Irrelevant updates and nonmonotonic assumptions

Ján Šefránek

Comenius University, Bratislava, Slovakia, sefranek@fmph.uniba.sk

Abstract. The second postulate of Katsuno and Mendelzon characterizes irrelevant updates. We show that the postulate has to be modified, if nonmonotonic assumptions are considered. Our characterization of irrelevant updates is based on a dependency framework, which provides an alternative semantics of multidimensional dynamic logic programming.

Keywords: foundations of logic-based AI systems, nonmonotonic knowledge bases, updates, logic programming, nonmonotonic reasoning.

1 Introduction

Background Nonmonotonic knowledge bases (NMKB) represent an important topic for a logic-based research in artificial intelligence. There are essentially two sources of nonmonotony in knowledge bases – an evolution of incomplete knowledge and a use of assumptions which can be overridden (falsified).

We discuss in this paper the second postulate for updates by Katsuno and Mendelzon, 2[KM] hereafter: if an update follows from a knowledge base (KB), then the updated KB and the original KB should be equivalent according to the postulate [10]. In other words: if an update follows from a KB, then the update is irrelevant.

Tautological and unsupported cyclic updates represent an important problem also for multidimensional dynamic logic programming (MDyLP) research [2, 11, 12, 3]). MDyLP contributed to logic-based knowledge representation research by focusing on dynamic aspects of knowledge; it can be considered as a formal model of NMKB with preferences (also along dimensions different from time). The role of both sources of nonmonotony, as mentioned above, is taken into account in MDyLP.

Problem Unwanted generation of new models caused by cyclic or tautological updates (they should be irrelevant) has been a serious problem of MDyLP for a time. Recently, the problem has been solved for dynamic logic programs in [3] and for general MDyLP in [4]. The solution of [3] is based on a principle, called refined extension principle and its ambition is to express fundamental features enabling to distinguish the "right" semantics of logic program updates. Unfortunately, the principle has not been extended to the general case of MDyLP and, moreover, the trivial semantics assigning empty set of models to each dynamic logic program satisfies the principle, see [4]. Finally, the principle is expressed in terms too close to the specific conceptual apparatus of MDyLP. Semantics of MDyLP are based on rejection of rules. They satisfy the causal rejection principle (CRP): if there is a conflict between heads of rules, the less preferred rule is rejected. CRP has some drawbacks,[1] see [14], which challenges an effort to find a more general platform for expressing the notion of irrelevant updates.

On the other hand, 2[KM] is not an appropriate one, if (nonmonotonic) assumptions are considered (see Example 25).

Goal and proposed solution The main task of this paper is an analysis of the problem of irrelevant updates. Cyclic and tautological updates do not represent the only source of unwanted generation of models after an update. We are aiming to extend the horizon behind CRP and behind cyclic and tautological updates.

We represent an NMKB in terms of a dependency framework, where nonmonotonic assumptions and dependencies on assumptions are "first class citizens". Motivations for the dependency framework and an alternative semantics of MDyLP were presented in [14]. Each knowledge base consisting of rules (hence, also a logic program and a MDyLP) can be mapped into the dependency framework. Nonmonotony of a knowledge base is manifested in the framework by (nonmonotonic) assumptions which can be falsified and by solving of conflicts. We emphasize the role of nonmonotonic assumptions for the theory of updates of NMKB. The notion of irrelevant updates proposed in this paper modifies 2[KM] by focusing on nonmonotonic assumptions.

For a short version of this paper see [15],

Main contributions of the paper are as follows:

- analysis and definition of irrelevant updates,
- an adaptation of 2[KM] for updates of NMKB,
- we show that irrelevant updates do not generate new models and that models of the original program are preserved after an irrelevant update (Theorem 33 and Consequence 34).

The rest of the paper is organized as follows: First, an informal explanation of the notion of nonmonotonic knowledge base is presented. A brief view on semantics of MDyLP based on CRP is provided in Section 3. The dependency framework is introduced in Section 4. Irrelevant updates are discussed in Section 5 and defined in Section 6.

2 Nonmonotonic knowledge base

In this section we provide a schematic view on a NMKB. We assume an unspecified language $\mathcal{L}$ for knowledge representation. (An example of such language is presented in Section 3.) Pairs of conflicting expressions are specified for $\mathcal{L}$. A notion of rule is defined in $\mathcal{L}$. A knowledge base is a set of rules. We assume that a semantics of $\mathcal{L}$ is defined as a mapping from $\mathcal{L}$ to a set of models. (An example of such semantics, stable model semantics, is presented in Section 3.)

We are proceeding to the basic features of $\mathcal{L}$ and of its semantics which enable to speak about nonmonotony of a knowledge base.

[1] There are conflicts which cannot be recognized according to CRP. On the other hand, some conflicts recognized as conflicts between heads of rules should be considered as irrelevant.

- Some expressions of $\mathcal{L}$ are considered as assumptions. To each assumptions is assigned a conflicting expression. (Negative literals are assumptions of the language introduced in Section 3.)
- The semantics of assumptions satisfies the condition as follows: an assumption is true unless it is known that the conflicting expression is true.
- Some principles for solving conflicts in sets of expressions of $\mathcal{L}$ are specified.
- A nonmonotonic consequence operator is specified for $\mathcal{L}$.

MDyLP represent a well understood formalization of NMKBs, see Section 3. Also programs in AnsProlog [5], prioritized logic programs [6], DeLP [8] etc. can be considered as idealizations of NMKBs. We are interested in mappings from sets of rules to sets of dependencies (where dependencies on nonmonotonic assumptions are of crucial importance). Comparison of various formalisms in terms of the dependency framework is a goal of our future research.

MDyLP as a language for NMKBs representation (and its mapping into the dependency framework) is considered in this paper. However, the dependency framework is independent on MDyLP and our results can be generalized into a form independent on MDyLP.

3 Multidimensional dynamic logic programs

The language of MDyLP. i.e. a propositional language with default negations also in heads, is introduced in this section.

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 *negative*. Notation: $Ngt = \{not\ A \mid A \in \mathcal{A}\}$. 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. 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 $I^- = I \cap Ngt$. A literal is *satisfied* in an interpretation I iff $L \in I$. A set of literals S is satisfied in I iff $S \subseteq I$. Stable model of generalized logic programs is defined in [2] as follows.

Definition 1 ([2]) 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. □

The set of all stable models of a program P is denoted by $SM(P)$. A program is called *coherent*[2] iff it has a stable model.

A literal L (set of literals X) *SM-follows* from a program P iff it is satisfied in each $S \in SM(P)$ (notation: $P \models_{SM} L$, $P \models_{SM} X$). A rule r SM-follows from P iff for each $S \in SM(P)$ holds that $head(r)$ is satisfied in S whenever $body(r)$ is satisfied in S ($P \models_{SM} r$). If U is a set of rules then $P \models_{SM} U$ iff $\forall r \in U$ $P \models_{SM} r$.

A *multidimensional dynamic logic program* (also *multiprogram* hereafter) is a set of programs with a preference relation on programs. The relation is specified by an acyclic directed graph. If the preference relation collapsed to a sequence (to a strict linear order), the corresponding multiprogram is called *dynamic logic program*.

There are various different semantics of multiprograms, based on rejection of rules. For a comparison see [12, 13, 9]. They all can be viewed as instances of the abstract schema below.

Let a semantics be a mapping Σ from multiprograms to sets of total interpretations. If $\mathcal{P}$ is a multiprogram and M is a total interpretation then $M \in \Sigma(\mathcal{P})$ holds iff

$$M = least((\mathcal{P} \setminus Rejected(\mathcal{P}, M)) \cup Assumptions(\mathcal{P}, M). \qquad (1)$$

Various semantics differ in definitions of predicates $Rejected$ and $Assumptions$. Intuitively, $Rejected(\mathcal{P}, M)$ represents the set of all rules rejected from $\mathcal{P}$ w.r.t. M and $Assumptions(\mathcal{P}, M)$ represents the set of all accepted default negations.

In this paper we consider — because of limited space – only the simplest multiprograms of the form $\langle P, U\rangle$, where U is more preferred than P, notation $P \prec U$. U is called an update of P. However, it is straightforward to generalize the analysis, notions and results to the case of arbitrary multiprograms.

We will now define the instance of $Assumptions(\langle P, U\rangle, M)$ used in this paper as

$$\{not\ A \mid \neg\exists r \in P \cup U\ (A = head(r), M \models body(r))\}.$$

Two instances of $Rejected$ – Rej and Rej^R – are defined as follows. *Dynamic stable model* semantics [2] of a multiprogram $\langle P, U\rangle$ w.r.t. interpretation M, notation $DSM(\langle P, U\rangle, M)$, uses Rej:

$$Rej(\langle P, U\rangle, M) = \{r \in P \mid \exists r' \in U\ (r \bowtie r', M \models body(r'))\}.$$

Refined dynamic stable model semantics ($RDSM(\langle P, U\rangle, M)$) enables also mutual rejection of rules in one program, [3]. $Rej^R(\mathcal{P}, M)$ is defined as the set

$$\{r \in P \mid \exists r' \in P \cup U\ (r \bowtie r', M \models body(r'))\} \cup \{r \in U \mid \exists r' \in U\ (r \bowtie r', M \models body(r'))\}.$$

Dynamic stable model semantics of multiprograms suffers from tautological and cyclic updates.

[2] We prefer this term over ‘consistent”, see also [5].

Example 2 ([3])

$$P = \{d \leftarrow not\ n \qquad U = \{s \leftarrow v$$
$$n \leftarrow not\ d \qquad v \leftarrow s\}$$
$$s \leftarrow n, not\ c$$
$$not\ s \leftarrow$$

There are two dynamic stable models of $\langle P, U \rangle$: $M_1 = \{d, not\ n, not\ c, not\ s, not\ v\}$ and $M_2 = \{s, v, n, not\ d, not\ c\}$. However, M_2 is a counterintuitive model – truth of s and v is not supported and rejection of the fact $not\ s \leftarrow$ by a cyclic dependence of s on s is not a reasonable one. □

Refined semantics solves the problem. It obeys the refined extension principle (REP), introduced in [3]. However, the principle is expressed in terms too close to the specific conceptual apparatus of MDyLP. Two (not very intuitive) sequences of logic programs are considered in the definition of REP and the definition uses predicates $Assumptions$ and $Rejected$.

Moreover, refined semantics for the general case of multiprograms is not known. The well supported semantics for general multiprograms is defined in [4] and it solves the problem of cyclic updates. The well supported semantics for MDyLP coincides with the refined one on dynamic logic programs. We focus on the simplest dynamic logic programs of the form $\langle P, U \rangle$ in this paper, therefore the refined semantics is discussed in examples. However, our arguments are relevant w.r.t. any semantics based on rejection of rules and satisfying CRP.

The main goal of [3] is to explore the conditions guaranteeing that the addition of a set of rules to a dynamic logic program does not generate new models. The authors of REP observed in [4] that REP is too weak. For example, the trivial semantics that assigns to each dynamic logic program the empty set of models satisfies the principle. It is noted in [4] that stronger criteria and techniques are needed.

We believe that the dependency framework of [14] enables to create such criteria and techniques by providing a finer analysis of unwanted models of updated logic programs.

4 Dependency framework

We now introduce the dependency framework of [14] in order to be able to analyze the problem of irrelevant updates.

Definition 3 (Dependency relation) A *dependency relation* is a set of pairs $\{(L, W) \mid L \in Lit, W \subseteq Lit, L \notin W\}$. □

The notion of dependency relation is rather a general one and it is not connected to a special logical form (of a knowledge base or logic program). Each knowledge base consisting of a set of rules (with one literal in the head) can be mapped into a dependency relation. We define now a mapping for the language introduced in Section 3.

Definition 4 ($\ll_P$) 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 $\langle r_1, \ldots, r_k \rangle$, $k \geq 1$, $r_i \in P$ and

- $head(r_k) = L$,
- $W \models body(r_1)$,
- 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).

Example 5 Let P be

$$a \leftarrow not\ b$$
$$c \leftarrow a$$

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

Note that $\ll_P$ does not coincide with the derivability from P.

Definition 6 (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 $\mathrm{Cl}(\ll) = \ll$. □

Proposition 7 *Let P be a program. Then* $\mathrm{Cl}(\ll_P) = \ll_P$.

We have seen in Example 5 that dependencies on negative literals are crucial from the viewpoint of stable model semantics. Therefore the role of (default) assumptions is emphasized.

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

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

is non-empty and consistent.

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

The set of all (T)SSOAs w.r.t. $\ll$ is denoted by $(T)SSOA(\ll)$. □

Example 9 Let P be

$$a \leftarrow not\ b$$
$$b \leftarrow not\ a$$

There are two TSSOAs w.r.t. $\ll_P$: $Xs_1 = \{not\ b\}$ and $Xs_2 = \{not\ a\}$. $Cn_{\ll_P}(Xs_1) = \{not\ b, a\}$ and $Cn_{\ll_P}(Xs_2) = \{not\ a, b\}$. Notice that both TSSOAs generate (all) stable models of P. $\Box$

Also an empty set of literals may be a (T)SSOA w.r.t. some $\ll_P$.

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 $X \subseteq Ngt$, a TSSOA w.r..t. $\ll_P$ s.t. $S = Cn_{\ll_P}(X)$. $\Box$

Proof is straightforward from Definitions 1 and 4.

Semantics based on assumptions and dependencies. Consider two mappings Σ, Σ'. Let Σ assign to each program P the set of all its stable models. Let Σ' assign 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 $\langle P, U\rangle$ be a multiprogram. Then $\ll_{P\cup U}$ is well defined. It holds*

$$(\ll_P \cup \ll_U) \subseteq \ll_{P\cup U},$$

but the converse inclusion does not hold.

Sketch of a proof: Suppose that a sequence of rules from P or from U, which satisfies Definition 4 is given. Hence, we have a sequence of rules from $P \cup U$ satisfying Definition 4. The converse inclusion does not hold, see Example 12. $\Box$

Example 12 Consider a multiprogram $\langle P, U\rangle$, where P is as in Example 9 and U is as follows.

$$c \leftarrow a$$
$$b \leftarrow c$$
$$not\ b \leftarrow not\ a$$

$P \cup U$ is a program and it generates a dependency relation. Observe that $c \ll_{P\cup U} \{not\ b\}$, but $(c, \{not\ b\}) \notin (\ll_P \cup \ll_U)$. $\Box$

In general, dependencies on assumptions in a multiprogram can be conflicting. There are essentially two possible sources of incoherence/inconsistency:[3]

[3] (In)coherent dependency relation is defined in Definition 18 and we use (in)coherence as a technical term in the paper.

- two conflicting literals depend on a set of literals,
- or literal L_1 depends on a set of literals W, $(L_1, L_2) \in CON$ and $L_2 \in W$.

Definition 13 It is said that $\ll$ *contains a conflict* C (where $C \subseteq \ll$) iff for some $A \in \mathcal{A}$ is $C = \{(A, Y), (not\ A, Y) \mid Y \subseteq Ngt\}$ or $C = \{(A, Y) \mid Y \subseteq Ngt, not\ A \in Y\}$. $\Box$

Definition 13 does not contain a reference w.r..t. a (multi)program. Moreover, it is possible to replace pairs $A, not\ A$ by a more general objects – pairs of conflicting literals containing a (general) nonmonotonic assumption.

Example 14 Dependency relation $\ll_{P \cup U}$ from Example 12 contains conflicts $C_1 = \{(b, \{not\ a\}), (not\ b, \{not\ a\})\}$ and $C_2 = \{(b, \{not\ b\})\}$. $\Box$

It is assumed that the preference relation $P \prec U$ is preserved also for dependency relations assigned to the programs: $\ll_P \prec \ll_U$. It enables us to prefer some dependencies according to the preference relation defined for programs when solving conflicts.

We propose to solve a conflict by ignoring some dependencies (taking away from given dependency relation). Good solutions of a conflict are sets of dependencies, which are minimal (w.r.t. the set inclusion) and minimally preferred (w.r.t. the given preference relation).

Definition 15 It is said that a set of dependencies D is a *solution* of the conflict C iff each $d \in D$ is of the form $L \ll_P W$ or of the form $L \ll_U W$ and $C \not\subseteq Cl((\ll_P \cup \ll_U) \setminus D)$. 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 $\forall d \in D\ (d \in \ll_P \wedge \kappa(d) \in \ll_P \cup \ll_U)$. If the cardinality of D and D' is the same then for at least one $d \in D$ holds $d \in \ll_P \wedge \kappa(d) \in \ll_U$. A minimal solution D of a conflict C is called *good solution* iff there is no more suitable solution of C. $\Box$

Example 16 Consider conflicts from Example 14. $D = \{(not\ b, \{not\ a\})\}$ is a minimal solution of $C_1 = \{(b, \{not\ a\}), (not\ b, \{not\ a\})\}$. However, $D^{'} = \{(b, \{not\ a\})$ is a more suitable solution of C_1 than D and it is also the good solution of C_1 $\Box$

Definition 17 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 $Xs \subseteq Ngt$ is falsified in $\ll$ iff it contains a literal falsified in $\ll$. $\Box$

The notion of falsified assumption is illustrated in Example 21.

Definition 18 (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. $\Box$

The approach based on the dependency framework is focused on looking for assumptions which can serve as TSSOA w.r.t. a subset of given dependency relation $\ll_{P\cup U}$. The goal is to construct all possible dependency (sub)relations which are coherent (w.r.t. a TSSOA).

Definition 19 (Semantics of multiprograms) Semantics of multiprograms (of the form $\langle P, U\rangle$) is a mapping Σ which assigns to $\langle P, U\rangle$ a set of pairs of the form (Z, View), where View is a coherent subset of $\ll_{P\cup U}$ and Z is a TSSOA w.r.t. View.

Example 20 Recall examples 12 and 14. $\Sigma(\langle P, U\rangle) = \{(\{not\ a, not\ b, not\ c\}, \mathit{View})\}$, where $\mathit{View} = \ll_U$.

5 Irrelevant updates – intuitions

We motivate the notion of irrelevant updates by an analysis of a set of examples in this section.[4] Afterward a definition of irrelevant updates is given in next section.

Example 21 ([7])

$$P = it\_is\_cloudy \leftarrow it\_is\_raining$$
$$it\_is\_raining \leftarrow$$
$$U = not\ it\_is\_raining \leftarrow not\ it\_is\_cloudy$$

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

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 assumptions (default negations) which are falsified by the multiprogram. Interpretations generated by falsified assumptions do not provide an appropriate candidate for a semantic characterization of a multiprogram (according to our view). A remark is in place: conflicts involving assumptions did not attract an adequate attention until now.

If an update U has only such TSSOAs w.r.t. $\ll_U$ which are falsified in $\ll_{P\cup U}$, we consider it as irrelevant. However, some further criteria of irrelevant updates are needed. The first, rather naive, idea how to understand the principle of minimal change for the case $\langle P, U\rangle$ is as follows: if $P \cup U$ is a coherent program, then an update of P specified by U is irrelevant and the meaning of $P \cup U$ is retained by inertia.[5]

Next example shows that that idea is not an appropriate one.

[4] Multiprograms of the form $\langle P, U\rangle$ are used in the examples. However, the dependency framework is used in the analysis and our intuitions apply to an arbitrary NMKB which can be mapped to the dependency framework.

[5] Consider Example 21. The only stable model of $P \cup U$, $\{it\_is\_raining, it\_is\_cloudy\}$, provides a reasonable semantic characterization of $\langle P, U\rangle$.

Example 22 Let P be $\{not\ a \leftarrow not\ b\}$ and U be $\{a \leftarrow not\ b; b \leftarrow not\ a\}$.

$P \cup U$ has only one stable model $S = \{not\ \ a, b\}$. However, if we respect the preference of U over P then we have to ignore the information of P. The dependence of $not\ a$ on $not\ b$ should be ignored. Hence, also $\{not\ b\}$ is a TSSOA w.r.t. the modified dependency relation and interpretation $S' = \{not\ b, a\}$ represents an intended meaning of $\langle P, U\rangle$, too.

We emphasize the role of (new) assumptions. Acceptance of new assumptions can provide a basis for a generation of some alternative belief sets. In general, this observation may be relevant for investigation of hypothetical reasoning. □

Next step when looking for a formalization of irrelevant updates may be 2[KM]. It can be expressed in terms of logic program updates as follows: if $P \models_{SM} U$ then update of P by U is equivalent to P. It means, a (stable-models-like) semantic characterization of $\langle P, U\rangle$ should coincide with stable models of P. It is straightforward to show that a weaker condition than coherence of $P \cup U$ is supposed in 2[KM], if P is coherent.

Proposition 23 *Let P be coherent. If $P \models_{SM} U$ then $P \cup U$ is coherent (but not vice versa).*

Proof Sketch: If $P \models_{SM} U$ and P is coherent, then each stable model of P is also a stable model of $P \cup U$. A counterexample to the converse implication: Example 22 ($P \cup U$ is coherent, but $P \not\models_{SM} U$). □

Note that condition $P \models_{SM} U$ is an important one. If we add seemingly irrelevant (cyclic) update U to a program P and $P \not\models_{SM} U$, then this may lead to cutting off some models.

Example 24 ([3])

$$\begin{array}{rl} P = \{friends \leftarrow not\ alone & \quad U = \{depressed \leftarrow alone \\ alone \leftarrow not\ friends & \quad alone \leftarrow depressed\} \\ happy \leftarrow not\ depressed & \\ depressed \leftarrow not\ happy\} & \end{array}$$

P has four models (only first letters are used): $\{f, d, not\ a, not\ h\}$, $\{f, h, not\ a, not\ d\}$, $\{a, h, not\ d, not\ f\}$, $\{a, d, not\ h, not\ f\}$. Notice that $P \not\models_{SM} U$. It is natural to reject the models of P which do not satisfy the more preferred program U. U eliminates two of the models of P: $\{f, d, not\ \ a, not\ \ h\}$ and $\{a, h, not\ \ d, not\ \ f\}$ (if alone is true, depressed is forced to be true and vice versa). □

It seems that 2[KM] could be a criterion of irrelevant updates of logic programs. Unfortunately, 2[KM] does not work as the criterion.

Example 25

$$\begin{array}{rl} P = \{a_1 \leftarrow not\ b_1 & \quad U = \{b_2 \leftarrow not\ a_2\} \\ b_1 \leftarrow not\ a_1 & \\ a_2 \leftarrow not\ b_2 & \\ not\ b_2 \leftarrow not\ a_2\} & \end{array}$$

There are two stable models of P: $S_1 = \{not\ b_1, a_1, not\ b_2, a_2\}$ and $S_2 = \{not\ a_1, b_1, not\ b_2, a_2\}$. U is satisfied both in S_1 and in S_2. Note that assumption $not\ b_2$ holds in both models.

However, there is no reason to reject an alternative assumption $not\ a_2$ (which is false in both stable models of P). The set of assumptions $\{not\ a_2\}$ is a SSOA w.r.t. $\ll_U$ and $Cn_{\ll_P}(\{not\ a\}) \cup Cn_{\ll_U}(\{not\ a\})$ is inconsistent. The inconsistency can be overridden if we prefer $b_2 \ll_U \{not\ a_2\}$ over $not\ b_2 \ll_P \{not\ a_2\}$.

Hence, it is reasonable to accept also interpretations $S_3 = \{not\ b_1, a_1, not\ a_2, b_2\}$ and $S_4 = \{not\ a_1, b_1, not\ a_2, b_2\}$ as intended meanings of the updated program. U is really a relevant update: it provides a reasonable alternative assumption $not\ a_2$. By "reasonable" we mean that unwanted dependencies on $\{not\ a_2\}$ are overridden because of the preference relation. $\Box$

If a set of axioms is extended in a monotonic logic then the set of models is reduced or the same. However, in NMKBs (and in stable model semantics, too) it is not true:

Example 26 ([1]) Let be $P = \{a \leftarrow not\ b; b \leftarrow not\ a; c \leftarrow not\ a; c \leftarrow not\ c\}, U = \{c \leftarrow\}$.

While P has the only stable model $S = \{not\ a, b, c\}$, $P \cup U$ has two stable models – besides S also $S' = \{a, not\ b, c\}$. Observe that the only model of P encodes in a way that the truth of c is dependent on the assumption $not\ a$. Hence, the dependence of beliefs on assumptions is implicit also in the stable model semantics.

Note that $P \models_{SM} U$, but $Cn_{\ll_U}(\emptyset) \setminus Cn_{\ll_P}(\emptyset) \neq \emptyset$. Some literals depend on $\emptyset$ w.r.t. $\ll_U$, but they do not depend on $\emptyset$ w.r.t. $\ll_P$. This could be generalized to a criterion of a relevant update. $\Box$

Example 27

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

$P \models_{SM} U$, but U introduces a new assumption, which is false in all stable models of P and generates a new stable model of $P \cup U$. $\Box$

In order to summarize: If $P \not\models_{SM} U$, then U is a relevant update of P. Otherwise, if $P \models_{SM} U$, then U is a relevant update of P in two (classes of) cases. First, U introduces assumptions, which contribute to a new TSSOA w.r.t. $\ll_{P \cup U}$ (see Example 26 or Example 27). Second, U introduces a set of assumptions which is inconsistent with P, but a coherent view on $\ll_{P \cup U}$ is possible thanks to the preference relation (see Example 25).

Let $P \models_{SM} U$. A set of assumptions $Xs \subseteq Ngt$ of a relevant update U satisfies the conditions as follows. The conditions 1 - 3 are common to both classes of cases mentioned above. The condition 4 should be satisfied by the first case (contribution to a new TSSOA w.r.t. $\ll_{P \cup U}$). The conditions 5 and 6 should be satisfied by the second case.

1. Xs is not falsified in $\ll_{P \cup U}$,
2. Xs is false in each stable model of P,
3. $Xs \in SSOA(\ll_U)$ and $Cn_{\ll_U}(Xs) \setminus Xs \neq \emptyset$,

4. there is $Bs \subseteq Ngt$ s.t. $Xs \subseteq Bs$ and $Bs \in TSSOA(\ll_{P \cup U})$,
5. $Cn_{\ll_P}(Xs) \cup Cn_{\ll_U}(Xs)$ is inconsistent,
6. there is $View \subset \ll_{P \cup U}$ and $Bs \subseteq Ngt$ s.t. $Xs \subseteq Bs$ and $Bs \in TSSOA(View)$

Note that the condition 3 qualifies cyclic updates as irrelevant. Recall Example 2, it holds that $Cn_{\ll_U}(Xs) \setminus Xs = \emptyset$ for each set $Xs \subseteq Ngt$.

6 Irrelevant updates – formal elaboration

Irrelevant updates are defined in this section for programs of the form $\langle P, U \rangle$ and for corresponding dependency relations $\ll_{P \cup U}$. However, we can abstract from programs – a generalization to arbitrary dependency relation (on which a preference is defined) is straightforward.

Convention 28 Let $\langle P, U \rangle$ be a multiprogram. Then U is an irrelevant update of P iff $TSSOA(P \cup U) = TSSOA(P)$.

We are going to specify criteria for $TSSOA(P \cup U) = TSSOA(P)$ in terms of assumptions and dependencies. First some trivial implications.

Proposition 29 *If $P \not\models_{SM} U$, then $TSSOA(P) \not\subseteq TSSOA(P \cup U)$*

Proof Sketch: There is $S \in SM(P)$ s.t. $S \not\models_{SM} U$. Hence, $S \notin SM(P \cup U)$ The rest follows from Theorem 10. □

Proposition 30 *If $TSSOA(P \cup U) = TSSOA(P)$, then $P \models_{SM} U$*

Proof Sketch: Consider $S \in SM(P)$. $S^- \in TSSOA(P) = TSSOA(P \cup U)$. Therefore, U is satisfied in S. □

Proposition 31 *If $P \models_{SM} U$, then $TSSOA(P) \subseteq TSSOA(P \cup U)$.*

Proof Sketch: Let be $S \in SM(P)$, i.e. $S^- \in TSSOA(P)$. It is supposed that U is satisfied in S. Hence, $S = Cn_{\ll_P}(S^-) = Cn_{\ll_{P \cup U}}(S^-)$. □

Example 27 shows that $TSSOA(P \cup U) \subseteq TSSOA(P)$ does not hold in general, if $P \models_{SM} U$.

The condition $P \models_{SM} U$ is a necessary, but not sufficient condition for irrelevancy of an update U of P, see Examples 25 and 27.

2[KM] is a very intuitive postulate for updates (an intuition of updates is also in the background of dynamic logic programming according to [12]). However, 2[KM] cannot be accepted literally for updates of NMKB. Defeasible (nonmonotonic) assumptions are not considered by Katsuno and Mendelzon in [10]. A careful treatment of assumptions and dependencies on assumptions is required for an appropriate understanding of updates of NMKB.

We can define now irrelevant update of a program.

Definition 32 (Irrelevant update) Let $\langle P, U \rangle$ be a multiprogram, P be coherent and $P \models_{SM} U$.

It is said that U is an irrelevant update of P iff there is no $Xs \subseteq Ngt$ s.t.

- $Xs \in SSOA(\ll_U)$ and $Cn_{\ll_U}(Xs) \setminus Xs \neq \emptyset$,
- Xs is not falsified in $\ll_{P \cup U}$,
- Xs is false in each stable model of P,
- there is $Bs \subseteq Ngt$ s.t. $Xs \subseteq Bs$ and $Bs \in TSSOA(\ll_{P \cup U})$ or
  - $Cn_{\ll_P}(Xs) \cup Cn_{\ll_U}(Xs)$ is inconsistent,
  - there is $View \subseteq \ll_{P \cup U}$ and $Bs \subseteq Ngt$ s.t. $Xs \subseteq Bs$ and $Bs \in TSSOA(View)$

A nondeterministic algorithm for computation of TSSOAs is presented in [14].

Theorem 33 *If U is an irrelevant update of P then $TSSOA(\ll_{P \cup U}) = TSSOA(\ll_P)$.*

Proof Sketch: It is needed to prove that $TSSOA(P \cup U)) \subseteq TSSOA(P)$, the rest follows from Proposition 31. Let be $X \in TSSOA(P \cup U)$ If $X \notin TSSOA(P)$, then some condition of irrelevant updates from Definition 32 is violated. $\square$

Note that the converse implication does not hold – see Example 25. The following trivial consequence states that the semantics of an original program P is preserved after an irrelevant update.

Consequence 34 *Let Xs be a TSSOA w.r.t. $\ll_P$ and U be an irrelevant update of P. Then Xs is a TSSOA w.r.t. $\ll_{P \cup U}$.*

7 Conclusions

A notion of irrelevant updates based on a dependency framework is introduced in the paper. The dependency framework provides a general base for discussing updates of NMKBs. The role of nonmonotonic assumptions in updates (and also in hypothetical, nonmonotonic reasoning) has been emphasized.

It has been shown that irrelevant updates do not generate new TSSOAs (sets of assumptions, which generate stable models, Theorem 33) and that TSSOAs corresponding to the original program generate also all stable models of updated program (Consequence 34). The dependency framework solves also troubles caused by tautological and cyclic updates, [14].

As regards our research concerning conditions of updates of NMKBs, a relevancy postulate can be added to postulates from [14].

Attention has been frequently focused on MDyLP in this paper. The reason is that MDyLP is a well understood idealization of NMKB and also because of importance of the problem of irrelevant updates in MDyLP. Finally, our approach to irrelevant updates is based on the dependency framework, which provides an alternative semantics of MDyLP [14]. However, our notion of irrelevant updates can be expressed independently on MDyLP and it is among our future goals.

A short comment concerning related work: a modified version of 2[KM], [10], which reflects the presence of nonmonotonic assumptions, is presented. A more general view on irrelevant updates is given as in [3]. The notion of falsified assumptions enables to cover a more broad range of irrelevant updates. The trivial semantics (which satisfies REP) is not a problem for our approach.

References

1. Alferes, J.J., Pereira, L.M.: Reasoning with logic programming.
2. 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
3. Alferes, J.J., Banti, F., Brogi, A., Leite, J.A.: The refined extension principle for semantics of dynamic logic programming. Studia Logica **1** (2005)
4. 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
5. Baral, C.: Knowledge representation, reasoning and declarative problem solving. Cambridge University Press, 2003.
6. 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
7. 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
8. Garcia, A., Simari, G.: Defeasible Logic Programming: An Argumentative Approach. Theory and Practice of Logic Programming, Vol 4(1) pp 95-138, 2004.
9. Homola, M.: Dynamic Logic Programming: Various Semantics Are Equal on Acyclic Programs. CLIMA V 2004: 78-95
10. Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. Proc. of KR'91
11. Leite, J.A., Alferes, J.J., Pereira, L.M.: Multi-dimensional dynamic logic programming. In: Procs. of CLIMA'00. (2000) 17–26
12. Leite, J.A.: Evolving Knowledge Bases: Specification and Semantics. IOS Press (2003)
13. Leite, J. On Some Differences Between Semantics of Logic Program Updates. IBERAMIA 2004: 375-385
14. Šefránek, J.: Rethinking semantics of dynamic logic programming. Accepted for NMR 2006.
15. Šefránek, J.: Irrelevant updates of nonmonotonic knowledge bases. Accepted as a poster for ECAI 2006.