知識獲得とメタ推論

岡藤　進，宮地泰造，北上　始，古川康一
(財)新世代コンピュータ技術開発機構

〔1〕 概要

第5世代コンピュータ・システムの目標は知識情報処理システムのプロトタイプの確立にある．知識情報処理システム向きの知識ベース管理の基本機能としては知識ベース・アクセス機能，知識獲得機能，や知識表現機能などがある．本稿ではこれら諸機能の実現に有用なメタ推論機能について，論理型言語DEC-10 Prolog上での実現方式を述べる．具体的にはDEC-10 Prolog上で対象知識とメタ知識の融合(1)を行うメタ推論方式を確立した後，その結果を暗黙値を含む知識獲得システムへ適用した．その設計指針，インプリメント上の問題点と解決法を述べ，我々の提案するメタ推論方式の有用性を報告する．

〔2〕 メタ推論とその方式

本稿で考察しているメタ推論では，証明可能性概念を直接反映するdemo述語というメタ述語を導入し，それを用いて対象知識とメタ知識とを同一言語上で操作し，「知識の使い方に関する知識を含む推論」を行う技法のことである．demo述語そのものの概念的有用性は既に[4]で示されていたが，彼らはその論理型言語上での実現法を示さなかった．それに対して我々はDEC-10 Prolog向きのdemo述語実現法を提案し，その各種応用例[1,3]を開拓中である．

我々がインプリメント中のdemo述語は，次のような形式をしている．

```
(1) demo(World,Goal)
(2) demo(World,Goal,Result)
(3) demo(World,Goal,Result,Control)
(4) demo(World,Goal,Channel)
(5) demo(World,Goal,Result,Channel)
(6) demo(World,Goal,Result,Control,Channel)
```

最も一般的な形式(6)の意味は，「ある世界Worldにおいて，他の世界と通信用変数Channelを通じてのメッセージ交換をしながら，与えられた制御情報Controlに従って，与えられたゴールGoalを解くプロセスを起動し，その証明のProof Treeそのものから必要な情報Resultを抽出するプロセスをシミュレートすること」である．

タイプ(1)の実例は既に[1]で示したので省略する．タイプ(3)については制御子Controlの形式化という今後の課題を残しているが，幾つかの実際的応用例は確立した．タイプ(4)～(6)はそれぞれ(1)～(3)のConcurrent Prolog版であるが，これについては別の機会に報告する．従って本報告では，タイプ(2)の実現法とその応用に関する考察に話題を集中する．

〔3〕 証明過程からの情報抽出

タイプ(2)の特長は，証明のProof Treeの変形過程そのものから必要な情報を抽出し，その結果を第三引数Resultにフィードバックすることである．Resultは情報の作成の仕方により，次のような情報抽出が可能である．

(i) Resultは"証明の結果"：Prologでは証明の失敗を全てfailで同一視するが，failの原因を分析し，failureかerrorか等を識別し，その結果をリターンコードとして返す場合．

(ii) Resultは"証明の過程"：証明のtrue/failパターン，あるいはProof Tree全体を変数Resultの中に展開し

て残していく場合.

(ロ) Result="抽出メタ論理的概念":
同一構造の他のWorld,別の構造のWorld,あるいは真に物理的な外部のWorldへの情報伝達を通してResultを抽出する場合.

上述のResult情報は，それぞれ次のような分野に適用した場合に有用である.

(イ) 暗黙値を含む知識獲得，デバッガ.

(ロ) 説明・診断・トレース機能.

(ハ) 暗黙推論，協調問題解決，物理I/OやOSとの通信.

これら三種のResultを実現するのに次のようなインプリメント技法を確立した.

(A) 副作用(assert, retract)利用方式

(B) メタfail(failしないfail)方式

方式(A),(B)のそれぞれをインプリメントし，(1)で示した知識獲得プログラムに組込んだ. それらの長短は次章で述べる.

〔4〕 知識獲得への適用

本稿で述べたメタ推論方式は，人間が行う暗黙推論を逐次形推論マシーン上で実行させるのに適している. その最も簡単な適用例として暗黙値を含む知識獲得プログラムを作成したので，その概略を説明する

〔4.1〕 副作用利用方式

否定"not"の実行時に特定の述語への副作用を残し，その値を管理することにより，与えられた知識ベースに対するブール値としてのResultを返すdemo述語を構成することができる.

〔知識ベース〕

```
can_fly(X):-bird(X),
            not(cannot_fly(X)).
bird(condor).
bird(ostrich).
cannot_fly(ostrich).
```

〔入出力例〕

```
| ?- demo(can_fly(ostrich),R).
R = false
yes
| ?- demo(can_fly(condor),R).
R = true
yes
| ?- demo(can_fly(bat),R).
no
```

上述の結果を知識獲得プログラム(1)に組込むのはやさしい. 本方式の欠点は，副作用をかける同期の問題である.

〔4.2〕 メタfail方式

メタ言語上のfailはfailしないものと仮定し，暗黙値コール結果をResultとしてトップレベルに引上げる方式である. demo述語の第4引数Resultに空リストが戻れば正常終了，それ以外のリストが戻れば暗黙値推論の起動の結果を示す. 本方式の方が，下に見られるように知識獲得用メタ推論方式として使い易く，また情報抽出の自由度という点でも応用可能性が広い.

〔demo述語〕

```
demo(KB,true,Cut,[]):-!.
demo(KB,fail(X),Cut,[default_value(X)]):-!.
demo(KB,!,Cut,[]).
demo(KB,!,cut,[]).
demo(KB,(P;Q),Cut,L):-
    !,(demo(KB,P,Cut,L);demo(KB,Q,Cut,L)).
demo(KB,(P,Q),Cut,L3):-
    !,demo(KB,P,Value,L1),
    (Value==cut,Cut=cut,!;
    demo(KB,Q,Cut,L2),append(L1,L2,L3)).
demo(KB,P,Cut,L):-
    clause(KB,P,Q),demo(KB,Q,Cut,L),
    (Cut==cut,!,fail;true).
demo(KB,P,Cut,L):-system(P),P.
```

〔多世界知識ベース〕

```
pkb((can_fly(X):-bird(X),
            (cannot_fly(X),!,false(X);
             true))).
pkb(bird(condor)).
pkb(bird(ostrich)).
pkb(cannot_fly(ostrich)).
```

〔知識獲得入出力例〕

```
| ?- passimilate([pkb,mkb],can_fly(ostrich)).
 Pinput is exception !
yes
| ?- passimilate([pkb,mkb],can_fly(condor)).
 Pinput is deducible from PKB !
yes
| ?- passimilate([pkb,mkb],can_fly(bat)).
 Pinput is acquired in PKB !
yes
```

〔謝辞〕 貴重なご意見をいただいた竹内彰一，横田治夫(ICOT)両氏に感謝する.

〔文献〕 (1) 国藤 他: [illegible] 人工知能研究会資料, 1983. (2) Bowen et al: Amalgamating Language and Meta Language in Logic Programming, Academic Press, 1982. (3) 北上 他: (1)と同一出典, 資料等-2.