ISSN 2007-9737

An Alternative Definition of Stable Models Via Łukasiewicz Logic

Mauricio Osorio[1], Aldo Figallo-Orellano[2], Diego Huerta[1]

[1] Universidad de las Américas Puebla,
Departamento de Actuaría, Física y Matemáticas,
Mexico

[2] Universidade Federal do Rio Grande do Norte,
Departamento de Informática e Matemática Aplicada,
Brazil

{mauricioj.osorio, diego.huertaoa}@udlap.mx,
aldofigallo@gmail.com

Abstract. Extensions of G_3 (3-valued Gödel logic) were studied as tools for knowledge representation and nonmonotonic reasoning. One of these extensions was studied and baptized as G'_3 by Osorio et al. as a tool to define semantics of logic programming. In this work, we will explore the possibility to use another fuzzy logic for knowledge representation. In particular, we show that Łukasiewicz 3-valued logic (for short, $Ł_3$) can be used for knowledge representation based on logic programming. Firstly, we prove that the definition of stable model for $Ł_3$ is equivalent to the obtained through G_3 for augmented programs, but when we consider more general programs we obtain more answers for stable models on $Ł_3$ than G_3. Finally, we present and explore a new definition of stable model based on Lukasiewicz n-valued logic via the use of Monteiro-Baaz Δ operator.

Keywords. Knowledge representation, stable semantics, Łukasiewicz logic.

1 Introduction

Answer set programming (ASP) is a form of declarative programming based on the stable model (answer set) semantics of logic programming.

Recall that stable semantics allows us to handle problems with default knowledge and produce nonmonotonic reasoning using the concept of negation as failure.

It was originally defined for the class of theories called normal programs [11]. Currently, ASP is robust and mature enough, offering many important language constructs like aggregation, (weak) constraints, different types of negations, and optimization statements to mention a few, as high-performance solvers do.

An example of a state-of-the-art and award-winning ASP solver is clasp [9] demonstrating its competitiveness and versatility, by winning first places at various solver contests since 2011 (e.g., ASP, CASC, MISC, PB, and SAT competitions).

The efficiency of such programs has increased the list of practical applications in the areas of planning, logical agents and artificial intelligence.

On the other hand, the term fuzzy logic emerged in the development of the theory of fuzzy sets by Zadeh in 1965, [14, 24].

Although the concept of uncertainty had been studied by philosophers, the significance of Zadeh's work was that it challenged not only probability theory, as the sole agent for uncertainty, but also the very foundations upon which probability theory is based: Aristotelian two-valued logic.

When A is a fuzzy-set and x is a relevant object, the proposition "x is a member of A" is not necessarily either true or false, as required by two-valued logic, but it may be true to some degree, the degree at which x is actually a member of A [15].

Furthermore, fuzzy logic was introduced to study the question of uncertainty from a foundational point of view based on many-valued logics. In this sense, fuzzy logic can be considered as a degree-based approach to uncertainty.

Some systems like Łukasiewicz, product and Gödel logics are, just like fuzzy sets, valued over the real interval [0,1]. This supports the idea of fuzzy logic being as a kind of foundational counterpart of fuzzy set theory which is a discipline mainly devoted to engineering applications.

In particular, the infinite-valued Lukasiewicz logic Ł, introduced for philosophical reasons by Jan Lukasiewicz, is among the most important and widely studied of all non-classical logics. Later, MV-algebras were introduced by C. Chang in order to prove completeness with respect to the calculus Ł, see for instance, [3].

The logic Ł was introduced taking in mind that if statements about future events are already true or false, then the future is as much determined as the past and differs from the past only in so far as it has not yet come to pass.

In order to avoid the situations in which further development is impossible, it was proposed to reject the law of Excluded Middle, that is, the assumption that every proposition is true or false. The n-valued Łukasiewicz logic (Ł$_n$) was axiomatizated by Grigolia in 1977 in [12].

It is worth mentioning that initially Łukasiewicz proposed the 3-valued version of Ł that today is called Ł$_3$. Afterwards, Lukasiewicz generalized his three-valued logic to n values and also to an infinite-valued system.

In this paper, we consider the Ł$_3$ logic in order to show a non-standard application of fuzzy logic. Firstly, we show how Łukasiewicz logic can be used for knowledge representation based on logic programming.

In particular, we present a definition of stable propositional Łukasiewicz models using the $\{\wedge, \vee, \rightarrow, \neg, \Box\}$ connectives of Łukasiewicz logic. Secondly, we show that our definition is equivalent to the standard definition of stable models for augmented programs, due to Lifschitz et al. [16].

Moreover, we show that this equivalence fails when we consider the proposed definition of stable models for arbitrary propositional theories given in [23], see Remark 1. However, it is possible to reduce any propositional theory into the class of augmented programs [20].

Gelfond showed that the perfect models of stratified logic programs can be characterized in terms of extensions of the corresponding autoepistemic theory [10] . His characterization is based on the interpretation of $\mathrm{not}\ \mathrm{a}$ as $\neg\Box a$, where $\Box$ is Monteiro-Baaz operator.

In this paper, $\neg\ a$ is interpreted as $\Box\neg a$, where both operators $\neg$ and $\Box$ are the standard operators in Łukasiewicz logics. Some connections between Łukasiewicz logics and the stable semantics were recognized in [18, 22] but all results in this paper are new, to the best of our knowledge. Our paper is structured as follows.

In section 2, we summarize some definitions, logics and semantics necessary to understand this work. Besides, we present a characterization of the stable models semantics for augmented programs in terms of the intermediate logic G_3.

Then, in section 3, we present our results; namely, we introduce the notion of Ł$_3$-stable models. We show its correspondence with standard stable models for the class of augmented programs. Later on, we generalize theses results to many-valued Łukasiewicz logics. Finally, in section 4 we present some conclusions.

2 Background

In this section, we summarize some basic concepts and definitions necessary to understand this paper.

2.1 Logic Programs

A signature $\mathcal{L}$ is a finite set of elements that we call atoms, or propositional symbols. The language of a propositional logic has an alphabet consisting of:

– propositional symbols: $p_0, p_1, \ldots$
– constant symbols: $\bot$, $\top$.
– binary connectives: $\wedge$, $\vee$, $\leftarrow$.
– unary connectives: $\neg$, $\sim_{\mathrm{G}_3}$, $\sim_{L_3}$.
– auxiliary symbols: $($, $)$.

where $\wedge$, $\vee$, $\leftarrow$ are 2-place connectives and $\neg$, $\sim_{\mathrm{G}_3}$, $\sim_{L_3}$ are 1-place connectives. Theories and formulas are built up as usual in logic.

If x is any particular unary connective, we write T_x to denote a theory in which the unique unary connective allowed is x.

A program is a finite theory and by default, we assume that our formulas and programs are constructed using the $\neg$ connective, unless stated otherwise. In the following, we will say that:

A literal is either an atom a, called positive literal; or the negation of an atom $\neg a$, called negative literal.

A $\{\vee, \wedge, \neg\}$-formula is a formula built using only the connectives in $\{\vee, \wedge, \neg\}$.

An augmented clause is a formula of the form: $H \leftarrow B$, where both H and B are $\{\vee, \wedge, \neg\}$-formulas.

Besides, we call H the head of the formula and B the body of the formula. $H \leftarrow B$ corresponds to the standard formula $B \rightarrow H$.

We define an augmented program P, as a finite set of augmented clauses.

The body of an augmented formula could be just the $\top$ constant, in which case the formula is known as an extended fact and can be denoted just by the head formula H.

We write $\mathcal{L}_P$, to denote the set of atoms that appear in the clauses of P.

Given a set of atoms M and a signature $\mathcal{L}$, we define $\neg\widetilde{M} = \{\neg b \mid b \in \mathcal{L} \setminus M\}$, and $\neg\neg M = \{\neg\neg a \mid a \in M\}$.

Given a program P and a set of atoms M, we write $< P, M >$ to denote the program: $P \cup \neg\widetilde{M} \cup \neg\neg M$.

Given the unary connective $\sim_{\mathrm{G}_3}$ and the program $< P, M >$, we write $< P, M >_{\sim_{\mathrm{G}_3}}$ to the denote the theory obtained from $< P, M >$ by replacing the connective $\neg$ by the connective $\sim_{\mathrm{G}_3}$.

Furthermore, it is worth observing that $< P, M >$ is also an augmented program. Now, let us consider the following example:

Example 1. Let P be the following program:

$$\{a \leftarrow \neg b,\ \ b \leftarrow \neg a\}, \tag{1}$$

And let $M = \{a\}$, then $< P, M >$ is $\{a \leftarrow \neg b,\ b \leftarrow \neg a\} \cup \{\neg b\} \cup \{\neg\neg a\}$. So, $< P, M >_{\sim_{\mathrm{G}_3}} = \{a \leftarrow \sim_{\mathrm{G}_3} b,\ b \leftarrow \sim_{\mathrm{G}_3} a\} \cup \{\sim_{\mathrm{G}_3} b\} \cup \{\sim_{\mathrm{G}_3}\sim_{\mathrm{G}_3} a\}$.

2.2 Logics

Now, we will review some logics that are relevant to this paper to characterize different semantics of normal and more general programs.

Moreover, we present definitions in terms of true values as well as Hilbert style definitions for most of these logics. The logics considered here have the modus ponens as a unique inference rule.

2.2.1 Łukasiewicz's Logic

Łukasiewicz logic is presented in the language $\{\rightarrow, \neg\}$, further connectives are definable from $\rightarrow$ and $\neg$ as follows:

- $\phi \oplus \psi := \neg\phi \rightarrow \psi$.
- $\phi \otimes \psi := \neg(\neg\phi \oplus \neg\psi)$.
- $\phi \vee \psi := (\phi \rightarrow \psi) \rightarrow \phi$.
- $\phi \wedge \psi := \neg(\neg\phi \vee \neg\psi)$.
- $\varphi \leftrightarrow \psi := (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$.

The truth constants are defined as follows: $\top := \phi \rightarrow \phi$ and $\bot := \neg\top$. So, it is possible to define the following for every positive integer k:

- $(k+1)\phi := k\phi \oplus \phi$.
- $\phi^{(k+1)} := \phi^k \otimes \phi$.

When $k = 0$ we have $k\phi = \phi^k = \top$. It was originally defined as a three-valued Łukasiewicz logic, denoted by Ł$_3$.

Afterwards, Łukasiewicz generalized his three-valued logic to n values and also to an infinite-valued system.

Recall that the propositional Łukasiewicz logic is defined as the following Hilbert style system of axioms and rule ([13, 3]):

– (Ł1) $\phi \to (\psi \to \phi)$.

– (Ł2) $(\phi \to \psi) \to ((\psi \to \chi) \to (\phi \to \chi))$.

– (Ł3) $(\phi \to \psi) \to (\neg\psi \to \neg\phi)$.

– (Ł4) $(\phi \vee \psi) \to (\psi \vee \phi)$.

– (MP) The rule of modus ponens: $\dfrac{\phi,\ \phi \to \psi}{\psi}$.

Now, for every integer $n \geq 2$, the n-valued Łukasiewicz logic (for short, Ł$_n$) is defined as an extension of Ł by means of adding the following axioms ([12, 13]):

– (Ł1) $(n-1)\phi \leftrightarrow n\phi$,

– (Ł2) $(t\phi^{(t-1)})^n \leftrightarrow (n\phi)^t$, for every $t = 2, 3, \cdots, (n-2)$ such that t does not divide $n-1$.

A matrix for Ł$_n$, with $n-1$ as the unique designated value, is MV$_n$-algebra $\langle\{0, 1, 2, \cdots, (n-1)\}, \oplus, \neg, 0\rangle$ where $x \oplus y := \min(n-1, x+y)$ and $\neg x := (n-1) - x$, see [12, Examples pag. 4] and [3, Definition 1.1.1].

It is well-known that the class of MV-algebras is semantics for Łukasiewicz logic and its extension Ł$_n$ has the class of MV$_n$-algebras as algebraic semantics. For more details about the theory of MV$_n$-algebras the reader can consult the book [3, Section 8.5].

2.2.2 Łukasiewicz's 3-Valued Logic

The polish logician and philosopher 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 by Ł$_3$, we first have to modify the syntax of our formulas to allow only as primitive connectives: the constant (0-place connective) $\perp$ (false) and the 2-place connective $\to$ (implication).

These connectives operate over a domain $D = \{0, 1, 2\}$, with 2 as the unique designated value, and are defined as follows:

$$x \to y = \min(2, 2 - x + y). \tag{2}$$

Table 1. Truth tables of connectives in Ł$_3$

x	$\sim_{L_3} x$	$\Box x$	$\Diamond x$
0	2	0	0
1	1	0	2
2	0	2	2

$\to$	0	1	2
0	2	2	2
1	1	2	2
2	0	1	2

Other connectives in Ł$_3$ are introduced in terms of $\perp$ and $\to$ as follows:

– $\sim_{L_3} A := A \to \perp$.
– $\top := \sim_{L_3} \perp$.
– $A \vee B := (A \to B) \to B$.
– $A \wedge B := \sim_{L_3} (\sim_{L_3} A \vee \sim_{L_3} B$.
– $\Box A := \sim_{L_3} (A \to \sim_{L_3} A)$.
– $\Diamond A := \sim_{L_3} A \to A$.

The truth tables of most connectives are shown in Table 1, the conjunction and disjunction connectives (not shown) coincide with the $\min$ and $\max$ functions respectively.

A syntactic characterization of the modal content of Ł$_3$ is studied in [17], where the behavior of modal operators is checked against some of the relevant modal principles.

It is important to note that the algebra $\langle\{0, 1, 2\}, \oplus, \sim_{L_3}, 0\rangle$ is an MV-algebra in terms of [3, Definition 1.1.1], where $\phi \oplus \psi := \neg\phi \to \psi$ using Table 1.

For more general results in Łukasiewicz logics, the reader can be referred to [3]. Another observation worth mentioning is that the modal operators $\Box$ and $\Diamond$ where considered in the context of Łukasiewicz logic by Cignoli, [2].

Furthermore, these operators was considered on Gödel-Dummmett logic by Baaz, [1]. Indeed, Baaz baptized this operator with Δ and it is known in the literature as Monteiro-Baaz operator.

This operator was studied on fuzzy logics by Esteva, Godo, Hájek, Montagna and others. These authors presented an algebraic study of these modals operators over certain fuzzy logics, see for instance [6, 7].

Table 2. Truth tables: connectives in G_3

x	$\sim_{\mathrm{G}_3} x$
0	2
1	0
2	0

$\to$	0	1	2
0	2	2	2
1	0	2	2
2	0	1	2

2.2.3 G_n Logic or Gödel n-Valued Logic

Gödel defined, in fact, a family of many-valued logics G_n with truth values over the domain $D = \{0, 1, \ldots, n-1\}$ and with $n-1$ as the unique designated value. Logic connectives are defined as:

– $x \wedge y := \min(x, y)$; $x \vee y = \max(x, y)$,

– $x \to y := n-1$ if $x \leq y$ and y otherwise,

– $\sim_{\mathrm{G}_n} x = x \to 0$.

Table 2 illustrates the implication and negation connective for G_3. Notice that $\sim_{\mathrm{G}_3}\ x$ can be expressed using Ł$_3$:

$$\sim_{\mathrm{G}_3} x := \Box \sim_{L_3} x. \tag{3}$$

2.3 Semantics

We assume that the reader is familiar with the notions of interpretations, models, and logical consequence in the usual way. Recall that our interest is in the above non classical logics.

Both have only 2 as the only designated value. Hence a model of a theory P is an interpretation I, such that $I(P) = 2$. Given a logic X, we use the notation $\models_X \alpha$ to indicate that the formula α is tautology in X.

With respect to Logic Programming, we adopt the following conventions. Given a set of atoms M and a signature $\mathcal{L}$, we define the interpretation I_M based on M as: $I_M(x) = 2$ if $x \in M$, and $I_M(x) = 0$ otherwise, namely when $x \in \mathcal{L} \setminus M$.

Notice that the values in $\{0, 2\}$ behave as in classical logic in our both non-classical logic considered in our paper. Hence I_M models a theory P in G_3 iff I_M models P in Ł_3.

Given a program P and a set of atoms M, the expression $P \Vdash_X M$ will mean (in this paper) that I_M is a model of P and for every formula $\alpha \in M$, α is a logical consequence of P in logic X.

By abuse of the language we may sometimes say that M is a model of P if I_M models a theory P. By our last considerations, we do not have to refer to any specific logic.

A very important remark on notation is the following. When we write any of the following statements: $\models_X\ \alpha$, or α is a tautology in X, or $Q \Vdash_X M$ etc, it is understood that the behaviour of the connectives in our formulas correspond the standard connectives in logic X.

For instance $\models_{\mathrm{G}_3}\ \{\neg a \to b\}$ assumes that $\neg$ and $\to$ are the connectives of G_3 logic. Similarly $\models_{\text{Ł}_3}\ \{\neg a \to b\}$ assumes that $\neg$ and $\to$ are the connectives of Ł_3 logic.

However, if one makes a particular connective such as $\sim_{\mathrm{G}_3}$ in $\models_{\text{Ł}_3} \{\sim_{\mathrm{G}_3}\ a \to b\}$ explicit then of course, it is understood that $\to$ is our implication operator of Ł_3 logic, but $\sim_{\mathrm{G}_3}$ is the negation operator in G_3 logic.

But, after all, it is also a non-primitive operator in Ł_3 logic constructed using the primitive operators of the same logic.

2.4 Expressing Stable Semantics based on G_3

We present a general characterization of stable models in terms of G_3 logic. It is important to observe that in the following result we can use intuitionistic logic instead of G_3 logic.

This characterization allows us to understand what stable models are without having to appeal to the original definition.

Theorem 1. [21] Given a general logic program P, a set of atoms $M \subseteq \mathcal{L}_P$ is a stable model of P if and only if $< P, M >_{\sim_{\mathrm{G}_3}} \Vdash_{\mathrm{G}_3} M$.

Example 2. Let P be the following program:

$$\{a \leftarrow \neg b\}, \tag{4}$$

And let $M_1 = \{a\}$ and $M_2 = \{b\}$. According to the definition of stable semantics, since $P \cup \{\sim_{\mathrm{G}_3} b\} \cup \{\sim_{\mathrm{G}_3}\sim_{\mathrm{G}_3}\ a\} \Vdash_{G_3}\ \{a\}$, then M_1 is a stable model of P.

However, it is false that $P \cup \{\sim_{\mathrm{G}_3} a\} \cup \{\sim_{\mathrm{G}_3}\sim_{\mathrm{G}_3} b\} \Vdash_{G_3} \{b\}$, hence M_2 is not a stable model of P as the reader can easily check. In fact, M_1 is the unique stable model of P.

Recall that in our programs we write $\neg$, but in the notation $< P, M >_{\sim_{\mathrm{G}_3}}$, $\neg$ is substituted by $\sim_{\mathrm{G}_3}$.

Example 3. Let P be the following program:

$$\{a \leftarrow \neg b, \;\; b \leftarrow \neg a\}\,. \tag{5}$$

And let $M_1 = \{a\}$ and $M_2 = \{b\}$. According to the definition of stable semantics, since $P \cup \{\sim_{\mathrm{G}_3} b\} \cup \{\sim_{\mathrm{G}_3}\sim_{\mathrm{G}_3} a\} \Vdash_{G_3} \{a\}$ and $P \cup \{\sim_{\mathrm{G}_3} a\} \cup \{\sim_{\mathrm{G}_3}\sim_{\mathrm{G}_3} b\} \Vdash_{G_3} \{b\}$, then M_1 and M_2 are stable models of P as the reader can easily check.

2.5 Some Examples

We now provide some examples of augmented programs and their stable models. This with the purpose that the reader could check the models and thus verify if the previously presented theory has been well understood.

– Program (example)

  – $a \vee b \vee \neg e \leftarrow \top$,

  – $e \leftarrow \top$,

  – $c \leftarrow a \wedge e \wedge \neg b \wedge \neg a$,

  – $\neg b \leftarrow d \wedge \neg e$.

  Stable models: $\{\{b, e\}, \{a, e\}\}$

– Program (example)

  – $d \vee c \leftarrow a \wedge b$,

  – $\neg e \leftarrow c \wedge e \wedge \neg d$,

  – $b \vee d \vee \neg b \leftarrow a$,

  – $c \leftarrow \neg a \wedge \neg b$,

  – $e \vee \neg b \leftarrow e \wedge d$.

  Stable models : $\{\{c\}\}$

– Program (example)

  – $a \vee d \leftarrow \neg c$,

  – $\neg e \leftarrow b$,

  – $\neg a \leftarrow e \wedge c$.

  Stable models : $\{\{d\}, \{a\}\}$

– Program (example)

  – $b \vee c \leftarrow a \wedge c \wedge \neg e \wedge \neg a$,

  – $\neg d \leftarrow b$.

  Stable models : $\{\{\}\}$

– Program (example)

  – $\perp \leftarrow \neg e$,

  – $c \vee \neg d \leftarrow c$.

  Stable models : $\{\}$

– Program (example)

  – $c \vee e \vee \neg a \leftarrow \neg b$,

  – $b \vee a \vee \neg b \leftarrow \neg c$,

  – $a \vee \neg a \leftarrow c \wedge \neg d$,

  – $d \vee a \leftarrow e \wedge \neg d$.

  Stable models : $\{\{\}, \{b\}, \{a, e\}, \{a, c\}\}$.

3 Stable Models in Łukasiewicz's Logic Ł$_3$

In this section, we present our main results, namely, we introduce Ł$_3$-stable models and we show their correspondence with standard stable models. Then we generalize this results to consider many-valued logics.

3.1 3-Valued Logics

Theorem 2. Given an augmented theory P and a 3-valued interpretation I, I models P in G_3 logic iff I models $P_{\sim_{\mathrm{G}_3}}$ in Ł$_3$ logic.

Proof. It is enough to take an augmented clause C to prove the theorem. So, we have to see that for every 3-valued interpretation I, I models C in G_3 logic iff I models $C_{\sim_{\mathrm{G}_3}}$ in Ł$_3$ logic.

Now, let C be an augmented clause and I a 3-valued interpretation. By definition, C has the form $H \leftarrow B$, where both H and B are $\{\vee, \wedge, \neg\}$-formulas.

Then $I(H)$ and $I(B)$ do not change their value when considering the two different logics. The only possible difference in C is in the implication: $1 \to 0 = 0$ in G_3 logic and $1 \to 0 = 1$ in L_3 logic.

But in neither of these two possibilities, the valuation takes true-valued 2. Consequently, if I models C in one logic, then I models C in the other. Thus, since P is a finite set of augmented clauses, we have I models P in G_3 logic iff I models $P_{\sim_{\mathrm{G}_3}}$ in Ł$_3$ logic.

We have showed that Łukasiewicz 3-valued logic can be used for knowledge representation based on logic programming. Now, consider again our example 2:

Suppose that both implication and negation correspond to Łukasiewicz's logic Ł$_3$. Then clearly M_2 would be a stable model of P, which fails the very basic notion of negation as failure in Logic Programming.

It is well accepted to consider negation as a failure a kind of modality operator, see for instance [19], which is considered as follows: $\sim_{\mathrm{G}_3} x := \Box \sim_{L_3} x$. We adopt this negation operator ($\sim_{\mathrm{G}_3}$) in the following definition.

Definition 1. Given a logic program P, we define a set of atoms $M \subseteq \mathcal{L}_P$ to be a Ł$_3$-stable model of P if $< P, M >_{\sim_{\mathrm{G}_3}} \Vdash_{\mathrm{L}_3} M$.

Here, we assume the use of Ł$_3$ implication which behaves differently to the one of G_3 logic. Furthermore, we will present a simple example that shows that in general the Ł$_3$-stables model of P do not correspond to the standard stable models. To see this, we will display the following Remark:

Remark 1. Let us start by considering the following program $Q = \{\sim_{\mathrm{G}_3} (a \to \bot)\}$. Then, it is not hard to see that Q does not have stable models in G_3 logic. In contrast, $\{a\}$ is a Ł$_3$-stable model of Q.

Theorem 3. Given an augmented logic program P and a set of atoms $M \subseteq \mathcal{L}_P$. M is a stable model of P iff M is a Ł$_3$-stable model of P.

Proof. Recall that M is a stable model of P iff $< P, M > \Vdash_{\mathrm{G}_3} M$ iff I_M is a model of $< P, M >$ in G_3 logic and M is a logical consequence of $< P, M >$ in G_3.

On the other hand, M is a Ł$_3$-stable model of P iff $< P, M >_{\sim_{\mathrm{G}_3}} \Vdash_{\mathrm{L}_3} M$ iff I_M is a model of $< P, M >$ in L_3 logic and M is a logical consequence of $< P, M >_{\sim_{\mathrm{G}_3}}$ in L_3.

We have already noticed that I_M is a model of $< P, M >$ in G_3 logic iff I_M is a model of $< P, M >_{\sim_{\mathrm{G}_3}}$ in L_3 logic, this is thanks to Theorem 2.

Moreover, M is a logical consequence of $< P, M >$ in G_3 iff $< P, M >_{\sim_{\mathrm{G}_3}}$ is a logical consequence of in L_3. Hence, M is a stable model of P iff M is a Ł$_3$-stable model of P as desired.

Notice that in the examples provided in Section 2.5, the theories belong to the class of augmented programs and therefore the stable models correspond to the Ł$_3$-stable models.

Lemma 1. Let I be a 3-valued interpretation and $S = \{\alpha \in \mathcal{L} : I(\alpha) = 1\}$. Let I_S be a 3-valued interpretation defined as:

$$I_S(\alpha) = \begin{cases} I(\alpha) & \text{if} \quad x \notin S, \\ 2 & \text{if} \quad x \in S \end{cases} \qquad (6)$$

Then, $I_S(\beta) = I(\sim_{\mathrm{G}_3}\sim_{\mathrm{G}_3} \beta)$ for every $\{\vee, \wedge, \neg\}$-formula β.

Proof. We will give the proof by induction on β. Indeed, let β be an atom. So, notice that if $I(\beta) \in \{0, 2\}$, then $I_S(\beta) = I(\beta)$ and if $I(\beta) = 1$, then $I_S(\beta) = 2$. Therefore $I_S(\beta) = I(\sim_{\mathrm{G}_3}\sim_{\mathrm{G}_3} \beta)$.

Now, let us suppose $I_S(\beta_i) = I(\sim_{\mathrm{G}_3}\sim_{\mathrm{G}_3} \beta_i)$ as induction hypothesis. Let β be $\beta_1 \# \beta_2$ where $\#$ can represent $\wedge$ or $\vee$. Then, $I_S(\beta) = I_S(\beta_1 \# \beta_2) = I_S(\beta_1) \# I_S(\beta_2)$.

So, due to the induction hypothesis we have that $I_S(\beta_1)\#I_S(\beta_2) = I(\sim_{G_3}\sim_{G_3}\beta_1)\#I(\sim_{G_3}\sim_{G_3}\beta_2) = I(\sim_{G_3}\sim_{G_3}\beta_1\#\sim_{G_3}\sim_{G_3}\beta_2) = I(\sim_{G_3}\sim_{G_3}(\beta_1\#\beta_2)) = I(\sim_{G_3}\sim_{G_3}\beta)$. Therefore, $I_S(\beta) = I(\sim_{G_3}\sim_{G_3}\beta)$.

Finally, let β be $\sim_{G_3}\beta_1$, then we have that $I_S(\beta) = I_S(\sim_{G_3}\beta_1) = \sim_{G_3} I_S(\beta_1) = \sim_{G_3} I(\sim_{G_3}\sim_{G_3}\beta_1) = I(\sim_{G_3}\sim_{G_3}\sim_{G_3}\beta_1) = I(\sim_{G_3}\sim_{G_3}\beta)$. Therefore, $I_S(\beta) = I(\sim_{G_3}\sim_{G_3}\beta)$ and the proof is completed.

Theorem 4. Given an augmented logic program P and a set of atoms $M \subseteq \mathcal{L}_P$. M is a stable model of P iff for every 3-valued interpretation I:

$$I\left(\bigwedge < P, M >_{\sim_{G_3}}\right) = I\left(\bigwedge\left(M\cup\sim_{G_3}\widetilde{M}\right)\right), \quad (7)$$

where the connective of implication used in the above expression could be any of the two defined in our 3-valued logics considered in this paper.

Proof. For this proof, we will consider the connective of implication of Ł$_3$ logic and let P and $M \subseteq \mathcal{L}_P$ be under the conditions of the hypothesis.

From left-to-right, let us suppose that M is a stable model of P. So, we have to prove that every 3-valued interpretation I fulfills that $I(\bigwedge < P, M >_{\sim_{G_3}}) = I(\bigwedge(M\cup\sim_{G_3}\widetilde{M}))$.

In the following, we will consider an arbitrary, but fixed 3-valued interpretation I, and suppose that:

– $I(\bigwedge(M\cup\sim_{G_3}\widetilde{M})) = 0$:

Then, $I(\bigwedge(M)) = 0$, or $I(\bigwedge(\sim_{G_3}\widetilde{M})) = 0$. If $I(\bigwedge(\sim_{G_3}\widetilde{M})) = 0$, then automatically $I(\bigwedge < P, M >_{\sim_{G_3}}) = 0$. Moreover, if $I(\bigwedge(M)) = 0$, then $\exists\alpha \in M : I(\alpha) = 0$. So, $\exists\alpha \in M : I(\sim_{G_3}\sim_{G_3}\alpha) = 0$ and then $I(\bigwedge(\sim_{G_3}\sim_{G_3} M)) = 0$. Consequently, $I(\bigwedge < P, M >_{\sim_{G_3}}) = 0$.

– $I(\bigwedge(M\cup\sim_{G_3}\widetilde{M})) = 1$:

Notice that $\nexists\alpha \in \mathcal{L}_P : I(\sim_{G_3}\alpha) = 1$, then it must happen that $I(\bigwedge(\sim_{G_3}\widetilde{M})) = 2$ and $I(\bigwedge(M)) = 1$. Thus, $\forall\alpha \in M$ we have that $I(\alpha) = 1$, or $I(\alpha) = 2$ and then, $\exists\alpha \in M$ such that $I(\alpha) = 1$. Consequently, $\forall\alpha \in M$ we have that $I(\sim_{G_3}\sim_{G_3}\alpha) = 2$ and then $I(\bigwedge(\sim_{G_3}\sim_{G_3} M)) = 2$. Furthermore, $\forall\alpha \in \mathcal{L}_P \setminus M : I(\sim_{G_3}\alpha) = 2$ then $\forall\alpha \in \mathcal{L}_P \setminus M : I(\alpha) = 0$.

Recall the interpretation I_M previously defined as: $I_M(x) = 2$ if $x \in M$, and $I_M(x) = 0$ if $x \in \mathcal{L}_P \setminus M$. Now, let us consider the interpretation I_S defined in Lemma 1, notice that $I_S = I_M$.

Since M is a stable model of P, I_M is a model of $< P, M >_{\sim_{G_3}}$, then $\forall p \in P : I_M(p) = 2$. Recall that P is a set of augmented clauses and an augmented clause has the form $H \leftarrow B$, where H and B are $\{\vee, \wedge, \neg\}$-formulas.

Consider an augmented clause $p \in P$ of the form $H_p \leftarrow B_p$ such that $I(p) = 0$, then $I(H_p) = 0$ and $I(B_p) = 2$, therefore $I_M(H_p) = I_S(H_p) = 0$ and $I_M(B_p) = I_S(B_p) = 2$, then $I_M(p) = 0$ which is a contradiction. Therefore $\nexists p \in P : I(p) = 0$ and then $I(\bigwedge(P)) \neq 0$.

In this setting, we also have that $I(\bigwedge(P)) \neq 2$ because if $I(\bigwedge(P)) = 2$, then $I(\bigwedge < P, M >_{\sim_{G_3}}) = 2$, and M is a stable model of P then $I(\bigwedge(M)) = 2$ which is a contradiction. Consequently, $I(\bigwedge(P)) = 1$ and therefore $I(\bigwedge < P, M >_{\sim_{G_3}}) = 1$.

– $I(\bigwedge(M\cup\sim_{G_3}\widetilde{M})) = 2$:

Then, $I(\bigwedge(M)) = 2$ and $I(\bigwedge(\sim_{G_3}\widetilde{M})) = 2$. Consequently, $\forall\alpha \in M : I(\alpha) = 2$, and also $\forall\alpha \in \mathcal{L}_p \setminus M : I(\sim_{G_3}\alpha) = 2$ then $\forall\alpha \in \mathcal{L}_p \setminus M : I(\alpha) = 0$. Thus, $I = I_M$. Besides, M is a stable model of P, then I_M is a model of $< P, M >_{\sim_{G_3}}$. As a result, $I(\bigwedge < P, M >_{\sim_{G_3}}) = 2$.

Therefore, every 3-valued interpretation maintains the desired equality. Then, the first condition is proved.

Now, let us prove the right-to-left direction. First, let I be an arbitrary 3-valued interpretation such that $I(\bigwedge < P, M >_{\sim_{G_3}}) = I(\bigwedge(M\cup\sim_{G_3}\widetilde{M}))$. Thus, we have to prove that M is a stable model of P. Indeed, consider the interpretation I_M.

The value of the conjunction of all elements of M under I_M is 2, that is, $I_M(\bigwedge(M)) = 2$. Moreover, the value of all the atoms that are not elements of M under I_M is 0, then the value of their negation under I_M is 2, that is, $I_M(\bigwedge(\sim_{G_3}\widetilde{M})) = 2$.

Consequently, $I_M(\bigwedge(M\cup\sim_{G_3}\widetilde{M})) = 2$, and then $I_M(\bigwedge < P, M >_{\sim_{G_3}}) = 2$. Therefore, I_M is a model of $< P, M >_{\sim_{G_3}}$.

Let I be a model of $< P, M >_{\sim_{G_3}}$, namely, let I be a 3-valued interpretation such that $I(\bigwedge < P, M >_{\sim_{G_3}}) = 2$. Consequently, $I(\bigwedge(M \cup \sim_{G_3} \widetilde{M})) = 2$ and then $I(\bigwedge(M)) = 2$.

This means that every formula $\alpha \in M$ is a logical consequence of $< P, M >_{\sim_{G_3}}$. Therefore, we have that $< P, M >_{\sim_{G_3}} \Vdash M$, thus M is a stable model of P. Hence, the second condition and the theorem have been proved.

Let us observe that in the proof of Theorem 4, the implication of Ł$_3$ logic has been considered. The following Corollary is an immediate consequence of the last Theorem:

Corollary 1. Given an augmented logic program P and a set of atoms $M \subseteq \mathcal{L}_P$. M is a stable model of P iff:

$$\bigwedge < P, M >_{\sim_{G_3}} \leftrightarrow \bigwedge(M \cup \sim_{G_3} \widetilde{M}). \tag{8}$$

Is a tautology in any of our 3-valued logics considered in this paper.

3.2 Generalization

In this subsection, we will work on the family of logics G_n and Ł$_n$, for $n \geq 3$, see [5, 3, 13].

Recall that in the logic Ł$_n$, it is possible to define the binary operators $\oplus$ and $\otimes$ as follows:

$$x \oplus y = \neg x \rightarrow y, \tag{9}$$

$$x \otimes y = \neg(\neg x \oplus \neg y). \tag{10}$$

Furthermore, we can define the operator $\square$ for the n-valued logic Ł$_n$ recursively as follows:

$$\square x = x^{(n+1)}. \tag{11}$$

It is not hard to see that $\sim_{G_n} x = \square \sim_{L_n} x$. We again adopt this negation operator ($\sim_{G_n}$) in Definition 2.

Theorem 5. Given an augmented theory P and an n-valued interpretation I, I models P in G_n logic iff I models $P_{\sim_{G_n}}$ in Ł$_n$ logic.

Proof. Let C be an augmented clause and I an n-valued interpretation. By hypothesis, we have that C has the form $H \leftarrow B$, where both H and B are $\{\vee, \wedge, \neg\}$-formulas. Then, $I(H)$ and $I(B)$ do not change their value when considering the two different logics.

Since I is an n-valued interpretation and $n-1$ is the unique designated value, let us assume that I models C in G_n logic. So, $B \rightarrow H = n-1$ in G_n.

From the latter and the order defined through the implication on G_n, we have that $I(B) \leq I(H)$. So, we have that:

– $0 \leq I(H) - I(B)$.

– $n-1 \leq n-1+I(H)-I(B)$.

– $\min(n-1, n-1-I(B)+I(H)) = n-1$.

Therefore, we can infer that that $B \rightarrow H = n-1$ in Ł$_n$ logic. Thus, I models $C_{\sim_{G_n}}$ in Ł$_n$ logic if I models C in G_n logic as desired.

Conversely, by an analogous argument but now using the definition of the implication on G_n and the order relation given through Łukasiewicz implication, we have I models C in G_n logic every time that I models $C_{\sim_{G_n}}$ in Ł$_n$ logic, which completes the proof.

Now, we are in conditions to define the notion Ł$_n$-stable model via the following definition in which the use of Ł$_n$ implication is involved. Indeed:

Definition 2. Given a logic program P, we define a set of atoms $M \subseteq \mathcal{L}_P$ to be a Ł$_n$-stable model of P if $< P, M >_{\sim_{G_n}} \Vdash_{\text{Ł}_n} M$.

Here, we assume the use of Ł$_n$ implication, which behaves differently to the one of G_n.

Theorem 6. Given an augmented logic program P and a set of atoms $M \subseteq \mathcal{L}_P$. M is a stable model of P iff M is a Ł$_n$-stable model of P.

Proof. Having in mind Theorem 1, we know that M is a stable model of P iff $< P, M > \Vdash_{G_n} M$ iff I_M is a model of $< P, M >$ in G_n logic and M is a logical consequence of $< P, M >$ in G_n.

By Definition 2, we have that M is an Ł$_n$-stable model of P iff $< P, M >_{\sim_{G_n}} \Vdash_{\text{Ł}_n} M$ iff I_M is a model of $< P, M >$ in $\mathrm{Ł}_n$ logic and M is a logical consequence of $< P, M >_{\sim_{G_3}}$ in $\mathrm{Ł}_n$.

Taking into account Theorem 5, we obtain that I_M is a model of $< P, M >$ in G_n logic iff I_M is a model of $< P, M >_{\sim_{\mathrm{G}_n}}$ in $\mathrm{Ł}_n$ logic. Thus, M is a logical consequence of $< P, M >$ in G_n iff $< P, M >_{\sim_{\mathrm{G}_n}}$ is a logical consequence of in $\mathrm{Ł}_n$, which completes the proof.

4 Conclusion and Future Work

ASP has proven to be a powerful tool to encode and solve problems in both academia and industry, becoming one of the most attractive KRR frameworks [8].

Therefore, ASP has applications in many fields, and it is widely used for knowledge representation due to its effectiveness.

In this paper, we have shown that Łukasiewicz logic can be used for knowledge representation based on logic programming.

Furthermore, we have also presented an alternative definition of stable models for general programs in terms of Ł$_3$ logic and we have studied the circumstances in which they are equivalent to the standard stable models.

We have taken advantage of the fact that on Łukasiewicz's logic Ł$_3$ is possible to express the negation on G_3 logic via the use an specific modal operators. Finally, we have defined Ł$_n$-stable models and we have generalized the results previously obtained to many-valued logics.

As future work, we will focus on the study of the correspondence of standard state models and Ł$_3$ stable models when the theories do not belong to the class of augmented programs.

Remark 1 shows us that if we take more general programs, we can obtain more answers on Ł$_3$. Moreover, a generalization of Theorem 4 for multivalued logics will also be studied.

Besides, the theory presented in this paper will be studied further with the aim of extending our characterization to model preferences in a natural way, following the work presented in [4].

Another important point that we are interested in is giving a notion of stable semantics in terms of MV$_n$-algebras which would imply translating the result given here for generating algebra for a more general setting.

References

1. **Baaz, M. (1996).** Infinite-valued Gödel logics with 0-1-projections and relativizations. Lecture Notes Logic, pp. 23–33.
2. **Cignoli, R. (1982).** Proper n-valued Lukasiewicz algebras as s-algebras of Lukasiewicz n-valued propositional calculi. Studia Logica, Vol. 41, pp. 3–16.
3. **Cignoli, R., D'Ottaviano, I., Mundici, D. (2000).** Algebraic Foundations of Many-Valued Reasoning. Springer Netherlands. DOI: 10.1007/978-94-015-9480-6.
4. **Confalonieri, R., Nieves, J. C., Osorio, M., Vázquez-Salceda, J. (2010).** Possibilistic semantics for logic programs with ordered disjunction. Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 133–152. DOI: 10.1007/978-3-642-11829-6_11.
5. **Coniglio, M. E., Esteva, F., Godo, L. (2016).** On the set of intermediate logics between the truth- and degree-preserving łukasiewicz logics. Logic Journal of IGPL, Vol. 24, No. 3, pp. 288–320. DOI: 10.1093/jigpal/jzw006.
6. **Esteva, F., Godo, L., Hájek, P., Navara, M. (2000).** Residuated fuzzy logic with an involutive negation. Archive for Mathematical Logic, Vol. 39, pp. 103–124. DOI: 10.1007/s001530050006.
7. **Esteva, F., Godo, L., Montagna, F. (2001).** The ł $\prod$ and ł $\prod \frac{1}{2}$ logics: Two complete fuzzy systems joining Lukasiewicz and product logics. Archive for Mathematical Logic, Vol. 40, pp. 39–67.
8. **Falkner, A., Friedrich, G., Schekotihin, K., Comploi-Taupe, R., Teppan, E. (2018).** Industrial applications of answer set programming. Künstliche Intelligenz, Vol. 32, pp. 165–176. DOI: 10.1007/s13218-018-0548-6.
9. **Gebser, M., Kaufmann, B., Schaub, T. (2012).** Conflict-driven answer set solving: From theory to practice. Artif. Intell., Vol. 187, pp. 52–89. DOI: 10.1016/j.artint.2012.04.001.
10. **Gelfond, M. (1987).** On stratified autoepistemic theories. Proceedings of the 6th National Conference on Artificial Intelligence. Seattle, WA, USA, July 1987, pp. 207–211.
11. **Gelfond, M., Lifschitz, V. (1988).** The Stable Model Semantics for Logic Programming. 5th Conference on Logic Programming, pp. 1070–1080.

ISSN 2007-9737

12. **Grigolia, R. (1977).** Algebraic analysis of łukasiewicz-Tarski's n-valued logic systems. In Selected papers on Łukasiewicz sentencial calculi. Ossolineum, Wroclaw, pp. 81–92.
13. **Hájek, P. (1998).** Metamathematics of fuzzy logic.
14. **Hájek, P. (2006).** Fuzzy logic. plato.stanford.edu/entries/logic-fuzzy/#4.
15. **Klir, G. J., Yuan, B. (1995).** Fuzzy Sets and Fuzzy Logic: Theory and Applications.
16. **Lifschitz, V., Tang, L. R., Turner, H. (1999).** Nested expressions in logic programs. Annals of Mathematics and Artificial Intelligence, Vol. 25, pp. 369–389.
17. **Minari, P. (2003).** A note on Łukasiewicz's three-valued logic. Annali del Dipartimento di Filosofia dell'Università di Firenze, pp. 163–190.
18. **Osorio, M., Carranza, J. L. C. (2020).** An extension of the stable semantics via łukasiewicz logic. Electronic Notes in Theoretical Computer Science, Vol. 354, pp. 141–155. DOI: 10.1016/j.entcs.2020.10.011.
19. **Osorio, M., Navarro, J., Arrazola, J., Borja, V. (2005).** Ground nonmonotonic modal logic S5: New results. Journal of Logic and Computation, Vol. 15, No. 5, pp. 787–813. DOI: 10.1093/logcom/exi042.
20. **Osorio, M., Navarro, J. A., Arrazola, J. (2005).** Safe beliefs for propositional theories. Annals of Pure and Applied Logic, Vol. 134, No. 1, pp. 63–82.
21. **Osorio, M., Navarro Pérez, J. A., Arrazola, J. (2005).** Safe beliefs for propositional theories. Annals of Pure and Applied Logic, Vol. 134, No. 1, pp. 63–82.
22. **Osorio, M., Zepeda, C., Carballido, J., López, D. (2010).** Yet another application of fuzzy logic. 20th International Conference on Electronics, Communications and Computer, pp. 217–221. DOI: 10.1109/CONIELECOMP.2010.5440764.
23. **Pearce, D. (1999).** From Here to There: Stable Negation in Logic Programming. In What Is Negation? pp. 161–181.
24. **Zadeh, L. A. (1965).** Fuzzy sets. Information and Control, Vol. 8, pp. 338–353.

Article received on 09/04/2022; accepted on 06/06/2022.
Corresponding author is Mauricio Osorio.