メタ推論とその応用

——並列メタ推論用メタ述語simulateについて——

國藤　進，竹内　彰一，古川　康一　（（財）新世代コンピュータ技術開発機構）
上田　和紀　（日本電気（株）C&Cシステム研）

［概要］　並列実行型論理プログラム言語Concurrent Prolog 上で，対象知識とメタ知識間で通信を行いながら，与えられた両知識の下で与えられた問題を解くプロセスをシミュレートするメタ述語simulateをベースとするメタ推論方式の基礎を確立した．述語simulateベースのメタ推論方式に関する検討結果と，その各種応用可能性について検討する．

［1］　はじめに

本稿で考察しているメタ推論とは「メタ知識を用いた推論」のことである．ここにメタ知識とは「対象知識（実世界の対象に関する知識）あるいはメタ知識の使い方に関する知識」のことである．対象知識としては，Concurrent Prolog (CP)で表現可能なホーン論理の公理集合のみ取扱うことにする．CP上でメタ推論を実現するには，CP処理系の使い方に関する知識を表現し利用するメタ述語を，CPユーザに提供する必要がある．そこでCP上でゴールを解く過程を反映するメタ述語simulateを導入し，その表現利用技術にメタ推論を還元し，メタ推論機能をユーザに解放することにする．

メタ知識と対象知識を単一の論理プログラム言語で表現し，両知識を融合[1] した形式で利用することにより，対象知識の更新を含む高度なメタ推論機能を簡単に実現できる．逐次実行型論理プログラム言語DEC-10 Prolog の場合，このことはdemo述語という概念の有用性として文献[1] で提案され，著者らの論文[2] でその実現法，各種メタ推論方式の提案，およびそれら諸方式の応用例の開拓がなされた．本稿は，並列実行型言語CP上での，各種メタ推論方式の提案，メタ述語simulateの実現方式の例示，simulate述語の各種応用可能性について述べる．

［2］　メタ推論とその方式

simulate述語に基づくメタ推論を実行するため，著者らは知識表現の対象としている対象知識の世界，すなわち対象世界，とそれを操作対象とし管理しているメタ知識の世界，すなわちメタ世界，とをそれぞれ論理的に閉じた世界として構築する．ここに，論理的に閉じているとは，他の世界との情報交換が，特定の通信用チャネルに対するユニフィケーション操作によってしか行われないこと，および対象世界のfailがメタ世界のfailに波及しないことを意味する．これによって，ある世界の内と外との間の論理的つながりを，通信用論理変数を除いて，切断することができる．

著者らが提案する最も基本的なメタ推論用simulate述語は，次のような形式をしている．

(1) simulate(World,Goal)

simulateは，ある世界World において，与えられたゴールGoal を解くプロセスをシミュレートする．世界World をいかに実現するかは，本述語の設計方針，実現法に依存する．

(1) をベースにして，simulate述語にさらに次のような機能を追加することが考えられる．

(a) ゴールを解く過程で，World に対して節の追加，削除を（assert/retractによって）命令的に行うことが考えられる．この場合，引数World そのものを書替えるのではなく，World の変更結果を返すような別の引数を用意すべきである．

(b) 対象世界とメタ世界との通信，すなわち入出力を，命令的にではなくストリームによって行うならば，それらの入出力チャネルをGoalの引数に含ませておけばよい．しかし，入出力を命令的に行いたいときは，個々の入出力命令が操作対象とするチャネルを，simulate述語の引数として用意しなければならない．

(c) simulateの実行過程そのものから抽出されるメタ情報（たとえばproof tree）を利用可能ならしめるためには，simulate述語に，その取出しのための引数を追加する必要がある．これは，対象世界の success/fail 情報の取出しにも利用できる．

(d) simulateの実行プロセスに対して，制御情報（たとえばdepth-first とbreadth-first の切替え）を与えることが考えられる．そのためには，simulate述語に制御情報入力用の引数を追加すればよい．

以上の機能拡張をほどこしたsimulate述語の一般形は，たとえば次のような形式をしている．

(2) simulate(World,NewWorld,Goal,Result,Control,Channel)

このsimulate述語は，ある世界World において，他の世界と通信チャネルChannel を通じてメッセージ交換しながら，与えられたゴールGoalを制御条件Control 下で解くプロセスをシミュレートする．シミュレーションの結果，世界World は新世界NewWorldに更新されると同時に，ゴールを証明していくプロセスそのものから，必要なメタ情報Resultが順次抽出されていく．

なお，対象世界として，内部データベース自身を用いる場合は，

(1) の第1引数World は省略できる．これは "CP interpreter in CP"であり，文献[3] に示されている．

[3] メタ述語simulateの実現例

DEC-10 Prolog 上のCPによるsimulate述語(1) の実現例を以下に示す．これは，文献[3] のメタ述語 "reduce(Goal)" を参考にしている．CPの構文および意味は，文献[4] を参照されたい．

```
simulate(World,true).
simulate(World,(A,B)) :- true¦
    simulate(World,A?), simulate(World,B?).
simulate(World,(A & B)) :- true¦
    simulate(World,A?) & simulate(World,B?).
simulate(World,A) :-
    w_clauses(World,A,Clauses)¦
    resolve(World,A,Clauses,Body),
    simulate(World,Body?).
simulate(World,A) :- cpsystem(A)¦cp(A).
simulate(World,A) :- prolog(A)¦true.

resolve(World,A,[(A<--(Guard¦Body))¦Cs],Body):-
    simulate(World,Guard)¦true.
resolve(World,A,[C¦Clauses],Body):-
    resolve(World,A,Clauses,Body)¦true.

w_clauses(World,A,Clauses) :-
    wait(A,A1)&cpsystem(A1)¦fail.
w_clauses(World,A,Clauses) :-
    wait(A,A1)¦
    prolog((P=..[World,(X<--Y)],
    bagof((X<--Y),(P,unif(A1,X)),Clauses))).
```

このプログラムを用いる場合は，世界を構成する節は，共通の世界名を付して，

世界名((A <-- G1# ... #Gn | B1# ... #Bm)).

（ただし，#は逐次and "&" あるいは並列and ","）

という形で内部データベースにあらかじめ登録しておくものとする．そして，第1引数には，その世界名を与えるものとする．

ここでは世界を名前によって識別させる方法をとったが，第1引数に，世界を構成する節のリストを直接与えるような仕様にすることも可能である．さらに，節集合をコンパイルしたものを第1引数として受付けるようなsimulate述語を，システムで用意することも考えられる．

上述のsimulate述語の実行例を以下に示す．

```
w((primes <-- true ¦
     integers(2,I),sift(I?,J),outstream(J?))).
w((integers(N,[N¦I]) <-- N1 := N+1 ¦
     integers(N1,I))).
w((sift([P¦I],[P¦R1]) <-- true ¦
     filter(I?,P,R),sift(R?,R1))).
w((filter([N¦I],P,R) <-- 0 =:= N mod P ¦
     filter(I?,P,R))).
w((filter([N¦I],P,[N¦R]) <-- 0 =\= N mod P ¦
     filter(I?,P,R))).
w((outstream([X¦S]) <-- writesp(X) ¦
     outstream(S?))).
```

```
¦ ?- cp simulate(w,primes).

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
```

[4] おわりに

本稿で提案したsimulate述語の応用例として，以下のものが考えられる．

(a) 知識プログラミング言語Mandala[5]における多重相続の実現
(b) 知識同化を含む知識ベース管理への応用
(c) OSの核 "shell"の実現
(d) 並行プロセスの知的なエディタ，デバッガへの適用
(e) 並列に動作するマシン間のインタフェース合せ
(f) 協調問題解決への応用

最後に，メタ推論に関する今後の課題は，次のような諸研究の成果を踏まえて、並列メタ推論のための言語仕様を洗練し，その実現法を検討することである．

(a) 高速な多世界データベース・アクセス機能の実現に関する研究
(b) メタ情報 ((2) のResult) 抽出法に関する研究
(c) 制御条件 ((2) のControl) 付与の方法に関する研究
(d) 通信チャネル構成法に関する研究

[謝辞] 本研究を進めるに当って熱心にご討論していただいたICOI核言語設計タスクグループの諸氏，特に横森貴博士（富士通・国際研）に感謝します．

[References]

[1] Bowen, K.A., Kowalski, H.A.: Amalgamating Language and Meta Language in Logic Programming, School of Computer and Information Sciences, Univ. of Syracuse(1981).
[2] Kunifuji, S., Asou, M., Takeuchi, A., Sakai, K., Miyachi, T., Kitakami, H., Yokota, H., Yasukawa, H., Furukawa,K.: Amalgamation of Object Knowledge and Meta Knowledge in Prolog and its Application, Information Processing Society of Japan, Preprints of WG on Knowledge Engineering and Artificial Intelligence, June 1983 (in Japanese).
[3] Shapiro, E.: Systems Programming in Concurrent Prolog, 11th Annual ACM SIGACT/SIGPLAN Symposium on Principles of Programming Languages, Salt Lake City, Utah, Jan. 1984.
[4] Shapiro, E.: A Subset of Concurrent Prolog and its Interpreter, ICOT TR-003,1983.
[5] Furukawa, K., Takeuchi, A., Kunifuji, S.: Mandala: Knowledge Representation System in Concurrent Prolog, Information Processing Society of Japan, Preprints of WG on Knowledge Engineering and Artificial Intelligence , Nov. 1983.