PLPの優先的解集合計算プログラムの開発と評価*

黄檗洋輔 若木利子†
芝浦工業大学 システム工学部 電子情報システム学科‡

1 はじめに

優先度付き論理プログラム [3](Prioritized Logic Program, 略して PLP) は論理プログラムに優先度情報を付与したものであり, その意味論は優先的解集合 (Preferred Answer Sets) で与えられる. PLP はプリファレンスを扱う非単調推論やアブダクションの枠組みの表現が可能である. 最近, 解集合プログラミング (ASP: Answer Set Programming) により, PLP の優先的解集合を求める健全かつ完全な手続き (*CompPAS*) が提案された [4]. この手続きは, PLP に静的プリファレンスのみならず動的プリファレンスも扱うことを可能にして, PLP の意味論を拡張している. 本研究では, 当該手続きに基づき, ASP ソルバの *dlv*[5] 及び C++を用いて PLP の優先的解集合を効率的に計算する優先的解集合計算プログラム compPLP を開発し, 種々の非単調推論問題への適用と評価を行った.

2 優先度付き論理プログラム：PLP

2.1 PLP の構文

PLP[3] は宣言的知識を (P, Φ) で表現する. ここで P は, 以下の形式のルールからなる GEDP (General Extended Disjunctive Program) の論理プログラムである.

$$L_1 \mid \ldots \mid L_k \mid notL_{k+1} \mid \ldots \mid notL_l \leftarrow L_{l+1} \mid \ldots \mid L_m \mid notL_{m+1} \mid \ldots \mid notL_n$$

(但し, L_i はリテラル, not は NAF, $n \geq m \geq l \geq k \geq 0$)

Φ は以下で定義される Lit_P(P の言語のリテラル集合) のリテラル間のプライオリティの集合として定義される.

定義 1 $L^* = Lit_P \cup \{not\ L | L \in Lit_P\}$ とする. $e_1, e_2 \in L^*$ について, e_2 が e_1 よりも高い優先度を持つ時,$e_1 \preceq e_2$ と表して, これを"プライオリティ"と称する. $e_1 \preceq e_2$ かつ $e_2 \not\preceq e_1$ である時, $e_1 \prec e_2$ と表す.

2.2 PLP の意味論

所与の PLP(P, Φ) において, P の解集合 [1] の集合を AS とし, Φ の反射・推移閉包を Φ^* とすると, AS 上のプリファレンス関係 ($\sqsubseteq$) は以下の (i),(ii) で定義される.

(i)$S_1, S_2, S_3 \in AS$ に対して, $S_1 \sqsubseteq S_2$ if

$$\exists e_2 \in S_2 \backslash S_1\ [\ \exists e_1 \in S_1 \backslash S_2 \text{ such that } (e_1 \preceq e_2) \in \Phi^* \wedge \neg \exists e_3 \in S_1 \backslash S_2 \text{ such that } (e_2 \prec e_3) \in \Phi^*]$$

(ii)$\sqsubseteq$ は反射律, 推移律を満たす.

PLP の意味論は, AS 上のプリファレンス関係 $\sqsubseteq$ を用いて, 以下の優先的解集合で与えられる [3].

定義 2 PLP(P,Φ) において, S, S' を P の解集合とする. S が任意の S' について *($S \sqsubseteq S'$ ならば $S' \sqsubseteq S$)* の時, S は (P,Φ) の優先的解集合 *(preferred answer set)* である.

*Implementing the procedure of computing preferred answer sets of PLP

†Yosuke Kiwada, Toshiko Wakaki
‡Shibaura Institute of Technology

3 優先的解集合の計算手続き:*CompPAS*

[4] のアプローチでは, 所与の PLP (P,Φ) と P の任意の解集合 S から変換論理プログラム $T[P,\Phi,S]$ を構成し, その解集合よりプリファレンス ($\sqsubseteq$) を生成する. この $T[P,\Phi,S]$ について, 優先的解集合計算手続きの健全性と完全性を保証する以下の定理が成り立つ. (紙面の都合上,$T[P,\Phi,S]$ の詳細は省略)

定理 1 [4] S を P の任意の解集合とする. 無矛盾な $T[P,\Phi,S]$ の任意の解集合を E とすると, $S' \stackrel{def}{=} E \cap Lit_P$ なる $S'(\neq S)$ について, $S \sqsubseteq S'$ が成り立つ. 逆に, $S \sqsubseteq S'$ なる P の解集合 $S'(\neq S)$ が存在すれば, $T[P,\Phi,S]$ は無矛盾である.

この手続き [4] ではメタプログラミングを用いており, 生成検査法で優先的解集合を判定している (図.1).

procedure $CompPAS(P,\Phi,\Delta)$
入力: PLP(P,Φ)
出力: (P,Φ) の全ての優先的解集合の集合 Δ

Step1 P の全ての解集合を計算し,AS に格納.

Step2 Φ が空集合ならば,$\Delta := AS$ を *return*
さもなければ,

(a) $|AS|$ の数の新固体定数 s_i を導入
s_i からなる集合を Ω とする

(b) 各解集合 $S \in AS$ に識別子 $s \in \Omega$ を対応付ける
但し, $\Omega := \{s_i | 1 \leq i \leq |AS|\}$

Step3 任意の解集合 $S \in AS$ について $T[P,\Phi,S]$ が無矛盾ならば, $T[P,\Phi,S]$ の全ての解集合 E について以下を実行.

(a) $S' := E \cap Lit_p$ の識別子 s' を求める.

(b) $\Sigma := \Sigma \cup \{\sqsubseteq (s,s') \leftarrow\}$

Step4 $\Psi \cup \Sigma \cup \{as(s) \leftarrow | s \in \Omega\}$ の解集合 U を計算

Step5 集合 Δ を計算し,Δ を *return*
$\Delta \stackrel{\text{def}}{=} \{S \in AS | S$ は $p\text{-}as(s) \in U$ であるような識別子 s の解集合 $\}$

図 1: 優先的解集合の計算手続き:*CompPAS*

$\sqsubseteq (x,x) \leftarrow as(x)$,
$\sqsubseteq (x,z) \leftarrow \sqsubseteq (x,y),\ \sqsubseteq (y,z)$,
$\sqsubset (x,y) \leftarrow \sqsubseteq (x,y),\ not \sqsubseteq (y,x)$,
$worse(x) \leftarrow \sqsubset (x,y)$,
$p\text{-}as(\mathrm{x}) \leftarrow as(x),\ not\ worse(x)$.

図 2: Ψ のルール集合

4 優先的解集合計算プログラム:compPLP

図 1 の *CompPAS* の手続きに従い, *dlv*[5] と C++を用いて PLP の優先的解集合を計算するプログラム:compPLP を開発した. compPLP のデータ構造として, 述語記号表は述語記号をキーとしたハッシュ表として実現し (図 3), これにより,Lit_P のリテラルに関する高速な探索が可能となる. また, 引数表を作成する段階で用いる固体定数表を作成する際には, 二分探索木を構築して高速化を図った.

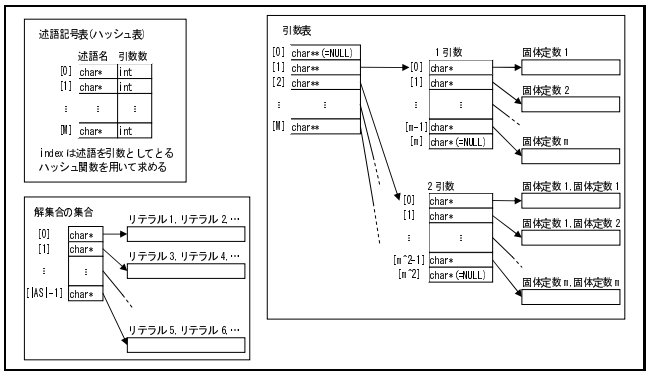

図 3: 述語記号表と解集合のデータ構造

5 法的推論への応用

PLP の優先的解集合計算プログラム compPLP は, 現在, linux, 及び Windows 環境下の ASP ソルバ：*dlv* 上で稼動している．PLP では静的プリファレンスしか扱えなかったが, 動的プリファレンスを扱う法的推論の例 [2] が図 1 の手続き [4] を実現した本プログラムで計算できることを以下に示す.

5.1 法的推論の例

Gorden の法的推論の問題 [2] を考える.

(法的問題)
ある人が船を所有しているが, 船の登記をしていない。
彼にその船の担保権があるか否か？

ここで米国に, UCC(Uniform Commercial Code) と SMA(Ship Mortgage Act) という 2 つの法令が存在する.

UCC: 品物を所有している人に品物の担保権が有る。
SMA: 船を所有していても登記していなければ担保権は無い。

上記の知識は以下の論理プログラム P で表現される．*dlv* により計算された P の解集合 (*answer sets*) を図 4 に示す.

< 論理プログラム: P>

```
perfected :- posses, not ab1.
-perfected :- ship, -filstate, not ab2.
posses.
ship.
-filstate.
ab1 v n_ab1.
ab2 v n_ab2.
:- ab1, ab2.
ucc :- not ab1.
sma :- not ab2.
```

```
C:\CompPLP>dlv -silent legalP
{posses, ship, -filstate, ab1, -perfected, n_ab2, sma}
{posses, ship, -filstate, perfected, ab2, n_ab1, ucc}

C:\CompPLP>
```

図 4: dlv による論理プログラム P の解集合計算

P の 2 つの解集合 (図 4) を以下の S_1, S_2 とする.
$S_1 = \{posses, ship, -filstate, ab1, -perfected, nab2, sma\}$
$S_2 = \{posses, ship, -filstate, perfected, ab2, nab1, ucc\}$
よって, $\neg perfected \in S_1$, $perfected \in S_2$ (1)
(1) は UCC と SMA の 2 つの法令が互いにコンフリクトして, “担保権が有るか無いか?” が決定できないことを示す．このような法令間のコンフリクトを解消するために,

新法優先 (Lex Posterior): より新しい法令を優先する。
上位法優先 (Lex Superior): より上位の法令を優先する。

などの法的原則 (*legal principle*) が現実に法律家等に用いられている．次に, 以下の事実が存在したとする.

UCC は SMA より最近制定された。 SMA は連邦法, UCC は州法である。連邦法は州法よりも上位の法律である。

これらの法令間の優先関係の知識は, PLP の Φ を拡張して, 条件付きプライオリティをルール表現した層状論理プログラム $\tilde{\Phi}_1$ として表現される. この結果, $(P, \tilde{\Phi}_1)$ の優先的解集合が,compPLP を用いて図 5 のように計算される.

< 優先情報: $\tilde{\Phi}_1$>

```
perfected :- posses , not ab1.
moreRecent(ucc,sma).
fed(sma).
state(ucc).
lp(Y,X) :- moreRecent(X,Y).
ls(Y,X) :- fed(X) , state(Y).
preceq(Y,X) :- lp(Y,X), not conf1(X,Y).
preceq(Y,X) :- ls(Y,X), not conf1(X,Y).
%conf1(Y,X) :- lp(X,Y), ls(Y,X), not conf2(X,Y).
```

```
C:\CompPLP>CompPLP legal
Total Number of Answer Sets of P:2
{-filstate, -perfected, ab1, n_ab2, posses, ship, sma}
{-filstate, ab2, n_ab1, perfected, posses, ship, ucc}

Total Number of Preferred Answer Sets of (P,Phi):2
{-filstate, -perfected, ab1, n_ab2, posses, ship, sma}
{-filstate, ab2, n_ab1, perfected, posses, ship, ucc}

C:\CompPLP>
```

図 5: compPLP による PLP$(P, \tilde{\Phi}_1)$ の計算結果

再び, S_1, S_2 が優先的解集合として得られ, 担保権の有無について決定できない。今回は上位法優先と新法優先が互いにコンフリクトしているためであるので, これらの法的原則間に以下のメタな優先関係を新たに追加する.

新法優先より上位法優先を優先する.

上記のメタな優先情報を追加したものを $\tilde{\Phi}_2$ とすると,
$\tilde{\Phi}_2 = \tilde{\Phi}_1 \cup$
$\{conf_1(Y,X) \leftarrow lp(X,Y), ls(Y,X), not\ conf_2(X,Y).\}$

```
C:\CompPLP>CompPLP legal2
Total Number of Answer Sets of P:2
{-filstate, -perfected, ab1, n_ab2, posses, ship, sma}
{-filstate, ab2, n_ab1, perfected, posses, ship, ucc}

Total Number of Preferred Answer Sets of (P,Phi):1
{-filstate, -perfected, ab1, n_ab2, posses, ship, sma}

C:\CompPLP>
```

図 6: compPLP による PLP$(P, \tilde{\Phi}_2)$ の計算結果

図 6 の結果より, $(P, \tilde{\Phi}_2)$ から, 担保権が無い ($\neg perfected$) という結果が得られたことが分かる.

6 まとめ

現在 Web ページ [6] 上で compPLP のプログラムを公開している．また, 文献 [3, 4] の著者にプログラムを提供し, PLP で表現した様々な非単調推論の例の実行結果に関するプログラムの検証や計算時間等の評価を行った.

参考文献

[1] M. Gelfond and V. Lifschitz: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, pp.264-374,1991.

[2] T. F. Gorden: The Pleadings Game: An Artificial Intelligence Model of Procedural Justice. Ph.D. thesis, TU Darmstadt, 1993.

[3] C. Sakama and K. Inoue: Prioritized logic programming and its application to commonsense reasoning, Artificial Intelligence 123, pp.185-222, 2000.

[4] T. Wakaki, K. Inoue, C. Sakama and K. Nitta: Computing Preferred Answer Sets in Answer Ste Programming, Proc. 10th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR'04), pp. 259-273, LNAI 2850, Springer-Verlag, 2003.

[5] dlv <http://www.dbai.tuwien.ac.at/proj/dlv>

[6] compPLP <http://www.ailab.se.shibaura-it.ac.jp/comppas.html>