Computing and Informatics, Vol. 26, 2007, 225–238

UPDATES OF LOGIC PROGRAMS

Ján Šefránek

Department of Applied Informatics
Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava, Slovakia
e-mail: sefranek@fmph.uniba.sk

Revised manuscript received 8 December 2006

Abstract. Dynamic aspects of knowledge representation has been tackled recently by a variety of approaches in the logic programming style. We consider the approaches characterized by the causal rejection principle (if there is a conflict between rules, then more preferred rules override those less preferred). A classification and a comparison of the approaches is presented in the paper. We compare them also to our own approach based on Kripke structures.

Keywords: Multidimensional logic programming, causal rejection principle, Kripke structure

1 INTRODUCTION

Dynamic aspects of knowledge and reasoning did not attract a sufficient attention in logic-based artificial intelligence research for a long time. On the other side, an evolution of incomplete knowledge and reasoning with such knowledge represents a challenging chunk of foundational problems important also for practical applications.

A family of approaches to theoretical investigation of dynamic aspects of knowledge representation is presented in this paper. Our attention is focused on evolving knowledge bases. Recently the problem of evolving knowledge bases has been tackled by a variety of approaches in the logic programming style; see [8, 4] and others. A detailed and comprehensive information is available in [6]. The presented model consists of a set of logic programs (each of them represents a module of the knowledge base) and of a preference relation on modules. The conflicts between the modules

are resolved according to the preference relation: if there is a conflict between rules, then more preferred rules override those less preferred.

We discuss some examples with the only goal in the mind: to help the reader understand our exposition. For a more ambitious application of dynamic logic programming we refer to the project of [17]. The computational complexity of that approach equals the computational complexity of answer set programming.

The goals of this paper are

- an introduction of a point of view useful for a classification of the approaches based on the causal rejection principle,
- a comparison of those approaches to our approach [11, 12].

The paper is structured as follows: Technical prerequisites are presented in Section 2. Two basic types of rules rejection (those of [2] and [4]) are described in Section 3. Then, in Section 4 two strategies of accepting the default assumptions (when updating) are described. We combined them with the strategies of rule rejection and obtained four types of semantics of logic program updates. The relations between these four types are summarized. Finally, our approach, based on a Kripkean semantics, is compared to the approaches based on the causal rejection principle.

2 PRELIMINARIES

Consider a set of propositional symbols (atoms) $\mathcal{A}$. A *literal* is an atom (a positive literal) or an atom preceded by the default negation (a negative literal), *not* A. The set $\{notA : A \in \mathcal{A}\}$ will be denoted by $\mathcal{D}$ (defaults, assumptions). For each atom A, A and *not* A are called *conflicting literals*. A set of literals is called *consistent*, if it does not contain a pair of conflicting literals. A convention as follows is used: if literal L is of the form *not* A, where $A \in \mathcal{A}$, then *not* $L = A$.

A *rule* is a formula r of the form $L \leftarrow L_1, \ldots, L_k$, where $k \geq 0$, and L, L_i are literals (for each i). We will denote L also by $head(r)$ and the set of literals $\{L_1, \ldots, L_k\}$ by $body(r)$. If $body(r) = \emptyset$, then r is called a *fact*. The subset of all positive (negative) literals of $body(r)$ is denoted by $body^+(r)$ ($body^-(r)$). For each rule r we denote the rule $head(r) \leftarrow body^+(r)$ by r^+. We say that two rules, r and r', are *conflicting rules* if $head(r) = not\ head(r')$, notation: $r \bowtie r'$.

The set of all rules (over $\mathcal{A}$) forms the language $\mathcal{L}$. A finite subset of $\mathcal{L}$ is called a *generalized logic program* (program hereafter).[1]

A *partial interpretation* of the language $\mathcal{L}$ is a consistent set of literals. The set of all partial interpretations of the language $\mathcal{L}$ is denoted by $Int_{\mathcal{L}}$. A *total interpretation* is a partial interpretation I such that for each $A \in \mathcal{A}$ either $A \in I$ or *not* $A \in I$. Sometimes it will be convenient to speak about interpretations and sets of facts interchangeably. For this reason, we introduce the notation as follows: Let M be an interpretation, then $rule(M) = \{L \leftarrow : L \in M\}$.

[1] In this paper only the language of generalized logic programs is used. We incorporate some ideas of [4] into this framework.

We accept a convention as follows: All programs use only propositional symbols from $\mathcal{A}$. Interpretations of all programs are subsets of $Int_{\mathcal{L}}$.

A literal L is *satisfied* in a partial interpretation I if $L \in I$. A set of literals S is satisfied in a partial interpretation I if each literal $L \in S$ is satisfied in I. A rule r is satisfied in a partial interpretation I if $head(r)$ is satisfied in I whenever $body(r)$ is satisfied in I. Notation: $I \models L$, $I \models S$, $I \models r$.

A *total* interpretation I is a *model* of a program P if each rule $r \in P$ is satisfied in I.

Notice that (propositional generalized logic) programs may be treated as Horn theories: each literal *not* A may be considered as a new propositional symbol. The least model of a Horn theory H is denoted by $least(H)$.

Definition 1 (Stable model, [2])**.** Let P be a program and S be a total interpretation. Let $S^- = \{not\ A \in S : A \in \mathcal{A}\}$.

Then S is a *stable model* of P iff $S = least(P \cup rule(S^-))$.

A program P is *consistent* iff there is a stable model of P, otherwise it is *inconsistent*.

Multidimensional dynamic logic program is defined as a set of generalized logic programs together with a preference relation on the programs [8]. A specification of the relation can be based on the edges of a graph.

Definition 2 ([8])**.** A *multidimensional dynamic logic program* (also *multiprogram* hereafter) is a pair $(\mathcal{P}, G)$, where $G = (V, E)$ is an acyclic digraph, $|V| \geq 2$, and $\mathcal{P} = \{P_v : v \in V\}$ is a set of generalized logic programs.

We denote by $v_i \prec v_j$ that there is a directed path from v_i to v_j and $v_i \preceq v_j$ means that $v_i \prec v_j$ or $i = j$. If $v_i \prec v_j$, we say that P_{v_j} is *more preferred* than P_{v_i}.

We denote the set of programs $\{P_{v_i} : v_i \preceq s\}$ by $\mathcal{P}_s$.

If G is a directed path, the multidimensionality is collapsed and we speak simply about dynamic logic programs. The elementary case is represented by $V = \{u, v\}$ and $E = \{(u, v)\}$.

3 STRATEGIES OF THE RULES REJECTION

We account for two strategies how to formalize the idea of causal rejection. They lead to the sets $reject(\mathcal{P}, s, M)$ and $rjct(\mathcal{P}, s, M)$, see below. The former has been defined for example in [2, 8], the latter for example in [4].

Definition 3. Let $(\mathcal{P}, G)$ be a multidimensional dynamic logic program, where $G = (V, E)$ and $\mathcal{P} = \{\mathcal{P}_v : v \in V\}$. Let M be an interpretation, $s \in V$.

$$\begin{aligned} reject(\mathcal{P}, s, M) &= \{r \in P_i : \exists j \in V\ \exists r' \in P_j\ (i \prec j \preceq s\ \wedge\ r \bowtie r' \\ &\qquad \wedge\ M \models \{body(r), body(r'))\}\}, \\ rjct^{\mathcal{P}}(s, M) &= \emptyset \end{aligned}$$

$$\begin{aligned} rjct^{\mathcal{P}}(i,M) &= \{r \in P_i : \exists j \in V\ \exists r' \in P_j \setminus rjct^{\mathcal{P}}(j,M) \\ &\quad (i \prec j \preceq s \wedge r \bowtie r' \wedge M \models \{body(r), body(r')\}\} \\ rjct(\mathcal{P},s,M) &= \bigcup_{i \preceq s} rjct^{\mathcal{P}}(i,M) \end{aligned}$$

A multiprogram is denoted by $\mathcal{P}$ in what follows. A graph $G = (V, E)$ is implicitly assumed.

Theorem 4 ([7, 5])**.** Let $\mathcal{P}$ be a multiprogram, $s \in V$, M be an interpretation. It holds that $rjct(\mathcal{P}, s, M) \subseteq reject(\mathcal{P}, s, M)$.

The inclusion $reject(\mathcal{P}, s, M) \subseteq rjct(\mathcal{P}, s, M)$ does not hold for some $\mathcal{P}$, s, M:

Example 5 ([10])**.** Let $\mathcal{P} = P_1, P_2, P_3$ be a dynamic logic program, $1 \prec 2 \prec 3$.

$$\begin{aligned} P_1 &= \{a \leftarrow b, b \leftarrow\} \\ P_2 &= \{not\ b \leftarrow a\} \\ P_3 &= \{b \leftarrow a\} \end{aligned}$$

Let w be $\{a, b\}$. Then

$$\begin{aligned} reject(\mathcal{P},3,w) &= \{b \leftarrow, not\ b \leftarrow a\} \\ rjct(\mathcal{P},3,w) &= \{not\ b \leftarrow a\} \end{aligned}$$

The rejection done by *rjct* is a minimal one, in a reasonable sense. If $\mathcal{P}$ is a multiprogram, M an interpretation, $s \in V$ and for each i, where $i \preceq s$ holds that P_i is consistent, then there is no pair of conflicting literals $(L_1.L_2)$ such that $\{L_1, L_2\} \subseteq rjct(\mathcal{P}, s, M)$.

Remark 6. Recently, a new strategy of rules rejection has been presented in [1]: $reject^*(\mathcal{P}, s, M) =$

$$\{r \in P_i : \exists j \in V\ \exists r' \in P_j\ (i \preceq j \preceq s\ \wedge\ r \bowtie r' \wedge\ M \models \{body(r), body(r')\}\}.$$

Thus, also conflicts between rules of the same program are solved. The operator *rjct* can be modified in the same style. This modification satisfies the refined extension principle introduced in [1] and it solves problems with cyclic updates for dynamic logic programs.

However, there are serious problems with its extension to multidimensional dynamic logic programs, see [16]. A solution of the cyclic updates problems for multidimensional dynamic logic programs has been presented recently in [3]. The solution is based on a notion of level mapping and on the well supported semantics.

Anyway, an update should not provide a solution of conflicts within one program (see also [9]); therefore, we do not devote attention to corresponding strategy of rules rejection.

4 STRATEGIES OF ASSUMPTIONS ACCEPTING

The meaning of a program depends both on rules and on default assumptions (see Definition 1). Therefore, the approach to default assumptions accepting is an essential one for updates of logic programs. Again, two strategies are discussed. Observe that two strategies of rules rejection may be combined with two strategies of the acceptance of default assumptions.

The first strategy is as follows: Consider a multi-program $\mathcal{P}$. The updated program consists of all rules of $\mathcal{P}$ except those rejected (according to the selected strategy). If S is a stable model of an updated program, then S^- is the corresponding set of default assumptions. The concepts of justified updates [6] and backward-justified updates use this strategy.

Definition 7. Let $\mathcal{P}$ be a multiprogram, $s \in V$.

An interpretation M is a *justified update* of $\mathcal{P}$ at state $s \in V$ iff M is a stable model of the program $\mathcal{P}_s \setminus reject(\mathcal{P}, s, M)$. An interpretation M is a *backward-justified update* of $\mathcal{P}$ at state $s \in V$ iff M is a stable model of the program $\mathcal{P}_s \setminus rjct(\mathcal{P}, s, M)$.

Theorem 8 ([6])**.** Let $\mathcal{P}$ be a program and $s \in V$. If S is a justified update of $\mathcal{P}$ at state s, then S is its backward-justified update at s.

Theorem 8 does not hold in the converse direction:

Example 9. Consider the multiprogram $\mathcal{P}$ from the Example 5. The interpretation $\{a, b\}$ is a backward-justified update of $\mathcal{P}$ at state 3, but it is not its justified update at 3.

(Backward-)justified updates suffer from some unpleasant properties:

Example 10 ([6])**.** Let $\mathcal{P} = \langle P_1, P_2 \rangle$, where $1 \prec 2$,

$$\begin{aligned} P_1 &= \{a \leftarrow\} \\ P_2 &= \{not\ a \leftarrow not\ a\} \end{aligned}$$

Both $M_1 = \{a\}$ and $M_2 = \{not\ a\}$ are justified (and backward-justified) updates of $\mathcal{P}$ at state 2.

There is no reason to accept the interpretation M_2 (more precisely, to accept the default assumption *not* a in M_2) and, consequently, to reject the fact $a \leftarrow$. Troubles are caused also by (more general) cyclic updates.

The (implicit) policy of accepting default assumptions which is behind the (backward) justified updates is not an adequate one. Certainly, a more subtle policy is required. Such a policy is proposed within the next strategy:

Definition 11 (Dynamic stable model at state s, [8])**.** Let $\mathcal{P} = (\mathcal{P}_D, D)$ be a multidimensional dynamic logic program, where $D = (V, E)$ and $\mathcal{P}_D = \{\mathcal{P}_v : v \in V\}$. Let M be an interpretation, A be an atom, $s \in V$. Then

$$\mathit{default}(\mathcal{P}, s, M) \quad = \quad \{\mathit{not}\ A : \neg\exists r \in \mathcal{P}_s : (\mathit{head}(r) = A\ \wedge\ M \models \mathit{body}(r))\}$$

An interpretation M is a *dynamic stable model* of $\mathcal{P}$ at state $s \in V$, iff

$$M = \mathit{least}((\mathcal{P}_s \setminus \mathit{reject}(\mathcal{P}, s, M)) \cup \mathit{default}(\mathcal{P}, s, M))$$

and M is a *backward-dynamic stable model* of $\mathcal{P}$ at state $s \in V$, iff

$$M = \mathit{least}((\mathcal{P}_s \setminus \mathit{rjct}(\mathcal{P}, s, M)) \cup \mathit{default}(\mathcal{P}, s, M)).$$

Remark 12. If the condition $M \models \{\mathit{body}(r), \mathit{body}(r')\}$ (from the definition of *reject* or *rjct*) is simplified to $M \models \mathit{body}(r')$ we get an equivalent notion of dynamic stable model: rule r whose body is not satisfied in M does not affect the least model (its head does not belong to the least model).

Theorem 13 ([4, 6])**.** If S is a dynamic stable model of $\mathcal{P}$ at state s, then it is its justified update at s.

The converse implication doesn't hold:

Example 14 ([6])**.** Consider the program from Example 10: $M_2 = \{\mathit{not}\ a\}$ is not a dynamic stable model of $\mathcal{P}$: $\mathit{default}(\mathcal{P}, 2, M_2)) = \emptyset$, but $M_2^- = M_2$.

Theorem 15 ([7, 5])**.** If S is a backward-dynamic stable model of $\mathcal{P}$ at state s, then it is its backward-justified update.

Also Theorem 15 does not hold in the converse direction, see Example 10. (Notice that for the sequences of two programs *reject* coincide with *rjct*, therefore their backward-justified updates coincide with justified updates, and their backward-dynamic stable models coincide with dynamic stable models.)

Theorem 16 ([7, 5])**.** Let $\mathcal{P}$ be a multiprogram, $s \in V$. If S is a dynamic stable model of $\mathcal{P}$ at state s then it is its backward-dynamic stable model at s.

Theorem 16 does not hold in the converse direction, see again the examples 5 and 9: $w = \{a, b\}$ is not a dynamic stable model of $\mathcal{P}$ at state 3, but it is its backward-dynamic stable model.

Summary: For each given multi-program $\mathcal{P}$ holds: dynamic stable models of $\mathcal{P}$ (at each state s) create a proper subset of justified updates of $\mathcal{P}$ and of backward-dynamic stable models of $\mathcal{P}$. And both last mentioned interpretations create a proper subset of backward-justified updates.

Remark 17. A detailed comparison of semantics based on causal rejection of rules is presented in [5]. The semantics studied in [5] coincide on a restricted class of programs (sufficiently acyclic programs and acyclic programs).

5 KRIPKEAN SEMANTICS

The problems with tautological and cyclic updates are not removed completely by the introduction of dynamic stable model semantics:

Example 18 ([6])**.** Let $\mathcal{P}$ be $\langle P_1, P_2\rangle$, where $1 \prec 2$ and

$$P_1 = \{not\ a \leftarrow \qquad\qquad P_2 = \{a \leftarrow a\}$$
$$a \leftarrow\}$$

$S = \{a\}$ is the (backward-)dynamic stable model of $\mathcal{P}$ at state 2.

Similarly, if $\mathcal{P}'$ is $\langle P_1, P_2'\rangle$, where $P_2' = \{a \leftarrow b; b \leftarrow a\}$, then $S = \{a, b\}$ is the only dynamic stable model of $\mathcal{P}'$ at state 2.

A recent semantics, called refined dynamic stable model semantics [1], solves the problem of tautological (cyclic) updates which can resolve inconsistencies in a program by a simple straightforward method – conflicting rules in the program are rejected mutually. However, the solution holds only for sequences of programs. The problem is not resolved for the multidimensional case, see [16].

Moreover, the semantics of (refined) dynamic stable models suffers from other fundamental problems. It enables irrelevant updates[2], see Example 19. On the other hand, a semantics based on the causal rejection principle is not able to recognize such conflicts between programs that are not manifested by conflicts between rules, see Example 20.[3]

Example 19 ([4])**.** Let $\mathcal{P}$ be $\langle P_1, P_2\rangle$, with $1 \prec 2$, where

$$P_1 = \{a \leftarrow b \qquad\qquad P_2 = \{not\ b \leftarrow not\ a\}$$
$$b \leftarrow\}$$

Dynamic stable models of $\mathcal{P}$ at state 2 are $S_1 = \{a, b\}$ and $S_2 = \{not\ a, not\ b\}$. If the information from P_1 and P_2 is given, there is no sufficient reason to believe in S_2 and to reject the fact $b \leftarrow$.

Moreover and most importantly, there are some conflicts between the programs that are principally not solvable on the level of the conflicts between rules:

Example 20. Let P_2 be a more preferred program than P_1.

$$P_1 = \{a \leftarrow b, c\} \qquad\qquad P_2 = \{b \leftarrow not\ a$$
$$c \leftarrow b\}.$$

There is no conflict between the rules of both programs. No rule can be rejected and the meaning of $P_1 \cup P_2$ cannot be updated according to the dynamic logic

[2] For a discussion of irrelevant updates see [14].

[3] See also [15].

programming [2] paradigm and according to a semantics based on rejection of rules. However, there is a sort of conflict between the programs (between their meanings): union of two consistent programs is inconsistent. It is not natural to solve only conflicts (inconsistencies) caused by conflicts of rules and ignore the other sources of inconsistency.[4]

In our example, the literal a in the less preferred program P_1 depends on a set of assumptions (on the set of literals $w = \{b, c\}$). On the other hand, every literal in w is dependent on a default assumption (on the literal *not* a) in the more preferred program P_2. Notice that there is a circular dependency of a on *not* a in $P_1 \cup P_2$. The problem of circular dependency can be resolved by rejecting the less preferred dependency of a on w.

Hence, our goal is to define rejection of dependencies. We need a more rich semantic structure, in order to be able to do it.

Our approach provides a semantic treatment of dependencies between sets of literals (belief sets). The dependencies are encoded in (rather nonstandard) Kripke structures.[5] The intuition is as follows: if the world (or our knowledge of the world) is represented by an interpretation w then (the meaning of) a program P may be viewed as a set of transitions to other worlds, compatible, in a sense, with w. The transitions are specified as follows: if $body(r)$ is satisfied in w for some $r \in P$ then the world $w \cup \{head(r)\}$ is compatible with w, if it is consistent. It is a natural choice to require that the compatibility relation is transitive and irreflexive.

We present a simplified (but a sufficient one for our current needs) version of the definition of a Kripke structure associated with a program. For the more complicated version see [12].

Definition 21 (Kripke structure associated with a program)**.** Let P be a program. A Kripke structure $\mathcal{K}^P$ associated with P is a a pair (W, ρ), where:

- $W = Int_{\mathcal{L}} \cup \{w_\perp\}$, W is called the set of possible worlds, $w_\perp$ is the representative of the set of all inconsistent sets of literals,
- ρ is a binary relation on $W \times W$, it is called the accessibility relation and it contains the set of all pairs (w, w'), where $w \neq w'$, satisfying exactly one of the conditions:
  1. $w' = w \cup \{head(r)\}$ for some $r \in P$ such that $w \models body(r)$,
  2. $w' = w_\perp$ iff $\exists r \in P\ (w \models body(r) \wedge not\ head(r) \in w)$.

Definition 22. If $e = (u, v) \in \rho$, it is said that e is a *ρ-edge* and u (v) is called the *source* (the *target*) of e. A *ρ-path* is a sequence σ of ρ-edges $\langle e_1, e_2, \ldots, e_n \rangle$, where $n \geq 0$ and the source of e_{i+1} is the target of e_i, in a Kripke structure $\mathcal{K}$.

[4] If the original knowledge base is consistent and its update is also consistent, then the updated knowledge base should be also consistent according to the third postulate for updates, as it has been expressed by Katsuno and Mendelzon [9]. For a discussion of that postulate in the frame of dynamic logic programming see [15].

[5] A dependency framework has been introduced recently in [15].

We say that this σ is *rooted* in w_0 (also w_0-rooted). If there is no ρ-edge (w_n, w) in $\mathcal{K}$, we say that σ is *terminated* in w_n, w_n is called a *terminal* of σ.

Paths of the form $\langle (w_0, w_1), (w_1, w_2) \ldots, (w_(n-1), w_n)\rangle$ are usually denoted by a shorthand of the form $\langle w_0, w_1, \ldots, w_n\rangle$.

Remark 23. We consider also empty paths rooted in a node.

We are now ready to state (in terms of nodes and paths in $\mathcal{K}^P$) the conditions of being a stable model of a program P.

Definition 24 (Distinguished paths, good worlds)**.** Let P be a program, σ be a ρ-path $\langle w_0, \ldots, w_n\rangle$, where $n \geq 0$, in $\mathcal{K}^P$. We say that σ is *correctly rooted*, if $w_0 \subseteq \mathcal{D}$.

A correctly rooted ρ-path σ terminated in a total interpretation w is called a *distinguished* path and w is called a *good world*.

Theorem 25 ([11])**.** Let P be a program, $\mathcal{K}^P$ be the Kripke structure associated with P.

Then w_n is a good world in $\mathcal{K}^P$ iff it is a stable model of P.

Remark 26. If $\mathcal{D}$ is a terminal in $\mathcal{K}^P$, it is the (only) stable model of P. The trivial sequence $\langle \mathcal{D}\rangle$ is correctly rooted and terminated in $\mathcal{D}$. Suppose that a total interpretation $w \neq \mathcal{D}$ is a stable model of P, we get $(\mathcal{D}, w_\perp) \in \rho$.

6 UPDATES OF KRIPKE STRUCTURES

While the semantics presented in Sections 3 and 4 are based on rules rejection, our semantics of updates is based on a rejection of some edges in Kripke structures. In other words, the updates change the compatibility relation between possible worlds.

A removal of some edges may be interpreted as overriding the corresponding dependencies between belief sets. On the other hand, connecting the edges from one Kripke structure to the edges from another may be interpreted as an enrichment of the dependencies between belief sets.

Suppose two programs,[6] P and U, and the Kripke structures, $\mathcal{K}^P = (W, \rho^P)$ and $\mathcal{K}^U = (W, \rho^U)$, associated with P and U, respectively. Moreover, U (the updating program) is more preferred than P (the original program).

Our approach enables to recognize conflicts between programs even if there are no conflicts between rules. In such cases some edges are rejected, but there is no reason to reject a rule. We regard this as an essential observation:

Example 27. Consider Example 20. We "translate" its idea into Kripke structures. The world $\{a, b, c\}$ (and, consequently, the literal a) is dependent on the world

[6] In this paper only the elementary case of two programs is considered (because of the limited size). The more general theory is presented in [12].

$w = \{b, c\}$ in the less preferred Kripke structure $\mathcal{K}^P$. Similarly, the world $w_\perp$ is dependent on the world $w' = \{not\ a, b, c\}$ in the less preferred structure and w' is dependent on the world $\{not\ a\}$ in the more preferred Kripke structure $\mathcal{K}^U$. We do not want to accept the circular dependency of a on $not\ a$, therefore we propose to reject the (less preferred) edge $(\{not\ a, b, c\}, w_\perp) \in \rho^P$ (leading to inconsistency).

We intend to define an operation $\oplus$ on Kripke structures. The resulting Kripke structure $\mathcal{K}^{U\oplus P} = \mathcal{K}^U \oplus \mathcal{K}^P = (W, \rho^{U\oplus P})$ should be based on $\mathcal{K}^U$ while a reasonable part of $\mathcal{K}^P$ is preserved. Notice that the set of nodes, W, remains unchanged, but some ρ^P-edges should be rejected.

Definition 28 (Attacked edges)**.** Let $W = Int_\mathcal{L}$. Let $\tau_1, \tau_2 \subseteq W \times W$ be binary relations. Let L be a literal.

We say that $e = (u, u \cup \{L\}) \in \tau_1$ is *attacked* by $e' = (u, u \cup \{not\ L\}) \in \tau_2$.

Of course, there is a symmetry: if e is attacked by e' then e' is attacked by e, too. Nevertheless, we want to prefer "one side". We intend to reject an edge from ρ^P if it is attacked by an edge in ρ^U.

Sometimes it is needed to reject a ρ^P-edge of the form $(w, w_\perp)$, see the Example 20. In this case the analysis is a little bit more complicated.[7]

The basic rule is as follow: if $(w, w_\perp) \in \rho^P$ and w occurs on a ρ_2-path, then $(w, w_\perp)$ should be rejected, if the rejection is not blocked. The idea of blocking is illustrated in the next Example.

Example 29 (Blocking of rejections)**.** Let $\mathcal{P}$ be $\langle P, U\rangle$.

$$P = \{a \leftarrow \qquad\qquad U = \{not\ a \leftarrow not\ b$$
$$b \leftarrow \qquad\qquad not\ b \leftarrow not\ c$$
$$c \leftarrow\} \qquad\qquad not\ c \leftarrow not\ a\}$$

Let be $w_1 = \{not\ a, not\ b, not\ c\}$. There are three ρ^U-paths to w_1. One of them is $\sigma = \langle\{not\ a\}, \{not\ a, not\ c\}, w_1\rangle$. Notice that $e_1 = (\{w_1, w_\perp) \in \rho^P$.

Observe that the ρ^U-paths mentioned above are rooted in default assumptions (in$\{not\ a\}$ or in $\{not\ b\}$ or in $\{not\ c\}$). There is no support for these assumptions. We decided to prefer the facts from P to default assumptions from U. Hence, the facts from P should override default assumptions from U. Therefore, the rejection of e_1 by σ should be considered to be blocked.

The facts from P are conflicting with respect to each root of a ρ^U-path to w_1, w_1 is not supported in U. On the contrary, $(\{not\ a\}, w_\perp), (\{not\ b\}, w_\perp), (\{not\ c\}, w_\perp) \in \rho^P$ and there is no reason to reject them.

The idea of rejections blocking is the fundamental one.

[7] More motivations and examples are presented in [12].

Definition 30 (Blocking)**.** Let programs P and U be given. The corresponding Kripke structures are denoted by $\mathcal{K}_P$ and $\mathcal{K}_U$. Let $(w_1, w_\perp) \in \rho^P$. Let L_1 and L_2 be conflicting literals.

A ρ^U-edge (w_0, w_1) is *blocked* iff $L_1 \in w_1$ and there is a ρ^P-path from $\emptyset$ to w such that $L_2 \in w$.

Remark 31. The basic attitude behind the notion of blocking is as follows: default assumptions in a more preferred program may be falsified by facts from a less preferred program.

Notice that Definition 30 is sufficiently general: if a literal L_1 from w_0 is conflicting with a literal L_2 supported (by a path form $\emptyset$ in $\mathcal{K}_P$ (as in Example 29), then also $L_1 \in w_1$.

Definition 32 (Overriding)**.** A ρ^U-edge $e = (w_0, w_1)$ *overrides* a ρ^P-edge $(w_1, w_\perp)$ if e is not blocked.

Definition 33 (Rejected edges)**.** Consider $\mathcal{K}^P = (W, \rho^P)$ and $\mathcal{K}^U = (W, \rho^U)$. We say that $e \in \rho^P$ is rejected, if

- e is attacked by some $e' \in \rho^U$,
- or there is a ρ^U-edge $e' = (w_0, w_1)$, overriding $e = (w_1, w_\perp) \in \rho^P$.

The set of rejected edges is denoted by $\mathit{Rejected}_{\rho^U}(\rho^P)$.

Definition 34 (Update on Kripke structures)**.**

$$\begin{aligned} \mathcal{K}^U \oplus \mathcal{K}^P &= \mathcal{K}^{U \oplus P} = (W, \rho^{U \oplus P}) \\ \rho^{U \oplus P} &= \rho^U \cup (\rho^P \setminus (\mathit{Rejected}_{\rho^U}(\rho^P)) \end{aligned}$$

The causal rule rejection principle is satisfied in updated Kripke structures:

Proposition 35 ([10])**.** Let $\mathcal{P} = \langle P_1, P_2 \rangle$ be a multiprogram, where $1 \prec 2$. Let $w \in \mathit{Int}_{\mathcal{L}}$, $r \in P$ and $w \models \mathit{body}(r)$.

If $e = (w, w \cup \{\mathit{head}(r)\}) \in \mathit{Rejected}_{\rho^{P_2}}(\rho^{P_1})$ then $r \in \mathit{reject}(\mathcal{P}, 2, w)$ and $r \in \mathit{rjct}(\mathcal{P}, 2, w)$.

Remark 36 ([10])**.** The converse of Proposition 35 does not hold. If a rule is rejected, some edges "generated" by this rule may be not rejected. Let P_1 be $\{a \leftarrow\}$, P_2 be $\{\mathit{not}\ a \leftarrow b\}$ and w be $\{a, b\}$. Then $(\emptyset, \{a\}) \notin \mathit{Rejected}_{\rho^{P_2}}(\rho^{P_1})$ but the rule $a \leftarrow$ is both in $\mathit{reject}(\mathcal{P}, 2, w)$ and in $\mathit{rjct}(\mathcal{P}, 2, w)$.

Therefore, the rejection defined within the frame of Kripkean semantics is more sensitive (able to make a more fine distinguishing) than a rejection of rules.

Remark 37. There is no analogy of the Proposition 35 for the edges with the target $w_\perp$. It may happen that $(w, w_\perp) \in \mathit{Rejected}_{\rho^U}(\rho^P)$, but there is no rule $r \in P \cap \mathit{reject}(\mathcal{P}, 2, w)$ or in $P \cap \mathit{rjct}(\mathcal{P}, 2, w)$: According to Definition 32 there is a rule $r \in P$ such that $w \models \mathit{body}(r)$ and $\mathit{not}\ \mathit{head}(r) \in w$. However, $\mathit{not}\ \mathit{head}(r)$

may be not introduced by a rule from U, it may be in the root of each path to w, see Example 20.

Of course (similarly as in the Remark 36), if a rule is rejected, it is possible that $(w, w_\perp) \notin \mathit{Rejected}_{\rho^U}(\rho^P)$, where the transition from w to $w_\perp$ is generated by the rejected rule.

Problems with tautological and cyclic updates are removed in the Kripkean semantics. Tautologies do not influence Kripke structures.

Proposition 38. Let P be program. Let $P' = P \cup \{r\}$, where $\mathit{head}(r) \in \mathit{body}(r)$. Then $\mathcal{K}^P = \mathcal{K}^{P'}$.

The following theorem shows that cycles from P_2 do not cause an update of $\mathcal{K}_{P_1}$: paths generated by cycles are terminated in $\mathcal{K}^{P_1 \oplus P_2}$ by $w_\perp$ if there is a conflict between a rule in P_1 and a rule in the cycle of P_2:

Theorem 39. Let for all $r \in P_2$ there is a $r' \in P_2$ such that $\mathit{head}(r) \in \mathit{body}(r')$. Let $w = \{\mathit{head}(r) : r \in P_2\}$.

If there is a ρ^{P_1}-path $\langle \emptyset, \ldots, u \rangle$ such that *not* $\mathit{head}(r) \in u$ for some $r \in P_2$ then for each w' such that $w \subseteq w'$ holds that $(w', w_\perp) \in \mathcal{K}^{P_1 \oplus P_2}$.

Notice now that good worlds of $\mathcal{K}^{P_2 \oplus P_1}$ play the crucial role in the update semantics.

Finally, Kripkean semantics do not suffer from irrelevant updates of the type illustrated by Example 19 and it is able to recognize the conflicts not distinguishable by semantics based on the causal rejection principle.

Theorem 40. Let $P \cup U$ be consistent. Then the good worlds in $\mathcal{K}^{P \cup U}$ coincide with good worlds in $\mathcal{K}^{U \oplus P}$.

7 CONCLUSIONS

The results of the paper:

- a classification of the semantics based on the rule rejection principle is given,
- a Kripkean semantics of logic program updates is presented,
- the Kripkean semantics enables to solve the problem of tautological, cyclic and irrelevant updates and it is able to recognize conflicts indistinguishable according to the causal rejection principle.

A dependency framework is introduced recently in [15, 13, 14]. The framework provides a simplification of Kripke structures presented in this paper and also its computational properties are more appropriate.

The author thanks one of the anonymous referees for helpful observations and comments.

The work on this paper has been partially supported by a grant of the Slovak Agency VEGA, No. 1/3112/06 and a grant of the Agency for Promotion of Research and Development under the contract Šefránek, J. APVV-20-P04805.

REFERENCES

[1] Alferes, J. J.—Banti, F.—Brogi, A.—Leite, J. A.: Semantics for Dynamic Logic Programming: A Principled Based Approach. In Proceedings of LPNMR 2004, Springer.

[2] Alferes, J. J.—Leite, J. A.—Pereira, L. M.—Przymusinska, H.—Przymusinski, T. C.: Dynamic Updates of Non-Monotonic Knowledge Bases. The Journal of Logic Programming, Vol. 1–3, 2000, No. 45, pp. 43–70.

[3] Banti, F.—Alferes, J. J.—Brogi, A.—Hitzler, P.: The Well Supported Semantics for Multidimensional Dynamic Logic Programs. LPNMR 2005, LNCS 3662, Springer, pp. 356–368.

[4] Eiter, T.—Fink, M.—Sabbatini, G.—Tompits, H.: On Properties of Update Sequences Based on Causal Rejection. 2001.

[5] Homola, M.: Dynamic Logic Programming: Various Semantics Are Equal on Acyclic Programs. Proceedings of CLIMA V, 2004.

[6] Leite, J.: Evolving Knowledge Bases. IOT Press, 2003.

[7] Leite, J.: On Some Differences Between Semantics of Logic Program Updates. IBERAMIA 2004, pp. 375–385.

[8] Leite, J. A.—Alferes, J. J.—Pereira, L. M.: Multi-Dimensional Dynamic Knowledge Representation. In: Eiter, T., Faber, W., Truszczynski (Eds.): LPNMR 2001, Springer, pp. 365–378.

[9] Katsuno, H.—Mendelzon, A. O.: On the Difference Between Updating a Knowledge Base and Revising It. Proc. of KR '91.

[10] Mariničová, E.: Semantic Characterization of Dynamic Logic Programming. Diploma Thesis, Comenius University, Bratislava, 2001.

[11] Šefránek, J.: A Kripkean Semantics for Dynamic Logic Programing. Logic for Programming and Automated Reasoning, Springer, 2000.

[12] Šefránek, J.: Semantic Considerations on Rejection. Proceedings of NMR 2004, Whistler, BC, Canada.

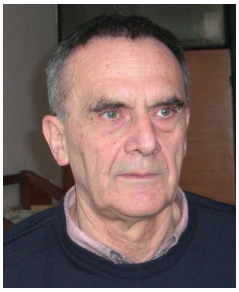
[13] Šefránek, J.: Nonmonotonic Integrity Constraints. Proceedings of 20th Workshop on Logic Programming (WLP 2006). Vienna 2006.

[14] Šefránek, J.: Irrelevant Updates and Nonmonotonic Assumptions. Proceedings of JELIA 2006.

[15] Šefránek, J.: Rethinking Semantics of Dynamic Logic Progreamming. Proceedings of NMR 2006.

[16] Šiška, J.: Refined Extension Principle for Multidimensional Dynamic Logic Programs. Proceedings of Znalosti 2005.

[17] Šiška, J.: Dynamic Logic Programming and World State Evaluation in Computer Games. Proceedings of 20th Workshop on Logic Programming (WLP 2006). Vienna 2006.



Ján Šefránek works at Comenius University, Faculty of Mathematics, Physics and Informatics, Department of Appled Informatics. His main research interests are knowledge representation, nonmonotonic reasoning and cognitive semantics.