Updates in Answer Set Programming based on structural properties
Fernando Zacarías, Mauricio Osorio, J. C. Acosta Guadarrama* and Jürgen Dix
U. Autónoma de Puebla, 14 Sur y Av. San Claudio, PUE. Mexico,
fzflores@siu.buap.mx
U. of the Americas, Puebla, Sta. Catarina Mártir, Cholula, PUE. 72820 Mexico,
josorio@mail.udlap.mx
Clausthal University of Technology, Julius-Albert-Str. 4, Clausthal-Zellerfeld, 38678 Germany,
{guadarrama, dix}@in.tu-clausthal.de

Abstract
Revising and updating knowledge bases is an important issue in knowledge representation and reasoning. Various proposals have been made recently for updating logic programs, in particular with respect to answer set programming. So far, most of these approaches are based on the causality rejection principle but most of them are showing an unintuitive behaviour. Our update semantics (based on minimal generalised answer sets) satisfies several structural properties and avoids problems of the other proposals. In addition we introduce some new properties that we consider an updating/dynamic semantics should fulfill too: Weak Irrelevance of Syntax and Strong Consistency. We compare our approach with the well-known upd operator due to Eiter et al. and show that it satisfies the new properties.

Keywords: Answer set programming; $N_2$ logic; Updates; Properties.

1 Introduction
In the last few years, a lot of work on updating knowledge in the context of logic programming focused on semantics satisfying certain structural properties [Zacarías Flores, 2005; Osorio and Zacarías, 2004; Alferes et al., 2004; Leite, 2003; Eiter et al., 2002]. This dates back to ideas originally introduced by Makinson, Kraus, Lehmann and Magidor and investigated in detail for logic programming semantics by Dix (see [Dix, 1995; Brewka and Dix, 1998; Dix et al., 2001]).

However, as pointed out in [Alferes et al., 2004], despite several existing semantics for updates, there is still no common agreement on which is the “right” semantics. Some authors have tackled this problem by a detailed analysis and comparison of different semantics based on structural properties (see [Eiter et al., 2000; Leite, 2003]).

We believe that, besides the properties described in [Eiter et al., 2000; Leite, 2003], other important properties are necessary to test the adequacy of semantics for logic program updates. So we have started to work in this direction in [Osorio and Zacarías, 2003; 2004].

Regarding the structural properties for updates that we consider a semantics should fulfill, we have one that says that if we can update a theory $T$ by $T_1$, the result should only depend on the logical contents of $T_1$, and not on the particular syntax used to write $T_1$. This property is called Weak Irrelevance of Syntax (WIS in short).

In addition, Pearce [Lifschitz et al., 2001] noticed that answer sets can be expressed naturally in $N_2$ (obtained from intuitionistic logic by adding the axiom schema $F \lor (F \rightarrow G) \lor \neg G$ and axioms of Nelson logic—see [Rasiowa, 1974] for more details). As a consequence one can define two theories $T_1$ and $T_2$ to be equivalent wrt. $N_2$, if they are equivalent in $N_2$: $T_1 \equiv_{N_2} T_2$.

From the viewpoint of answer set programming, however, $T_1$ and $T_2$ are equivalent if they have the same answer sets, denoted by $T_1 \equiv T_2$ (there are some other notions of equivalence, notably uniform and strong equivalence, but for our purposes the notion just defined is sufficient).

Accordingly we combine these approaches to update nonmonotonic knowledge bases represented as extended logic programs under the answer set semantics and define the property of Weak Irrelevance of Syntax.

Definition 1.1. (WIS): If $T_1 \equiv_{N_2} T_2$ then $T \odot T_1 \equiv T \odot T_2$ where $\odot$ represents our update operator.

Note that $\equiv_{N_2}$ is much stronger than $\equiv$. Replacing $\equiv_{N_2}$ by $\equiv$ does not make sense as can be seen by the following example: $T := \{b\}$, $T_1 := \{a \leftarrow b \leftrightarrow b\}$, $T_2 := \{a \leftarrow \neg b, b \leftarrow b\}$.

Moreover, in this paper we introduce a definition for updates based on the notion of minimal generalized answer sets that satisfies WIS, as well as a new property that we call Strong Consistency. We show that the latter, together with a set of basic structural properties [Osorio and Zacarías, 2004], is satisfied for this definition.

Intuitively, Strong Consistency states that the addition of rules like $\{a \leftarrow b, b \leftarrow a\}$, should not result in any additional answer sets.

Consider the following example, inspired from [Alferes et al., 2004], describing some beliefs about the sky.

Example 1.1. Let $P_1$ be:

\[
day \leftarrow \neg \text{night}
\]
The only answer set is \{day, \neg see\text{(stars)}\}.

But consider the following program and suppose that \(P_1\) is updated with it.

Let \(P_2\) be:

\[
\begin{align*}
\text{see\text{(stars)}} & \leftarrow \text{see\text{(constellations)}} \\
\text{see\text{(constellations)}} & \leftarrow \text{see\text{(stars)}}
\end{align*}
\]

As we can see, \(P_2\) contains only one new constant \text{constellations} and a new atom \text{“see\text{(constellations)}”} with respect to \(P_1\). Moreover, \text{see\text{(constellations)}} is considered synonymous with \text{see\text{(stars)}} by the two defining rules (note there are no other rules mentioning \text{see\text{(constellations)}}).

Thus this can be considered a conservative extension of \(P_1\): the language is extended but all answer sets should be extensions of the old answer sets: \text{see\text{(constellations)}} ought to be true in any of them iff \text{see\text{(stars)}} is true in it. However, according to [Alferes et al., 2004], \(P_2\) introduces a new answer set for nearly all existing update-recipes: \text{see\text{(stars)}, see\text{(constellations), night}}, which does not coincide with our intuition. The reason is that although, intuitively, \text{see\text{(stars)}} can not be true (because of the constraint) introducing \text{see\text{(constellations)}} gives another reason for \text{see\text{(stars)}} to be true. In the semantics cited so far, an additional answer set is introduced.

One can think of several principles relating to conservative extensions (extension-by-definition) to make sure that this does not happen. In our approach, we formulate later on the stronger property of **Strong Consistency** to avoid such a behaviour.

The paper is structured as follows. In the next section we introduce the general syntax of our framework. We then introduce (Section 3) a new definition for updates based on the notion of minimal generalised answer sets and show that it satisfies WIS. Section 4 contains our main results. They include a set of basic structural properties, as well as the Weak Irrelevance of Syntax and a property called Conservative Extension. We also compare our approach to the well-known upd operator (due to Eiter et al.). In Section 5 we consider logic programming with ordered disjunctions. Finally, the conclusions are contained in Section 6.

### 2 Basic Notation and Background

We consider logic programs consisting of rules built over a finite set \(A\) of propositional atoms \(A\). Negative atoms \(\neg A\) (weakly negated atoms) correspond to default negation. We then introduce strong negation as done in [Osorio and Zacarías, 2004].

**Formulas** are built from propositional atoms, the propositional constants \(\top\) and \(\bot\), using negation (represented by \(\neg\)) and conjunction (represented by a comma “,”). A *rule* is an expression of the form:

\[
A \leftarrow B_1, \ldots, B_m, \neg B_{m+1}, \ldots, \neg B_n
\]  

where \(A\) and \(B_i\) are atoms. \(\neg B\) is also called weakly negated.

If \(B_1, \ldots, B_m, \neg B_{m+1}, \ldots, \neg B_n\) is \(\top\) then we identify rule (1) with \(A\). If \(A\) is \(\bot\) then the rule (1) can be seen as a constraint. A *program* is a finite set of rules.

#### 2.1 Adding strong negation

Strong negation is denoted by a unary connective “\(\neg\)”. Syntactically, the status of the strong negation operator “\(\neg\)” is both intuitionistically and epistemologically different from the status of operator “\(\neg\)”. The difference is the following: not \(p\) can be denoted by \(\bot \leftarrow p\), i.e., we use “\(\bot\)” when it is believed that there is no evidence about \(p\). In contrast, we use “\(\neg p\)” when we know that \(p\) does not exist, is false or doesn’t happen.

Answer Sets are usually defined for logic programs possessing both default negation “\(\neg\)” and the second kind of negation (called strong negation) just introduced. A literal, \(L\), is either an atom \(A\) (a positive literal) or a strongly negated atom \(\neg A\) (a negative literal).

A literal \(L\), the **complement** of \(L\), is \(\neg L\). If \(L\) \(\neg\neg L\), we use the logic program \(\neg A\). For a set \(S\) of literals, we define \(\neg S = \{\neg \, L \mid L \in S\}\). Finally, give a set of literals \(A\) and a program \(P\), we denote by \(\neg A = \{\neg a \mid a \in A\}\) and we define \(\neg A\) \(\text{Lit}_P \setminus A\). In a similar way as [Osorio and Zacarías, 2004], we use the logic \(N_2\) in this paper.

#### 2.2 Answer Sets

In this paper, we use the Gelfond-Lifschitz transformation as used e.g. in [Brewka et al., 1997]. However, we need to generalise this definition of answer sets in a similar way as done in [Pearce, 1999], where the author has given a characterisation in terms of certain non-classical logics. His definition (taken from [Osorio et al., 2004a]) gives a complete characterisation of answer sets for any theory.

We need the following notation: \(T \vdash_{N_2} M\) is shorthand for \(T\) is consistent (as a \(N_2\) theory) and \(T \vdash_{N_2} M\).

**Theorem 2.1 (Characterisation of Answer Sets, [Osorio et al., 2004a])**. Let \(P\) be any program and \(M\) a consistent set of literals. \(M\) is an answer set for \(P\) if \(P \cup \neg M \cup \neg\neg M \vdash_{N_2} M\).

#### 2.3 Minimal Generalized Answer Sets

In this section we recapitulate some basic definitions about syntax and semantics of abductive logic programs. These semantics are given by minimal generalised answer sets (MGAS), which provide a more general and flexible semantics than standard answer sets.

**Definition 2.1 (Abductive Logic Program, [Balduccini and Gelfond, 2003])**. An abductive logic program is a pair \((P, A)\) where \(P\) is an arbitrary program and \(A\) a set of literals, called abducibles.
Definition 2.2 (Generalized Answer Set GAS, [Balduccini and Gelfond, 2003]). $M(\Delta)$ is a generalized answer set (GAS) of the abductive program $(P, A)$ iff $\Delta \subseteq A$ and $M(\Delta)$ is an answer set of $P \cup \Delta$.

Definition 2.3 (Abductive Inclusion Order, [Balduccini and Gelfond, 2003]). We can establish an ordering among generalized answer sets as follows: Let $M(\Delta_1)$ and $M(\Delta_2)$ be generalized answer sets of $(P, A)$, we define $M(\Delta_1) \leq_A M(\Delta_2)$ iff $\Delta_1 \subseteq \Delta_2$.

Example 2.1. Suppose $\{a, b\}$ are abducibles and $P = \{a \leftarrow b, b \leftarrow a, c \leftarrow a\}$

Then $\{a, b, c\} \{a\}$ (that is, the resulting answer set is $\{a, b, c\}$ and $\{a\}$ is the abducible) is a GAS, since $\{a, b, c\}$ is an answer set of $P \cup \{a\}$, as well as $\{a, b, c\} \{a, b\}$ and $\{\}$. Therefore, $\{a, b, c\} \{a\} \leq \{a, b, c\} \{a, b\}$, since $\{a\} \subseteq \{a, b\}$. However, $\{\}$ is the minimal GAS of $P$, as $\{\}$ is a subset of any set.

Definition 2.4 (Minimal Generalized Answer Set MGAS, [Balduccini and Gelfond, 2003]). $M(\Delta)$ is a minimal generalized answer set of $(P, A)$ iff $M(\Delta)$ is a generalized answer set of $(P, A)$ and it is minimal w.r.t. abductive inclusion order.

It is worth mentioning that minimal generalized answer sets are used to define the semantics of CR-Prolog. Consistency Restoring Rules is defined in [Balduccini and Gelfond, 2003] using this semantics.

3 Updates using Minimal Generalized Answer Sets

In the last few years several proposals have been defined for update logic programs [Eiter et al., 2000; Osorio and Zacarías, 2003; 2004]. According to these semantics, knowledge is given by a sequence of logic programs (see [Eiter et al., 2000; Osorio and Zacarías, 2003; 2004]) where each program is considered an update of the previous one. All of them are based on the notion of causal rejection of rules, which enforces that, in case of conflicts between rules, more recent rules override older ones.

In particular [Eiter et al., 2000] is a proposal that presents a complete analysis with respect to properties that an update operator should have, with the aim to define a safe and reliable evolution of beliefs for agents, and ours follows this approach.

At this point we have presented alternative solutions to the examples in [Osorio and Zacarías, 2003; 2004], as well as a semantics using a new mechanism of minimal generalized answer sets MGAS for updates that consist of the following definitions: We only consider update pairs (instead of sequences). Formally, by a update pair, we understand a pair $(P_1, P_2)$ logic programs. We say that $P$ is an update pair of $A$ iff $A$ represents the set of atoms currying in $P_1 \cup P_2$.

Definition 3.1 (Update). Given an update pair $P = (P_1, P_2)$ over a set of atoms $A$, we define the update program $P_0 = P_1 \circ P_2$ over $A^*$ (extending $A$ by new abducible atoms) consisting of the following items:

(i) all constraints in $P_1$

(ii) for each $r \in P_1$ we add an abducible $b$ (a new atom) and the rule $r \leftarrow \neg b$

(iii) all rules $r \in P_2$

Note that we do not need nested rules (although they are perfectly defined as formulae in $N_2$). We may simple replace each rule $\text{head} \leftarrow \text{body} \in P_1$ by the rule $\text{head} \leftarrow \text{body}, \neg b$. So our update program is an ordinary logic program.

Last, $P$ represents our update operator.

Definition 3.2. Let $P = (P_1, P_2)$ be an update pair over a set of atoms $A$. Then, $S \subseteq Lit_A$ is an update answer set of $P$ if only if $S = S' \cap Lit_A$ for some answer set $S'$ of $P_0$.

Next, we present an example taken from [Eiter et al., 2000] illustrating that our mechanism sometimes coincides with their proposal.

Example 3.1. Let us illustrates a daily update regarding energy flaw.

Let $P_1$ be:

\[
\begin{align*}
\text{sleep} & \leftarrow \neg \text{tv(on)} \\
\text{night} & \leftarrow \top \\
\text{watch(tv)} & \leftarrow \text{tv(on)} \\
\text{tv(on)} & \leftarrow \top
\end{align*}
\]

Let $P_2$ be:

\[
\begin{align*}
\neg \text{tv(on)} & \leftarrow \text{power(failure)} \\
\text{power(failure)} & \leftarrow \top
\end{align*}
\]

by applying the update definition given in [Eiter et al., 2000] to both programs, we get that the single answer set of $P = (P_1, P_2)$ is:

\[
S = \{ \text{power(failure)}, \neg \text{tv(on)}, \text{sleep}, \text{night} \}
\]

On the other hand, by codifying this example under our new semantics we have that $P_1$ is transformed as follows: for each rule of $P_1$, we add an atom of $A$ (abducible). Moreover, we add to each rule the classic negation of such an abducible at the end of the rule. $P_2$ is not transformed, it remains the same. Therefore, the updated program is:

Abducibles: $\{y_1, y_2, y_3, y_4\}$.

Rules:

\[
\begin{align*}
\text{sleep} & \leftarrow \neg \text{tv(on)}, \neg y_1 \\
\text{night} & \leftarrow \neg y_2 \\
\text{tv(on)} & \leftarrow \neg y_3 \\
\text{watch(tv)} & \leftarrow \text{tv(on)}, \neg y_4 \\
\neg \text{tv(on)} & \leftarrow \text{power(failure)} \\
\text{power(failure)} & \leftarrow \top
\end{align*}
\]
the only answer set of this program coincides with the one in [Eiter et al., 2000].

Note that strictly following Definition 3.1, rule \( \text{sleep} \leftarrow \neg \text{tv(on)} \) is translated into

\[
(sleep \leftarrow \neg \text{tv(on)}) \leftarrow \neg b
\]

and this rule is equivalent in \( N_2 \) to \( \text{sleep} \leftarrow \neg \text{tv(on)}, \neg b \).

### 3.1 Disjunctive programs

Unlike DLP, another advantage of our approach is that it may be applied to disjunctive logic programs.

**Example 3.2.** Let us model a situation in which men and women apply for a grant in Mexico, where a CONACYT Grant is also known as National Grant.

Let \( P_1 \) be:

\[
\begin{align*}
\text{man} \lor \text{woman} & \leftarrow T \\
\text{good(grades)} & \leftarrow T \\
\text{conacyt(grant)} & \leftarrow \text{woman}, \text{good(grades)} \\
\sim\text{conacyt(grant)} & \leftarrow T
\end{align*}
\]

Now suppose that \( P_1 \) is updated with \( P_2 \):

\[
\begin{align*}
\text{conacyt(grant)} & \leftarrow \text{national(grant)} \\
\text{national(grant)} & \leftarrow \text{conacyt(grant)}
\end{align*}
\]

As we can see, \( P_1 \) has only one answer set:

\[
\{ \text{good(grades)}, \text{man}, \sim\text{conacyt(grant)} \}
\]

However, just as Alferes points out, programs with new information introduce new models (for nearly all existing updating semantics).

In this example, we update \( P_1 \) with new information represented by \( P_2 \). Therefore, \( P_1 \) has only one answer set. According to Alferes, several semantics for updates based on the causal rejection principle, \( P_1 \) updated with \( P_2 \) adds a second answer set, namely,

\[
\{ \text{woman}, \text{national(grant)}, \text{conacyt(grant)}, \text{good(grades)} \}
\]

The causal rejection principle states that all models of \( P_1 \) updated with \( P_2 \) must also be models of \( P_1 \) updated with \{\}. Then, with our semantics we get the update of \( P_1 \) with \( P_2 \) as follows:

**Abducibles** = \{\( y_1, y_2, y_3, y_4 \)\}

**Rules:**

\[
\begin{align*}
\text{man} \land \text{woman} & \leftarrow \neg y_1 \\
\text{good(grades)} & \leftarrow \neg y_2 \\
\text{conacyt(grant)} & \leftarrow \text{woman}, \text{good(grades)}, \neg y_3 \\
\sim\text{conacyt(grant)} & \leftarrow \neg y_4 \\
\text{conacyt(grant)} & \leftarrow \text{national(grant)} \\
\text{national(grant)} & \leftarrow \text{conacyt(grant)}
\end{align*}
\]

Then the only answer set of this program is

\[
\{ \text{good(grades)}, \text{man}, \sim\text{conacyt(grant)} \}
\]

which coincides with our intuition.

### 4 Main results

In this section, we present our main results of our update semantics. We begin by presenting a set of basic structural properties that the update operator in [Eiter et al., 2000] satisfies and that our proposal does too. Note that \( P_1 \equiv P_2 \) means that \( P_1 \) and \( P_2 \) have the same answer sets.

1. **Initialisation:** \( \emptyset \odot P \equiv P \)

   This is presented in [Eiter et al., 2000] as follows: the update of an initial empty knowledge base yields just the update itself.

2. **Idempotence:** \( P \odot P \equiv P \)

   This property means that the update of program \( P \) by itself has no effect.

3. **Weak Noninterference:** If \( P_1 \) and \( P_2 \) are programs defined over disjoints alphabets, and either both of them have answer sets or do not, then \( P_1 \odot P_2 \equiv P_1 \odot P_1 \).

   This property implies that the order of updates that do not interfere with each other, normally does not matter.

4. **Augmented update:** If \( P_1 \subseteq P_2 \) then \( P_1 \odot P_2 \equiv P_2 \).

   Updating with additional rules makes the previous update obsolete.

5. **Strong Consistency:** Suppose \( P_1 \cup P_2 \) has at least an answer set. Then \( P_1 \cup P_2 \equiv P_1 \cup P_2 \).

   The update coincides with the union when \( P_1 \cup P_2 \) has at least an answer set.

6. **Weak Irrelevance of Syntax:** Let \( P_1, P_2 \), and \( P_3 \) be logic programs under the same language \( L \), if \( P_1 \equiv_{N_2} P_2 \) then \( P \odot P_1 \equiv_{N_2} P \odot P_2 \).

   It says that if we update a program \( P \) by \( P_1 \) (or \( P_2 \)), the result should depend on the logical contents of \( P_1 \) (or \( P_2 \)), not on the particular syntax used to write \( P_1 \) (or \( P_2 \)).

   It is very important to point out that **Strong Consistency** corresponds to the second property mentioned in [Katsuno and Mendelzon, 1991].

Now let us introduce our main theorem that satisfies all properties previously mentioned.

**Theorem 4.1.** Our update operator \( \odot \) satisfies the six properties previously mentioned.

**Proof of Theorem 4.1.** (Sketch)

**Initialisation:** \( \emptyset \odot P \equiv P \) by construction. Hence \( \emptyset \odot P \equiv P \).

**Strong Consistency:** Let \( M \) be an answer set of \( P_1 \cup P_2 \) (it exists by hypothesis). Then \( M_A \) is a generalized answer of \( P_1 \cup P_2 \). Hence, the minimal generalized answer sets of \( P_1 \cup P_2 \) must be of the form \( M' \), for some \( M' \). Those are exactly the answer sets of \( P_1 \cup P_2 \).

**Idempotence:** If \( P \) does not have answer sets, then nor does \( P \odot P \). If \( P \) has answer sets, then \( P \odot P \) does, and hence, by Strong Consistency, \( P \odot P \equiv P \odot P \).

**Weak Noninterference:** If both \( P_1 \) and \( P_2 \) have exactly the same consistent completions, the set \( A \) is
a subset of the abducibles. Thus, $P \circ P_1$ and $P \circ P_2$ have exactly the same generalized answer sets. Therefore, $P \circ P_1$ and $P \circ P_2$ have exactly the minimal generalized answer sets. Hence, $P_1 \cup P_2 \equiv P_1 \circ P_2$.

(Augmented Update): If $P_2$ does not have answer sets, nor does $P_1 \circ P_2$. If $P_2$ has at least one answer set, then, since $P_1 \subseteq P_2$ and by Strong Consistency $P_1 \cup P_2 \equiv P_1 \circ P_2$. In each case $P_2 \equiv P_1 \circ P_2$.

(Weak Noninterference): If both $P_1$ and $P_2$ lack of answer sets then the update (in any order) lacks of answer sets. If $P_1$ and $P_2$ have answer sets, then $P_1 \cup P_2$ does too — because they are defined over disjoint alphabets. By Strong Consistency, $P_1 \cup P_2 \equiv P_1 \circ P_2$. Also $P_2 \cup P_1 \equiv P_2 \circ P_1$. Hence, $P_1 \circ P_2 \equiv P_2 \circ P_1$.

4.1 Comparing our approach with upd

In this section, we show the importance of building an approach on basic structural properties; how our approach handles the example mentioned in the introduction (and is therefore compatible with Strong Consistency); as well as a comparison with the one presented in [Eiter et al., 2000]. But let us illustrate through an example from [Alferes and Pereira, 2002] the main differences.

Example 4.1. Consider Example 1.1 again but, $P_2$ with a slight change like it is shown at once.

Let $P_2$ be:

$$see(stars) \leftarrow see(stars)$$

now suppose $P_1$ is updated with $P_2$. Then, by applying the update definition in [Eiter et al., 2000], the answer sets of this update are: $\{see(stars), night\}$ and $\{day, \sim see(stars)\}$. As we can see, this update adds a second answer set. According to Alferes, this new model arises since the update $P_2$ causally rejects the rule of $P_1$ which stated that it was not possible to see the stars, and it is present in nearly all proposals for updating semantics of logic programs. However, we present a solution based on our previous configuration for updates.

By applying our proposal to $P_1$ and $P_2$ we get the following program:

Abducibles: $\{z_1, z_2, z_3, z_4, z_5\}$

Rules:

- $day \leftarrow \neg night, \neg z_1$
- $night \leftarrow \neg day, \neg z_2$
- $see(stars) \leftarrow night, \neg cloudy, \neg z_3$
- $\sim see(stars) \leftarrow \neg z_4$
- $see(stars) \leftarrow see(stars)$

By applying our update mechanism defined previously (Def-1) we realise that the only answer set is:

$\{day, \sim see(stars)\}$

As we can see, this result coincides with our intuition just as noted in [Alferes and Pereira, 2002].

Example 4.2. Consider again previous Example 4.1 but, now with a slight modification as follows:

Let $P_1$ be:

$$day \leftarrow \neg night, \neg z_1$$
$$see(stars) \leftarrow night, \neg cloudy$$
$$night \leftarrow \neg day$$
$$\sim see(stars) \leftarrow \top$$
$$sky(hear) \leftarrow \top$$

the only answer set of this program is

$\{day, \sim see(stars)\}$

Now, suppose that $P_1$ is updated with $P_2$:

$$see(stars) \leftarrow see(constellations)$$
$$\sim see(constellations) \leftarrow \top$$

$P_2$ represents information containing Strong Consistency (Tautologies, in Alferes’ words). Now considering the proposal presented in [Eiter et al., 2000] this update adds new models, which contradicts our intuition as mentioned by Alferes et al., and with whom we coincide. However, using our proposal this example is codified as follows:

Abducibles: $\{z_1, z_2, z_3, z_4, z_5\}$

Rules:

- $day \leftarrow \neg night, \neg z_1$
- $night \leftarrow \neg day, \neg z_2$
- $see(stars) \leftarrow night, \neg cloudy, \neg z_3$
- $\sim see(stars) \leftarrow \neg z_4$
- $sky(hear) \leftarrow \neg z_5$

- $see(stars) \leftarrow see(constellations)$
- $see(constellations) \leftarrow see(stars)$
- $\sim see(constellations) \leftarrow \top$

And the only answer set of this update is $\{day, \sim see(constellations), \sim see(stars)\}$, which agrees with Alferes et al. In contrast, [Eiter et al., 2000] does not solve this problem.

5 Logic Programming with Ordered Disjunctions

Ordered Logic programming is defined by [Brewka, 2002] as follows: a simple ordered disjunction program is a set of rules of the form:

$$C_1 \times \cdots \times C_n \leftarrow A_1, \ldots, A_m, \neg B_1, \ldots, \neg B_k$$

where $C_1, A_j$ and $B_l$ are all ground literals. $C_1, \ldots, C_n$ are usually named the choices of a rule and their intuitive reading is as follows: The ordered disjunctions are used in the rule heads to select some of the answer sets of a program as the preferred ones. If $C_1$ is possible, then $C_1$; if $C_1$ is not possible, then $C_2$; if none of $C_1, \ldots, C_{n-1}$ is possible then

\[^1\text{Note that } z's \text{ and } y's \text{ are out by definition.}\]
Moreover, we can identify some special cases such as: if \( n = 0 \) we call the rule a constraint; and finally, we call facts to those rules where \( m = k = 0 \). In our proposal we will consider ordered disjunctive programs where \( n = 2, m = k = 0 \) to denote the subset of logic programs ordered disjunctions. For space limitations we do not present it in more detail, but the reader is invited to consult [Brewka, 2002; Brewka et al., 2004].

It is very important to notice that \textbf{Psmodels} [Brewka et al., 2004] is an efficient tool used in our examples: a modification of \textbf{smodels} that can be used to compute preferred stable models of normal logic programs under the ordered disjunction semantics. It tells us how many times the test program is invoked to check whether a given stable model is a preferred one. All examples presented here have been run and tested using this program: they are is correct and coincide with the discussion in [Alferes and Pereira, 2002; Eiter et al., 2000; Osorio and Zacarías, 2004].

\textbf{Example 5.1.} Consider Example 1.1 again and recall that this example was codified as follows:

\textbf{Abducibles:} \( \{ y_1, y_2, y_3, y_4 \} \)

\textbf{Rules:}

\[
\begin{align*}
\text{sleep} & \leftarrow \neg \text{tv(on)}, \neg y_1 \\
\text{night} & \leftarrow \neg y_2 \\
\text{tv(on)} & \leftarrow \neg y_3 \\
\sim \text{tv(on)} & \leftarrow \text{power(failure)} \\
\text{power(failure)} & \leftarrow \top
\end{align*}
\]

Following [Osorio et al., 2004b] the minimal generalized answer sets of every abductive program \( \langle P, A \rangle \) correspond to the intended models of some ordered disjunctive program \( P' \) that can be easily computed from \( \langle P, A \rangle \). The translation in this particular example corresponds to the following ordered disjunctive program

\[
\begin{align*}
z_1 & \times y_1 \\
z_2 & \times y_2 \\
z_3 & \times y_3 \\
z_4 & \times y_4 \\
\text{day} & \leftarrow \neg \text{night}, \neg y_1 \\
\text{night} & \leftarrow \neg \text{day}, \neg y_2 \\
\text{see(stars)} & \leftarrow \text{night}, \neg \text{cloudy}, \neg y_3 \\
\sim \text{see(stars)} & \leftarrow \neg y_4 \\
\text{see(stars)} & \leftarrow \text{see(constellations)} \\
\text{see(constellations)} & \leftarrow \text{see(stars)}
\end{align*}
\]

Here, \( z_1, z_2, z_3 \) and \( z_4 \) are new atoms. Recall that this transformation is correct and described in detail in [Osorio et al., 2004b].

Executing this program in \textbf{Psmodels} under ordered disjunctions semantics, its only answer set is: \( \{ \text{day}, \sim \text{see(stars)} \} \), which coincides with our intuition.

\section{6 Conclusions}

We have introduced a new semantics for updates that is more general than the one presented in [Eiter \textit{et al.}, 2000]. In addition, we have emphasised the importance of an approach based on key \textit{structural properties}. Our semantics can be used for any type of program. Moreover, it satisfies the WIS property introduced in [Osorio and Zacarías, 2004]. We also illustrate through a couple of examples how our proposal solves several problems occurring in recent update-semantics. Alferes et al. noticed these problems as well and addressed them differently. In contrast, we introduced a new property called \textbf{Strong Consistency} and show its usefulness.

We would like to point out that stronger properties can be obtained by replacing \( \equiv \) with the stronger notions of \textit{uniform} or even \textit{strong} equivalence introduced by Pearce.

Finally, we would like to illustrate how to do multiple updates as a future work with the following example:

\textbf{Example 6.1. Three updates.}

Let \( P_1 \) be:

\[
\begin{align*}
a & \leftarrow \top \\
\sim a & \leftarrow \top \\
b & \leftarrow \top
\end{align*}
\]

Then the updated program would be:

\[
\begin{align*}
z \times r_1 \times r_2 \\
a & \leftarrow \neg r_1 \\
\sim a & \leftarrow \neg r_2 \\
b & \leftarrow \top
\end{align*}
\]

The updated answer set is \( \{ b, \sim a \} \).

Suppose now that \( P_3 \) were:

\[
\begin{align*}
a & \leftarrow \top
\end{align*}
\]

Then the updated answer set is \( \{ a \} \).

\section{References}


