An extension of the stable semantics via Lukasiewicz logic

Mauricio Osorio[1] and José Luis Carballido[2]

[1] Universidad de las Américas-Puebla,
`osoriomauri@gmail.com`
[2] Benemérita Universidad Atónoma de Puebla

Abstract. Logic Programming and fuzzy logic are active areas of research,and their scopes in terms of applications are growing fast. Fuzzy logic is a branch of many-valued logic based on the paradigm of inference under vagueness. In this work we recall some of the interplay between three 3-valued logics that are relevant in these areas: The Lukasiewicz logic, the intermediate logic G_3 and the paraconsistent logic G'_3, and we present a contribution to the area of answer sets that consists in extending a definition of stable model based on proof theory in logic G_3, to a more general definition that can be based on any of the more expressive logics G'_3 or Lukasiewicz.

Keywords: Knowledge representation, stable semantics, paraconsistency.

1 Introduction

The stable semantics allows us to handle problems with default knowledge and produce non-monotonic reasoning using the concept of negation as failure. The p-stable semantics is an alternative semantics, except that in some cases offers models where the stable semantics has none. There are two popular software implementations to compute the stable models: `dlv`[3] and `smodels`[4]. 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, there exist different approaches for knowledge representation based on the p-stable semantics, such as, updates [14], preferences [15], and argumentation [1]. Currently, in [16], a schema for the implementation of the p-stable semantic using two well known open source tools: Lparse and Minisat is described. The authors also present a prototype[5] written in Java of a tool based on that schema.

The term *fuzzy logic* emerged in the development of the theory of fuzzy sets by Lotfi A. Zadeh (1965) [3]. It is generally agreed that an important point in the evolution of the modern concept of uncertainty was the publication of the

[3] http://www.dbai.tuwien.ac.at/proj/dlv/
[4] http://www.tcs.hut.fi/Software/smodels/
[5] http://cxjepa.googlepages.com/home

seminal paper by Lotfi A. Zadeh in 1965 [20], where he introduced a theory whose objects, called fuzzy-sets, have boundaries that are not precise. The membership in a fuzzy-set is not a matter of affirmation or denial, but rather a matter of a degree. Although the concept of uncertainty had been studied by philosophers, the significance of Zadeh's paper was that it challenged not only probability theory as the sole agent for uncertainty, but 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 [4].

We can distinguish two main directions in fuzzy logic [3]. The first one corresponds to *fuzzy logic in the broad sense*, it serves mainly as apparatus for fuzzy control, analysis of vagueness in natural language and several other application domains. The second one corresponds to *fuzzy logic in the narrow sense* which is symbolic logic with a comparative notion of truth developed fully in the spirit of classical logic. So, this fuzzy logic has syntax, semantics, axiomatization, truth-preserving deduction, completeness, etc. It is a branch of many-valued logic based on the paradigm of inference under vagueness. This last direction in fuzzy logic is a relatively recent discipline, both serving as a foundation for the fuzzy logic in a broad sense and of independent logical interest, since it turns out that strictly logical investigation of this kind of logical calculi can go rather far (interested readers can see [2,9]). Currently it is possible to find strong formal systems of fuzzy logic, such as, Łukasiewicz, Gödel and product logic [3].

In particular, Łukasiewicz and Post gave the first published systematic descriptions of many-valued logical systems in the modern era [19]. Łukasiewicz argued 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, he proposed to reject the law of excluded middle, that is, the assumption that every proposition is true or false. Moreover, he proposed a logic system where a third truth-value is added, which is read as *possible*. The Łukasiewicz logic [19] is a non-classical, many-valued logic. It was originally defined as a three-valued logic, denoted by $Ł_3$, and as we mentioned, it belongs to the classes of fuzzy logics. Afterwards, Łukasiewicz generalized his three-valued logic to n values and also to an infinite-valued system [19]. In this paper, we consider the $Ł_3$ logic in order to show a non-standard application of fuzzy logic. We show how Łukasiewicz logic can be used for knowledge representation based on logic programming. Our results are based on the fact that the stable semantics and the p-stable semantics can be expressed in terms of similar expressions involving the G_3 logic and the G'_3 logic respectively, and the fact that the Łukasiewicz logic can express these two logics.

Our paper is structured as follows. In section 2, we summarize some definitions, logics and semantics necessary to understand this work. In section 3, we show how to express the stable and the p-stable semantics of normal programs in terms of Łukasiewicz logic. We present a definition of stable model for more general programs in terms of the intermediate logic G_3, and then we extend such definition, in a conservative way, in terms of the paraconsistent logic G'_3. 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: $p_0,\ p_1,\ \ldots;$
connectives: $\wedge,\ \vee,\ \leftarrow,\ \neg;$ and
auxiliary symbols: $(,\ ),$

where $\wedge$, $\vee$, $\leftarrow$ are 2-place connectives and $\neg$ is a 1-place connective. Formulas are built up as usual in logic. A *literal* is either an atom a, called *positive literal*; or the negation of an atom $\neg a$, called *negative literal*.

A *normal* clause is a clause of the form

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

where a and each of the b_i are atoms for $1 \leq i \leq n+m$. We define a *normal program* P, as a finite set of normal clauses.

The body of a normal clause could be empty, in which case the clause is known as a *fact* and can be denoted just by: $a \leftarrow$.

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 $\neg\widetilde{M} = \{\neg a \mid a \in \mathcal{L}\backslash M\}$.

Since we shall restrict our discussion to propositional programs, we take for granted that programs with predicate symbols are only an abbreviation of the ground program.

2.2 Logics

We review some logics that are relevant in this paper to characterize different semantics of normal and more general programs.

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.

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 by $Ł_3$, we first have to modify the syntax of our formulas to allow, as primitive connectives, only: the 0-place connective $\bot$ (*failure*) and the 2-place connective $\to$ (*implication*). These connectives operate over a domain $D = \{0, 1, 2\}$, with 2 as the unique designated value, and are defined as follows:

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

Other connectives in $Ł_3$ are introduced in terms of $\bot$ and $\to$ as follows:

$$\neg A := A \to \bot \qquad\qquad \top := \neg\bot$$
$$A \vee B := (A \to B) \to B \qquad\qquad A \wedge B := \neg(\neg A \vee \neg B)$$
$$\Box A := \neg(A \to \neg A) \qquad\qquad \Diamond A := \neg A \to A$$

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 [8], where the behavior of modal operators is checked against some of the relevant modal principles. Let us observe that, in logic $Ł_3$, the formula $(a \wedge \neg a) \to b$ is

x	$\neg x$	$\Box x$	$\Diamond x$
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

Table 1. Truth tables of connectives in $Ł_3$.

not a tautology, which implies a paraconsistent behavior.

Now we present an axiomatization of $Ł_3$ [19]:

(1) $(p \to q) \to ((q \to r) \to (p \to r))$
(2) $(\neg p \to \neg q) \to (q \to p)$
(3) $((p \to \neg p) \to p) \to p$,

using the rules of substitution and *modus ponens*.

Łukasiewicz generalized his three-valued logic to n values and also to an infinite-valued system [19]. The matrix for the infinite-valued system is defined on the rational numbers in the unit interval from 0 to 1. For x, y in the interval, we have: $x \to y = min(1, 1 - x + y)$, $\neg x = 1 - x$. If instead of the whole rational interval, a finite subset closed under the above functions is chosen, the result is a set of $n-$valued Łukasiewicz connectives. For more general results in Łukasiewicz logics, including the case where a different set of designated values is adopted in $Ł_m$, the reader is referred to [18].

G$'_3$ logic It is defined as a three-valued logic with truth values in the domain $D = \{0, 1, 2\}$ where 2 is the designated value. The evaluation functions of the logic connectives are 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. An axiomatization of G$'_3$ is given in [10].

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 G$'_3$.

G$_3$ logic Gödel defined, in fact, a family of many-valued logics G$_i$ with truth values over the domain $D = \{0, 1, \ldots, i-1\}$ and with $i-1$ as the unique designated value. Logic connectives are defined as:

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

The only difference between the true tables of G$_3$ and G$'_3$ is the negation of the value 1 that is 0 in G$_3$.

A Hilbert style version of this logic G$_3$ is obtained from intuitionistic logic [7] by adding the following axiom:$(\neg b \rightarrow a) \rightarrow (((a \rightarrow b) \rightarrow a) \rightarrow a)$. Therefore the set of theorems in this logic is the set of tautologies of Gödel's 3-valued logic G$_3$.

From now on we will denote by $\neg$ the negation of G$'_3$ and by $\sim$ the negation of G$_3$.

Classical logic Classical logic, C, is obtained from intuitionistic logic [7] by adding the following axiom: $(\neg a \rightarrow a) \rightarrow a$. 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 well known standard "truth table" logic of two values [6].

2.3 Semantics

From now on, we assume that the reader is familiar with the notion of classical minimal model [6].

We present the characterization of the stable semantics for normal programs in terms of the G_3 logic. A similar characterization exists for p-stable semantics in terms of logic G'_3.

We use the notation $\Vdash_X P$ to indicate that formula P is a theorem or a tautology in logic X depending on logic X. For a finite family of formulas $Q =$

$q_1, q_2, ..., q_n$ and a set of atoms M, the expression $Q \Vdash_X M$ will mean that M is a classical model of Q and that $\Vdash_X q_1 \rightarrow (q_2 \rightarrow (...(q_n \rightarrow m)...)$ for each m in M

Stable semantics From now on, we assume that the reader is familiar with the notion of classical minimal model [6].

The characterization of the stable semantics for normal programs in terms of logic G_3 is given in the following definition.

Definition 1. *[17] Given a normal program P, a set of atoms $M \subseteq \mathcal{L}_P$ is a stable model of P if $P \cup \sim \widetilde{M} \Vdash_{G_3} M$.*

Let us observe that, in particular, M is also a classical model of P as we mentioned.

Example 1. Let P be the following normal program:

$$\{a \leftarrow\sim b, \;\; b \leftarrow\sim a\}$$

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

2.4 Defining $\mathrm{G_3}$ and $\mathrm{G_3'}$ via Ł$_3$

One of our motivations to study Łukasiewicz's Ł$_3$ logic is the fact that it is able to express the semantics of other logics such as Gödel's $\mathrm{G_3}$ and $\mathrm{G_3'}$.

This subsection presents some results from [12]. We first define, in Table 3, the implication and negation connectives for $\mathrm{G_3}$ and $\mathrm{G_3'}$ via Ł$_3$ (connectives that are not subscripted correspond to Ł$_3$). Table 4 shows the truth tables of these connectives for the $\mathrm{G_3}$ and $\mathrm{G_3'}$ logics. The reader can easily verify that the definitions here given 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. Hence, these two connectives have the same semantics in Ł$_3$, $\mathrm{G_3}$, and $\mathrm{G_3'}$ logics.

$$\begin{array}{ll}
\sim_{\mathrm{G_3}} x & := \Box\neg x \\
x \rightarrow_{\mathrm{G_3}} y & := (x \rightarrow y) \wedge \sim_{\mathrm{G_3}}\sim_{\mathrm{G_3}} (\sim_{\mathrm{G_3}}\sim_{\mathrm{G_3}} x \rightarrow y) \\
\neg_{\mathrm{G_3'}} x & := \neg\Box x \\
x \rightarrow_{\mathrm{G_3'}} y & := x \rightarrow_{\mathrm{G_3}} y
\end{array}$$

Table 3. Connectives of $\mathrm{G_3}$ and $\mathrm{G_3'}$ in Ł$_3$.

x	$\sim_{G_3} x$	$\neg_{G_3'} x$
0	2	2
1	0	2
2	0	0

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

Table 4. Truth tables: connectives in G_3 and G_3'.

2.5 The $X - or$ operator

An interesting property we try to express in terms of our logics, is a characterization of the $X - or$ operator endowed with an encoding-decoding property. In its original form, this operator works as an exclusive disjunction in two variables: 0 (false) and 1 (true). When implementing this particular generalization to three true values we obtain the arrangement given in table 5.

Let us denote the new $X - or$ operator by the symbol $\ominus$. We can characterize this operator by the following three properties in order to obtain the values we need. Observe that the last equation reflects the encoding-decoding property and also that the commutativity of the operator follows from the equations shown below.

$$x \ominus 0 = x$$
$$x \ominus x = 0$$
$$(x \ominus y) \ominus y = x$$

According to these properties we have from $(1 \ominus 1) \ominus 1 = 1$ and $(2 \ominus 2) \ominus 2 = 2$, that $0 \ominus x = x$, furthermore from $(1 \ominus 2) \ominus 2 = 1$ it follows that $1 \ominus 2 = 1$. A similar argument shows that $2 \ominus 1 = 2$.

Therefore we get the Table 5 for the $X - or$ operator in three truth values.

$\ominus$	0	1	2
0	0	1	2
1	1	0	1
2	2	2	0

Table 5. The $X - or$ operator.

What we want next is to express this truth table as a function of two variables, however this is not possible in logic G_3, since as it is well known, any function f of two variables defined in this logic has the property that $f(2,2) = 2$ whenever $f(1,2) = 1$ [13], and according to the table $f(2,2)$ must be 0.

Let us remember that the symbol $\neg$ is the G_3'-negation, the symbol $\sim$ is the G_3-negation, and that $\sim x = x \to (\neg x \wedge \neg\neg x)$, then in G_3' the $X - or$ operator is expressed by the following formula:

$(x \vee y) \wedge (\neg x \vee \neg y) \wedge (x \vee \sim x) \wedge ((\neg\neg x \vee \sim x) \vee (\neg\neg y \vee \sim y))$

3 Expressing p-stable and stable semantics based on Ł$_3$

Here, we show how to express stable semantics via Ł$_3$ logic. This is possible due to two facts: Ł$_3$ logic is able to express the semantics of G$_3$ logic, and the stable semantics is defined in terms of this logic.

We first define a function that obtains the clause that results when we substitute the G$_3$ connectives by the Ł$_3$ connectives from a given clause.

Given a clause r expressed in terms of the G$_3$ connectives, we define $Trad_{G_3to\text{Ł}_3}(r)$ as the clause that results when we substitute the G$_3$ connectives for the Ł$_3$ connectives according to Table 3. Given a normal program P, we define $Trad_{G_3to\text{Ł}_3}(P)$ as the set $\{Trad_{G_3to\text{Ł}_3}(r) \mid r \in P\}$.

Example 2. Let us consider the program $P_1 = P \cup \{\sim b\}$ where P is the program of Example 1, namely the program

$$\begin{array}{l} a \leftarrow \sim b. \\ b \leftarrow \sim a. \\ \sim b. \end{array}$$

We can see that $L_1 = Trad_{G_3to\text{Ł}_3}(P_1)$ is the following program:

$$\begin{array}{l} (\Box\neg b \to a) \wedge \Box\neg\Box\neg(\Box\neg\Box\neg\Box\neg b \to a). \\ (\Box\neg a \to b) \wedge \Box\neg\Box\neg(\Box\neg\Box\neg\Box\neg a \to b). \\ \Box\neg b. \end{array}$$

Now we present the definition of the stable semantics based on the definition of G$_3$ via Ł$_3$. Similar results are obtained for the p-stable semantics and logic G$'_3$.

Theorem 1. *[12] Given a normal program P, a set of atoms $M \subseteq \mathcal{L}_P$ is a stable model of P if $Trad_{G_3to\text{Ł}_3}(P \cup \sim \widetilde{M}) \Vdash_{\text{Ł}_3} M$.*

Proof. According to the relations between the Ł$_3$ connectives and the G$_3$ connectives, and our interpretation of the symbol $\Vdash_X$, we have that the relation $P \cup \sim \widetilde{M} \Vdash_{G_3} M$ is equivalent to $Trad_{G_3to\text{Ł}_3}(P \cup \sim \widetilde{M}) \Vdash_{\text{Ł}_3} M$.

Example 3. Let us consider the program P of Example 1. Let $M_1 = \{a\}$ and $M_2 = \{b\}$. Then we have that $Trad_{G_3to\text{Ł}_3}(P \cup \{\sim b\}) \Vdash_{\text{Ł}_3} \{a\}$ and $Trad_{G_3to\text{Ł}_3}(P \cup \{\sim a\}) \Vdash_{\text{Ł}_3} \{b\}$ where $Trad_{G_3to\text{Ł}_3}(P \cup \{\sim b\})$ corresponds to the program L_1 of Example 2. Hence, M_1 and M_2 are stable models of P as we obtained in Example 1.

Next we take advantage of the fact that logic G$_3$ can be expressed in terms of paraconsistent logic G'_3 to provide a definition, by way of paraconsistency, that extends the concept of stable model to general programs. In order to do this, we state a general definition of stable model in terms of G$_3$ logic, which in turn generalizes the definition we have for normal programs. It is important to observe that in the following formula we can use intuitionistic logic instead of G$_3$ logic.

Definition 2. *[11] 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 $P \cup \sim \widetilde{M} \cup \sim\sim M \Vdash_{\mathrm{G}_3} M$.*

As expected, this definition is a generalization of definition 1 for normal programs; it uses the fact that the double negation of the atoms in the set M are added as premises in the equation.

As it is well known, it is desirable that replacing equivalent formulas as parts of programs would leave the same stable models as in the original program. This property does not hold if we start with two programs that have the same stable models. For example, take the two programs $P_1 = \{a \leftarrow\sim b\}$ and $P_2 = \{a\}$ with the same stable models, and now consider the two programs $Q_1 = \{a \leftarrow\sim b, b\}$ and $Q_2 = \{a, b\}$ originated from the previous two programs after adding the same atom to both of them, these two programs are no longer equivalent according to the next definition.

Definition 3. *Two programs P_1 and P_2 are equivalent if they have the same stable models.*

In [5] the authors present the following convenient definition which is stronger than the previous one.

Definition 4. *Two programs P_1 and P_2 are strongly equivalent if for every program P, $P_1 \cup P$ and $P_2 \cup P$ have the same stable models.*

In order to present the generalization of the stable semantics in terms of paraconsistency, we remind the reader that the symbol $\sim$ that appears in the next definition is the negation of G_3 and the program P is expressed in terms of logic G_3'.

Definition 5. *Let P be a program defined in the language of G_3' and $M \subseteq \mathcal{L}_P$ be a set of atoms. We say that M is an L-stable model of P if $Trad_{G_3to\text{Ł}_3}(P \cup \sim \widetilde{M} \cup \sim\sim M) \Vdash_{\text{Ł}_3} M$*

We observe that this definition is a conservative extension of Definition 2 since the language of logic G_3 is fully expressed in terms of the paraconsistent logic G_3' as previously noted. Also observe that the G_3'-negation appears only in the program P.

As a consequence of this definition we have the following result for L-stable models of programs expressed in the G_3'-language.

Theorem 2. *Let P and Q programs expressed in the language of logic G_3'. If $P \equiv_{G_3'} Q$, then they are strongly equivalent in the context of L-stable models.*

Proof. The result follows from Definition 5 since $Trad_{G_3to\text{Ł}_3}(P \cup \sim \widetilde{M} \cup \sim\sim M)$ and $Trad_{G_3to\text{Ł}_3}(Q \cup \sim \widetilde{M} \cup \sim\sim M)$ are equivalent in Ł_3.

As a natural consequence that follows from our definitions we have:

Theorem 3. *Let P be a program expressed in the original language of the stable semantics, then Definitions 2 and 5 provide the same sets as stable and L-stable models respectively.*

Proof. We only need to observe that a program in the original language of the answer set semantics can be interpreted as written in terms of the G_3 language, then the conclusion follows from the relation that exists between G_3 and $Ł_3$.

Finally, as an example let us compute the L-stable models of the program given by the formula $x \ominus y$. This program is expressed below in terms of its clauses

$$\begin{array}{l} x \vee y \\ \neg x \vee \neg y \\ x \vee \sim x \\ (\neg\neg x \vee \sim x) \vee (\neg\neg y \vee \sim y) \end{array}$$

The language of the program is $\{x, y\}$. According to Definition 5, a set of atoms M is a L-stable model of P if $Trad_{G_3 to Ł_3}(P \cup \sim \widetilde{M} \cup \sim\sim M) \Vdash_{Ł_3} M$. We work with logic G'_3 whose language is more natural. Note also that the expression $Q \Vdash_X M$ as defined in the background, is equivalent to $(q_1 \wedge q_2 \wedge ...q_n) \to m$ for each m in M when X is G'_3 or $Ł_3$.

Let us propose $M = \{x\}$, then $\sim \widetilde{M} = \sim y$ and $\sim\sim M = \sim\sim x$

The left hand side of this implication is a conjunction of the four rules that express $x \ominus y$ plus the two rules or facts: $\sim y$ and $\sim\sim x$

Let us assume that for certain valuation x takes the value 1, then the third rule of the antecedent takes the value 1 and therefore the antecedent cannot take the value 2. Now if we assume that the atom x takes the value 0, then the last rule of the antecedent takes the value 0 and so does the antecedent. We conclude that the implication is a tautology in G'_3 and the set $M = \{x\}$ is a L-stable model.

Now, if we try $M = \{y\}$, then the last two rules of the implication we are working with, become: $\sim x$ and $\sim\sim y$.

Let us assume that for certain valuation y takes the value 1, then the first and fifth rules of the antecedent cannot be 2 at the same time. In the case the atom y takes the value 0 the last rule of the antecedent is 0 too. Therefore the implication is a tautology and the set $\{y\}$ is a L-stable model too.

In a similar way we can see that the valuation $x = 2, y = 1$ shows that the set $M = \{x, y\}$ is not a L-stable model.

4 Conclusions

We show that Łukasiewicz logic can be used for knowledge representation based on logic programming. We review how two useful semantics to represent knowledge, stable and p-stable, are characterized via $Ł_3$ logic. In particular we take advantage of the fact that Łukasiewicz's logic $Ł_3$ is able to express G_3 and G'_3

logics, which characterize those two semantics for normal programs respectively. We also present the definition of stable model for general programs in terms of logic G_3 and extend it for programs in the languages of G'_3 and Ł$_3$. Finally we present the concept of strong equivalence in the new context of paraconsistency.

References

1. J. L. Carballido, J. C. Nieves, and M. Osorio. Inferring Preferred Extensions by Pstable Semantics. *Revista Iberomericana de Inteligencia Artificial*, 13(41):38–53, 2009.
2. P. Hájek. *Metamathematics of fuzzy logic.* Kluwer Academic Publisher, 1998.
3. P. Hajek. Fuzzy logic. Stanford Encyclopedia of Philoshophy. http://plato.stanford.edu/entries/logic-fuzzy/#4. Last consulted: November 15, 2009, 2006.
4. G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications.* Prentice Hall PTR, 1995.
5. V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Trans. Comput. Logic*, 2(4):526–541, Oct. 2001.
6. J. W. Lloyd. *Foundations of Logic Programming.* Springer, Berlin, second edition, 1987.
7. E. Mendelson. *Introduction to Mathematical Logic.* Wadsworth, Belmont, CA, third edition, 1987.
8. P. Minari. A note on Łukasiewicz's three-valued logic. *Annali del Dipartimento di Filosofia dell'Università di Firenze*, pages 163–190, 2003.
9. J. Močkoř, V. Novák, and I. Perfilieva. *Mathematical principles of fuzzy logic.* Kluwer Academic Publisher, 2000.
10. M. Osorio and J. L. Carballido. Brief study of G'$_3$ logic. *Journal of Applied Non-Classical Logic*, 18(4):475–499, 2008.
11. 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.
12. 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.
13. M. Osorio, J. A. N. Pérez, and J. Arrazola. Equivalence in answer set programming. pages 57–75, 2001.
14. M. Osorio and C. Zepeda. Update sequences based on minimal generalized pstable models. In *MICAI*, pages 283–293, 2007.
15. M. Osorio and C. Zepeda. Pstable theories and preferences. In *Electronic Proceedings of the 18th International Conference on Electronics, Communications, and Computers (CONIELECOMP 2008)*, March, 2008.
16. S. Pascucci and A. Lopez. Implementing p-stable with simplification capabilities. *Submmited to Inteligencia Artificial, Revista Iberoamericana de I.A.*, Spain, 2008.
17. D. Pearce. Stable Inference as Intuitionistic Validity. *Logic Programming*, 38:79–91, 1999.
18. J. Rosser and A. Turquette. *Many-Valued Logics.* Amsterdam: North-Holland Publishing Company, 1952.
19. A. Urquhart. Many-valued logic. Chapter in book: Handbook of Philosophical Logic, Vol. III. In Gabbay and Guenthner (eds.), D.Reidel Publishing Company, l984.
20. L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.