情 報 学 基 礎 1－2
（1986・7・24）

意味記述用言語ＳＲＬ／０の設計とＤＣＫＲ

奥村学，小池康晴，田中穂積
（東京工業大学　工学部）

本研究室で開発されたＤＣＫＲは、ホーン節形式をベースにした知識表現形式である。このＤＣＫＲを自然言語の意味処理に応用すると、辞書項目の柔軟な意味記述が可能になるとともに、それを利用した意味処理用プログラムの中核がPrologに組み込みの機能で代用可能になる。しかし、ＤＣＫＲを応用したこの辞書の記述形式を用いて意味解析を行なうと、意味的に異常な文をも正常な文として受理してしまう。本稿では、この問題点を説明し、それに対する解決策について述べる。また、ＤＣＫＲを辞書記述の機械語レベルの形式として考え、ユーザが辞書項目を記述する辞書記述用の高水準言語ＳＲＬ／０を設計したので、それについても述べる。

WG FI 1 - 2

"The Design of Semantic Representation Language SRL/0 and DCKR"
(in Japanese)

by Manabu OKUMURA, Yasuharu KOIKE and Hozumi TANAKA
(Department of Computer Science, Tokyo Institute of Technology,
2-12-1 Ookayama Meguro-ku, Tokyo, 152, Japan)

DCKR, which has developed in our group, is a knowledge representation form based on Horn clauses. DCKR not only gives us the flexible description of semantics of each dictionary entry but also alleviates our programming efforts, since the essential part of semantic processing can be executed directly by Prolog interpreter. However, the naive DCKR representation of dictionary entries often accepts even semantically abnormal sentences. The reasons will be explained and one of the solutions will be given. We also design high level knowledge representation language called SRL/0, regarding DCKR as the machine-language level of the dictionary representation.

1　はじめに

本研究室で開発されたＤＣＫＲ(Definite Clause Knowledge Representation)[小山85 , 田中 86] は、ホーン節形式をベースにした知識表現形式であり、Structured Object（以後これを単に Objectとよぶ。）を構成する各スロットを、述語 sem をヘッドとする１つのホーン節で表現する。従って、１つの object は、第１引数が等しい sem述語をヘッドとするホーン節（スロット）の集合とみなすことができる。このことから、ＤＣＫＲで記述した知識に関する推論を行なうプログラムのほぼ全てを、Prologに組み込みの機能で代用できる。

自然言語の意味処理に、このＤＣＫＲを応用すると、2.1で述べるように、辞書項目の柔軟な意味記述が可能になるとともに、それを利用した意味処理用プログラムの中核が、Prologに組み込みの機能で代用可能になる。

しかし、ＤＣＫＲを応用したこの辞書の記述形式（以後これをＤＣＤ(Definite Clause Dictionary)[田中 85]とよぶ。）を用いて意味解析を行なうと、意味的に異常な文をも正常な文として受理してしまう。2.2ではこの問題点を説明し、それに対する解決策については2.3で述べる[奥村 86]。

また、ＤＣＫＲを辞書記述の機械語レベルの形式として考え、ユーザが辞書項目を記述する辞書記述用の高水準言語を今回設計した。３では、この辞書記述用言語の第０版ＳＲＬ／０について述べる。

最後に、４では、まとめと今後の課題について述べる。

2　ＤＣＫＲを応用した意味解析

2.1　ホーン節による辞書記述形式ＤＣＤ

フレームは、意味解析を行なうための辞書記述の形式としてよく用いられてきた表現である。フレームは、スロットの集合から構成されている。一般に、スロットは、スロット名，制約条件，アクションの３つ組から構成される。スロット名は、フィラーが文中で満たすべき統語制約条件を表わす。制約条件には、そのスロットを満たすことができるフレーム（これをフィラーとよぶ。）の意味的性質を記述する。これを以下、意味制約条件とよぶ。

辞書記述としてフレームを用いる意味解析の手順は、およそ次のようである。

(i)　選択したスロットの統語ならびに意味制約条件をフィラーが満たせば、アクションを起動して終了、さもなければ(ii)へ。

(ii)　次のスロットがあれば、それを選択して(i)へ、さもなければ(iii)へ。

(iii)　上位の概念があれば、そのフレームを取り出し(i)へ、さもなければ意味処理失敗として終了。

(i)から(iii)の意味解析の手順を観察すると、

[a] (i)の意味制約条件は論理式で表現することが多く、これはＤＣＫＲで容易に表わすことができる

[b] (ii)のスロットの選択には、Prologに組み込みのバックトラック機構が利用できる。
これは、ＤＣＫＲでは、スロットを１つのホーン節として表現しているからである

[c] (iii)の知識の継承は、ＤＣＫＲのもつ知識継承機構で容易に実現できる

ことがわかる。

以上のことから、意味解析の中核となるプログラムを、Prologの基本計算機構で代用可能なことが読み取れる。以下では、それを実例により示す。図2.1にＤＣＤによる辞書項目 open の記述例を示す。

```
:- op( 100 , xfy , ~ ) ,
   op( 100 , xfy , : ) .
sem( open , subj : N ~ In ~ Out ) :- nonvar( N ) ,
   ( sem( N , isa:human ) ,
       addProp( agent : N , In , Out ) ;          ..... (2-1)
     ( sem( N , isa:event ) ;
       sem( N , isa:thingOpen ) ) ,
       addProp( object : N , In , Out ) ;         ..... (2-2)
     sem( N , isa:instrument ) ,
       addProp( instrument : N , In , Out ) ;     ..... (2-3)
     sem( N , isa:wind ) ,
       addProp( reason : N , In , Out ) ) .       ..... (2-4)
sem( open , obj : N ~ In ~ Out ) :- nonvar( N ) ,
   ( sem( N , isa:event ) ;
     sem( N , isa:thingOpen ) ) ,
       addProp( object : N , In , Out ) .         ..... (2-5)
sem( open , with : N ~ In ~ Out ) :- nonvar( N ) ,
   sem( N , isa:instrument ) ,
     addProp( instrument : N , In , Out ) .       ..... (2-6)
sem( open , Prop ) :-
   isa( act , Prop ) .                            ..... (2-7)
```

図2.1 ＤＣＤによる open の辞書記述例

図2.1では、

(i) １つのスロットを、sem述語をヘッドとする１つのホーン節で表現している。これを以下ではDCD節とよぶ。例えば、(2-5)は、objスロットを表現している。(2-5)では、objスロットを満たしうるフィラーは、”事象”(event)もしくは”開くもの”(thingOpen)でなければならず、また、フィラーがこの意味制約条件を満たすなら、アクションとしてaddPropという述語を呼び出し、フィラーの深層格が対象格(object)であるという意味を抽出する、ということが記述されている。

(ii) スロットの並びは、DCD節の並びとして表現している。それにより、スロットの選択が、Prologのバックトラック機構により自動的になされる。

(iii) スロット中の意味制約条件が、DCD節のボディに直接記述されている。従って、意味制約条件の検査は、Prologプログラムを直接実行することに置き換えられる。

(iv) フィラーが、ある概念を含意するかどうかを調べるための、上位概念をたどる述語もまた述語 sem によって記述されている。述語 sem は、辞書項目をDCD節で定義するための述語そのものであるから、上位概念をたどるための特別な述語を用意する必要がない。

(v) 知識の継承は、DCD節(2-7)で行う。それにより、それより前のスロット(subj , obj , with)をフィラーが満たしえなか

った時には、Prologのバックトラックおよびユニフィケーション機構をそのまま利用して、上位のactにあるスロットを調べることができる。

(vi) Prologの基本計算機構をそのまま利用して、DCD節で記述した辞書項目を検索し、応答することができる。

(v)と(vi)は、概念のリンクを下位から上位に向けてたどり、下位から、上位の性質をアクセスするものであった。それに対して、

```
?- sem( X , bird ) .
```

を実行することにより、bird からはじまり、bird の下位にある概念を次々に得ることもできる。従って、

(vii) 概念のリンクを、上位から下位に向けてたどることができる。
それにより、上位から、下位に位置する概念を知ることができる。

(vi)と(vii)の結果は、次のことを意味している。

(viii) 意味解析結果がＤＣＤ形式であれば、それに対する質問応答は、Prologに組み込みの機構をそのまま利用して行うことができる。

このように、ＤＣＤを用いると、意味解析に必要なプログラムの大部分を、Prologに組み込みの基本計算機構で代用することができる。また、計算効率も、フレームをリスト構造で表現する場合に較べ、改善されることが期待できる。

2.2 ＤＣＤの問題点

2.1で述べたように、ＤＣＤを用いると、意味解析用プログラムの大部分が、Prologの基本計算機構で代用でき、効率の上からも有効であると認められる。しかし、このＤＣＤを用いて意味解析を行うと、意味的に正しくない文も、正しい文として受理してしまうことがある。この問題点について、以下、意味解析の実例を取り上げ説明する。解析には、図2.1の open に関するＤＣＤ形式の辞書を用いとする。

case 1 : exp.) "* The key opened a debate ."

" a debate "は、(2-5)を用いて解析され、深層格 object が与えられる。次に、" the key "が(2-3)を用いて解析される。

このような意味的に異常な文の意味解析が成功してしまう理由は、

open に関する複数の別々のフレームをまとめて、１つのフレームとして記述してしまっているため、正しいとして受理する文の範囲が広がってしまっている

ということである。図2.1で記述されているフレームは、実際には、少なくとも３つのフレームとして、別々に記述されるべきなのである。

case 2 : exp.) " * We opened a debate with a key ."

" a debate "は、(2-5)を使って解析され、また、" a key " , " we "はそれぞれ、(2-6),(2-1)を用いて解析される。

このような意味的に異常な文を受理してしまう理由は、

(2-6)の withスロットについての規則は、目的語(obj)として解析したものが、上位概念として thingOpen を持つ場合にしか適用できない、ということが規則中に記述されていない

ということである。従って、case 2の問題点を解決するには、(2-6)の規則中に、何らかの形で、現在の時点での意味解析の途中結果を参照して、その内容を検査するプログラムが必要であるということになる。

case 3 : exp.) " * I opened ."

" I "が、(2-1)を用いて解析される。

この場合に、" * I opened ." という意味的に異常な文を受理してしまうのは、

open に関する３つのフレームそれぞれにおいて、文型パターンとの関連で、ど

のスロットは不可欠で、どのスロットはなくてもよいかという記述がなされていない

からである。従って、case 3の問題点の解消のためには、ＤＣＤによる解析と、文型パターンに関するチェックとの間につながりを設けてやる必要がある。

以上、ＤＣＤの問題点について述べてきたが、まとめると次のようになる。

① 異なるフレームをまとめて、１つのフレームとして記述している。
② スロット間の共起関係を記述できない。
③ スロットの必須・任意性についての記述ができない。

2.3では、これらの問題点に対する解決策を述べる。

2.3 ＤＣＤの改良

2.2で述べた問題点①～③に対する解決策は、まとめると次のようになる。

(1) 異なるフレームは、別々のフレームとして記述する。
(2) フレーム中の意味制約条件に、他のスロットのフィラーに関する条件も書けるようにする。
(3) ＤＣＤによる意味解析と、文型パターンのチェックとの間に、関連性をつける機構を設ける。

以下、それぞれについて、実際にＤＣＤに加えた改良点について説明する。

case 1で、図2.1に記述されたフレームは、本来、それぞれ別々のフレームとして記述されるべきであることを述べた。そこで、ＤＣＤの中に、意味規則番号なる情報を付加できるようにし、図2.2のように、open に関する３つのフレームをＤＣＤ形式で記述できるようにした。図2.2で、見出し語とスロット名の間に書かれた数字が、意味規則番号を表わし、同じ番号を持つ規則が、全体で１つのフレームを表している。この意味規則番号を意味解析過程でその途中結果とともに保持し、それを解析に用いることで、case 1のような例を回避できるのである。

さて、次に(2)についてであるが、上のcase 2の場合に、意味的に異常な文を正しく排除するためには、withスロットについて、下のような意味制約条件が必要となる。

```
sem( Filler , isa:instrument ) holds
  where
    there is sem( V , object : X ) in Result
    and
    sem( X , isa:thingOpen ) holds
```

上のような意味制約条件を、ＤＣＤで記述するには、case 2で述べたような、現在の意味解析の途中結果を参照・検索するプログラムが必要となってくる。また、それと同時に、いわゆるデモンと同等の動作をする機構が、意味解析プログラムには必要である。これは、関係節など、目的語となる名詞句が左外置された構文中の場合のように、用言を修飾する前置詞句を解析する際には、まだ objスロットが満たされていない場合もありうるということに対処するためである。必要なプログラムをＤＣＤに付加し、上の意味制約条件をＤＣＤで表現したのが図2.2の(2-8)である。(2-8)のような withスロットの記述により、case 2のような意味的に異常な文を排除することができるようになった。

最後に、(3)についてであるが、図2.2の open についての３つのフレームを観察すると、文型パターンとの関連で、

フレーム 1 について、subjスロットが必須、
フレーム 2 について、subj,objスロットが必須、
フレーム 3 について、subj,objスロットが必須、withスロットが任意

であることがわかる。英語の場合には、ここで言う必須スロットが、文を解析し終えた時点で未解析なまま残っていると、意味的に異常になるわけである。そこで、解析には、(1)で述べた意味規則番号とともに必須スロットのリストも用い、必須スロットについて解析を行なう際には、必須スロットのリストから解析したスロットを削除していくという手法を

とった（この必須スロットのリストは、HG(Head Grammar)[Pollard 84]におけるSUBCAT feature と同等のものである。）。そして、文の解析を終了した時点で、必須スロットのリストにまだ要素が残っている場合には、その規則についての解析は失敗することになる。

```
sem( open , 1 ^ subj : N ^ In ^ Out ) :- nonvar( N ) ,
   ( sem( N , isa:event ) ;
     sem( N , isa:thingOpen ) ) ,
       addProp( object : N , In , Out ) .
sem( open , 2 ^ subj : N ^ In ^ Out ) :- nonvar( N ) ,
   ( sem( N , isa:instrument ) ,
       addProp( instrument : N , In , Out ) ;
     sem( N , isa:wind ) ,
       addProp( reason : N , In , Out ) ) .
sem( open , 2 ^ obj : N ^ In ^ Out ) :- nonvar( N ) ,
   sem( N , isa:thingOpen ) ,
     addProp( object : N , In , Out ) .
sem( open , 3 ^ subj : N ^ In ^ Out ) :- nonvar( N ) ,
   sem( N , isa:human ) ,
     addProp( agent : N , In , Out ) .
sem( open , 3 ^ obj : N ^ In ^ Out ) :- nonvar( N ) ,
   ( sem( N , isa:event ) ;
     sem( N , isa:thingOpen ) ) ,
       addProp( object : N , In , Out ) .
sem( open , 3 ^ with : N ^ In ^ Out ) :- nonvar( N ) ,
  ( member( sem( V , object : X ) , In ) ->
    ( sem( X , isa:thingOpen ) ,
      nonvar( N ) , sem( N , isa:instrument ) ,
      addProp( instrument : N , In , Out ) ) ;
    addProp1( demon( sem( V , object : X ) ^
                sem( V , 3 ^ with : N ^ TIn ^ TOut ) ) , In , Out ) ) .
                                                    …… (2-8)
```

図2.2　ＤＣＤによる open の辞書記述（改訂版）

３　辞書記述用言語ＳＲＬ／０

2.3で述べたＤＣＤの改良により、古い版のＤＣＤにおける意味的に異常な文の受理という問題は解消できるようになった。しかし、図2.1の記述から分かるように、ＤＣＤは、ユーザが記述するにはその記述性において問題があり、また、今回の改良により、その記述はますます複雑さを増し、ユーザがそのままＤＣＤで辞書記述するのは、もはや困難である。そこで、辞書記述用の高水準言語ＳＲＬ／０を設計した。図2.3に open についてＳＲＬ／０で記述した例を示す。この記述がトランスレータにより図2.2の形のDCD節に変換されることになる。

```
open ::
   [ subj   $ isa:event ;
              isa:thingOpen              => object ]
    ::
   [ subj   $ isa:instrument             => instrument ;
              isa:wind                   => reason     ]
   [ obj    $ isa:thingOpen              => object ]
    ::
   [ subj   $ isa:human                  => agent      ]
   [ obj    $ isa:thingOpen ;
              isa:event                  => object     ]
   ( with   $ isa:instrument
              where
              obj!caller isa:thingOpen   => instrument )
   ( with   $ isa:animal                 => coagent )
    ::
   ((at ;
     in)    $ isa:place                  => location ).
```

図2.3　ＳＲＬ／０による open の辞書記述

以下、このＳＲＬ／０のシンタックスについて若干説明する。

1　'#'の間に記述されているスロットの並びが１つの意味規則を表している。

2　AND,OR,NOTの記法はすべて、Prologにおけるものと同じ ',' , ';' , 'not' を用いている。

3　任意スロットは、そのスロット名を '(' , ')' で囲む。

4　すべての意味規則に共通の任意スロットは、それらの規則とは別に最後に記述する。

5　他のスロットのフィラーを参照する記法として

　　スロット名!caller

という形を用いている[Mukai 85]。

また、ＳＲＬ／０を用いて object に関する知識を記述した例、およびそのＤＣＫＲによる対応表現を付録に示す。

4　おわりに

以下に本研究の結論および今後の課題を述べる。

(1) 本研究では、ＤＣＫＲを応用した辞書記述形式ＤＣＤの問題点を指摘し、その改良を行なった。この改良により、スロット間の共起関係のチェックや、必須，任意スロットに関する処理がＤＣＤの枠組内で実現できるようになった。

しかし、2.2で述べた意味的に異常な文は、特別な文脈が与えられれば、正常な文として解釈できる場合もある。現在のＤＣＤはこのような状況には対処できず、すべて意味的に異常な文として排除してしまう。今後は、意味的に異常な文でも、正常な文として受理できるような文脈が存在する場合には、排除しないようなメカニズムをＤＣＤに付加しなければならない。

(2) また、ＤＣＫＲを機械語レベルの形式として考え、ユーザが辞書項目を記述する辞書記述用の高水準言語ＳＲＬ／０を設計した。
今後は、ＤＣＫＲで記述された知識のうち、まだＳＲＬ／０で記述を試みていない、例外に関するものや、ＤＣＫＲの[小山 85]以後の拡張点[小山 86]についても記述できるように、ＳＲＬ／０をヴァージョンアップしていく予定である。

(3) また、このＳＲＬ／０を含めて、辞書の開発環境を整備していく必要があり、その中でも特に、辞書項目の修正、追加に伴う、辞書の保守・管理を行なうプログラムを開発する必要がある。

謝辞

貴重な時間を割き、この研究に関して、熱心に討論していただいた田中研究室の諸氏に感謝致します。

参考文献

[小山 85]：小山晴生，田中穂積，" Definite Clause Knowledge Representation ――Prologによる structured object の表現形式と推論――"，Proceedings of the Logic Programming Conference '85 , 4.3 , 1985 .

[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論理と自然言語特集号（予定），1986 .

[田中 85]：田中穂積，池田光生，奥村学，" Definite Clause Dictionary ――Prologによる辞書項目記述と意味処理"，Proceedings of the Logic Programming Conference '85 , 12.1 , 1985 .

[奥村 86]：奥村学，日本語理解システムに関する基礎的研究，東京工業大学大学院修士論文 , 1986 .

[Pollard 84]：Pollard,C. , Generalized Phrase Structure Grammars, Head Grammars, and Natural Language , Ph.D.dissertation , Stanford University , 1984 .

[Mukai 85]：Mukai,K. , " Unification over Complex Indeterminates in Prolog " , Proceedings of the Logic Programming Conference '85 , 11.1 , 1985 .

[小山 86]：小山晴生，Definite Clause Knowledge Representation―― Prologによる structured object の表現形式と推論――，東京工業大学大学院修士論文 , 1986 .

付録

```
human ::
    [ isa : mammal ] .          →    sem( human , Prop , N ) :-
                                         isa( mammal , Prop , [ human | N ] ) .

mammal ::
    [ bloodTemp : warm ]        →    sem( mammal , bloodTemp : warm , _ ) .
    [ isa : animal ] .          →    sem( mammal , Prop , N ) :-
                                         isa( animal , Prop , [ mammal | N ] ) .

animal ::
    [ isa : creature ]          →    sem( animal , Prop , N ) :-
                                         isa( creature , Prop , [ animal | N ] ) .
    [ hasa : body ] .           →    sem( animal , Prop , N ) :-
                                         hasa( body , Prop , [ animal | N ] ) .

creature ::
    [ age : X                   →    sem( creature , age : X , N ) :-
      when                               bottomOf( N , B ) ,
    { X is 1986 - birthYear!caller }] .  sem( B , birthYear : Y , _ ) ,
                                         X is 1986 - Y .
A ::
    [ isa : csFaculty           →    sem( X#P , isa : csFaculty , _ ) :-
      if B isa : csStudent ,             sem( Y#Q , isa : csStudent , _ ) ,
         B adviser : A ,                 sem( Y#Q , adviser : X#P , _ ) ,
         { A ¥== B } ] .                 X#P ¥== Y#Q .

A ::
    [ worksIn : B               →    sem( X#P , worksIn : Y#Q , _ ) :-
      if B isa : department ,            sem( Y#Q , isa : department , _ ) ,
         B manager : A ] .               sem( Y#Q , manager : X#P , _ ) .

A ::
    [ boss : B                  →    sem( X#P , boss : Y#Q , _ ) :-
      if A worksIn : C ,                 sem( X#P , worksIn : Z#R , _ ) ,
         C manager : B ,                 sem( Z#R , manager : Y#Q , _ ) ,
         { A ¥== B } ] .                 X#P ¥== Y#Q .
```