Declarative Semantics for Contradictory Modular Logic Programs

Anastasia Analyti & Sakti Pramanik
Department of Computer Science, Michigan State University, E. Lansing, MI 48824
E-mail: analyti@cps.msu.edu, pramanik@cpswh.msu.edu

Abstract. A complex reasoning system can be designed as an interaction between reasoning modules. A module consists of a declaration of exported/imported predicates and a set of rules containing both negation by default and classical negation. A *prioritized modular logic program* (*PMP*) consists of a set of modules and a partial order $<_{\text{def}}$ on the predicate definitions (M, p), where M is a module and p is a predicate exported by M. Because of the classical negation, conflicts may arise within and among modules. The partial order $<_{\text{def}}$ denotes the relative reliability of the predicate definitions contributing to the conflict. We present the *reliable semantics* for *PMPs*. The goal of the reliable semantics is to draw reliable conclusions from possibly contradictory *PMPs*.

1 Introduction

Modules in a reasoning system arise as a result of a functional decomposition of a complex reasoning task into a number of simpler subtasks. Each module is an interactive reasoning subsystem that is used for the (often partial) *definition* of its exported predicates. Each module contains a set of rules viewed as an open logic theory [1] with a set of input literals. A module represents an incomplete specification of some domain because its input literals are defined in other modules. However, a module becomes a standard extended logic program (closed module) when the truth values of its input literals are known.

A *prioritized modular logic program* (*PMP*) consists of a set of modules and a partial order $<_{\text{def}}$ on the predicate definitions (M, p), where M is a module and p is a predicate exported by M. We assume that modules are internally consistent. However, a *PMP* is possibly not globally consistent. When a conflict occurs, $<_{\text{def}}$ expresses our relative confidence in the predicate definitions contributing to the conflict. Each module has a set of *local* literals that are inaccessible to other modules. Literals that can be accessed (imported) by any module have the form $\{M_1,...,M_n\}$:A, $\{M_1,...,M_n\}$: $\neg A$ or $\{M_1,...,M_n\}$: $\sim A$, where M_i are module names and A is a conventional atom whose predicate is exported by all M_i. Intuitively, a literal $\{M_1,...,M_n\}$:A is evaluated as true if (i) A is derived from a module M_i, and (ii) if $\neg A$ is derived from a module $M_j \neq M_i$ then result A has higher priority than result $\neg A$. A literal $\{M_1,...,M_n\}$: $\sim A$ is evaluated as true if $\{M_1,...,M_n\}$: $\neg A$ is true or $\sim A$ is true in all modules M_i.

We present a semantics for *PMPs*, called *reliable semantics* (*RS*), which assigns a truth value *true*, *false* or *unknown* to every literal. Every *PMP* has at least one *stable m-model*. The *RS* of a program P is the least fixpoint of a monotonic operator and the least (w.r.t. $\subseteq$) stable m-model of P. When a *PMP* is contradictory, exported results (represented by indexed literals) are considered *unreliable* if: (i) they contribute to a contradiction, and (ii) they do not have higher priority than the other contributing results. The *RS* of a *PMP*, *P*, represents the skeptical meaning of *P* and thus, none of its conclusions is based on unreliable exported results. Credulous conclusions are obtained by isolating the conflicting results in the multiple stable *m*-models of *P*.

2 Informal Presentation and Intuitions

Our framework can be used for the representation of result-sharing cooperating agents [14]. A complex task is statically decomposed into a set of simpler subtasks, each assigned to an agent. Agents may have overlapping or even identical capabilities. Therefore, it is possible that they export agreeing or contradictory results. When agents M_1, M_2 export contradictory conclusions about a literal L, the truth value of L w.r.t. M_1, M_2 (expressed by the truth value of the literal $\{M_1, M_2\}$:L) is unknown. Yet, agents M_1, M_2 maintain their individual beliefs about L which is expressed by the truth value of the literals $\{M_1\}$:L, $\{M_2\}$:L, respectively.

Example 2.1: Sensors S_1, S_2 are gathering information from two different angles about the persons entering a building. Modules M_1, M_2 are assigned with the identification of terrorists based on the information collected from sensors S_1, S_2, respectively. Each module M_1, M_2 exports the result *entered*(*terrorist*) (resp. ¬*entered*(*terrorist*)) iff it reaches the conclusion that an (resp. no) terrorist has entered the building. It is possible that M_1, M_2 disagree, i.e., M_1 exports *entered*(*terrorist*) whereas M_2 exports ¬*entered*(*terrorist*). The results exported by M_1, M_2 can be queried by other modules in various ways. For example,

- $Query_1 \leftarrow \{M_1\}$:*entered*(*terrorist*), $\{M_2\}$:*entered*(*terrorist*).
  $Query_1$ is true if *entered*(*terrorist*) is true in both M_1, M_2. $Query_1$ is false by default if *entered*(*terrorist*) is false (by default or classically) in at least one of M_1, M_2.
- $Query_2 \leftarrow \{M_1, M_2\}$: *entered*(*terrorist*).
  $Query_2$ is true if *entered*(*terrorist*) is true in at least one of M_1, M_2 and M_1, M_2 do not disagree. $Query_2$ is false by default if *entered*(*terrorist*) is false in both M_1, M_2.
- $Query_3 \leftarrow \{M_1\}$: ~*entered*(*terrorist*).
  $Query_3$ is true if *entered*(*terrorist*) is false in M_1 (even if *entered*(*terrorist*) is true in M_2).

Individual agent theories are assumed to be consistent. Yet, the consistency of the union of agent theories is not assured. As we saw in Example 2.1, one case of contradiction is when independent modules export contradictory results. In this case, the contradiction depends only on the independent modules and it is relatively easy to resolve. Yet, generally, contradictions may involve several module interactions. For example, an agent exports a faulty result, this result is imported by another agent which exports a faulty result based on the imported faulty result. After a few module interactions, contradiction may arise in two ways : (i) Complementary literals are derived inside an module. (ii) Complementary literals are exported from two different modules.

When contradiction appears, the sources of the contradiction are traced back to the contributing exported results. Domain specific information might indicate that some exported results are more reliable (have higher priority) than others. Let res_1 and res_2 be two exported results contributing to the contradiction. If res_1 has higher priority than res_2 and no contradiction arises without res_2 then only res_1 is taken into account. If the priority of res_1, res_2 cannot be compared then both are eliminated from *RS* (skeptical approach).

3. *m*-models for Prioritized Modular Logic programs

Our alphabet contains a finite set of constant, predicate and variable symbols from which ordinary atoms are constructed in the usual way. It also contains a finite set of module names. An *indexed atom* has the form $\{M_1,...,M_n\}:A$, where A is an ordinary atom and M_i are module names. A *classical literal* is either an atom A or its classical negation $\neg A$. The classical negation of a literal L is denoted by $\neg L$, where $\neg(\neg L)=L$. A *default literal* is the default negation $\sim L$ of a classical literal L, where $\sim(\sim L)=L$. A literal is called *indexed* when its corresponding atom is indexed. We define $\{M_1,...,M_n\}:\neg A = \neg\{M_1,...,M_n\}:A$ and $\{M_1,...,M_n\}:\sim A= \sim\{M_1,...,M_n\}:A$, where M_i are module names and A is an ordinary atom. The classical literals L, $\neg L$ are called *complementary* to each other. The predicate of a literal L is denoted by $pred_L$.

A module with name M is a tuple $<Exp_M, Imp_M, R_M>$. The set Exp_M contains the predicates that are exported (defined) by M. The set Imp_M contains the predicates imported by M. The set R_M contains rules of the form: $L\leftarrow L_1,...,L_m,\sim L_{m+1},...,\sim L_n$, where L is a non-indexed classical literal and L_i are classical literals. If an indexed literal $\{M_1,...,M_n\}:L$ is in the body of a rule then M imports L from the modules $M_1,...,M_n$. If a non-indexed literal L is in the body of a rule and $pred_L\in Imp_M$ then M imports L from all modules that export $pred_L$.

A *prioritized modular logic program* (*PMP*), P, is a pair $<Mod_P, <_{\text{def}}>$. Mod_P is a set of modules and $<_{\text{def}}$ a partial order on Def_P, where $Def_P=\{(M,p) \mid M\in Mod_P$ and $p\in Exp_M\}$. Each pair $(M,p)\in Def_P$ represents the *definition* of predicate p in module M. If $\{M_1,...,M_n\}:L$ is an indexed literal appearing in P then $(M_i,pred_L)\in Def_P, \forall i\leq n$.

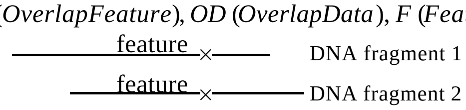
Individual modules are assumed to be consistent but their union may be inconsistent. Thus, when complementary literals are derived within a module M, it is because of unreliable information imported by M. When literals L, $\neg L$ are derived from different modules M, M', it is because the definition of $pred_L$ in M, M' is unreliable or the information imported by M, M' is unreliable. When conflict occurs, $<_{\text{def}}$ expresses our relative confidence in the predicate definitions contributing to the conflict. Let (M,p) and (M',p') be in Def_P. The notation $(M,p) < (M',p')$ (resp. $(M,p) \not< (M',p')$) means that the definition of p in M is less (resp. not less) trusted than that of p' in M'. Intuitively, a literal L exported by a module M is reliable if it does not contribute to any derivation of complementary literals caused by definitions $(M',p') \not< (M,pred_L)$. Note that, the reliability of L is not affected by predicate definitions less trusted than the definition of $pred_L$ in M.

We define $S_L=\{M\in Mod_P \mid pred_L\in Exp_M\}$, where L is a literal. The indexed literals $S_L:L$ are called *globally indexed*. To simplify the presentation of the semantics a *renaming* mechanism is employed. Let r be a rule in a module M. Then, the head L of r is replaced by a new literal $M\#L$. Every non-indexed literal L in the body of r with $pred_L \notin Imp_M$ is replaced by $M\#L$. Every non-indexed literal L in the body of r with $pred_L \in Imp_M$ is replaced by the globally indexed literal $S_L{:}L$. Literals $M\#L$ are called *local* to M and they are not accessed by other modules. In contrast to local literals, *indexed* literals are accessible to all modules. When we refer to a *PMP*, we assume that the above renaming has already been done. Note that after renaming, only local and indexed literals appear in a module M. We define $M\#\neg A = \neg M\#A$ and $M\#{\sim}A = {\sim}M\#A$, where M is a module name and A is an ordinary atom.

The set of all instantiations of the classical literals appearing in P and the globally indexed classical literals $S_L{:}L$, where L appears in P, is called the *Herbrand Base* (HB_P) of P. Let M be a module. We define $close_u(M)$ as the extended program that results if we replace every indexed literal in M with the special proposition u (meaning *undefined*). We say that M is ***internally consistent*** iff the *extended well-founded semantics* [PeAl92] of $close_u(M)$ is defined (the *WFM* of $close_u(M)$ is not contradictory). All modules of a *PMP* are assumed to be internally consistent.

If S is a set of literals then ${\sim}S = \{{\sim}L \mid L \in S\}$ and $\neg S = \{\neg L \mid L \in S\}$. The notations $Head_r$ and $Body_r$ denote the head and body of a rule r.

Example 3.1: A *contig* is a set of overlapping DNA fragments that span some region of a genome. One method to detect overlaps uses common features. The idea is that if the same feature is found in two DNA fragments then they probably overlap. The overlap is not certain because of unreliable experimental results and feature repetition in the genome. Consider the following *PMP* (before renaming) with modules *OF* (*OverlapFeature*), *OD* (*OverlapData*), *F* (*Feature*): (variables start with capital)



module *OF* **exports** *overlap* **imports** *feat*
/* If the same feature *Feat* is found in *Frag*1 and *Frag*2 then the fragments overlap */
rules *overlap*(*Frag*1,*Frag*2)← *feat*(*Frag*1,*Feat*), *feat*(*Frag*2,*Feat*).

module *OD* **exports** *overlap*
/* Very reliable *Overlap* data */
rules ¬*overlap*(*frag*1,*frag*2).

module *F* **exports** *feat*
/* A feature *x* is found in *frag*1 and *frag*2 */
rules *feat*(*frag*1, *x*). *feat*(*frag*2, *x*).

(*F*,*feat*) < (*OD*,*overlap*) /* *overlap* data in *OD* are more reliable than data in *F* */

Even though the modules *OD*, *OF* and *F* are internally consistent, their union is inconsistent because both *overlap*(*frag*1,*frag*2) and ¬*overlap*(*frag*1,*frag*2) can be derived from the above rules. Note that ¬*overlap*(*frag*1,*frag*2) is derived from module *OD* and that the derivation of *overlap*(*frag*1,*frag*2) from module *OF* depends on *feat*(*frag*1,*x*) and *feat*(*frag*2,*x*), exported by module *F*. Since (*F*,*feat*)< (*OD*,*overlap*), i.e., the definition of *feat* in *F* is less reliable that of *overlap* in *OD*, the results *feat*(*frag*1,*x*) and *feat*(*frag*2,*x*) are considered unreliable. Thus, the truth value of the literals *feat*(*frag*1,*x*) and *feat*(*frag*2,*x*) is considered unknown and the literal

$\neg overlap(frag1,frag2)$ is evaluated as true. After the renaming mechanism is employed, modules become:

module *OF* **exports** *overlap* **imports** *feat*
rules $OF\#overlap(Frag1,Frag2) \leftarrow \{F\}:feat(Frag1, Feat), \{F\}:feat(Frag2, Feat)$.

module *OD* **exports** *overlap*
rules $\neg OD\#overlap(frag1,frag2)$.

module *F* **exports** *feat*
rules $F\#feat(frag1, x)$. $F\#feat(frag2, x)$.

Definition 3.1 (interpretation): Let P be a *PMP*. An interpretation I is a set of literals $T \cup \sim F$, where T and F are disjoint sets of classical literals. I is *consistent* iff $T \cap \neg T = \emptyset$. I is *coherent* iff it satisfies the *coherence property*: $\neg T \subseteq F$.

In the above definition, T contains the *classically true* literals, $\neg T$ contains the *classically false* literals and the set F contains the literals *false by default*. The *coherence operator* (*coh*) [PeAl92] is used to transform an interpretation to a coherent one. If I is a set of literals then $\boldsymbol{coh}(I)=I \cup \{\sim L \mid \neg L \in I\}$.

Definition 3.2 (truth valuation of a literal): A literal L is true (resp. false) w.r.t. an interpretation I iff $L \in I$ (resp. $\sim L \in I$). A literal that is neither true nor false w.r.t. I, it is undefined w.r.t. I.

An interpretation I can be seen equivalently as a function from the set of classical literals to $\{0,1/2,1\}$, where $I(L)=1$ when L is true w.r.t. I, $I(L)=0$ when L is false w.r.t. I and $I(L)=1/2$ when L is undefined w.r.t. I. We define $I(\emptyset)=1$ and $I(S)= \min\{I(L) \mid L \in S\}$, where S is a non-empty set of literals.

The truth value of a literal $M\#L$ represents the truth value of L in module M. A classical literal $\{M_1,...,M_n\}:L$ is true iff $M_i\#L$ is true for an $i \leq n$ and $\{M_1,...,M_n\}:L$ is reliable. A default literal $\sim\{M_1,...,M_n\}:L$ is true iff $\sim M_i\#L$ is true for all $i \leq n$ and $\sim\{M_1,...,M_n\}:L$ is reliable. Let I be a set of literals known to be true. In Def. 3.5, the concept of reliable indexed literal is defined which is used in defining the *m*-models and in the fixpoint computation of the reliable semantics. To decide if an indexed literal $S:L$ is reliable w.r.t. a definition (M,p), all possible derivations of complementary literals caused by definitions $(M',p') \not< (M,p)$ should be considered. $S:L$ is reliable if it does not contribute to any such derivation of complementary literals. Intuitively, $\mathrm{Pos}_{[M,p],I}$ contains the literals possible to derive when results exported by modules M' with $(M',pred_L) < (M,p)$ are ignored.

Definition 3.3 (possible literal set): Let P be a *PMP*, I,J sets of literals and $(M,p) \in Def_P$. The *possible literal set* w.r.t. (M,p) and I, denoted by $\mathrm{Pos}_{[M,p],I}$, is defined as follows:

- $\mathbf{PT}_{[M,p],I}(J)=\{M'\#L \mid \exists\ M'\#L \leftarrow L_1,...,L_n$ in module M' s.t. $L_i \in J$, $\forall i \leq n\} \cup$
  $\{S:L \in HB_P \mid \neg S:L \notin I$ and $\exists\ M' \in S$ s.t. $M'\#L \in J$ and $(M',pred_L) \not< (M,p)\}$
- $\mathbf{PF}_{[M,p],I}(J)$ is the greatest set $U \subseteq HB_P$ s.t.
  (i) if $M'\#L \in U$ and r is a rule s.t. $Head_r = M'\#L$ then $\exists K \in Body_r$ s.t. $K \in U$ or $\sim K \in J$,
  (ii) if $S:L \in U$ then $\forall M' \in S$, $M'\#L \in U$ and $(M',pred_L) \not< (M,p)$.
- $\mathbf{PW}_{[M,p],I}(J) = coh(\mathbf{PT}_{[M,p],I}(J) \cup \sim\mathbf{PF}_{[M,p],I}(J))$.
- $\mathrm{Pos}_{[M,p],I}$ is the least (w.r.t. $\subseteq$) fixpoint of the operator $\mathbf{PW}_{[M,p],I}$.

Intuitively, a literal $S:L$ is reliable w.r.t. (M,p) and I iff there are no K, $\neg K$ in $\mathrm{Pos}_{[M,p],I}$ s.t. the derivation of K depends on $S:L$. The *dependency set* of K w.r.t. (M,p)

and I is the set of literals in $\mathrm{Pos}_{[M,p],I}$ that the derivation of K depends on. Since I is a set of literals known to be true, the *dependency set* of a literal $K \in I$ equals $\{\}$.

Definition 3.4 (dependency set): Let P be a *PMP*, I a set of literals and $(M,p) \in Def_P$. The *dependency set* of a literal K w.r.t. (M,p) and I, denoted by $\mathrm{Dep}_{[M,p],I}(K)$, is the least (w.r.t. $\subseteq$) set $\mathrm{D}(K)$ s.t. if $K \in I$ then $\mathrm{D}(K)=\{\}$ else

(i) If $K = {\sim}M'\#L$ is a default literal then

(a) $\mathrm{D}(\neg M'\#L) \subseteq \mathrm{D}({\sim}M'\#L)$ and

(b) $\forall\ M'\#L \leftarrow L_1,...,L_n$ in M', if ${\sim}L_i \in \mathrm{Pos}_{[M,p],I}$ for $i \leq n$ then $\mathrm{D}({\sim}L_i) \subseteq \mathrm{D}({\sim}M'\#L)$.

(ii) If $K = M'\#L$ is a classical literal then

if $M'\#L \leftarrow L_1,...,L_n$ in M' s.t. $\{L_1,...,L_n\} \subseteq \mathrm{Pos}_{[M,p],I}$ then $\mathrm{D}(L_i) \subseteq \mathrm{D}(M'\#L)$, $\forall i \leq n$.

(iii) If $K = {\sim}S{:}L$ is a default literal then

(a) $\mathrm{D}(\neg S{:}L) \subseteq \mathrm{D}({\sim}S{:}L)$ and

(b) if $\forall M' \in S$, $(M',pred_L) \not< (M,p)$ and ${\sim}M'\#L \in \mathrm{Pos}_{[M,p],I}$ then

$\mathrm{D}({\sim}M'\#L) \subseteq \mathrm{D}({\sim}S{:}L)$, $\forall M' \in S$ and ${\sim}S'{:}L \in \mathrm{D}({\sim}S{:}L)$, $\forall S' \subseteq S$.

(iv) If $K = S{:}L$ is a classical literal then

(a) $\mathrm{D}(M'\#L) \subseteq \mathrm{D}(S{:}L)$, $\forall M' \in S$ s.t. $(M',pred_L) \not< (M,p)$ and $S'{:}L \in \mathrm{D}(S{:}L)$, $\forall S' \subseteq S$.

Definition 3.5 (reliable indexed literal): Let P be a *PMP*, I a set of literals, $S{:}L$ a literal, $(M,pred_L) \in Def_P$, $M \in S$ and p be equal to $pred_L$.

- The literal $S{:}L$ is *unreliable* w.r.t. (M,p) and I iff $\exists\ \neg K \in \mathrm{Pos}_{[M,p],I}$ s.t. if $K = S'{:}L$ with $S' \supset S$ then $S{:}L \in \mathrm{Dep}_{[M,p],I}(M'\#L)$ for an $M' \in S'$ s.t. $(M',p) \not< (M,p)$ else $S{:}L \in \mathrm{Dep}_{[M,p],I}(K)$.
- The literal $S{:}L$ is *reliable* w.r.t. (M,p) and I iff it is not *unreliable* w.r.t. (M,p) and I.

Assume that $S{:}L$ is *unreliable* w.r.t. (M,p) and I. If K of Definition 3.5 is a local literal $M'\#L'$ then (i) the literals K, $\neg K$ are derived inside module M' and (ii) $S{:}L$ contributes to the derivation of K. If $K = S'{:}L' \neq S{:}L$ is an indexed literal then: (i) there are literals $M'\#L'$, $\neg M''\#L'$ derived in modules $M' \in S'$ and $M'' \in S'$ s.t. $(M',p) \not< (M,p)$ and $(M'',p) \not< (M,p)$, and (ii) $S{:}L$ contributes to the derivation of $M'\#L'$. If $K = S{:}L$ then there are literals $M'\#L$, $\neg M''\#L$ derived in modules $M' \in S$ and $M'' \in S$ with $(M',p) \not< (M,p)$ and $(M'',p) \not< (M,p)$.

Example 3.2: Let P be the *PMP* of Example 3.1 and $I = \emptyset$.

$\mathrm{Pos}_{[OD,overlap],I} = coh(\{\neg OD\#overlap(frag1,frag2),\ \ \neg\{OD\}{:}overlap(frag1,frag2),$
$\neg\{OD,OF\}{:}overlap(frag1,frag2)\})$

The literal $\neg\{OD\}{:}overlap(frag1,frag2)$ is reliable w.r.t. $(OD,overlap)$ and I because $\neg\{OD\}{:}overlap(frag1,frag2) \notin \mathrm{Dep}_{[OD,overlap],I}(\neg K)$, $\forall K \in \mathrm{Pos}_{[OD,overlap],I}$.

Similarly, $\neg\{OD,OF\}{:}overlap(frag1,frag2)$ is reliable w.r.t. $(OD,overlap)$ and I.

$\mathrm{Pos}_{[F,feat],I} = coh(\{F\#feat(frag1,x),\ \ \{F\}{:}feat(frag1,x),\ \ F\#feat(frag2,x),$
$\{F\}{:}feat(frag2,x),\ \ OF\#overlap(frag1,frag2),\ \ \{OF\}{:}overlap(frag1,frag2),$
$\{OD,OF\}{:}overlap(frag1,frag2),\ \ \neg OD\#overlap(frag1,frag2),$
$\neg\{OD\}{:}overlap(frag1,frag2),\ \ \neg\{OD,OF\}{:}overlap(frag1,frag2)\})$

The literal $\{F\}{:}feat(frag1,x)$ is unreliable w.r.t. $(F,feat)$ and I because

(i) $\neg\{OD,OF\}{:}overlap(frag1,frag2) \in \mathrm{Pos}_{[F,feat],I}$ and

(ii) $\{F\}$:$feat(frag1,x) \in \mathrm{Dep}_{[F,feat],I}(\{OD,OF\}$:$overlap(frag1,frag2))$.
Similarly, $\{F\}$:$feat(frag2,x)$ is unreliable w.r.t. $(F,feat)$ and I.

$\mathrm{Pos}_{[OF,overlap],I} = \mathrm{Pos}_{[F,feat],I}$. The literal $\{OF\}$:$overlap(frag1,frag2)$ is reliable w.r.t. $(OF,overlap)$ and I whereas the literal $\{OD,OF\}$:$overlap(frag1,frag2)$ is not.

Definition 3.6 (*m*-model): Let P be a *PMP*. A consistent, coherent interpretation I is an *m-model* of P iff (i) $\forall$ rule r, $I(Head_r) \geq I(Body_r)$ and (ii) $\forall$ classical literal S:L, both of the following are true:

– if $I(S{:}L) \neq 1$ then $\forall M \in S$ s.t. $I(M\#L)=1$, S:L is unreliable w.r.t. $(M,pred_L)$ and I

– if $I(S{:}L)=0$ then $I(\neg S{:}L)=1$ or $\forall M \in S$, $I(M\#L)=0$.

Since condition (i) defines 3-valued models [9], an *m*-model of P is a 3-valued model of every module of P. In condition (ii), the first subcondition expresses that if S:L is a classical literal, $M \in S$ and $I(M\#L)=1$ then $I(S{:}L)$ can be $\neq 1$ only if S:L is unreliable w.r.t. $(M,pred_L)$ and I. The purpose of the second subcondition in condition (ii) is to allow S:L to be false when $\neg S$:L holds, even if $\exists M \in S$ s.t. $I(M\#L)>0$.

Example 3.3: Let P be as in Example 3.1. Then, I is an *m*-model of P where
$I=coh(\{F\#feat(frag1,x),\ F\#feat(frag2,x),\ \neg OD\#overlap(frag1,frag2),$
$\neg\{OD\}{:}overlap(frag1,frag2),\ \neg\{OD,OF\}{:}overlap(frag1,frag2)\})$.

4. Reliable Semantics for Prioritized Modular Logic Programs

In this Section, we define the *reliable model*, *stable m-models* and *reliable semantics* of a *PMP*, P. We show that the reliable model of P is the least (w.r.t. $\subseteq$) stable *m*-model of P.

Definition 4.1 (*m*-unfounded set): Let P be a *PMP* and J a set of literals. A literal set $U \subseteq HB_P$ is *m-unfounded* w.r.t. J iff

(i) if $M\#L \in U$ and r is a rule with $Head_r = M\#L$ then $\exists K \in Body_r$ s.t. $K \in U$ or $\sim K \in J$,

(ii) if $S{:}L \in U$ then $\forall M \in S$, $M\#L \in U$ and $\sim S{:}L$ is reliable w.r.t. $(M,pred_L)$ and J.

The $\mathbf{W}_P$ operator extends the $\mathbf{W}_P$ operator of the *WFS* [11], to *PMPs*.

Definition 4.2 ($\mathbf{W}_P$ operator): Let P be a *PMP* and J a set of literals. We define:

- $\mathbf{T}(J)=\{M\#L \mid \exists\ M\#L \leftarrow L_1,...,L_n$ in module M s.t. $L_i \in J, \forall i \leq n\} \cup$
  $\{S{:}L \in HP_P \mid M\#L \in J, M \in S$ and $S{:}L$ is reliable w.r.t. $(M,pred_L)$ and $J\}$
- $\mathbf{F}(J)$ is the greatest *m*-unfounded set w.r.t. J.
- $\mathbf{W}_P(J)=coh(\mathbf{T}(J) \cup \sim\mathbf{F}(J))$.

The union of two *m*-unfounded sets w.r.t. J is an *m*-unfounded set w.r.t. J. So, $\mathbf{F}(J)$ is the union of all *m*-unfounded sets w.r.t. J. We define the transfinite sequence $\{I_a\}$ as follows: $I_0=\{\}$, $I_{a+1}=\mathbf{W}_P(I_a)$ and $I_a=\cup\{I_b \mid b<a\}$ if a is a limit ordinal.

Proposition 4.1: Let P be a *PMP*. $\{I_a\}$ is a monotonically increasing (w.r.t. $\subseteq$) sequence of consistent, coherent interpretations of P.

Since $\{I_a\}$ is monotonically increasing, there is a smallest ordinal d s.t. $I_d=I_{d+1}$.

Proposition 4.2: Let P be a *PMP*. Then, I_d is an *m*-model of P.

Definition 4.3 (reliable semantics): Let P be a *PMP*. The *reliable model* of P, RM_P, is the *m*-model I_d. The *reliable semantics* of P is the "meaning" represented by RM_P.

It is possible that a local literal $M\#L \in RM_P$ but $\{M\}{:}L \notin RM_P$. This, intuitively, means that module M concludes L but that conclusion may be erroneous.

Example 4.1: Let P be the program of Example 3.1 and I be the *m*-model of Example 3.3. Then, I is the reliable model of P. When $<_{def} = \{\}$, $RM_P = coh(\{F\#feat(frag1,x), F\#feat(frag2,x), \neg OD\#overlap(frag1,frag2)\})$.

Proposition 4.3: Let P be a *PMP*. The complexity of RM_P is polynomial w.r.t. $|P|$.

The reliable model of a *PMP* corresponds to its skeptical meaning. Credulous meanings can be obtained using the transformation $P/_m I$, where I is an interpretation of P. The transformation P/I is defined in [5, 9] for a normal program P.

Definition 4.4 (transformation $P/_m I$): Let P be a *PMP* and I an interpretation. The program $P/_m I$ is obtained from P as follows:

(i) Remove all rules that contain in their body a default literal $\sim L$ s.t. $I(L)=1$.
(ii) Remove any rule r s.t. $I(\neg Head_r)=1$.
(iii) Remove from the body of the remaining rules any default literal $\sim L$ s.t. $I(L)=0$.
(iv) Replace all remaining default literals $\sim L$ with u.
(v) For all $S{:}L \in HB_P$ s.t. $I(S{:}L)=1/2$,

– for all $M \in S$, if $I(M\#L)=1$ then add $S{:}L \leftarrow u$ else add $S{:}L \leftarrow M\#L$,
– if $\exists M \in S$ s.t. $\sim S{:}L$ is unreliable w.r.t. $(M, pred_L)$ and I then add $S{:}L \leftarrow u$.

(vi) For all $S{:}L \in HB_P$ s.t. $I(S{:}L) \neq 1/2$ and $I(\neg S{:}L) \neq 1$, add $S{:}L \leftarrow M\#L$, $\forall M \in S$.

We say that a stable *m*-model I of P is the *least*$_v$ *model* of P iff $I(L) \leq I'(L)$, for any stable *m*-model I' and classical literal L of P.

Definition 4.5 (stable m-model): Let P be a *PMP* and I an *m*-model of P. I is a *stable m-model* of P iff $least_v(P/_m I)=I$.

Let I be a stable *m*-model of P. If $S{:}L$ is unreliable w.r.t. $(M, pred_L)$ and I, for an $M \in S$ then the truth value of $S{:}L$ can be unknown w.r.t. I even if $I(M\#L)=1$. If $\sim S{:}L$ is unreliable w.r.t. $(M, pred_L)$ and I, for an $M \in S$ then the truth value of $S{:}L$ can be unknown w.r.t. I even if $I(M\#L)=0$, for all $M \in S$.

The *export rule set* of P is defined as $ER_P = \{S{:}L \leftarrow M\#L \mid S{:}L \in HB_P$ and $M \in S\} \cup \{\sim S{:}L \leftarrow \sim M_1\#L, ..., \sim M_n\#L \mid S{:}L \in HB_P$ and $S = \{M_1, ..., M_n\}\}$. An interpretation I of P satisfies $r \in ER_P$ iff $I(Body_r) \neq 1$ or $I(Head_r)=1$. Let I_1, I_2 be two stable *m*-models of P. We say that $I_1 \leq_{sat} I_2$ iff (i) $\forall r \in ER_P$, if I_1 satisfies r then I_2 satisfies r and (ii) $I_1 \subseteq I_2$ or $\exists r \in ER_P$ s.t. I_2 satisfies r and I_1 does not satisfy r. In other words, $I_1 \leq_{sat} I_2$ iff I_2 satisfies more export rules than I_1 or ($I_1 \subseteq I_2$ and I_2 satisfies the same export rules as I_1). Maximal (w.r.t. $\leq_{sat}$) stable *m*-models can be seen as the credulous meanings of P.

Example 4.2: Let P be the program of Example 3.1. Then P has four stable *m*-models:
$I_1 = RM_P$, $\quad I_2 = RM_P \cup coh(\{\{F\}{:}feat(frag1,x)\})$,

I_3=$RM_P \cup coh(\{\{F\}{:}feat(frag2,x)\})$ and I_4=$RM_P \cup coh(\{\{F\}{:}feat(frag1,x)$, $\{F\}{:}feat(frag2,x), OF\#overlap(frag1,frag2), \{OF\}{:}overlap(frag1,frag2)\})$.
I_2, I_3 and I_4 are maximal (w.r.t. $\leq_{sat}$) stable *m*-models of *P*. Note that
Model I_2 does not satisfy $\{F\}{:}feat(frag2,x) \leftarrow F\#feat(frag2,x)$,
Model I_3 does not satisfy $\{F\}{:}feat(frag1,x) \leftarrow F\#feat(frag1,x)$ and
Model I_4 does not satisfy $\{OD,OF\}{:}overlap(frag1,frag2) \leftarrow OF\#overlap(frag1,frag2)$.

Proposition 4.4: Let *P* be a *PMP*. The reliable model of *P* is a stable *m*-model of *P*.

Proposition 4.5: The reliable model of a *PMP* is its least (w.r.t. $\subseteq$) stable *m*-model.

According to *RS* presented in the previous sections, the confidence in a globally indexed default literal $\sim L$, derived by the default rule for negation, depends on the minimal priorities of $(M, pred_L) \in Def_P$. Thus, in case of conflict, $\sim L$ may not be considered less reliable than literals that their derivation is not based on closed-world assumptions. When this is undesirable for a set of predicates $Pred_\sim$, a new module $M_\sim$ can be added which has no rules but exports all predicates in $Pred_\sim$. Moreover, $(M_\sim, p') < (M,p)$ for all $p' \in Pred_\sim$ and definitions (M,p) other than $(M_\sim, p)$.

An *extended program with rule prioritization* (*EPP*) is naturally translated into a *PMP* by considering each rule as a module that imports the predicates appearing in its body and exports its head predicate. Thus, we have also defined the reliable semantics for EPPs. The *RS* for *EPPs* extends the *well-founded semantics* for normal programs [11] and *extended well-founded semantics* for extended programs [7].

6. Related Work

The *contradiction removal semantics* (*CRS*) for extended programs [8] avoids contradictions brought about by closed world assumptions. For example, the *CRS* of $P=\{\neg p \leftarrow \sim a.\quad p \leftarrow.\quad b \leftarrow.\}$ is $\{p, b\}$ which is non-contradictory. Yet, contradictions not based on closed world assumptions cannot be resolved. For example, nothing is concluded from $P' = \{\neg p \leftarrow.\quad p \leftarrow.\quad b \leftarrow.\}$ though *b* should be true. The same is true for the semantics in [3, 12]. However, the $RM_{P'} = \{\}$.

The *conservative reasoning* for extended programs, presented in [13], is as follows: if *r* is a rule and $Body_r$ is true then $Head_r$ is true iff for every rule r' s.t. $Head_{r'} = \neg Head_r$, $Body_{r'}$ cannot be derived. For example, the conservative semantics of $P = \{r_1{:}\ \neg a \leftarrow b.\quad r_2{:}\ a.\quad r_3{:}b.\}$ is $\{b\}$. In *RS*, r_3 is considered unreliable and $RM_{P'} = \{\}$.

Prioritization of rules is investigated in the various variations of *ordered logic* [4, 6]. Even though these semantics are defined for all ordered logic programs, negation by default is not supported. Moreover, the rule ordering in ordered logic represents *exceptions* and not *reliability*. For, example, the ordered logic semantics of $P = \{r_1{:}\ \neg a \leftarrow b.\quad r_2{:}\ a.\quad r_3{:}b.\}$ with $r_3 < r_2 < r_1$ is $\{b\}$ where as $RM_{P'} = \{a, \sim\neg a\}$ since $r_3 < r_2$. When the prioritization of the rules is ignored, ordered logic and conservative reasoning [13] behave similarly. Prioritization of rules is also supported in [2]. However, there rules are considered to be clauses, i.e., there is no distinction between the head and the body of a rule. Thus, in [2], program $P' = \{p \leftarrow \sim p.\}$ is considered equivalent with $\{p.\}$ whereas the *WFM* of *P'* is {}. The semantics of the above program *P* according to [2] coincides with $RM_{P'}$.

In [10], local *DBs* are combined with a supervisory *DB* in a framework based on *annotated logic*. However, though the supervisory *DB* can access literals defined in the local *DBs*, local *DBs* can access only local information. The resolution of conflicts between the local *DBs* is the responsibility of the supervisory *DB*.

8. Conclus ions

We have presented the reliable semantics (*RS*) of *prioritized modular logic programs* (*PMPs*). The purpose of the reliable semantics is to derive reliable information from contradictory *PMPs*. Every *PMP* has at least one *stable m-model*. The *reliable model* of a program *P* is the least (w.r.t. $\subseteq$) stable *m*-model of *P* and it represents the skeptical "meaning" of *P*. Maximal (w.r.t. $\leq_{sat}$) stable *m*-models of *P* represent the credulous "meanings" of *P*.

REFERENCES

[1] A. Bossi, M. Gabbrielli, G. Levi, M.C. Meo, "Contributions to the Semantics of Open Logic Programs", Proc. of the Intern. Conference of Fifth Generation Computer Systems (FGCS'92), 1992, pp. 570-580.

[2] S. Brass, U.W. Lipeck, "Semantics of Inheritance in Logical Object Specifications", 2nd Intern. Conf. on Deductive and Object-Oriented Databases, 1991.

[3] P.M. Dung, "An Argumentation Semantics for Logic Programs with Explicit Negation", 10th Intern. Conf. on Logic programming (ICLP'93), 1993, pp.616-630.

[4] D. Gabbay, E. Laenens, D. Vermeir, "Credulous vs. Sceptical Semantics for Ordered Logic Programs", Proc. of the 2nd International Conference on Knowledge Representation and Reasoning (KR'91), 1991, pp. 208-217.

[5] M. Gelfond, V. Lifschitz, "The Stable Model Semantics for Logic Programming", Proc. of the 5th Symposium on Logic Programming, 1988.

[6] E. Laenens, D. Vermeir, "Assumption-free Semantics for Ordered Logic Programs: On the Relationship Between Well-founded and Stable Partial Models", Journal of Logic and Computation, 2(2), 1992, pp. 133-172.

[7] L.M. Pereira, J.J. Alferes, "Well-founded Semantics for Logic Programs with Explicit Negation", 10th European Conf. on Artificial Intelligence, 1992, pp. 92-96.

[8] L.M. Pereira, J.N. Aparicio, J.J. Alferes, "Nonmonotonic Reasoning with Logic Programming", Journal of Logic Programming, 17, 1993, pp. 227-263.

[9] T. Przymusinski, "Well-Founded Semantics Coincides with the Three-Valued Stable Semantics", Fundamenta Informaticae XIII, 1990, pp. 445-463.

[10] V.S. Subrahmanian, "Amalgamating Knowledge Bases", to be published in ACM Transactions on Database Systems (TODS), 1994.

[11] A. van Gelder, K.A. Ross, J.S. Schlipf, "The Well-Founded Semantics for General Logic Programs", Journal of the ACM, 38(3), July 1991, pp.620-650.

[12] C. Witteveen, "Expanding Logic Programs", D. Pearce and G. Wagner (eds). Logics in AI, LNCS 633, 1992, pp. 372-390.

[13] G. Wagner, "Reasoning with Inconsistency in Extended Deductive Databases", Proc. of the 2nd Intern. Workshop on Logic Programming and Non-monotonic Reasoning, 1993, pp.300-315.

[14] C.L. Mason, R.R. Johnson, "DATMS: A Framework for Distributed Assumption Based Reasoning", Distributed Artificial Intelliegence, 2, 1989, pp. 293-317.