D-2-5

知識表現形式ＤＣＫＲの高速化と機能拡張

High-Speed and Extended DCKR

山田　基司，　奥村　学，　田中　穂積
（東京工業大学　工学部）

当研究室で開発された知識表現形式ＤＣＫＲ(Definite Clause Knowledge Representaion)[田中 86]は，ユニフィケーションを利用した知識の継承，手続き付加などが容易に行えるが，知識の量が増えるにつれて速度が低下するという問題がある．この問題を解決するため，我々はＤＣＫＲの持つ知識検索に関する一般性と整合性を保ち，しかも高速に動作する新しい知識の内部表現形式を開発したのでその概要を実験結果と共に説明する．また，その機能の拡張性についても説明する．

１．はじめに

ＤＣＫＲでは、オブジェクトを構成する各スロットを、semという単一の述語をヘッドとするホーン節の集合と見なす．たとえば，clyde#1は「象」で，「色は白」，象は「サイズが大きい」という知識のＤＣＫＲによる記述は，(1a),(1b),(1c)のようになる．

```
sem(clyde#1, color:white).                    (1a)
sem(clyde#1, P) :-                            (1b)
        isa(elephant, P).
sem(elephant, size:large).                    (1c)
```

ここで，isaの定義は以下のようである．

```
isa(Upper, P) :-
        P=isa:upper;
        sem(Upper, P).
```

述語semの第一引数はオブジェクト名で，第二引数は，スロット名とスロット値の対である．このようなＤＣＫＲで記述した知識に対する推論は，Ｐｒｏｌｏｇに組み込みの機能をそのまま用いることができるので，推論を行うための特別なプログラム（インタープリタ）を作る必要はない．また，知識継承についても，Ｐｒｏｌｏｇのユニフィケーション機構をそのまま利用できる．たとえば，Ｐｒｏｌｏｇのトップレベルに於いて

```
?- sem(clyde#1, X).
```

を実行すれば以下に示す値が得られる．ここでclyde#1の上位のelephantの持つ性質が得られていることに注意してほしい．

```
X = color:white;
X = isa:elephant;
X = size:large;
```

一方このように全ての知識が単一のsemという述語の中に格納されているため，たとえばclyde#1の性質を取り出したいときに述語名のレベルでは，全ての知識とパターンマッチする．また第一引数を変数にして，その値を取り出したいとき，すなわち

```
?- sem(X, size:Y).                            (1d)
```

を実行すると述語名だけでなく第一引数までパターンマッチする．さらに(1b)のような階層をたどる知識については，(1d)は(1b)などのヘッドと必ずユニファイされるので，全ての知識継承階層を遡ってあらゆる知識とのユニフィケーションが繰り返されることになる．したがって大量の知識を扱う場合には検索速度がきわめて遅くなる[赤間 86]．第２章で示すが，我々はＤＣＫＲと知識検索に関する一般性と整合性を保ち，しかも高速で動作する新しい知識の内部表現を開発した．

２．知識の新しい内部表現形式による高速化と高水準知識表現言語ＳＲＬ／０

我々が開発した内部表現方式では，(1a)の知識は(2a1)~(2a3)，(1b)は(2b1)~(2b3)，(1c)は(2c1)~(2c3)のように表す．ここで，(2b1),(2b2)は知識継承階層を表す内部表現である．

```
clyde(1, color,   white, p).                  (2a1)
color(0, clyde#1, white, s).                  (2a2)
white(0, clyde#1, color, v).                  (2a3)

elephant(0, =>, clyde#1).                     (2b1)
clyde#1(1, <=, elephant).                     (2b2)
isa(clyde#1, elephant, s).                    (2b3)
```

```
elephant(0, size, large, p).                    (2c1)
size(0, elephant, large, s).                    (2c2)
large(0, elephant, size, v).                    (2c3)
```

第１引数の数字は個体を識別するためのナンバーで，述語名がクラスのときには数字の0が入る．また，最後の引数のp,s,vはそれぞれオブジェクト名，スロット名，スロット値を表す．検索はＰｒｏｌｏｇによって作成されたインタープリタで行う．このように，一つの知識に対して，オブジェクト名，スロット名，スロット値のそれぞれが述語になっていれば，そのいずれを検索する場合でも専用のルーチンを用いて検索対象をすばやく取り出すことができる．それにより，検索の高速化が期待できる．特に，(1d)のようにオブジェクト名が変数の（スロット名が分かっている）場合には，（(2c2)のような）スロット名が述語になっている内部表現から対応するオブジェクト名を取り出して，（(2b1)のような親から子への）知識継承階層の知識を用いて継承を下りながら，そのスロット値を継承するオブジェクト名を求めていくことができるので，ＤＣＫＲの場合と比べて効率的で高速な検索が可能になる．検索のためのトップレベルの述語はconとよばれ，オブジェクト名，スロット名，スロット値のそれぞれを検索するために，３つの述語con1,con2,con3を呼び出す．

```
con(X, S:V) :-
        is_object(X) -> con1(X, S:V);           (3a)
        nonvar(S)    -> con2(X, S:V);           (3b)
        nonvar(V)    -> con3(X, S:V).           (3c)
```

con1はX，すなわちオブジェクト名が変数でないときに呼び出され，

[1]オブジェクト名をヘッドの述語とする内部表現を探して，マッチすればそのスロット名とスロット値の対を返して[2]に行く．

[2]知識継承階層を表す内部表現があれば，継承を遡って親のオブジェクト名を取り出して[1]に行く，なければfailでリターン．

con2はXが変数で，Sが変数でないとき

[3]スロット名をヘッドの述語とする内部表現を探して，マッチすればオブジェクト名と，スロット値を返して[4]に行く．

[4]オブジェクト名に対し知識継承階層を表す内部表現があれば，継承を下って（スロット値を引き継ぐ）子のオブジェクト名を得て[4]に行く，なければfailでリターン．

con3はX,Sが変数で，Vが変数でないとき

[5]スロット値をヘッドに持つ内部表現から，検索するスロット値を持つスロット名を求めて，それぞれのスロット名，スロット値の対に対して[3]を実行する．

たとえばトップレベルで

```
?- con(clyde#1, X).
```

と入力すると，con1は(2a1)とマッチして

```
X = color:white;
```

を出力し，次に知識継承階層を表す知識である(2b2)によりclyde#1の知識継承階層を遡ってelephantがclyde#1の上位にあることを見つけ，

```
X = isa:elephant;
```

を出力する．以下同様に，(2c1)より

```
X = size:large;
```

という結果を得る．

また，(2a1)~(2a3),(2b1)~(2b3)のような内部表現は，直接書くには煩雑であるから記述言語として，本研究室で開発された高水準知識表現言語ＳＲＬ／０を用いる[奥村 86]．ＳＲＬ／０では，(2a1)~(2a3),(2b1)~(2b3)の知識を(4)のように書く．

```
clyde#1 ::
          [color:white]                         (4)
          [isa:elephant].
```

一般的な知識，すなわち，"Everyone who lives at Maple Street is a programmer." といった知識は，ＤＣＫＲでは(5)のように記述するが，ＳＲＬ／０では(6)のように記述する[Nilsson 80]．

```
sem(X#J, profession:programmer) :-
        sem(X#J, isa:human),                    (5)
        sem(X#J, loc:mapleStreet).
```

```
X#J ::
        [profession:programmer                    (6)
                if X#J isa:human,
                   X#J loc:mapleStreet].
```

システムの概要を図１に示す．

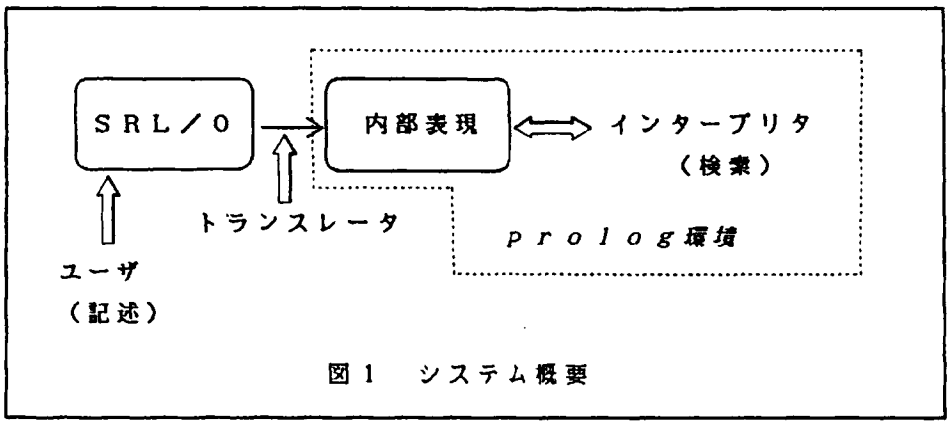

図１　システム概要

３．知識の検索速度の比較

ＤＣＫＲと今回作成したインタープリタの速度比較を図２から図４に示す．
縦軸は検索に要したｃｐｕｔｉｍｅ，横軸は知識の個数である．点線がＤＣＫＲで，実線が本内部表現形式を用いた場合の実測値である．図２～図４から明かなように，ＤＣＫＲでは，知識の量が増えると検索時間が指数関数的に増えるのに対し，本方式では比例関係にある．したがって，知識の規模が大きいとき，もしくは(5),(6)のように変数を含む知識検索を行う場合にはきわめて有効であることが分かる．

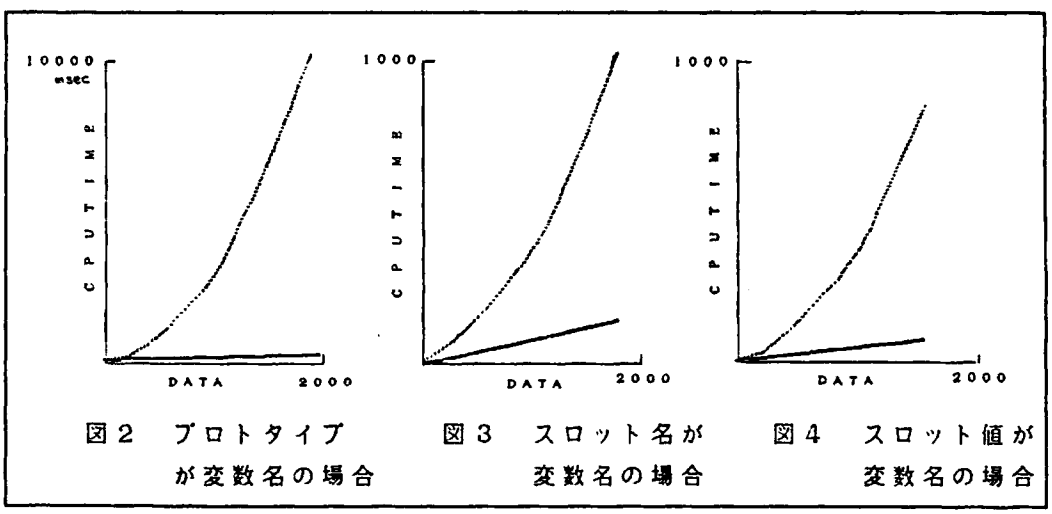

図２　プロトタイプが変数名の場合　　図３　スロット名が変数名の場合　　図４　スロット値が変数名の場合

４．機能の拡張

前述のように，今回開発した（Ｐｒｏｌｏｇによる）知識の表現方式では，知識の検索を行うときに，con1，con2,con3というインタープリタが常に介在するので，知識継承の際のオーバーライド等の機能の拡張が容易にできる．機能拡張の例として，ワールドの記述法について述べる[中島 86]．

ＳＲＬ／０を拡張したワールドの記述を(7)に示す．

```
in ワールド名 {
   when {
      <precondition>
   }                                        (7)
   ＳＲＬ／０による知識の記述　または
   （サブ）ワールドの定義
}
```

全てのワールドは，ルートワールドの下に作られる．宣言を記述するときに，ワールドを指定しなければ，ルートにあるものと見なされる．ワールド間の上下関係は，定義の際に明示的に与えなければならない．

検索の際には，トップレベルで

?- in(ワールド名).

としてあるワールドにはいると，そのワールドと，そのワールドの親のワールドの内容だけが見えるようになる．検索はサブセット優先で行われる．また，ある特定の性質を満たすワールドを選ぶために，記述言語中で"when"文の中に記述される<precondition>が存在する．すなわち，この場合の<preconditon>とは，そのワールドが満たさなければならない前提条件ではなく，ある条件を満たすワールドを検索して，そのワールドにはいるためのキーとしての役割を果たす．すなわち，トップレベルで，

?- when(<precondition>).

とすると，条件を満たすワールドにはいる．

あるワールドにはいるとき，ワールド名は特定の（自分で名前をつけた）スタックにプッシュ，ポップすることが可能で，スタックの系列も選ぶことができる．

５．おわりに

ＤＣＫＲの持つ知識検索に対する一般性と整合性を保ち，かつ高速な知識表現方式を提案した．知識の量が大きくなるほど速度の差は飛躍的に広がる．機能の拡張については，ｍｕｌｔｉｐｌｅ－ｉｎｈｅｒｉｔａｎｃｅ等の実現について考察中である．また，大規模なシステムへの実際の適用を検討している．

６．参考文献

[田中 86]:田中穂積，小山晴生，奥村学，" 知識表現形式ＤＣＫＲとその応用"，
コンピュータソフトウェア Vol.3 No.4 Oct.1986

[奥村 86]:奥村学（他），"意味記述用言語ＳＲＬ／０の設計とＤＣＫＲ"，
情報処理学会情報学基礎研究会資料,1986,pp.1-2

[Nilsson 80]:Nilsson.N.J.,"Principles of Artificial Intelligence",
Palo Alto. Calif, Tioga 1980

[赤間 86]:赤間清，"継承階層ＰＲＯＬＯＧと多重継承"，
日本ソフトウェア科学会第３回大会論文集,1986

[中島 86]:中島秀之，"多重世界機構の論理的意味"，
日本ソフトウェア科学会第３回大会論文集,1986

＜付録＞ワールドの記述例，実行例

＊＊＊記述例 １＊＊＊

```
john#1 :: [ born_in:america]
      [ birthYear:1955].
mary#1 :: [ mother_of:john#1].
peter#1 :: [ father_of:john#1].

in fifties [
        when [
                time :: [1950~1959].
        ]
        mary#1 :: [ nation:japan].
        peter#1 :: [ nation:england].
]

in sixties [
        when [
                time :: [1960~1969].
        ]
        mary#1 :: [ nation:japan].
        peter#1 :: [ nation:japan].
]

in american_style_law [
        X :: [ nation:N
                if X born_in:N].
]

in japanese_style_law [
        X :: [ nation:N
                if X birthYear:Y.
                   Z father_of:X .
                   [ whentime, Y). ]
                   Z nation:N ].
]

in english_style_law [
        X :: [ nation:N
                if X born_in:N.
                   X birthYear:Y.
                   [ Z mother_of:X ;
                     Z father_of:X ].
                   [ whentime, Y). ]
                   Z nation:N ].
]
```

＊＊＊実行例 １＊＊＊

```
[ ?- in(gYamerican_style_law),
     con(john#1, nation:N).

N = america ;

no
[ ?- in(gYjapanese_style_law),
     con(john#1, nation:N).

N = england ;

no
[ ?- in(gYenglish_style_law),
     con(john#1, nation:N).

no
[ ?-
```

＊＊＊記述例 ２＊＊＊

```
champ ::        [points:10].
second ::       [points:6].
thrid ::        [points:3].

hoshino#1 ::    [team:leytonHouse].
matsumoto#1 ::  [team:cabin].
takahashi#1 ::  [team:advan].
takahashi#2 ::  [team:sagawakyubin].
nakajima#1 ::   [team:epson].
hagiwara#1 ::   [team:leytonHouse].

in race#1 [
        when [
                time:: [7~7].
                place:: [cuicuit:suzuka].
                place:: [prefecture:aichi].
        ]
        hoshino#1::     [isa:champ].
        matsumoto#1::   [isa:second].
        takahashi#2::   [isa:third].
]

in race#2 [
        when [
                time:: [8~8].
                place:: [cuicuit:fuji].
                place:: [prefecture:shizuoka].
        ]
        nakajima#1::    [isa:champ].
        hagiwara#1::    [isa:second].
        matsumoto#1::   [isa:third].
]

in race#3 [
        when [
                time:: [9~9].
                place:: [cuicuit:sugo].
                place:: [prefecture:sendai].
        ]
        takahashi#1::   [isa:champ].
        hoshino#1::     [isa:second].
        nakajima#1::    [isa:third].
]
```

＊＊＊実行例 ２＊＊＊

```
[ ?- when(place, cuicuit:fuji),
     con(X, isa:champ).

W = gYrace#2,
X = nakajima#1 ;

no
[ ?- when(time, 9, W),
     in(W),
     con(hoshino#1, P).

W = gYrace#3,
P = team:leytonHouse ;

W = gYrace#3,
P = isa:second ;

W = gYrace#3,
P = points:6 ;

no
[ ?- in(gYrace#N),
     con(X, isa:champ).

N = _236,
X = hoshino#1 ;

N = _236,
X = nakajima#1 ;

N = _236,
X = takahashi#1 ;

no
[ ?-
```