A Kripkean Semantics for Dynamic Logic Programming

Ján Šefránek

Institute of Informatics, Comenius University
811 03 Bratislava, Slovakia
e-mail: sefranek@fmph.uniba.sk

Keywords: knowledge representation and reasoning, nonmonotonic reasoning, knowledge evolution, updates, dynamic logic programming, stable model, Kripke structure, dynamic Kripke structure

Abstract. The main goal of the paper is to propose a tool for a semantic specification of program updates (in the context of dynamic logic programming paradigm). A notion of Kripke structure $\mathcal{K}_P$ associated with a generalized logic program P is introduced. It is shown that some paths in $\mathcal{K}_P$ specify stable models of P and vice versa, to each stable model of P corresponds a path in $\mathcal{K}_P$. An operation on Kripke structures is defined: for Kripke structures $\mathcal{K}_P$ and $\mathcal{K}_U$ associated with P (the original program) and U (the updating program), respectively, a Kripke structure $\mathcal{K}_{P \oplus U}$ is constructed. $\mathcal{K}_{P \oplus U}$ specifies (in a reasonable sense) a set of updates of P by U. There is a variety of possibilities for a selection of an updated program.

1 Introduction

Knowledge evolution is a problem of crucial importance from the non-monotonic reasoning point of view. In fact, the non-monotony of reasoning is only a symptom of the evolution of knowledge.[1]

A formalization of some essential features of knowledge evolution was proposed recently in [3], see also the predecessors [14, 16, 2, 10, 11]. Knowledge bases (KB) are represented in [3] by generalized logic programs which allow default negation also in heads of the rules. As a consequence, both insertions and deletions may be specified by the rules of a program. The basic situation is as follows. A program P (the initial program) is given. P is updated by another program U (the updating program). A new program $P \oplus U$ (the updated program) is the result of the update. This situation is generalized in [3] to sequences of program updates $P \oplus U_1 \oplus \cdots \oplus U_n$. The paradigm of dynamic logic programming provides an appropriate tool for a representation of dynamically changing knowledge (dynamic knowledge bases).

[1] " ... non-monotonic behaviour ... is a *symptom*, rather than the essence of non-standard inference" according to [20].

The approach of [3] is based on this basic decision: an update KB' of one knowledge base KB by another knowledge base U should not just depend on the semantics of the knowledge bases KB and U but it should also depend on their syntax (the dependencies among literals are encoded in the syntax). The decision is implemented via a syntactic transformation. First, the set of propositional letters is extended. For each propositional letter a quintuple of new propositional letters is introduced. Second, the updated program $P \oplus U$ contains for each original clause from P and U a modified clause in the extended language. $P \oplus U$ also contains for each original propositional letter six new clauses.

The main goal of this paper is to investigate semantic foundations of dynamic logic programming paradigm. For each generalized logic program P an associated Kripke structure $\mathcal{K}_P$ is defined. Dependencies among literals are encoded in the accessibility relation of the Kripke structure. We can specify updated logic programs using a new Kripke structure $\mathcal{K}_{P\oplus U}$. $\mathcal{K}_{P\oplus U}$ is the result of an operation on Kripke structures $\mathcal{K}_P$ and $\mathcal{K}_U$, associated with an original program P and an updating program U, respectively. There is no need for an extended language and for some new types of clauses when the updated programs are created.

Updated programs are not specified by the operation in a unique way. It is not a drawback, it is a basic general property of updates. In this paper we propose some simple, "cautious" approaches to the updated program selection. In a next paper we investigate the problem more thoroughly. The approach of [3] will be discussed from the viewpoint of possible-world semantics in a more detail in the forthcoming paper, too. The main goals of this paper are:

- the introduction of the Kripkean semantics,
- a demonstration that the semantics is useful for stable models identification (computation),
- and that there is an operation on Kripke structures which can be used as a basis for a specification of updates of generalized logic programs.

The paper is structured as follows. The problem is introduced, motivated, and the preliminary technicalities are sketched in the Sections 2 – 4. The kernel of the paper: Section 5 is devoted to Kripke structures associated with given generalized logic programs. It is proved that stable models are encoded in Kripke structures (a method of stable models computation is implicit in this encoding). A construction of the Kripke structure $\mathcal{K}_{P\oplus U}$ is introduced in Section 6. The construction is defined over given Kripke structures $\mathcal{K}_U$ and $\mathcal{K}_P$ associated with programs U and P, respectively. Finally, $\mathcal{K}_{P\oplus U}$ is presented as a tool for a semantic specification of an update of P by U in Section 7 . Some results concerning the correctness of the specification are proved.

2 Interpretation updates and dynamic logic programs

The so called interpretation update approach emphasizes the role of a semantics in updating: A KB' is considered to be an update of KB by U if the set of models of KB' coincides with the set of updated models of KB. We may express it as $Mod(KB') = Update_U(Mod(KB))$, where $Mod(X)$ is the set of (relevant)[2] models of X and $Update_U(M)$ is an update of a set M of models. The update is determined by the program U, more precisely by a set of (relevant) models of U.

The goal (and a strength) of the interpretation update is an abstraction from the superficial syntactic features when specifying updates. Unfortunately, it is impossible to respect dependencies among literals, to account for justifications, using the interpretation update (and using the traditional AGM-postulates, [1, 8], too, see [21]). This is the reason why the interpretation update is refused in [10], and then also in [3]. The fact that $Update_U(KB)$ should not just depend on the interpretations of KB and U is illustrated by a simple example:

Example 1 ([3]) Let P be a program: $innocent \leftarrow not\ found\_guilty$.

Consider the stable model semantics [9] as the representation of the program meaning. The meaning of P is $Mod(P) = \{\{innocent\}\}$, the only stable model of P is $S = \{innocent\}$.

If P is updated by $U = \{found\_guilty \leftarrow\}$, then according to the interpretation update approach we should insert $found\_guilty$ into S, i.e.

$$Update(Mod(P)) = \{\{innocent, found\_guilty\}\}.$$

Of course, $\{innocent, found\_guilty\}$ is not the intended semantic characterization of the update of P by U. $\Box$

Therefore, it is decided to base the updated program $P \oplus U$ on a syntactic transformation, see [3].

3 Motivation

Our next goal is to propose a new semantics of a generalized logic program. An important feature of the semantics should be an ability to handle and to record the dependencies among literals, the justifications.

Example 2 (Continuation of the Example 1) In a sense, *innocent* is justified (in P) by *not found_guilty*. This justification is uprooted by the updating program U. It seems that dependencies, justifications, arguments are important from the semantic point of view. We propose a Kripkean semantics in order to provide a semantic characterization of the dependencies, justifications, arguments. The justifications are represented (encoded) by the accessibility relation (between interpretations).

The graphs GP and GU of the Figure 1 visualize the relevant parts of the Kripke structures associated with programs P and U, respectively. The nodes of the graphs (the possible worlds) represent (partial) interpretations. An accessibility relation is defined on the interpretations as follows. A partial interpretation

[2] For example, the relevant models may be the stable models.

M is accessible from another partial interpretation M', if the body of a rule of the given program is satisfied in M' and both the body and the head of the rule are satisfied in M (M is justified by M').

The graph GC provides a semantic characterization of the update of P by U. It is constructed over the graph GU. Some parts of the graph GP may be – in general – connected to GU, but in our example it is impossible: no edge of GP can be appended to $u1$ (no edge of GP is compatible with *found_guilty*).

Therefore, $GU = GC$ and the stable model of the updated program should be the same as the stable model of the updating program. Of course, *innocent* is not true in GC. □

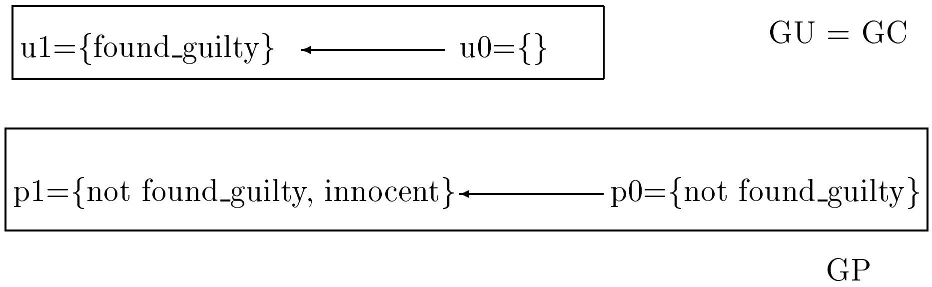

Fig. 1. The node $p0$ represents the interpretation $\{not\ found\_guilty\}$, $p1 = \{not\ found\_guilty, innocent\}$, $u0 = \emptyset$, $u1 = \{found\_guilty\}$. The edges $(p0, p1)$ and $(u0, u1)$ represent the dependencies among literals (the second member of a pair is justified by the first member). The update is determined by U, therefore the graph associated with the update (GC) is constructed over the graph associated with the program U (GU). Some parts of the graph associated with P (GP) may be – in general – connected to GU, but in our case it is impossible: no edge of GP can be put before $u0$, similarly, no edge can be appended to $u1$ (no edge of GP is compatible with *found_guilty*).

The example shows that there is a possibility of an adequate semantic treatment of dependencies among literals. Moreover, the semantics enables to identify and to compute stable models and it enables also to connect relevant parts of one Kripke structure to another. This "connectivity" serves as a basis for updates specification in terms of a purely semantic construction.

We are going to the details.

4 Preliminaries

Consider a finite set of propositional symbols $\mathcal{L}$. The set $\mathcal{L}_{not}$ is defined as $\mathcal{L} \cup \{not\ A : A \in \mathcal{L}\}$. A member of $\mathcal{L}_{not}$ is called *literal*. We will denote the set $\{not\ A : A \in \mathcal{L}\}$ by $\mathcal{D}$ (defaults, assumptions).

A *generalized clause* is a formula c of the form $L \leftarrow L_1, \ldots, L_k$, where L, L_i are literals. We will denote L also by $head(c)$ and the conjunction $L_1, \ldots, L_k$ by $body(c)$. A set of generalized clauses is called a *generalized logic program*. In the following, whenever we use "clause" or "program" we mean "generalized clause" and "generalized logic program", respectively.

For each $A \in \mathcal{L}$, A and $not\ A$ are called *conflicting literals*. A set of literals is *consistent*, if it does not contain a pair of conflicting literals. *Partial interpretation* (of a language $\mathcal{L}_{not}$) is a consistent subset of $\mathcal{L}_{not}$. *Total interpretation* is a partial interpretation $\mathcal{I}$ such that for each $A \in \mathcal{L}$ either $A \in \mathcal{I}$ or $not\ A \in \mathcal{I}$. We are interested in sets of propositional symbols determined by programs. By $\mathcal{L}^P$ we denote the set of all propositional symbols used in the program P. A partial interpretation of a *program* P is a consistent subset of $\mathcal{L}^P_{not}$. The set of all partial interpretations of P we denote by Int_P. Each inconsistent set of literals we denote by $w_\perp$.

A literal L is *satisfied* in a partial interpretation $\mathcal{I}$ if $L \in \mathcal{I}$. A clause $L \leftarrow L_1, \ldots, L_k$ is satisfied in a partial interpretation $\mathcal{I}$ if L is satisfied in $\mathcal{I}$ whenever each L_i is satisfied in $\mathcal{I}$. A partial interpretation $\mathcal{I}$ is a *model* of a program P if each clause $c \in P$ is satisfied in $\mathcal{I}$. Notice that propositional generalized logic programs can be treated as Horn theories: each literal $not\ A$ can be considered as a new propositional symbol (if $not\ A \in \mathcal{L}$ it has to be renamed). The least model of the Horn theory H we denote by $Least(H)$.

Definition 3 (Stable model, [3]) Let P be a generalized logic program and S be an interpretation of P. It is said that S is a stable model of P iff $S = Least(P \cup S^-)$, where $S^- = \{not\ A : not\ A \in S\}$. □

We will visualize Kripke structures as graphs. If e is an edge (w_i, w_{i+1}) of a graph G, the node w_i is called the *source* of e and w_{i+1} the *target* of e. A sequence σ of edges $(w_0, w_1), (w_1, w_2), \ldots, (w_{n-1}, w_n)$ is called a path, w_0 we denote also by $begin(\sigma)$ and w_n by $end(\sigma)$.

5 Kripke structure associated with a program

A notion of Kripke structure associated with a program is defined in this Section. Moreover, it is shown that some distinguished paths in the defined structure represent stable models of logic programs and, conversely, for each stable model there is a distinguished path in the Kripke structure.

The basic idea of our approach was illustrated in the Example 2. A more complicated example is presented below.

Example 4 ([17]) Let P be

$$p \leftarrow not\ q, r$$
$$q \leftarrow not\ p$$
$$r \leftarrow not\ s$$
$$s \leftarrow not\ p.$$

A fragment of the $\mathcal{K}_P$ is depicted in the Figure 2. The nodes are partial interpretations. We distinguish two kinds of edges – ρ_1, and ρ_2.

Consider $(w1, w2)$, an example of an ρ_1-edge, where $w1 = \{not\ p\}$ and $w_2 = \{not\ p, q, s\}$. There are two clauses with the body satisfied in $w1$. Consequences of these clauses are appended to $w1$, the possible world $w2$ is the result of this operation.

Finally, a motivation for ρ_2. There is no total interpretation u such that $(w_2, u) \in \rho_1$, i.e. no clause is applicable to the partial interpretation $w_2 = \{not\ p, q, s\}$ (except of $q \leftarrow not\ p$ and $s \leftarrow not\ p$, but they do not change the possible world $w2$). It means, that P does not enable to justify the truth of r (if we suppose $w2$). Therefore, we may assume by default that r is not true (w.r.t. P and $w2$). The ρ_2-edge from $w2$ to $w3$ represents a completion of $\{not\ p, q, s\}$ by $not\ r$.

□

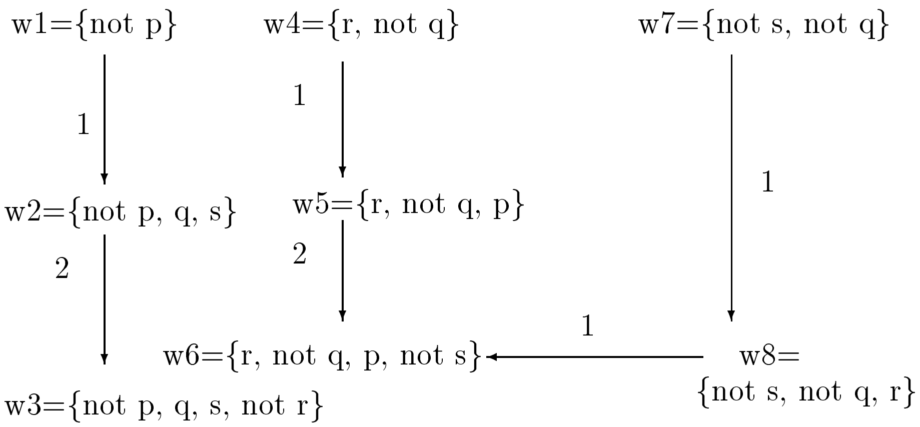

Fig. 2. A fragment of $\mathcal{K}_P$. An edge labeled by i is a ρ_i-edge.

Let us summarize: A ρ_1-edge corresponds to an application of a clause to a partial interpretation. A clause c is applicable to a partial interpretation w if $w \models body(c)$. In general, for each $c \in P$: if w is a model of $body(c)$, then $head(c) \in w'$ for some w' such that $w \subseteq w'$ and $(w, w') \in \rho_1$. Intuitively, (w, w') represents a step in a computation bottom-up.

If an atom A is not computed (bottom-up), we assume that $not\ A$ holds. The relation ρ_2 represents a completion (by default negations) of partial interpretations that cannot be changed by some clauses of P.

Now we are ready to define a Kripke structure $\mathcal{K}_P$ associated with P.

Definition 5 Let P be a program. A Kripke structure $\mathcal{K}_P$ associated with P is a pair (W, ρ), where:

- $W = Int_P \cup \{w_\perp\}$, W is called the set of possible worlds, Int_P is the set of all partial interpretations of P, $w_\perp$ 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 is composed of two relations: $\rho = \rho_1 \cup \rho_2$, where
  1. the accessibility relation ρ_1 contains the set of all pairs (w, w') such that $w' = w \cup \{head(c_i) : i = 1, \ldots, k\}$, where $c_1, \ldots, c_k$ are (not necessary all) clauses from P such that $w \models body(c_i)$,
  2. if w is not a total interpretation and for no $u \neq w$ there is an edge $(w, u) \in \rho_1$, then $(w, w') \in \rho_2$, where $w' = w \cup \{not\ A : A \notin w\}$.

□

Of course, $\mathcal{K}_P$ may be viewed as a graph.

Definition 6 ρ-path is a sequence σ of edges $(w_0, w_1), (w_1, w_2), \ldots, (w_{n-1}, w_n)$ in $\mathcal{K}_P$ such that each $(w_i, w_{i+1}) \in \rho$.

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

Sometimes we denote paths by the shorthand $\langle w_0, w_1, w_2, \ldots, w_{n-1}, w_n \rangle$. Similarly, a ρ_1-path could be defined.

We have seen that Kripke structures are appropriate for recording justifications (of interpretations by another interpretations). The justifications have to be non-circular. There are two kinds of basic assumptions – facts (with empty interpretation as the justification, edges to facts are $\emptyset$-rooted) and default negations (subsets of $\mathcal{D}$), called non-monotonic assumptions in TMS [6]: if there is no evidence against, we assume *not* A (where A is an atom). Therefore, the Kripke structure $\mathcal{K}_P$ associated with a program P enables to identify (and to compute) the stable models of P.

Example 7 Let us return to the Example 4 (and to the Figure 2)

There is no fact in P, hence there is no $\emptyset$-rooted path in $\mathcal{K}_P$. As a consequence, relevant paths are only those rooted in some w such that $\emptyset \neq w \subseteq \mathcal{D}$ (only defaults can be assumed). There is a $\{not\ s, not\ q\}$-rooted ρ-path terminated in a stable model $\{not\ s, r, not\ q, p\}$ and a $\{not\ p\}$-rooted (similarly, also a $\{not\ p, not\ r\}$-rooted) ρ-path terminated in another stable model $\{p, not\ q, not\ s, r\}$. □

Now we are ready to state conditions for stable models in terms of nodes and paths in $\mathcal{K}_P$.

Definition 8 Let P be a program, σ be an acyclic ρ-path $\langle w_0, w_1, \ldots, w_n \rangle$ from $\mathcal{K}_P$. We say that σ is *correctly rooted*, if

- either $w_0 = \emptyset$
- or $\emptyset \neq w_0 \subseteq \mathcal{D}$. □

Theorem 9 *Let P be a program, $\mathcal{K}_P$ be the Kripke structure associated with P, $\sigma = (w_0, w_1), (w_1, w_2), \ldots, (w_{n-1}, w_n)$ be an acyclic ρ-path in $\mathcal{K}_P$ terminated in a total interpretation w_n.*

If σ is correctly rooted, then w_n is a stable model of P.

Proof Sketch:
Let P be a generalized logic program. Let P' be $P \cup \{not\ A \leftarrow: not\ A \in w_n^-\}$. Consider P' as a definite program (each literal *not* A is a new propositional letter) with integrity constraints of the form $\leftarrow A, not\ A$ for each propositional symbol $A \in \mathcal{L}^P$.

According to [3], see also the Definition 3: w_n is a stable model of P iff $w_n = Least(P \cup w_n^-)$, where $w_n^- = \{not\ A : not\ A \in w_n\}$.

We assume that $\sigma = \langle w_0, w_1, \ldots, w_{n-1}, w_n \rangle$ is correctly rooted and w_n is a total interpretation. If $(w_{n-1}, w_n) \in \rho_1$ it is straightforward to show that $w_n = Least(P \cup w_n^-)$. Otherwise, notice that $w^* = w_0 \cup (w_n \setminus w_{n-1}) \subseteq w_n^-$ and $\langle w^*, (w_1 \cup w^*), \ldots, (w_{n-1} \cup w^*) \rangle$ is a correctly rooted acyclic ρ_1-path terminated in w_n. It means, $Least(P') = w_n$. Clearly, integrity constrains are satisfied in w_n. Finally, $Least(P') = Least(P \cup w_n^-)$. □

Theorem 10 *Let S be a stable model of a generalized logic program P and $\mathcal{K}_P$ be a Kripke structure associated with P.*

There is a correctly rooted and acyclic ρ-path $\sigma = \langle w_0, \ldots, w_n, S \rangle$ in $\mathcal{K}_P$ terminated in S.

Proof Sketch:
We again use $S = Least(P \cup S^-)$. We can construct a correctly rooted (in S^-) ρ-path terminated in S both if $S^- = \emptyset$ and if $S^- \neq \emptyset$. □

Fact 11 *Let P, $\mathcal{K}_P$ be as in the Theorem 10. If $(\mathcal{D}, w_\perp) \notin \rho_1$, then $\mathcal{D}$ is the only stable model of P.*

Proof: First, $\mathcal{D}$ is a stable model of P: Let $\mathcal{D}' \neq \emptyset$ be a proper subset of $\mathcal{D}$. Then $\langle \mathcal{D}', \mathcal{D} \rangle$ is a correctly rooted ρ-path terminated in the total interpretation $\mathcal{D}$.

Let $\langle w_0, \ldots, w_n \rangle$ be a correctly rooted ρ-path terminated in a total interpretation $w_n \neq \mathcal{D}$. Hence, $A \in w_n$ for at least one atom A. Of course, there is an atom A, a rule $A \leftarrow L_1, \ldots, L_k$, and a correctly rooted ρ-path $\langle u_0, \ldots, u_m \rangle$ such that $u_m = w_n$ and $u_0 \models L_1, \ldots, L_k$, where $u_0 \subset \mathcal{D}$. Therefore, $\mathcal{D} \models L_1, \ldots, L_k$ and $(\mathcal{D}, w_\perp) \in \rho_1$. It means, $\mathcal{D}$ is the only stable model of P. □

Fact 12 *Let P and $\mathcal{K}_P$ be as in the Theorem 10. If $\sigma = \langle w_0, w_1, \ldots, w_n \rangle$ is a ρ-path in $\mathcal{K}_P$, terminated in $w_n \neq w_\perp$, then w_n is a model of P.*

If M is a model of P, then there is a ρ-path in $\mathcal{K}_P$ terminated in M.

Proof: If $c \in P$ and $w_i \models body(c)$ for some w_i, then $head(c) \in w_{i+1}$.

M is not an isolated node: If $M = \mathcal{D}$, we can use the edge $(\mathcal{D}', \mathcal{D})$ from the proof of the Fact 11. If $M \neq \mathcal{D}$ and $w = M \setminus \mathcal{D}$, then there is a path σ in $\mathcal{K}_P$ such that $begin(\sigma) = w$ and $end(\sigma) = M$.

M is a terminal node: $(M, w_\perp) \notin \rho_1$, otherwise there is a clause $c \in P$ which is not true in M. □

6 Updated Kripke structures

We are going to construct a Kripke structure $\mathcal{K}_{P \oplus U}$ over two Kripke structures, over $\mathcal{K}_P$ (let us recall that it specifies the semantics of an original program P) and over $\mathcal{K}_U$ (specifying the semantics of an updating program U). We intend to use the structure $\mathcal{K}_{P \oplus U}$ as a semantic specification of an updated program.

First we motivate definitions of some notions needed for the construction of $\mathcal{K}_{P \oplus U}$. The concept called *continuation node* is the most important one.

We assume that the nodes of $\mathcal{K}_{P \oplus U}$ are the (partial) interpretations of the language $\mathcal{L}^{P \cup U}$.

Example 13 ([3]) Let $P = \{s \leftarrow not\ t; a \leftarrow t; t \leftarrow\}$ be given. We assume that P is updated by $U = \{not\ t \leftarrow p; p \leftarrow\}$. The relevant parts of $\mathcal{K}_P$ and $\mathcal{K}_U$ are illustrated on the Figure 3. We construct $\mathcal{K}_{P \oplus U}$ over $\mathcal{K}_U$, the update is dominated by $\mathcal{K}_U$. If P can consistently add something to U, it should be accepted. Hence, some paths from $\mathcal{K}_P$ may be connected to $\mathcal{K}_U$.

Consider possible worlds from $\mathcal{K}_P$: $w1 = \emptyset$, $w2 = \{t\}$, $w3 = \{t, a\}$, $w4 = \{t, a, not\ s\}$, $w5 = \{not\ t\}$, $w6 = \{not\ t, s\}$. Similarly, the relevant possible worlds from $\mathcal{K}_U$ are: $u1 = \emptyset$, $u2 = \{p\}$, $u3 = \{p, not\ t\}$.

An important decision should be made: Which paths of $\mathcal{K}_P$ may be connected to which nodes of $\mathcal{K}_U$?

Above all, the nodes of $\mathcal{K}_U$ which terminate ρ_1-paths are the reasonable continuation nodes. If we connect a path of $\mathcal{K}_P$ to an intermediate node of a ρ_1-path of $\mathcal{K}_U$, then some information of U could be lost. On the other hand, the acceptance of default assumptions should be postponed until all ρ_1-paths of $\mathcal{K}_{P \oplus U}$ are constructed.

Let us summarize, we have a first example of continuation nodes – the terminal nodes of ρ_1-paths.

Now we proceed to the connection of relevant paths to the continuation nodes. A path σ of $\mathcal{K}_P$ may be connected to a continuation node w of $\mathcal{K}_U$, if $begin(\sigma)$ is compatible – in a sense – with w.

In our simple example, the only relevant continuation node is $u3$. If we connect the path $\langle w1, w2, w3, w4 \rangle$ to the continuation node $u3 = \{p, not\ t\}$, the first edge $(w1, w2)$ leads to $w_\perp$ – the node $w2 = \{t\}$ contradicts the node $u3$.

On the contrary, the path $(w5, w6)$ may be connected successfully to the node u_3. The node $w5$ is compatible with the node $u3$: $w5 \subseteq u3$, it means that every literal satisfied in $u3$ is satisfied in $w5$, too. Moreover, $w6$ and $u3$ are consistent.

Therefore, the path of $\mathcal{K}_{P \oplus U}$ could be $\sigma = \langle u1, u2, u3, w, w' \rangle$, where $w = u3 \cup w6$ (notice that $u3 = u3 \cup w5$) and $w' = w \cup \{not\ a\}$. The edge (u_3, w) we obtain by connecting (w_5, w_6) to $u3$. The last edge, (w, w') is a ρ_2-edge. This completion is made w.r.t. the language $\mathcal{L}^{P \cup U}_{not}$. The relevant part of $\mathcal{K}_{P \oplus U}$ is on the Figure 3.

The path σ is correctly rooted and it is terminated by the total interpretation w'. We can consider a correctly rooted path from $\mathcal{K}_{P \oplus U}$ which terminates in a total interpretation to be a basis for a semantic specification of updated programs $P \oplus U$.

By the way, $w' = \{p, not\ t, s, not\ a\}$ is the only stable model (modulo irrelevant literals) of the updated program $P \oplus U$, as defined in [3]. □

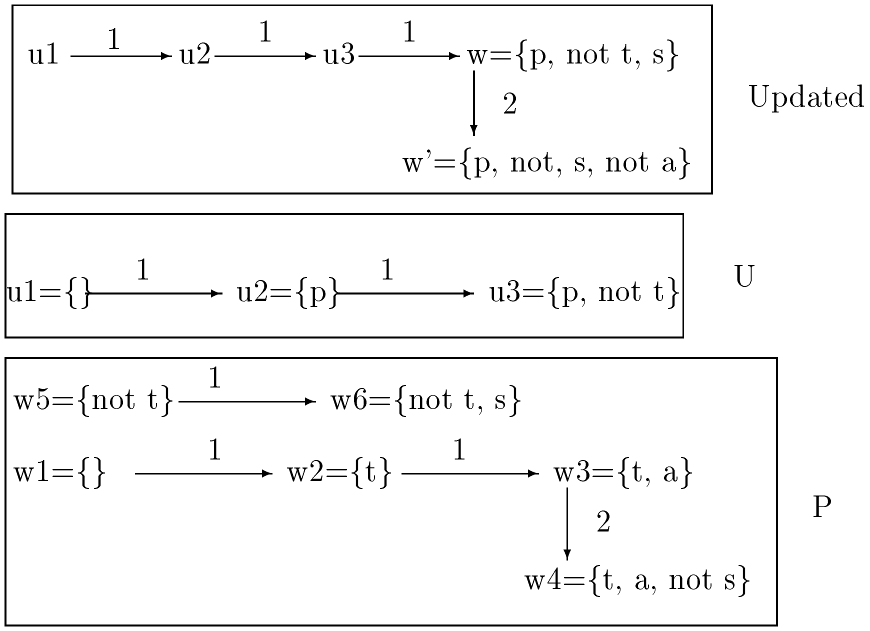

Fig. 3. The relevant parts of $\mathcal{K}_P$ and $\mathcal{K}_U$ from the Example 13. The edges are labeled as in the Figure 2. The edge $(w5, w6)$ from $\mathcal{K}_P$ is connected to the path $\langle u1, u2, u3\rangle$ from $\mathcal{K}_U$ and the path is completed by the edge (w, w'). The resulting path from $\mathcal{K}_{P \oplus U}$ is $(u1, u2), (u2, u3), (u3, w), (w, w')$, where $w = u3 \cup w6$ and $w' = w \cup \{not\ a\} = \{p, not\ t, s, not\ a\}$.

The example motivates our first decision about continuation nodes: Each terminal node of a ρ_1-path from $\mathcal{K}_U$ is a continuation node. Let w be a continuation node. We may connect a path σ from $\mathcal{K}_P$ to w, if all formulae satisfied in w are satisfied also in $begin(\sigma)$ and if a consistency criterion is satisfied. The node w can be considered as a justification of the connected path.

Now we extend our idea of continuation nodes: It is acceptable to connect some paths of $\mathcal{K}_P$ before some nodes of $\mathcal{K}_U$: Possible continuation nodes are also $w_0 = \emptyset$ and $\emptyset \neq w_0 \subseteq \mathcal{D}$, if there is in $\mathcal{K}_U$ no ρ_1-path rooted in w_0.

We are now ready to present a series of definitions.

Definition 14 Let $\mathcal{K}_U$ be a Kripke structure associated with an update program U.

Continuation nodes of $\mathcal{K}_U$ are

(i) all nodes terminated a ρ_1-path
(ii) $\emptyset$ or w such that $\emptyset \neq w \subseteq \mathcal{D}$, if they are not the source of a ρ_1-edge.

$\Box$

Definition 15 The path $\sigma = \langle w_0, w_1, \ldots, w_n \rangle$ from $\mathcal{K}_P$ may be connected to a node w from $\mathcal{K}_U$ iff $w_0 \subseteq w$ and $w \cup w_1$ is consistent. $\Box$

Definition 16 Let $\sigma = \langle u_0, \ldots, u_n \rangle$ be a ρ-path and w be a node.

Then *connect* σ to w is a partial operation as follows: if σ may be connected to w, then $(w, u_1 \cup w), \ldots, (u_{n-1} \cup w, u_n \cup w)$ is a ρ-path. If for some $i > 1$ holds that $w \cup u_i$ is inconsistent, it is replaced by $w_\perp$ and the rest of the path is removed. $\Box$

Definition 17 Let $\mathcal{K}_P$ and $\mathcal{K}_U$ be the Kripke structures associated with non-empty programs P and U, respectively.

We construct $\mathcal{K}_{P \oplus U}$ as follows:

1. each ρ_1-edge from $\mathcal{K}_U$ is an ρ_1-edge of $\mathcal{K}_{P \oplus U}$,
2. for each continuation node w from $\mathcal{K}_U$ and each ρ_1-path $\sigma = \langle u_0, u_1, \ldots, u_n \rangle$ from $\mathcal{K}_P$: connect σ to w,
3. introduce new ρ_2-edges whenever it is possible.

$\Box$

7 Updated programs specification

In this Section we present some useful properties of $\mathcal{K}_{P \oplus U}$ and then we sketch some simple methods of updated programs construction.

7.1 Good worlds and the stability condition

First, we introduce a definition in order to simplify the description of $\mathcal{K}_{P \oplus U}$. By analogy to the results of Section 5, correctly rooted ρ-paths terminated in a total interpretation from $\mathcal{K}_{P \oplus U}$ deserve a special interest. We will use them as a basis for a specification of $P \oplus U$.

Definition 18 (Good worlds) Let a Kripke structure $\mathcal{K}_{P \oplus U}$ be given. Let σ be a correctly rooted ρ-path from $\mathcal{K}_{P \oplus U}$ terminated in a total interpretation w.

We say that σ is a *distinguished* ρ-path and w is a *good world*. $\Box$

Now it can be said that we will use distinguished ρ-paths and good worlds as a tool for a specification of $P \oplus U$. We accept a cautious strategy in this paper: for each distinguished ρ-path σ (and the corresponding good world w) from $\mathcal{K}_{P\oplus U}$ we are aiming at specifying a program Π such that w is the only stable model of Π. It means, we consider $\mathcal{K}_{P\oplus U}$ as the specification of a variety of updates.

Our next goal is to define a criterion of a reasonable update of P by U. Updated programs specified by $\mathcal{K}_{P\oplus U}$ should satisfy the criterion. The criterion is called the stability condition. It provides a natural characterization of what to accept (or what to reject) from the original program P, if a model M of the updating program U is given. The model M represents an (alternative) belief set dominating the update.

The results of this Subsection – Fact 23, Theorem 24, and Consequence 26 show that

- stability condition and good worlds agree, in a sense,
- both concepts (stability condition, good worlds) enable to specify updated programs compatible with U,
- good worlds are stable models of the updated programs.

A crucial issue is what to accept and what to reject from the original program P, if the updating program U is given. Next example motivates why sometimes the defaults from U override facts from P.

Example 19 Let P be $\{a \leftarrow; b \leftarrow a\}$ and U be $\{not\ b \leftarrow c; c \leftarrow not\ a; a \leftarrow not\ c\}$.

U specifies an intuitively acceptable update of P: a new propositional symbol c is introduced, the meaning of c is the opposite to the meaning of a, and c is a condition for $not\ b$ (while a – according to P – is a condition for b). Notice that no path of $\mathcal{K}_U$ is rooted in $\emptyset$ and the stable models of U are based on some default assumptions.

The relevant parts of $\mathcal{K}_P$, $\mathcal{K}_U$, and $\mathcal{K}_{P\oplus U}$ are illustrated on the Figure 4. The continuation nodes of $\mathcal{K}_U$ are $w2$ and $w4$. The ρ-path $\langle u0, u1, u2\rangle$ from $\mathcal{K}_P$ may not be connected to $w2$, the edge $(u0, u1)$ leads immediately to the $w_\perp$ ($w2 \cup u1$ is not consistent). If we connect the path to the node $w4$ we get $w = \{not\ c, a, b\}$ (a redundant cycle $(w4, w4) = (w4, w4 \cup u1)$ is removed).

Let us summarize – we have two ρ-paths terminated in a total interpretation in $\mathcal{K}_{P\oplus U}$: $\langle w3, w4, w\rangle$ and $\langle w0, w1, w2\rangle$. The total interpretation w respects the facts from P, but the total interpretation $w2$ does not respect them – it prefers the default assumptions of U.

Our attitude here is a cautious one: we allow both interpretations to determine an updated program $P \oplus U$. $\Box$

The example 19 shows that sometimes it is justified to reject some facts of P. Let us suppose that a literal L holds in a stable model S of the updating program U and $L' \leftarrow$ is a fact of the original program P, where L and L' are conflicting literals. The fact is rejected, if we accept the belief set S.

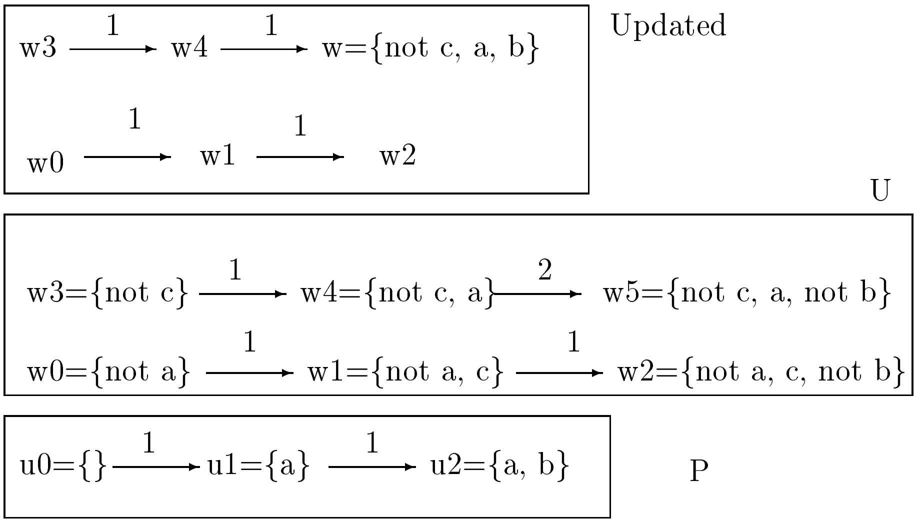

Fig. 4. A fragment of graphs from the Example 19. The relevant parts of $\mathcal{K}_{P \oplus U}$ are the same as of $\mathcal{K}_U$ with the only exception – the node $w = \{not\ c, a, b\}$ instead of $w5$ and $(w4, w) \in \rho_1$.

Definition 20 Let M be an interpretation of an updating program U, and L, L' be conflicting literals. Let P be an original program.

- $Rejected(M) = \{c \in P : (\exists c' \in U)\ ((head(c), head(c')$ are conflicting literals and $M \models body(c')\} \cup \{(L \leftarrow) \in P : L' \in M\}$
- $Residue(M) = U \cup (P \setminus Rejected(M))$
- $Defaults(M) = \{not\ A : (\forall c \in Residue(M))\ (head(c) = A \Rightarrow M \not\models body(c))\}$, where A is an atom.

□

Our definition of rejected clauses slightly differs from that of [3]. The basic difference is that in [3] facts from P are not rejected when they are in conflict with a stable model S.[3] Similarly, our definition of defaults is different: we define defaults with respect to the $Residue(M)$, while in [3] they are defined w.r.t. $P \cup U$.

Definition 21 (Stability condition) Let programs P, U be given. Let w be a possible world from $\mathcal{K}_{P \oplus U}$. We say that w satisfies the *stability condition*, if holds

$$w = Least(Residue(w) \cup Defaults(w)).\square$$

[3] From this point of view, the approach of [10, 11] is similar to our approach. On the other hand, $Rejected(M)$ may be defined in a distinct way also in our setting. A more detailed comparison and an analysis of some possibilities will be presented in the forthcoming paper.

Next example shows that some good worlds which do not satisfy the stability condition as defined in [3][4] satisfy our Definition 21. Moreover, each good world satisfies the condition (see Theorem 24 below).

Example 22 Let us recall the Example 19. One of the distinguished paths terminates in the good world $w2 = \{not\ a, c, not\ b\}$. Consider a modification of $Residue(w2)$ and $Defaults(w2)$. Let $\Pi \subseteq P$ be a consistent set of clauses such that $(a \leftarrow) \in \Pi$. Let Δ be $\{not\ A : \ \forall c \in (\Pi \cup U)\ (head(c) = A \Rightarrow w2 \not\models body(c))\}$. Then $w2 \neq Least(\Pi \cup U \cup \Delta)$, because of $not\ A \notin Least(\Pi \cup U \cup \Delta)$. It means, the good world $w2$ does not satisfy the stability condition for the modified $Residue(w2)$ and $Defaults(w2)$.

Notice that $Residue(w2)$ as defined in [3] contains $a \leftarrow$.

According to our Definition 20: $Rejected(w2) = P$, $Residue(w2) = U$, and $Defaults(w2) = \{not\ a, not\ b\}$, hence $Least(Residue(w2) \cup Defaults(w2)) = w2$. $\square$

We proceed to the results of this Subsection. The stability condition provides an important criterion: Each possible world w satisfying this condition respect the information of the updating program U, w is a model of U. Moreover, w is a stable model of $Residue(w)$, where $Residue(w)$ can be viewed as a natural updated program.

Fact 23 *Let P, U be programs. If a possible world w from $\mathcal{K}_{P \oplus U}$ satisfies the stability condition, then*

- *w is a model of U*
- *w is a stable model of $Residue(w)$.*

Proof Sketch: It is straightforward to show that w is a model of U: $w = Least(Residue(w) \cup Defaults(w)) = Least(U \cup (P \setminus Rejected(w)) \cup Defaults(w))$.

If $not\ A \in Defaults(w)$, then $A \notin w$, therefore $not\ A \in w^-$, i.e.

$$Least(Residue(w) \cup Defaults(w)) \subseteq Least(Residue(w) \cup w^-).$$

Let us suppose that $not\ A \in w^-$ and there is no clause $c \in Residue(w)$ such that $head(c) = not\ A$ and $w \models body(c)$. Therefore, for each clause $c' \in Residue(w)$ holds that if $head(c') = A$, then $w \not\models body(c')$ (otherwise $A \in w$). Hence, it holds that

$$Least(Residue(w) \cup w^-) \subseteq Least(Residue(w) \cup Defaults(w)).\square$$

Now we demonstrate the important role of distinguished paths and good worlds for updated programs specification. Good worlds and worlds satisfying the stability condition coincide.

Theorem 24 *Let P, U be given. Then w_n is a good world from $\mathcal{K}_{P \oplus U}$ iff w_n satisfies the stability condition.*

[4] The term "stability condition" is not used in [3].

Proof Sketch:
$\Rightarrow$
We assume a correctly rooted ρ-path $\sigma = \langle w_0, w_1, \ldots, w_n \rangle$ terminated in w_n.
If $(w_{n-1}, w_n) \in \rho_1$, then $\mathit{Defaults}(w_n) = w_0$.
Otherwise, $\mathit{Defaults}(w_n) = w_0 \cup (w_n \setminus w_{n-1})$ and in both cases we have a "computation bottom-up" starting in w_0 and terminated in w_n, i.e.

$$w_n = \mathit{Least}(\mathit{Residue}(w_n) \cup \mathit{Defaults}(w_n)).$$

$\Leftarrow$
$w_n = \mathit{Least}(\mathit{Residue}(w_n) \cup \mathit{Defaults}(w_n))$ is assumed. According to the Fact 23, w_n is a stable model of the $\mathit{Residue}(w_n)$. It means, there is a correctly rooted ρ-path σ in $\mathcal{K}_{\mathit{Residue}(w_n)}$ terminated in w_n (the Theorem 10). Lemma 25 shows that w_n is a good world also w.r.t. $\mathcal{K}_{P \oplus U}$. $\Box$

Lemma 25 *Let P and U be programs and w_n be a total interpretation from $\mathcal{K}_{P \oplus U}$.*

If $\sigma = \langle w_0, \ldots, w_n \rangle$ is a correctly rooted ρ-path from $\mathcal{K}_{\mathit{Residue}(w_n)}$ which is terminated in w_n, then there is a correctly rooted ρ-path σ' in $\mathcal{K}_{P \oplus U}$ which is terminated in w_n.

Proof Sketch: If $(w_i, w_{i+1}) \in \sigma$ and there are clauses $c \in U$ and $d \in P$ such that $w_i \models \mathit{body}(c)$, $w_i \models \mathit{body}(d)$, and $\mathit{head}(c), \mathit{head}(d) \in w_{i+1}$, $\mathit{head}(c) \neq \mathit{head}(d)$, then there is a path $\langle w_i, w', w_{i+1} \rangle$, where $w' = w_i \cup \{L \in w_{i+1} : \exists c \in U \; (\mathit{head}(c) = L)\}$.

By repeating this construction we get a path from $\mathcal{K}_{P \oplus U}$ which is correctly rooted and terminated in w_n. $\Box$

Finally, the next straightforward consequence shows that good worlds from $\mathcal{K}_{P \oplus U}$ have reasonable properties from the viewpoint of updated programs specification.

Consequence 26 *Let P, U be programs and w be a good world of $\mathcal{K}_{P \oplus U}$. Then*

- *w is a model of U,*
- *w is a stable model of $\mathit{Residue}(w)$.*

It is time to specify $P \oplus U$ (using distinguished ρ-paths and good worlds).

7.2 Updated programs

In general, each (non-trivial) update may be realized in different ways. (Moreover, we accept the stable-model semantics, therefore it is natural to allow more results of an update.)

The most simple possibility is to consider $\mathit{Residue}(w)$ as an updated program (for any good world w).

A further possible specification of an updated program: $\mathcal{K}_{P \oplus U}$ determines a set $\mathcal{S}$ of programs[5] as follows. Each distinguished ρ-path σ determines one program Π from the set.

The construction of Π: Let a distinguished ρ-path $\sigma = \langle w_0, \ldots, w_n \rangle$ be given. For each edge $(w_i, w_{i+1}) \in \rho_1 \cup \rho_2$ let $w_i = \{L_1, \ldots, L_m\}$ and $w_{i+1} \setminus w_i = \{L'_1, \ldots, L'_k\}$. We put $L'_j \leftarrow L_1, \ldots, L_m$ into Π for each $j = 1, \ldots, k$.

The good world $end(\sigma)$ of σ is the (only) stable model of Π:

Fact 27 *Let Π be constructed from $\mathcal{K}_{P \oplus U}$ over a distinguished ρ-path σ as above.*

Then the good world $end(\sigma)$ of σ is the (only) stable model of Π.

Proof Sketch: First, $end(\sigma)$ is a stable model of Π: it is a good world and a terminal of a correctly rooted path from $\mathcal{K}_\Pi$. Second, it is the only total interpretation of $\mathcal{K}_\Pi$ which terminates a correctly rooted ρ-path. □

Π introduced above is a member of a family of representatives of $P \oplus U$ in a sense.

Of course, there are more sophisticated possibilities how to construct $P \oplus U$. A special attention deserves an idea of partial evaluation of P with respect to the continuation nodes of $\mathcal{K}_U$, see [12].

All presented proposals for a specification of an updated program on the basis of $\mathcal{K}_{P \oplus U}$ are cautious, they select one of the possible alternatives. Skeptical solutions will be discussed in a forthcoming paper.

Remark 28 Our approach can be expressed also in terms of stable model (answer set) programming paradigm [15, 13, 17]. Consider a model w of U. It can be said that the model represents the information of U (from a point of view). The model can be viewed as a basis of a constraint satisfaction process and the rules of P can be viewed as constraints. Some of the constraints are not applicable to w (w does not satisfy the constraints), they are rejected. The rest of the constraints is applicable and may be added to the rules from U. The application of the constraints results in some modifications of w (the solutions of the constraint satisfaction process).

8 Conclusions

The approach presented in this paper shows that updates of programs may be specified in a purely semantic frame. The approach is very simple, it does not need an extension of the language and/or of the program(s). There is a variety of syntactic implementations of given semantic specification. In this paper some straightforward constructions are proposed.

The main contributions of the paper may be summarized as follows:

- a semantic treatment of justifications in terms of Kripke structures,

[5] We may say that $\mathcal{S}$ is a family of representatives for $P \oplus U$.

– a characterization of stable models in terms of Kripke structures,
– a semantic (and sensitive w.r.t. justifications) characterization of generalized logic programs revisions.

A forthcoming paper will be devoted to a more thorough comparison of the approach of [3] and of the approach presented here. Further, more sophisticated possibilities of $P \oplus U$ specification in terms of $\mathcal{K}_{P \oplus U}$ will be investigated. Similarly for an extension to the case of dynamic program updates specification by $\mathcal{K}_{\oplus\{P_s : s \in S\}}$ (some priorities have to be assigned to the edges of the Kripke structures).

Also the topic of inconsistent generalized logic programs and their revisions (their use in dynamic logic programming) devotes an interest.

Another open problem is a compilation of stable model computing in the spirit of [4], see also [5]. The off-line part of the computation provides a construction of the Kripke structure associated with the given program. The on-line part consists in identifying the stable models in the Kripke structure.

Our approach uses an old idea of TMS, [6] (and a formal reconstruction of TMS by Elkan, [7]). Updates must respect dependencies among literals. Justifications of believed facts are important parts of knowledge bases. Argumentation must not be a circular one. There are some basic assumptions of each argumentation (justification) – axioms (facts) and default assumptions.

Last, some remarks about dynamic Kripke structures (DKS): The concept was introduced and studied in [18, 19]. The basic idea about DKS consisted in some transformations of possible worlds. A possibility to modify dynamically the accessibility relation was proposed in [19]. Now, in the present paper the dynamics is implicit in the operation on Kripke structures. Hence, a generalization of the DKS concept (and its applications to the study of knowledge evolution, of hypothetical, nonmonotonic reasoning) is a goal of our research in the future.

Acknowledgments I would like to thank anonymous referees for their comments. Thanks to Štefan Baloc, Damas Gruska, and Ivan Strohner for the remarks to an earlier version of the paper. The work was partially supported by Slovak agency VEGA under the grant 1/7654/20.

References

1. Alchourrón, C., Makinson, D., Gärdensfors, P. *On the logic of theory change. Partial meet contraction and revision functions.* Journal of Symbolic Logic, 50:510-530 (1985)
2. Alferes, J.J., Pereira,L.M. *Update-programs can update programs.* LNAI 11126, Springer 1996
3. Alferes, J.J., Leite, J.A., Pereira, L.M., Przymusinska, H., Przymusinski, T.C. *Dynamic Logic Programming.* Proc. KR'98, 1998
4. Cadoli, M., Donini, F.M., Schaerf, M. *Is intractability of non-monotonic reasoning a real drawback?* Artificial Intelligence 88, 1-2, 215-251

5. Cadoli, M., Donini, F.M., Liberatore, P., Schaerf, M. *Space Efficiency of Propositional Knowledge Representation Formalisms.* Journal of Artificial Intelligence Research 13 (2000), 1-31
6. Doyle, J. *A Truth Maintenance System.* AI Journal 12 (1979),231-272
7. Elkan, C. *A Rational Reconstruction of Nonmonotonic Truth Maintenance Systems.* AI Journal 43 (1990) 219-234
8. Gärdenfors, P., Rott. H. *Belief Revision.* In D. Gabbay, C. Hogger, J. Robinson: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 4, Epistemic and Temporal Reasoning, 35-132, 1995
9. Gelfond, M., Lifschitz, V. *The Stable Model Semantics for Logic Programming.* Proc. 5th ICLP, MIT Press, 1988, 1070-1080
10. Leite, J., Pereira, L. *Generalizing Updates: from models to programs.* In LNAI 1471, 1997
11. Leite, J., Pereira, L. *Iterated Logic Programs Updates.* In Proc. of JICSLP98
12. Lifschitz, V., Turner, H. *Splitting a Logic Program.* Proc. of the 11th Int. Conf. on Logic Programming, 1994, 23-37
13. Lifschitz, V. *Answer set planning.* Proc. of ICLP, 1999
14. Marek, W., Truszczynski, M. *Revision Programming.* Theoretical Computer Science, 190 (1998), 241-277
15. Marek, W., Truszczynski, M. *Stable models and an alternative logic programming paradigm.* In The Logic Programming Paradigm: a 25-Year Perspective, 375-398, Springer 1999
16. Przymusinski, T., Turner, H. *Update by inference rules.* The Journal of Logic Programming, 1997
17. Niemelä, I. *Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm.* Workshop on computational aspects of nonmonotonic reasoning, Trento, 1998
18. Šefránek, J. *Dynamic Kripke Structures.* Proc. of CAEPIA'97, Malaga, Spain
19. Šefránek, J. *Knowledge,Belief, Revisions, and a Semantics of Non-Monotonic Reasoning.* Proc. LPNMR'99, Springer 1999
20. J. Van Benthem, *Semantic Parallels in Natural Language and Computation*, in: Logic Colloquium '87, eds. Ebbinghaus H.-D. et al., 1989, 331-375, North Holland, Amsterdam
21. Witteveen, C., Brewka, G. *Skeptical reason maintenance and belief revision.* Artificial Intelligence 61 (1993), 1-36