

汎用エキスパートツールPROTONの開発

1 N-1

進藤 静一 梶野 喜久 澤本 誠*

(株)三菱電機中央研究所 (財)新世代コンピュータ技術開発機構

1 はじめに

PROTONはPSI上に実現されたエキスパートシステム構築支援ツールであり[1]、

①ルール表現とフレーム表現によるマルチパラダイム
②複数の知識源によるルール表現のモジラリティーの向上
を知識表現の特徴として持つ。現在、PROTONはその第一版の開発を終了し、諸々のエキスパートシステムをその上に実現することにより評価を行っている[2]。本稿ではPROTONの実装、特に、PSIのエンドユーザー言語であるESPとの関連について報告する。

2 PROTONの構成

図1にPROTONの構成を示す。ルールベース主体の推論を行う。記述対象世界に存在する事実は大域的な作業領域(WH)であるファクト管理部に格納され、メタKS、KSはWHの内容を参照/変更することによって推論を進めて行く。ルールセット(知識源、KS)はサブタスク毎に割り当てられ、前向き/後向き推論(KS毎に固定)を行う。前向き推論はOPS5に準拠、後向き推論はprologに準拠する。メタKSは前向き推論でKSを起動する。ルール左辺でKSを起動するべきWHの条件を、ルール右辺で起動するKSを指定する。

3 ルール処理部の実現

図2にルール処理部を構成するESP classを示す。図中、top-meta(top-fc)はメタルール(前向き推論KS内のルール)インタプリタであり前向き推論のrecognize-actサイクルを繰り返す。top-bcは後向き推論KS内のルールのインタプリタであり、ゴールのWHとのマッチング、ルールを用

いたサブゴールへの展開を行う。メタKS、前向き推論KS、後向き推論KSはユーザーが定義した各種ルールセットのESPコードへのトランスレーション結果である。以下に例としてtop-fc、前向き推論KSのESPコードを示す。

<top-fcの内容>

```
class top-fc has
:go(-,Ks):-
repeat,
clear-agenda(Agenda),
(:rule-LHS(Ks,Rule-name,Vars,Priority,
Severity,Tags),
① add-rule-to-agenda(Rule-name,Vars,Priority,
Severity,Tags,Agenda)
fail;true),
② (empty(Agenda),!);
③ select-rule(Agenda,Rule-name,Vars)
④ :rule-RHS(Ks,Rule-name,Vars),
⑤ !,fail);
end.
```

[注釈]

- ①発火可能なルールをすべて集める。:rule-LHS/6は前向き推論KSに対応するESP classへのメソッドコールである。
- ②発火可能なルールがない場合、推論を中止する。
- ③衝突解消策で実行するルールをひとつ選ぶ。
- ④⑤で選ばれたルールの右辺を実行する。
- ⑤③に戻る。

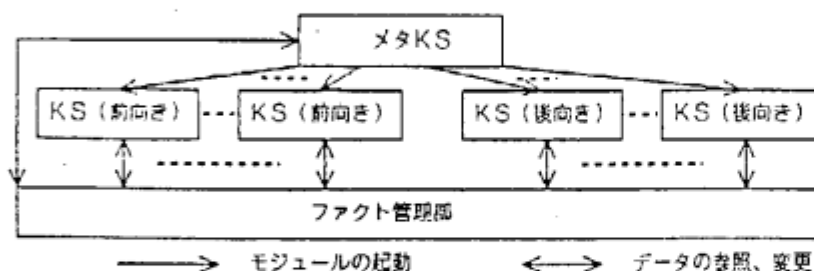


図1 PROTONの構成

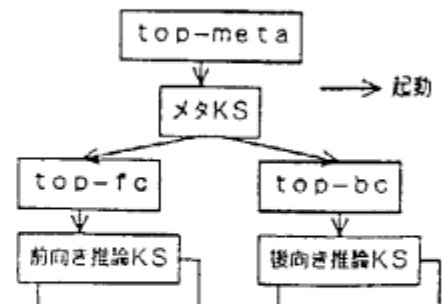


図2 ルール処理部を構成するESP class

Implementation of Expert Shell PROTON on PSI

S. Shindo Y. Hosono J. Sawamoto*

Mitsubishi Electric Corporation *ICOT

＜ルールのトランスレーション＞

ソース

! (他のルール)

check ::

te:queen:Q1!(x^X,y^Y,status^checking),

te:queen:Q2!(x^X1,y^Y1,status^unchecked),

==>

(V is (X-X1)*(Y-Y1)*(X-X1-(Y-Y1))*(X-X1*(Y-Y1)))

modify(te:queen:Q1!(status^checkvalue,value^V)),

modify(te:queen:Q2!(status^ok)).

! (他のルール)

ESP コード

class n-queen has

! (他のルールのコード)

:rule-LHS(-,check,{Q1,X,Y,Q2,X1,Y1,V},

100,Severity,Tags):-

:find-elm-first(\$wms,te,queen,Q1,

[[X,X],[Y,Y],[status,checking]],Tag1),

:find-elm-first(\$wms,te,queen,Q2,

[[X,X1],[Y,Y1],[status,unchecked]],Tag2),

Severity=2,

Tags-[[te,queen,Q1,Tag1],[te,queen,Q2,Tag2]]:

:rule-RHS(-,check,{Q1,X,Y,Q2,X1,Y1,V}):-

V is (X-X1)*(Y-Y1)*(X-X1-(Y-Y1))*(X-X1*(Y-Y1)),

:update-elm(\$wms,te,queen,Q1,

[[status,checkvalue],[value,V]]),

:update-elm(\$wms,te,queen,Q2,[[status,ok]]):

! (他のルールのコード)

end.

[注釈]

:find-elm-first/5,:update-elm/5 はファクト管理を行う ESP class へのメソッドであり、それぞれ、ファクト探索要求、ファクト変更要求の依頼である。

4 ファクト管理部の実装

ファクトは問題領域の構成要素(TE)、及び、TE間の関係(TR)からなる。WHにはTE、TRのinstanceが格納される。ファクトの持つ値はTEの属性値で表現される。フレームシステムと同様、TE間の関係、それを反映した属性の継承、付加手続(daeon)が書ける。例を以下に示す。

te substance

attrib

position-x restrict single real &

position-y restrict single real &

position-z restrict single real &

end.

属性値の制限記述

te block

super substance

attrib

width restrict single real (0<x)&

length restrict single real (0<x)&

height restrict single real (0<x)&

density restrict single real (0<x)&

weight restrict single real (0<x)

if-gotten null before

weight<-density*width*length*height

end.

付加手続

TRは引数の制限、置換式(下の例では、above、且つ、contactであればonの代用とできる、等)等を記述できる。

tr on

super above,contact

restrict 2

subst

on(upper,lower) -- under(lower,upper)

or

on(upper,lower) --

above(upper,lower)&contact(upper,lower)

end.

TE、TRともに、その各々が1 ESP class に変換される。上位概念("super"で定義される)はそのまま上位classに対応付けられる。TEでは、属性がinstance attribute、付加手続がinstance method にほぼそのまま変換される。

ルールからのWH探索/変更要求はclass wmsが窓口となる。wmsは推論で用いられているTE/TRをpoolで保持する。WHの要素であるTE/TRのinstanceは個々のclassで管理される(例、TE blockのinstanceはblockのclass attributeで一括して持つ)。TEに対する共通の処理(探索、属性値の変更、付加手続の起動、等)は、class teで実現される。TEの各classにこのclassを必ず継承させることにより各TEがこれらの処理を行えるようになっている。TRに關しても同様にclass trにTRに対する共通の処理を実現させている。更に、TE/TRに共通の処理(例、instanceのWHからの削除)をclass te,trのsuperclassであるtfで実現させている。このように、ファクト管理部ではESPの継承機能を用いて簡潔な記述が実現される。

5 おわりに

PROTONのルール処理部、ファクト処理部の実現について報告した。ESPの以下の機能が主に利用されている。

- ・ルール処理部: backtrack,unification
- ・ファクト処理部: class間の継承機能

[参考文献]

- [1] 澤本ほか「PSI上のエキスパートシステム開発支援ツール(1)~(4)」第33回情報大全,p1115~1122,1986.9
- [2] 水谷ほか「PROTONによるプログラム変換エキスパートシステム」本予稿集1N-2