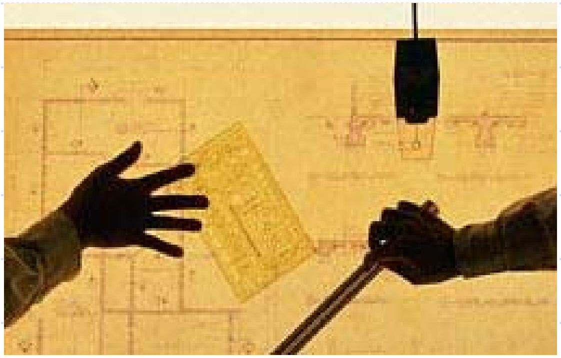
Semantics of Prioritized Default Rules System



Theory & Formal Methods Lab.
Dept. of CSE, Korea University
Hee-Jun Yoo
hyoo@formal.korea.ac.kr

LiComR, SongGwangSa, August 2003

Content

- Introduction
- Background
  - Extended Logic Programs
  - Default Rules
  - Prioritized Default Rules
- Previous Approach for Prioritized Defaults Rules
- Motivated Example
- Our Approach
- Conclusion and Future Works
- Reference

Introduction

- Extended Logic Program is *contradictory* if it has inconsistent answer set.
  - Answer Set is *inconsistency* if it contains a pair of complementary literals.
- Default Rules are described by extended logic notation.
- Most of Default Rules Semantics deal with such a inconsistency as this.
- Problem
  - Default Rules could have another inconsistency during making the consistent answer set.
- We suggest that example occurs a problem.
- We define a new semantics for solving above problem.

Background

- negation as failure - *not*
  - $\neg Q \leftarrow$ *not* P
  - Q is false, if there is is no evidence that P is true.

- Extended logic Program (Π) (*Baral's Representation*)
  - A collection of rules of the form

    $L_0 \leftarrow L_1, \ldots, L_m,$ *not* $L_{m+1}, \ldots,$ *not* L_n

    - L : literals, *i.e.* formulas of the form p or $\neg p$
    - p is an atom.
  - *Lit* : the set of all literals in the language of Π.
  - *Lit(p)* : the collection of ground literals formed by the predicate p.

Background (cont'd)

- Answer sets

  : sets of literals corresponding to beliefs which can be built by a rational reasoner on the basis of Π.

- The answer set of Π not containing *not* is the smallest subset S of Lit such that
  - For any rule $L_0 \leftarrow L_1, \ldots, L_m$ from Π, if $L_1, \ldots, L_m \in S$, then $L_0 \in S$;
  - If S contains a pair of complementary literals, then $S = Lit$.

Background (cont'd)

=⊨⊑≠∨⊢⇒ ⊤∃¬ = ⊥℘⇔⁻=⊨⊑≠∨⊢⇒ ⊤∃¬ = ⊥℘⇔⊣=⊨⊑≠∨⊢⇒ ⊤∃¬ = ⊥℘⇔⊣

Default Reasoning

- Defaults are statements containing words "normally, typically, as a rule".
- A large part of our education seems to consists of learning various defaults, their exceptions, and the skill of reasoning with them.
- Defaults do not occur in the language of mathematics, and therefore were not studied by classical mathematical logic.
- Reiter's notation

$\mathrm{BIRD}(x) \wedge \neg\mathrm{PENGUIN}(x) \wedge \neg\mathrm{OSTRICH}(x) \wedge \ldots \supset \mathrm{FLY}(x)$

$$\frac{\mathrm{BIRD}(x) : \mathrm{M}\,\mathrm{FLY}(x)}{\mathrm{FLY}(x)}$$

"M" is to be read as "it is consistent to assume".

Background (cont'd)

- Prioritized Default Reasoning (*Gelfond's Representation*)
  - Language $L_0(\sigma)$
    - parameterized by a multi-sorted signature σ.
    - containing names for objects, functions, and relations of the user's domain.
  - σ contains two special collections of terms of the language
    - name defaults.
    - strict(non-defeasible) rules.
  - d_i : defaults of $L_0(\sigma)$
  - r_i : rules of $L_0(\sigma)$

Background (cont'd)

- Prioritized Default Reasoning (*Gelfond's Representation*)
  - d, d_1, d_2 are defaults names, l_0, ..., l_n are literals of $L_0(\sigma)$ and [ ] is the list operator

$rule(r, l_0, [l_0, \ldots, l_m]);$

$default(d, l_0, [l_0, \ldots, l_m]);$

$conflict(d_1, d_2);$

$prefer(d_1, d_2);$

are literals of $L_0(\sigma)$.

Theory & Formal Methods

Background (cont'd)

- Prioritized Default Reasoning Example
  - defaults
    $default(d_1, p, [\,]);$
    $default(d_2, q, [r]);$
  - rules
    $rule(r_1, \neg p, [q]);$
    $rule(r_2, \neg q, [p]);$
  - fact
    r
  - conflict
    $conflict(d_1, d_2)$
  - prefer
    $prefer(d_1, d_2)$

$$p \leftarrow not\ \neg p$$
$$q \leftarrow r,\ not\ \neg q$$
$$\neg p \leftarrow q$$
$$\neg q \leftarrow p$$
$$r \leftarrow$$
$$conflict(d_2, d_1) \leftarrow$$
$$prefer(d_2, d_1) \leftarrow$$

Previous Approach (by Gorosof)

- Augment ELP syntax and modify ELP semantics :
  - Add optional label (name) to each rule.
  - Include prioritization facts of form *Overrides(i, j)* ←
    - *Overrides* is a special reserved predicate.
    - *Overrides(i, j)* means *i* has higher priority than *j*.
    - Overrides is a strict partial order on labels.
  - Locale is “definition” of one ground atom.

    Locale(p) = {all rules whose head is p or ¬p}
  - Stratify the CLP into locales. (Dependency graph)
    - The answer set is defined via constructive induction along the stratification.
  - *Prioritization Sub-Program* is defined as set of positive ground facts about *Overrides*.

Previous Approach (*Cont'd*)

- Syntax
  - *DLP* : Default Logic Program
    - $DLP = DLP_{main} \cup DLP_{Overrides}$
    - *DLP* is required to be *acyclic*.
  - DLP^{instd}
    - *DLP* that results when rule in *DLP* having variables has been replaced by set of all its possible ground instantiations.
  - $\rho = p_1, \ldots, p_m$
    - ρ be a sequencing of all the ground atoms of DLP^{instd}.
    - ρ be a total stratification of the atoms when ρ is a reverse-direction topological sort of atom dependency graph.
    - p_i stand for the i^{th} (ground) atom in this sequence ρ.

Previous Approach (*Cont'd*)

=⊨⊑≠∨⊢⇒⊤∃¬=⊥℘⇔⁻=⊨⊑≠∨⊢⇒⊤∃¬=⊥℘⇔⊣=⊨⊑≠∨⊢⇒⊤∃¬=⊥℘⇔⊣

- Sematics
  - The answer set is constructed iteratively :

$$S_0 = \emptyset$$

$$S_i = \bigcup_{j=1,\dots,i} T_j, \quad i \geq 1$$

$$S = \bigcup_i T_i$$

σ : classical sign

label(*j*,*r*) : *j* is label for rule *r*

Head(*r*), *body*(*r*) : head or body of rule *r*

$$T_i = \left\{ \sigma p_i \mid Cand_i^{\sigma} \neq \emptyset, \forall k \in Cand_i^{\boxed{\neg\sigma}} . \exists j \in Cand_i^{\boxed{\sigma}} . S_{i-1} \models Overrides(j,k) \right\}$$

$$Cand_i^{\sigma} = \{ j \mid label(j,r), Head(r) = \sigma p_i, S_{i-1} \models Body(r) \}$$

Motivated Example

- Some Definition by International Agreement
  - One creature has exactly one species name.
  - There are no life on this planet that has two species name.
  - Two different species has no same name.

- Motivated Example

<Wat> Fishes(?ani) ← LiveInWater(?ani)

<Pla> Mammal(?ani) ← HasPlacenta(?ani)

Overrides (Pla, Wat) ←

LiveInWater(whale) ←

HasPlacenta (whale) ←

Our approach

- Object of Approach
  - Ensure consistency
  - Unique answer set; thus conceptually simple
  - Simple to specify override (priorities)
  - Inferencing tractable
  - Include consistent “extended” LP and “general” LP (acyclic)

Theory & Formal Methods

Our approach (*Cont'd*)

=⊨⊑≠∨⊢⇒ ⊤∃¬ = ⊥℘ ⇔⁻=⊨⊑≠∨⊢⇒ ⊤∃¬ = ⊥℘ ⇔⊣=⊨⊑≠∨⊢⇒ ⊤∃¬ = ⊥℘ ⇔⊣

- Definition
  - A partition $\pi_0, \ldots, \pi_k$ of the set of all predicate symbols of a default logic program *DLP* is a ***stratification*** of *DLP*, if for any rule of the definite type and for any $p \in \pi_s$, $0 \le s \le k$ if $L_0 \in atoms(p)$, then :
    1. for every $1 \le i \le m$ there is q and $j \le s$ such that $q \in \pi_j$ and $L_i \in atoms(q)$
    2. for every $m+1 \le i \le n$ there is q and $j < s$ such that $q \in \pi_j$ and $L_i \in atoms(q)$
  - A program is called ***stratified*** if it has a stratification.

Our approach (*Cont'd*)

- Dependency graph (D_{DLP}) is consist of
  - Vertices : predicate names.
  - Labeled edge : $\langle P_i, P_j, s \rangle$
    - P_i : head of rule
    - P_j : body of rule
    - s : label $s \in \{ +, - \}$
      - denoting whether P_j appears in positive or a negative literal in the body of *r*.
  - Negative cycle
    - Dependency graph has cycle if it contains at least one edge with negative label.

Our approach (*Cont'd*)

- Some Definitions
  - Ground : Formulas and rules not containing variables.
  - *HB*(*DLP*) – Herbrand base of program(*DLP*)
    - Set of all ground atoms in the language of a program(*DLP*).
  - Stable model of a definite program *DLP*
    - is the smallest subset S of HB such that for any rule $L_0 \leftarrow L_1, \dots, L_m$ from Π, if $L_1, \dots, L_m \in S$, then $L_0 \in S$.
  - DLP^S be a program obtained from *DLP* by deleting
    1. each rule that has a formula *not L* in its body with $L \in S$.
    2. all formulas of the form *not L* in the bodies of the remaining rules.
  - The programs are called *categorical*, if it have a unique stable model.

Our approach (*Cont'd*)

- Proposition 1
  - A default logic program is stratified iff its dependency graph D_{DLP} does not contain any negative cycles.
- Proposition 2
  - Any stratified default logic program is categorical.
- Theorem 3
  - A consistent logic program whose dependency graph does not have a cycle with only positive edges has at least one stable model.
- Lemma 4
  - For any stable model S of a default logic program *DLP*.
    1. For any ground instance of a rule of the definite type from *DLP*, if $\{L_1, \ldots, L_m\} \subseteq \{L_{m+1}, \ldots, L_n\} \cap S = \emptyset$ then $L_0 \in S$.
    2. If S is stable model of *DLP* and $L_0 \in S$, then there exists a ground instance of a rule of the definite type from *DLP* such that $\{L_1, \ldots, L_m\} \subseteq \{L_{m+1}, \ldots, L_n\} \cap S = \emptyset$ then $L_0 \in S$.

Our Approach (*Cont'd*)

- Syntax
  - DLP : Default Logic Program
    - DLP = $DLP_{main} \cup DLP_{Overrides}$
    - DLP is required to be *acyclic*.
  - DLP^{instd}
    - DLP that results when rule in DLP having variables has been replaced by set of all its possible ground instantiations.
  - $\rho = p_1, \ldots, p_m$
    - ρ be a sequencing of all the ground atoms of DLP^{instd}.
    - ρ be a total stratification of the atoms when ρ is a reverse-direction topological sort of atom dependency graph.
    - p_i stand for the i^{th} (ground) atom in this sequence ρ.

Our approach (Cont'd)

- Sematics
  - The answer set is constructed iteratively :

$$S_0 = \emptyset$$

$$S_i = \bigcup_{j=1,\ldots,i} T_j, i \geq 1$$

$$S = \bigcup_i T_i$$

$$Cand_i^{\sigma} = \{ j \mid label(j,r), Head(r) = \sigma p_i, S_{i-1} \models Body(r)\}$$

$$T_i = \{< \sigma p_i, r_i > \mid Cand_i^{\sigma} \neq \emptyset, \forall k \in Cand_i^{\neg\sigma}. \exists j \in Cand_i^{\sigma}. S_{i-1} \models Overrides(j,k)\}$$
$$\cup \{< \sigma p_i, r_i > \mid \forall j. label(j,r), Head(r) = \neg\sigma p_i,$$
$$\exists k. label(k,s), S_{i-1} \models Body(s), Overrides(k,j)\}$$

Conclusion and Future Works

Conclusion

- We show new inconsistent case that may happen in default logic program.
- We define a new semantics of default logic program for solving given inconsistent case.
- Our semantics keeps unique consistent answer set for inconsistency that is defined by Gelfond and Lifschitz.

Future Works

- We will extend our semantics to solve problem that is occurred when complementary literal overrides each other repeatedly.
- We will convert our default logic program to relational algebra.

Reference

1. A Logic for Default Reasoning, R. Reiter, Artificial Intelligence, 12:81 – 132, 1980.
2. Foundations of Logic Programming (second, extended edition), J.W. Lloyd, Springer-verlag, 1987.
3. Classical Negation in Logic Programs and Disjunctive Databases, Michael Gelfond, Vladimir Lifschitz, New Generation Computing, 1991.
4. Logic Programming and Knowledge Representation, Chitta Baral, Michael Gelfond Journal of Logic Programming, 1994.
5. Courteous Logic Programs : Prioritized Conflict Handling For Rules, Benjamin N. Grosof, IBM Research Report RC 20836, 1997.
6. Reasoning with Prioritized Defaults, Michael Gelfond, Tran Cao Son Lecture Notes in Computer Science, 1998.
7. Compiling Prioritized Default Rules into Ordinary Logic Programs, Benjamin N. Grosof, IBM Research Report RC 21472, 1999.