Rethinking Semantics of Dynamic Logic Programming

Ján Šefránek
Institute of Informatics, Comenius University Bratislava, Slovakia, sefranek@fmph.uniba.sk

Abstract

A dynamic logic program represents an evolving knowledge base. Research in the field is dominated by the causal rejection principle: the conflicting rule from the less preferred program is rejected in the case of conflict. Some drawbacks of the causal rejection principle (irrelevant updates, inconsistencies which cannot be solved according to the causal rejection principle, disagreement with other fields relevant for updates of nonmonotonic knowledge bases such as belief revision or preferences handling) are identified and discussed in the paper. The discussion of those drawbacks pointed out the role of assumptions and dependencies on assumptions for a careful attitude to the topic. A solution based on a dependency framework is presented and evaluated in the paper.

Keywords: multidimensional dynamic logic programming, nonmonotonic knowledge base, stable model semantics, belief revision, update, preference, dependency framework.

Introduction

Background. Multidimensional dynamic logic programming (MDyLP) (Alferes et al. 1998; Leite et al. 2001; Leite 2003; Alferes et al.2005; Banti et al. 2005) contributed to logic-based knowledge representation research by focusing on dynamic aspects of knowledge. MDyLP can be considered as a formal model of (evolving) nonmonotonic knowledge bases (NMKB). Evolution of (incomplete) knowledge is a key to understanding of nonmonotonic reasoning. Therefore, a foundational research of the framework proposed by MDyLP is of interest also for other approaches to non-monotonic reasoning and knowledge representation.

Problem. However, there are some drawbacks of the approach that didn't attract a sufficient attention until now. They are connected to the causal rejection principle (CRP),[1] which dominates research in the field. We will identify and discuss

- irrelevant updates,
- inconsistencies which cannot be solved according to the causal rejection principle,
- disagreement with other approaches relevant for updates of NMKB (f.ex. with research in the fields of belief revision or preferences handling, see (Gärdenfors and Rott 1995; Delgrande et al. 2003) and many others).

Proposed solution. Non-monotonic (defeasible, default) assumptions play a crucial role in non-monotonic reasoning. Our analysis of drawbacks of CRP leads to an opinion that the attention should be moved from conflicts of rules to conflicts involving assumptions and dependencies on assumptions. A dependency framework is proposed in the paper. Rejection or insertion of rules is avoided; the framework is aiming at a coherent view on a (possibly incoherent) MDyLP[2] by *ignoring* some dependencies. Our approach is close to the spirit of answer set programming (thanks to the stress on sets of assumptions, see ([Konczak et al. 2004; Novák draft)).

The presented dependency framework evolved from our Kripkean semantics of MDyLP (Šefránek 2000; Šefránek 2004). It can be said that the Kripkean semantics served as a scaffold for the proposal of the dependency framework, which is simpler, more general, has better intuitive and computational properties. We believe that our approach[3] contributes to a shift from a discussion of examples and counterexamples to a principle-based approach (see also (Alferes et al.2005; Banti et al. 2005)) to the updates of NMKB topic.

Contributions. Main contributions of the paper are as follows:

- some drawbacks of MDyLP, which did not attract a sufficient attention until now are discussed,
- a dependency framework is proposed in which those drawbacks are overcome,
- a method of constructing a coherent semantic view on a set of dependencies is presented,

[1] If there is a conflict between the heads of rules then reject the less preferred rule.

[2] We use the shorthand MDyLP both for MDyL- programs and programming. It is hoped that this ambivalence do not cause problems.

[3] For a preliminary report see (Hpomola et al. 2005).

- properties of the semantics based on dependencies on assumptions are discussed.

Roadmap. First basic definitions relevant for understanding MDyLP are recapped. Then the dependency framework is introduced. After that we analyze solutions of conflicts between dependencies. Postulates specifying solutions of conflicts are introduced. Next, drawbacks caused by the principle of causal rejection are analyzed. Semantics of logic program updates based on the dependency framework is described. Finally, our approach is evaluated, contributions are summarized and open problems characterized.

Multidimensional dynamic logic programming

Let $\mathcal{A}$ be a set of atoms. The set of *literals* is defined as $Lit = \mathcal{A} \cup \{not\ A : A \in \mathcal{A}\}$. Literals of the form $not\ A$, where $A \in \mathcal{A}$ are called *subjective* (not is intended as default negation). Notation: $Subj = \{not\ A \mid A \in \mathcal{A}\}$. A convention: $not\ not\ A = A$. If X is a set of literals then $not\ X = \{not\ L \mid L \in X\}$.

A *rule* is each expression of the form $L \leftarrow L_1, \ldots, L_k$, where $k \geq 0$, L, L_i are literals. If r is a rule of the form as above, then L is denoted by $head(r)$ and $\{L_1, \ldots, L_k\}$ by $body(r)$. A finite set of rules is called *generalized logic program* (program hereafter).

The set of *conflicting literals* is defined as $CON = \{(L_1, L_2) \mid L_1 = not\ L_2\}$. Two rules r_1, r_2 are called conflicting, if $head(r_1)$ and $head(r_2)$ are conflicting literals. Notation: $r_1 \bowtie r_2$. A set of literals S is *consistent* if it does not contain a pair of conflicting literals, i.e. $(S \times S) \cap CON = \emptyset$. An *interpretation* is a consistent set of literals. A *total* interpretation is an interpretation I such that for each atom A either $A \in I$ or $not\ A \in I$. Let I be an interpretation. Then $I^- = I \cap Subj$.

A literal is *satisfied* in an interpretation I iff $L \in I$. A set of literals S is satisfied in I iff $S \subseteq I$.

The basic semantic concepts introduced by (Alferes et al. 1998; Leite et al. 2001; Leite 2003; Alferes et al.2005) are recapped below.

Definition 1 ((Alferes et al. 1998)) A total interpretation S is a *stable model* of a program P iff

$$S = least(P \cup S^-),$$

where $P \cup S^-$ is considered as a Horn theory and $least(P \cup S^-)$ is the least model of the theory. A program is *coherent* iff it has a stable model. □

Definition 2 ((Leite et al. 2001)) A *multidimensional dynamic logic program* (also *multiprogram* hereafter) is a pair $\mathcal{P} = (\Pi, G)$, where $G = (V, E)$ is an acyclic digraph, $|V| \geq 2$, and $\Pi = \{P_i : i \in V\}$ is a set of (generalized logic) programs.

We denote by $i \prec j$ that there is a path from i to j and $i \preceq j$ means that $i \prec j$ or $i = j$. If i and j are incomparable w.r.t. $\preceq$, it is denoted by $\|$. If $i \prec j$, we say that P_j is *more preferred* than P_i (we denote it by $P_i \prec P_j$ with a little abuse of $\prec$). □

If G is a path, we speak about *dynamic* logic program. If $\mathcal{P}$ is a multiprogram then $\mathcal{A}$ is the set of all atoms occurring in $\bigcup_{j \in V} P_j$ and for each $P_i \in \Pi$ an interpretation of P_i is a consistent subset of $\mathcal{A} \cup not\ \mathcal{A}$.

Definition 3 (Dynamic stable model, (Leite et al. 2001)) Let $\mathcal{P}$ be a multiprogram. A total interpretation M is called *dynamic stable model* of $\mathcal{P}$ iff

$$M = least(\bigcup_{i \in V} P_i \setminus Rej(\mathcal{P}, M)) \cup Def(\mathcal{P}, M)), \qquad (1)$$

where $Rej(\mathcal{P}, M) = \{r \in P_i \mid \exists r' \in P_j\ (i \prec j, r \bowtie r', M \models body(r'))\}$ and $Def(\mathcal{P}, M) = \{not\ A \mid \neg\exists r \in \bigcup_{i \in V} P_i\ (A = head(r), M \models body(r))\}$. □

Refined dynamic stable model is defined in (Alferes et al.2005) similarly, with only a little difference – condition $i \preceq j$ is used in the definition of rejected rules instead of $i \prec j$. We will use for that modified concept notation $Rej^R(\mathcal{P}, M)$). The set of all refined dynamic stable models of $\mathcal{P}$ is denoted by $RDSM(\mathcal{P})$. Troubles with tautological and cyclic updates of dynamic logic programs are overcome in refined semantics. However, the refined semantics for the general case of multiprograms is not known. The well supported semantics of multiprograms is defined in (Banti et al. 2005), in order to improve the behaviour of semantics based on CRP. The well supported semantics for MDyLP coincides with the refined one on dynamic logic programs.

We will use refined semantics in the analysis of examples, which contain elementary dynamic logic programs of the form $\langle P, U \rangle$, where $P \prec U$. P can be viewed as an original program and U as an updating program. We have chosen the most simple case in order to be comprehensible, but the analysis of examples is relevant for CRP in general.

Dependency framework

A specification of a semantics of multiprograms in terms of dependencies and assumptions is presented in this paper. Our approach is aiming at overcoming some drawbacks of semantics based on the causal rejection principle. The basic features of proposed dependency framework are described in this section.[4]

Two notions of dependency relation are going to be defined. First, a more general one and second, a dependency relation generated by a program.

Definition 4 (Dependency relation) A *dependency relation* is a set of pairs $\{(L, W) \mid L \in Lit, W \subseteq Lit, L \notin W\}$.

A literal L *depends* on a set of literals W, $L \notin W$, *with respect to a program P* ($L \ll_P W$) iff there is a sequence of rules $\langle r_1, \ldots, r_k \rangle$, $k \geq 1$, and

- $head(r_k) = L$,

[4] We could choose between two policies. The first one begins with a motivation and then provides a formal framework. The second one, to the contrary, uses the formal framework for an analysis of motivating examples. We prefer the second one in this paper. Motivation for our framework is detailed later.

- $W \models body(r_1)$,
- for each i, $1 \leq i < k$, $W \cup \{head(r_1), \ldots, head(r_i)\} \models body(r_{i+1})$.

It is said that the dependency relation $\ll_P$ is *generated* by the program P. □

Notice that a literal cannot depend on itself (also in a context of other literals). The dependency relation $\ll_P$ cannot be identified with derivability from P. If $P = \{a \leftarrow b\}$ then $a \ll_P \{b\}$, but a is not derivable from P.

Definition 5 (Closure property) A *closure operator Cl* assigns the set of all pairs $\{(L, W) \mid L \ll W \vee (\exists U\ (L \ll U \wedge \forall L' \in U \setminus W\ (L' \ll W)))\}$ to a dependency relation $\ll$.

A dependency relation $\ll$ has the closure property iff $\mathrm{Cl}(\ll) = \ll$. □

Notice that $\ll_P$ has the closure property.

Proposition 6 *Let P be a program. Then* $\mathrm{Cl}(\ll_P) = \ll_P$.

Proof sketch: Suppose that $L \ll_P U, \forall L' \in U \setminus W\ L' \ll_P W$. Consider a sequence of rules, satisfying the conditions of Definition 4 such that each $L' \in U \setminus W$ is derived from W. Concatenate a sequence deriving L from U. We have proved $L \ll_P W$, □

Example 7 Let P be

$$\begin{array}{lcl} \{a & \leftarrow & not\ b \\ b & \leftarrow & not\ a \\ c & \leftarrow & a \\ d & \leftarrow & b\} \end{array}$$

It holds that $a \ll_P \{not\ b\}$, $b \ll_P \{not\ a\}$, $c \ll_P \{a\}$, $d \ll_P \{b\}$, $c \ll_P \{not\ b\}$, $d \ll_P \{not\ a\}$, but also $c \ll_P \{not\ b, not\ d\}$, $d \ll_P \{not\ a, not\ c\}$ or $d \ll_P \{not\ a, not\ c, not\ d\}$ etc. We can see that some dependencies of L on W are of crucial interest, namely those, where $W \subseteq Subj$ and W generates (or contributes to a generation) of a stable model. □

We have seen in Example 7 that dependencies on subjective literals are crucial from the viewpoint of stable semantics. Therefore the role of (default) assumptions is emphasized.

Definition 8 (SSOA, TSSOA) $Ay \subseteq Subj$ is called a *sound set of assumptions* (SSOA) with respect to the dependency relation $\ll$ iff the set

$$Cn_{\ll}(Ay) = \{L \in Lit \mid L \ll Ay\} \cup Ay$$

is non-empty and consistent.

It is said that Ay, a SSOA, is *total* (TSSOA) iff for each $A \in \mathcal{A}$ holds either $A \in Cn_{\ll}(Ay)$ or $not\ A \in Cn_{\ll}(Ay)$. □

Example 9 Let $\ll$ be $\{(a, \{not\ c\}), (not\ a, \{not\ d\})\}$. There is no TSSOA w.r.t. $\ll$. Both $\{not\ c\}$ and $\{not\ d\}$ are SSOAs w.r.t. $\ll$.

An important remark: updates of conflicting multiprograms can be viewed as a construction of TSSOAs from SSOAs while ignoring some dependencies. □

Next theorem shows that the dependency framework is relevant from the stable (answer set) semantics point of view.

Theorem 10 *X is a TSSOA w.r.t. $\ll_P$ iff $Cn_{\ll_P}(X)$ is a stable model of P.*

Let S be a stable model of P. Then there is $X \subseteq Subj$, a TSSOA w.r.t. $\ll_P$ s.t. $S = Cn_{\ll_P}(X)$. □

Semantics based on assumptions and dependencies. Consider two mappings Σ, Σ'. Let Σ assigns to each program P the set of all its stable models. Let Σ' assigns to each program P the set of all TSSOAs w.r.t. $\ll_P$. We have seen in Theorem 10 that (the semantics characterized by) Σ is equivalent to (the semantics characterized by) Σ'. So, we can speak about a semantics based on assumptions and dependencies.

Dependencies in a multiprogram. We intend to use our framework for handling conflicting dependencies in a multiprogram. Note that dependencies in a multiprogram are well defined:

Proposition 11 *Let $\mathcal{P}$ be a multiprogram. Then $\ll_{\bigcup_{i \in V} P_i}$ is well defined. It holds*

$$\bigcup_{i \in V} \ll_{P_i} \subseteq \ll_{\bigcup_{i \in V} P_i},$$

but the converse inclusion does not hold. □

We are going to introduce an approach (and some concepts) which enables to handle conflicts involving dependencies and assumptions.

Conflicts, solutions, postulates

There are essentially two possible sources of incoherence in a (multi)program $\mathcal{P}$:

1. two conflicting literals depend on a set of literals;
2. a literal L depends on a set of literals W and $not\ L \in W$; dependencies on subjective literals are crucial in our framework, so we are focused only on the case that L is an atom.

Definition 12 Let $\ll$ be a dependency relation. It is said that $\ll$ *contains a conflict* C (where $C \subseteq \ll$) iff for some $A \in \mathcal{A}$ is $C = \{(A, Y), (not\ A, Y) \mid Y \subseteq Subj\}$ or $C = \{(A, Y) \mid Y \subseteq Subj, not\ A \in Y\}$. □

We are interested in solving conflicts between dependencies generated by multiprograms. It is assumed here that the preference relation on programs is preserved also for corresponding dependency relations, i.e. if P_i is more preferred than P_j then $\ll_{P_i}$ is more preferred than $\ll_{P_j}$.

Definition 13 Let $\mathcal{P}$ be a multiprogram and $C \subseteq \ll_{\bigcup_{i \in V}}$ be a conflict.

It is said that a set of dependencies D is a *solution* of the conflict C iff $C \not\subseteq Cl(\bigcup_{i \in V} \ll_{P_i} \setminus D)$. D is called *minimal* iff there is no proper subset of D which is a solution of C.

Let D and D' be minimal solutions of C. It is said that D' is *more suitable* than D iff there is an injection $\kappa : D' \longrightarrow D$ such that $\forall d \in D'$ $((d \in \ll_{P_j} \wedge \kappa(d) \in \ll_{P_i}) \Rightarrow j \preceq i)$. If the cardinality of D and D' is the same then for at least one $d \in D'$ holds $j \prec i$.

A minimal solution D of a conflict C is called *good solution* iff there is no more suitable solution of C. □

Some comments to the definition: A solution of a conflict is focused on dependencies generated by a single program. Only elementary pieces of a chain of dependencies (dependencies from some $\ll_{P_i}$, $i \in V$) are ignored.[5] It is more suitable to ignore less preferred dependencies. Good solutions consist of minimally preferred dependencies and the principle of minimal change is obeyed. Note that in general there are different good solutions of a conflict. Another approach to alternative solutions of a conflict is presented in (Šefránek 2006a).

Our attention is focused also on conflicting assumptions. First a notion of falsified assumptions is defined. It is used in the analysis of Example 18.

Definition 14 An assumption $not\ A$, where $A \in \mathcal{A}$, is *falsified* in a dependency relation $\ll$ iff $A \ll \emptyset$, $not\ A \not\ll \emptyset$ and $\emptyset$ is a SSOA w.r.t. $\ll$.

A set of assumptions $Ay \subseteq Subj$ is falsified in $\ll$ iff it contains a literal falsified in $\ll$. □

We can now proceed to postulates. Our goal is to transfer the discussion about problems with logic program updates from examples and counterexamples to some general principles.[6] Our postulates specify which dependencies should be ignored (i.e. not considered when creating a coherent semantic view on a set of dependencies). We believe that the postulates are clear and can be continually improved, if needed (note that a translation of many formalisms to the dependency framework is possible). The construction of a coherent semantic view on a set of dependencies presented in this paper satisfies the postulates.

It is supposed in Postulates below that

- a dependency relation $\ll$ is given,
- a finite set $\mathcal{D} = \{\ll_1, \ldots, \ll_k\}$, where $\ll_i \subset \ll$, is given,
- an acyclic, transitive and irreflexive preference relation ρ on $\mathcal{D}$ is defined.

[5]Our approach does not reject or insert some rules. Its ambition is to provide a coherent view on a (possibly incoherent) MDyLP (NMKB) by *ignoring* some dependencies.

[6]An allied approach is presented in (Alferes et al.2005; Banti et al. 2005) as regards irrelevant updates (according to our terminology). We discuss the case of irrelevant updates in (Šefránek 2006b).

If $\ll_i, \ll_j \in \mathcal{D}$ and $\ll_i\ \rho \ll_j$, it is said that $\ll_j$ ($\ll_i$) is more (less) preferred as $\ll_i$ ($\ll_j$). Similarly, if $d \in \ll_j$ and $d' \in \ll_i$, it is said that d (d') is more (less) preferred than d' (d).

Postulate 1 *Let Ay be a set of assumptions falsified in $\ll$. Then all dependencies of the form $L \ll W$, where $Ay \subseteq W$, are ignored.*

Note that Postulate 2 is a generalization of two special cases (for two kinds of conflicts from Definition 12. The postulate corresponds partly to the CRP, but it also extends the possibilities of solving conflicts. Moreover, the general formulation of Postulate 2 enables also other kinds of solutions of a conflict of the form $\{L_1 \ll W, L_2 \ll W\}$ than those enabled by CRP, see Example 20 (we emphasize that bridging the gap between belief revision community and dynamic logic programming community is enabled as a consequence). Postulate 2 admits non-determinism: alternative good solutions of a conflict are possible. This non-determinism is an appropriate one in the context of stable model semantics.

Postulate 2 *Let C be a conflict. If there is a set S of good solutions of C, then one $D \in S$ is selected and ignored.*

Finally, the last postulate is introduced below. The postulate enables to distinguish more preferred TSSOAs (stable models) in order to be able solve the problems in the style of prioritized logic programming. Of course, there are different motivations behind (logic program) updates and preferential reasoning. On the other hand, *multidimensional* dynamic logic programming is aiming at distinguishing various kinds of preferences (w.r.t. time, agents, hierarchical instances, domains of knowledge, hierarchy of power, viewpoint etc. etc.) and we expect that it should (even must) be able also to select more preferred stable models in the style of prioritized logic programming. May be, different strategies of conflict solving along different dimensions in a multiprogram are needed (for a more detailed discussion see Example 21). We are proposing a first step toward that goal below. A preference relation on sets of assumptions is defined.

Example 15 If a less preferred program is $P = \{a \leftarrow not\ b\}$ and a more preferred program, its update, is $U = \{b \leftarrow not\ a\}$, we get two standard TSSOAs: $Ay_1 = \{not\ a\}$ and $Ay_2 = \{not\ b\}$ (and corresponding standard stable models).

However, we can consider Ay_1 as more preferred than Ay_2. An example from (Delgrande et al. 2003) is discussed in Example 21. □

We accept that sometimes it is useful also discriminate a preference relation on (sets of) assumptions. It enables to gain capabilities of logic programming with preferences in multidimensional dynamic logic programming, see Example 21.

Definition 16 Let be $S(not\ A) = \{i \in V \mid \exists r \in P_i\ not\ A \in body(r)\}$.

Let be $L, L' \in Subj$. The assumption L is *preferred at least as* the assumption L' iff for each maximal $i \in S(L')$ and each maximal $j \in S(L)$ holds either $i \preceq j$ or $i \parallel j$.

L is *more preferred* than L' iff L is preferred at least as L' and for at least one pair i, j holds $i \prec j$.

A set of subjective literals S is more preferred than the set of subjective literals S' iff each $L \in S \setminus S'$ is preferred at least as each $L' \in S' \setminus S$ and there is an $L \in S \setminus S'$ more preferred as each $L' \in S' \setminus S$. □

Definition 17 Let $\mathcal{P} = (\Pi, G)$ be a multiprogram, $G = (V, E)$, let be $i, s, t \in V$.

It is said that a set of assumptions Ay is *falsified w.r.t.* a more preferred set of assumptions X and a dependency relation $View \subseteq \ll_{\bigcup_{i \preceq s} P_i}$ iff

- X is a TSSOA w.r.t. $View$,
- there are $L_1 \in X$ and $L_2 \in Ay$ such that $not\ L_2 \in Cn_{View}(X)$,
- X is not falsified and it is also not falsified w.r.t. some Y and some $View' \subseteq \ll_{\bigcup_{i \preceq t} P_i}$, where $s \prec t$, $View \subseteq View'$, Y is a TSSOA w.r.t. $View'$. □

We will use a shorthand "wrt-falsified" supposing that the corresponding more preferred set of assumptions and the corresponding dependency relation are implicitly clear or it does not matter what set of assumptions and what dependency relation are considered.

Finally, the last postulate is introduced for the case, if a preference relation on sets of assumptions is defined.

Postulate 3 *Let σ be an acyclic, transitive and irreflexive preference relation on sets of assumptions. If Ay is wrt-falsified, then all dependencies of the form $L \ll W$, where $Ay \subseteq W$, are ignored.*

Semantics of multiprograms based on assumptions and dependencies. We outline now a preliminary characterization of semantics of a multiprogram (in our dependency framework). The semantics is characterized by a mapping from a multiprogram $\mathcal{P}$ to a set of pairs of the form $(Ay, View)$, where $View \subseteq \ll_{\bigcup_{i \in V} P_i}$ and Ay is a TSSOA w.r.t. $View$. It means that our semantics is looking for a conflict-free subset of $\ll_{\bigcup_{i \in V} P_i}$ and for a reasonable set of assumptions (while Postulates 1 – 3 are satisfied). Technical details of the semantics are motivated and elaborated below.

Semantics based on rejection of rules

We will point out some drawbacks of semantics based on rejection of rules in this section. Only examples of the form $\langle P, U \rangle$, where $P \prec U$, are used (with the only exception). Refined dynamic stable model semantics is used in the analysis of the examples, but our arguments are applicable to any semantics focused only on conflicts between the heads of rules (hence, also to the general multiprograms).

We are proceeding to an example illustrating problem of *irrelevant* updates.

Example 18 ((Eiter et al. 2002))

$$\begin{array}{rcl} P = it\_is\_cloudy & \leftarrow & it\_is\_raining \\ it\_is\_raining & \leftarrow & \\ U = not\ it\_is\_raining & \leftarrow & not\ it\_is\_cloudy \end{array}$$

$RDSM(\langle P, U \rangle) = \{\{not\ it\_is\_raining, not\ it\_is\_cloudy\}, \{it\_is\_raining, it\_is\_cloudy\}\}$.

Notice that $it\_is\_raining \ll_{P \cup U} \{not\ it\_is\_cloudy\}$, $not\ it\_is\_raining \ll_{P \cup U} \{not\ it\_is\_cloudy\}$, but the assumption $not\ it\_is\_cloudy$ is *falsified* in $\ll_{P \cup U}$ because of $it\_is\_cloudy \ll_{P \cup U} \emptyset$. Information given by U do not override the information of P (which is based on the empty set of assumptions). The only TSSOA w.r.t. $\ll_{P \cup U}$ is $\emptyset$. Consider Postulate 1.

Irrelevant updates are analyzed in a more detail in (Šefránek 2006b).

In general, troubles with all semantics based on rejection of rules are caused also by a too free choice of an interpretation involved in the fixpoint condition (1). We mean an interpretation containing falsified assumptions (default negations). Interpretations generated by falsified assumptions do not provide an appropriate candidate for a semantic characterization of a multiprogram. A remark is in place: conflicts involving assumptions did not attract an adequate attention until now. □

Incompleteness of updates based on CRP is illustrated by the next example.

Example 19 [7]

$$\begin{array}{rcl} P & = & \{obedient \leftarrow punish\} \\ U & = & \{punish \leftarrow not\ obedient\}. \end{array}$$

There is no conflict between heads of rules. Hence, no rule can be rejected and the meaning of $P \cup U$ cannot be updated according to CRP. However, there is a kind of conflict between both programs. $P \cup U$ has no stable model, but both P and U are coherent.

There is no clear reason why to solve conflicts between heads of rules and not to solve other conflicts. Also conflicts caused by some assumptions and conflicts between dependencies on some assumptions are relevant (for nonmonotonic reasoning).

The third postulate of (Katsuno and Mendelzon 1991), 3[KM] hereafter (if both ψ and μ are satisfiable then $\psi \diamond \mu$ is also satisfiable, where ψ is a knowledge base, μ is an update and $\diamond$ is an update operation) cannot be satisfied by a semantics focused only on rejection of rules with conflicting heads. □

We propose to ignore less preferred dependencies in order to cut off a dependency of an atom A on the assumptions containing $not\ A$.[8]

[7] This is a variant of an example from (Pereira and Pinto 2005).

[8] We do not exclude that reasoning based on reduction ad absurd (Pereira and Pinto 2005) is useful for knowledge based systems. However, we prefer here constructive mode of reasoning and a kind of compatibility with answer set programming paradigm.

Another of the drawbacks is that CRP is not able to recognize alternative solutions of a given inconsistency (note a striking difference w.r.t. the belief revision research).

Example 20 [9]

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

$Rej^R(\langle P, U\rangle, \{not\ \ a, b\}) = \{a\ \ \leftarrow\}$ and $RDSM(\langle P, U\rangle) = \{\{not\ \ a, b\}\}$. However, it is not clear why $a \leftarrow$ can be rejected and $b \leftarrow$ cannot be rejected. There are two (if we respect the preference relation) maximal coherent subsets of incoherent $P \cup U$ and two corresponding stable models – besides $\{not\ \ a, b\}$ also $\{not\ b, a\}$.

Alternative solutions of this conflict using *nonmonotonic integrity constraints* are proposed in (Šefránek 2006a). However, notice that the more general formulation of Postulate 2 in present paper enables alternative solutions, too. □

Let us proceed to logic programs with preferences. Note that there is a trivial correspondence between prioritized logic programs and multidimensional dynamic logic programs. Consider first static preferences (on rules). Let a MDyLP $\mathcal{P}$ be given. If $P_i \prec P_j$ then for each $r \in P_i$ and each $r' \in P_j$ holds $r \prec r'$. Otherwise, rules are incomparable.

Conversely, let a prioritized logic program be given as a pair $(\{r_i \mid i \in I\}, \prec)$. The corresponding MDyLP we obtain as a set of programs (singletons) $P_i = \{r_i\}$ preserving $\prec$: $P_i \prec P_j$ iff $r_i \prec r_j$.

If preference relation is a dynamic one, then a dynamic preference relation on programs is needed. A possibility of such extension of MDyLP is supposed (f.ex. in (Alferes et al. 1998)), but we are not aware of a realization of the possibility. However, it is feasible and straightforward.

Example 21 [10] $P_1 = \{c \leftarrow not\ \ b\}, P_2 = \{a \leftarrow\}, P_3 = \{b \leftarrow a, not\ c\}$; $P_1 \prec P_3$, $P_1 \parallel P_2 \parallel P_3$.
$RDSM(\langle P, U\rangle) = \{\{a, b, not\ c\}, \{a, c, not\ b\}\}$. There are no conflicting rules in this multiprogram, hence no rule can be rejected.

The more preferred model (not only) according to (Delgrande et al. 2003) is $\{a, b, not\ c\}$.

Suppose now that $Ay_1 = \{not\ c\}$ is more preferred than $Ay_2 = \{not\ b\}$. Hence, "it is not known b" seems not to be a reasonable assumption and we can consider it as falsified by the *more preferred* set of assumptions.

In technical terms introduced in Definition 16: $S(not\ \ b) = \{1\}$ and $S(not\ \ c) = \{3\}$, hence $not\ \ c$ is more preferred than $not\ b$. Further, $\{not\ c\}$ is a TSSOA w.r.t. $\ll_{P_1 \cup P_2 \cup P_3}$, it is neither falsified nor wrt-falsified and finally, $b \in Cn_{\ll_{P_1 \cup P_2 \cup P_3}}(\{not\ c\})$. Therefore, $\{not\ b\}$ is falsified w.r.t. $\{not\ c\}$ and $\ll_{P_1 \cup P_2 \cup P_3}$.

[9] This example is due to Martin Baláž. More thorough discussion of this Example is in (Šefránek 2006a).

[10] This is an adapted version of an example from (Delgrande et al. 2003). For a more thorough discussion see (Šefránek 2006a).

There is an intuitive difference between updates and preferences (see (Alferes and Pereira 2000)). However, the *multidimensional* approach of MDyLP should represent also preferential reasoning. May be, different strategies for different dimensions are needed. Moreover, there are some problems with very notion of update, if default negations are allowed (even in heads of rules). We will devote a future research to the topic of updates and revisions of NMKBs. □

A coherent semantic view on a set of dependencies

We are going to define a reasonable semantic view on a set of dependencies $\ll$. The view will be a set of dependencies $View$, which is

- coherent (in the sense defined below),
- a subset of $\ll$.

Definition 22 (Coherent dependency relation) A dependency relation $\ll$ is called *coherent* iff there is a TSSOA w.r.t. $\ll$. A dependency relation is called *incoherent* iff it is not a coherent one. □

Consequence 23 *Let a dependency relation $\ll$ be coherent. Let Ay be a TSSOA w.r.t. $\ll$. Then there is no $C \subseteq \ll$ s.t. $C = \{A \ll Ay, not\ A \ll Ay \mid A \in \mathcal{A}\}$ or $C = \{A \ll Ay \mid A \in \mathcal{A} \wedge not\ A \in Ay\}$.*

Proof: $Cn_{\ll}(Ay)$ is consistent. □

In general, $\ll_{\bigcup_{i \in V} P_i}$ can be incoherent. Our approach to semantics of MDyLP is focused on looking for assumptions which can serve as TSSOA w.r.t. a corresponding subset of given dependency relation $\ll_{\bigcup_{i \in V} P_i}$. Note that more reasonable semantic views on a set of dependencies are possible (of course, this can be expected – stable model semantics is at the background of our constructions).

There are different TSSOAs w.r.t. different subsets of $\ll_{P \cup U}$ in next example.

Example 24

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

$Ay_1 = \{not\ \ a\}, Ay_2 = \{not\ \ b\}$, $View_1 = Cl((\ll_P \cup \ll_U) \setminus \{not\ b \ll_P \{not\ a\}\})$, $View_2 = Cl((\ll_P \cup \ll_U) \setminus \{not\ a \ll_P \{not\ b\}\})$.

$View_1$ is coherent (Ay_1 is a TSSOA w.r.t. $View_1$) and also $View_2$ is coherent (Ay_2 is a TSSOA w.r.t. $View_2$) □

We are aiming to construct all possible dependency (sub)relations which are coherent (w.r.t. a TSSOA), see Example 24. Note that also coherent dependency relations can be revised (until we obtain all possible pairs of the form $(Z, View)$, where Z is a TSSOA w.r.t. $View$. Hence, we are aiming to solve conflicts connected to some subsets of $Subj$.

Definition 25 If $Y \subseteq \mathit{Subj}$ is fixed and $C \subseteq \ll$ be a conflict (i.e. $C = \{(A, Y), (not\ A, Y) \mid Y \subseteq \mathit{Subj}\}$ or $C = \{(A, Y) \mid Y \subseteq \mathit{Subj}, not\ A \in Y\}$), then it is said that C is a conflict *of* Y. □

If we are looking for coherent subsets of a dependency relation, we are focused on solving conflicts of candidates for TSSOAs. Pairs of assumptions and dependencies are relevant for our semantics of dynamic logic programs.

We will describe a construction of a coherent dependency relation from an incoherent $\ll_{\bigcup_{i \in V} P_i}$. The constructed relation represents – in a sense – a coherent semantic view on an incoherent multiprogram. We present a nondeterministic algorithm which iteratively finds preferred minimal solutions of conflicts in $\ll_{\bigcup_{i \in V}}$ and simultaneously constructs TSSOAs, see Figure 1. The algorithm iteratively solves conflicts and modifies assumptions. Observe that Postulates of Section *Conflicts, solutions, postulates* are obeyed in the algorithm.

Let describe the behaviour of the algorithm. It is assumed that there is a set Ω containing pairs of the form (Z, View), where Z are assumptions and View is a dependency relation. Z is neither falsified in $\ll_{\bigcup_{i \in V} P_i}$ (nor falsified w.r.t. a more preferred set of assumptions, if a preference relation on sets of assumptions is accepted). Initially, View is $\ll_{\bigcup_{i \in V} P_i}$.

If Z is a TSSOA w.r.t. View, the task is done. Otherwise, conflicts are solved in a REPEAT – UNTIL cycle. First a set of all good solutions of a conflict is identified. For each good solution a coherent view View_i is computed. Only the first computed view is processed, alternative views are collected in Ω.

The value of variable i is 0 after the *for each*-cycle, if there is no solution of the conflict and the computation failed. If Z is not a TSSOA and the conflict has been solved in the current run of the REPEAT-UNTIL cycle and if there are no more conflicts in current View^T then $Cn_{\mathit{View}^T}(Z)$ is consistent but not complete and the cycle is finished in the next run because of $i = 0$. Otherwise, Z is a TSSOA w.r.t. the computed View^T.

We have to overcome the final complication before proceeding to the semantic characterization of multiprograms. It is proposed to accept minimal sets of assumptions. Let a multiprogram $\mathcal{P}$ be given. If Y and Y' are TSSOAs w.r.t. $\mathit{View}, \mathit{View}' \subseteq \ll_{\bigcup_{i \in V} P_i}$, respectively, and $Y \subset Y'$, then only Y is selected as a proper semantic characterization of $\mathcal{P}$. See next Example as an illustration.

Example 26 [11]

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

$Ay_1 = \{not\ c\}$, $a \ll_{P \cup U} Ay_1, b \ll_{P \cup U} Ay_1$. $Ay_2 = \{not\ c, not\ b\}$, $\mathit{View}_2 = \mathit{Cl}((\ll_P \cup \ll_U) \setminus \{a \ll_P \{not\ c\}\}$.

[11] This Example is proposed by J.Leite.

INPUT: a pair (Z, View) from Ω, where $Z \subseteq \mathit{Subj}$, $\mathit{View} \subseteq \ll_{\bigcup_{i \in V} P_i}$
OUTPUT: a pair (Z, View^T), where Z is a TSSOA w.r.t. View^T or the decision that it is not possible to construct an TSSOA from Z
begin
  if Z is a TSSOA w.r.t. View **then** RETURN (Z, View)
  fi
  $\mathit{View}^T := \mathit{View}$
  REPEAT
    if View^T contains a conflict C of Z **then**
      SELECT the set Π of all D, good solutions of C
    fi
    $i := 0$;
    for each $D \in \Pi$ **do**
      $i := i + 1$; $\mathit{View}_i^T := \mathit{Cl}(\mathit{View}^T \setminus D)$
      if $i > 1$ **then** $\Omega := \Omega \cup (Z_i^T, \mathit{View}_i^T)$ **fi**;
      $\mathit{View}^T := \mathit{View}_1^T$
    od
    if $i = 0$ **then** FAILURE := true
      else FAILURE := false **fi**
  UNTIL Z is a TSSOA w.r.t. View^T or FAILURE
  if not FAILURE **then** RETURN (Z^T, View^T)
    else RETURN FAILURE **fi**
end

Figure 1: Non-deterministic algorithm removing conflicts and computing TSSOAs

There are two TSSOAs: Ay_1 w.r.t. $\ll_{P \cup U}$ and Ay_2 w.r.t. View_2. The second one contains more non-monotonic assumptions and we prefer to accept minimal sets of assumptions. □

Definition 27 Let $\mathcal{P}$ be a multiprogram. Let $\mathit{View} \subseteq \ll_{\bigcup_{i \in V} P_i}$ be coherent and Z be a TSSOA w.r.t. View.

It is said that Z is a good set of assumptions (GSOA) w.r.t. View iff there is no TSSOA Z' w.r.t. a $\mathit{View}' \subseteq \ll_{\bigcup_{i \in V} P_i}$ s.t. $Z' \subset Z$. □

Definition 28 (Semantics of multiprograms) Semantics of multiprograms is a mapping Σ from multiprograms to sets of pairs of the form (Z, View), where View is a coherent subset of $\ll_{\bigcup_{i \in V} P_i}$ w.r.t. assumptions Z and Z is a GSOA w.r.t. View. □

We can create canonical programs[12] determined by all GSOAs of a multiprogram.

Definition 29 Let $\{Z_1, \ldots, Z_k\}$ be all GSOAs w.r.t. $\{\mathit{View}_1, \ldots, \mathit{View}_k\}$, respectively, of a multiprogram $\mathcal{P}$. Then a program of the form $\{L \leftarrow Z_i \mid L \in Cn_{\mathit{View}_i}(Z_i), L \notin Z_i, i = 1, \ldots, k\}$ is called *canonical*

[12] For similar construction see (Novák draft; Šefránek 2004; Šefránek 2000).

program representing a (maximally) coherent semantic view on $\mathcal{P}$. □

Theorem 30 *Let Q be a canonical program representing a multiprogram $\mathcal{P}$. The set of all stable models of Q coincide with the set of sets $\{Cn_{View_i}(Z_i) \mid Z_i$ is a GSOA w.r.t. $View_i\}$*

Evaluation

Some important properties of the semantics based on dependency framework are presented in this section.

We consider 3[KM] as a criterion of completeness of our approach: if all programs from a multiprogram are coherent then a coherent view (on the dependency relation generated by the programs of multiprogram) is required. Solution of all conflicts generated by interactions of programs seems to be a reasonable minimal condition for maintaining coherence of multiprograms. Incoherence of a single program is a problem of another kind than solution of conflicts between programs.

An analogy of 3[KM] is satisfied in our dependency framework. We express it first for the simplest case.

Theorem 31 *Let $\langle P, U\rangle$ be a multiprogram. If $\ll_P$ and $\ll_U$ are coherent then there is also a coherent dependency relation $View \subseteq \ll_{P\cup U}$.*

Proof: Suppose that there is no Ay which is a TSSOA w.r.t. $\ll_{P\cup U}$. It means that for each Ay is $Cn_{\ll_{P\cup U}}(Ay)$ inconsistent or incomplete. Incompleteness is excluded if TSSOAs w.r.t.$\ll_U$ or $\ll_P$ are considered.

Consider Ay which is a TSSOA w.r.t. $\ll_U$ and it is not falsified. We supposed that there is a conflict C contained in $\ll_{P\cup U}$, i.e. $A \ll_{P\cup U} Ay, not\ A \ll_{P\cup U} Ay$ or $A \ll_{P\cup U} Ay, not\ A \in Ay$. Of course, the conflict is not caused by $\ll_U$, so there is $D \subseteq \ll_P$ such that $Cl(\ll_{P\cup U} \setminus D)$ does not contain C.

If Ay is falsified: let be $X = \{A \in \mathcal{A} \mid A \ll_P \emptyset, not\ A \in Ay\}$. It holds for each TSSOA Bss w.r.t. $\ll_P$ that $X \subseteq Cn_{\ll_P}(Bss)$. If Bss is falsified w.r.t. a set of assumptions Y and $\ll_{P\cup U}$ then there is nothing to prove (Y is not falsified and it is a TSSOA w.r.t. $\ll_U$, the situation can be reduced to that of the previous paragraph).

Hence, assume that Bss is not falsified w.r.t. to some set of assumptions and $\ll_{P\cup U}$. Therefore, only conflicts of the form $\{A \ll_{P\cup U} Bss, not\ A \ll_{P\cup U} Bss\}$ have to be solved (and there is a good solution of such conflicts).

Finally, conflicts removing can be repeated until a coherent subset is reached. □

Consequence 32 *Let $\mathcal{P} = \langle \mathcal{P}_\infty, \dots, \mathcal{P}_\|\rangle$ be a dynamic logic program. If each $\ll_{P_i}$ is coherent then there is also a coherent dependency relation $View \subseteq \ll_{\bigcup_j P_j}$.*

Conflicts among incomparable programs are not solvable in a natural way, so for the general case of multiprograms we accept also the condition that the union of incomparable programs is coherent.

Theorem 33 *Let $\mathcal{P} = (\Pi, G)$ be a multiprogram. Let be $\ll_{P_i}$ coherent for each $i \in V$. Let $\mathcal{I}$ be the set of all incomparable programs in $\mathcal{P}$ (i.e., for each $P_i, P_j \in \mathcal{I}, i \neq j$ holds $i \parallel j$), let be $N = \{i \in V \mid P_i \in \mathcal{I}\}$.*

If $\ll_{\bigcup_{i\in N} P_i}$ is coherent, then there is also a coherent dependency relation $View \subseteq \ll_{\bigcup_{j\in V} P_j}$.

Our framework is immune w.r.t. tautological updates.

Proposition 34 (Tautological updates) *Let P be a program, τ be a tautology (i.e. $head(\tau) \in body(\tau)$) and $U = \{\tau\}$. Then each GSOA w.r.t. $\ll_{P\cup U}$ is a GSOA w.r.t. $\ll_P$.*

Proof is trivial - U does not generate a new dependency. □

Consequence 35 *Let $\mathcal{P}$ be a multiprogram (some of the programs in Π may be empty). Let for each $i \in V$ there is a (possibly empty) set of tautologies T_i. Let $\Sigma(\mathcal{P})$ be the set of all GSOAs w.r.t. corresponding $View \subseteq \ll_{\bigcup_{i\in V} P_i}$.*

Then the set of all GSOAs w.r.t. the corresponding $View \subseteq \ll_{\bigcup_{i\in V} P_i \cup T_i}$ is precisely $\Sigma(\mathcal{P})$.

Irrelevant updates (and unsupported cyclic updates, as a special case) are avoided in our semantics.

Rejection of conflicting rules (according to the dynamic stable model semantics) is satisfied in the dependency framework.

Proposition 36 *If $r \in Rej(\mathcal{P}, M)$ then there is a good solution of conflict $C = \{head(r) \ll_{P_i} M, not\ head(r) \ll_{P_j} M\}$, where $i \prec j$.*

Conclusions, discussion, open problems

We summarize main contributions of the paper as follows:

- some drawbacks of CRP, which did not attract a sufficient attention until now, were discussed,
- a step toward bridging the gap between the research in dynamic logic programming on the one hand and belief revision or preferences handling on the other hand is done,
- a dependency framework (close in the spirit to stable model semantics) has been developed as a basis for an analysis of conflicts contained in a MDyLP and also for their solution,
- a semantics of MDyLP based on the dependency framework has been developed,
- a set of postulates governing solution of conflicts is proposed,
- the proposed semantics is immune w.r.t. tautological, cyclic and irrelevant updates, it is complete (w.r.t 3[KM]), rejection of conflicting rules is satisfied,
- the semantics uses a construction of a coherent semantic view on a set of dependencies (a non-deterministic algorithm is presented); the construction respects postulates expressed in the paper; the semantics avoids drawbacks analyzed in the paper.

Remarks of technical nature: Only generalized logic programs are considered in the paper in order to support a simple discussion of refined dynamic stable semantics (Alferes et al.2005). However, an extension to generalized extended logic programs (GELP) is straightforward and will be presented in a forthcoming paper. Examples with pairs of programs P and U, where U is more preferred than P, are used in order to be as clear and simple as possible. However, the construction is applicable (and results hold) for the general case.

Presented approach opens some problems and topics for our ongoing or future research.

At the first place we would like to mention the problem of updates of NMKB. Second postulate of (Katsuno and Mendelzon 1991) cannot be taken literally because of the presence of default (nonmonotonic) assumptions in NMKB. We believe that our dependency framework represents a good starting point for better understanding of updates of NMKB. Similarly, the relation of updates and revisions in the framework of NMKB is an open and interesting problem including a construction of a unified view on update and revision (revisions of incomplete and imprecise knowledge about changing world).

We are aiming at a more detailed elaboration of postulates for logic program updates. An evaluation of our Kripkean semantics (Šefránek 2000; Šefránek 2004) in terms of postulates and also an evaluation and comparison of other frameworks for NMKB and defeasible reasoning (argumentation) in terms of the dependency framework is a topic for future research. Default negations in heads should be rethought. A challenging task is to use different strategies for conflict solutions along different dimensions in MDyLP. The relation of our approach to prioritized logic programs deserves a more detailed attention. A detailed analysis and comparison of the refined extension principle to irrelevant updates should be done. Satisfaction of fifth postulate of (Katsuno and Mendelzon 1991) and existence of a state condensing operator of (Leite 2003) in our framework is also an interesting open problem. We intend to investigate connections of our approach to other approaches to belief revision based on a notion of dependency (Darwiche and Pearl 1997; del Cerro and Herzig 1996) and others. Last, but not least, computational properties of our semantics have to be studied. Some of the problems mentioned above are topics of an ongoing research.

Acknowledgments. The work was supported under grants APVV-20-P04805 and VEGA 1/3112/06. I am grateful to one of the anonymous reviewers for his comments and also to Martin Baláž, Martin Homola and Jozef Šiška for many discussions about a previous version of the dependency framework.

References

Alferes, J.J., Pereira, L.M.: Reasoning with logic programming. Springer 1996

Alferes, J.J., Leite, J.A., Pereira, L.M., Przymusinska, H., Przymusinski, T.C.: Dynamic logic programming. In: Procs. of KR'98. (1998) 98–109

Alferes, J.J., Pereira, L.M.: Updates and preferences, Proc. of JELIA 2000. Springer.

Alferes, J.J., Banti, F., Brogi, A., Leite, J.A.: The refined extension principle for semantics of dynamic logic programming. Studia Logica **1** (2005)

Banti, F., Alferes, J.J., Brogi, A., Hitzler, P.: The well supported semantics for multidimensional dynamic logic programs. LPNMR 2005, LNCS 3662, Springer, 356-368

Darwiche, A., Pearl, J.: On the logic of iterated belief revision. Artificial Intelligence, 89, 1997

del Cerro, L.F., Herzig, A,: Belief change and dependence. Procs.of TARK VI, Morgan Kaufmann, 1996

Delgrande, J., Schaub, T., Tompits, H.: A Framework for Compiling Preferences in Logic Programs, Theory and Practice of Logic Programming 3(2), 2003, pp. 129-187

Eiter, T., Sabbatini, G., Fink, M., Tompits, H.: On properties of update sequences based on causal rejection. Theory and Practice of Logic Programming (2002) 711–767

Gärdenfors, P., Rott, H.: Belief revision. In: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 4 (Epistemic and Temporal Reasoning), Claredon Press. Oxford 1995

Homola, M., Šefránek, J., Baláž, M., Abstract dependency theory: preliminary report. http://www.computational-logic.org/content/events/iccl-ss-2005/talks/MartinHomola.ps

Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. Proc. of KR'91

Konczak, K., Linke, T., Schaub, T.: Graphs and Colorings for Answer Set Programming: Abridged Report. LPNMR 2004: 127-140

Leite, J.A., Alferes, J.J., Pereira, L.M.: Multi-dimensional dynamic knowledge representation. In: LPNMR 2001, Springer, 365–378

Leite, J.A.: Evolving Knowledge Bases: Specification and Semantics. IOS Press (2003)

Novák, P.: Stable Model Semantics Algorithm: Approach Based on Relation of Blocking Between Sets of Defaults; 2004; http://peter.aronde.net/publications.html

Pereira, L.M., Pinto, A.M.: Revised Stable Models - A Semantics for Logic Programs. EPIA 2005: 29-42

Šefránek, J.: Semantic considerations on rejection. In: Procs. of NMR 2004.

Šefránek, J.: A Kripkean semantics for logic program updates. : In M. Parigot, A. Voronkov (eds.), Logic for Programming and Automated Reasoning. Springer 2000, LNAI 1955

Šefránek, J.: Nonmonotonic integrity constraints. In Fink M., Tompits, H., Woltran, S. (eds.): Proc. of 20th Workshop on Logic Programming (WLP 2006), Vienna, 2006.

Šefránek, J.: Irrelevant updates of nonmonotonic knowledge bases. Accepted as a poster for ECAI 2006.