CommonESPにおけるメタ述語の実現について

6E−3

田中吉廣　実近憲昭　黒澤芳夫　齊藤勝彦
(株) AI言語研究所　(財) 日本情報処理開発協会

1 はじめに

CommonEsp (以下CESPと呼ぶ) は第五世代コンピュータ開発機構で開発された, 論理型言語とオブジェクト指向を融合した言語ESPをもとに, 汎用ワークステーション上での普及と, より一層の機能強化を図って開発している言語である [1].

すでに,1989年度は最終版の基礎となる基本仕様版の開発を終了し, 現在フルセット版を開発中である.

本稿ではこのCESP基本仕様版において取り入れた, 実行時にダイナミックに述語の実行順序を制御するメタ述語の実現方法について述べる.[1]

2 メタ述語の概要

一般のprolog処理系はインタプリタ方式を採用しており, 実行時にダイナミックに述語の追加・削除が可能である. しかしcespはコンパイラ方式を採用しており, そのままでは述語のダイナミックな追加・削除はできない. このため対象となる述語にたいしてコンパイル時にダイナミック宣言をしておくことにより, インタプリタと同じように述語の追加・削除・検索などが利用できる機能を追加した.

さらにcespの場合はオブジェクト指向を取り入れており, メタ述語に関してもクラス毎に管理され, 述語の追加・削除・検索も当該クラスに対してだけ効果を持つ. 従って, モジュール数が膨大な大規模プロジェクトの開発にとってもメタ述語を利用したダイナミックな述語の制御を局所化して実行させることができる.

これはデバッグ作業等においても開発効率を高めることになる.

2.1 ダイナミック宣言

メタ述語の対象となるのはクラス内のローカル述語に対してである. 対象述語に対してコンパイル時に次の様なdynamic宣言をしておく.

```
class meta_test has
   :a(_,X,Y):-a(X,Y);
local
   :-dynamic a/2;
   a(X,Y):-b(X),c(Y);
   :-dynamic b/1;
   c(1);
end.
```

以上の例はクラス "meta_test" のローカル述語 a/2, b/1 に対してダイナミック宣言を行なっている.a/2 のようにダイナミック宣言と同じ述語がソースに記述されている場合は実行時に後で述べる :assert で追加したのと同じ扱いとなり, 検索や削除の対象と成り得る.

2.2 メタ述語クラス

上記ダイナミック宣言をしたローカル述語にたいして追加・削除・検索の機能はライブラリクラス "meta_predicate" が提供する [2].

追加 :assert, :asserta, :assertz

削除 :retract, :abolish, :erase

検索 :clause, :get_one_clause, :predicate

次のような例を見てみよう.

```
class mammal has
    :a(_,X):-a(X);
local
     a(monkey);a(elephant);
     a(dog);
end.

class animal has
nature mammal;
   :a(_,X):-a(X);
local
    :-dynamic a/1;
end.
```

クラス animal に対して:assert 述語で a(crocodile) の追加を行なった場合, :a(#animal, A) で質問すると, 最初自分のクラスである animal の中の crocodile を捜し出す. 次に別解を求めると継承クラスである mammal の方へ探しに行き, monkey,elephant ,dog を次々に捜し出す.

```
??- :asserta(#meta_predicate,
                animal, a(crocodile)).
```

[1]The implementation of meta-predicate on CommonESP

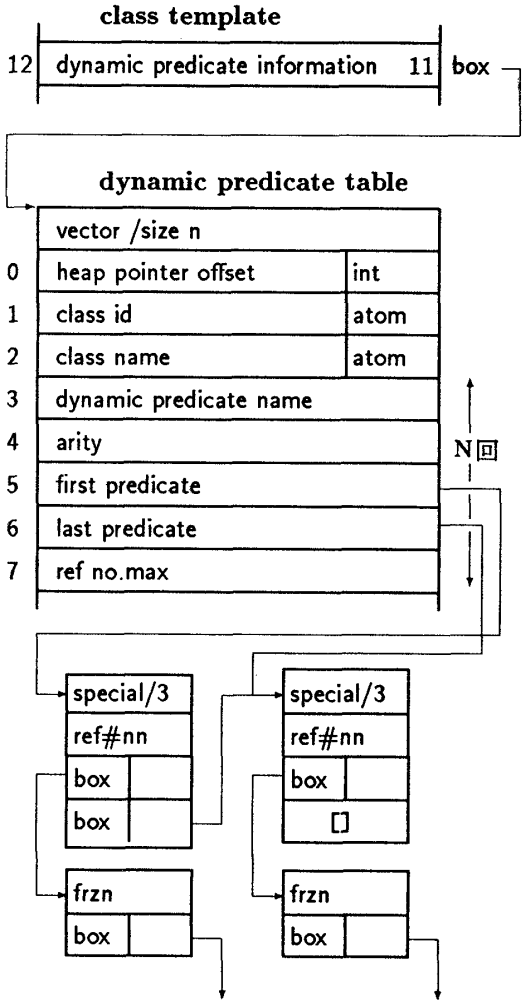
Yoshihiro Tanaka, Noriaki Sanetika
AI Language Research Institute,Ltd
Yoshio Kurosawa, Katsuhiko Saitoh
Japan Information Processing Development Center.

```
??- :a(#animal, A).

a = crocodile;
a = monkey;
a = elephant;
a = dog;

fail!!
```

3 メタ述語の実現方式



:assert 述語を実行する毎にソースレベルのデータを freeze された構造体として, ヒープ領域に登録する. 登録された述語に制御を渡すのはインタプリタ命令経由で行ない, 制御が渡ってきたら登録述語を逐次インタプリトして行く. コンパイルしたコード例は次の通り.

```
<ソース>
  :a(_,X,Y):-a(X,Y);
 local
  :-dynamic a/2;
<コンパイル後の内容>
  :a(_,X,Y):-:call(#interpreter,IWT_NO,a,{X,Y})
```

次にその管理方式をみてみよう. cesp はそのクラス毎の管理をクラス・テンプレート情報として保持している. ダイナミック述語情報もこのクラス・テンプレート情報としてクラス毎に管理され, 更にダイナミック述語テーブル (dynamic predicate table) を介してそれぞれの述語毎の情報へとポイントしている.

各述語の動きとしては例えば :assert が実行された場合ヒープエリアに登録できる形であるフローズン構造体にして,special/3 のセルに連結する. :asserta の場合は first predicate で指される special/3 の直下に連結し, :assert 及び :assertz は last predicate で指される special/3 の直下に連結する.

ref が出力項目として指定されている場合は ref 番号を取り出して,ref とユニファイする.ref は 3 要素のベクタで次のような情報を表す.

```
ref = {n1,n2,n3}
        |  |  └→代替節の追加番号
        |  └→同一クラスの dynamic 述語の連番
        └→当該 dynamictable へのヒープ
             ポインタ相対アドレス
```

逆に :retract の場合は special/3 セルの先頭からユニファイしてゆき, 該当する述語があったらチェーンから外し, 残りを繋いでおき更に次の述語を内部変数でポイントしておく. これは redo が起こった際に必要な処理をする為である.

:abolish は該当述語全てを, special/3 からはずす.:erase は ref で指定した述語を special/3 からはずし,:retract 同様に残りを繋いでおく.

検索系の :clause も同様に special/3 の先頭からユニファイして行きユニファイした述語のボディ部と body とユニファイし更に次の述語を内部変数でポイントしておく.

4 まとめと今後の課題

cesp はメタ述語がなくともオブジェクト指向の機能だけでも充分大規模なモジュール設計に耐えられる仕様を持っているが,prolog から移行してくるプログラマにとってはメタ述語は馴染みが深く, 容易に導入が図れるものと思われる.

しかし, 今回の実装に関しては実行速度を上げる為のインデキシング等による最適化は行なっていない. 今後, 実行速度の向上に関してさらに引続き検討を加えてゆきたい.

参考文献

[1] 田中ほか "基本仕様版 CommonESP システムの概要" 情報処理学会第 40 回全国大会 (1990)

[2] CESP ライブラリ 2.0 版 (株)AI 言語研究所 (1990)