Updates based on p-stable and properties

José Luis Carballido, Claudia Zepeda, Sergio Arzola and Mario Rossainz

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
{jlcarballido7,czepedac, sinrotulos, mrossainzl}@gmail.com

Abstract We present an update operator and several of its properties. The semantics of our update operator is based on a concept called minimal generalized p-stable models which is based on a novel semantics called the p-stable semantics.
Keywords: Logic Programming, Update semantics

1 Introduction

Update by definition means that there is new information that must be added to the older, and some information could be changed. Intelligent agents use this, in order to bring new knowledge to their knowledge base. But there is a main problem that updates can present, and it is inconsistency. So, it is important to use an approach to avoid inconsistencies in the knowledge base. For instance, it could be that in an initial moment we can infer a from a knowledge base (KB), and later the KB is updated with the new information $-a$ (where $-$ denotes negation). It is easy to see, that if we only take the union of the initial KB and $-a$ we will have an inconsistency. Then it is useful to apply an update approach that avoids the inconsistency and now allows to infer $-a$ since the newer knowledge has priority over the older.

Currently there are several approaches in non-monotonic reasoning dealing with updates, such as [6,11,4]. It is important to point out that several of the update approaches in non-monotonic reasoning are based on the stable semantics (also called answer set semantics)[1] , so their results do not agree with a classical logic point of view. We can make this clear with the following example. Let $P_1 = \{a \leftarrow \neg b, a \leftarrow b\}$ and $P_2 = \{b \leftarrow a\}$. From a classical logic point of view and considering that $\neg$ denotes classical negation, we would expect that $\{a, b\}$ correspond to the result of updating P_1 with P_2. However, when we apply the approach in [6] based on stable semantics to update P_1 with P_2 there is no result.

Moreover, since classical logic identifies a class of formal logics that have been most intensively studied and most widely used, it turns out to be very useful to have some updates approaches that allow us to keep a compromise with such logic. For this reason, as part of the contribution of this paper, we propose an update semantics that allow us to keep a compromise with classical logic. Thus the result of using our semantics to update program P_1 with program P_2 is $\{a, b\}$.

[1] Readers interested in knowing the definition of stable semantics can see [7].

Our semantics is based on a concept called minimal generalized p-stable models. The definition of minimal generalized p-stable models is inspired by a concept called minimal generalized answer sets of abductive programs [8]. The semantics of minimal generalized answer sets is based on the stable semantics. The minimal generalized answer sets have been used to restore consistency [8,2], to obtain the preferred plans of planning problems [14], to get the preferred extensions of an argument framework [14], and to define update operators [13]. Hence, we consider that minimal generalized p-stable models can also have similar applications and be an alternative to those applications that use minimal generalized answer sets, however the minimal generalized p-stable models consider a classical logic point of view.

The semantics of minimal generalized p-stable models is based on the p-stable semantics. The p-stable semantics is based on G'_3 which is a paraconsistent logic that has recently been studied in some detail in [9,10,12]. Here, we also present several properties for our update operator. These properties correspond to the properties of the update operator defined and analized by Eiter et al. [5] and J. J. Alferes et al. in [1], except for one of them called *independent parts property*. This last property refers to the general principle that asserts that completely independent parts of a program should not interfere with each other.

In section 2 we summarize some basic concepts and definitions used to understand this paper. In section 3 we review the minimal generalized p-stable models. In section 4 we present our update semantics and some formal properties. Finally, in section 5 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

We use the language of propositional logic in the usual way. We consider *propositional symbols*: $p, q, \ldots$; *propositional connectives*: $\wedge, \vee, \rightarrow, \neg, -$; and *auxiliary symbols*: '(',')','.'. Well formed propositional formulas are defined as usual. We consider two types of negation: strong or classical negation (written as $-$) and negation-as-failure (written as $\neg$). Intuitively, $\neg a$ is true whenever there is no reason to believe a, whereas $-a$ requires a proof of the negated atom. An *atom* is a propositional symbol. A *literal* is either an atom a or the strong negation of an atom $-a$.

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 $1 \leq i \leq n+m$. In a slight abuse of notation we will denote such a clause by the formula $a \leftarrow \mathcal{B}^+ \cup \neg\mathcal{B}^-$ where the set $\{b_1, \ldots, b_n\}$ will be denoted by $\mathcal{B}^+$, and the set $\{b_{n+1}, \ldots, b_{n+m}\}$ will be denoted by $\mathcal{B}^-$. Given a normal clause $a \leftarrow \mathcal{B}^+ \cup \neg\mathcal{B}^-$ we say that a is the *head* and $\mathcal{B}^+ \cup \neg\mathcal{B}^-$ is the *body* of the clause. A clause with an empty body is called a *fact*;

and a clause with an empty head is called a *constraint*. Facts and constraints are also denoted as $f \leftarrow$ and $\leftarrow g$ respectively. We define a *normal logic program* P, as a finite set of normal clauses. The signature of a normal logic program P, denoted as $\mathcal{L}_P$, is the set of atoms that occur in P. Given a set of atoms M and a signature $\mathcal{L}$, we define $\neg\widetilde{M} = \{\neg a \mid a \in \mathcal{L} \setminus M\}$. 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. From now on, by *program* we will mean a normal logic program when ambiguity does not arise.

In our programs we will manage the strong negation $-$ as follows: each atom $-a$ is replaced by a new atom symbol a' which does not appear in the language of the program and we add the constraint $\leftarrow a \wedge a'$ to the program.

Finally, we give a definition that will help us to define the p-stable semantics for programs.

Definition 1. *[12] Let P be a program and M be a set of atoms. We define* $RED(P, M) = \{a \leftarrow \mathcal{B}^+ \cup \neg(\mathcal{B}^- \cap M) \mid a \leftarrow \mathcal{B}^+ \cup \neg\mathcal{B}^- \in P\}$.

Example 1. [12] Let $P = \{b \leftarrow \neg a, \quad a \leftarrow \neg b, \quad p \leftarrow \neg a, \quad p \leftarrow \neg p, \quad c \leftarrow p\}$. Given $M = \{a, p\}$, it follows that $RED(P, M) = \{b \leftarrow \neg a, \ \ a \leftarrow, \ \ p \leftarrow \neg a, \ \ p \leftarrow \neg p, \ \ c \leftarrow p\}$.

2.2 The p-stable semantics

Here, we present a fixed point characterization, using classical logic, of the p-stable semantics for normal programs. This kind of characterization is useful for implementations of a semantics.[2] Following a similar approach to [7] for the stable semantics, the p-stable semantics uses the $RED(P, M)$ reduction as a fixed point operator in terms of classical logic.

Definition 2. *[9] Let P be a 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 following examples illustrate how to obtain the p-stable models. The first example shows a program with a single p-stable model, which is also a classical model. The second example shows a program which has no stable models and whose p-stable and classical models are the same.

Example 2. Let $P = \{q \leftarrow \neg q\}$. Let us take $M = \{q\}$ then $RED(P, M) = \{q \leftarrow \neg q\}$. It is clear that M models P in classical logic and $RED(P, M) \models M$ since $(\neg q \rightarrow q) \rightarrow q$ is a theorem in classical logic with the negation $\neg$, now interpreted as classical negation. Therefore M is a p-stable model of P.

[2] An implementation of the p-stable semantic is at `http://sites.google.com/site/computingpstablesemantics/`

Example 3. Let $P = \{a \leftarrow \neg b,\ \ a \leftarrow b,\ \ b \leftarrow a\}$. We can verify that $M = \{a, b\}$ models the clauses of P in classical logic. We find that $RED(P, M) = P$. Now, from the first and third clause, it follows that $(\neg b \rightarrow b)$ where the negation $\neg$ is now interpreted as classical negation. Since $(\neg b \rightarrow b) \rightarrow b$ is a theorem in classical logic, it follows that $RED(P, M) \models M$. Therefore, M is a p-stable model of P.

Not all programs have p-stable models, that is why it is convenient to have the next definition.

Definition 3. *Let P be a program. We say that P is* p-stable consistent *if P has at least one p-stable model. We say that P is p-stable inconsistent if P does not have p-stable models.*

We also remark that the authors of [9] show that the p-stable model semantics for normal logic programs is powerful enough to express any problem that can be expressed with the stable model semantics for disjunctive logic programs. It is worth mentioning that there exists also a characterization of the p-stable semantics in terms of the paraconsistent logic G'_3, interested readers can see [12].

Now, we present two notions of equivalence for programs.

Definition 4. *[3] Two programs P_1 and P_2 are* equivalent*, denoted by $P_1 \equiv P_2$, if P_1 and P_2 have the same p-stable models. Two programs P_1 and P_2 are* strongly equivalent*, denoted by $P_1 \equiv_{SE} P_2$, if $(P_1 \cup P) \equiv (P_2 \cup P)$ for every program P.*

The following lemma[3] indicates that given a program P and an atom x that does not occur in P, we can define a new program P' such that P and P' are equivalent and $\mathcal{L}_{P'} = \mathcal{L}_P \cup \{x\}$. The two programs must have the same clauses except for one of them. One of the clauses in P' corresponds to one of the clauses in P after adding $\neg x$ to its body. This way, P and P' have the same p-stable models since x does not appear as the head of any clause in P'.

Lemma 1. *Let P be a program and x be an atom, $x \notin \mathcal{L}_P$. Let r be any clause $a \leftarrow \mathcal{B}^+ \cup \neg\mathcal{B}^-$ in P. Then M is a p-stable model of P iff M is a p-stable model of $(P \setminus \{r\}) \cup \{a \leftarrow \mathcal{B}^+ \cup \neg(\mathcal{B}^- \cup \{x\})\}$.*

3 Minimal generalized p-stable models

The definition of our update semantics is based on Minimal Generalized (MG) p-stable models. The intuition behind the MG p-stable models is simple. Given a logic program P and a set of atoms A, the MG p-stable models of P are the p-stable models of $P \cup \Delta$ that are obtained by adding the minimal subset $\Delta \subseteq A$ to P for which $P \cup \Delta$ has p-stable models.[4] For instance, let us consider a program $P = \{-a,\ a \leftarrow \neg b\}$ that does not have p-stable models and let $A = \{b, c\}$ then,

[3] Its proof is straightforward.

[4] By "adding the minimal subset $\Delta \subseteq A$ to P", we mean that Δ is interpreted as a set of facts defined by its elements.

the MG p-stable model is $\{b, -a\}$ where the minimal subset of A added to P is $\{b\}$.

Next, we present the definition of abductive logic programs and their semantics in terms of the minimal explicit generalized p-stable models. Then, we define the MG p-stable models based on the minimal explicit generalized p-stable models. These definitions are similar to the definitions of syntax and semantics of abductive logic programs as presented in the context of the stable semantics in [2].

Definition 5. *An abductive logic program is a pair $\langle P, A\rangle$ where P is a program and A is a set of atoms, called abducibles. $\langle M, \Delta\rangle$ is an explicit generalized (EG) p-stable model of the abductive logic program $\langle P, A\rangle$ iff $\Delta \subseteq A$ and M is a p-stable model of $P \cup \Delta$.*

We give an ordering among EG p-stable models in order to get the minimal of them.

Definition 6. *Let $T = \langle P, A\rangle$ be an abductive logic program. Let $\langle M_1, \Delta_1\rangle$ and $\langle M_2, \Delta_2\rangle$ be two EG p-stable models of T, we define $\langle M_1, \Delta_1\rangle < \langle M_2, \Delta_2\rangle$ if $\Delta_1 \subset \Delta_2$; this order is called inclusion order. $\langle M, \Delta\rangle$ is a Minimal Explicit Generalized (MEG) p-stable model of T iff $\langle M, \Delta\rangle$ is an EG p-stable model of T and it is minimal w.r.t. inclusion order.*

For practical purposes, given a MEG p-stable model, $\langle M, \Delta\rangle$, we are only interested in its first entry, namely M, and we call it a Minimal Generalized (MG) p-stable model of an abductive logic program.

Example 4. Let $\langle P, A\rangle$ be the abductive logic program where the set of abductive atoms is $A = \{x_1, x_2\}$ and $P = \{b \leftarrow \neg x_1,\ \ a \leftarrow b \wedge \neg x_2,\ \ -a\}$. There are three EG p-stable models of $\langle P, A\rangle$ which are: $\langle\{-a, x_1\}, \{x_1\}\rangle$, $\langle\{-a, b, x_2\}, \{x_2\}\rangle$, and $\langle\{-a, x_1, x_2\}, \{x_1, x_2\}\rangle$. We can see that for $\Delta = \emptyset$ there is no EG p-stable models. Therefore, the MEG p-stable models are $\langle\{-a, x_1\}, \{x_1\}\rangle$ and $\langle\{-a, b, x_2\}, \{x_2\}\rangle$, and the MG p-stable models are $\{-a, x_1\}$ and $\{-a, b, x_2\}$.

The following lemma presents some results about MEG p-stable models that will be useful in a later section to show the properties of our update operator. The proof of this lemma is straightforward.

Lemma 2. *Let $T = \langle P, A\rangle$ be an abductive logic program such that P is p-stable consistent. Then, M is a p-stable model of P iff M is a MG p-stable model of T; and if $\langle M, \Delta\rangle$ is a MEG p-stable model of T then $\Delta = \emptyset$.*

4 Updates semantics and formal properties

In this section, we define the semantics of our update operator based on the concept of MG p-stable models, and we study some of its properties. We use $\odot$ to represent the update operator. In order to obtain the $\odot$-update p-stable models of

a pair of logic programs $\mathsf{P} = (P_1, P_2)$, we define an update logic program, denoted as $\mathsf{P}_\odot$. The update logic program is obtained by joining P_1' to P_2, where P_1' is the resulting program from transforming P_1 as follows: at the end of each clause of P_1 which is not a constraint we add the negation-as-failure of an abducible (a new atom). The intuition behind the transformation applied to a program P_1 consists in weakening the knowledge in P_1 when giving more relevance to the knowledge contained in P_2 whose rules are not modified.

Definition 7. *Let $\mathsf{P} = (P_1, P_2)$ be a pair of logic programs over $\mathcal{L}_\mathsf{P}$ such that the number of clauses in P_1 that are not constraints is n. Let $\mathcal{L}_\mathsf{P}^* = \mathcal{L}_\mathsf{P} \cup A$ where A is a set of n new abducible atoms, namely $A = \{a_i, 1 \leq i \leq n \mid a_i$ is an atom, $a_i \notin \mathcal{L}_\mathsf{P}$ and $a_i \neq a_j$ if $i \neq j\}$. We define the* update logic program *$\mathsf{P}_\odot = P_1 \odot P_2$ over $\mathcal{L}_\mathsf{P}^*$, as the program consisting of the following clauses:*

1. *all constraints in P_1,*
2. *we add the clause $a \leftarrow \mathcal{B}^+ \cup \neg(\mathcal{B}^- \cup \{a_i\})$ if $r_i = a \leftarrow \mathcal{B}^+ \cup \neg\mathcal{B}^- \in P$, $1 \leq i \leq n$ and $a_i \in A$,*
3. *all clauses $r \in P_2$.*

We define the abductive logic program of P as follows: $T_\odot = \langle \mathsf{P}_\odot, A \rangle$.

In this way, the intended $\odot$-update p-stable models of a pair of logic programs $\mathsf{P} = (P_1, P_2)$ are obtained by removing the abducible atoms from the MG p-stable models of $T_\odot$. Finally, the $\odot$-update p-stable models are chosen as those that contain more information, i.e. maximal in the sense of inclusion of sets, from the intended $\odot$-update p-stable models.

Definition 8. *Let $\mathsf{P} = (P_1, P_2)$ be an update pair over $\mathcal{L}_\mathsf{P}$ and $T_\odot$ its abductive logic program. Then, $M \subseteq \mathcal{L}_\mathsf{P}$ is an* intended $\odot$-update p-stable model *of P if and only if $M = M' \cap \mathcal{L}_\mathsf{P}$ for some MG p-stable model M' of $T_\odot$. In case M is an intended $\odot$-update p-stable model of P and is maximal among all intended $\odot$-update p-stable model of P w.r.t. inclusion order, then M is an* $\odot$-update p-stable model *of P.*

We can illustrate our semantics with the following example.

Example 5. Let $\mathsf{P} = (P_1, P_2)$ be an update pair over $\mathcal{L}_\mathsf{P} = \{a, b\}$ where, P_1 and P_2 are the following logic programs, $P_1 = \{b \leftarrow, \quad a \leftarrow b\}$ and $P_2 = \{-a \leftarrow\}$. We can see that the *update logic program* $\mathsf{P}_\odot = P_1 \odot P_2$ over $\mathcal{L}_\mathsf{P}^*$ corresponds to the program P of Example 4 where the x_i are the abducible a_i. The intended $\odot$-update p-stable models of P are $\{-a\}$ and $\{-a, b\}$; and its only $\odot$-update p-stable model is $\{-a, b\}$.

Now, we show that our update operator ($\odot$) satisfies several formal properties. These properties have been deeply analyzed, in the context of stable semantics, by several authors such as J. J. Alferes et al. in [1] or T. Eiter in [6], except for the last property. We will see that all the properties are expressed in terms of equivalence, hence it is useful to recall the two notions of equivalence for logic

programs given in Definition 4. Since the p-stable models of a logic program are sets of literals, we can see easily that $\equiv$ represents an equivalence relations, and the logic programs P_1 and P_2 can be of any kind defined in this paper.

The following two definitions are used to define the last of our properties. Given $\mathbf{A} = \{A_1 \dots A_n\}$ where the A_i, $1 \leq i \leq n$ are sets, and $\mathbf{B} = \{B_1 \dots B_m\}$ where the B_j, $1 \leq j \leq m$ are sets, we define $\mathbf{A} \uplus \mathbf{B} = \{A_i \cup B_j \mid A_i \in \mathbf{A} \text{ and } B_j \in \mathbf{B}\}$.

Definition 9. *Let $\mathsf{P} = (P_1, P_2)$ be a pair of logic programs over $\mathcal{L}_\mathsf{P}$. We define the update semantic function of P as follows:*

$$SEM_\odot(\mathsf{P})^5 = \{M \mid M \text{ is an } \odot\text{-update p-stable model of } \mathsf{P}\}.$$

Now we define the properties. Since the intuition behind the first six properties is easy, hence we only give a deeper explanation about the last property below.

P1. *Initialisation:* If P is a logic program then $\emptyset \odot P \equiv P$.
P2. *Strong consistency:* Let P_1 and P_2 be logic programs. Suppose $P_1 \cup P_2$ has at least one p-stable model. Then $P_1 \odot P_2 \equiv P_1 \cup P_2$.
P3. *Idempotence:* If P is a logic program then $P \odot P \equiv P$.
P4. *Weak noninterference:* If P_1 and P_2 are logic programs defined over disjoint alphabets, and both of them have p-stable models or do not, then $P_1 \odot P_2 \equiv P_2 \odot P_1$.
P5. *Weak irrelevance of syntax:* Let P, P_1 and P_2 be logic programs under $\mathcal{L}_\mathsf{P}$. If $P_1 \equiv_{SE} P_2$ then $P \odot P_1 \equiv P \odot P_2$.
P6. *Augmented update:* Let P_1 and P_2 be logic programs such that $P_1 \subseteq P_2$. Then $P_1 \odot P_2 \equiv P_2$.
P7. *Independent parts property.* Let $\mathsf{P}_1 = (P_1, P_1')$ and $\mathsf{P}_2 = (P_2, P_2')$ such that $(\mathcal{L}_{\mathsf{P}_1} \cap \mathcal{L}_{\mathsf{P}_2}) = \emptyset$. Then $SEM_\odot((P_1 \cup P_2), (P_1' \cup P_2')) = SEM_\odot(\mathsf{P}_1) \uplus SEM_\odot(\mathsf{P}_2)$.

Property **P7** indicates that our update operator does not violates the general principle that completely independent parts of a logic program should not interfere with each other. Hence the property **P7** of operator $\odot$ indicates that if we update the union of a pair of logic programs $(P_1 \cup P_2)$ by the union of a different pair of logic programs $(P_1' \cup P_2')$ such that P_1 and P_1' are defined under a different language from the language of logic programs P_2 and P_2' then, the result can be also obtained from a particular union of the update of P_1 by P_1' and the update of P_2 by P_2'. This particular union of updates corresponds to our Definition 9.

Example 6. [11]

Let P_1 be:
$openSchool \leftarrow .$
$holiday \leftarrow \neg workday.$

Let P_1' be:
$-openSchool \leftarrow holiday.$
$workday \leftarrow \neg holiday.$

Let P_2 be:
$seeStars \leftarrow .$

Let P_2' be:
$-seeStars \leftarrow .$

[5] Let us notice that $SEM_\odot(\mathsf{P})$ is a set of sets.

Let $\mathsf{P}_1 = (P_1, P_1')$, $\mathsf{P}_2 = (P_2, P_2')$, and $\mathsf{P} = ((P_1 \cup P_2), (P_1' \cup P_2'))$. We can see that $(\mathcal{L}_{\mathsf{P}_1} \cap \mathcal{L}_{\mathsf{P}_2}) = \emptyset$. According to *independent parts property* we have that, $SEM_{\odot}((P_1 \cup P_2), (P_1' \cup P_2')) = SEM_{\odot}(\mathsf{P}_1) \uplus SEM_{\odot}(\mathsf{P}_2)$ since $SEM_{\odot}((P_1 \cup P_2), (P_1' \cup P_2')) = \{\{openSchool, workday, -seeStars\}\} = \{\{openSchool, workday\}\} \uplus \{\{-seeStars\}\} = SEM_{\odot}(\mathsf{P}_1) \uplus SEM_{\odot}(\mathsf{P}_2)$.

Theorem 1. *The update operator ($\odot$) satisfies properties,* **P1**, **P2**, **P3**, **P4**, **P5**, **P6**, *and* **P7**.

Proof. First, it is straightforward to verify that given a p-stable consistent program P, if M is p-stable model of P then there is not another p-stable model M' of P such that $M' \subset M$. So, by this last fact and by Lemma 2, it is also straightforward to verify that given an abductive logic program $\langle P, A\rangle$, where P is p-stable consistent, then if M is a MG p-stable model of $\langle P, A\rangle$ then there is not another MG p-stable model M' of $\langle P, A\rangle$ such that $M' \subset M$.
(**P1. Initialisation**): $\emptyset \odot P = P$ by construction. Hence $\emptyset \odot P \equiv P$.
(**P2. Strong consistency**): Let $Q = (P_1 \cup P_2)$ such that Q is p-stable consistent. Let P be the update (P_1, P_2). We must prove that M is an $\odot$-update p-stable model of P iff M is a p-stable model of Q.
Let us notice that programs Q and $\mathsf{P}_{\odot}$ have the same clauses except for some of them, namely in $\mathsf{P}_{\odot}$ there are some clauses that have an abducible atom (a new atom) in their body and these atoms do not occur in Q. So when we apply iteratively Lemma 1, two things are certain:

(1) S is a p-stable model of Q, iff S is also a p-stable model of $\mathsf{P}_{\odot}$, and
(2) if Q is p-stable consistent, then $\mathsf{P}_{\odot}$ is p-stable consistent too.

($\Rightarrow$) By hypothesis M is an $\odot$-update p-stable model of P, then by Definition 8, there exists a MG p-stable model M' of the abductive logic program of P, $\langle \mathsf{P}_{\odot}, A\rangle$, such that $M = M' \cap \mathcal{L}_{\mathsf{P}}$. Then by Definition 6, there exists Δ, $\Delta \subseteq A$ such that $\langle M', \Delta\rangle$ is a MG p-stable model of $\langle \mathsf{P}_{\odot}, A\rangle$, where $M = M' \cap \mathcal{L}_{\mathsf{P}}$.
By hypothesis, Q is p-stable consistent then, by (2) $\mathsf{P}_{\odot}$ is p-stable consistent too. Hence applying Lemma 2, it is possible to verify that $\Delta = \emptyset$ and $M' = M$. So $\langle M, \emptyset\rangle$ is a MEG p-stable model of $\langle \mathsf{P}_{\odot}, A\rangle$. Finally by Definition 5, M is a p-stable model of $\mathsf{P}_{\odot}$. Thus by (1) we have that M is a p-stable model of Q.
($\Leftarrow$) Let M be a p-stable model of Q. By (1), M is a p-stable model of $\mathsf{P}_{\odot}$. By Lemma 2, M is a MG p-stable model of the abductive logic program of P, $\langle \mathsf{P}_{\odot}, A\rangle$. By Lemma 2, $M \cap A = \emptyset$. Hence by Definition 8, M is an $\odot$-update p-stable model of P.
(**P3. Idempotence**): If P does not have p-stable models, then neither does $P \odot P$. If P has p-stable models, then $P \cup P$ does, hence by Strong Consistency, $P \cup P \equiv P \odot P$. Hence in each case $P \odot P \equiv P$.
(**P4. Weak noninterference**): If each of P_1 and P_2 lacks of p-stable models then the update (in any order) lacks of p-stable models. If P_1 and P_2 have p-stable models, then $P_1 \cup P_2$ does too — because they are defined over disjoint alphabets. By Strong Consistency, $P_1 \cup P_2 \equiv P_1 \odot P_2$. Also $P_2 \cup P_1 \equiv P_2 \odot P_1$. Hence, $P_1 \odot P_2 \equiv P_2 \odot P_1$.
(**P5. Weak irrelevance of syntax**): Let P, P_1, and P_2 be logic programs

under the same language $\mathcal{L}$. Since $P_1 \equiv_{SE} P_2$, then for every program P, $P \cup P_1$ is strongly equivalent to $P \cup P_2$. Thus, $(P \cup A) \cup P_1$ and $(P \cup A) \cup P_2$ have exactly the same p-stable models. Thus, $P \odot P_1$ and $P \odot P_2$ have exactly the same EG p-stable models. Therefore, $P \odot P_1$ and $P \odot P_2$ have exactly the same MG p-stable models. Hence, $P \odot P_1 \equiv P \odot P_2$.
(**P6. Augmented update**): If P_2 does not have p-stable models, neither does $P_1 \odot P_2$. If P_2 has at least one p-stable model and $P_1 \subseteq P_2$ then, $(P_1 \cup P_2)$ has at least one p-stable model too. By strong consistency $P_1 \odot P_2 \equiv P_1 \cup P_2$. Hence in each case $P_1 \odot P_2 \equiv P_2$.
(**P7. Independent parts**): Let $\mathsf{P}_1 = (P_1, P_1')$, $\mathsf{P}_2 = (P_2, P_2')$ such that $(\mathcal{L}_{\mathsf{P}_1} \cap \mathcal{L}_{\mathsf{P}_2}) = \emptyset$, and $\mathsf{P} = ((P_1 \cup P_2), (P_1' \cup P_2'))$. Let M_1 and M_2 be a $\odot$-update p-stable model of P_1 and a $\odot$-update p-stable model of P_2 respectively. It is clear that M_1 and M_2 are disjoint, since $(\mathcal{L}_{\mathsf{P}_1} \cap \mathcal{L}_{\mathsf{P}_2}) = \emptyset$. We have to prove that M is a $\odot$-update p-stable model of P iff $M = M_1 \cup M_2$.
($\Rightarrow$) By Definition 8, if M is a $\odot$-update p-stable model of P, then there exists M', a MG p-stable model of $\langle \mathsf{P}_\odot, B \rangle$, such that $M = M' \cap \mathcal{L}_{\mathsf{P}}$. Then by Definition 6, there exists Δ, $\Delta \subset B$ such that $\langle M', \Delta \rangle$ is a EG p-stable model of $\langle \mathsf{P}_\odot, B \rangle$ and it is minimal. By Definition 5, M' is a p-stable model of $\mathsf{P}_\odot \cup \Delta$.
Moreover, since $(\mathcal{L}_{\mathsf{P}_1} \cap \mathcal{L}_{\mathsf{P}_2}) = \emptyset$, we can verify the following:
(1) $\mathsf{P}_\odot = \mathsf{P}_{1\odot} \cup \mathsf{P}_{2\odot}$[6],
(2)$\Delta = \Delta_1 \cup \Delta_2$ such that $\Delta_1 = \Delta \cap \mathcal{L}_{\mathsf{P}_1}$, $\Delta_2 = \Delta \cap \mathcal{L}_{\mathsf{P}_2}$ and $\Delta_1 \cap \Delta_2 = \emptyset$,
(3)$M' = M_1' \cup M_2'$ such that M_1' is a p-stable model of $\mathsf{P}_{1\odot} \cup \Delta_1$ and M_2' is a p-stable model of $\mathsf{P}_{2\odot} \cup \Delta_2$.
Now by Definition 5, $\langle M_1', \Delta_1 \rangle$ is a MEG p-stable model of $\langle \mathsf{P}_{1\odot}, B_1 \rangle$ and $\langle M_2', \Delta_2 \rangle$ is a MGE p-stable model of $\langle \mathsf{P}_{2\odot}, B_2 \rangle$ where B_1 is the set of abducible atoms of $\mathsf{P}_{1\odot}$ and B_2 is the set of abducible atoms of $\mathsf{P}_{2\odot}$.
Finally by Definition 8, $M_1 = M_1' \cap \mathcal{L}_{\mathsf{P}_1}$ and $M_2 = M_2' \cap \mathcal{L}_{\mathsf{P}_2}$ are $\odot$-update p-stable models of P_1 and P_2 respectively.
($\Leftarrow$) This proof is similar to the proof of the first part above. Taking into account that $\mathsf{P}_\odot = \mathsf{P}_{1\odot} \cup \mathsf{P}_{2\odot}$; and if $\Delta = \Delta_1 \cup \Delta_2$ then there exists a p-stable model $M' = M_1' \cup M_2'$ of $\mathsf{P}_\odot \cup \Delta$ such that M_1' as a p-stable model of $\mathsf{P}_{1\odot} \cup \Delta_1$ and M_2' as a p-stable model of $\mathsf{P}_{2\odot} \cup \Delta_2$. □

5 Conclusions

Our approach for update logic program is based on the concept of Minimal generalized p-stable models, and we also present properties that our update operator satisfies. A comparative study of our semantics and other updates semantics will be realized as future work.

[6] This is possible if we select the appropriate abducibles from $\mathsf{P}_\odot$ to define $\mathsf{P}_{1\odot}$ and $\mathsf{P}_{2\odot}$ (see Definition 7).

Acknowledgement

This research has been supported by the Fondo Sectorial SEP-CONACyT, Ciencia Básica Project (Register 101581).

References

1. J. J. Alferes, F. Banti, A. Brogi, and J. A. Leite. The refined extension principle for semantics of dynamic logic programming. *Studia Logica*, 79(1):7–32, 2005.
2. M. Balduccini and M. Gelfond. Logic Programs with Consistency-Restoring Rules. In P. Doherty, J. McCarthy, and M.-A. Williams, editors, *International Symposium on Logical Formalization of Commonsense Reasoning*, AAAI 2003 Spring Symposium Series, Mar 2003.
3. J. L. Carballido, M. Osorio, and J. Arrazola. Equivalence for the G'$_3$-stable models semantics. *J. Applied Logic*, 8(1):82–96, 2010.
4. J. Delgrande, T. Schaub, and H. Tompits. A preference-based framework for updating logic programs. In C. Baral, G. Brewka, and J. Schlipf, editors, *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*, volume 4483 of *Lecture Notes in Artificial Intelligence*, pages 71–83. Springer-Verlag, 2007.
5. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. Considerations on updates of logic programs. In *JELIA '00: Proceedings of the European Workshop on Logics in Artificial Intelligence*, pages 2–20, London, UK, 2000. Springer-Verlag.
6. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. On properties of update sequences based on causal rejection. *Theory and Practice of Logic Programming*, 2(6):711–767, 2002.
7. 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.
8. A. C. Kakas and P. Mancarella. Generalized stable models: a semantics for abduction. In *Proceedings of ECAI-90*, pages 385–391. IOS Press, 1990.
9. M. Osorio, J. Arrazola, and J. L. Carballido. Logical weak completions of paraconsistent logics. *Journal of Logic and Computation*, Published on line on May 9, 2008.
10. M. Osorio and J. L. Carballido. Brief study of G'$_3$ logic. *Journal of Applied Non-Classical Logic*, 18(4):79–103, 2009.
11. M. Osorio and V. Cuevas. Updates in answer set programming: An approach based on basic structural properties. *Theory and Practice of Logic Programming*, 7(04):451–479, July 2007.
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. F. Zacarias, M. O. Galindo, J. C. A. Guadarrama, and J. Dix. Updates in Answer Set Programming based on structural properties. In *Proceedings of the 7th International Symposium on Logical Formalizations of Commonsense Reasoning. Dresden University Technical Report*, pages 213–219, Corfu, Greece, May 2005. TU-Dresden, Fakultt Informatik.
14. C. Zepeda, M. Osorio, J. C. Nieves, C. Solnon, and D. Sol. Applications of preferences using answer set programming. In *Answer Set Programming: Advances in Theory and Implementation (ASP 2005)*, pages 318–332, University of Bath, UK, July 2005.