Basic Transformations that Preserve G'3-stable Equivalente

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

[1] Benemérita Universidad Autóma de Puebla,
Mathematics Department, Puebla, México
carballido, arrazola@fcfm.uap.buap.mx
[2] Universidad de las Américas,
Sta. Catarina Mártir, Cholula, Puebla, México
osoriomauri@gmail.com

Abstract We study the notion of strong equivalence between two disjunctive logic programs under the G'_3-stable model semantics, also called the P-stable semantics, and we show how some particular cases of testing strong equivalence between programs can be reduced to verify if a formula is a theorem in some paraconsistent logics or in some cases in classical logic. We also present some program transformations for disjunctive programs which can be used to reduce the size of a program.

1 Introduction

In [11,12], a new semantics for non-monotonic reasoning was introduced in terms of weak completions of the three-valued G'_3-logic. It is known that this semantics, as well as semantics defined with the same construct by other paraconsistent logics, is equivalent to the P-stable model semantics for normal programs [12]. In [8] the result is extended to the class of disjunctive programs. The P-stable semantics of a normal or disjunctive program P is defined by means of a fixed point operator in terms of classical logic, after applying to P a transformation similar to the reduction used to define the stable semantics [4]. Similar results for disjunctive programs are presented in [13] for the more common stable semantics. The stable semantics turns out to be equivalent, for disjunctive programs, to semantics defined in terms of weak completions of any intermediate logic. In [8] the authors present a simple translation of a disjunctive program D into a normal program N, such that the P-stable semantics of N corresponds to the stable semantics of D over the common language.
The logic G'_3 was introduced in [1] only to prove that the formula $A \vee (A \rightarrow B)$ is not a theorem of the logic C_w. In fact, the set of theorems of G'_3 is a superset of the set of theorems of C_w. One interesting feature of the G'_3-logic is that it can be expressed in terms of the Lukaciewicz L_3-logic, and vice-versa, the Lukaciewicz L_3-logic can be expressed in terms of the G'_3-logic [9]. In the same survey it is proven that the logic G'_3 admits a finite axiomatization, in fact the axiomatization presented there, consists of the axioms for Cw plus four new

Advances in Artificial Intelligence and Applications
Research in Computer Science 32, 2007, pp. 89–98

Received 17/06/07
Accepted 31/08/07
Final version 21/09/07

axioms.
Two programs P_1 and P_2 are said to be strongly G'_3-stable equivalent, if for every program P, the programs $P_1 \cup P$ and $P_2 \cup P$ are G'_3-stable equivalent, i.e. they have the same G'_3-stable models. The notion of strongly equivalent logic programs is interesting since, given two sets of rules that are strongly equivalent, one of them can be replaced by the other one in any logic program without changing the declarative semantics of the program. This replacement can be a step toward program simplification. The notion of equivalence between logic programs has been studied by several authors in the context of the stable semantics, for example [3,6,10]. In the present work a different kind of equivalence is considered along with strong G'_3-stable equivalence for disjunctive programs, it is called uniform G'_3-stable equivalence and is weaker than strong G'_3-stable equivalence. G'_3 is one of several paraconsistent logics that can express the p-stable semantics for disjunctive programs, as defined in [8]. Therefore conditions under which two programs are strongly G'_3-equivalent also guarantee that two disjunctive programs are strongly equivalent in the p-stable semantics. We present two main results that guarantee G'_3 strong equivalence, one for two arbitrary programs and another one for a couple of programs of the form P , $P \cup \{a\}$, where P is a disjunctive program and a is an atom. A similar result for the stable semantics is presented in lemma 4.2 of [10] for disjunctive programs, but in this case the necessary condition for the strong equivalence of P and $P \cup \{a\}$ is more stringent. Both of our results depend on verifying that certain formula is a theorem in some particular logic. Finally, we present some program transformations that help reduce the size of a program. In most cases the transformed program is strongly G'_3-equivalent to the original program.
The structure of the paper is as follows: we start with basic background and definitions of the G'_3-logic, the P-stable semantics and the X-stable semantics for any logic X. Section 3 presents first the main results relative to strong equivalence of two disjunctive programs, and then it deals with some basic transformations of programs. Then we present our conclusions and ideas for future work.

2 Background

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

proposition symbols: $p_0, p_1, \ldots$
connectives: $\wedge$, $\vee$, $\leftarrow$, $\neg$ 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. If F is a formula we will refer to its signature $\mathcal{L}_F$ as the set of atoms that occur in F. The formula $F \equiv G$ is an abbreviation for $(F \leftarrow G) \wedge (G \leftarrow F)$. A *literal* is either an atom a, or the negation of an atom $\neg a$.

When a formula is constructed as a conjunction (or disjunction) of literals. $F = \bigwedge \ell$ (or $F = \bigvee \ell$) with ℓ a set of literals, we denote by $Lit(F)$ such set of

literals ℓ. A *clause* is a formula of the form $H \leftarrow B$ where H and B, arbitrary formulas in principle, are known as the *head* and *body* of the clause respectively. The body of a clause could be empty in which case the clause is known as a *fact* and can be noted just by H.

An *augmented* clause is a clause where H and B are any formulas constructed by using the $\vee, \wedge, \neg$ connectives. A *free* clause is a clause of the form $\bigvee(\mathcal{H}^+ \cup \neg\mathcal{H}^-) \leftarrow \bigwedge(\mathcal{B}^+ \cup \neg\mathcal{B}^-)$ where $\mathcal{H}^+$, $\mathcal{H}^-$, $\mathcal{B}^+$, $\mathcal{B}^-$ are, possibly empty, sets of atoms. Sometimes such a clause may be written as $\mathcal{H}^+ \vee \neg\mathcal{H}^- \leftarrow \mathcal{B}^+, \neg\mathcal{B}^-$ following typical conventions for logic programs. When $\mathcal{H}^- = \emptyset$, there is no negation in the head, the clause is called a *general* clause. If, moreover, $\mathcal{H}^+ \neq \emptyset$ (i.e. it not a constraint) the clause is called a *disjunctive* clause. When the set $\mathcal{H}^+$ contains exactly one element the clause is called *normal*.

Finally, a *program* is a finite set of clauses. If all the clauses in a program are of a certain type we say the program is also of this type. For instance a set of augmented clauses is an *augmented program*, a set of free clauses is a *free program* and so on.

For general programs, and proper subclasses, we will use $HEAD(P)$ to denote the set of all atoms occurring in the head of clauses in P.

Next, we proceed to give definitions of the relevant semantics and the three-valued logic G'_3

2.1 P-stable Semantics

We will use the following notation: $\models$ denotes the consequence relation in classical logic. We denote by $\vdash_{C_w}$ the inference relation in logic C_w which is the minimal paraconsistent logic defined by DaCosta [2]. We also assume that the reader is familiar with the notion of classical minimal model [7].

Here we define the P-stable semantics for disjunctive programs.

Definition 1. *[8] Let P be a disjunctive program and M be a set of atoms. We define:*

$$RED(P, M) = \{H \leftarrow B^+, \neg(B^- \cap M) \mid H \leftarrow B^+, \neg B^- \in P\}$$

Definition 2. *[8] Let P be a disjunctive program and M be a set of atoms. We say that M is a p-stable model of P if the conjunction of the atoms in M is a logical consequence in classical logic of $RED(P, M)$ (denoted as $RED(P, M) \models M$) and M is a classical model of P (i.e. a model in classical logic).*

2.2 G'_3 Logic

G'_3 logic is defined through a 3-valued logic with truth values in the domain $D = \{0, 1, 2\}$ 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.

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 1. Truth tables of connectives in G'_3.

2.3 The $\dot{X}$-stable Semantics

Given an arbitrary program P and a set of atoms $M \subset L_P$, we call the construct $P \cup \neg M^c$ a *weak completion* of the program P (with respect to the set of atoms M), where the superscript c, denotes set theoretical complement operator with respect to L_P.

Definition 3. *Let P be any theory and X be any logic. Also let M be a set of atoms. M is a X-stable model of P if $P \cup \neg M^c \vdash_X M$ and M is a classical model of P.*

The G'_3-stable semantics Of particular interest to us is the G'_3-stable semantics, which is the result of using the logic G'_3 in the previous definition.

Example 1. Consider the following logic program:

$$P = \{b \leftarrow \neg a, a \leftarrow \neg b, p \leftarrow \neg a, p \leftarrow \neg p\}$$

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

The next result was first proven for normal programs in [12]; more recently it has been extended to disjunctive programs in [8].

Theorem 1. *[8] Let P be a disjunctive program and M be a set of atoms. M is a p-stable model of P iff M is a G'_3-stable model of P.*

As can be seen, G'_3-stable models are defined for propositional logic programs only. However this definition can be extended to *predicate programs*, which allow the use of predicate symbols in the language, but without function symbols to ensure the ground instance of the program to be finite. So a *term* can only be either a variable or a constant symbol. The ground instance of a predicate program, $Ground(P)$, is defined in [5] as the program containing all ground instances of clauses in P. Then M is defined as a G'_3-stable model of a predicate program P if it is a G'_3-stable model for $Ground(P)$.

We want to stress the fact that the general approach for calculating $G3'$-stable models of logical programs is to work with their ground instances.

3 Main Results

In this section we discuss the main contributions of the paper, one of them provides sufficient conditions for the strong G'_3-stable equivalence of two arbitrary programs, the other one deals more specifically with a couple of programs of the form P and $P \cup \{a\}$.

Definition 4. *We denote by C the class of certain programs i.e. normal, disjunctive, positive, augmented, etc. We say that two programs P_1 and P_2 in C are G'_3-stable equivalent if they have the same G'_3-stable models. We say that the programs P_1 and P_2 in C are strongly G'_3-stable equivalent with respect to the class C, if for any other program $P \in C$, $P \cup P_1$ and $P \cup P_2$ are G'_3-stable equivalent. We say that the programs P_1 and P_2 are uniformly G'_3-stable equivalent if for any set of atoms M, $P_1 \cup M$ and $P_2 \cup M$ are G'_3-stable equivalent.*

Proposition 1. *Let P_1 and P_2 be programs in the same class C. If P_1 and P_2 are strongly G'_3-stable equivalent with respect to the class C then they are uniformly G'_3-stable equivalent. If P_1 and P_2 are uniformly G'_3-stable equivalent then they are G'_3-stable equivalent.*

Proof. The proof of this statement is a direct consequence of the definitions.

The next two examples show that two programs that are G'_3-stable equivalent are not necessarily uniformly G'_3-stable equivalent. Our first counterexample is with normal programs:

Example 2.

$$P_1 = \{a \leftarrow \neg b, \quad b \leftarrow b\}$$

$$P_2 = \{a, \ b \leftarrow b\}$$

Both programs have as unique G'_3-stable model the set: $\{a\}$.
If we take

$$P = \{b\},$$

then we see that $P \cup P_1$ has only one G'_3-stable model, namely: $\{b\}$, whereas $P \cup P_2$ has as its unique G'_3-stable model the set $\{a, b\}$.
As a second counterexample we take two programs in the class of disjunctive programs:

$$P_1 = \{a \vee c \leftarrow \neg b\}, P_2 = \{a \vee c\}.$$

These programs do not have G'_3-stable models, but if we add the fact b, to both of them, we get a program for which the set $\{b\}$ is a G'_3-stable model and a program that does not have G'_3-stable models.

Proposition 2. *Let P_1 and P_2 be arbitrary programs, if $P_1 \equiv_{G'_3} P_2$ (i.e. they are equivalent in the G'_3 logic), then they are strongly G'_3-stable equivalent with respect to the class of arbitrary programs.*

Proof. Let P be another arbitrary program, let M be a G'_3-stable model of $P_1 \cup P$, this means that M is a classical model of $P_1 \cup P$ and also that $P_1 \cup P \cup \neg M^c \vdash_{G'_3} M$. Since $P_1 \equiv_{G'_3} P_2$, it follows that $P_1 \cup P \equiv_{G'_3} P_2 \cup P$ and then we conclude that M is a classical model of $P_2 \cup P$ and $P_2 \cup P \cup \neg M^c \vdash_{G'_3} M$ as desired.

A symmetric argument shows that a G'_3-stable model of $P_2 \cup P$ is also a G'_3-stable model of $P_1 \cup P$.

The converse of the proposition in general is not true. We consider a counterexample in the class of disjunctive programs.

Example 3. Let P_1 and P_2 be the following disjunctive programs:

$$P_1 = \{\vee c_i \leftarrow \wedge a_j \wedge \neg b, \quad b \leftarrow\}$$

$$P_2 = \{\vee c_i \leftarrow c_1, \quad b \leftarrow\}$$

Let us see first that P_1 and P_2 are not equivalent in the G'_3-logic: in order to do this, it is enough to show that the two rules in P_2 are not strong enough in the G'_3-logic to imply the first rule in P_1. In other words, $[(c_1 \rightarrow \vee c_i) \wedge b] \rightarrow [(\wedge a_j \wedge \neg b) \rightarrow \vee c_i]$ is not a G'_3 tautology. This can be seen by working out a truth table.

Now we proceed to check that the two programs are strongly G'_3-stable equivalent. Let P be a disjunctive program, we want to see that $P_1 \cup P$ and $P_2 \cup P$ have the same p-stable models (by Theorem 1).

Let M be a p-stable model of $P_1 \cup P$, in particular it must be the case that $b \in M$. By definition, M is a classical model of $P_1 \cup P$ and also $RED(P_1 \cup P, M) \models M$, this last condition is equivalent to: $P_1 \cup RED(P, M) \models M$.

Now, from the fact that the expression: $b \rightarrow [(\neg b \wedge A) \rightarrow C]$ is a tautology in classical logic for any formulas A and C, it follows that $P_2 \models P_1$. It is also true that $P_1 \models P_2$ since the rule in P_2 that does not appear in P_1 is a tautology.

It follows that M is a classical model of $P_2 \cup P$ and that $RED(P_2 \cup P, M) \models M$ since $P_2 \cup RED(P, M) = RED(P_2 \cup P, M)$. The rest of the proof consist of a similar argument.

Observation 1: The same example works out as a counter-example for the class of normal programs if we remove the disjunctions in the heads of the disjunctive rules.

Observation 2: Notice that if we do not restrict our programs to the class of disjunctive programs, the counterexample does not work: consider the program $Q = \{d \leftarrow (\vee c_i \leftarrow \wedge a_j \wedge \neg b)\}$ consistent of a single clause. The two programs $P_1 \cup Q$ and $P_2 \cup Q$ do not have the same G'_3-stable models, so in this context P_1 and P_2 are not strongly G'_3-equivalent.

Lemma 1. *Let P be a disjunctive program and M be a classical model for P, then $P \cup M$ has G'_3-stable models.*

Proof. By hypothesis M is a classical model for P, then it is also a classical model for $P \cup M$. Since $RED(P \cup M, M) = RED(P, M) \cup M$, it is clear that $RED(P \cup M, M) \models M$. Hence M is a G'_3-stable model of $P \cup M$.

For the next result we need the following fact [8].

Lemma 2. *For a disjunctive program P, and an atom a we have that $P \models a$ if and only if $P \vdash_{C\omega} a$.*

Proposition 3. *Let P be a disjunctive program and a be an atom, then $P \models a$ if and only if P and $P \cup \{a\}$ are strongly G'_3-stable equivalent, if and only if P and $P \cup \{a\}$ are uniformly G'_3-stable equivalent.*

Proof. From $P \models a$. it follows that $P \vdash_{C\omega} a$ and from the fact that the G'_3-logic is stronger than the $C\omega$-logic, it follows that $P \vdash_{G'_3} a$. Hence we conclude that $P \equiv_{G'_3} P \cup \{a\}$, this in turn implies by proposition 2 that P and $P \cup \{a\}$ are strongly G'_3-stable equivalent and in particular, uniformly G'_3-stable equivalent. Conversely, let us assume now that the two programs are uniformly G'_3-stable equivalent. Assume for a moment that $P \not\models a$: then there exists a classical model of P, say M such that $M(a) = 0$. By Lemma 1, M is a G'_3-stable model of $P \cup M$, but M is not a G'_3-stable model of $P \cup \{a\} \cup M$. It means that P and $P \cup \{a\}$ are not uniformly G'_3-stable equivalent, but this is a contradiction and the proof is complete.

Definition 5. *Given a disjunctive program P, we define*

$$to - pos(P) = \{\bigvee_i a_i \vee \bigvee_j c_j \leftarrow \wedge_l b_l \mid \vee_i a_i \leftarrow \bigwedge_l b_l \wedge \bigwedge_j \neg c_j \in P\}.$$

The transformation just moves negative literals from the body of every clause to their heads by changing them to positive.

Corollary 1. *Let P be a disjunctive program and a be an atom. Then $to - pos(P) \vdash_{C\omega} a$ if and only if P and $P \cup \{a\}$ are strongly G'_3-stable equivalent, if and only if P and $P \cup \{a\}$ are uniformly G'_3-stable equivalent.*

Proof. Follows from the fact that $to - pos(P)$ and P are equivalent in classical logic, then by using Lemma 2 the result is simply a restatement of Proposition 3.

Definition 6. *Given a disjunctive program P. $Pos(P)$ is the program obtained from P after deleting all rules containing negative atoms.*

In [10] it is shown that for a disjunctive program P and an atom a, P is strongly equivalent to $P \cup \{a\}$ under the stable semantics if and only if $Pos(P) \vdash_I a$ (here the I denotes Intuitionism). In the case of G'_3-semantics the equivalence does not hold. From Proposition 3 it follows that if $Pos(P) \vdash_I a$ then P and $P \cup \{a\}$ are strongly G'_3-stable equivalent. The next examples show that the converse is not true.

Example 4. Let P_1 be: $\{a \leftarrow \neg a\}$ and let P_2 be: $P_1 \cup \{a\}$.
Since $(\neg a \rightarrow a) \rightarrow a$ is a G'_3-tautology, it follows that $P_1 \equiv_{G'_3} P_2$ which implies strong G'_3-stable equivalence, however $Pos(P_1) = \emptyset$ and from the empty set no atom can be derived.

Example 5. Let P_3 and P_4 the following disjunctive programs:

$$P_3 = \{a \leftarrow b, \quad a \leftarrow \neg b, \quad b \leftarrow a\}$$

$$P_4 = \{a, \quad a \leftarrow b, \quad a \leftarrow \neg b, \quad b \leftarrow a\}$$

The two programs are equivalent in the G'_3-logic, therefore they are strongly G'_3-stable equivalent, but the atom a is not implied by the rules of $Pos(P_3)$.

Finally we present some transformation defined for disjunctive programs that can help reduce the size of the program, most of them do not change the G'_3-stable semantics of the original program. Similar results in the context of stable semantics have been presented in [10].

Definition 7. *The transformation RED^+ replaces a rule $A \leftarrow B^+ \wedge \neg B^-$ by $A \leftarrow B^+ \wedge \neg(B^- \cap Head(P))$.*

Proposition 4. *For a disjunctive program P, P and $RED^+(P)$ are equivalent under the G'_3-stable semantics.*

Proof. The result follows from the observations:
1) Any G'_3-stable model M of P has the property that $M \subset Head(P)$
2) Any set of atoms $M \subset Head(P)$ models a clause of the form $A \leftarrow B^+ \wedge \neg B^-$ if and only if, it models the clause $A \leftarrow B^+ \wedge \neg(B^- \cap Head(P))$
3) The two clauses, $A \leftarrow B^+ \wedge \neg(B^- \cap M)$ and $A \leftarrow B^+ \wedge \neg(B^- \cap Head(P) \cap M)$ are the same whenever $M \subset Head(P)$

Definition 8. *The transformation RED^- deletes a rule $A \leftarrow B^+ \wedge \neg B^-$ if there is another rule $A' \leftarrow \top$ such that $A' \subset B^-$.*

Proposition 5. *For a disjunctive program P, P and $RED^-(P)$ are strongly G'_3-stable equivalent with respect to the class of disjunctive programs.*

Proof. This follows from the fact that, under the hypothesis $A' \subset B^-$, the formula $A' \rightarrow (B^+ \wedge \neg B^- \rightarrow A)$is a tautology in classical logic, for then we can derive all of the rules of P from $RED^-(P)$ and vice versa.

Definition 9. *The transformation Sub deletes a rule $A \leftarrow B^+ \wedge \neg B^-$ if there is another rule $A' \leftarrow B'^+ \wedge \neg B'^-$ such that $A' \subset A$, $B'^+ \subset B^+$ and $B'^- \subset B^-$.*

Proposition 6. *For a disjunctive program P, P and $Sub(P)$ are strongly G'_3-stable equivalent with respect to the class of disjunctive programs.*

Proof. This follows from the fact that $P \equiv_{G'_3} Sub(P)$, which in turn follows from the various implications: $B^+ \wedge \neg B^- \rightarrow B'^+ \wedge \neg B'^-$, $B'^+ \wedge \neg B'^- \rightarrow A'$ and $A' \rightarrow A$.

Definition 10. *The transformation Taut deletes a rule $A \leftarrow B^+ \wedge \neg B^-$ if $A \cap B^+ \neq \emptyset$*

Proposition 7. *For a disjunctive program P, P and $Taut(P)$ are strongly G'_3-stable equivalent with respect to the class of disjunctive programs.*

Proof. This follows from the fact that a rule $A \leftarrow B^+ \wedge \neg B^-$ for which $A \cap B^+ \neq \emptyset$ is a tautology in the G'_3-logic, and so $P \equiv_{G'_3} Taut(P)$.

Definition 11. *If P contains the clause $a \leftarrow$, and there is also a clause: $A \leftarrow B^+ \wedge \neg B^-$ such that $a \in B^+$, then the transformation Dsuc replaces it by the rule: $A \leftarrow (B^+ - \{a\}) \wedge \neg B^-$.*

It is not difficult to see that the transformation Dsuc preserves strong G'_3-stable equivalence.

Definition 12. *The transformation Failure deletes a rule: $A \leftarrow B^+ \wedge \neg B^-$, from the program P if $B^+ \cap (Head(P))^c \neq \emptyset$.*

The transformation Failure preserves G'_3-stable equivalence but it des not preserve strong G'_3-stable equivalence as the following example shows:

Example 6. Let P_1 and its transformed program P_2 according to Failure be:

$$P_1 = \{b \leftarrow, \quad a \leftarrow b, \quad d \leftarrow c\}$$

$$P_2 = \{b \leftarrow, \quad a \leftarrow b\}$$

P_1 and P_2 are G'_3-stable equivalent but they are not uniformly G'_3-stable equivalent as it can be seen by adding to P_1 and P_2 the rule: $c \leftarrow$.

4 Conclusion

The results presented in this paper generalize to the G'_3-stable semantics, and in particular, to the P-stable semantics some results dealing with equivalence and strong equivalence under the stable semantics presented in [10]. The results allow to substitute strongly equivalent programs or to add known facts to programs in order to simplify them and make easier the computation of G'_3-stable and in particular P-stable models. All the transformations presented here were studied in [10] in the context of the stable semantics, but there are some transformations presented there that we have not explored yet. Among them, the generalized principle of partial evaluation and the Dloop transformation. Those transformations offer future work to continue our research of the P-stable semantics.

References

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

2. N. da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, 15(4):497–510, 1974.
3. D. de Jongh and L. Hendriks. Characterization of strongly equivalent logic programs in intermediate logics. *Theory and Practice of Logic Programming*, 3(3):259–270, 2003.
4. 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.
5. V. Lifschitz. Foundations of logic programming. in principles of knowledge representation, pages 69-127. CSLI publications, 1996.
6. V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2:526–541, 2001.
7. J. W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, second edition, 1987.
8. M. Osorio, J. Carballido, and J. Arrazola. Logical weak completions of paraconsistent logics. *submitted to the Journal of Logic and Computation.*
9. M. Osorio and J. L. Carballido. Brief study of G'$_3$ logic. In *Unpublished paper*, 2007.
10. M. Osorio, J. A. Navarro, and J. Arrazola. Equivalence in Answer Set Programming. In A. Pettorossi, editor, *Logic Based Program Synthesis and Transformation. 11th International Workshop, LOPSTR 2001*, number 2372 in LNCS, pages 57–75, Paphos, Cyprus, Nov. 2001. Springer.
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. D. Pearce. Stable Inference as Intuitionistic Validity. *Logic Programming*, 38:79–91, 1999.