TM-0067

CAD用知識処理言語—CADLOG—の開発

後藤　敏
(日本電気株式会社)

July, 1984

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

1. まえがき

近年，人間の思考や理解といった知的活動をいかに形式化して，コンピュータで実現するかという知識情報処理の研究が活発になっている。この研究の重要な課題として，問題解決・推論機能があげられる。問題解決・推論機能は，人間の思考メカニズムを明確にし，コンピュータにより人間に近い知的処理を行おうというものである。このような機能を実現するための言語として，述語論理型プログラミング言語 Prolog が注目されている。Prolog は柔軟なリスト処理と問題解決・推論のための基本機能を備えている。このため，エキスパートシステムの構築や，知識を表現する言語として注目され，実用化に向けて，改良，拡張の研究が行なわれている。[1, 2]

しかし，大規模問題を取扱うシステムの

図表の位置

を現在の Prolog でうべて走行するため，現在のハードウェア環境と（Prologの処理系では）、実行速度とメモリ使用量の点から見て問題がある。これは、引数の受け渡し，バックトラック時に代入を元に戻すための情報を保存するために多くのメモリを使うこと，述語はパターンマッチングによって呼出されるため，述語の数が多くなると，述語の格納，取り出しに時間がかかることが原因と考えられる。

一方，手続きが明確な問題に対しては，従来のアルゴリズムによって問題の解決をはかってきており，これらは FORTRAN 言語等により，効率のいいプログラムが作成されている。しかし，手続きが不明確な問題や手続きを見付けるのに試行錯誤を要する問題に対しては，FORTRAN 等の手続き型言語では，プログラム開発の効率が悪く，Prolog 等の論理型言語が適していると考えられる。

図表の位置

以上の点から，手続き型言語と論理型言語とを結合した言語を作り，おのおのの利点を生かした使い分けをすれば，より高度な知識情報処理を効率良く行なえると考えられる。また，既に，手続き型言語によって現在までに構築された（既存の）データベースや数多くのプログラム資産がFORTRAN言語とのリンク機能により，PROLOG上で有効に使用できる。

ここでは，Prolog言語の標準機能に，新たにFORTRAN言語とのリンク機能を加えたProlog インタプリタ（CADLOG）を提案し，

知識処理技術を用いたCADシステム（知的CAD）の実現の方法を説明する。

述語論理に基づくプログラミング言語で,

No. 4

図表の位置

2. CADLOGの特長

Prolog は一階述語論理に基づく, その CADLOG プログラムは、節(Clause)の集合である。 それぞれの節は記号 ':-' で結ばれる２辺から成り、左辺を頭部(Head)右辺を本体部(Body)と呼ぶ。論理的には、頭部は結論、本体部は条件と考えられる。

節は、次の３つの形をしている。

（１） $P:-Q_1,Q_2,Q_3,\cdots\cdots,Q_n.$

（２） $P.$

（３） $:-Q_1,Q_2,Q_3,\cdots\cdots,Q_n.$

（１）の形は、「Ｐを実行するには、Q_1 -- Q_n を実行すればよい」または「Ｐが成り立つことは、Q_1 -- Q_n がそれぞれ成り立つことである」という意味である。 （１）の形は、ルール節(Rule clause)と呼ばれる。

（２）の形は、（１）の本体部が空の特別な場合で、無条件でＰが成立する意

味である。　（２）の形は、ユニット節（Unit clause）と呼ばれる。　又、　'主張'、'宣言'（Assertion）と呼ばれることもあり、事実を述べる時に用いる。

（３）の形は、（１）の頭部がない場合で、「Q_1 から Q_n まで実行せよ」又　「Q_1 から Q_n まで同時に満たす解を求めよ」という意味である。　（３）の形は、ゴール節（Goal clause）と呼ばれる。

節の頭部は、単一の述語（Predicate）から成り、本体部は、複数個の述語から成る。　本体部の述語は、論理積のみで結合され、記号 ','は論理積を表わす。

（例）　LIKE(HANAKO,*X):-COLOR(*X,
RED) ,VEGETABLE(*X).

COLOR(STRAWBERRY,RED).

COLOR(TOMATO,RED).

FRUIT(STRAWBERRY).

VEGETABLE(TOMATO).

第１番目は、ルール節で「花子が好きなものは、赤い色の野菜である」を意味する。　２番目から５番目の節は、ユニット節である。　これらのプログラムに対して、

>:-LIKE(HANAKO,*X).

というゴール節を入力すると、*X　に TAMATO　がユニファイされて終了する。

ＣＡＤＬＯＧは、~~一階述語論理を基礎にして開発された~~ Prolog を基本にし、FORTRAN 言語とのリンクを可能にした　Prolog　インタープリタである。構文規則は、DEC-10版　Prolog[5] に基づき、文字は、大文字、数字、記号の３種であり、変数は '*' を付加して記述する。例えば、ＡＰＰＥＮＤの定義はＣＡＤＬＯＧでは、

APPEND([],*L,*L).

APPEND([*X;*L1],*L2,[*X;*L3]): -APPEND(*L1,*L2,*L3).

となる。

プログラムは、節によって構成され、この節はいくつかの述語を組み合わせて表現する。　述語の引数は、定数として、文字列、数値、構造を記述することができる。　算術演算子（加算、減算、乗算、除算）、数の比較、バックトラックに対する制御を行なう、REPEAT、！（カット）などが組込み述語として用意されている。

Prolog　では、処理するデータ間の関係を宣言したものがプログラムである。

しかし、手続き的処理が多い問題に関しては、記述しにくい場合がある。　また、プログラムの実行効率の面から考えると、既に処理手順が定まった処理は、手続き型言語のよって表現し、実行するほうが実用的である。　実行時間は、大規模問題を取り扱う場合、特に重要な要素となるからである。　このために、ＣＡＤＬＯＧは　Prolog　のプログラムから　FORTRAN サブルーチンをコールできる機能を持っている。　FORTRAN で書かれたプログラムとのリンクは、これに対応する述語によってとられる。　この述語のことを、外部ファンクション述語と呼ぶ。　この述語は、Prolog　の中では、頭文字に '$' を付加して他の述語と区別し、組込み述語と同様に取り扱われる。

FORTRAN 言語とのリンク機能を持っているため、この言語で構築されているデータ構造を使用することができる。 これによって、大規模なデータ処理が効率よく処理でき、実行時間も短縮される。

ＣＡＤＬＯＧでは、このデータのやりとりは、データをアクセスする FORTRAN サブルーチンを Prolog プログラムからコールすることによって実現できる。

ここで、FORTRAN で構築されているシステム内のデータ構造を用いる場合、1つ問題がある。 この問題とは、一般に Prolog ではバックトラック時に、以前代入された値を元の変数に戻すだけであり、もしデータ構造内のデータが変更されていても、更新されたままで、オルタナティブサーチを実行することである。

バックトラックが生じたということは、更新されたデータが誤りのデータであるかもしれない。誤りのデータが格納されたままで処理が進むと、求めた解が正しいかどうか疑わしい。 したがって、バックトラックが生じると、更新したデータを元に戻すことを考慮して、プログラムを組む必要がある。 プログラマは、

データを更新するか否かを常に意識し、バックトラックが起こると、どのような処理を行なわなければならないか考えて、プログラムしていく。　これは、プログラマに負担をかけるばかりでなく、プログラムを読みにくくする。　ＣＡＤＬＯＧでは、更新する外部ファンクション述語の中に、データのリカバリ処理を組み込んでおき、バックトラックが生じてこの述語まで戻った時に、組み込まれているリカバリ処理を行なう機能を持っている。　この機能によって、データ構造内のデータの整合性を常に保つことができ、プログラムも記述しやすくなる。　以下、ＣＡＤＬＯＧの特長的な機能である、外部ファンクション述語と、データのリカバリ処理について詳しく述べる。

２．１　外部ファンクション述語

Prolog の手続きの各引数は、それが入力変数であるか、出力変数であるかという入出力規定は、実行以前にはない。

手続き呼び出し時に、定数が渡される場合には、Call by Value のように働き、逆に変数が渡されて手続き呼び出し時、あるいは手続き実行時に、その変数の値が決まる場合は、Call by Referance のように働く。　しかし、FORTRAN 言語では、Prolog のような引数の評価ではなく、必ず入出力を規定しなければならない。　したがって、ＣＡＤＬＯＧでは，外部ファンクション述語の引数の入出力の整合がとれなければ、FAIL として、バックトラックが生じる。

例えば、FUNC という節を下記のように表現する。

```
FUNC(*X,*Y):-$PROC1(*X,*Y).
FUNC(*X,*Y):-$PROC2(*X,*Y).
FUNC(*X,*Y):-$PROC3(*X,*Y).
```

図表の位置

ここで、 $PROC1 は、第１引数が入力変数で、何らかの処理をした結果を第２引数にセットする FORTRAN サブルーチンに対応する外部ファンクション述語である。 $PROC2 は、変数の入出力関係がこの逆である外部ファンクション述語を表わしている。 $PROC3 は、両引数とも入力変数である時、この引数が正しいかどうかをチェックするFORTRAN サブルーチンに対応している。 ＣＡＤＬＯＧでは、外部ファンクション述語と FORTRAN サブルーチンとの対応は、引数の入出力関係から一意的に決まっている。

この FUNC 述語を用いて、SUB_GOAL1 という節を次のように定義する。

```
SUB_GOAL1(*X,*Y,*Z):-FUNC(*X,*Y),FUNC(*Z,*Y).
```

第１引数が定数、第２、３引数が変数でユニフィケーションされると、 $PROC1 、$PROC2に対する FORTRAN サブルーチンが順に実行され、第２、３引数の値が決定される。 第２引数が定数、第１、３引数が変数の場合は、 $PROC2 に対する FORTRAN サブルーチンが異なる引数で２回実行される。

このように、外部ファンクション述語を含んだ節では、引数の入出力関係の整合がとれなければ、バックトラックが発生して、オルタナティブの探索を行なう。

2・2 データのリカバリ処理

FORTRAN 言語とのリンクを可能としているため、FORTRAN で構築されたシステム内のデータ構造が使用できる。 システム内のデータ構造を使用する場合、外部ファンクション述語によって更新した後、バックトラックが起こり、この外部ファンクション述語まで戻ってくると、データを更新前に戻さなければならない場合がある。 つまり、データのリカバリ処理を必要とする場合である。 この

図表の位置

リカバリ処理は、データを更新する外部ファンクション述語に対応する FORTRAN サブルーチンに、常に更新前のデータを保持する処理を追加していくことによって、実現している。 バックトラックが発生して、更新した外部ファンクション述語まで戻ると、ＣＡＤＬＯＧは、保持しているデータを用いて、データの改復処理を行なう。 改復処理を行なう FORTRAN サブルーチンは、外部ファンクション述語によって異なるが、一対一で対応づけられている。

一方、データの更新後、バックトラックが発生してもデータのリカバリをする必要がない場合も考えられる。 ＣＡＤＬＯＧは、データのリカバリ処理を制御する組込み述語、RECON 、RECOFFが組み込まれている。 デフォルトは、RECONで，常にデータのリカバリを実行する。

RECOFF は、この述語がコールされた時点でこれまで保持していたデータをすべて解放し、RECOFF 以後の述語でバックトラックが発生しても、RECON がコールされるまで、データのリカバリは行なわない機能を持っている。

ＬＳＩの配線設計システムにおける設計仮定の１つの手順を、例にとって説明する。

図 1

図表の位置

図 1(a) は、ネットBの既配線によって、ネットAの一方のピンを妨害しているため、ネットAの経路が発見できない場合を表わしている。ネットとは、同一電位で直接結合すべき端子の集合である。

図 1は、以下の ①～③ の手続きを表わしている。

①. ネットBを削除する。
（図 1(b)）

②. ネットAを結線する。
（図 1(C)）

③. 削除されたネットを結線する。
（図 1(d)）

図 2に、これらの手順をCADLOGで表現したプログラムを示す。頭に '$' が付加しているものは、外部ファンクション述語であり、 '*' が付加しているものは変数である。 *NET_A は、未結線ネット番号、*NET_B は未結線ネットを妨害しているネット番号を意味してい

図2

る。 BLOCKING の節は、ネットを構成するピンの片方に妨害している既配線が存在した時に、その既配線のネット番号を求める処理を表わしている。 PROC 節の本体部の述語には、バックトラックが生じた時に、データのリカバリ処理が必要な外部ファンクション述語が含まれている。その述語は、 $DELETE 、$CONNECTである。 $DELETE は、既配線を削除する処理なので、リカバリ処理は再び結線することである。 $CONNECT は経路を発見する処理なので、リカバリ処理は、発見された経路を削除することである。

ここで、PROC 節の第３番目の述語 $CONNECT(*NET_A) が失敗した時、つまり、図 1(C) のような経路が発見されなかった時、バックトラックが発生する。 この場合、バックトラックが生じると、未結線ネットが多くなる。 配線設計システムの目的は、 100% 結線であるから、このままでは目的を達成することはできない。 したがって、元の状態に戻すには、 $DELETE がコールされた時、保持していたデータを用いて、 $DELETEとは反対の処理である $CONNECT に対応している FORTRAN サブルーチンをコールしてデータの改復を行なう。

このように、FORTRAN で構築されたシステム内のデータ構造を用いる場合でも、ＣＡＤＬＯＧはデータのリカバリ用サブルーチンを組み込むことにより、常にデータの整合性を保ことができる。

図表の位置

3. 処理系の概要

ＣＡＤＬＯＧは、インタープリタ方式を採用している。 Prolog では、節は独立したステートメントとして考えられるので、ＣＡＤＬＯＧインタープリタは、１節単位に情報をコード化している。

図 3に、インタープリタシステム構成を示す。 ＣＡＤＬＯＧインタープリタは、指定したファイル又は端末装置から節を

入力し、構文解析、文法エラーをチェックし、節を構成する変数、引数などの情報を、内部コードテーブルに編集する。

入力された節が、ゴール節の場合、述語解釈部で内部コードテーブルを走査しながら実行テーブルを作成する。実行テーブルは、いままでユニフィケーションされた述語の情報が格納されている。この実行テーブルに従って、探索が制御される。

図表の位置

<図4>

図4にゴール節が入力された時の処理の流れを示す。実行テーブルの初期化をし、述語を取り出し、述語名を表わすコードと、変数又は定数が実行テーブルに登録される。この述語が、組込み述語、外部ファンクション述語であれば、各々の実行部へ制御を渡し、結果を実行テーブルに書き込み、再び次の述語を取り出す。そうでなければ、内部コードテーブルをサーチし、ユニフィケーションが行なわれる。ユニフィケーションが成功すると、実行テーブルに結果が登録され、次の述語を取り出す。バックトラックは、ユニフィケーションが失敗した時、又は、外部ファンクション述語に対応するFORTRANサブルーチンの処理が失敗した時に発生し、必要であればデータのリカバリ処理をし、オルタナティブの探索を行なう。

外部ファンクション述語の実行部とのインターフェイスは、実行テーブルに登録されている述語を表わすコードと引数を、インターフェイス用テーブルにコピーすることによってとられる。外部ファンクション述語名を表わすコードに対応する FORTRAN サブルーチンの引数の数と、入出力関係は、CADLOGインタープリタのプログラムの中に、サブルーチン毎に組み込まれている。この情報を参照して引数の個数、入出力関係、データタイプのチェックを行なう。
FORTRAN サブルーチンは、外部ファンクション述語と1対1に対応がとれるので、引数の整合がとれるとコールして実行される。実行後、FORTRAN サブルーチンによって値が定まった変数は、実行テーブルに登録して、次の述語の取り出しを行なう。

図表の位置

4. 知的CAD

図5にCADLOGを用いた知的CADシステムの構成例を示す。この知的CADシステムは，知識ベース（Knowledge Database）とCADデータベース（CAD Database）の2種類のデータベースをもっている。知識データベースは，問題解決のための設計者の知識がルールとして格納されており，CADデータベースには，設計に関する情報（構造，特性データ）が格納されている。既存のFORTRANで書かれたCADシステムは，CADデータベースとの間で，データの読み，書きを行なう，設計を進める。より高度な設計を行なうために，設計者のノウハウを知識データベースにルールとして格納し，CADLOGにより，既存CADシステムと総合して設計を実行することができる。

CADLOGは，LSIの記述エキスパートシステム[6，7]を構築する言語と

図表の位置

1て作用し，新しい人工知能処理言語として，実用的にも十分，有効である
ことが確かめられている。

5．　あとがき

本稿では、Prolog の実用化に伴う拡張機能として、基本的機能に加え、FORTRAN 言語とリンク機能を持つ Prolog インタープリタ ＣＡＤＬＯＧ を提案した。 この追加機能により、処理方法が一定の手続き的処理の実行を高速に行なうことができる。 また、FORTRAN で構築されたシステム内のデータ構造とのアクセスも可能になり、大規模問題を取り扱う場合でも、このデータ構造を用いて、データ処理が効率よく実行できる。

図表の位置

参考文献

[1] 中島，" Prolog/KR の概要" ，情処，記号処理，18-5，1982.

[2] 梅村，中嶋，小長谷，幅田，島津，横田，" 文字処理を導入した Prolog:Shapeup について" ，Proc. of the Logic Programming Conference，1983.

[3] 藤田，後藤，" CAD 向け Prolog 型インタープリタ" ，情処大会第27回全国大会，pp 379-380，1983.

[4] R.Kowalski，" Logic for Problem Solving " ，Elsevier North Holland Inc.，1979.

[5] W.F.Clocksin and C.S.Mellish，" Programming in Prolog " ，Spring-Verlong，1981.

[6] 森，光本，藤田，後藤，" 配線エキスパートシステム" ，情処大会第28回全国大会，pp 1073-1074，1984.

[7] K.Mitsumoto，H.Mori，T.Fujita and S.Goto，" AI Applioach for VLSI Routing Problem" ，ISCAS'84，pp449-452 ，1984.

No.

図表の位置

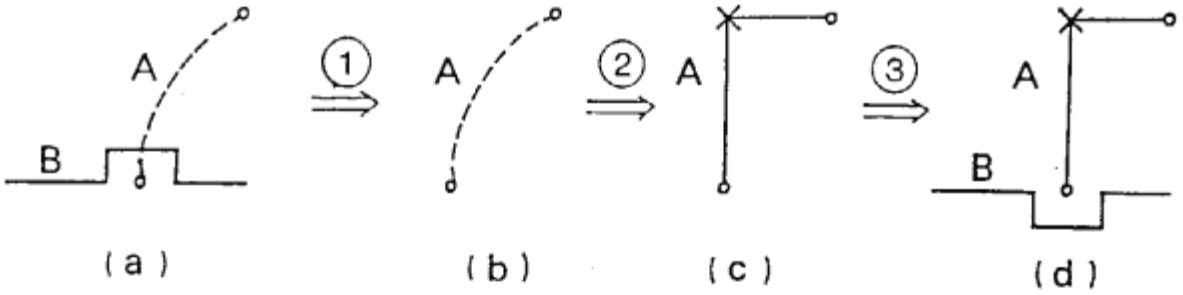
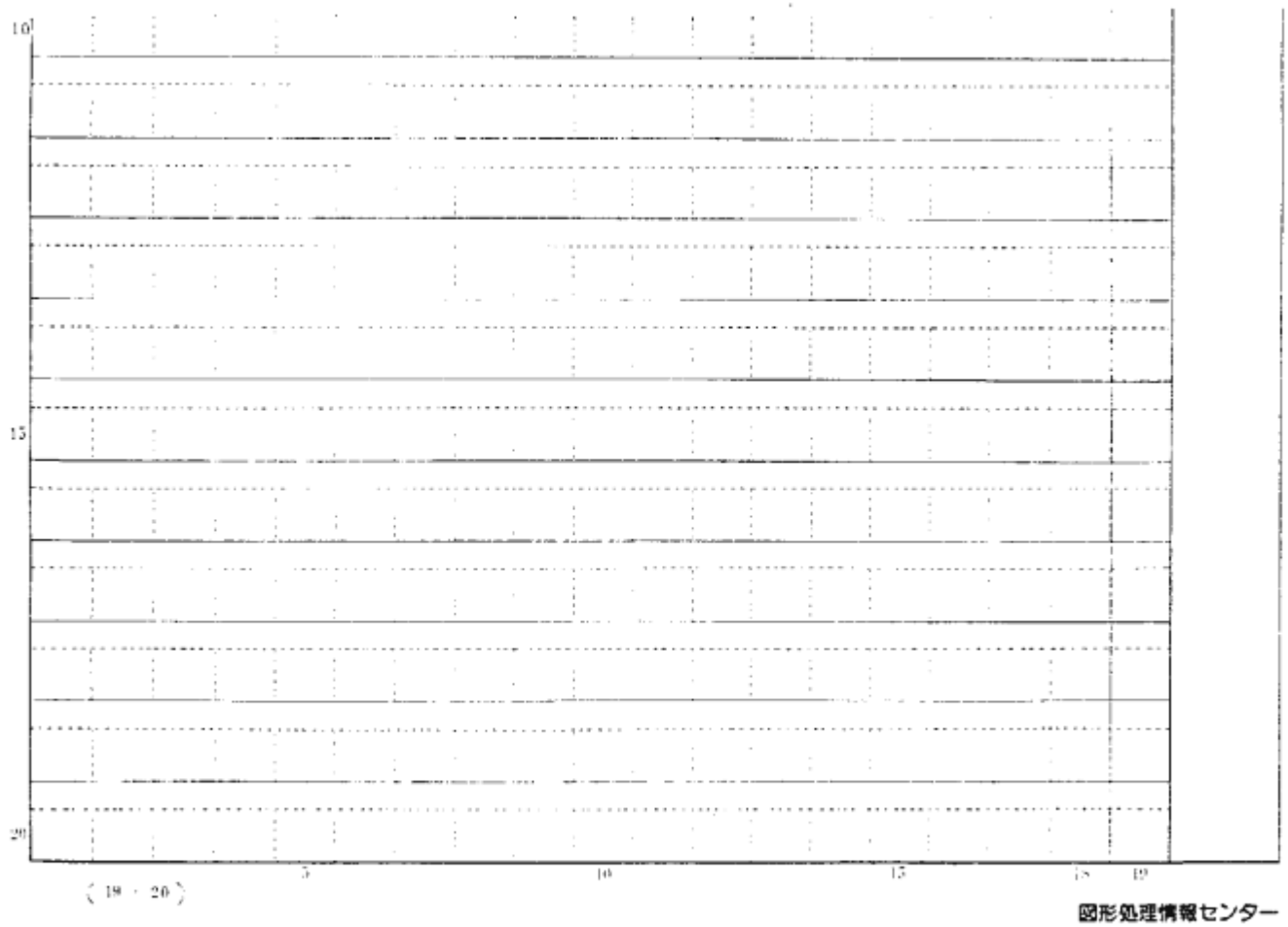


図1. 配筋設計の手続き例

```
PROC(*NET_U):-BLOCKING(*NET_U,*NET_B),
                $DELETE(*NET_B),
                $CONNECT(*NET_U),
                $CONNECT(*NET_B).
BLOCKING(*NET_U,*NET_B):-GETPIN(*NET_U,*LAY,:X,*Y),
                          EQ(*LAY,1),
                          DIRECTION(*Y,*DIR),
                         $GRID_SEARCH(*DIR,*LAY,*Y,*NET _B).
```

図２．　ＣＡＤＬＯＧのプログラム例



〈 19 · 20 〉

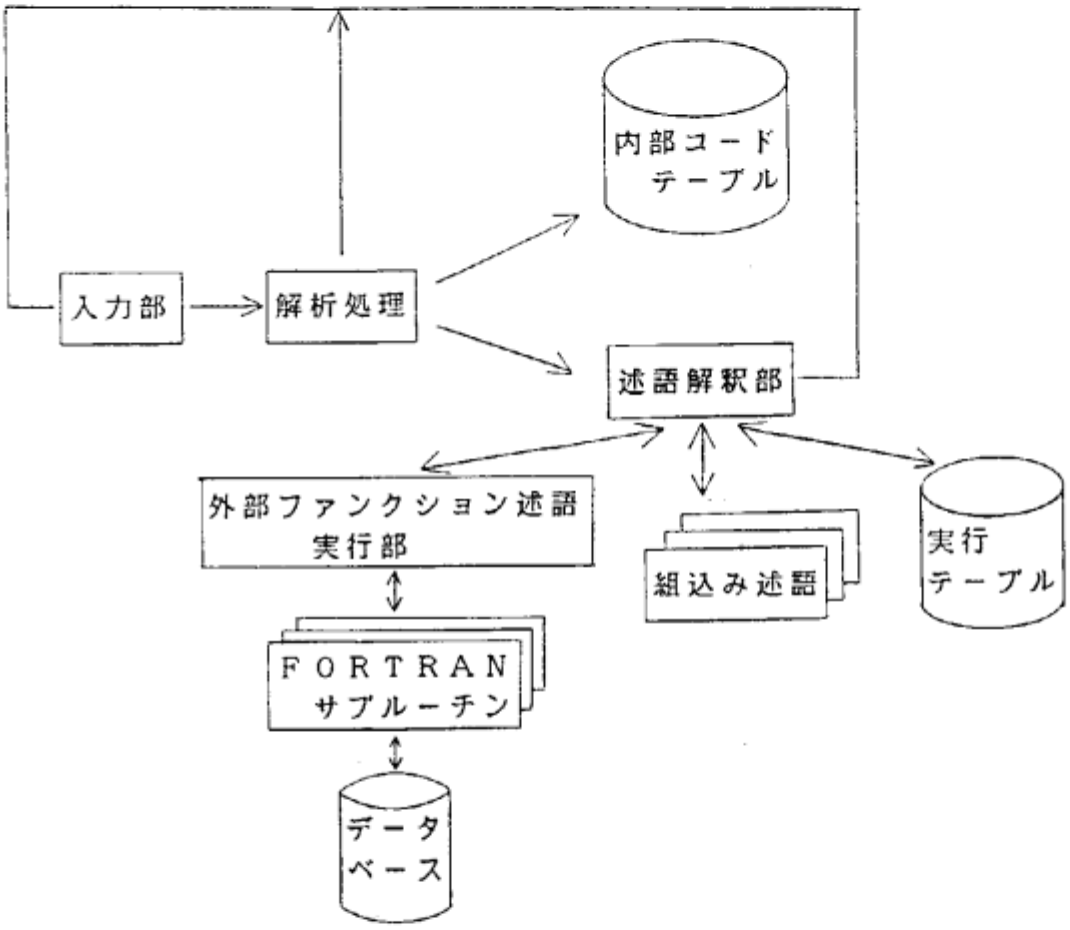


図３．　ＣＡＤＬＯＧインタープリタ構成

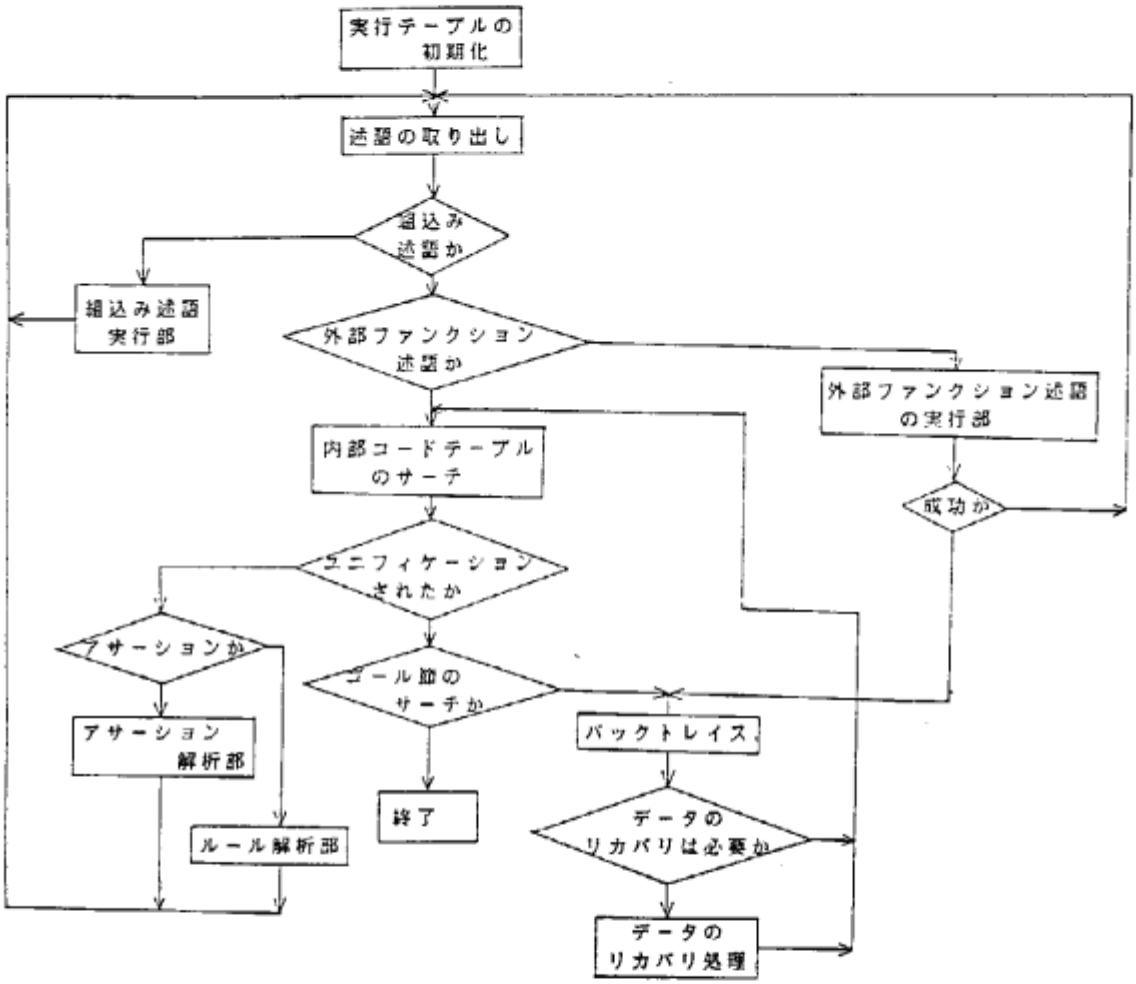


図4. 処理の流れ

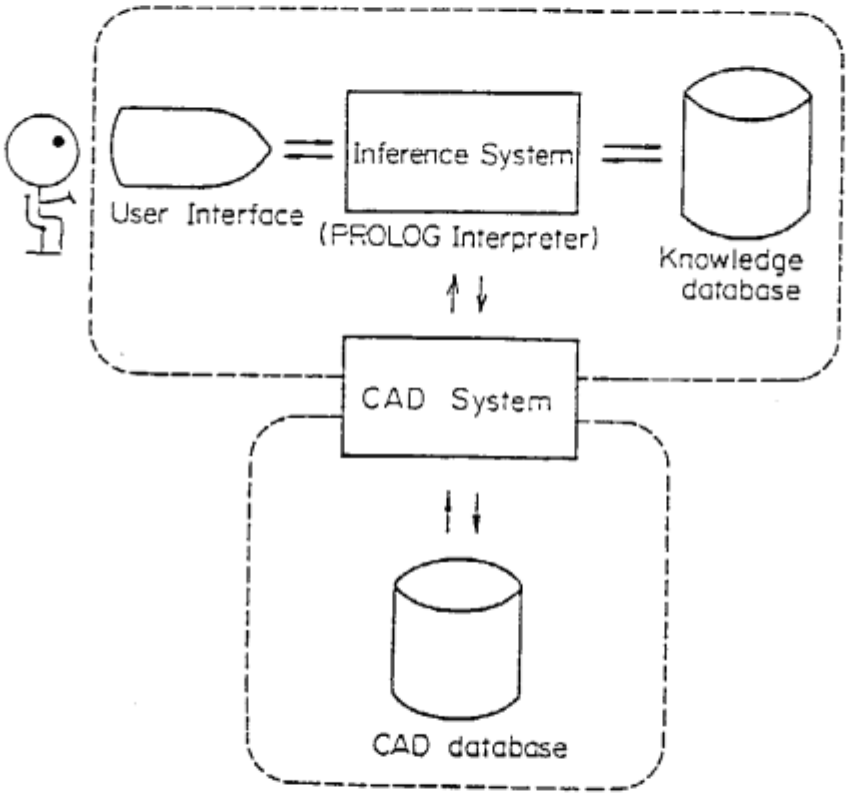


図5. 知的CADシステムの構成