A dependency theory for logic program updates

Ján Šefránek
e-mail:sefranek@fmph.uniba.sk

Abstract

Dynamic aspects of knowledge represent a challenging research topic with many open problems. An approach to the field has been presented recently in the logic programming style. The approach is based on rejection of rules: if there is a conflict between rules then more preferred rules override those less preferred.

However, meaning of a nonmonotonic knowledge base is determined both by rules and by nonmonotonic assumptions. The role of assumptions in the conflicts is also significant. Therefore, also beliefs dependent on assumptions might be affected by conflicts.

The approach presented in this paper is focused on the dependencies between beliefs. A dependency theory is introduced together with a set of postulates for solving conflicts between dependencies. The approach is evaluated with respect to well known troubles with cyclic and irrelevant updates in the approaches based on the causal rejection principle.

1 Introduction

Reasoning about dynamic knowledge (knowledge evolving with time, knowledge integrated/permanently integrating from multiple sources, knowledge reflecting the changing environment or changing point of view etc.) is a challenging research topic with a critical importance for practical applications. Moreover, it is a topic with many open theoretical problems of fundamental significance. A deeper understanding of the dynamics of knowledge belongs to the most interesting and hardest tasks in knowledge representation research.

Logic programming has been proved as a powerful framework suitable for theoretical investigation of knowledge representation [4, 2, 5]. Nevertheless, the topic of the changing knowledge has been studied only recently in the logic programming style, see [3, 6, 8, 9] and others.

The basic paradigm of the research is based on the observation that "not all the information borne by a logic program is contained within its set of models" [7] and focus has been shifted from interpretation updates to program updates. The main attention is devoted to the conflicts between rules. They are resolved according to a causal rejection principle: if there is a conflict between rules then more preferred rules override those less preferred.

Unfortunately, causal rejection principle does not apply to all relevant cases. There are some conflicts between programs that are not identifiable on the level of conflicts between rules. Moreover, the causal rejection principle applies with respect to some irrelevant – from the updates point of view – cases and produces undesirable consequences. A special case of irrelevant updates are updates caused by tautologies and cycles in the more preferred programs.

Our approach is sensitive not only to conflicts between rules, but also to conflicts between beliefs. Semantics of nonmonotonic knowledge bases is determined both by rules and by nonmonotonic assumptions. We argue that it is important to record in the semantics also dependencies on assumptions. A dependency theory is introduced in this paper in order to obtain a solid base for logic program updates. The language of generalized extended logic programs is used. This tool enables to express both assumptions that $\neg A$ is true and also that A is true.

The paper is structured as follows: A more detailed motivation and an analysis of the problem is given in Section 2. Basic prerequisites and elements of the dependency theory are presented in Section 3, semantics of *true because of* in Section 4. Then the core of the paper paper follows. It consists in an application of dependencies and of a notion of *true because of* on updates of logic programs. Postulates constraining updates are introduced. Finally, a discussion and evaluation of the approach is presented in the Section 7.

We summarize the main contributions of the paper: A dependency theory is constructed as a basis of logic program updates. Postulates identifying source of conflicts and specifying rejection of dependencies are presented. It is shown that our semantics [11], which is able to distinguish finer conflicts than the conflicts between the rules, satisfy the postulates. The semantics does not suffer from the troubles with updates by irrelevant rules.

2 Motivation

Two rules r, r' are considered as conflicting in dynamic logic programming (DyLoP hereafter) paradigm iff their heads are L and $not\ L$, respectively.

We are aiming to argue in this Section that it is not sufficient to be focused only on conflicts between rules.

Two examples are given and analyzed below. First of them describes a conflict between programs that is not manifested by a conflict between rules. The conflict can be characterized in terms of dependencies between literals. Second example is aiming to show that some influence of dependencies between literals is latent also in the (classical) stable model semantics.

Example 1 Let U be a more preferred program than P.

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

There is no conflict (according to the DyLoP paradigm) between rules of both programs. No rule can be rejected and the meaning of $P \cup U$ cannot be updated according to the causal rejection principle.

However, there is a sort of conflict between the programs: The literal a in the less preferred program P depends on the literal c. On the other hand, the literal c depends on the default assumption $not\ a$ in the more preferred program U. Hence, there is a dependency of a on $not\ a$. It can be resolved by rejecting the less preferred dependency. □

Example 1 shows that there are conflicts not identifiable as conflicts between rules. However, they are identifiable on a level of dependencies. Meaning of logic programs (and meaning of nonmonotonic knowledge bases) is determined both by rules and by nonmonotonic assumptions. Therefore, it is important to distinguish also "finer" conflicts involving (dependencies on) nonmonotonic assumptions.

The dependency of beliefs on assumptions is implicit also in the stable model semantics. Examine the example as follows:

Example 2 ([2]) Let

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

While P has the only stable model $S = \{not\ a, b, c\}$, P' has two stable models – besides S also $S' = \{a, not\ b, c\}$. This seems to be a strange behaviour of the stable model semantics: adding a fact, which is true in the only stable model, increases the number of stable models. However, observe that the only model of P encodes in a way that the truth of c is dependent on the assumption $not\ a$ (it can be said that c is *true because of* $not\ a$). On the other hand, c is true in P' without a dependence on an assumption and, therefore, either a or b may be considered as true. □

Summary: A semantic treatment of dependencies on assumptions and of the conflicts between those dependencies is relevant for understanding and formalizing updates. We emphasize the importance of distinguishing those more "fine" conflicts as a key to overcoming well-known troubles with updates based on the causal rejection principle. Moreover, we are aiming at providing clear foundations for a theory of logic program updates. The discussion about approaches to logic program updates is often focused on some hard examples and some intuitions are provided in favor of (or against) some solutions. The arguments are based mainly on intuitions. We are interested in explicit principles underlying the approaches to logic program updates instead of ad hoc solving clashes of intuitions. This goal is realized in presented paper by a set of postulates. The postulates specify how to identify and how to solve conflicts between dependencies.

3 Basics

We assume a language of generalized extended logic programs (with default and explicit negation, both are allowed in heads of rules). Let $\mathcal{A}$ be a set of atoms. *Objective* literals are defined as follows: $Obj = \mathcal{A} \cup \{\neg A : A \in \mathcal{A}\}$. $Subj = \{not\ L : L \in Obj\}$ is the set of subjective literals. The set of *literals* is defined as $Lit = Obj \cup Subj$. Notation: $\neg(\neg A)) = A$, $not\ (not\ L) = L$.

The set of conflicting pairs is denoted by CON and for each $A \in \mathcal{A}$ holds $(A, \neg A) \in CON$, similarly $(L, not\ \ L) \in CON$ for each $L \in Obj$. It is not excluded that other pairs of literals belong to CON. A set of literals S is called *consistent* iff $(S \times S) \cap CON = \emptyset$, otherwise it is called *inconsistent*.

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 $body(r)$; $body(r)$ consists of two disjoint sets $body^+ \subseteq Obj$ and $body^-(r) \subseteq Subj$.

A finite set of rules is called *generalized extended logic program* (program hereafter). Each program might be considered as a definite program, where literals of the form $\neg A$, $not\ \ L$ are handled as new atoms. Hence, for each program exists his least model.

An *interpretation* is a consistent set of literals. A *total* interpretation is an interpretation such that for each literal L either $L \in I$ or $not\ \ L \in I$.

Our dependency theory is aiming at an agreement with stable model semantics.[1]

Definition 3 Let P be a program and *least* be the operator assigning the least model to a definite program. Let S be an interpretation and $S^- = \{L \in Subj \mid L \in S\}$.

Then S is a *partial stable* model of P iff $least(P \cup S^-) = S$. If partial stable model is a total interpretation, it is called *stable model*.

A program is *consistent* iff it has a stable model.

Let P be a program. The set $\bigcup_{r \in P} body^-(r)$ is called the set of *assumptions* in P (we denote it by $Ass(P)$).

Definition 4 (Dependency) Let a program P be given. A literal L *depends directly* on a set of literals V with respect to P (notation $L <_P V$) iff there is a rule $r \in P$ such that $L = head(r)$ and $V = body(r)$.

A literal L *depends* on a set of literals V with respect to P (notation $L \ll_P V$) iff $L <_P V$ or there is a set of literals U such that $L <_P U$, $L' \in U$, $L' \ll_P W$ and $V = (U \setminus \{L'\}) \cup W$.

A literal L_1 *depends* on a literal L_2 with respect to a program P ($L_1 \ll_P L_2$) iff there is a set of literals V such that $L_1 \ll_P V$ and $L_2 \in V$. $\Box$

The set of all dependencies with respect to a program P is denoted by Dep_P.

Fact 5 *If $L \ll_P W$ then there is a sequence $L <_P W_1, L_1 <_P W_2, \ldots, L_{k-1} <_P W_k = W$, $L_i \in W_i, i = 1, \ldots, k-1$.* $\Box$

It is said that $L <_P W_1$ is an *initial witness* of $L \ll_P W$.

[1] However, it is shown in Section 7 that we can overcome the limits of stable semantics.

Definition 6 Let $L \ll_P W$ and for no $L' \in W$ there is U such that $L' \ll_P U$. Then it is said that L is P-*grounded* (or grounded) in W.

A literal L is called *supported* in a program P iff $L \ll_P \emptyset$. A set of literals S is supported in P iff each its member is supported in P. □

A literal cannot be grounded because of a cyclic dependency. A dependency $L \ll_P W$ is called *cyclic* iff $L \in W$.

Proposition 7 *Let $L \ll_P W$ and $L \in W$. Then L is not grounded in W.*

Example 8 Let P be as follows:

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

Then $Dep_P = \{b \ll_P \{a\}, c \ll_P \{not\ d\}, a \ll_P \emptyset, b \ll_P \emptyset, b \ll_P a, c \ll_P$ $not\ d$, a, b are supported in P. Finally, $Ass(P) = \{not\ d\}$ and c is grounded in $\{not\ d\}$. □

4 True because of

Stable model semantics of a logic program P is determined by two components: by a set D of default assumptions and by the deductive closure of $P \cup D$. Therefore, each literal true in a stable model can be considered as true with respect to the corresponding set of assumptions. On the other hand, we will see that the stable model semantics is not able to distinguish some important aspects of meaning which might be expressed in terms of *true because of*.

Example 9 Consider the set of objective literals $\{a, b\}$ and a program P:

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

A straightforward idea to consider simultaneously a true because of $not\ b$ and b true because of $not\ a$ is certainly not a sound one. The set $\{a, b, not\ b, not\ a\}$ (the set of all assumptions and their consequences) cannot serve as a coherent set of beliefs. □

We are aiming to base the concept of *true because of* on sets of assumptions, but carefully – maintaining consistency (in accordance with the stable model semantics).

Definition 10 Let P be a program, $A \subseteq Ass(P)$, L be a literal.

It is said that L is in P *true because* of A (notation $val_P^A(L) = t$) iff $L \in A$ or $L \ll_P A$, L is grounded in A.

A is called a *sound* set of assumptions with respect to P iff the set $\{L \in Lit : val_P^A(L) = t\}$ is consistent. □

The notion of true because of is "immune" to cyclic dependencies.

Proposition 11 *If L is not grounded in A then $val_P^A(L)$ cannot be t.*

Definition 12 Let A be a sound set of assumptions with respect to P.

If $(L_1, L_2) \in CON$ and $val_P^A(L_1) = t$ then $val_P^A(L_2) = f$. If neither $val_P^A(L_1) = t$ nor $val_P^A(L_1) = f$ then $val_P^A(L_1) = u$. □

Example 13 Consider Example 9. $Ass(P) = \{not\ b, not\ a\}$, $A_1 = \{not\ b\}, A_2 = \{not\ a\}$. Therefore, $val_P^{A_1}(a) = t = val_P^{A_1}(not\ b)$, $val_P^{A_1}(not\ a) = f = val_P^{A_1}(b)$ and $val_P^{A_2}(b) = t = val_P^{A_2}(not\ a)$, $val_P^{A_2}(a) = f = val_P^{A_2}(not\ b)$. □

Example 14 Consider $P = \{$

$$\begin{array}{rcl} a & \leftarrow & b \\ b & \leftarrow & c \\ c & \leftarrow & a\}. \end{array}$$

Observe that $Ass(P) = \emptyset$ and $val_P^{\emptyset}(L) = u$ for each literal L.

Example 15 shows that for some sets of assumptions (and for a given program P) function val_P^A is not defined.

Example 15 $P_1 = \{a \leftarrow, \neg a \leftarrow\}$. □

Finally, there are programs with more sets of assumptions allowing to define val_M^A, see Example 13.

Definition 16 A set of assumptions $A \subseteq Ass(P)$ is called a *maximal sound set of assumptions* in P iff

- there is a finite set of sound sets of assumptions $A_1, A_2, \ldots, A_k, i \geq 1$, such that $A = A_1 \cup \cdots \cup A_k$,
- the set of all literals L such that $val_P^{A_i}(L) = t, i = 1, \ldots, k$, is consistent,
- there is no superset of A with this property. □

Example 17 Let P be

$$\begin{array}{rcl} a_1 & \leftarrow & not\ b_1 \\ b_1 & \leftarrow & not\ a_1 \\ a_2 & \leftarrow & not\ b_2 \\ b_2 & \leftarrow & not\ a_2 \end{array}$$

Maximal sound sets of assumptions: $A_1 = \{not\ b_1, not\ b_2\}, A_2 = \{not\ b_1, not\ a_2\}, A_3 = \{not\ a_1, not\ b_2\}, A_4 = \{not\ a_1, not\ a_2\}$.

Theorem 18 *Let A be a maximal sound set of assumptions of a program P. Then the set $S = \{L \in Lit \mid val_P^A = t\}$ is a partial stable model of P. If S is a total interpretation then S is a stable model of P.*

Definition 19 A literal L is true because of a maximal sound set of assumptions $A = \{A_1 \cup \cdots \cup A_k\}$ (notation: $val_P^A(L) = t$) iff there is A_i such that $val_P^{A_i}(L) = t$. If $val_P^A(L) = t$ and $(L, L') \in CON$, then $val_P^A(L') = f$. If neither $val_P^A(L) = t$ nor $val_P^A(L') = t$ then $val_P^A(L) = u = val_P^A(L')$. □

Definition 20 Let A be a maximal sound set of assumptions for a program P, A' be a set of subjective literals such that $A \subseteq A' \subseteq Subj$ and S be $\{L \in Lit \mid val_P^A(L) = t\}$.

It is said that A' is a *completion* of A for P w.r.t. S iff $A' \setminus A = \{not\ L \mid L \in Obj, L \notin S\}$.

We extend the definition of val for completions: holds:

- if $val_P^A(L) = t$ then $val_P^{A'}(L) = t$,
- if $val_M^A(L) = f$ then $val_P^{A'}(L) = f$,
- if $not\ L \in A' \setminus A$ then $val_P^{A'}(not\ L) = t$ and $val_P^{A'}(L) = f$. □

Example 21 Let P be $\{a \leftarrow b\}$. Suppose that Lit is $\{a, b, \neg a, \neg b, not\ a, not\ \neg a, not\ b, not\ \neg b\}$ and CON is $\{(a, \neg a), (b, \neg b), (a, not\ a), (b, not\ b), (\neg a, not\ \neg a), (\neg b, not\ \neg b)\}$.

Empty set is a sound set of assumptions: it holds that $val_L^{\emptyset}(L) = u$ for each literal L. $A' = \{not\ a, not\ b, not\ \neg a, not\ \neg b\}$ is a completion of $\emptyset$ for P: all assumptions are true because of A', and all objective literals are false with respect to A'. □

Theorem 22 *If S is a completion of a maximal sound set of assumptions for a program P then S is a stable model of P.*

5 Updates and dependencies

We intend to apply the dependency theory to a specification (constraining) of logic program updates. Our main point is that it is necessary to proceed (within the mainstream of research devoted to logic program updates) from the approach based primary on intuitions to the approach based on clear foundations when analyzing problems of logic program updates.

Let P and U are programs.[2] The former is called original program, the later updating program in DyLoP. The result of the update is called updated program and denoted by $P \oplus U$.

Updates should obey a principle of *inertia*, see for example [3, 6]. The intuitive meaning of this principle is as follows: The truth of literals from P remains unchanged

[2] Only the elementary case is considered in this paper because of limited size and also because of a need to explain the basic ideas for a simple setting. An extension to the general case is a straightforward one, however the formulation is a little bit more complicated.

in $P \oplus U$ unless it is directly affected by the update. Another principle emphasized by researchers in belief revision is the minimality of change: if an update is required then unnecessary losses of information are to be avoided.

We motivate our approach by some examples. First a very simple one.

Example 23 Let P be $\{\neg a \leftarrow\}$ and U be $\{a \leftarrow\}$. In terms of dependencies holds: $\neg a \ll_P \emptyset$, $a \ll_U \emptyset$.

U is more preferred program than P, therefore it is natural to reject the dependency $\neg a \ll_P \emptyset$. This principle is expressed by the postulate P2 in Section 6. The set of updated dependencies $Dep_{P \oplus U}$ is $\{a \ll_{P \oplus U} \emptyset\}$. $\square$

Example 24

$$\begin{array}{rl} P = a \leftarrow & \quad U = not\ a \leftarrow not\ b \\ b \leftarrow a & \end{array}$$

In dynamic stable semantics [8] the meaning of updated program $P \oplus U$ is specified by two dynamic stable models $\{a, b\}$ and $\{not\ a, not\ b\}$, but the stable model of $P \cup U$ is $\{a, b\}$.

According to our postulate P1, if $P \cup U$ is a consistent program then no update is needed (this requirement fits well the principle of minimal change). Hence, if Dyn is a “dynamic” semantics based on a “static” semantics Sem and $P \cup U$ is a consistent program then should hold that $Dyn(P \oplus U) = Sem(P \cup U)$.

If $P \cup U$ is consistent then all dependencies from both P and U hold also in $P \oplus U$, i.e. $Dep_P \cup Dep_U \subseteq Dep_{P \oplus U}$:

$$\begin{array}{rl} a \ll_P \emptyset & \quad not\ a \ll_U \{not\ b\} \\ b \ll_P \{a\} & \\ b \ll_P \emptyset & \end{array}$$

The crucial task is to extend concepts of *true because of* to $P \oplus U$.

$$\begin{array}{rcl} val_P^{\emptyset}(a) & = t = & val_P^{\emptyset}(b) \\ val_P^{\emptyset}(not\ a) & = f = & val_P^{\emptyset}(not\ b) \\ val_U^{\{not\ b\}}(not\ a) & = t = & val_U^{\{not\ b\}}(not\ b) \\ val_U^{\{not\ b\}}(a) & = f = & val_U^{\{not\ b\}}(b) \end{array}$$

Resolving the conflicts between dependencies is the main problem of our paper and we solve it in terms of *true because of*. If there is a conflict between dependencies then dependencies from U override those from P unless assumptions from U are falsified by “facts” from P (i.e. by literals supported by empty sets of assumptions from P). $val_P^{\emptyset}(not\ b) = f$ and we want not accept $\{not\ b\}$ as a sound set of assumptions for $P \oplus U$ (see postulate P4).

$$\begin{array}{rcl} val_{P \oplus U}^{\emptyset}(a) & = t = & val_{P \oplus U}^{\emptyset}(b) \\ val_{P \oplus U}^{\emptyset}(not\ a) & = f = & val_{P \oplus U}^{\emptyset}(not\ b) \end{array}$$

The dependency of $not\ a$ on $not\ b$ holds also in $P \oplus U$ but we do not consider $\{not\ b\}$ as a sound set of assumptions. □

Two important types of conflicts are distinguished:

1. conflicting beliefs depend on the same belief set,
2. conflicts between assumptions.

The first type corresponds to conflicts between rules. The second cannot be expressed in terms of conflicts between rules.

Postulates specify conditions for rejecting (better to say, ignoring) dependencies affected by conflicts. In dependency theories characterized by [12] conflict resolution is performed by the addition of a set of internally generated justifications. In DyLoP it is performed by rejection of some rules (justifications). Our approach is aiming at conflict resolution on purely semantic level and no addition or rejection of rules is specified. Rules may contain an useful information, even if they are responsible for a conflict with respect to the current state of knowledge (or knowledge base).

6 Postulates

We can now list the postulates. The conflicts generated by one (inconsistent) program are not handled by the postulates in this paper – we are not interested in this problem. However, it is easy to add a postulate of the form as follows. If $L_1 \ll_P W$, $L_2 \ll_P W$ and $(L_1, L_2) \in CON$, then both dependencies are ignored (rejected).

Common assumptions for all postulates and examples: $(L_1, L_2) \in CON$, S is a set of literals. We assume that $P \cup U$ is a consistent program in the postulate P1, in P2, P3 it is assumed that $P \cup U$ is inconsistent.

P1 if S is a (partial) stable model of $P \cup U$ then for each $L_1 \in S$ holds $val^{S^-}_{P \oplus U}(L) = t$ and for each L_2 holds $val^{S^-}_{P \oplus U}(L_2) = f$, otherwise – if S is not a (partial) stable model of $P \cup U$ – then $val^{S}_{P \oplus U}$ is not defined,

P2 let $L_1 \ll_P W$, if $L_2 \ll_U W$, $L_1 \not\ll_U W$, then each initial witness $L_1 <_P W_1$ (denoted by d) is rejected ($d \in Rej$),

P3 if $L_1 \ll_P W$ and $W \ll_U \{L_2\}$ and $L_1 \not\ll_P \emptyset$ then each initial witness $d = (L_1 <_P W_1)$ is rejected ($d \in Rej$),

P4 if $val^{\emptyset}_P(L_1) = t$, $val^{\emptyset}_U(L_2) \neq t$ then $val^{\{L_2\}}_{P \oplus U}$ is not defined, i.e. $\{L_2\}$ is not a sound set of assumptions.

P5 if $val^{\emptyset}_P(L_1) = t$, $val^{\emptyset}_U(L_2) \neq t$, $L_2 \ll_U W$, $W \neq \emptyset$, $L_2 \not\ll_U \emptyset$ then $val^{\{L_2\}}_{P \oplus U}$ is not defined, i.e. $\{L_2\}$ is not a sound set of assumptions.[3]

[3]Postulates P4 and P5 are expressed for singletons, but they can be generalized to arbitrary sets of assumptions.

The dependencies can be rejected only according to the postulates P1 – P3.

The following example illustrate Postulates P3 and P

Example 25 (P3) Let P be $\{a \leftarrow c\}$ and U be $\{b \leftarrow \mathit{not}\ a; c \leftarrow b\}$.
$a <_P \{c\}$ is rejected because of $\{c\} \ll_U \{\mathit{not}\ a\}$. $\Box$

Example 26 (P5) Let P be $\{a \leftarrow; b \leftarrow a\}$ and U be $\{\mathit{not}\ b \leftarrow \mathit{not}\ c\}$. $\mathit{val}_{P \oplus U}^{\{\mathit{not}\ c\}}$ is not defined.

We are going to define the set of all non-rejected *direct* dependencies in $P \oplus U$. After that the set $\mathit{Dep}_{P \oplus U}$ of all (non-rejected) dependencies is defined in the style used in Definition 4.

Definition 27 Let Direct_P (Direct_U) be the set of all direct dependencies in Dep_P and Dep_U. Then $\mathit{Direct}_{P \oplus U} = (\mathit{Direct}_P \cup \mathit{Direct}_U) \setminus \mathit{Rej}$. $\Box$

The relation $\ll_{P \oplus U}$ is defined exactly according to Definition 4. Notions of sound, maximal sound set of assumptions w.r.t. $P \oplus U$ and three-valued or two-valued valuations are defined as in Section 4 except of the requirement to satisfy Postulates P4 and P5.

7 Discussion

Dependency theory allows a more finer semantics than the stable semantics. Consider programs as follows.

Example 28

$$
\begin{array}{ll}
P_1 = \{a \leftarrow \mathit{not}\ b\} & P_2 = \{a \leftarrow; \mathit{not}\ b \leftarrow\} \\
P_3 = \{a \leftarrow; b \leftarrow b\} & P_4 = \{a \leftarrow \mathit{not}\ b; \mathit{not}\ b \leftarrow\}
\end{array}
$$

$S = \{a, \mathit{not}\ b\}$ is the only stable model of all four programs. However, there are subtle differences in (the intuitive) meaning of the programs. The differences can be recorded by dependencies defined in this paper. $\Box$

The stable semantics can be defined over dependencies, if assumptions contain only default negations. A semantics more suitable for the goals illustrated by Example refzajac can be defined for more general sets of assumptions.

Such a generalization enables also to introduce a semantics based on the concept of strong equivalence.

Definition 29 ([10]) Two programs P and Q are said to be strongly equivalent ($P \equiv_s Q$) iff for each program R the programs $P \cup R$ and $P \cup Q$ have the same sets of models.

This is a topic of our ongoing research.

Our approach does not suffer from troubles as tautological, cyclic and irrelevant updates. Moreover, it enables to distinguish conflicts not identifiable from the causal rejection principle point of view. Finally the Kripkean semantics of [11] is correct with respect to postulates introduced in this paper.

A Kripkean semantics of (multidimensional) dynamic logic programs is presented in [11]. A Kripke structure is assigned to each program. The structure records – in a sense – dependencies between sets of literals. An update operation is defined on Kripke structures. We present (without the necessary prerequisites because of the limited size of the paper) the main idea of the theorem claiming that Kripkean semantics is correct with respect to our postulates. A detailed exposition will be presented in a forthcoming paper.

Theorem 30 *Postulates P1 – P5 are satisfied by the update operation on Kripke structures associated with logic programs.* □

It is shown in [11] that Kripkean semantics does not suffer from tautological, cyclic and irrelevant updates and that it is able to recognize conflicts not distinguishable on the level of conflicts between rules. The same holds also for our dependency theory (for updates of dependencies and of the notion *true because of*). A more detailed discussion is prepared also for a forthcoming paper.

8 Conclusions

A dependency theory for logic programs has been introduced in this paper. Dependencies are defined explicitly and they are used for a definition of *true because of*. The semantics based on that notion is built. Updates of logic programs are specified in our approach as rejections of some dependencies. Postulates P1 – P5 for specifying rational rejections of dependencies and for constraining sound sets of assumptions are introduced. We do not reject or insert new rules. Our goal is only to make a coherent view of the current state of a modular knowledge base (represented by logic programs). The rejection of dependencies is more sensitive (able to make a more fine distinguishing) than the causal rejection principle.

The substantial improvement gained by our approach is based on the observation that the meaning of nonmonotonic knowledge bases is determined both by the rules and by the nonmonotonic assumptions. Therefore, updates should respect both conflicts between rules and conflicts caused by assumptions.

Main results:

- our dependency theory enables to solve the problem of cyclic and irrelevant updates,
- it is able to identify the conflicts not distinguishable by the causal rejection principle,
- the postulates P1 – P5 (contrary to the AGM postulates [1] and other postulates of that sort) specify dependencies responsible for the conflicts between beliefs

(and/or belief sets); moreover, they allow to handle nonmonotonic dependencies based on assumptions.

Some topics for future research: Investigation of the logic corresponding to *true because of* with the property that both beliefs from a conflicting pair can be false. Relations to well founded semantics. Relation of *true because of* to strong equivalence of logic programs.

References

[1] Alchourron, C., Gärdenfors, P., Makinson, D. *On the logic of theory change. Partial meet contraction and revision functions.* Journal of Symbolic Logic, 50:510-530, 1985

[2] Alferes, J., Pereira, L. *Reasoning with Logic Programming.* Springer 1996

[3] Alferes, J.J., Leite, J.A., Pereira, L.M., Przymusinska, H., Przymusinski, T.C. *Dynamic Updates of Non-Monotonic Knowledge Bases.* The Journal of Logic Programming 45 (1-3):43-70, September/October 2000

[4] Baral, C., Gelfond, M. *Logic Programming and Knowledge Representation.* Journal of Logic Programing 1994:19, 20: 73-148

[5] Dix, J., Brewka, G. *Knowledge Representation with Logic Programs.* Universität Koblenz-Landau, 1996

[6] Eiter, T., Fink, M., Sabbatini, G., Tompits, H. *On properties of update sequences based on causal rejection.* 2001

[7] Leite, J., Pereira, L. *Generalizing Updates: from models to programs.* In LNAI 1471, 1997

[8] Leite, J. *Evolving Knowledge Bases.* IOT Press, 2003.

[9] Leite, J.A., Alferes, J.J., Pereira, L.M. *Multi-dimensional dynamic knowledge representation.* In: Eiter, T., Faber, W., Truszczynski (Eds.): LPNMR 2001, Springer, 365-378

[10] Lifschitz, V., Pearce, D., Valverde, A. *Strongly equivalent logic programs.* ACM Transactions on Computational Logic, 2(4):526-541, 2001

[11] Šefránek, J. *Semantic considerations on rejection.* Proceedings of NMR 2004, Whistler, BC, Canada.

[12] Witteveen, C., Brewka, G. *Skeptical reason maintenance and belief revision.* Artificial Intelligence 61 (1993), 1-36