# Well founded semantics for logic program updates

F. Banti[1], J. J. Alferes[1], and A. Brogi[2]

[1] CENTRIA, Universidade Nova de Lisboa, Portugal
[2] Dipartimento di Informatica, Università di Pisa, Italy

**Abstract.** Over the last years various semantics have been proposed for dealing with updates of logic programs by (other) logic programs. Most of these various semantics extend the stable models semantics of normal, extended (with explicit negation) or generalized (with default negation in rule heads) logic programs. In this paper we propose a well founded semantics for logic programs updates. We motivate our proposal with both practical and theoretical argumentations. Various theoretical results presented here show how our proposal is related to the stable model approach and how it extends the well founded semantics of normal and generalized logic programs.

## 1 Introduction

When dealing with knowledge bases modelling knowledge that may change over time, an important issue is that of how to automatically incorporate new (updated) knowledge without falling into an inconsistency each time this new knowledge is in conflict with the previous one. When these knowledge bases are represented by logic programs (LPs), this issue boils down to that of how to deal with logic programs updates. In this context, updates are represented by sequences of sets of logic programming rules, also called *dynamic logic programs* (DLPs), the first set representing our initial knowledge, while later ones represent new incoming information. In the last years, several semantics had been proposed for logic programs updates [1, 2, 5, 9, 14–16, 19, 20]. Most of these semantics are extensions of the stable models semantics of extended (with explicit negation) [13] or generalized (allowing default negation in rule heads) [17] logic programs. This is a natural choice given the appropriateness of stable models for knowledge representation, and the simplicity of the definition of stable model semantics for normal logic programs, which allows various extensions in a natural way. However, it is our stance that there are application domains for logic programs updates with requirements demanding a different choice of basic semantics, such as the well founded semantics [11]. One of such requirements is that of computational complexity: in applications that require the capability of dealing with an overwhelming mass of information, it is very important to be able to quickly process such information, even at the cost of losing some inference power. In this respect, as it is well known, the computation of stable models is NP-hard, whereas that of the well founded model is polynomial. Another requirement not fulfilled by stable model semantics is that of being able to answer queries about a given part of the knowledge without the need to, in general, consult the whole knowledge base. The well founded semantics complies with the property of relevance [8], making it possible to implement query driven proof procedures that, for any given query, only need to explore a part of the knowledge base. Moreover, in domains

with a great amount of highly distributed and heterogeneous knowledge, inconsistencies are bound to appear not only when new knowledge conflicts with old knowledge, but also within the new (or old) knowledge alone. To deal with contradictions that appear simultaneously, the mechanisms of updates are of no use, and some form of paraconsistent semantics [7] is required, i.e. a semantics where these contradictions are at least detected, and isolated. The issue of removing these contradictions (which is left out of this paper) can then be viewed as orthogonal, and forms of revision may, or not, be applied afterwards depending also on the requirements of efficiency.

The well founded semantics (of single programs) is successfully used in several domains where these requirements are important, such as deductive and relational databases [18], implementation of agents and multi-agents [21]. A well founded based semantics for logic programs updates seems to be the answer for domains where the above requirements are added with the need to update knowledge in dynamic domains. However, as we mentioned above, most of the existing semantics are stable models based. A few attempts to define a well founded semantics for DLPs can be found [2, 3, 14]. Unfortunately, as discussed in Section 5.1, none of these is, in our opinion, satisfactory, be it because they lack a declarative definition of the semantics, or because they are too skeptical.

In this paper we define the (paraconsistent) well founded semantics of dynamic logic programs. This semantics is a generalization for sequences of programs of the (paraconsistent) well founded semantics of normal [11] and generalized programs [6]. Moreover it is sound wrt to the stable models semantics for DLPs as defined in [1]. As for most of the existing semantics for logic programs updates, the approach herein is also based on the causal rejection principle [9, 15], which states, informally: an old rule is rejected if there exists a more recent one which is supported and whose conclusions are in conflict with the ones of the older rule. We extend this principle from a 2-valued to a 3-valued setting, and apply it to the well founded semantics.

The rest of the paper is organized as follows. Section 2 recalls some preliminary notions and establishes notation. Section 3 presents the extension of the causal rejection principle to the 3-valued case. In section 4 the well founded semantics for DLPs is defined, and in section 5 some of its properties are studied and relations with existing proposals (briefly) established. We end, in section 6, with some concluding remarks. Lack of space prevents us from providing the proofs here.

## 2 Background: Language, concepts and notation

In this section we briefly recall the syntax of DLPs, a language introduced in [2] for dealing with logic programs updates, and their semantics as defined in [1]. Our choice on this semantics for introducing the background is based on the fact that, among the extant ones, is the more credulous and that it properly overcomes some problems of the extant, as shown in [1].

To represent negative information in logic programs and their updates, DLP uses generalized logic programs (GLPs) [17], which allow for default negation *not A* not only in the premises of rules but also in their heads. A GLP defined over a propositional language $\mathcal{L}$ is a (possibly infinite) set of ground rules[1] of the form $L_0 \leftarrow L_1, \ldots, L_n$,

[1] As usual, a programs with variables stands for the possibly infinite set of rules resulting from replacing, in every possible way, the variables by elements of the Herbrand universe.

where each $L_i$ is a literal in $\mathcal{L}$, i.e., either a propositional atom $A$ in $\mathcal{L}$ or the default negation *not* $A$ of a propositional atom $A$ in $\mathcal{L}$. We say that $A$ is the *default complement* of *not* $A$ and viceversa. Given a rule $\tau$ as above, by $head(\tau)$ we mean $L_0$ and by $body(\tau)$ we mean $\{L_1, \ldots, L_n\}$. Since we will be interested in defining a paraconsistent and 3-valued (well founded) semantics for DLPs, our definition of interpretation will be quite general. In the sequel an *interpretation* is simply a set of literals of $\mathcal{L}$. A literal $L$ is *true* (resp. *false*) in $I$ iff $L \in I$ (resp. *not* $L \in I$) and *undefined* in $I$ iff $\{L, not\ L\} \cap I = \{\}$. A conjunction (or set) of literals $C$ is true (resp. false) in $I$ iff $C \subseteq I$ (resp. there exists $L \in C$ such that $L$ is false in $I$).We say that $I$ is *consistent* iff for each atom $A \in \mathcal{L}$ *at most* one of $A$ and *not* $A$ belongs to $I$, otherwise we say $I$ is *paraconsistent*. We say that $I$ is *2-valued* iff for each atom $A \in \mathcal{L}$ *exactly* one of $A$ and *not* $A$ belongs to $I$.

A *dynamic logic program* over a language $\mathcal{L}$ is a finite sequence $P_1 \oplus \ldots \oplus P_n$ (also denoted $\oplus P_i$, where the $P_i$s are GLPs indexed by $1, \ldots, n$), where all the $P_i$s are defined over $\mathcal{L}$. Intuitively such a sequence may be viewed as the result of, starting with program $P_1$, updating it with program $P_2$, ..., and updating it with program $P_n$. For this reason we call the singles $P_i s$ *updates*. We use $\rho(\mathcal{P})$ to denote the multiset of all rules appearing in the programs $P_1, ..., P_s$.

The *refined stable model semantics* for DLPs is defined in [1] by assigning to each such sequence a set of stable models (that is proven there to coincide with the stable models based semantics defined in [17] when the sequence is formed by a single GLP). The basic idea of the semantics is that, if a later rule $\tau$ has a true body (according to a given interpretation), then former rules in conflict with $\tau$ should be *rejected* (causal rejection principles). Moreover, any atom $A$ for which there is no rule with true body (given an interpretation) in any of the programs in the sequence, is considered false by default. The semantics is then defined in term of a fixpoint equation that, given an interpretation $I$, tests whether $I$ has exactly the consequences obtained after removing from the multiset $\rho(\mathcal{P})$ all the rules rejected given $I$, and imposing all the default assumptions given $I$. Formally, let:

$$Default(\oplus P_i, I) = \{not\ A \mid \nexists\ A \leftarrow body \in \rho(\mathcal{P}) \wedge body \subseteq I\}$$
$$Rej^S(\oplus P_i, I) = \{\tau \mid \tau \in P_i | \ \exists\ \eta \in P_j\ i \leq j,\ \tau \bowtie \eta \wedge\ body(\eta) \subseteq I\}$$

where $\tau \bowtie \eta$ means that $\tau$ and $\eta$ are conflicting rules, i.e. the head of $\tau$ is the default complement of the head of $\eta$.

**Definition 1.** *Let $\oplus P_i$ be a DLP over language $\mathcal{L}$ and $M$ a two valued interpretation. $M$ is a refined stable model of $\oplus P_i$ iff $M$ is a fixpoint of $\Gamma^S_{\oplus P_i}$:*

$$\Gamma^S_{\oplus P_i}(M) = least\left(\rho(\mathcal{P}) \setminus Rej^S(M) \cup Default(M)\right)$$

*where $least(P)$ denotes the least Herbrand model of the definite program obtained by considering each negative literal not $A$ in $P$ as a new atom*[2].

The definition of dynamic stable models of DLPs [2] is as the one above, but where the $i \leq$ in the rejection operator is replaced by $i < j$. I.e., if we denote this other rejection operator by $Rej(DLP, I)$, and define $\Gamma_{\oplus P_i}(M)$ by replacing in $\Gamma^S$ $Rej$ by $Rej^S$, then the stable models of $\oplus P_i$ are the interpretations $I$ such that $I = \Gamma_{\oplus P_i}(M)$.

[2] Whenever clear from the context, hereafter we omit the $\oplus P_i$ in any of the above defined operators.

Comparisons among these two definitions, as well as further details, properties and motivation for the definition of this language and semantics are beyond the scope of this paper, and can be found in [1, 2].

## 3 The notion of causal rejection for 3-valued semantics.

According to the above mentioned causal rejection principle [9, 15], a rule from an older program in a sequence is kept (by inertia) unless it is rejected by a more recent conflicting rule whose body is true. On the basis of this, not only stable models have to be defined (as reviewed above), but also the very basic notion of model has to be modified when dealing with updates. In the static case, a model of a program is an interpretation that satisfies all the rules of the program, where a rule is satisfied if its head is true or its body is false. If we want to adapt this idea to the updates setting taking in consideration the casual rejection principle we should only require non rejected rules to be satisfied. Also the concept of supported model [4] has to be revisited when dealing with updates. In the static case, a model $M$ of $P$ is supported iff for every atom $A \in M$, there is a rule in $P$ whose head is $A$ and whose body is satisfied in $M$. If we extend the concept of supportedness to logic programs with updates, it would be unnatural to allow rejected rules to support a the truth of a literal.

The causal rejection principle is defined for 2-valued semantics, and we want now to extend it to a 3-valued setting, in which literals can be $undefined$, besides being $true$ or $false$. In the 2-valued setting, we have seen, a rule is rejected iff there is a rule in a later update whose body is true in the considered interpretation. In this context, this is the same as saying that the body of the rejecting rule is not false. In a 3-valued setting this is no longer the case, and the following question arises: should we reject rules on the basis of rejecting rules whose body is *true*, or on the basis of rules whose body is *not false*? We argue that the correct answer is the latter. In the remainder we give both practical and theoretical reasons for our choice, but we want now to give an intuitive justification. Suppose initially we believe a given literal $L$ is true. Later on we get the information that $L$ is false if some conditions hold, but that those conditions are (for now) undefined. As usual in updates, we prefer later information to the previous one. On the basis of such information, can we be *sure* that $L$ remains true? It seems to us we cannot. The more recent source of information says that if some conditions hold then $L$ is not true, and such conditions *may* hold. We should then reject the previous information and consider, on the basis of the most recent one, that $L$ is undefined.

On the basis of these intuitions, and the corresponding consequent notion of causal rejection, we extend the definition of update model and update supported model to the 3-valued setting.

**Definition 2.** *Let $\oplus P_i$ be any DLP, and $M$ a 3-valued interpretation. $M$ is an* update 3-valued model *of $\oplus P_i$ iff for each rule $\tau$ in any given $P_i$, $M$ satisfies $\tau$ (i.e. $head(\tau) \in M$ or $body(\tau) \not\subseteq M$) or there exists a rule $\eta$ in $P_j$, $i < j$ such that $\tau \bowtie \eta$ and $body(\eta)$ is not false in $M$. We say $M$ is a* supported 3-valued update model *of $\oplus P_i$ iff it is an update model and*

1. *for each atom $A \in M$, $\exists\ \tau \in P_i$ with head $A$ such that $body(\tau) \subseteq M$ and $\nexists\ \eta \in P_j$, $i < j$ such that $\tau \bowtie \eta$, and $body(\eta)$ is not false in $M$.*

2. *for each negative literal not A, if not* $A \in M$*, then for each rule* $A \leftarrow body \in \rho(\mathcal{P})$ *such that body is true in* $M$*, there exists a rule* $\eta$*, in a later update whose head is not A, and such that* $body(\eta)$ *is true in* $M$*.*

We illustrate, via an example, the intuitive meaning of the defined concepts.

*Example 1.* Sara, Cristina and Bob, are deciding what they will do on Saturday. Sara decides she is going to a museum, Cristina wants to go shopping and Bob decides to go fishing in case Sara goes to the museum. Later on they update their plans: Cristina realizes she has no money and hence she decides not to go shopping, Sara decides she will not go to the museum in case it snows and Bob decides he will also go fishing if it is a sunny day. Moreover we know from the forecast that Saturday can be either a sunny day or a raining day. We represent the situation with the DLP $P_1 \oplus P2$, where:

$$\begin{array}{lll} P_1 : museum(s). & P_2 : fish(b) \leftarrow sunny. & sunny \leftarrow not\, rain. \\ \quad shopping(c). & \quad not\, shopping(c). & rain \leftarrow not\, sunny. \\ \quad fish(b) \leftarrow museum(s). & \quad not\, museum(s) \leftarrow snow. & \end{array}$$

The intended meaning of $P_1 \oplus P_2$ is that it will not snow, but we do not know if it will rain or not, Sara will go to the museum, Bob will go fishing and, finally, Cristina will not go shopping. In fact, every 3-valued update model of $P_1 \oplus P_2$ contains $\{museum(s), not\, shopping(c), fish(b)\}$. Suppose now Bob decides that, in the end, he does not want to go fishing if it rains, i.e our knowledge is updated with:

$$P_3 : not\, fish(b) \leftarrow rain$$

The intuitive meaning is that, after $P_3$, we do not know whether Bob will go fishing since we do not know whether Saturday is a rainy day. And, according to definition 2, there is a supported 3-valued update model of $P_1 \oplus P_2 \oplus P_3$ in which $shopping(c)$ is false, $museum(s)$ is true and $fish(b)$ is undefined.

It can be checked that, according to all existing stable models based semantics for updates of [1, 2, 5, 9, 15, 16], $P_1 \oplus P_2 \oplus P_3$ has two stable models: one where $rain$ is true and $fish(b)$ is false, and another where $rain$ is false and $fish(b)$ is true. A notable property of the well founded semantics for (static) programs is that the well founded model is always a subset of all stable models. If one wants to preserve this property in DLPs, in the well founded model of this example one should neither conclude $fish(b)$ nor $not fish(b)$. If a rule would only be rejected in case there is a later one with *true* body (rather than *not false* as we advocate), since the body of the rule $not\, fish(b) \leftarrow rain$ is not true, we would not be able to reject the initial rule $fish(b) \leftarrow museum(s)$, and hence would conclude $fish(b)$. This shows that, to preserve this relation to stable models based semantics, the well founded model semantics for DLPs must relay on this notion of 3-valued rejection described above.

## 4 The Well founded semantics for DLPs

Update models and supported models are just the starting point of our investigation. In this section, based on the notion of causal rejection just presented, we define the

Well Founded Semantics for DLPs. Formally, our definition is made in a way similar to the the definition of the well founded semantics for normal in [10], where the well founded model is characterized by the least alternating fixpoint of the Gelfond-Lifschitz operator $\Gamma$ (i.e. by the least fixpoint of $\Gamma^2$). Unfortunately, if we apply literally this idea, i.e. define the well founded model as the least alternating fixpoint of the operator used for the dynamic stable (or refined stable) models of DLPs, the resulting semantics turns out to be too skeptical:

*Example 2.* It is either day or night (but not both). Moreover, if the stars are visible it is possible to make astronomical observations. This knowledge is updated with the information that: if it is night the stars are visible; the observatory is closed if it is not possible to make observations; and the starts are not visible:

| | | |
|---|---|---|
| $P_1 : observe \leftarrow see\_stars.$ | $day \leftarrow not\, night.$ | $night \leftarrow not\, day.$ |
| $P_2 : see\_stars \leftarrow night.$ | $not\, see\_stars.$ | $closed(obs) \leftarrow not\, observe.$ |

The intended meaning of $P_1 \oplus P_2$ is that currently the stars are not visible, it is not possible to make astronomical observations and, hence, the observatory is closed. However, it is easy to check, the least alternating fixpoint of $\Gamma_{P_1 \oplus P_2}$ is $\{not\, see\_stars\}$, in which one is not able to conclude that the observatory is closed. This is, in our opinion, not satisfactory: since we conclude that we cannot see the stars, we should also conclude that we cannot make astronomical observations and that the observatory is closed. Notably, the least alternating fixpoint of $\Gamma^S$ yield even more skeptical results. In fact, this is a general result which is an immediate consequence of Lemma 1 below.

In order to overcome this problem, instead of defining the well founded model in terms of the least fixpoint of the double application of one of the operators for stable models of DLPs, we define it as the least fixpoint of the composition of two different (anti-monotonous) operators. Such operators have to deal with the causal rejection principle described above, in which a rule is to be rejected in case there is a later conflicting one whose body is *not false*. In the well founded semantics of normal logic programs, if there exists a rule $A \leftarrow body$ (where $A$ is an atom), such that $body$ is not false in the well founded model, then $A$ is not false as well. Consider now the same situation in an update setting with a rule $L \leftarrow body_1$, where $body_1$ is not false. In this situation we should conclude that $L$ is not false *unless* there exists a rule $not\, L \leftarrow body_2$, where $body_2$ is true in the same or in a later program in the sequence. In fact, note that the rule for $not\, L$ is not rejected by the one for $L$. Since the body of the former is true, according to the causal rejection principle $not\, L$ should be true (i.e. $L$ should be false) unless the rule is rejected by some later rule. In any case, $L \leftarrow body_1$ is no longer playing any role in determining the truth value of $L$. For this reason we allow rules to reject other rules in previous *or in the same* update while determining the set of non-false literals of the well founded model and, accordingly, we use the $\Gamma^S_{\oplus P_i}$ operator, as the first operator of our composition.

For determining the set of true literals according to the causal rejection principle, only the rules that are not rejected by conflicting rules in *later* updates should be put in place. For this reason we use the operator $\Gamma_{\oplus P_i}$ as the second operator, the well founded model being thus defined as the least fix point of the $\Gamma\Gamma^S$.

**Definition 3.** *The well founded model $WFDy(\oplus P_i)$ of a DLP $\oplus P_i$ is the (set inclusion) least fixpoint of $\Gamma_{\oplus P_i}\Gamma^S_{\oplus P_i}$.*

Since both $\Gamma$ and $\Gamma^S$ are antimonotonous (cf. [1, 15]), $\Gamma\Gamma^S$ is monotonous, and so it always has a least fixpoint. In other words, $WFDy$is uniquely defined for every DLP. Moreover $WFDy(\oplus P_i)$ can be obtained by (transfinitely) iterating $\Gamma\Gamma^S$, starting from the empty interpretation.

In alternating fixpoint definitions for the well founded semantics of normal programs, true atoms in the well founded model are those that belong to the least fixpoint of the double application of the Gelfond-Lifschitz operator [12] (that is defined over sets of *atoms*), the false ones being those that do not belong to the application of the Gelfond-Lifschitz operator to the set of true atoms. This is not the case in definition 3, where both true and false literals are obtained at once. This is possible because both operator in the composition apply directly to sets of literals. However, given the proposition below, it is easy to see that an approach similar to that of normal programs could have been used instead.

**Proposition 1.** *Let $I$ be any fixpoint of $\Gamma\Gamma^S$. Then, for each negative literal not $A$, not $A \in I \Leftrightarrow A \notin \Gamma^S(I)$.*

The reader can check that, for the DLP $P_1 \oplus P_2$ of example 2, the well founded model is $\{not\ see\_stars,\ not\ observe, closed(obs)\}$. So, in this example, $WFDy$ yield the desired less skeptical conclusions. In fact, in general $WFDy$ is less skeptical than any semantics resulting from any other combination of $\Gamma^S$ and $\Gamma$.

**Lemma 1.** *Let $\oplus P_i$ be a DLP, and let $X$, $Y$ be two interpretations such that $X \subseteq Y$. Then $\Gamma^S(Y) \subseteq \Gamma(X)$.*

From this Lemma it follows that the least fixpoint of any other combination is a prefixpoint of $\Gamma\Gamma^S$ and, as such, a subset of the least fixpoint of $\Gamma\Gamma^S$.

*Example 3.* Let $P_1$, $P_2$ and $P_3$ be the programs of example 1. The reader can check that $WFDy(P_1 \oplus P_2 \oplus P_3)$ is, as desired, $\{not\ snow,\ museum(s), not\ shopping(c)\}$.

Note in this example that the single program $P_1 \cup P_2 \cup P_3$ is consistent, but $WFDy(P_1 \oplus P_2 \oplus P_3)$ is nonetheless different from the well founded model of the single program $P_1 \cup P_2 \cup P_3$. In fact, the well founded model of that single program contains $fish(b)$. This, as argued in example 1, is not in accordance with the intended meaning of that sequence of updates. Moreover, this (desired) behavior is not particular to the well founded semantics. In fact, the intersection of all stable models of $P_1 \cup P_2 \cup P_3$ contains $fish(b)$, but $fish(b)$ is not in the intersection of all stable models of $P_1 \oplus P_2 \oplus P_3$, in all of the stable model based semantics for DLPs [1, 2, 5, 9, 15, 16].

As shown in example 1, $WFDy(P_1 \oplus P_2 \oplus P_3)$ is a supported 3-valued update model. This result holds in general, whenever the well founded model does not contain any pair of complementary literals.

**Theorem 1.** *Let $\oplus P_i$ be a DLP and $W$ its well founded model. Then, if $W$ contains no pair of complementary literals, $W$ is a supported 3-valued update model of $\oplus P_i$.*

The proviso of $W$ not containing any pair of complementary literals is due to the fact that, since we are using a notion of interpretation that allows contradictory sets of literals, in principle nothing prevents the well founded model of a DLP from being contradictory. We say that a DLP $\oplus P_i$ is consistent (or non contradictory) iff $WFDy(\oplus P_i)$ is consistent i.e. it does not contain any pair of complementary literals. Note that for contradictory DLPs the very notion of model is not applicable.

## 5 Properties

Our motivation for defining a new semantics for logic programs updates, as described in the Introduction, is based on a number of requirements. Though lack of space prevents us from presenting here a detailed study on properties, this paper would not be complete without at least stating in what term those requirements are indeed met by $WFDy$, and briefly comparing with existing approaches.

One of the important requirements is that of having a semantics computable in polynomial time. It is not difficult to check that the computation of both $\Gamma_{\oplus P_i}(I)$ and $\Gamma^S_{\oplus P_i}(I)$ is polynomial in the size of $DLP$, and so:

**Proposition 2.** *The well founded model of any finite ground dynamic logic program $\oplus P_i$ is computable in polynomial time on the number of rules in $\oplus P_i$.*

Another required property is that of relevance [8], so as to guarantee the possibility of defining query driven proof procedures. Informally, in normal (single) programs, a semantics complies with relevance if the truth value of any atom $A$ in a program only depends on the rules relevant for this literal (i.e. those rules with head $A$, or with a head $A'$ such that $A'$ belongs to the body of (another) relevant rule). In order to establish results regarding relevance of $WFDy$ we have first to define what is the relevant part of a DLP (rather than a single program) wrt a literal (rather than atom).

**Definition 4.** *Let $\oplus P_i$ be any DLP in the language $\mathcal{L}$ and $L$, $B$, $C$ literals in $\mathcal{L}$. We say $L$ directly depends on $B$ iff $B$ occurs in the body of some rule in $\oplus P_i$ with head $L$ or not $L$. We say $L$ depends on $B$ iff $L$ directly depends on $B$ or there is some $C$ such that $L$ directly depends on $C$ and $C$ depends on $B$. We call $Rel_L(\oplus P_i)$ the dynamic logic programs $P_1^{(L)} \oplus \ldots \oplus P_n^{(L)}$ such that $P_i^{(L)}$ is the set of all rules of $P_i$ with head is $L$ or not $L$ or some $B$ on which $L$ depends on.*

This definition simply applies the above intuition of relevance for normal program, but now considering sequences of programs, and by stating that rules for *not* $A$ are relevant for $A$ (and vice-versa). And $WFDy$ complies with relevance exactly in these terms:

**Theorem 2.** *. Let $\oplus P_i$ be a DLP in the language $\mathcal{L}$ and $A$ any atom of $\mathcal{L}$. Then $WFDy(\oplus P_i) \cap \{A,\ not\, A\} = WFDy(Rel_A \oplus P_i) \cap \{A,\ not\, A\}$.*

As noted above, $WFDy$ can be contradictory. In these cases, inconsistent conclusions for a given atom may follow, but without necessarily having a contradiction in all atoms. However, as desired in updates, these contradictions in atoms may only appear in case there are two conflicting simultaneous rules (i.e. in a same program of the sequence) which are both supported, and none of them is rejected by some later update:

**Theorem 3.** *The well founded model $W$ of a sequence $\oplus P_i$ is noncontradictory iff for all $\tau, \eta \in P_i$ such that: $\tau \bowtie \eta$, $W \models body(\tau)$, $W \models body(\eta))$ there exists $\gamma \in P_j$, $i < j$ such that $\gamma \bowtie \tau$ or $\gamma \bowtie \eta$ and $\Gamma^S(W) \models body(\gamma)$.*

Finally, it was our goal to find a proper generalization of the well founded semantics single programs into logic programs updates. It is thus important, to guarantee that $WFDy$ coincides with the well founded semantics of GLPs [6] when the considered DLP is a single program $P$, and with the well founded semantics of normal programs [11] when, furthermore, that single program has no negation in rule heads. Denoting by $WFG(P)$ the well founded semantics of the generalized logic program $P$ according to [6]:

**Theorem 4.** *Let $P$ be a generalized program. Then $WFG(P) = WFDy(P)$.*

Since $WFG(P)$ coincides with the well founded semantics of [11] when $P$ is a normal program (cf. [6]), it follows that $WFDy$ coincides with the semantics of [11] whenever the sequences is made of a single program without default negation in rule heads.

### 5.1 Brief Comparisons

Among the various semantics defined for sequences of logic programs [1, 2, 5, 9, 15, 16, 20], $WFDy$ shares a close relationship with the refined stable models semantics of [1], resembling that between stable and well founded semantics of normal programs.

**Proposition 3.** *Let $M$ be any refined stable model of $\oplus P_i$. The well founded model $WFDy(\oplus P_i)$ is a subset of $M$. Moreover, if $WFDy(\oplus P_i)$ is a 2-valued interpretation, it coincides with the unique refined stable model of $\oplus P_i$.*

This property does not hold if, instead of the refined semantics, we consider any of the other semantics based on causal rejection [2, 5, 9, 15, 16]. This is so because these semantics are sometimes overly skeptical, in the sense that admit to many models, and thus have a smaller set of conclusions (in the intersection of all stable models). One particular case when this happens is when a sequence is updated with tautologies. Though, intuitively, updating our knowledge with a tautology should have no influence on the results, this is not the case in any of those semantics. For example, with all the cited semantics, updating the program $P_1 \oplus P_2$ of example 2 (which, according to all of them, has a single stable model containing $not\ see\_starts$) with the program $P_3$ consisting of the tautological rule $see\_stars \leftarrow see\_stars$ leads to two stable models: one in which $see\_stars$ is true and the other in which $see\_stars$ is false. Thus, this intuitively harmless update prevents $not\ see\_stars$ from being concluded. This is not the case with [1], nor with $WFDy$, which are both immune to tautologies. In this example, both conclude that $not\ see\_stars$ is true, before and after the update. For further details on this topic see [1].

Notably, all the attempts found in the literature [2, 3, 14] to define a well founded semantics for logic programs updates are overly skeptical as well. According to all the cited semantics, the well founded model of $P_1 \oplus P_2 \oplus P_3$ is the empty set, hence, unlike $WFDy$, they are not able to conclude that $not\ see\_stars$ is true. Though there it cannot be detailed here, other class of programs exist, besides the ones with tautologies, where the cited semantics bring more skeptical results then $WFDy$. Moreover the definition of all these semantics is based on a complex syntactical transformation of sequences into single programs, making it difficult to grasp what the declarative meaning of a sequence is.

## 6 Concluding remarks

Guided by the needs of applications, it was our purpose in this paper to define a semantics for DLPs fulfilling some specific requirements. Namely: a semantics whose computation has polynomial complexity; that is able to deal with contradictory programs, assigning them a non trivial meaning; that can be used to compute answers to queries without always visiting the whole knowledge base. With this in mind, we have defined a well founded semantics for DLPs.

The defined semantics is a generalization of the well founded semantics of normal and generalized logic programs, and it coincides with them when the DLP consists of a single program. It has polynomial complexity and obeys relevance. Regarding the requirement of being able to deal with contradictions, lack of space prevented us from further elaborating on the properties of the semantics in these cases, and on how it can in general be used to detect contradictory literals and literals that depend on contradictions. We have, nonetheless, provided a complete characterization of the non contradictory cases.

We briefly compared the proposed semantic to the existent ones for DLPs that are based on the causal rejection principle, and shown that it is a skeptical approximation of the refined stable model semantics for DLPs, and less skeptical than all other existing well founded based semantics for updates. Comparisons to semantics of updates that are not based on the causal rejection principle are outside the scope of this paper. For an analysis of these semantics, and comparisons to the above ones see e.g. [15].

The definition of $WFDy$, as well as the various definitions of the well founded semantics for normal logic programs, is based on ground programs only. Though not presented, an operational semantics that allows query evaluation also in the non ground case already exists. Future work involves the development of implementations of $WFDy$ based on such operational semantic, and its usage in applications.

## References


1. J. J. Alferes, F. Banti, A. Brogi, and J. A. Leite. Semantics for dynamic logic programming: a principled based approach. In V. Lifschitz and I. Niemelä, editors, *LPNMR-7*, volume 2923 of *LNAI*, pages 8–20. Springer, 2004.
2. J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska, and T. C. Przymusinski. Dynamic updates of non-monotonic knowledge bases. *The Journal of Logic Programming*, 45(1–3):43–70, September/October 2000.
3. J. J. Alferes, L. M. Pereira, H. Przymusinska, and T. Przymusinski. LUPS: A language for updating logic programs. *Artificial Intelligence*, 132(1 & 2), 2002.
4. K. R. Apt and R. N. Bol. Logic programming and negation: A survey. *The Journal of Logic Programming*, 19 & 20:9–72, May 1994.
5. F. Buccafurri, W. Faber, and N. Leone. Disjunctive logic programs with inheritance. In D. De Schreye, editor, *ICLP'99*, pages 79–93. MIT Press, 1999.
6. C. V. Damásio and L. M. Pereira. Default negation in the heads: why not? In R. Dyckhoff, H. Herre, and P. Schroeder-Heister, editors, *Int. Ws. Extensions of Logic Programming*, volume 1050 of *LNAI*. Springer, 1996.
7. C. V. Damásio and L. M. Pereira. A survey on paraconsistent semantics for extended logic programas. In D. M. Gabbay and Ph. Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 2, pages 241–320. Kluwer, 1998.
8. J. Dix. A classification theory of semantics of normal logic programs II: Weak properties. *Fundamenta Mathematicae*, 22(3):257–288, 1995.
9. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. On properties of semantics based on causal rejection. *Theory and Practice of Logic Programming*, 2:711–767, November 2002.
10. A. Van Gelder. The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences*, 1992.
11. A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.
12. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. A. Bowen, editors, *ICLP'88*, pages 1070–1080. MIT Press, 1988.

13. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In Warren and Szeredi, editors, *7th ICLP*, pages 579–597. MIT Press, 1990.
14. J. A. Leite. Logic program updates. Master's thesis, Dept. de Informática, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, November 1997.
15. J. A. Leite. *Evolving Knowledge Bases*, volume 81 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, December 2002.
16. J. A. Leite and L. M. Pereira. Iterated logic program updates. In J. Jaffar, editor, *JICSLP-98*, pages 265–278. MIT Press, 1998.
17. V. Lifschitz and T. Woo. Answer sets in general non-monotonic reasoning (preliminary report). In B. Nebel, C. Rich, and W. Swartout, editors, *KR-92*. Morgan-Kaufmann, 1992.
18. W. May, B. Ludasher, and G. Lausen. Well-founded semantics for deductive object-oriented database languages. In *Procs. of the 5th DOOD*, volume 1341 of *LNAI*, pages 320–336. Springer, 1997.
19. C. Sakama and K. Inoue. Updating extended logic programs through abduction. In M. Gelfond, N. Leone, and G. Pfeifer, editors, *LPNMR-99*, volume 1730 of *LNAI*, pages 147–161. Springer, 1999.
20. Y. Zhang and N. Y. Foo. Updating logic programs. In Henri Prade, editor, *ECAI-98*, pages 403–407. John Wiley & Sons, 1998.
21. Y. Zou, T. Finin, L. Ding, H. Chen, and R. Pan. Using semantic web technology in multi-agent systems: a case study in the taga trading agent environment. In *Proceedings of the 5th international conference on Electronic commerce*, pages 95 – 101. ACM Press, 2003.