Nonmonotonic Integrity Constraints

Ján Šefránek

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

Abstract. Semantics of multidimensional dynamic logic programming is traditionally based on the causal rejection principle: if there is a conflict between rules then the rule from a less preferred program is rejected. However, sometimes it is useful to solve a conflict between the heads of rules by blocking the body of a rule. Moreover, semantics based on the causal rejection principle, is not able to recognize conflicts, which are not manifested as conflicts between the heads of rules.

Nonmonotonic integrity constraints are discussed in this paper. They provide alternative solutions of conflicts (as compared with solutions based on causal rejection principle). Conceptual apparatus introduced in this paper enables also to distinguish more preferred interpretations and, consequently, it is relevant for logic programming with preferences. Nonmonotonic integrity constraints and other notions introduced in the paper (falsified assumptions, more preferred assumptions) contribute to bridging the gap between research in fields as belief revision or preference handling on the one hand and multidimensional dynamic logic programming on the other hand.

Keywords: nonmonotonic reasoning, updates, multidimensional dynamic logic programming

1 Introduction

Multidimensional dynamic logic programming (MDyLoP) [1, 11, 12, 3] provides an interesting and promising approach to representation of dynamic aspects of knowledge in the context of logic-based knowledge representation research. Most semantics of MDyLoP respect the causal rejection principle (CRP): if the heads of two rules are conflicting then the less preferred rule is rejected.

Our research [16, 15, 17] is aiming to overcome some drawbacks of CRP. We are focused on assumptions, dependencies on assumptions, conflicts involving assumptions or dependencies. A dependency framework is introduced [17] in order to be able to handle assumptions and dependencies explicitly and to solve also conflicts, which are not manifested as conflicts between the heads of rules. Nonmonotonic (defeasible, default) assumptions play a crucial role in nonmonotonic reasoning and so the dependency framework can be useful also for a foundational research of nonmonotonic reasoning and for a comparison of various approaches to logic program updates (and to defeasible reasoning/argumentation). In this paper a more detailed attention is devoted to nonmonotonic integrity constraints introduced in [17]. We address the topic of removal of nonmonotonic integrity constraints using falsification w.r.t. more preferred assumptions (the topic has not been detailed in [17]). If nonmonotonic integrity constraints, preference on assumptions and falsification of assumptions are considered then disagreement of MDyLoP with other approaches relevant for updates of nonmonotonic knowledge bases (NMKB) can be overcome (f.ex. with research in the fields of belief revision, see [9, 10, 13] and others, or preference handling, see [7, 8, 5, 6] and others).

Main contributions of the paper: A detailed discussion of nonmonotonic integrity constraints. A demonstration that they are useful for alternative solutions of conflicts. It is shown also that other concepts (preference on assumptions, falsification w.r.t. a set of assumptions, introduced in order to provide for nonmonotony of integrity constraints) enable to recognize more preferred stable models. A modified nondeterministic algorithm for computing a coherent view on a dependency relation is presented.

Roadmap: Basics of multidimensional dynamic logic programming are recapped in Section 2. Dependency framework is described in Section 3. An introduction of integrity constraints is motivated in Section 4. After that, in Section 5 the dependency framework is extended and a semantics of multidimensional dynamic logic programs based on the dependency framework is sketched in Section 6. Finally, in Section 7 main contributions of the paper and some open problems are listed.[1]

[1] This research has been supported by grants APVV-20-P04805, VEGA 1/0173/03 and 1/3112/06.

2 Preliminaries

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*. Notation: $Subj = \{not\ A \mid A \in \mathcal{A}\}$. We use not as default negation, with intuitive meaning "it is not known that ...". A convention: $not\ not\ A = A$.

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 $(head(r_1), head(r_2)) \in CON$. Notation: $r_1 \bowtie r_2$. A set of literals S is *consistent* if it does not contain a pair of conflicting literals, $(S \times S) \cap CON = \emptyset$. 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 L is *satisfied* in an interpretation I iff $L \in I$. A set of literals S is satisfied in I iff $S \subseteq I$.

Definition 1 ([1]) 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 ([11]) 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 $i = j$. We denote by $i \parallel j$ that i and j are incomparable w.r.t. $\preceq$. If $i \prec j$, we say that P_j is *more preferred* than P_i. □

If G is a path, we speak about *dynamic* logic program.

Definition 3 (Dynamic stable model, [11]) Let $\mathcal{P}$ be a multiprogram. A total interpretation M is called *dynamic stable model* of $\mathcal{P}$ iff

$$M = least(\bigcup_{i \in V} P_i \setminus Rejected(\mathcal{P}, M)) \cup Defaults(\mathcal{P}, M)), \tag{1}$$

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

Refined dynamic stable model is defined in [3] 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 $Rejected^R(\mathcal{P}, M)$). The set of all refined dynamic stable models of $\mathcal{P}$ is denoted by $RDSM(\mathcal{P})$. Troubles with tautological and cyclic updates are overcome in refined semantics. However, the refined semantics is defined only for dynamic logic programs. Refined semantics for the general case of multiprograms is not known. The well supported semantics of multiprograms is defined in [4], in order to improve the behaviour of semantics based on CRP.

We will use refined semantics in the analysis of examples, which contain dynamic logic programs. The well supported semantics for MDyLoP coincides with the refined one on dynamic logic programs.

3 Dependency framework

Idea of nonmonotonic integrity constraints is based on a dependency framework presented in [17]. We recap the basic features of the framework in this section.

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

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 $\langle r_1, \ldots, r_k \rangle$ with $k \geq 1, r_i \in P$ and

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

It is said that the dependency relation $\ll_P = \{(L, W) \mid L \ll_P W\}$ is *generated* by the program 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 $\mathrm{Cl}(\ll) = \ll$.

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

Proof: 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 $L \ll_P W$,

The converse inclusion is (yet more) trivial. □

Dependencies on subjective literals are crucial from the viewpoint of stable semantics. Therefore the role of (default) assumptions is emphasized.

Definition 7 (SSOA, TSSOA) $Ass \subseteq Subj$ is called a *sound set of assumptions* (SSOA) with respect to the dependency relation $\ll$ iff the set

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

is non-empty and consistent.

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

Theorem 8 *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. $S = Cn_{\ll_P}(X)$.

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

Proposition 9 *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.

Proof Sketch: Each sequence of rules from $\bigcup_{i \in V} P_i$ (which satisfies conditions of Definition 4) determines a dependency of a literal on a set of literals (w.r.t. the program $\bigcup_{i \in V} P_i$).

It is straightforward to show an example falsifying the converse inclusion. □

Definition 10 (Coherent dependency relation) A dependency relation $\ll$ is called *coherent* iff there is an TSSOA w.r.t. $\ll$. A dependency relation is called *incoherent* iff it is not a coherent one. □

In general, $\ll_{\bigcup_{i \in V} P_i}$ can be incoherent. Our approach to semantics of MDyLoP is focused on looking for sets of assumptions which can serve as a TSSOA w.r.t. a (coherent) subset of given dependency relation $\ll_{\bigcup_{i \in V} P_i}$. A (maximal) coherent subset of an incoherent dependency relation can be considered as a reasonable semantic view on the dependency relation. 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). Therefore, we are aiming at finding all reasonable TSSOAs w.r.t. some corresponding subsets of a given dependency relation.

A construction (a non-deterministic algorithm) is described in [17]. We now present the basic idea of this construction. Later, in Section 5, we extend the construction for the case of nonmonotonic integrity constraints.

There are essentially two possible sources of incoherence in the union $\bigcup_{i \in V} P_i$:
(1) two conflicting literals depend on a set of literals;
(2) an atom A depends on a set of literals W and $not\ A \in W$.

Hence, we apply two criteria for constructing a coherent semantic view on a set of dependencies. The criteria specify which dependencies should be ignored.[2]

Definition 11 Let a dependency relation $\ll$ be given. Let a finite set $\mathcal{D} = \{\ll_1, \ldots, \ll_k\}$, where $\ll_i \subset \ll$, be specified. Suppose that an acyclic, transitive and irreflexive preference relation ρ on $\mathcal{D}$ is defined. 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).

1. Let be $(L_1, L_2) \in CON$, $d_1 = L_1 \ll W$, $d_2 = L_2 \ll W$. If d_1 is less preferred than d_2 then a minimal set of dependencies D such that $d_1 \notin Cl(\ll \setminus D)$ is ignored.
2. If $A \ll W$, $not\ A \in W$ then a minimal set of dependencies D such that $(A, W) \notin Cl(\ll \setminus D)$ is ignored.

Note that criterion 1 corresponds to the CRP, but the other criterion extends the possibilities of solving conflicts. A more radical extension of our dependency framework is introduced in Section 5 thanks to nonmonotonic integrity constraints.

Definition 12 Let $\mathcal{P}$ be a multiprogram. It is said that $\ll_{\bigcup_{i \in V}}$ *contains a conflict* C (where $C \subseteq \ll_{\bigcup_{i \in V}}$) iff for some $A \in \mathcal{A}$ is $C = \{(A, Y), (not\ A, Y)\}$ or $C = \{(A, Y)\}$ with $not\ A \in Y$.

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 $L \ll_{P_i} W$ and $C \nsubseteq Cl(\ll_{\bigcup_{i \in V}} \setminus D)$.

D, a solution of C, 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 $\forall d \in D \exists d' \in D'\ ((d' = L \ll_{P_j} W) \wedge (d = L \ll_{P_i} W) \wedge j \prec i)$. A minimal solution D of a conflict C is called *good solution* iff there is no more suitable solution of C. $\square$

A solution of a conflict is focused on dependencies generated by a single program. Only elementary pieces of a chain of dependencies are ignored (dependencies from a $\ll_{P_i}$). Good solutions are focused on less preferred dependencies.

Dependency framework will be finalized in Section 5 after a motivation in next section

4 Motivation

Introduction of nonmonotonic integrity constraints is in [17] motivated by an analysis of some drawbacks of the CRP. One 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).

[2] Our approach does not reject or insert some rules. Its ambition is to provide a coherent view on a (possibly incoherent) MDyLoP (NMKB) by *ignoring* some dependencies and by accepting some assumptions in the role of integrity constraints.

Example 13 [3] Let $\mathcal{P}$ be $\langle P_1, P_2 \rangle$, where $1 \prec 2$.

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

$RDSM(\mathcal{P}) = \{\{not\ a, b\}\}$ and $Rejected^R(\mathcal{P}, \{not\ a, b\}) = \{a \leftarrow\}$. 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 $P_1 \cup P_2$ and two corresponding stable models – besides $\{not\ a, b\}$ also $\{not\ b, a\}$.

Notice that the empty set of assumptions justifies inconsistent set of literals. Criterion 1 of Definition 11 enables to create only one coherent subset of $\ll_{P_1 \cup P_2}$. We have the set of dependencies $\ll_{P_1 \cup P_2}$ as follows: $\{(a, \emptyset), (b, \emptyset), (not\ a, \{b\}), (not\ a, \emptyset)\}$. If $a \ll_{P_1} \emptyset$ is ignored according to criterion 1 of Definition 11 then the coherent subset $View$ of $\ll_{P_1 \cup P_2}$ is $Cl(\ll_{P_1 \cup P_2} \setminus \{a \ll_{P_1} \emptyset\})$. So we obtain $Cn_{View}(\emptyset) = \{b, not\ a\}$. However, $\{a, not\ b\}$ is impossible to get in such a way, even if an assumption $not\ b$ is accepted. So, simple adding of new assumptions does not work as a means for generating alternative solutions of a conflict illustrated by our example.

If we add assumptions $not\ a$ or $not\ b$ in the role of (nonmonotonic) integrity constraints, we can get alternative solutions of the inconsistency considered here. If $not\ b$ is accepted in the role of an integrity constraint then justification of b is blocked and only a and $not\ b$ are justified. Similarly, if $not\ a$ is accepted then justification of a is blocked and only $not\ a$ and b are justified.

A formalization of this idea is as follows. We consider two sets of subjective literals, besides assumptions also integrity constraints. Let accept $not\ b$ as an integrity constraint (a set of integrity constraints is denoted by IC) and $Ass = \emptyset$ as a set of assumptions. $View$, a subset of $\ll_{P_1 \cup P_2}$ is obtained by ignoring all dependencies of the form $b \ll_{P \cup U} W$ because of the integrity constraint. Finally, we can define $ICn_{View}((IC \mid Ass))$ as the set of all literals $\{L \mid (L, Ass) \in View \wedge not\ L \notin IC\} \cup Ass \cup IC$, hence we obtain a reasonable model $\{not\ b, a\}$ for $IC = \{not\ b\}$ and $Ass = \emptyset$.

Similarly for an integrity constraint $not\ a$.

Note that nonmonotonic integrity constraints are not needed if solutions of conflicts do not respect the preference relation on dependencies. However, we want to preserve this feature of dynamic logic programming. Moreover, nonmonotonic integrity constraints can be viewed also as concise representations of some less succinct representations of alternative conflict solutions. This topic should be understood in a more detail. $\Box$

The nature of integrity constraints is nonmonotonic.

Example 14 Suppose that a third, the most preferred, program P_3 is added to $\mathcal{P}$ from Example 13. Let be $P_3 = \{c \leftarrow; a \leftarrow c; b \leftarrow c\}$ and the corresponding multiprogram be denoted by $\mathcal{P}'$.

It would be natural to reject (nonmonotonic) integrity constraints accepted for $\mathcal{P}$ (and also dependency $not\ a \ll_{P_2} \{b\}$). The first suggestion could be to reject an integrity constraint if it is "generated" by a program P_i and falsified by a program P_j, where $i \prec j$.

Similarly for $P_3' = \{a \leftarrow not\ c; b \leftarrow not\ c\}$. In this case we will speak about falsification w.r.t. some (more preferred) assumptions. $\Box$

Consider now logic programs with preferences. We sketch only the basic idea – distinguishing the more preferred assumptions enables to distinguish (and select) more preferred TSSOAs (and, consequently, more preferred stable models). A detailed exposition of our approach to logic programs with preferences and a comparison with other approaches is postponed to a future paper.

Example 15 ([7]) This example contains explicit ("classic") negation and names of rules are used. However, we believe that there is no problem with tracing the exposition below. Literal $\neg a$ could be considered as a new atom, if needed; n_i is the name of rule r_i, atom $n_3 \prec n_2$ means that rule r_2 is more preferred than

[3] This example is due to Martin Baláž.

r_3.

$$
\begin{aligned}
r_1 &= \neg a \leftarrow \\
r_2 &= b \leftarrow \neg a, not\ c \\
r_3 &= c \leftarrow not\ b \\
r_4 &= n_3 \prec n_2 \leftarrow not\ d
\end{aligned}
$$

Program containing rules $r_1 - r_4$ has two regular stable models (answer sets). $M_1 = \{\neg a, b, n_3 \prec n_2, not\ c, not\ d\}$, $M_2 = \{\neg a, c, n_3 \prec n_2, not\ b, not\ d\}$. However, r_2 overrides r_3, hence M_1 is the only preferred answer set (r_2 must be used before r_3, therefore r_3 is blocked, it is not applicable).

The same selection of more preferred stable model (TSSOA) can be obtained using the notion of more preferred assumptions, see Example 16. □

Note that there is a trivial correspondence between logic programs with preferences and multidimensional dynamic logic programs. Consider first static preferences (on rules). If a logic program with preferences is given as a pair $(\{r_i \mid i \in I\}, \prec)$, then the corresponding MDyLoP we obtain as a set of programs (singletons) $P_i = \{r_i\}$ preserving $\prec$: $P_i \prec P_j$ iff $r_i \prec r_j$.

Conversely, let a MDyLoP $\mathcal{P}$ 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.

If preference relation is a dynamic one (as in Example 15, where it can be modified by rules) then a dynamic preference relation on programs is needed. A possibility of such extension of MDyLoP is supposed (f.ex. in [1]), but we are not aware of a realization of the possibility. However, it is feasible and straightforward.

We can now to proceed to an adapted version of Example 15.

Example 16

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

A straightforward translation from Example 15 is possible, too. Our choice here is to present and analyze a multiprogram $\langle P_1, P_2\rangle$, with $P_1 \prec P_2$, while preserving the main features of the original program. It means, the preference of $b \leftarrow a, not\ c$ over $c \leftarrow not\ b$ is preserved. What is changed as compared with Example 15: atom a is used instead of $\neg a$, atoms with relational symbol $\prec$ are not used, so atom d is not needed. Preference of $a \leftarrow$ over $c \leftarrow not\ b$ is added, but it is not an essential change: $\neg a$ holds in Example 15 in both answer sets.

$RDSM(\langle P_1, P_2\rangle) = \{\{a, b, not\ c\}, \{a, c, not\ b\}\}$. There are no conflicting rules in this multiprogram and it is not possible to reject the less preferred model (according to the CRP).

However, the notion of more preferred assumptions enables to select the more preferred answer set from Example 15.

Observe that b is justified in the more preferred program. Hence, "it is not known b" seems not to be a reasonable assumption. We will consider the assumption $\{not\ c\}$ as more preferred than assumption $\{not\ b\}$. A set of assumptions can be falsified also by a more preferred set of assumptions. In our example: the assumption $\{not\ b\}$ is falsified w.r.t. the assumption $\{not\ c\}$ and dependency relation $\ll_{P_1 \cup P_2}$. The more preferred set of assumptions "generates" the more preferred stable model: $Cn_{\ll_{P_1 \cup P_2}}(\{not\ c\}) = \{a, b, not\ c\}$ and assumption $\{not\ b\}$ is falsified in $Cn_{\ll_{P_1 \cup P_2}}(\{not\ c\})$.

There is an intuitive difference between updates and preferences (see [2]). However, the *multidimensional* approach of MDyLoP should represent also "preferential" reasoning. May be, different strategies for different dimensions are needed. Moreover, there are some problems with very notion of updates, if default negations are allowed (even in heads of rules), see [17]. Updates of NMKB provide a challenging problem for future research. □

Formal definitions motivated in this section are introduced in the next section.

5 Nonmonotonic integrity constraints

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 $Ass \subseteq Subj$ is falsified in $\ll$ iff it contains a literal falsified in $\ll$. □

Definition 18 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 19 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 Ass 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 $L_2 \in Ass$ such that $not\ L_2 \in Cn_{\ll_{View}}(X)$,
- 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. $View'$. □

Example 20 Consider Example 16. $S(not\ b) = \{1\}$ and $S(not\ c) = \{2\}$, hence $not\ c$ is more preferred than $not\ b$.

Further, $\{not\ c\}$ is a TSSOA w.r.t. $\ll_{P_1 \cup P_2}$, it is neither falsified nor falsified w.r.t. a more preferred set of assumptions and a dependency relation and finally, $b \in Cn_{\ll_{P_1 \cup P_2}}(\{not\ c\})$.

Therefore, $\{not\ b\}$ is falsified w.r.t. $\{not\ c\}$ and $\ll_{P_1 \cup P_2}$. □

We have introduced some new features (for the dependency framework):

- notions of falsification and falsification w.r.t. a set assumptions and a dependency relation,
- preferences on (sets of) assumptions.

We are going to extend the dependency framework by nonmonotonic integrity constraints.

Assumptions with integrity constraints (we will use "i-assumption" as a shorthand) are pairs of the form $(X \mid Y)$, where $X, Y \subseteq Subj$. Literals from X are called (nonmonotonic) integrity constraints.

Definition 21 (ISSOA, ITSSOA) Let $(X \mid Y)$ be i-assumptions and $\ll$ be a dependency relation. Let $Cn_{\ll}(Y)$ be a SSOA w.r.t. $\ll$ and X, Y be not falsified or falsified w.r.t. a set of assumptions and a dependency relation. Then

$$ICn_{\ll}((X \mid Y)) = \{L \in Lit \mid L \in Cn_{\ll}(Y) \wedge not\ L \notin X\} \cup X$$

It is said that $ICn_{\ll}((X \mid Y))$ obeys integrity constraints X. If $ICn_{\ll}((X \mid Y))$ is not empty then $(X \mid Y)$ is dubbed ISSOA w.r.t. $\ll$ (ITSSOA in the case of total interpretation). □

A remark: it is straightforward to reconcile notions of SSOA (TSSOA) with ISSOA (ITSSOA); X is an SSOA w.r.t. $\ll$ iff $(\emptyset \mid X)$ is an ISSOA w.r.t. $\ll$.

Example 22 Remind Example 13. $ICn_{View}((\{not\ a\} \mid \emptyset)) = \{not\ a, b\}$, similarly, $ICn_{View}((\{not\ b\} \mid \emptyset)) = \{not\ b, a\}$. □

Similarly, Example 20 can be re-interpreted in terms of ITSSOAs using $(\emptyset, \{not\ c\})$ instead of $\{not\ c\}$. Of course, some modifications of our formalization are required: Definition 19 have to be modified and a preference relation on pairs of sets of assumptions have to be defined. It is possible to combine Examples 13 and 16 in order to show that non-empty integrity constraints can play a role in preferential reasoning (in logic programs with preferences). Nonmonotonic integrity constraints together with notions of falsification or falsification w.r.t. to a set of assumptions and a dependency relation enable to recognize more preferred ITSSOAs and, therefore, more preferred stable models. Application of the dependency framework to a semantic characterization of logic programs with preferences will be detailed in a forthcoming paper.

6 Semantics based on the dependency framework

Semantics of a multiprogram $\mathcal{P}$ in the dependency framework is a mapping Σ which assigns the set of pairs of the form $((X \mid Y), \mathit{View})$, where $(X \mid Y)$ is an ITSSOA w.r.t. View, to the multiprogram.

Definition of coherent dependency relation is adapted to ITSSOA in [17]. Our approach to semantics of MDyLoP described in [17] is focused on looking for i-assumptions which can serve as a(n I)TSSOA w.r.t. a (coherent) subset of a given dependency relation $\ll_{\bigcup_{i \in V} P_i}$. An ITSSOA $(X_1 \mid Y_1)$ is called a good sound set of assumptions (GSSOA) iff there is no ITSSOA $(X_2 \mid Y_2)$ such that $Y_2 \subset Y_1$, i.e. ITSSOAs with minimal sets of assumptions are preferred. A construction of coherent dependency relation from an incoherent $\ll_{\bigcup_{i \in V} P_i}$ is proposed. The construction is described in terms of a non-deterministic algorithm. The constructed relation represents – in a sense – a coherent semantic view on an incoherent multiprogram. The construction is in the spirit of answer set programming: all consequences are derived from a set of (i-)assumptions Ass via non-conflicting dependencies on Ass (integrity constraints can block some derivations). If the set of all GSSOAs of a multiprogram $\mathcal{P}$ is $\{Z_1, \ldots, Z_k\}$ w.r.t. $\{\mathit{View}_1, \ldots, \mathit{View}_k\}$, respectively, where $Z_i = (X_i, Y_i)$, then there is a *canonical program* of the form $\{L \leftarrow \mid L \in X_i\} \cup \{L \leftarrow Y_i \mid L \in \mathit{ICn}_{\mathit{View}_i}(Z_i)\}, i = 1, \ldots, k$. It holds that the set of all stable models of the canonical program coincide with the set of sets $\{\mathit{ICn}_{\mathit{View}_i}(Z_i) \mid Z_i$ is a GSOA w.r.t. $\mathit{View}_i\}$.

Finally, we present in Figure 1 a modified non-deterministic algorithm constructing an ITSSOA w.r.t. a $\mathit{View} \subseteq \ll_{\bigcup_{i \in V} P_i}$. It is assumed that there is a set Ω containing pairs of the form (Z, View), where Z are i-assumptions and View is a dependency relation. Initially. i-assumptions are of the form $(\emptyset, Y)$, where Y is neither falsified in $\ll_{\bigcup_{i \in V} P_i}$ nor falsified w.r.t. a more preferred set of assumptions and View is $\ll_{\bigcup_{i \in V} P_i}$. Nonmonotonic integrity constraints are added to i-assumptions by the algorithm. A strategy for generating nonmonotonic integrity constraints IC is as follows: if a dependency of an atom A on a set of literals W belongs to the set D then $not\ A$ is included into IC; similarly, View is reduced.

INPUT: a pair (Z, View) from Ω, where $Z = (X \mid Y)$, $\mathit{View} \subseteq \ll_{\bigcup_{i \in V} P_i}$
OUTPUT: a pair (Z^T, View), where Z^T is an ITSSOA w.r.t. View or the decision that it is not possible to construct an ITSSOA from Z
begin
  if Z is an ITSSOA w.r.t. View **then** RETURN (Z, View) **fi**
  $Z^T := Z; X^T = X, \mathit{View}^T := \mathit{View}$
  REPEAT
    if View^T contains a conflict C of View^T **then**
      SELECT ALL pairs π of the form (IC, D), where D is a good solution of C
        and IC are integrity constraints **fi**
    $i := 0$;
    for each $(IC, D) \in \pi$ **do**
      if $X^T \cup IC$ is not falsified or falsified w.r.t. a more preferred set of assumptions **then**
        $i := i + 1; X_i^T := X^T \cup IC; Z_i^T := (X_i^T \mid Y); \mathit{View}_i^T := \mathit{Cl}(\mathit{View}^T \setminus D)$ **fi**
      if $i > 1$ **then** $\Omega := \Omega \cup (Z_i^T, \mathit{View}_i^T)$ **fi**;
    od
    $\mathit{View}^T := \mathit{View}_1^T; Z^T := Z_1^T; X^T := X_1^T$
    if $i = 0$ **then** FAILURE := true **else** FAILURE := false **fi**
  UNTIL $\mathit{ICn}_{\mathit{View}^T}(Z^T)$ is an ITSSOA w.r.t. View^T or FAILURE
if not FAILURE **then** RETURN (Z^T, View^T) **else** RETURN FAILURE **fi**
end

Fig. 1. Non-deterministic algorithm removing conflicts and computing ITSSOAs

7 Discussion, conclusions

First, we will show that integrity constraints introduced in the dependency framework are nonmonotonic and that our framework enables to distinguish more preferred models (ITSSOAs) of logic programs (with preferences).

Example 14 illustrates nonmonotony of our integrity constraints. In general, nonmonotonic integrity constraints can be falsified or falsified w.r.t. more preferred assumptions when some rules are added to the corresponding multiprogram. This fact follows in a very straightforward way from our framework.

Fact 23 *Let $\mathcal{P}$ be a multiprogram with $G = (V, E)$. Let $(X|Y)$ be ITSSOA w.r.t. a $View \subseteq \ll_{\bigcup_{i \in V} P_i}$.*
Then $(X|Y)$ are nonmonotonic *i-assumptions in the sense as follows:*

- *There is a multiprogram $\mathcal{P}'$, an extension of $\mathcal{P}$ by a program P_z, where for each $i \in V$ is $i \prec z$ and $View \cap \ll_{P_z} = \emptyset$.*
- *X is falsified (or falsified w.r.t. some i-assumptions $(X'|Y')$ and $\ll_{i \in V'}$, where $V' = V \cup \{z\}$).* □

Of course, i-assumptions cannot be ITSSOAs if their integrity constraints are falsified.

Fact 24 *Let $\ll$ be a dependency relation. Let X and Y be sets of assumptions such that both are SSOAs w.r.t. $\ll$, i.e. $Cn_{\ll}(X)$ and $Cn_{\ll}(Y)$ are consistent, but for some atom $A \in Cn_{\ll}(Y)$ holds that not $A \in X$.*

If Y is more preferred than X and Y is neither falsified nor falsified w.r.t. a set of assumptions and a dependency relation then $(\emptyset, Y)$ is an ISSOA w.r.t. $\ll$ and $(\emptyset, X)$ is not an ISSOA w.r.t. $\ll$.

Fact 24 can be generalized to i-assumptions (with non-empty integrity constraints).

Main contributions of the paper are as follows. A more detailed analysis of nonmonotonic integrity constraints as compared with [17] is given. It is shown also that our dependency framework enables preference handling. A modified non-deterministic algorithm for computing ITSSOAs is presented.

The paper is a product of a research devoted to semantics of MDyLoP based on dependencies and assumptions. The semantics is aiming to overcome some drawbacks of semantics based on CRP. Current state of our research is presented in [17], where the dependency framework is introduced, irrelevant updates are defined, a coherent semantic view on a multiprogram is specified, a non-deterministic algorithm producing such coherent view is presented.

Some open problems: Comparisons of other approaches to updates of NMKB and to defeasible reasoning (argumentation) from the dependency framework point of view. Comparison with the abductive framework of [14]. Rethinking relation of updates and revisions in NMKB. It is shown in [17] that postulates for updates of Katsumo and Mendelzon [10] cannot be understood literally in the context of logic program updates because of the presence of nonmonotonic assumptions, therefore a much more careful approach to the difference between updates and revisions is needed. Also a detailed application of presented framework to logic programs with preferences and a characterization of computational aspects of the framework are intended (and needed).

References

1. 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
2. Alferes, Pereira Updates and preferences. Proc. of JELIA 2000. Springer.
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. Brewka, G.: Well-Founded Semantics for Extended Logic Programs with Dynamic Preferences. Journal of Artificial Intelligence Research, 4 (1996),19-36
6. Brewka, G., Eiter, T.: Preferred answer sets for extended logic programs. Artificial Intelligence, 109 (1-2):297-356,1999

7. 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
8. Delgrande, J., Schaub, T., Tompits, H., Wang, K.: A classification of preference handling approaches in nonmonotonic reasoning. Computational Intelligence 20:2, 2004, 308-334
9. 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
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. Liberatore, P., Schaerf, M.: The compactness of belief revision and update operators. Fundamenta Informaticae XX (2004), 1-17, IOS Press
14. Sakama, C., Inoue, K.: Updating extended logic programs through abduction. Logic Programming and Nonmonotonic Reasoning. LNAI 1730, Springer, 1999
15. Šefránek, J.: Semantic considerations on rejection. In: Procs. of NMR 2004.
16. 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
17. Šefránek, J.: Rethinking semantics of dynamic logic programming. Submitted.