New properties of the update operator "$\oplus$"

Mauricio Osorio[1] and Fernando Zacarías[2]

Universidad de las Américas, CENTIA.
Sta. Catarina Mártir, Cholula, Puebla
72820 México
[1] josorio@mail.udlap.mx
http://mailweb.udlap.mx/~josorio
[2] fzflores@siu.buap.mx

Abstract. We have studied the update operator $\oplus$ defined in [8] without tautologies and we have observed that satisfies an interesting property. This property is similar to one postulate proposed by AGM but, in this case for nonmonotonic logic and that we called WIS. Also, we consider other five additional basic properties about update programs and we show that $\oplus$ satisfies them. So, this work continues the analysis about the AGM postulates with respect to operator $\oplus$ under the refinated view that includes knowledge and beliefs that we began in a recent previous paper and that satisfies the WIS property for closed programs under tautologies.

Keywords: Answer set programming; *Nelson* logic; Update programs; Strong negation; AGM postulates; Properties.

1 Introduction

A-Prolog (Stable Logic Programming [10] or Answer Set Programming) is the realization of much theoretical work on Nonmonotonic Reasoning and AI applications of Logic Programming (LP) in the last 15 years. This is an important logic programming paradigm that has now great acceptance in the community. Efficient software to compute answer sets and a large list of applications to model real life problems justify this assertion. The two most well-known systems that compute Answer sets are DLV [6] and SMODELS [22]. It has been recently provided a characterization of answer sets by intuitionistic logic as follows: a literal is entailed by a program in the stable model semantics if and only if it belongs to every intuitionistically complete and consistent extension of the program formed by adding only negated literals [19]. The idea of these completions using in general intermediate logics is due to Pearce [21]. This logical approach provides the foundations to define the notion of nonmonotonic inference of any propositional theory (using the standard connectives) in terms of a monotonic logic (namely intuitionistic logic), see [21, 19, 18]. The proposed interpretation would be the following: Given a theory T, its knowledge is understood as the formulas F such that F is derived in T using intuitionistic logic. This makes

sense, since in intuitionistic logic according to Brouwer, F is identified with "I knows F" (or perhaps some reader would prefer to understand the notion of "knowledge" as "justified belief"). An agent whose knowledge base is the theory T believes F if and only if F belongs to every intuitionistically complete and consistent extension of T by adding only negated literals (here "belief" could be better interpreted as "coherent" belief). Take for instance: $\neg a \rightarrow b$. The agent knows $\neg a \rightarrow b$, $\neg b \rightarrow \neg\neg a$ and so on and so forth. The agent does not know however a. Nevertheless, one believes more than one knows, but a cautious agent must have its beliefs consistent to its knowledge. This agent will then assume negated literals to be able to infer more information. Thus, in our example, our agent will believe $\neg a$ and so he/she can conclude b. It also makes sense that a cautious agent will believe $\neg a$ or $\neg\neg a$ rather than to believe a (recall that a is not equivalent to $\neg\neg a$ in intuitionistic logic). This view seems to agree with a point of view by Kowalski, namely that "Logic and LP need to be put into place: Logic within the thinking component of the observation-thought-action cycle of a single agent, and LP within the belief component of thought" [13]. As Pearce noticed, if we include strong negation we just have to move to Nelson logics [21]. We select here N, the least constructive (strong negation) extension of intuitionistic logic and because it is the minimum necessary to satisfy our properties. Also, N is the nearest Intuitionistic logic. For this reason we don't need of N2. We say that two theories T_1 and T_2 are equivalent knowledge if they are equivalent in N. We denote this fact by $T_1 \equiv_K T_2$. We say that T_1 and T_2 are equivalent if they have the same answer sets. We denote this fact by $T_1 \equiv T_2$.

In this paper we address our approach to update nonmonotonic knowledge bases represented as extended logic programs under the answer set semantics, two very good recent reviews with many references are [3, 8]. If new knowledge of the world is somehow obtained, and it doesn't have conflicts with the previous knowledge then this new knowledge only expands knowledge. If by the contrary, new knowledge is inconsistent with the previous knowledge, and we want knowledge to be always consistent so that our agents can act in all moment, we should solve this problem somehow. We point out that new information is incorporated into the current knowledge base subject to a causal rejection principle, which enforces that, in case of conflicts between rules, more recent rules are preferred and older rules are overridden. Some well-known proposals are presented in [8] and [2]. In particular [8] presents a complete analysis with respect to properties that an update operator should have, with the purpose of obtaining a sure and reliable update process for our agents. In this context, it is necessary to point out that when one wants to choose a theory to develop its applications is very important to know the properties that in the theory are held. It is in this sense that we have focused to investigate the properties that are held under our theory, and not to present properties that it doesn't hold.

In this paper, we consider similar properties to the well-known AGM postulates. We think that is necessary to reinterpret them on the context of nonmonotonic reasoning via answer set programming. In addition, we pay particular attention to our view that distinguishes beliefs from justified beliefs. As a be-

ginning, we only study Dalal's principle of irrelevance of syntax, that according to Dalal's Principle [12] of Irrelevance of Syntax, the meaning of the knowledge that results from an update must be independent of the syntax of the original knowledge, as well as independent of the syntax of the update itself. In [8] the authors analyze and interpret the AGM postulate corresponding to Dalal's principle as follows:

$$T_1 \equiv T_2 \text{ implies } Bel(K \odot T_1) = Bel(K \odot T_2).$$

Where T_1 and T_2 are any theories, $Bel(T)$ defines the set of answer sets of T, $\odot$ is the revision operator, and understanding that equivalence means that both programs (T_1 *and* T_2) have the same answer sets. This interpretation expresses a very demanding principle of irrelevance of syntax, due to that the AGM postulates were introduced for monotonic logics. We propose to reconsider the AGM postulates [1] under our new interpretation that considers "justified beliefs" and "belief". To this aim we have introduced in [20] a new property, which we call Weak Irrelevance of Syntax, as follows:

$$\text{(WIS): } T_1 \equiv_K T_2 \text{ implies } Bel(K \odot T_1) = Bel(K \odot T_2).$$

We show that the proposal shown in [8] for updates almost satisfies this principle. In fact we show that for programs without tautologies, this principle holds. We should point out that this property is accepted as much in belief revision as in updates, as it is shown in [8]. Also, [8] notes, however, that tautological rules in updates are, as we believe, rare in practical applications and can be eliminated easily.

Our paper is structured as follows: In section 2 we present the general syntax of clauses, we also provide the definition of answer sets for augmented logic programs as well as some background on logic, in particular on N logic. Section 3 contains the definition about update programs given in [8] and some related concepts. Next, in section 4 we introduce our principal property called WIS and introduce our main contribution in this respect. Section 5 contains some additional properties about update operator. Related work and some considerations about our proposal are presented in 6. Finally, the conclusions are drawn in section 7.

2 Background

In this section, we give some general definitions for our theory. We define our theory about logic program, which consists of rules built over a finite set $\mathcal{A}$ of propositional atoms, where these programs can only contain default negation. Later, we introduce strong negation in similar form as in [15].

2.1 Preliminary

Rules are built from propositional atoms and the 0-place connectives $\top$ and $\bot$ using negation as failure (*not*) and conjunction (,). A *rule* is an expression of the form

$$Head \leftarrow Body \tag{1}$$

If *Body* is $\top$ then we identify rule 1 with rule *Head*. If a *Head* is $\bot$ then we identify rule 1 with a constraint. A *program* is a set of rules. An *Interpretation* I is a set of atoms A such that $I(A) = \top$. It allows us to establish the scenario to determine if a complex rule is $True$ or $False$. We define satisfaction of formulas as $I \models L$ iff $L \in I$, where L is an atom. We restrict our attention to finite logic programs. For a program P, I is a *model* of P, denoted $I \models P$, if $I \models L$ for all rules $L \in P$. As it is shown in [5] the Gelfond-Lifschitz transformation says that for a program P and a model $N \subseteq B_P$ (B_P denotes the set of atoms that appear in P) is defined by

$$P^N = \{rule^N : rule \in P\}$$

where

$(\ A \leftarrow B_1, ..., B_m,\ not\ C_1, ...,\ not\ C_n)^N$ is either:

a) $A \leftarrow B_1, ..., B_m$, if $\forall j \leq n : C_j \notin N$;

b) $\top$, otherwise.

Note that P^N is always a *definite* program (i.e., a program consisting of positive atoms only). We can therefore compute its least Herbrand model (denoted as M_{P^N}) and check whether it coincides with the model N which we started with.

Definition 1. *(STABLE [5]) N is an stable model of P iff N is the minimal model of* P^N.

2.2 Adding strong negation

With respect to strong negation, syntactically, the status of the strong negation operator $\sim$ is different from the status of the operator "*not*" the difference is the following: *not p* can be denoted by $p \leftarrow \bot$, i.e., we use "*not*" when evidence doesn't exist about p. And we use $\sim p$ when we know that p is false or it doesn't happen. We can say that answer sets are usually defined for logic programs possessing this second kind of negation, that as we mentioned previously expresses the direct or explicit falsity of an atom. In [11] this second negation is called "classical" and is denoted by $\sim$.

Let $\mathcal{A}$ be a set of propositional atoms where both default negation *not* and strong negation $\sim$ is available. For a set S of literals, we define $\sim S =$

$\{\sim L \mid L \in S\}$, and denote by $Lit_{\mathcal{A}}$ the set $\mathcal{A} \cup \sim\mathcal{A}$ for all literals over $\mathcal{A}$. A literal preceded by *not* is called a *weakly negated literal*.

Therefore, a rule is an expression of the form:

$$A \leftarrow B_1, ..., B_m, notB_{m+1}, ..., notB_n. \tag{2}$$

Where A and each B_i are literals. A literal L is either an atom A (a positive literal) or a strongly negated atom $\sim A$ (a negative literal), we call Extended Logic Programs (ELP's) to a set of type rules (2). For a rule r of this form we define $H(r) = \{A\}$ and $B(r) = \{B_1, ..., B_m, \ not \ B_{m+1}, ..., \ not \ B_n\}$.

We denote by $\mathcal{L}_{\mathcal{A}}$ the set of all constructible rules using the literals in $Lit_{\mathcal{A}}$. By $S(P)$ we denote the collection of all answer sets of P. If $S(P) \neq \emptyset$, then P is *satisfiable*. Following [8], we regard a logic program P as the *epistemic state* of an agent. The given semantics are used for assigning a *belief set* to any epistemic state P as follows.

Let $I \subseteq Lit_{\mathcal{A}}$ be an interpretation. Define $Bel_{\mathcal{A}}\ (I) = \{\mathrm{r} \in \mathcal{L}_{\mathcal{A}} \mid I \models \mathrm{r}\}$

Furthermore, for a class $\mathcal{I}$ of interpretations, let $Bel_{\mathcal{A}}(\mathcal{I})=\cap_{I\in\mathcal{I}} Bel_{\mathcal{A}}(I)$

Definition 2. *[8] For a logic program P, the belief set, $Bel_{\mathcal{A}}(P)$, of P is given by $Bel_{\mathcal{A}}(P) = Bel_{\mathcal{A}}(S(P))$.*

We write $P \models_{\mathcal{A}} r$ if $r \in Bel_{\mathcal{A}}(P)$, and for any program Q, we write $P \models_{\mathcal{A}} Q$ if $P \models_{\mathcal{A}} q$ for all $q \in Q$. Programs P_1 and P_2 are equivalent (modulo $\mathcal{A}$), symbolically $P_1 \equiv^{\mathcal{A}} P_2$, iff $Bel_{\mathcal{A}}(P_1) = Bel_{\mathcal{A}}(P_2)$. Since $\mathcal{A}$ is finite, then $P_1 \equiv^{\mathcal{A}} P_2$ is equivalent to the condition that P_1 and P_2 have the same answer sets modulo $\mathcal{A}$. We will drop the subscript "$\mathcal{A}$" in $Bel_{\mathcal{A}}(\cdot)$, $\models_{\mathcal{A}}$, and $\equiv^{\mathcal{A}}$ if no ambiguity can arise.

2.3 *N* Logic

Now, we give a brief description about N logic, because this gives an alternative interpretation of theoretical foundation to ASP [21]. Recall that a natural deduction system for intuitionistic logic can be obtained from the corresponding classical system by dropping the law of the excluded middle

$$F \vee \neg F$$

from the list of postulates. N is the extension of Intuitionistic logic with strong negation and axioms of Nelson logic. A formalization of N_2 can be obtained from intuitionistic logic by adding the axiom schema

$$F \vee (F \rightarrow G) \vee \neg F$$

The correspondence between the language of logic programs and the language of propositional formulas in the presence of two negations is described in [15]. The main theorem in [15] readily generalizes to the new setting: we can show that two extended programs are strongly equivalent if and only if they are equivalent in the N_2 logic [15].

The next theorem is a simple corollary of results in [15]. However notice that we do not require strong equivalence. Hence we just need N logic and not the full power of N_2.

Theorem 1. *For any P_1 and P_2 programs, $P_1 \equiv_N P_2$ implies that for every P program, $P_1 \cup P$ and $P_2 \cup P$ have the same answer sets.*

We will write $P \vdash_K \alpha$ to denote the fact that $P \vdash_N \alpha$. The idea for using K instead of N is due to two reasons: First, to emphasize the reading P "knows" α. Second, because strictly speaking we are translating the connective symbols. Similarly we will write $P_1 \equiv_K P_2$ instead of $P_1 \equiv_N P_2$.

3 Update programs

In [8] the authors define an *update sequence*, P as a series of two programs (P_1, P_2) of extended logic programs (ELPs). We say that P is an update sequence over $\mathcal{A}$ iff $\mathcal{A}$ represents the set of atoms occurring in the rules of the constituting elements P_i of P ($1 \leq i \leq 2$).

Giving an update sequence P = (P_1, P_2) over $\mathcal{A}$, we assume a set $\mathcal{A}^*$ extending $\mathcal{A}$ by new, pairwise distinct atoms rej(r) and A_i, for each r occurring in P, each atom $A \in \mathcal{A}$, and each i, $1 \leq i \leq 2$. We further assume an injective naming function $N(\cdot, \cdot)$, which assigns to each rule r in a program P_i a distinguished name, $N(r, P_i)$, obeying the condition $N(r, P_i) \neq N(r', P_j)$ whenever $i \neq j$. With a slight abuse of notation we shall identify r with $N(r, P_i)$ as usual. Finally, for a literal L, we write L_i to denote the result of replacing the atomic formula A of L by A_i.

Let us consider the definition about the update sequence given by Eiter et al. but only in the case of two programs and let us make a slight change of notation. Also, our proposal can be extended to general case $(P_1, P_2, ..., P_n)$ in the iterative form as shown in [8]. Under certain conditions, which exclude possibilities for local inconsistencies, the iterativity property holds.

Definition 3. *[8] Giving an update of two programs $\boldsymbol{P}_{\oplus}$ = (P_1, P_2) over a set of atoms $\mathcal{A}$, we define the update program $\boldsymbol{P}_{\oplus} = P_1 \oplus P_2$ over $\mathcal{A}^*$ consisting of the following items:*

(*i*) *all constraints in* $P_1 \cup P_2$;
(*ii*) *for each* $r \in P_1$;
$L_1 \leftarrow B(r)$, *not* $rej(r)$. *if* $H(r) = L$;
$rej(r) \leftarrow B(r), \neg L_2$. *if* $H(r) = L$;
(*iii*) *for each* $r \in P_2$;
$L_2 \leftarrow B(r)$. *if* $H(r) = L$;
(*iv*) *for each literal* L *occurring in* $\boldsymbol{P}$;
$L_1 \leftarrow L_2 \qquad L \leftarrow L_1$.

Definition 4. *[8] Let* $\boldsymbol{P} = (P_1, P_2)$ *be an update sequence over a set of atoms* $\mathcal{A}$. *Then,* $S \subseteq Lit_{\mathcal{A}}$ *is an update answer set of* $\boldsymbol{P}$ *iff* $S = S' \cap \mathcal{A}$ *for some answer set* S' *of* $\boldsymbol{P}_{\oplus}$. *The collection of all update answer sets of* $\boldsymbol{P}$ *is denoted by* $\mathcal{U}(\boldsymbol{P})$.

Consider the following example taken of [8].

Example 1. Let P be: sleep ← *not* tv-on.
night.
tv-on.
watch-tv ← tv-on.
And let P_1 be ∼tv-on ← power-failure.
power-failure.

Applying definition 3 to both programs, we obtain:
The single answer set of $\mathbf{P} = (P, P_1)$ using definition 4 is,

S = {power-failure, ∼tv-on, sleep, night}, as desired,

since the only answer set of $P_{\oplus}$ is given by:

{sleep1, night1, rej-r3, ∼tv-on2, power-failure, power-failure2, ∼tv-on1, power-failure1, sleep, night, ∼tv-on}

Following the case of single programs, an update sequence $\mathbf{P} = (\mathrm{P}_1, \mathrm{P}_2)$ is regarded as the epistemic state of an agent, and the belief set $Bel(\mathbf{P})$ is given by $Bel(\mathcal{U}(\mathbf{P}))$. The update sequence $\mathbf{P}$ is said to be satisfiable iff $\mathcal{U}(\mathbf{P}) \neq \emptyset$, and $\mathbf{P} \equiv \mathbf{P'}$ iff $Bel(\mathbf{P}) = Bel(\mathbf{P'})$ (P' some update sequence).

Definition 5. *Let us call two rules* r_1 *and* r_2 *conflicting iff* $H(r_1) = \sim H(r_2)$.

Let us consider the definition given in [8] about the rejection set of S for two programs $(Rej(S, (P_1, P_2)))$, as follows:

Definition 6. *[8] Given* $\boldsymbol{P} = (P_1, P_2)$, *over a set of atoms* $\mathcal{A}$ *and* $S \subseteq Lit_{\mathcal{A}}$ *based on the principle of founded rule rejection, we define the rejection set of* S *by* $Rej(S, \boldsymbol{P}) = Rej_1(S, \boldsymbol{P})$ *and* $Rej_1(S, \boldsymbol{P}) = \{r \in P_1 \mid \exists r' \in P_2 \setminus Rej_2(S, \boldsymbol{P})$ *such that* r *and* r' *are conflicting and* $S \models B(r) \cup B(r')\}$.

Let us consider the definition given in [8] about the rejection set of S using a weaker notion of rejection sets for two programs $(Rej'(S, (P_1, P_2)))$, as follows:

Definition 7. *Given* $\boldsymbol{P} = (P_1, P_2)$, *then* $Rej'(S, \boldsymbol{P}) = \{r \in P_1 \mid \exists\ r' \in P_2$, *such that* r *and* r' *are conflicting and* $S \models B(r) \cup B(r')\}$.

Note: We can see easily that Rej and Rej' coincides in case for two programs, that is, given $\mathbf{P} = (P_1, P_2)$, $Rej'(S, \mathbf{P}) = Rej(S, \mathbf{P})$.

Lemma 1. *Let* $\boldsymbol{P} = (P_1, P_2)$ *be an update sequence over a set of atoms* $\mathcal{A}$ *and* $S \subseteq Lit_{\mathcal{A}}$ *a set of literals. Then,* S *is an answer set of* $\boldsymbol{P}$ *iff* S *is an answer set of* $(P_1 \cup P_2 \setminus Rej'(S, P))^S$.

Proof: Directly by theorem 4 given in [8] and the previous note.

It is necesary to point out that in the proposal [8], update programs do not satisfy many of the properties defined in the literature. This is partly explained by the nonmonotonicity of logic programs and the causal rejection principle embodied in the semantics, which strongly depends on the syntax of rules.

Next, we present an example where the "equivalence" between two programs, P_1 and P_2 is not enough to preserve the equivalence when we update each one of these programs with onother program P.

Let P_1 and P_2 be two equivalent programs and let P_3 be a program then $P_1 \oplus \mathrm{P}_3 \equiv \mathrm{P}_2 \oplus \mathrm{P}_3$ is $false$.

Let P_1 = { a ←b. } , let P_2 = { a ← a. , b ← b. } and let P_3 = {b.}

We have that the stable models of P_1 and P_2 are empty, then applying the update process we have

$\mathrm{Sem}(\mathrm{P}_1 \oplus \mathrm{P}_3)$ = { a, b } and $\mathrm{Sem}(\mathrm{P}_2 \oplus \mathrm{P}_3)$ = { b } therefore

$$\{ \text{a, b} \} \neq \{ \text{b} \}$$

Hence, $\mathrm{P}_3 \oplus \mathrm{P}_1 \equiv \mathrm{P}_3 \oplus \mathrm{P}_2$ is $false$.

Also $\mathrm{Sem}(\mathrm{P}_3 \oplus \mathrm{P}_1)$ = { a, b } and $\mathrm{Sem}(\mathrm{P}_3 \oplus \mathrm{P}_2)$ = { b } therefore

$$\{ \text{a, b} \} \neq \{ \text{b} \}$$

4 Weak Irrelevance of Syntax

Within our main results, we can see that the proposal presented in [8] satisfies WIS considering programs without tautologies.

Let us see an example of our first approach.

Example 2. This example shows how WIS fails.

Let P be	∼d.
	d←h.
Let P_1 be	h.
	d←d.
Let P_2 be	h.

Here, P_1 and P_2 are strongly equivalent.

However when we update $P \oplus P_1 = \{\text{h, d}\}$ and
the update $P \oplus P_2$ doesn't have stable models, therefore WIS fails.

Definition 8. *P is tau-free w.r.t. a signature L if no rule of the form $l \leftarrow l,\alpha$ belongs to P, where α could be empty.*

4.1 Main results

Lemma 2. *Let $P_1 \cup P_2 \cup \{c\}$ be a tau-free program, $r \in P_1$, $P_2 \models_K c$, $M \models B(r) \wedge B(c)$, r and c are conflicting rules, $M \models P_2$. Then $\exists\, r' \in P_2 \mid r$ and r' are conflicting rules and $M \models B(r) \wedge B(r')$.*

Proof: Let c be the formula $x \leftarrow \theta$, $P_2 \vdash_K x \leftarrow \theta$, as M models P_2 and M models θ then $P_2 \cup \{\theta\}$ is consistent and $P_2 \cup \{\theta\} \vdash_K x$. Hence, $\exists\, x \leftarrow \beta \in P_2$ such that $P_2 \cup \{\theta\} \vdash_K \beta$ (Also, because we are in Nelson Logic N). Since $M \models P_2$ and $M \models \theta$ then
M models $P_2 \cup \{\theta\}$, and so $M \models \beta$. The rest of the proof follows directly.

Lemma 3. *Let P_1, P_2 and $\{c\}$ be tau-free programs. If $P_2 \vdash_K c$ then $P_1 \oplus P_2 \equiv P_1 \oplus (P_2 \cup \{c\})$.*

Proof:

S is an answer set of $P_1 \oplus P_2$ iff	by lemma 1
S is an answer set of $(P_1 \cup P_2) \setminus Rej'(S, (P_1, P_2))$ iff	by note a
S is an answer set of $(P_1 \setminus Rej'(S, (P_1, P_2))) \cup P_2$ iff	by note b
S is an answer set of $(P_1 \setminus Rej'(S, (P_1, P_2 \cup \{c\}))) \cup P_2$ iff	by note c
S is an answer set of $(P_1 \setminus Rej'(S, (P_1, P_2 \cup \{c\}))) \cup (P_2 \cup \{c\})$ iff	by note a
S is an answer set of $(P_1 \cup (P_2 \cup \{c\}) \setminus Rej'(S, (P_1, P_2 \cup \{c\}))$ iff	by lemma 1
S is an answer set of $P_1 \oplus (P_2 \cup \{c\})$	

Hence $P_1 \oplus P_2 \equiv P_1 \oplus (P_2 \cup \{c\})$ as desired.

Note a: Since Rej' only erase clauses from P_1 (not from P_2).
Note b: $Rej'(S, (P_1, P_2)) = Rej'(S, (P_1, P_2 \cup \{c\}))$ by lemma 2.
Note c: Since $P_2 \models_K c$, $P_2 \equiv_K P_2 \cup \{c\}$.

Lemma 4. *Let P, P_1 and R tau-free programs. Suppose, that $P_1 \vdash_K R$ then $P \oplus P_1 \equiv P \oplus (P_1 \cup R)$.*

Proof: By induction on the size of R.

Base case: Let $R = \emptyset$, then the result is immediate.
Induction Hypothesis: Let P_1, R, and $\{c\}$ be tau-free programs. We need to show:

$$\begin{aligned} &\text{if } P_1 \vdash_K R \cup \{c\} \\ &\text{then } P \oplus P_1 \equiv P \oplus (P_1 \cup (R \cup \{c\})). \end{aligned} \tag{I}$$

But, we know that $P_1 \vdash_K R \cup \{c\}$ means that $P_1 \vdash_K R$ and $P_1 \vdash_K c$ then applying induction hypothesis

$$P \oplus P_1 \equiv P \oplus (P_1 \cup R) \tag{II}$$

By lemma 3, $P_1 \cup R \vdash_K c$ then

$$P \oplus (P_1 \cup R) \equiv P \oplus ((P_1 \cup R) \cup \{c\}) \tag{III}$$

Now, from (I), (II), and (III) we have $P \oplus P_1 \equiv P \oplus ((P_1 \cup R) \cup \{c\})$
Since $P \oplus ((P_1 \cup R) \cup \{c\}) \equiv P \oplus (P_1 \cup (R \cup \{c\}))$ we obtain
$P \oplus P_1 \equiv P \oplus (P_1 \cup (R \cup \{c\}))$ as desired.

Theorem 2. *Let P, P_1 and P_2 be tau-free programs, if $P_1 \equiv_K P_2$ then $P \oplus P_1 \equiv P \oplus P_2$*

Proof: i) $P \oplus P_1 \equiv P \oplus (P_1 \cup P_2)$ applying lemma 4

Besides, if $P_1 \equiv_K P_2$ then $P_2 \equiv_K P_1$ and applying lemma 4 to $P \oplus P_2$ we have

ii) $P \oplus P_2 \equiv P \oplus (P_2 \cup P_1)$

Also, $P \oplus (P_1 \cup P_2) = P \oplus (P_2 \cup P_1)$ (module renaming)

Finally, by transitivity between i) and ii) we have $P \oplus P_1 \equiv P \oplus P_2$ as desired.

It is necessary to stand out that only lemma 3 depends on the properties of the operator. Therefore, we can say that any operator satisfying lemma 3 would have the WIS property, of course, assuming the answer set semantics.

5 Properties of update operator

As we have mentioned, the interpretation given in [8] of the AGM postulates expresses a very demanding principle of irrelevance of syntax, because of the AGM postulates were introduced for monotonic logics. After having revised several proposals about update programs such as [3, 8, 1], we have some interesting

properties of the style of the AGM postulates for update programs, but in our context of answer set semantics that consider the two notions: *Belief* and *Knowledge*. We call them BK-ASP properties.

Definition 9 (BK-ASP).
K1: $P \oplus x$ *is a theory.*
K2: $P \oplus x \vdash_K x$.
K3: $x \equiv_K \bot$ *implies* $(P \oplus x) \equiv_K \bot$.
K4: $P + x \vdash_K P \oplus x$.
K5: if $P_1 \equiv_K P_2$ *then* $P \oplus P_1 \equiv P \oplus P_2$. *(WIS)*
K6: if $P_1 \vdash_K R$ *then* $(P \oplus P_1) \cup R \equiv P \oplus (P_1 \cup R)$.

We consider a theory as a logic program. As we can see, our second (*K2*) property guarantees that the input sentence x is accepted in $P \oplus x$.

With respect to our third property (*K3*), it says that if x is inconsistent (at the knowledge level) then $(P \oplus x)$ can not be consistent.

With respect to our fourth property (*K4*), it says that an expansion always knows more (or equal) than an update.

Our fifth (*K5*) property says that update should be analyzed on the knowledge level and not on the syntactic level. For this reason, two logically equivalent sentences (at the knowledge level) should lead to equivalent updates (at the belief level). This is the most interesting property in our proposal and it is resolved and supported by our theorem 2.

Next, we present our main result with respect to BK-ASP properties.

Theorem 3. *The update operator* $\oplus$ *satisfies the six BK-ASP properties for tau-free programs.*

Proof: Our first four properties follow directly by construction. The fifth property follows by theorem 2. The last property can be proven as follows:
Under the assumption $P_1 \vdash_K R$ is easy to check that $P \oplus P_1 \equiv (P \oplus P_1) \cup R$, by lemma 4 $(P \oplus P_1) \cup R \equiv P \oplus (P_1 \cup R)$, as desired.

6 Related work

With respect to related work we can point out that in this work we continue the analysis about the AGM postulates begun in [20]. There, we have analyzed the update operator " $\oplus$ " defined in [8] under the refinated view that includes knowledge and beliefs. In [20], we have defined the WIS property and we have proven that it is satisfied for closed programs under tautologies. There, we have presented a more simple definition about the update operator that satisfies the WIS property for closed programs under tautologies and it is defined as follows:

Definition 10. *Given an update of two programs* $\boldsymbol{P}_{\otimes} = (P_1, P_2)$ *over a set of atoms* A*, we define the update program* $\boldsymbol{P}_{\otimes} = P_1 \otimes P_2$ *over* A^* *consisting of the following items:*

(i) *all constraints in* $P_1 \cup P_2$*;*
(ii) *for each* $r \in P_1$*;*
$\quad L \leftarrow B(r),\ not \sim L. \qquad$ *if* $H(r) = L$*;*
(iii) *all rules* r *in* P_2*.*

In the definition 10 we have used " $\otimes$ " as update operator. As we can see, this alternative definition for closed programs under tautologies is simple and practical for this class of programs. This definition is an interesting result that can be used in practical applications due to its simplicity. Also, allows to the agents to respond in a quick and opportune way.

Furthermore, we give some considerations about our presented work obtained after having made an analysis on the proposals presented in [1–4, 8, 12]. In [8] the authors analyze and interpret the AGM postulate corresponding to WIS and define an update operator that almost satisfies this postulate. In [8] the authors present an exhaustive analysis about the AGM postulates and about the different interpretations that some other authors have presented of these postulates. It's as from this analysis that we have carried out our interpretation of the AGM postulates considering the new results and extensions that, in ASP, have recently arisen. In this reinterpretation we consider the AGM postulates from the non-monotonic logic's point of view via ASP. Besides, we use a new interpretation about "justified beliefs" (also called knowledge) and "beliefs", that in ASP, this is done in a natural way. Also, we have used recent results obtained in [19, 15] on intuicionistic logic and intermediate logics.

With respect to Nelson logics, we prefer only N logic, because N logic is sufficient to satisfy our properties. Also, we want to keep up the closest possible to intuitionistic logic, which, in knowledge representation is more accepted and acknowledged by many authors. Due to this reason we prefer use N logic since it is the minimum constructive extension of intuitionistic logic sufficient for the properties here presented to hold. If we considered N_2 as the theory to use, we would have the next dilemma: N logic includes the following tautology: $(a \rightarrow b) \vee (b \rightarrow a)$ that means we know $(a \rightarrow b)$ or else $(b \rightarrow a)$. However, we really don't know neither $(a \rightarrow b)$ nor $(b \rightarrow a)$, moreover, we don't know neither a nor b. We may even consider a and b two disjointed sentences and though the tautology holds (It can be proved making its true table). It is necessary to point out that when one wants to choose a theory to develop its applications is very important to know the properties that in the theory are held. It is in this sense that we have focused on investigating the properties that are held under our theory.

We remark that if we consider Theorem 7 presented in [8], we can see easily that this theorem allows us under certain conditions, which exclude such posibilities for local inconsistencies, to uses the update operator iteratively. Furthermore, the conditions of theorem 7 are simple syntactic criteria, which can

be easily checked [8]. Is worth to mention that a weaker version of this Theorem 7 may be applied if updates should be incorporated instantaneously by only considering condition (*iii*) of the mentioned theorem. Also, we can incorporate consecutive updates which obey assertion (*iii*) is equivalent to the update program for the sequence of updates.

7 Conclusions

We studied new properties of the update operator. Different from other approaches we considered a view of answer sets based on Nelson logics. This allowed us to reconsider the AGM postulates in a more solid framework. Our main contribution is the proposal of six properties for an update operator and the proof that $\oplus$ satisfies all of them. However, we should continue working in this line, since there are properties such as the following:

$$P \oplus P_1 \oplus Q \equiv P \oplus P_2 \oplus Q$$

for every program P and Q, that are not always satisfied, due to our property is just satisfied by the right.

8 Acknowledgements

We are grateful to Michael Fink for some discussions that helped us to clarify our ideas around updates.

References

1. C.E. Alchourron, P. Gardenfors, and D. Makinson. On the logic of Theory Change, Partial Meet Functions for Contraction and Revision Functions. *Journal of Symbolic Logic*, vol. 50 pp. 510–530, 1985.
2. J. Alferes, J. Leite, P. Pereira, H. Przymusinska, and T. Przymusinski. Dynamic Logic Programming. In A. Cohn and L. Schubert, editors. Proc. KR98, pp. 98–109. Morgan Kaufmann, 1998.
3. J. J. Alferes, L. M. Pereira, Logic Programming Updating - a guided approach, in: A. Kakas, F. Sadri (eds.), Computational Logic: From Logic Programming into the Future - Essays in honour of Robert Kowalski, volume 2, pp. 382–412, Springer LNAI 2408, 2002.
4. G. Brewka. Declarative Representation of Revision Strategies. In Proc. Fourteenth European Conference on Artificial Intelligence (ECAI 2000), 2000.
5. G. Brewka, J. Dix, and K. Knonolige. Nonmonotonic Reasoning: An overview. CSLI Publication Eds. Leland Stanford Junior University, 1997.
6. DLV System: http://www.dbai.tuwien.ac.at/proj/dlv/
7. A. Darwiche and J. Pearl. On the Logic of Iterated Belief Revision. Artificial Intelligence, vol.89(1–2): pp. 1–29, 1997.

8. T. Eiter, M. Fink, G. Sabattini, and H. Thompits. Considerations on Updates of Logic Programs. In M.O. Aciego, L.P. de Guzmn, G. Brewka, and L.M. Pereira, editors, Proc. Seventh European Workshop on Logic in Artificial Intelligence JELIA 2000, vol. 1919 in LNAI, Springer 2000.
9. P. Gardenfors. Belief Revision : An Introduction. Cognitive Science, Department of Philosophy, Lund University, S–223, 50 Lund, Sweden, 1995.
10. M. Gelfond and V. Lifschitz. The stable model semantics for logic programs. Proceedings of the Fifth International Conference on Logic Programming 2 MIT Press. Cambridge, Ma. pp.1070–1080.
11. M. Gelfond and V. Lifschitz. Clasical negation in logic programs and Disjunctive databases. New Generation Computing. pp. 365–387, 1991.
12. H. Katsumo and A.O. Mendelzon. Propositional knowledge base revision and minimal change, Artificial Intelligence vol. 52, pp. 263–294, Elsevier, 1991.
13. R. Kowalski. Is logic really dead or just sleeping. In Proceedings of the 17th International Conference on Logic Programming, pages 2–3, 2001.
14. J. A. Leite, J. J. Alferes and L. M. Pereira, Multi-dimensional Dynamic Logic Programming, In F. Sadri and K. Satoh (eds.), Procs. of the CL-2000 Workshop on Computational Logic in Multi-Agent Systems (CLIMA'00), London, England, July 2000.
15. V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. ACM Transactions on Computational Logic, 2:526–541, 2001.
16. V. Lifschitz and T.Y.C. Woo. Answer sets in general nonmonotonic reasoning. In proc. Of IJCAI-91, 1991.
17. W. Marek and V. Subrahmanian. The relationship Between Logic Program Semantics and Non-monotonic Reasoning, in G. Levi and M. Martelli (eds.), Proc. of the 6. Int. Conf. on Logic Programming. MIT, 600–617, 1989.
18. M. Osorio, J.A. Navarro, and J. Arrazola. Equivalence in Answer Set Programming (extended version), Proceedings of LOPSTR 01, LNCS 2372, pp.57–75, Springer-Verlag, Paphos, Cyprus, November 2001.
19. M. Osorio, J.A. Navarro, and J. Arrazola. Applications of Intuitionistic Logic in Answer Set Programming, accepted in Journal of TPLP , 2003.
20. M. Osorio. and F. Zacarias. *"Irrelevance of Syntax in updating answer set programs"*, to appear in Workshop on Logic and Agents into Proc. of Fourth Mexican International Conference on Computer Science (ENC' 03) Apizaco Tlaxcala, Mxico 2003.
21. D. Pearce. From Here to There: Stable negation in Logic Programming, in D. Gabbay, H. Wansing Eds. What is Negation? Kluwer Academic Publishers, Dordrecht, forthcoming. 1999.
22. SMODELS System: http://saturn.hut.fi/pub/smodels/