Semi-negative normal programs based on p-stable semantics

Claudia Zepeda[1] and Jose Luis Carballido[2]

[1] Benemérita Universidad Atónoma de Puebla
Facultad de Ciencias de la Computación,
Puebla, Puebla, México
czepedac@gmail.com,
[2] Universidad Politécnica de Puebla
Tercer Carril del Ejido"Serrano",
San Mateo Cuanalá s/n.,
Juan C. Bonilla, Puebla, México
jlcarballido7@gmail.com

Abstract We introduce three different formats for normal programs with constraints. These forms help to simplify the search of p-stable models of the original program. In this way we further the study of the p-stable semantics. In this paper we indicate that the p-stable semantics for one of the normal forms proposed agrees with the Comp semantics.
Keywords: p-stable semantics, Clark's completion.

1 Introduction

Some approaches used to formalize non-monotonic reasoning (NMR) are based on the semantics of logic programs. In this work we are interested in furthers the study of one of the semantics useful in this formalization called *p-stable.*

In [8] the authors generalize to disjunctive programs what was done in [10]. They introduce the *p-stable semantics* for normal programs by using a transformation similar to the one used by Gelfond and Lifschits [4]. It is important to mention that the p-stable semantics, which can be defined in terms of paraconsistent logics, shares several properties with the stable semantics [4], but is closer to classical logic. For example, let $P = \{a \leftarrow \neg a\}$. We can verify that P does not have stable models. However the set $\{a\}$ could be considered the intended model for P in classical logic. Moreover, it is the p-stable model of P.

The present paper introduces three different formats for normal programs with constraints: *Negative normal programs*, *Restricted negative normal programs* and *Semi-Negative normal programs.* These forms help to simplify the search of p-stable models of the original program. Besides, for programs in these formats the p-stable and the stable semantics coincide. This furthers the interest in developing the p-stable semantics as a tool to help to understand the NMR.

This paper is structured as follows. In section 2 we give some definitions useful to understand this paper. In section 3 we give the definition of the different formats for normal programs with constraints: negative normal programs and restricted negative normal programs. We also show the different results about the stable semantics and the p-stable semantics of programs in these formats. In this section we also introduce another format for normal programs with constraints called Semi-negative normal programs. We show how to obtain the p-stable models of a Semi-negative normal program with constraints by means of the completion of a particular restricted negative normal program with constraints. Finally in section 4 we present some conclusions.

2 Preliminaries

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

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*: $\wedge$, $\vee$, $\leftarrow$, $\neg$; 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*. The formula $F \equiv G$ is an abbreviation for $(F \leftarrow G) \wedge (G \leftarrow F)$. A *clause* is a formula of the form $H \leftarrow B$ (also written as $B \rightarrow H$), 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 denoted just by: $H \leftarrow$. In the case when the head of a clause is empty, the clause is called a *constraint* and is denoted by: $\leftarrow B$.

A *normal* clause is a clause of the form $H \leftarrow \mathcal{B}^+ \cup \neg\mathcal{B}^-$ where H consists of one atom or can be empty, $\mathcal{B}^+$ is a conjunction of atoms $b_1 \wedge b_2 \wedge \ldots \wedge b_n$, and $\neg\mathcal{B}^-$ is a conjunction of negated atoms $\neg b_{n+1} \wedge \neg b_{n+2} \wedge \ldots \wedge \neg b_m$. $\mathcal{B}^+$, and $\mathcal{B}^-$ could be empty sets of atoms. A finite set of normal clauses P is a *normal program*. A finite set of normal clauses P joined with a set of constraints C is a *normal program with constraints*, denoted as $\langle P, C\rangle$. Since we shall restrict our discussion to propositional programs, then we take for granted that programs with predicate symbols are only an abbreviation of the ground program.

Finally we give two definitions useful to understand the definitions of stable and p-stable semantics for normal programs.

Let P be a normal program and M be a set of atoms. We define: $RED(P, M) = \{H \leftarrow B^+, \neg(B^- \cap M) \mid H \leftarrow B^+, \neg B^- \in P\}$.

For any program P, the positive part of P, denoted by $POS(P)$ is the program consisting exclusively of those rules in P that do not have negated literals.

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

Now we give the definition of stable semantics for normal programs as given in [8].

Definition 1. *[8] Let P be a normal program and let $M \subseteq \mathcal{L}_P$. Let us put $P^M = POS(RED(P, M))$, then we say that M is a* stable model *of P if M is a minimal classical model of P^M.*

The stable semantics for normal programs with constraints is defined as follows [8].

Definition 2. *[8] Let $\langle P, C\rangle$ be a normal program with constraints and let $M \subseteq \mathcal{L}_P$. We say that M is a stable model of $\langle P, C\rangle$ if M is a stable model of P and M is a classical model of C.*

Next we define the p-stable semantics for normal programs.

Definition 3. *[8] Let P be a normal program and M be a set of atoms. We say that M is a* p-stable model of P *if*

1. *M is a classical model of P (i.e. a model in classical logic), and*
2. *the conjunction of the atoms in M is a logical consequence in classical logic of $RED(P, M)$ (denoted as $RED(P, M) \models M$).*

The p-stable semantics for normal programs with constraints is defined as follows [8].

Definition 4. *[8] Let $\langle P, C\rangle$ be a normal program with constraints and let $M \subseteq \mathcal{L}_P$. We say that M is a p-stable model of $\langle P, C\rangle$ if M is a p-stable model of P and M is a classical model of C.*

We are going to present some transformations for normal programs that modify a program into another, in general, shorter program; the idea is that some program transformations may help to reduce the size of a program and keep at least the same p-stable semantics. These transformations are simple, and the transformed program is p-stable equivalent to the original program.

Definition 5. *Two normal programs P_1 and P_2 are* p-stable equivalent*, if P_1 and P_2 have the same p-stable models. Two normal programs with constraints (P_1, C_1) and (P_2, C_2) are* p-stable equivalent*, if (P_1, C_1) and (P_2, C_2) have the same p-stable models.*

Definition 6. *[2] If P contains the clause $a \leftarrow$, and there is also a clause $r : A \leftarrow B^+ \wedge \neg B^-$ such that $a \in B^+$, then the transformation $Dsuc_{a,r}$ replaces it by the rule: $A \leftarrow (B^+ - \{a\}) \wedge \neg B^-$.*

Proposition 1. *[2] The transformation Dsuc preserves p-stable equivalence for normal programs.*

The transformation Dsuc can also be applied to normal programs with constraints.

Definition 7. *[2] Let $\langle P, C\rangle$ be a normal programs with constraints. If P contains the clause $a \leftarrow$, and there is also a constraint $c \in C : \leftarrow B^+ \wedge \neg B^-$ such that $a \in B^+$, then the transformation $Dsuc_{a,c}$ replaces it by the rule: $\leftarrow (B^+ - \{a\}) \wedge \neg B^-$.*

Proposition 2. *[2] The transformation Dsuc preserves p-stable equivalence for normal programs with constraints.*

Now we define the transformation Failure for normal programs.

Definition 8. *[2] The transformation Failure deletes a rule $r_b : A \leftarrow B^+ \wedge \neg B^-$ that contains the atom b, from the program P, whenever $b \in B^+ \cap (Head(P))^c$.*

Proposition 3. *[2] The transformation Failure preserves p-stable equivalence for normal programs.*

The transformation Failure can also be applied to normal programs with constraints.

Definition 9. *[2] Let $\langle P, C\rangle$ be a normal programs with constraints. The transformation Failure deletes a constrain $c_b : \leftarrow B^+ \wedge \neg B^-$ that contains the atom b, from C, whenever $b \in B^+ \cap (Head(P))^c$.*

Proposition 4. *[2] The transformation Failure preserves p-stable equivalence for normal programs with constraints.*

Here we present a result that establishes a one-to-one correspondence between the p-stable models of a tight normal program and the classical models of its Clark's completion. This classical models can be obtained by using classical model generator systems.

Definition 10. *[1] Given a normal program P consisting of rules of the form: $a \leftarrow a_1 \wedge \ldots \wedge a_n \wedge \neg b_1 \wedge \ldots \wedge \neg b_m$ its completion $Comp(P)$ is obtained as follows:*

For each symbol a, let $S(a)$ denote the set of all clauses with a in the head. Suppose $S(a)$ is the set: $\{a \leftarrow Body_1, \ldots, a \leftarrow Body_k\}$ Replace this set with the single formula, $a \leftrightarrow Body_1 \vee \ldots \vee Body_k$ If $S(a) = \emptyset$, then replace it by $\neg a$.

Definition 11. *[1] A normal program P is said to be* tight*, if there exists a function η from $\mathcal{L}_P$ to the set of natural numbers, such that for every $a \leftarrow a_1, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n$ in P, and for every $1 \leq i \leq m : \eta(a) > \eta(a_i)$.*

Proposition 5. *[3] For any normal program, if P is tight then M is an stable model of P if and only if M is a model (in classical logic) of $Comp(P)$.*

3 Semi-Negative normal programs

In this section we give the definition of two formats for normal programs with constraints: *Negative normal programs* and *Restricted negative normal programs.* One of the main results indicates that the stable semantics and the p-stable semantics of a restricted negative normal program coincide. We also show how the p-stable semantics of a negative normal program corresponds to the p-stable semantics of a particular restricted negative normal program.

We start by introducing the definitions of negative normal programs and restricted negative normal programs without constraints.

Definition 12. *Let P be a normal program. We say that P is a* negative normal *(NN) program if it satisfies the first of the two conditions listed below, and we say that P is a* restricted negative normal *(RNN) program if it satisfies both conditions*

1. *every rule has its body composed of negative literals only;*
2. *the head of any rule does not appear in the body of the same rule.*

We also introduce the definitions of negative normal programs and restricted negative normal programs with constraints.

Definition 13. *Let $\langle P, C\rangle$ be a normal program with constraints. We say that $\langle P, C\rangle$ is a* NN program with constraints *if P is a NN program. We say that $\langle P, C\rangle$ is a* RNN program with constraints *if P is a RNN program.*

The following lemma will be useful to show that the stable semantics and the p-stable semantics of a RNN program coincide. Given a RNN program P, this lemma gives a characterization for a set $M \subseteq \mathcal{L}_P$, to be a p-stable model of P.

Let P be a RNN program, for any $M \subseteq \mathcal{L}_P$, we define $S_M = \{a \mid a \leftarrow \neg b_1, \wedge \ldots \wedge \neg b_m \in P,\ \{b_1, \ldots, b_m\} \cap M = \emptyset\}$.

Lemma 1. *Let P be a RNN program. If M is p-stable model of P then $M = S_M$.*

Proof. $RED(P, M) = \{a \leftarrow \neg(B^- \cap M) \mid H \leftarrow \neg B^- \in P\}$. By part 1) of the definition of p-stable model (Definition 3), it follows that $S_M \subset M$. Let us assume that $M \setminus S_M \neq \emptyset$ and take $m \in M \setminus S_M$. Let us define an interpretation: $I : \mathcal{L}_P \rightarrow \{0, 1\}$ such that $I(a) = 1$ for each $a \in \mathcal{L}_P \setminus \{m\}$ and $I(m) = 0$.

Now, using the fact that the program P has the property that the head of each rule does not appear in its body, it follows that I is a classical model for $RED(P, M)$ but it is not a model for M (Since $I(m) = 0$) this contradicts 2) in the definition of a p-stable model (Definition 3) and the lemma is proved.

The following theorem indicates that the stable semantics and the p-stable semantics of a RNN program coincide.

Theorem 1. *Let P be a RNN program, then the stable semantics of P coincides with the p-stable semantics of P.*

Proof. If M is a p-stable model of P then $M = S_M$ according to the lemma 1, and from the definition of P^M (Definition of a stable model), $P^M = \{a \leftarrow \mid a \in S_M\}$ then we conclude that M is a minimal model of P^M and then a stable model of P.

Let us now assume that M is a stable model of P. It follows that M is a minimal model for $P^M = \{a \leftarrow \mid a \in S_M\}$. Therefore $M = S_M$. From the fact that $P^M \subset RED(P, M)$, it follows that $RED(P, M) \models M$.

We still need to show that M is a classical model of P. Let us consider a rule $a \leftarrow \neg b_1 \wedge \neg b_2 \wedge \ldots \wedge \neg b_s \in P$, such that $a \notin M$. Then there is at least an i for which $b_i \in M$ (otherwise $a \in S_M = M$), then the rule is modeled by M. Therefore M models all of the rules in P as we wanted to show.

The following corollary of theorem 1 indicates that the stable semantics and the p-stable semantics of a RNN program with constraints also coincide.

Corollary 1. *Let $\langle P, C\rangle$ be a RNN program with constraints, then the stable semantics of $\langle P, C\rangle$ coincides with the p-stable semantics of $\langle P, C\rangle$.*

Proof. Straightforward from theorem 1, definition of stable semantics for normal programs with constraints (definition 2), and definition of p-stable semantics for normal programs with constraints (definition 4).

Let us observe that, from the definition, in a NN program the head of any rule could appear in the body of the same rule.

We shall show how the p-stable semantics of a NN program P corresponds to the p-stable semantics of a particular program associated to P. This new program is a RNN program and it is the result of applying the following transformation to P: For any normal program P, the transformed program, denoted by $TRN(P)$ is the program that results from P after deleting from the body of each rule the atom that appears as the head of the same rule. The rules that do not present this condition remain the same.

Definition 14. *For a rule r of the form $a \leftarrow \neg a \wedge \neg b_1 \wedge \ldots \wedge \neg b_n \in P$, we define the transformation TRN as follows: $TRN(r) = a \leftarrow \neg b_1 \wedge \ldots \wedge \neg b_n$. For a rule r for which $head(r) \notin body(r)$, we define $TRN(r) = r$. For a normal program P we define the transformation TRN as follows: $TRN(P) = \{TRN(r) \mid r \in P\}$. For a normal program with constraints $\langle P, C\rangle$ we define the transformation TRN as follows: $TRN(\langle P, C\rangle) = (TRN(P), C)$.*

Here we show how the p-stable semantics of a NN program P corresponds to the p-stable semantics of the RNN program $TRN(P)$.

Theorem 2. *Let P be a NN program, then the p-stable semantics of P coincides with the p-stable semantics of the RNN program $TRN(P)$.*

Proof. Let us assume that M is a p-stable model of P and let $a \leftarrow \neg a, \neg b_1, \ldots \neg b_n$ be one of the rules in P with the property that the head appears in the body. Assume that $a \notin M$. Since M is a classical model for the rule, then at least for one i, $b_i \in M$, making the rule true according to M; but then it is clear that the corresponding rule in $TRN(P)$ is also modeled by M. It follows that a classical model for P is also a classical model for $TRN(P)$. Next, by hypothesis we have that $RED(P, M) \models M$. Now, it is easy to see that for each rule $r \in P$, the rule $RED(r, M)$ is a logical consequence (in classical logic) of the rule $RED(TRN(r), M)$. Therefore we conclude that $RED(TRN(P), M) \models M$.

Now, if M is a p-stable model of $TRN(P)$, according to lemma 1, M consists of those atoms for which there exists a rule $r : a \leftarrow \neg b_1 \wedge \ldots \wedge \neg b_n$ such that $b_i \notin M$ for all i.

Let us show first that M is a classical model of P. It is enough to examine the rules in $P \setminus TRN(P)$: $a \leftarrow \neg a \wedge \neg b_1 \wedge \ldots \wedge \neg b_n$.

In the case for which $a \in M$, there is nothing to prove. In the case for which $a \notin M$, according to the lemma 1 there must exist $b_i \in M$ for some i, and then the rule is modeled by M. So M models P.

Now, it only remains to prove that $RED(P, M) \models M$. But again, M consists of those atoms for which there is a rule, $a \leftarrow \neg b_1 \wedge \ldots \wedge \neg b_n$ and $b_i \notin M$ for all i; then $RED(P, M)$ contains the rules $a \leftarrow$, for all $a \in M$. The desired conclusion follows now.

The following corollary of theorem 2 indicates that the p-stable semantics of a NN program with constraints $\langle P, C\rangle$ corresponds to the p-stable semantics of the RNN program $TRN(\langle P, C\rangle)$.

Corollary 2. *Let $\langle P, C\rangle$ be a NN program with constraints, then the p-stable semantics of $\langle P, C\rangle$ coincides with the p-stable semantics of the RNN program $TRN(\langle P, C\rangle)$.*

Proof. Straightforward from theorem 2 and definition of p-stable semantics for normal programs with constraints (definition 4).

Now we introduce another format for normal programs called Semi-negative normal programs.

Definition 15. *Let P be a normal program. We say that P is a* semi-negative normal *(semi-NN) program if for any atom that appears as a positive literal in the body of a rule, the following condition holds: It does not appear in the head of any rule unless it appears as a fact.*

We also give the definition of Semi-negative normal programs with constraints. We say that a positive constraint is a constraint that consists of a conjunction of positive literals in its body.

Definition 16. *Let $\langle P, C\rangle$ be a normal program with constraints. We say that $\langle P, C\rangle$ is a* semi-NN program with constraints *if P is a semi-NN program and every constraint in C is a positive constraint.*

The following proposition indicates that each semi-NN program has a p-stable equivalent NN-program.

Proposition 6. *If P is a semi-NN program, then P is p-stable equivalent to an NN program. If $\langle P, C\rangle$ is a semi-NN program, then $\langle P, C\rangle$ is p-stable equivalent to a NN program with constraints.*

Proof. After several applications (if necessary) of the program transformations $Dsuc$ and $Failure$, P and $\langle P, C\rangle$ can be transformed into a NN program or a NN program with constraints respectively with the same p-stable semantics.

Since NN programs, RNN programs and semi-NN programs without constraints are tight (see definition 11), it is possible to show that the p-stable models of a RNN program P correspond to the classical models of its completion $Comp(P)$.

Proposition 7. *Let P be a RNN program. M is a p-stable model of P if and only if M is a model (in classical logic) of $Comp(P)$.*

Proof. Let M be a pstable model of P. By theorem 1, M is a p-stable model of P if and only M is a stable model of P. Since P is tight; by proposition 5, M is an stable model of P if and only if M is a model (in classical logic) of $Comp(P)$.

Moreover, we also can show that the p-stable models of a RNN program with constraints $\langle P, C\rangle$ correspond to the classical models of its completion $Comp(\langle P, C\rangle)$. First we need to introduce two definitions: tight normal programs with constraints, and completion for a normal program with constraints.

We say that a normal program with constraints $\langle P, C\rangle$ is tight if P is tight. Therefore NN programs, RNN programs and semi-NN programs with constraints are tight.

The definition of completion for a normal program with constraints is based on the definition of completion for normal programs without constraints (see definition 10).

Definition 17. *For a positive constraint $c : \leftarrow a_1 \wedge a_2 \wedge \ldots a_n$ we define $Comp(c) = \neg a_1 \vee \neg a_2 \vee \ldots \vee \neg a_n$. If C is a set of constraints then $Comp(C) = \{comp(c) | c \in C\}$. For any normal program with constraints tight $\langle P, C\rangle$, we define $Comp(\langle P, C\rangle) = Comp(P) \cup Comp(C)$.*

Now we are ready to show that the p-stable models of a RNN program with constraints $\langle P, C\rangle$ correspond to the classical models of its completion $Comp(\langle P, C\rangle)$.

Proposition 8. *Let $\langle P, C\rangle$ be a RNN program with constraints. M is a p-stable model of $\langle P, C\rangle$ if and only if M is a model (in classical logic) of $Comp(\langle P, C\rangle)$.*

Proof. In particular $Comp(\langle P, C\rangle) = Comp(P) \cup Comp(C)$.

$\Rightarrow$) If M is a p-stable model of $\langle P, C\rangle$ then, by corollary 1, M is a stable model of $\langle P, C\rangle$. If M is a stable model of $\langle P, C\rangle$ then, by definition 2, M is a stable model of P and M models C in classical logic. By proposition 5 M is a classical model of $Comp(P)$ and also a classical model of $Comp(C)$ since C and $Comp(C)$ are equivalent in classical logic. Therefore M is a classical model of $Comp(\langle P, C\rangle)$.

$\Leftarrow$) Conversely if M is a classical model of $Comp(P) \cup Comp(C)$, then by proposition 5, M is a stable model of P. Also M is a classical model of R, since R and $Comp(R)$ are equivalent in classical logic. Therefore M is a stable model of $\langle P, C\rangle$. Again by corollary 1 M is a p-stable model of $\langle P, C\rangle$.

As a direct consequence of proposition 8 we can see that it is possible to obtain the p-stable models of a semi-NN program with constraints by means of the completion of a particular RNN program with constraints. By proposition 6 we showed that if $\langle P, C\rangle$ is a semi-NN program with constraints, then $\langle P, C\rangle$ is p-stable equivalent to an NN program with constraints $\langle P, C\rangle_{NN}$. By corollary 2 we showed that the p-stable semantics of $\langle P, C\rangle_{NN}$ corresponds to the p-stable semantics of the RNN program with constraints $TRN(\langle P, C\rangle_{NN})$; and by corollary 1 we showed that the p-stable semantics and the stable semantics of $TRN(\langle P, C\rangle_{NN})$ coincide. Finally by proposition 8 we showed that the p-stable models of $TRN(\langle P, C\rangle_{NN})$ correspond to the classical models of its completion, $Comp(TRN(\langle P, C\rangle_{NN}))$.

4 Conclusion

This paper furthers the study of p-stable semantics. We introduced three different formats for normal programs with constraints. We show that the stable semantics and the p-stable semantics of a Restricted Negative Normal program coincide. We also show that the p-stable semantics for Semi-Negative Normal programs with constraints agrees with the completion of a particular Restricted Negative Normal program with constraints.

References

1. C. Baral. *Knowledge Representation, reasoning and declarative problem solving with Answer Sets.* Cambridge University Press, Cambridge, 2003.
2. J. Carballido, J. Arrazola, and M. Osorio. Equivalence for the g3'-stable models semantics. In *Latin-American Workshop on Non-Monotonic Reasoning 2007 (LANMR07). Online= http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-286/*, volume 286, 2007.
3. F. Fages. Consistency of clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science*, 1:51–60, 1994.
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. G. Grossi, M. Marchi, E. Pontelli, and A. Provetti. Experiments with answer set computation over parallel and distributed architectures. In *Proceedings of 4th International Workshop on Answer Set Programming (ASP2007).*, September 2007. To appear an extended version in a special issue of the Journal of Logic and Computation.
6. J. W. Lloyd. *Foundations of Logic Programming.* Springer, Berlin, second edition, 1987.
7. R. C. Moore. Autoepistemic logic. In P. Smets, E. H. Mamdani, D. Dubois, and H. Prade, editors, *Non-Standard Logics for Automated Reasoning.* Academic Press, 1988.
8. M. Osorio, J. Arrazola, and J. L. Carballido. Logical weak completions of paraconsistent logics. *To apper in the Journal of Logic and Computation*, 2008.
9. M. Osorio, V. Borja, and J. Arrazola. Three valued logic of Łukasiewicz for modeling semantics of logic programs. In *Proceedings of IBERAMIA*, number 3315 in Lecture Notes in Computer Science, pages 343–352. Springer, 2004.
10. 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.
11. S. Pascucci and A. Lopez. Implementing p-stable with simplification capabilities. *Submmited to Inteligencia Artificial, Revista Iberoamericana de I.A.*, Spain, 2008.
12. D. Pearce. Stable Inference as Intuitionistic Validity. *Logic Programming*, 38:79–91, 1999.