \vdash \neg\beta$, $T \cup \{\neg\alpha\} \vdash \theta$, $T \cup \{\neg\alpha\} \vdash \neg\theta$. Hence $T \cup \{\alpha\} \vdash \beta \wedge \neg\beta$ and $T \cup \{\neg\alpha\} \vdash \theta \wedge \neg\theta$.
Thus $T \vdash \alpha \rightarrow (\beta \wedge \neg\beta)$ and $T \vdash \neg\alpha \rightarrow (\theta \wedge \neg\theta)$. From this is not hard to see that $T \vdash (\beta \wedge \neg\beta) \vee (\theta \wedge \neg\theta)$. But it is easy to verify that $\vdash \neg((\beta \wedge \neg\beta) \vee (\theta \wedge \neg\theta))$. So, $T \vdash \neg((\beta \wedge \neg\beta) \vee (\theta \wedge \neg\theta))$. Hence T is contradictory obtaining a contradiction.

If we have a noncontradictory theory T and a formula α. We define $T(\alpha) = T \cup \{\alpha\}$ if $T \cup \{\alpha\}$ is noncontradictory and otherwise $T(\alpha) = T \cup \{\neg\alpha\}$. By the above lemma, $T(\alpha)$ is noncontradictory.

Lemma 7. *If T is a noncontradictory theory, then there is a noncontradictory complete extension of T.*

Proof. Let $\beta_1, \beta_2, \ldots$ be an enumeration of all formulas based on the signature of T. Define a sequence $T_0, T_1, \ldots$ of theories defined as follows. T_0 is T. Suppose that T_n is defined with $n \geq 0$. If T_n does not decides β_{n+1}, then let T_{n+1} be $T_n(\beta_{n+1})$, otherwise we let $T_{n+1} = T_n$. Let T_ω be the union of all T_i $(i \geq 0)$. One can prove that every T_i is noncontradictory by induction on i. The base case is by hypothesis (recall that $T = T_0$ is noncontradictory) and the inductive step follows by Lemma 6. Then one can easily prove that T_ω is noncontradictory. If T_ω were contradictory the some T_i should be contradictory, but this is a contradiction. Also by construction T_ω is complete. Hence $T\omega$ is a noncontradictory and complete extension of T, as desired.

6 Nonmonotonic reasoning

The research community has long recognized the study of nonmonotonic 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 nonmonotonic reasoning. The original idea was to use well known modal logics and this idea can be traced back to [8]. Subsequently [7] 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. [10] suggested the use of autoepistemic logic (AEL) as an alternative formalization of nonmonotonic reasoning to avoid the problems encountered with standard modal logics.

More recently (in [15]) a family of nonmonotonic semantics was constructed based on modal logics. This 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. These type of semantics are called ground nonmonotonic modal logics. For a restricted class of formulas, it was recently shown that all ground nonmonotonic modal logics between T and S5 are equivalent [15]. Furthermore, it is also shown that these logics are equivalent to a nonmonotonic logic that can be constructed using the well known $\mathcal{FOUR}$ bilattice [15]. According to [1] $\neg\Box a$ is in general a paraconsistent negation. Hence, for this approach the main ingredient is the use of paraconsistent logics.

[17] presented a characterization of the stable model semantics of disjunctive programs 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.

Based on all the previous work, [16] suggest a construction of nonmonotonic semantics based on monotonic logics. This definition assumes some negation connective. Modal logics can of course be considered as long as we use $\neg\Box a$ for the definition of such negation operator.

Let as come back to Definition 1. If we use GLukG instead of G_3, we obtain a different semantics than the the well known stable semantics. Such semantics is also equivalent to GNM-S5 [16]. Since GLukG is related to paraconsistent logics it is interesting to study a very large fragment of such logics. Let us consider all logics stronger or equal to C_ω, the weakest paraconsistent logic, and weaker or equal to GLukG. It turns out that the weak completions of all these logics are equivalent to each other for disjunctive programs and that, moreover, they are equivalent to GNM-S5. Actually, a large fragment of logics that include many well known modal and paraconsistent logics have this invariant property. These results are presented in detail in [15,16].

Let us consider example P_1 from the introduction. Clearly $\{a, b\}$ is a GLukG-stable model of P_1. This is because $\{a, b\}$ is a classical 2-valued model of P_1 and P_1 proves both a and b in GLukG logic.

Consider now example P_2. Since P_2 does not have 2-valued models then it does not have GLukG-stable models.

As a final example consider program P_3:
$a \leftarrow \neg b$.
This program has three 2-valued models, namely $M_1 = \{a\}$, $M_2 = \{b\}$ and $M_3 = \{a, b\}$. One can see that M_1 is the unique GLukG-stable model of this program as follows. Note that $\neg b, \neg b \rightarrow a$ proves a in GLukG logic. Hence M_1 is a GLukG-stable model of P_3. M_2 is not a GLukG-stable model of P_3, since $\neg a, \neg b \rightarrow a$ does not prove b in GLukG logic. M_3 is not a GLukG-stable model of P_3, since $\neg b \rightarrow a$ does not prove $\{a, b\}$ in GLukG logic.

6.1 A nonmonotonic semantics that can deal with inconsistencies

In this subsection we introduce a nonmonotic semantics that can deal with inconsistent programs. Our semantics is different to the one presented in [13].

Let us consider an example to explain what we have in mind with respect to a nonmonotic semantics that can deal with inconsistent programs. Consider again our program example P_2: $a \rightarrow b$, $\neg a$, and a. Then P_2 does not have X-stable models, not matter the logic used.

We are interested in the definition of a semantics that can give the following set of literals as the unique output of P, namely $\{b, \neg a, a\}$.

Let $Pos(T)$ denote the set of positive formulas of a given theory T. Let $Neg(T)$ denote the set of negative formulas of a given theory T.

Definition 4. *Let P any finite propositional theory. Let M be a set of literals. Then M is a paraconsistent-*GLukG *stable model of P if*
(A) there exists a GLukG*-stable model N of P such that $Pos(M) = N$ and $Neg(M) = \mathcal{L}_P \setminus N$.*
(B) Otherwise, if P does not have GLukG*-stable models then P has only one paraconsistent-*GLukG *stable model, namely the set of all literals than be proved from P using* GLukG *logic.*

Following this approach, it turns out that $\{b, a, \neg a\}$ is the unique paraconsistent-GLukG stable model of our example P_2. Consider again the basic program B:
$a \leftarrow \neg b$.
Then B has one paraconsistent-GLukG stable model, namely $M_1 = \{a, \neg b\}$.

7 Conclusions and Future work

We have given an axiomatization of GLukG based on C_ω plus only 4 new axioms. For future work we consider two problems. The first one consists on determining wether or not GLukG is a maximal paraconsistent logic. It would be maximal if after adding any new axiom to GLukG that is not already a theorem we can

prove $(A \wedge \neg A) \rightarrow B$. The second problem consists on understanding in more detail how to take advantage of the paraconsistency flavor of GLukG in order to define a non-monotonic semantics that could deal properly with inconsistent programs. We provided already an initial idea in Definition 4 but future work must be done on this line of research.

References

1. J. Béziau. Paraconsistent logic from a modal viewpoint. *Journal of Applied Logic*, 3:7–14, 2005.
2. J. Béziau. The paraconsistent logic Z. *Journal of Logical Philosophy*, 15:99–111, 2006.
3. N. C. A. daCosta. *On the theory of inconsistent formal systems (in Portuguese)*. PhD thesis, Curitiva:Editora UFPR, Brazil, 1963.
4. F. M. Donini, D. Nardi, and R. Rosati. Ground nonmonotonic modal logics. *Logic and Computation*, 7(4), 1997.
5. M. Gelfond. On stratified auto-epistemic theories. In *Proceedings of AAAI*, pages 207–211. Morgan Kaufman, 1987.
6. M. Gelfond and V. 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.
7. D. McDermott. Nonmonotonic logic II: Nonmonotonic modal theories. *ACM Transactions on Computer Systems*, 29:33–57, 1982.
8. D. McDermott and J. Doyle. Non-monotonic logic I. *Artificial Intelligence*, 13:41–72, 1980.
9. E. Mendelson. *Introduction to Mathematical Logic*. Wadsworth, Belmont, CA, third edition, 1987.
10. 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.
11. J. C. Nieves, M. Osorio, U. Cortés, I. Olmos, and J. A. Gonzalez. Defining new argumentation-based semantics by minimal models. In *Seventh Mexican International Conference on Computer Science (ENC 2006)*, pages 210–220. IEEE Computer Science Press, September 2006.
12. M. Osorio, J. Arrazola, J. L. Carballido, and O. Estrada. An axiomatization of G'_3. In *In proceedings of the WS of Logic, Language and Computation, 2006, CEUR Vol-220*, 2006.
13. M. Osorio and J. L. Carballido. Brief study of G'_3 logic. In *Submitted for publication*, 2007.
14. M. Osorio and J. A. Navarro. Modal logic $S5_2$ and FOUR (abstract). In *2003 Annual Meeting of the Association for Symbolic Logic*, Chicago, June 2003.
15. M. Osorio, J. A. Navarro, J. Arrazola, and V. Borja. Ground nonmonotonic modal logic S5: New results. *Journal of Logic and Computation*, 15(5):787–813, 2005.
16. M. Osorio, J. A. Navarro, J. Arrazola, and V. Borja. Logics with common weak completions. *Journal of Logic and Computation*, 16(6):867–890, 2006.
17. D. Pearce. Stable inference as intuitionistic validity. *Logic Programming*, 38:79–91, 1999.
18. C. Sakama and K. Inoue. Paraconsistent stable semantics for extended disjunctive programs. *Journal of Logic and Computation*, 5:265–285, 1995.

19. C. Sakama and K. Inoue. Negation as failure in the head. *Journal of Logic Programming*, 35(1):39–78, 1998.
20. A. Urquhart. Many-valued logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Vol III*, volume 3. D. Reidel Publishing Co., 1986.