U.D.C. 〔681.32.06:159.95〕:800.92

# 知識工学基本ソフトウェア

## Basic Softwares for Knowledge Engineering

石原孝一郎* *Kôichirô Ishihara*
広瀬　正* *Tadashi Hirose*
芳賀博英* *Hirohide Haga*
近藤秀文* *Hidefumi Kondô*

日立製作所は，知識工学応用システムを構築するための基本ソフトウェアとして，汎用核言語LONLI，知識処理用汎用言語のS-LONLI及び知識ベース管理システムを開発している。

LONLIは推論機構を内蔵した知識記述言語として注目されているPrologと同じ分野の言語で，漢字処理や他言語とのインタフェースなど実用性を重視している。S-LONLIは，LONLIのもつ機能のほか対象指向機能などを追加し，応用システムの記述を容易にした。知識ベース管理システムは，知識を類別し，リレーショナルデータベースを利用して，知識の蓄積，推論検索を行なうシステムである。これら三つのシステムは，試作を終わり試用を通して評価・改良を進めている。

## 1 緒　　言

知識工学を適用したシステムを構築するための基本ソフトウェアについて述べる。「知識」を活用するには二つの重要な面があり，第一は知識の記述とその運用(知識を用いた推論)であり，第二は知識の獲得，蓄積，検索である。第一の知識の記述あるいは表現(Knowledge Representation)については多くの提案・研究がなされている[1]が，最近では第5世代コンピュータプロジェクトの核言語に採用されたPrologが注目を浴びている。2章では，このPrologと同じねらいをもったLONLI(Logic Oriented Language Inferencer)について述べる。LONLIを用いて，知識工学応用プログラムを書くこともできるが，むしろこれは基本的な核言語あるいはシステム記述言語として位置付けることができ，更に目的に応じた知識工学用のシステムが必要となる。3章のS-LONLI(Super-LONLI)は，このような目的でLONLIの上に対象指向の機能と，フレーム型[1]の知識表現を可能にした知識処理用の汎用言語システムである。前述した第二の面の知識の獲得，蓄積，検索については一般に知識表現に比べて研究が遅れているが，知識の分類と検索方法に関する日立製作所の今までの研究成果を4章に述べる。

以上のように，LONLIは知識工学応用システム構築のためのシステム記述言語として，実用化に必要な機能(主として大形機用に漢字処理，他言語インタフェース，データベースインタフェースなど)を充実させたものであり，LONLIをベースにエンドユーザー向けの知識処理用汎用言語S-LONLIや知識ベース管理システムを開発している。

本号掲載の別論文「知識処理のための推論ソフトウェア」で紹介するEUREKAファミリーは，特に中小形機を用いたシステム制御分野を対象に，応用システムの記述の容易化，推論処理の高速化をねらいとして開発した問題向き推論システムである。これらの幅広い要求にこたえられるLONLI，S-LONLI，EUREKAの開発により，ビジネス，システム制御両分野にわたり実用的なエキスパートシステムを短期間で構築することができる。

## 2 システム記述言語LONLI

LONLIは実用的知識情報処理システムの構築に利用可能な拡張Prolog系言語処理系である。実用的な応用システムの構築には少なくとも，

(1)　十分なメモリ空間と妥当な実行性能
(2)　既に蓄積されたソフトウェアやデータベース財産の利用など，既存システムとの結合の容易さ
(3)　プログラム開発環境の充実

が必要である。これらの要求にこたえるべくLONLIは汎用計算機(HITAC MシリーズVOS3：Virtual-storage Operating System 3)上で動作し，FORTRAN言語，PL/I言語プログラムとの結合機能(他言語インタフェース)，関係データベースインタフェース，デバッグ支援機能をもつ[2]。

### 2.1 LONLIの言語仕様

基本的には現在最も広く利用されている英国エジンバラ大学で開発されたDEC-10 PROLOGの仕様に準拠している。

プログラムは解くべき問題の性質や事実関係を，「事実」あるいは「規則(ルール)」という形式で宣言的に表わした文の集合から構成される。**図1**に簡単なプログラム例を示す。文①～文⑨が「リンゴの色は赤い」，「リンゴは果物である」などの事実を表わし，文⑩，文⑪が「ロバは色の赤い野菜を好む」，「ロバの好物はクマの好物である」などの規則を表わす(「:-」記号の右側が条件を，左側が結論を表わす)。プログラムの実行は，ある事実の成否を問い合わせる形式(質問)で行なわれる。システムは質問の事実と一致する事実あるいは規則の結論部が存在するか否かを特殊なパターンマッチング(ユニフィケーションと呼ぶ。)を繰り返しながら調べる。規則の結論部と一致する場合はその条件部の成否も自動的に調べ，解を求める。同図の文⑫，文⑬はそれぞれ「ロバの好物は？」，「ニンジンを好む動物は？」を問い合わせている。問い合せに対する処理は，プログラム(事事と規則)をあらかじめ特殊な内部コードに変換しておくことにより高速に実行される。

LONLIには上記の基本機能に加えて，

(1)　漢字，実数，配列の取扱い
(2)　TSS(Time Sharing System)コマンドインタフェース
(3)　他言語(FORTRAN，PL/I)インタフェース
(4)　関係データベース(RDB1：Relational Data Base 1)インタフェース

* 日立製作所システム開発研究所

例

プログラム

色(リンゴ, 赤). …① 色(ニンジン, 赤). …②
色(バナナ, 黄). …③ 色(トマト, 赤). ……④
果物(リンゴ). ……⑤ 野菜(ニンジン). ……⑥
果物(バナナ). ……⑦ 野菜(トマト). ……⑧
好物(ウサギ, ニンジン). ……⑨
} 事実

好物(ロバ, X) :- 色(X, 赤), 野菜(X). …⑩
好物(クマ, X) :- 好物(ロバ, X). ……⑪
} 規則

| ?- 好物(ロバ, X). …⑫ } 質問
X=ニンジン ;
X=トマト ; } システム応答
no
| ?- 好物(Y, ニンジン). …⑬ } 質問
Y=ウサギ ;
Y=ロバ ;
Y=クマ ; } システム応答
no
| ?-

図1 簡単なプログラム例 プログラムは「事実」と「規則(ルール)」の集まりである。実行は，それらに関する「質問」の形式で行なわれる。

(5) デバッグ支援

などの実用機能をもっている[3),4)]。

## 2.2 適用例と効果

**図2**(a)はLONLIを用いた日本語質問応答システム(行先案内システム)の作成例である。同図(b)に示した単語の組み合せで目的地と希望到着時刻を入力すると，最短時間の電車・バス乗り継ぎスケジュールを生成する〔同図(c)〕。地名・地図データ及び電車・バスのダイヤデータは，外部のデータベース(RDB1)から直接取り込む。また，CRT(Cathode Ray Tube)画面制御はFORTRAN言語で記述されたプログラムで行なっている。この例は簡単ながら，自然語の解析とエキスパートシステムとしての解探索の両方を含んでおり，論理形言語の記述力を示す例となっている。LONLIで139行(42文)の上記プログラムをFORTRAN言語で記述したところ，CRT画面制御部を除いて約1,500行を要した。

```
%     MAIN ROUTINE
:- extassert(fortran(sclear,'SCLEAR')).
:- op(100,xfx,:).
start :-
    repeat,sclear,
    write(' お帰りの乗り物を探します。質問を次の様に入力して下さい。'),
    write(' 例えば，"17時までに東京に行きたい" の要領です。'),nl,
    read(X),explode(X,L),入力文解析(_,T,D,A,L,[]),最適ルート(T,A,S),nl,
    write(' お帰りのスケジュールをお教えします。'),nl,nl,
    出力(S),nl,get0(Y),fail.
最適ルート(_,_,_)    :- abolish(f,2),asserta(f(00:00,[])),fail.
最適ルート(TT,A,_)   :- ルート(研究所,TT,P),
                        スケジュール(D,P,A,S),f(D1,_),
                        時刻比較(D1,D),abolish(f,2),assert(f(D,S)),fail.
最適ルート(_,_,S)    :- f(_,S).
出力([get(X,Y,D,A)|R]) :- tab(6),
                        write(X),write('発 '),時出力(D),write(' '),
                        write(Y),write('着 '),時出力(A),nl,出力(R),!.
出力([])  :- !.
時出力(X : Y) :- (X < 10,write('0');true),write(X),write('時'),
                 (Y < 10,write('0');true),write(Y),write('分').

:- write('    "start." と入力して下さい。'),nl.


%     PARSER
:- extdb('RDB1'(地名(char(10)),'CHIMEI',opt(extent,refer,imidiate),pw)).
入力文解析(F,T,D,A) --> 文節1(F,T,A), 文節1(F,T,A),動詞1.
文節1(F,T,A)        --> 時刻文節(A),([に];[ま,で,に]).
文節1(F,T,A)        --> 地名(T),([へ];[に]).
時刻文節(T)         --> 時(N),([時];[時,頃];[時,過,ぎ]),{T=N:00}.
時刻文節(T)         --> 時(N),[時,半],{T=N:30}.
時刻文節(T)         --> 時(H),[:],分(M),{T=H:M}.
動詞1 --> ([行,き,た,い];[着,き,た,い];[ゆ,き,た,い];[行,け,ま,す,か]).
時(N) --> 数(N),{0=<N,N=<24}.
分(N) --> 数(N),{0=<N,N=<59}.
数(T) --> [N,M],{数字(N,N1),数字(M,M1),!,T is N1*10 + M1}.
数(T) --> [N],{数字(N,T)}.
数字('1',1). 数字('2',2). 数字('3',3). 数字('4',4). 数字('5',5).
数字('6',6). 数字('7',7). 数字('8',8). 数字('9',9). 数字('0',0).
地名(T,X,Y) :- 地名(T),explode(T,W),append(W,Y,X).
```

```
%     ROUTE FINDING
:- extdb('RDB1'(経路(char(10),char(10),int),'KEIRO',opt(extent),pw)).
ルート(F,T,P)  :- ルート探索(F,T,Q),反転(Q,P).
ルート探索(X,Y,P)  :- ルート探索1(X,Y,[X],P).
ルート探索1(X,Y,P,[Y|P]) :-
        (経路(X,Y,_);経路(Y,X,_)),S+集合要素(Y,P).
ルート探索1(X,Y,P,Q) :-
        (経路(X,Z,_);経路(Z,X,_)),S+集合要素(Z,P),
        ルート探索1(Z,Y,[Z|P],Q).
集合要素(X,[X|_]).
集合要素(X,[_|Y]) :-集合要素(X,Y).
反転([X],[X]).
反転([X|Y],W) :-反転(Y,Z),追加(Z,[X],W).
追加([],X,X).
追加([A|X],Y,[A|Z]) :-追加(X,Y,Z).


%     SCHEDULE SEARCH
:- extdb('RDB1'(fdia(char(10),char(10),int,int,int,int),'DIA',
                                    opt(extent,refer),pw)).
スケジュール(D,[X,Y],A,[get(X,Y,D1,A1)]) :-
        ダイヤ(X,Y,D1,A1),
        乗り換え可(X,D,D1),乗り換え可(Y,A1,A),!.
スケジュール(D,[X,Y|R],A,[get(X,Y,D1,A2)|Sche]) :-
        スケジュール(A1,[Y|R],A,Sche),
        ダイヤ(X,Y,D1,A2),
        乗り換え可(Y,A2,A1),乗り換え可(X,D,D1),!.
乗り換え可(_,A,D) :- (var(A),A=D,!  ;  var(D),A=D,!  ;  時刻比較(A,D)).
時刻比較(Xh:Xm,Yh:Ym) :- (Xh < Yh ,!  ;  Xh==Yh,Xm=<Ym,!).
ダイヤ(X,Y,Dh:Dm,Ah:Am) :- fdia(X,Y,Dh,Dm,Ah,Am).
```

(a) ソースプログラムLONLIで139行(42文)

・・時　　に　　・・・へ　　行きたい
・・時頃　　までに　　・・・に　　着きたい
・・時過ぎ　　ゆきたい
ｘｘ：ｙｙ　　行けますか

(b) 入力可能な単語

お帰りの乗り物を探します。質問を次の様に入力して下さい。
例えば，'17時までに東京に行きたい'の要領です。
|: '18時半までに東京に行きたい'.

お帰りのスケジュールをお教えします。

研究所発　17時07分　柿生着　17時22分
柿 生 発　17時28分　新宿着　17時58分
新 宿 発　18時05分　神田着　18時21分

(c) 問い合わせ実行例

図2 行先案内システム(LONLI適用例) 疑似日本語文で行先と希望到着時刻を入力するとそれを解釈し，外部データベースにある交通機関の路線データ，ダイヤデータと参照して最適な経路及び乗り継ぎ時刻を出力する。

## 3　知識情報処理用高級言語S-LONLI

### 3.1　S-LONLI[5),6)]の設計思想

S-LONLIは，論理型言語LONLIの上に構築されたユーザー用高級言語であり，知識情報処理システム記述のための以下の機能を設けた。

(1)　知識データを簡潔に表現できること。

知識データは，個々のデータの独立性が高く，複雑な構造をもち，相互に関連しあって一つの体系を構成している。このようなデータを表現できるように，次の機能を実現した。

(a)　知識データのモジュール化機能

(b)　各モジュール内部の構造化機能

(c)　モジュールの階層化機能

(2)　推論制御を容易に表現できること。

推論制御も知識の一種であり，(1)の宣言型知識に対して，手続き型知識と呼ばれる。この推論制御を容易に表現できるように次の機能を実現した。

(a)　宣言型知識と手続き型知識の融合機能

(b)　並行処理，非決定処理の記述機能

S-LONLIは，マクロなレベルのモデリングに適した対象指向型言語[7)]をベースとし，ミクロなレベルのモデリングに適した論理型言語[2)]の機能を融合した複合機能型言語である。対象指向型言語をベースにすることにより，(1)の諸機能を実現し，論理型言語の機能を融合することによって，(2)の諸機能を実現した。

### 3.2　S-LONLIの機能の概要

#### 3.2.1　オブジェクト

S-LONLIのプログラムはオブジェクトの定義の集合である。オブジェクトとは，処理の対象となるデータとそのデータの処理手順をひとまとめにしたものである。このことを，**図3**の下半部に示したようなマルチウインドウシステムで，画面に表示されているウインドウを移動する操作を例にとって説明する。

従来言語でのプログラミングでは，ウインドウを表現するデータとウインドウを移動するルーチンが別々に存在していた。したがって，使用者はデータエリアにウインドウのデータを正しいフォーマットでセットし，その後移動ルーチンを起動する。つまりデータと操作は密接に関係しているにもかかわらず，あたかも独立しているかのように扱わねばならなかった。

一方，S-LONLIでは，ウインドウを一つのオブジェクトと考え，移動などの操作は，ウインドウ自身がもっている性質のようなものと考える。したがって，ウインドウを移動したいときには，ウインドウに「移動」という指令(これをメッセージと呼ぶ。)を送るだけで済み，手続きの詳細は全く知る必要がない。このようにオブジェクトを中心にプログラミングを行なうと，モジュール性が向上し，機能が明確に表現でき，モジュール間のインタフェースが簡潔になる。

S-LONLIのオブジェクトには，クラスオブジェクト(以下，クラスと呼ぶ。)とインスタンスオブジェクト(以下，インスタンスと呼ぶ。)の2種類がある。クラスは，例えば**図3**のウインドウシステムでは，ウインドウがどのような形であるとか，ウインドに送ることが可能なメッセージはどのようなものかといった，多くの実体に共通する性質を記述したオブジェクトであり，具体的な実体を生成するときのひな型となるものである。インスタンスは，同図のウインドウシステムでは，画面に表示された個々のウインドウを表わすオブジェクトで，クラスから生成された具体的な個々の実体に対応する。

マルチウインドウ システムの記述例

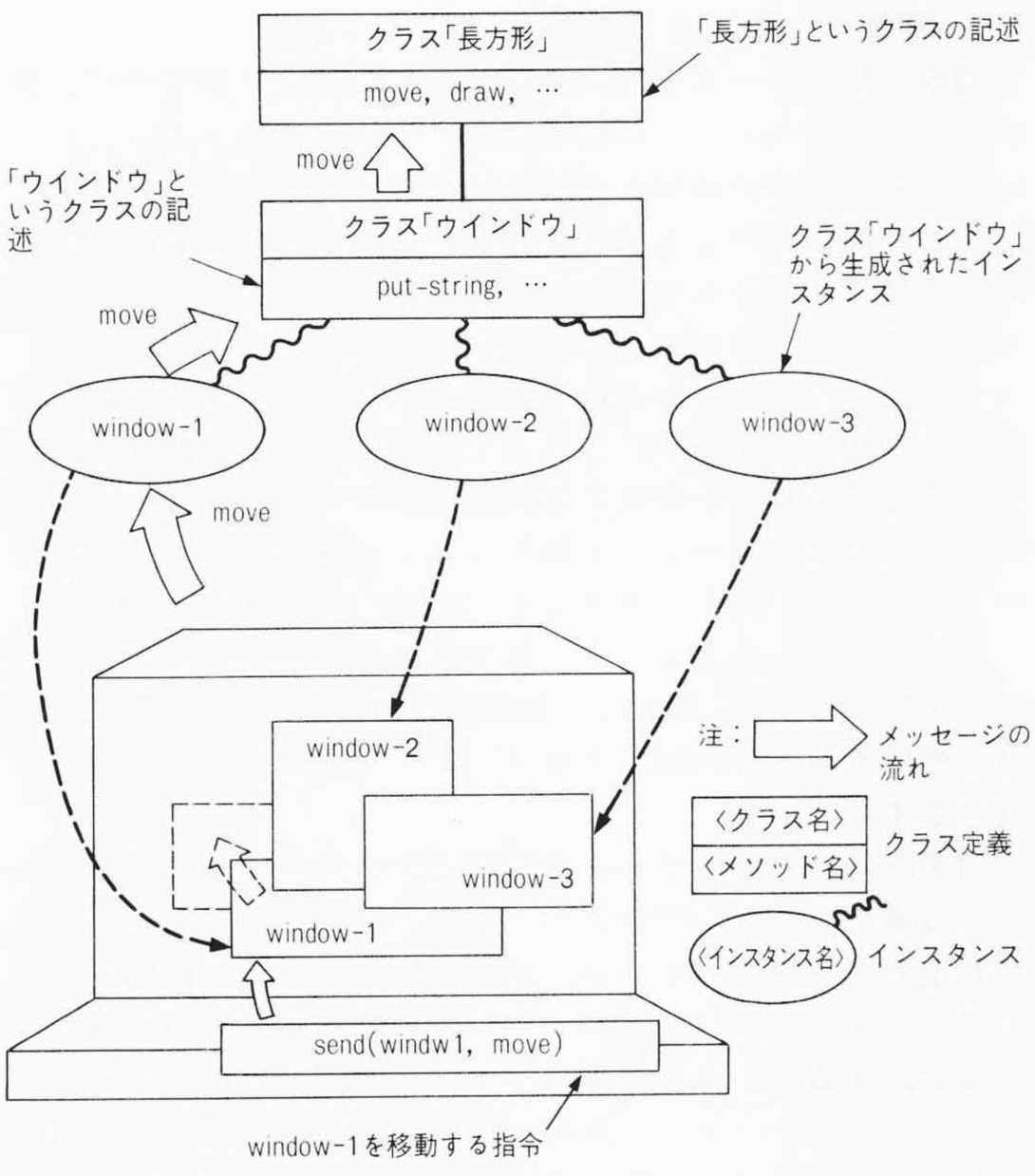


図3　実世界とオブジェクトの対応　　画面上の個々のウインドウはインスタンスオブジェクトで表現する。クラスオブジェクトは，インスタンスを生成するためのひな型となる。

#### 3.2.2　オブジェクトの階層化と機能の継承

S-LONLIでは，クラスを階層的に配置し，クラス間で機能を継承させることができる。**図3**の例では，ウインドウというクラスが長方形というクラスの下位に配置されている。この場合，クラス間の継承機能によって，下位のウインドウのクラスでは，上位の長方形のクラスに処理手続き(これをメソッドと呼ぶ。)を利用することができる。したがって，ウインドウのクラスでは，長方形のクラスで定義してある「移動」，「作画」といったメソッドを定義する必要はなく，「文字出力」といったウインドウに固有の操作のメソッドだけを定義すればよい。あるオブジェクトに定義されていないメッセージが送られてきた場合には，その上位のクラスに探しにゆく。例えば，ウインドウのクラスに「移動」というメッセージが送られてきた場合には，その上位のクラスである長方形のクラスの「移動」のメソッドが起動される。この継承機能によって，プログラムのモジュール化，記述量の大幅な削減などが図れる。

#### 3.2.3　メソッドの記述

メソッドは，メッセージを送られた場合の各オブジェクトの振舞を定義する手続きである。S-LONLIでは，メソッドの記述には，知識処理に必要な推論操作の記述に適したLONLIを使うこととした。これに加えて，対象指向プログラミングを支援するための幾つかの組込み述語を提供し，メソッドの記述性を向上している。**図4**はカウンタのオブジェクトをS-LONLIで記述した例である。この中のclear，up，setなどのメソッドは，LONLIと同一の形式で記述する。

```
befine-frame counter ;                    ←――  フレーム定義の開始を示す。
attribute   class ;
supers      root ;
i-vars      count (initial (0)) ;         ←――  カウントを示す内部変数の宣言
i-method clear ;
      clear :- set(0) ;                   }     countの初期値(0)の設定メソッド
 end ;
i-method up ;
      up :- Newcount is count + 1 ,       }
            change(count, Newcount),      }     countに1加えるメソッド
            show ;                        }
 end ;
 {method down もほとんど同じ}
 i-method show ;
       show :- write(count) ;             }     countの値を表示するメソッド
  end show ;
 i-method set ;
       set(Value) :- change(count, Value), }    countの値を変えるメソッド
                     show ;                }
  end ;
end counter                               ←――  フレームの定義の終了
```

図4 カウンタのプログラム例 四つのメソッドを含むS-LONLIのプログラムの例を示す。

## 4 知識ベース管理システム

本章では，知識ベース(蓄積知識の集合)の管理・検索機能を実現する知識ベース管理システムについて概説する。

### 4.1 知識ベース管理システムの基本機能

知識の種類については種々の考察があるが[8)~10)]，知識ベース管理システムが自動的に連想・推論検索(後述)を行なえるようにすることを目的とした知識として，次の3種類を設定した。

(1) 論理型知識

これには，「日立太郎の専門はロボット用言語である。」のような個別知識，個別知識を統合する知識(新しい概念の定義)などが含まれる。

(2) 一般化階層型知識

上位概念及び下位概念の関係を表わす知識

(3) 辞書型知識

同類語，反意語など用語に関する知識

知識ベース管理システムでは，上記3種類の知識を統一的に表現するため，ユニファイドスキーマと呼ぶ下記のような記述形式を用いる。

項目名｛項目データ｛項目名｛項目データ｛・・・｝｝｝｝

これによると，例えば，「日立太郎の専門はロボット用言語で，住所は横浜市である。」は，次のように表現される。

人名｛日立太郎｛専門｛ロボット用言語｝，住所｛横浜市｝｝｝

これらの知識が蓄積されている知識ベースを管理・検索するためには，次の三つの基本機能が必要である。

(1) 連想・推論検索機能

(a) 知識ベースの利用者が検索の道すじを正確に記述しなくても，システムが自動的に連想・推論し検索する機能

(b) 知識ベースの構造に関する予備知識がなくても，利用者が検索要求を行なえる機能

(2) 知識獲得機能

(a) 知識構造の定義がなくても，簡単に知識の蓄積が行なえる機能

(b) 蓄積済みの個々の知識から新しい知識を生成・蓄積する機能

(3) 無矛盾性管理機能

蓄積知識間の矛盾発生を防止する機能

上記以外に既存の蓄積知識を利用するために次の機能が有効である。

(4) 知識ベースとデータベースの融合機能

データベースは，どの知識も同じ項目名の組で構成される論理型知識とみなせるため，データベースを知識ベースとして利用する機能[11)]

### 4.2 知識ベース管理システムのシステム構成

**図5**に知識ベース管理実験システムの構成を示す[10),11)]。実験システムの中心的な機能は連想・推論検索である。これには，下記の五つの機能を組み合わせて実現している。

(1) 言い回しの異なる言葉を同一とみなす検索(**図5**の①)

(2) 蓄積されている知識に対して反意語を含むような問い合せでも推論して検索する機能(**図5**の②)

(3) 上位下位概念を利用する検索，例えば，ロボット用言語に関する知識が蓄積されているとき，ロボットに関する問い合せでもその知識を検索する機能(**図5**の③)

(4) 意味的に関連のある知識をたどる検索(**図5**の⑤，⑥)

(5) 部分的に一致する言葉をもつ知識の検索(**図5**の④)

**図5**の実験システムでは，知識ベースの蓄積手段として，RDB1を使用した。**図6**は，知識をRDB1のテーブル形式で蓄積した例である。同図の点線は，システムの自動的な検索経路を示す。

**図7**(a)は，特許のデータが既存のデータベースに蓄積されている場合を示す。データベースは，フィールド名をあらかじめ定めて〔同図の(a)では，発明名称，発明者，出願日〕，そのフィールドに従って大量のデータを蓄積するものである。そこで，知識ベース管理システムは，同図(b)に示すような定義によって，データベースのフィールド名を知識の主項目名と項目名に対応づけ，データベース中のデータを同じ主項目名，項目名をもつ大量の知識とみなせるようにする。この方法によって，知識ベース管理実験システムは**図6**に示す論理

型知識(特許)の代わりに，**図7**(a)に示す特許データベースを用いて，**図6**で説明した知識検索と全く同じ内容の処理を可能にする。

上記で述べた実験システムは，LONLI及びPL/I言語で記述してあるが，前節で述べた知識獲得機能のうち新しい知識の生成機能，知識の無矛盾性管理機能は実装していない。これらについては，今後順次実現を図っていく計画である。

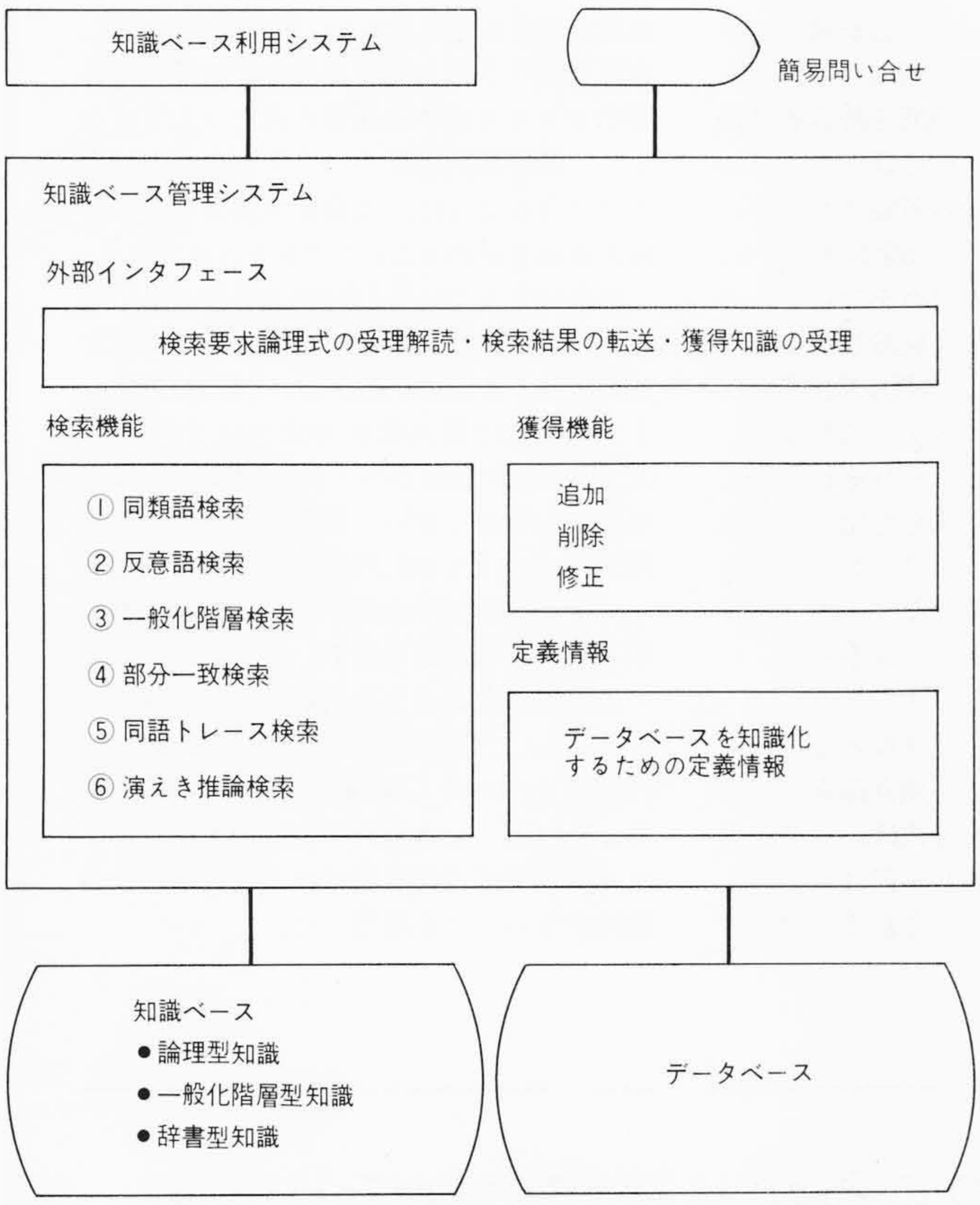


**図5　知識ベース管理実験システムの構成**　実験システムは，外部インタフェース，検索機能，獲得機能の三つの機能と，主メモリ上に記憶する定義情報及び二次記憶メモリ上の知識ベース，データベースから構成される。

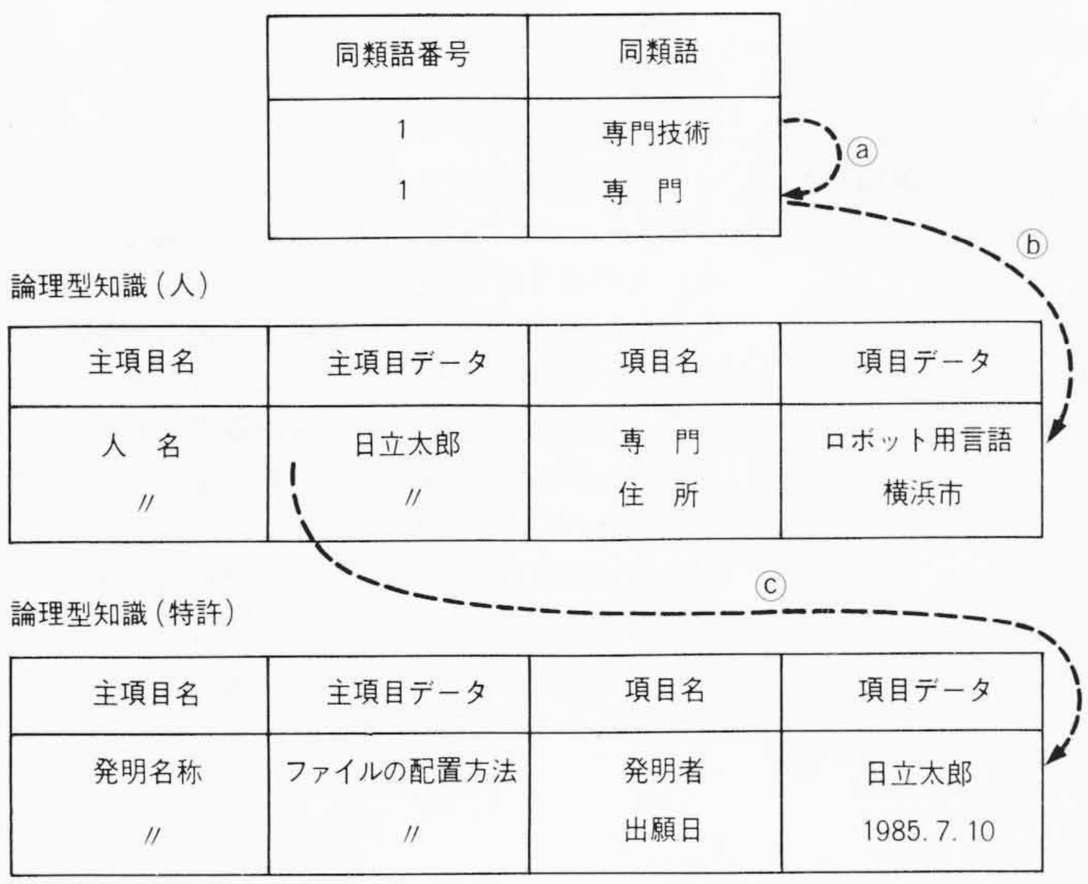


**図6　連想・推論検索の例**　「検索条件：発明者＝｛専門技術＝ロボット用言語｝，検索対象：発明名称」(発明者の専門技術がロボット用言語であるような発明名称)という検索要求に対して，システムは，ⓐ→ⓑ→ⓒの経路を自動的にたどって目的の知識を検索する。

| 発明名称 | 発明者 | 出願日 |
|---|---|---|
| ファイルの配置方法<br>・・・ | 日立太郎<br>・・・ | 1985. 7. 10<br>・・・ |

(a) 特許データベース

発明名称｛x｝発明者｛y｝
，出願日｛z｝｝｝

(b) データベースの知識化定義

**図7　データベースと知識の融合の例**　(b)に示すデータベースの知識化定義によって，(a)の特許データベースを，発明名称｛ファイルの配置方法｝発明者｛日立太郎｝，出願日｛1985. 7. 10｝｝｝という知識とみなす。

## 5 結　　言

知識工学応用システムを構築するための基本的なルーツとしての言語であるLONLIとS-LONLI及び知識ベース管理システムの概要について述べた。LONLIは，他言語やリレーショナルデータベースとインタフェースなど実用性を重視したPrologと同分野の言語であり，S-LONLIは，LONLIのもつ機能の上に，対象指向やフレームの機能を加えた知識処理用汎用言語である。知識ベース管理システムは，LONLIで記述され，LONLIから呼び出すことができる知識の蓄積・管理・検索システムである。この三つのシステムは，プロトタイプの試作を終わり，日立製作所社内での試用を重ねながら評価改良を進めている。知識工学全体として，大規模な応用開発は今後の課題であり，それに必要な基本ソフトウェアとして更に充実を図っていく考えである。

## 参考文献


1) A. Barr, E.A.Feigenbaum 編：Handbook of Artificial Intelligence(1980)
2) 迫田，外：論理型基本処理系「LONLI」の開発，情報処理学会第29回大会，4P-9(1984)
3) 広瀬，外：論理型基本処理系「LONLI」の機能と効果，情報処理学会第29回大会，4P-10(1984)
4) 広瀬，外：知識情報処理のための高機能言語「LONLI」，Computer Design **2**，2(1985)
5) 芳賀，外：知識情報処理システム記述用言語S-LONLIの概要，情報処理学会第29回大会，6Q-4(1984)
6) 芳賀，外：知識情報処理システム記述用言語S-LONLIの提案，日本ソフトウェア科学会第1回大会，2D-2(1984)
7) 米澤：オブジェクト指向プログラミングについて，コンピュータソフトウェア，**1**，1，29～41(1984)
8) Mylopoulous, J：An Overview of Knowledge Representation, Proceedings of the Workshop on Data Abstraction, Databases and Conceptual Modeling, ACM(Nov. 1980)
9) Minsky, M. L：A Framework for Representing knowledge, Psychology of Computer Vision, McGraw-Hill(1975)
10) 近藤，外：知識ベース管理システム—知識とデータの相違点に関する基本的考え方—，情報処理学会第31回全国大会(1985)
11) 小口，外：知識ベース管理システム—知識ベースとデータベースの融合方式—，情報処理学会第31回全国大会(1985)

論文抄録

# VLSI設計規則検証のための近接辺探索手法

日立製作所　築添　明・酒見淳也・他1名

**電子通信学会論文誌　J68－D，6，1344～1351（昭和60-6）**

LSIの大規模化，高集積化に伴い，その設計期間の短縮を図って，各種CADシステムが開発され使用されている。

設計規則検証CADは，マスクパターンの幅や間隔などの幾何学的距離が使用プロセスで決められた許容値未満である箇所を摘出し，それを違反候補として表示するプログラムである。その技術課題は大量データを高速に処理し，かつチップの高密度化のため多用される斜めパターンに対しても処理速度の劣化が少なく，違反箇所の検出漏れのないことを保証するパターン演算手法の開発である。

設計規則検証のためのパターン演算処理は，マスクパターンデータを向きをもつ辺であるベクトルで表現し，指定された許容値未満の間隔で近接するベクトルの組を探索する処理である。入力の並び順に注目ベクトルを決定し，探索範囲を限定せずにすべてのベクトルの組を検証する総当たり法は，探索処理時間がデータ数の二乗に比例するため，小規模のデータにしか適用できない。また，端点のy座標の順に注目ベクトルを決定し，y方向の探索範囲を許容値に限定する方法でも，設計規則違反となる斜めベクトルが検証漏れとなる可能性があり，探索処理時間がデータ数の1.5乗に比例する。

筆者らは，任意角度の斜めを含むデータに対しても設計規則違反の見逃しがなく，探索処理時間が入力ベクトル数に比例する性能をもつ2次元限定近接辺探索法を提案した。x，y方向の探索範囲はそれぞれ仮想スリット法，限定交点探索法を応用して限定する。仮想スリット法はパターンの輪郭取りや重なり部分の抽出を行なう論理演算手法，限定交点探索法は交点計算手法で，いずれも筆者らが考案した手法である。

実現方法としては，処理速度と主メモリ量の観点から辺探索方式と頂点探索方式の2方式を考えた。辺探索方式は，演算対象ベクトルを辺単位で管理し，設計規則違反のベクトルの両端点を共に含むように探索範囲を決める方式である。頂点探索方式は，演算対象ベクトルを頂点単位で管理し，必要な主メモリ量が辺探索方式より大きくなるが，設計規則違反のベクトルの片方の端点だけ含むように，辺探索方式より小さい探索範囲を決めることができる方式である。

製品化されているLSIマスクデータに対してプログラム評価した結果，頂点探索方式のほうが実用性が高いという結論を得た。主メモリ量は頂点探索方式のほうが1.5倍大きかったが，100万ベクトルでも100kバイト程度しか増加しない。探索処理速度は頂点探索方式のほうが2.7倍速く，100万ベクトルの探索処理CPU時間は100秒（使用計算機：HITAC M-280H）である。

頂点探索方式の2次元限定近接辺探索法を組み込んだ設計規則検証プログラムMACH（Mask Artwork Check Program）を開発した。MACHによるLSIの検証結果から，2次元限定近接辺探索法では違反箇所の検出漏れがないことを確認した。

# 0.45mass%C鋼のころがり接触疲れに及ぼす接線力の影響

日立製作所　波多野和好

**日本金属学会誌　49－7，495～500（昭60-7）**

ころがり接触を主要な動作とする軸受，歯車，圧延ロールなどで，接触部の滑りに伴う接線力は，ころがり接触破壊で大きな役割を果たしていると言われている。これまでに，接触部に作用する接線力の圧縮側から引張り側への移行に伴う寿命の低下，接触部に形成される塑性流動層と硬化あるいはX線半価幅との関連性などが報告されている。筆者は，十分にひずみ取り焼なまし処理した中炭素鋼（S45C）を供試材とし，接線力を制御できる二円筒式摩擦・摩耗試験機で，ころがり接触での接触面の硬さ，組織，有効ひずみなどの構造変化と微小き裂発生との関係を詳細に検討した。その結果，

(1)　ころがり接触部に作用する引張りの接線力の増加は，き裂及びPitting発生までの寿命を低下させる。

(2)　き裂発生と接触部表面の構造変化の間に相関があり，損傷（構造変化）の累積がある値に達すると，表面にき裂（主き裂）の発生が認められる。

(3)　接触部には接線力の作用方向に20～30度の傾き角度で塑性流動層が生じるが，これに直交して伝ぱする多数のき裂（副き裂）が認められるなどを明らかにした。

以上のことから，硬さ，有効ひずみ，微細結晶粒がほぼ一定になる時点は，転位密度が最大であることに対応するので，これは，繰り返し数に伴って増殖してきた転位が蓄積し，動けない状態に達したと解釈できる。したがって，接触応力及び接線力が，その後も繰り返し負荷されると，次の段階として転位が最も蓄積する箇所（サブ バウンダリなど）に微視的なき裂が発生し，これが伝ぱ拡大し，遂には主き裂になると推論できる。このように解釈することによって，ころがり接触表面層の各種挙動とき裂との関係が，より明確に証明できることを示した。一方，塑性流動層内に副き裂が観察されたことからPitting寿命は，この副き裂によって支配されていることが理解された。また，副き裂が関係して生じるPitting摩耗は，表面から大きな金属片の連続的な脱落を伴うので，主き裂発生時をもって寿命と考えるべきであることにも言及した。