論 文

定理の自動証明のためのプログラミングシステムの構成*

山崎 正人** 山本 明***

Abstract

Many strategies have been proposed for the resolution theorem provers. And there is a need for utilizing these various kind of strategies in practical theorem proving programs, because the effectiveness of each strategy depends on the charactor of a problem.

This paper presents a programming system which satisfies the need. The system is composed of five groups of LISP functions, those are (1) extracting a pair of clauses from lists of clauses, (2) handling a list of clauses, (3) executing resolution, factoring etc., (4) subsidiary functions which are used together with (2), or (3) functions, and (5) miscellaneous functions.

Using this system, users can easily implement a large number of known strategies, and can also evaluate the effectiveness of a new strategy. Another feature of this system is that it is designed in consideration of the extensibility to an interactive system.

1. はじめに

定理の自動証明に関しては，古典型な論理式の変形規則にもとづく，いわゆるnatural deductionの方法，1965年，J. A. Robinsonにより提案された導出原理(resolution principle)による方法，および，PLANNERに代表されるパターンマッチングの手法によるものがあげられる．これらのうち，導出原理によるものは，一貫した推論規則とアルゴリズムの完全性に特徴があり，従来多くの研究がなされ，プログラムの効率を高めるための多くの戦略アルゴリズムが提案され，検討されてきた．しかしながら，多くの問題に等しく有効であるような普遍的な戦略は少なく，むしろ，問題の形に応じたやや特殊な戦略を使い分け，さらに必要に応じて人間が介入する，いわゆるインタラクティブな手法をとるのが実際的であろうと考えられる．

この目的のため，ここでは，まず，すでに提案された多くの戦略アルゴリズムを容易に，かつ統一的に記述でき，また，新たな戦略アルゴリズムをも記述できるよう拡張性をもたせたプログラミングシステムを構成した．

システムは，導出原理にもとづく定理証明プログラムの戦略アルゴリズムを記述するために基本的に必要と考えられる，5群からなるLISP関数群により構成され，ユーザはこれらの関数を適当に組み合わせることによって，多くの戦略アルゴリズムを容易に，かつ，無理なく記述することができる．

このシステムは，セマンティック導出などの比較的複雑な戦略をも記述することができる高い記述能力を持ち，また，将来インタラクティブなシステムに発展させられるよう考慮されているなどの特長を持っている．

2. 戦略アルゴリズム

導出原理にもとづく定理の自動証明の過程は，与えられた充足不可能と考えられる節 (clause) の集合から，導出 (resolution) と呼ばれる一定の推論規則によって空節 (empty clause) を導くものであり，戦略アルゴリズムとは，この過程を，早く，効率的に進めるための手続きである．現在までに提案された戦略アルゴリズムの多くは，次の3種類に分類することができ

* A programming system for resolution theorem provers by Masato YAMAZAKI (Automatic Control division, Electrotechnical Laboratory) and Akira YAMAMOTO (Data Processing System division, OKI Electric Industry Co., Ltd.)

** 電子技術総合研究所制御部情報制御研究室

*** 沖電気工業（株）データ処理事業部総合技術部

る．すなわち，

（1） 証明の過程に不必要な節を消去したり生成されないようにするもの(以下，消去戦略と呼ぶ)．

（2） 導出を行う節の組を選ぶ場合に，何らかの評価関数による優先順位を決め，順位の高いものから順に導出を行おうとするもの（以下，優先戦略と呼ぶ)，および

（3） 無意味な導出が行われないよう導出の対象となる節の組を制限しようとするもの（以下，制限戦略と呼ぶ)

である．

そして，消去戦略としては，

① 他の節に論理的に包含 (subsume) される節を除去するもの，

② 恒真値をとる節を除去するもの，

③ 全ての節の集合の中に，補リテラルが存在しないリテラル (pure literal) をふくむ節を除去するもの，など，

優先戦略としては，

① 単位節（リテラル1個のみから成る節）優先戦略[1]，

② 節の論理的深さの浅いものを優先するもの[2]，

③ 節の長さ，節の深さなどで決まる評価値によって優先順位を決めるもの[2]，など，

制限戦略としては，

① 支持集合戦略 (Set of support strategy)[3]

② リニア導出 (Linear resorlution)[4]

③ セマンティック導出 (Semantic resolution)[5]

などがあげられ，各グループ内では，戦略は4.の例でも示すように，かなり統一的に記述することができる．

導出原理の基本的な推論規則は，与えられた節の集合から，それまでに取り出されたことのない節の組を取り出し，それに導出の操作をほどこすことによって新たな節を生成するものである．この推論規則と，上にのべた戦略に内在する共通の操作を抽出して，以下に示すプログラミングシステムを構成した．

3. プログラミングシステムの構成

3.1 システムの概観

本システムでは，節のリストは，主として，CLISTと呼ばれるアレイ領域上で処理される．CLISTからそれまでに取り出されたことのない節，または節の組を自動的に取り出すために，＊GPOINT，および，＊FPOINT と呼ばれるアレイ状のポインタが使用される．また，＊HIST と呼ばれるリストには，すべての節に関する履歴情報が記録されている．

システムの関数は次の5種類に大別される．

① 組み合わせ発生の関数： 導出を行うなどのためにCLIST上の指定された2つの節のリストから，それぞれ1個ずつ，それまでに取り出されたことのない節の組をとり出す．

② リスト操作のための関数： 主としてCLIST上に新たな節を加えたり，特定の節を削除したり，これを参照したりなどする．

③ 導出のための関数： 導出を行ったり，簡約形 (Factor) を求めたりなどする．

④ 補助関数： 節の長さ，論理的深さなどを調べたり，節が与えられた解釈 (Interpretation) のもとで偽値をとるか否かを調べるなど，節を評価するための関数で，通常，②，③の関数の下位関数として使用される．

⑤ 制御のための関数： システムをイニシャライズしたり，プログラムの流れを制御するための関数．

このシステムは，他の同種のシステム[6]にくらべて以下に示す特徴を持つ．

① 上記①と③の関数を分離することによって，取り出された節の組のうちで，ある条件（たとえば論理的深さの和が一定値以下であるなど）を満たすものだけ，あるいは，導出の可能性についての簡易なテスト（たとえば文献10)に示されるものなど）を通ったものだけについて導出を行うなどの戦略を記述することができる．

② 上記②と④の関数の組み合わせることによって，文献6)では別のルーチンで行っていた節の標準割り当て (standard assign) を他の記述と同一レベルで陽に表現することを可能にした．これは複数個の戦略を合わせて一つの戦略を構成する場合などに特に有効である．

③ 上記④の関数を補助関数として導入することによって，特に優先戦略を統一的に記述することが可能になった．

3.2 データ構造

3.1でのべたように，節は通常CLIST上に記憶され，この上で操作される．CLISTは大きさ8の一次元アレイで，各要素は節のリストである．節は次のリスト形式で表現されている．

(n vlist body)

ここに，n； 節に付された個有の番号

vlist； 節内にあらわれるすべての変数のリスト

body； 節の本体で $(l_1, l_2 \cdots l_n)$ と表わされる．

また，l_1，…，l_n は，節にふくまれるリテラルであり，その符号の肯定，否定に応じて，それぞれ

$$l_i=(P \quad v_1 \cdots v_m),$$

および，$(\# \quad P \quad v_1 \cdots v_m)$

ここに，P； m 引数の述語記号

v_i； 関数，変数，定数からなる項，

と表現される．

CLIST に関連して GTOP 1～8，＊GPOINT，および，＊FPOINT（それぞれ，2次元，および，1次元のアレイ構造のポインタ）があり，システムによって管理されている．これらについては，**3.3**，および，**3.4** でのべる．

節の履歴情報は＊HIST 上に記憶されている．この上で，たとえば (5 (2.1)(3.2)) と表わされた節は，番号2の節の第1番目のリテラルと，番号3の節の第2番目のリテラルに関して導出を行って得られたもので，新たに番号5が付されている．なお，最初に与えられた節のように親節が存在しないもの，および簡約形として得られた節のように一方の親節が存在しないものでは，その部分に NIL が，また，空節が生成された場合には，＊HIST 上に，

(CONTRADICTION n1 n2)

ここに n1，n2 は親節の番号

が記録される．

以上のほか，共通の変数として，関数 getpair で得られた節の組を記憶する PL，resolve，factor などによって得られた節のリストを記憶する CL，getclist で取り出された節のリストを記憶する GL，resolve，getpair 等で次の飛び先のラベル名を記憶する LB，および，補助関数と関連して使用される，VMAX，VH，VL がある．また，導出を行う場合に，変数の標準化 (standardization) のために，?X1，…，?X9，?Y1，…，?Y9，?Z1，…，?Z9 の変数が使用されている．ユーザは，これらシステムにより更新される変数によって，一時的な変数の更新にともなう煩雑な記述から解放されるが，一方では，変数名が重複しないように注意する必要がある．

3.3 組み合わせ発生の関数

ここに属するのは関数 getpair ただ1つである．

getpair [$(i \quad j)$; l_1; l_2]

CLIST の i，および j 番目の要素（それぞれ節のリスト）からそれまでに取り出されたことのない節の組を1組とり出し PL にセットする．そのような組が存在すればラベル l_1 を LB にセットし，存在しなければ PL には NOPAIR を，LB には l_2 をセットする．LB は条件による分岐を行うための変数であり，**3.7** の goev の引数として使用される．なお，引数 l_1，l_2 は，省略することができる．

この関数によって新たな節の組を探すために，CLIST の要素のすべての組み合わせに対応して，大きさ 8×8 のアレイ，＊GPOINT が使用されている．＊GPOINT の (i, j) 要素には，対応する CLIST の内容が空でないかぎり，4個のポインタ GCTOP i，j，GCURR i，j からなるリストが用意される．このリストと CLIST i，j 上の節の数を表わす変数 GTOP i，j とによって，システムは，getpair が呼ばれるたびにポインタ群を更新し，新たな節の組を取り出す．**Fig. 1** は CLIST i，j に，それぞれ，3および4個の節がふくまれている場合の例であり，getpair が n 回目に呼ばれた場合に得られる節（リスト内の節の順番で表わす）の組と，getpair の出口における各ポインタの値を表わす．

3.4 リスト操作の関数

ここに属するのは，addclist，getclist，delclist，および，clearclist の4つの関数である．

addclist [l; i]

l の内容（節のリスト）を CLIST の i 要素に後ろから追加する．ただし，第2引数が function [f_n]，の場合には，l の各要素に f_n をほどこして得られる

n	GTOP i	GCTOP i	GCURR i	PAIR	GCURR j	GCTOP j	GTOP j
0	3	1	0	…	0	1	4
1	3	1	1	(1 1)	0	1	4
2	3	2	0	(2 1)	1	1	4
3	3	2	1	(1 2)	0	2	4
4	3	2	2	(2 2)	0	2	4
5	3	3	0	(3 1)	1	2	4
6	3	3	0	(3 2)	2	2	4
7	3	3	1	(1 3)	0	3	4
8	3	3	2	(2 3)	0	3	4
9	3	3	3	(3 3)	0	3	4
10	3	3	1	(1 4)	0	4	4
11	3	3	2	(2 4)	0	4	4
12	3	3	3	(3 4)	0	4	4
13	3	3	3	NOPAIR	0	4	4

Fig. 1 An example of extracting pairs of clauses from CLIST using GETPAIR.

数に対応した所に追加する．関数の値は，更新されたCLIST の内容である．

getclist [i; n]

CLIST の i 要素の n 番目の節（n が NIL の場合には i 要素すべて）がとり出され GL にセットされるとともに関数の値となる．

delclist [i; n]

CLIST の i 要素の n 番目の節を除去する．値は除去された残りのリストである．

clearclist [i]

CLIST の i 要素，および，関連するポインタ類をクリアする．

これらの関数のうち，getclist を除く3つは特別な引数の場合を除いて CLIST の内容の変更をともなうが，このとき GTOP 1～8，＊GPOINT，および，＊FPOINT は，これらの関数によって自動的に更新され，ユーザがこれらの処理に煩はされることがないように考慮されている．**Fig. 2** に，これらポインタの更新処理のうち最も複雑な場合として，delclist にともなう GTOP i，および ＊GPOINT の更新処理の例を示す．

3.5 導出に関する関数

ここに属するのは，resolve, p-resolve, factor の基本的な関数，および，消去戦略で使用される sub-sume, tautology, pureliteral の6つの関数である．

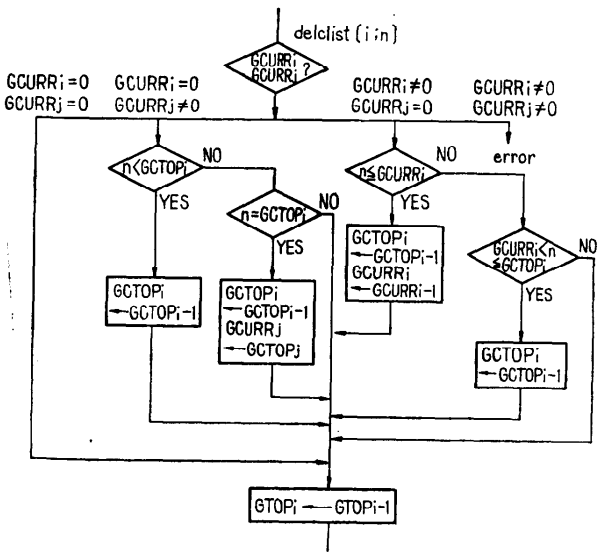

Fig. 2 The algorithm of modifying pointers when DELCLIST is called

resolve [pl; f_n; l_t; l_t; l_f]

節の組 pl から生成される導出形をすべて求め，これをリストの形で CL に，また，このとき，導出形の巾に空節をふくめば l_t を，導出形が存在すれば l_t を，導出形が存在しなければ l_f を，それぞれ LB にセットする．ここで生成された導出形には，すべて新たな節番号が付され，＊HIST 上に登録されるとともに，これらに補助関数 f_n をほどこした場合の最大値，および最小値が VH，および VL にセットされる．なお，resolve の第3引数以降は省略することができる．

p-resolve [pl; f_n; f_{n1}; f_{n2}; l_t; l_t; l_f]

resolve を特殊化した関数であり，pl の節の組それぞれにふくまれるリテラルに f_{n1}，および f_{n2} をほどこし，それぞれの評価が最も高いリテラルに関してのみ導出を行う点でのみ resolve と異なる．

この関数は，順序づけされた節 (Ordered Clauses) からクラッシュの導出形[5),7)]を求める場合に使用される基本的な関数と考えられる．

factor [cl; f_n; l_t; l_f]

節のリスト cl，または，CLIST 内の節のリスト(第1引数として CLIST 内の番号を与える）から生成されるすべての簡約形を求めて CL に，そして LB には l_t をセットし，簡約形が求まらなければ，CL には NIL を，LB には l_f をセットする．ここで生成された簡約形には，すべて新たな節番号が付され，＊HIST 上に登録されるとともに，これらに補助関数 f_n をほどこした場合の最大値，および最小値が VH，および，VL にセットされる．なお，factor の第3引数以降は省略することができる．

第1引数として CLIST [i] が指定された場合には，すでに簡約形が求められているものについて重複をさけるために，＊FPOINT [i] 上のポインタ FCURR i と GTOP i の値を参照しながら，FCURR i=GTOP i になるまで簡約形を求めつつ FCURR i を1ずつ増加する操作をくり返す．＊FPOINT は，このほか，addclist, delclist, clearclist によっても更新されるがこの場合の処理は ＊GPOINT の場合よりはるかに簡単である．

subsume [cl; i]

cl にふくまれる節で clist の i 要素の節に包含されるものがあれば，その節を除いた残り

の節のリストを値とし，同時に，これをCLにセットする．

tautology [cl]

clにふくまれる節の中に恒真値をとる節があれば，これを除いた残りの節のリストを値とし，同時にこれをCLにセットする．

pureliteral [cl]

clにふくまれる節の中にピュアリテラルをふくむ節があれば，これを除いた残りの節のリストを値とし，同時にこれをCLにセットする．

3.6 補助関数

ここに属する関数は，引数を何らかの形で評価し，その結果を上位の関数に反映させるためのもので，clength, cdepth, model, lorder, llastがある．しかし，ここにあげたものは，ごく基本的なものだけであり，また補助関数に適当なものを選ぶことによって，問題に適した戦略を構成し得る場合も多いと考えられるので，ユーザが適当な関数を追加して使用するのが望ましい．

clength [c]

節cの長さ（節を構成するリテラルの数）を値とする．このとき，値がVMAX，またはVHをこえていれば，それについて値を更新し，逆に値がVLより小さければ，VLの値をこの値で置きかえる．

VMAXは，数値を値とするような補助関数の値の最大値を記憶する変数であり，システムの初期設定(intlz)によって0にリセットされ，その後は，この種の補助関数によって更新される．VH, および VLは補助関数の上位関数のレンジ内での補助関数の最大値，最小値を記憶する変数であり，上位関数(addclist, resolve など）のレンジに入ると0にリセットされ，その後は補助関数によって更新される．

cdepth [c]

節cにふくまれるリテラルの論理的な深さの最大値，を値とすることを除いてclengthと同様である．

model [c]

ユーザが，別に関数の形で定義した解釈（モデル）のもとで節cが偽になるとき値は1，それ以外のときは値は2となる．この関数は **4.3** の例で示すようにaddclistと組み合わせて，節をモデルによって充足されないものと，それ以外のものとに分けるために使用される．なお，VMAX, VH, VLの更新は行わない．

lorder [c]

別にPORDERに設定されたリテラルの優先順位にしたがって，節cの中でも最も順位の高いリテラルをとり出し値とする．

llast [c]

節cを順序づけされた節と見て，最後のリテラルをとり出し値とする．

3.7 制御のための関数

ここには，イニシャライズ，条件文の記述，変数への値の設定などのための多くの関数があるが，それらのうち，本システムに固有なもの，および，LISP 1.5の関数と仕様の異なるものについてのみ以下にのべる．

intlz [ ]

システムのデータエリア，ポインタ類，その他，変数などに定められた初期値を設定し，システムを初期状態にする．

p-reset [p; i]

ポインタ p (*GPOINT, または *FPOINT) の i 要素をリセットする．

add [n; i]

変数 i の値に n を加え，これを i にセットする．

sub [n; i]

変数 i の値から n を引き。これを i にセットする．

cond [[l_{11}; l_{12}, …; l_{1n}]; …[l_{m1}; l_{m2}; …, l_{mn_m}]]

l_{11}, …, l_{m1} を順に評価し，その値が T となる最初の l_{i1} について l_{i2}, l_{i3}, …, l_{in_i} を順に評価し l_{in_i} を評価した値を関数の値とする．

goev [lb]

lb を評価し，これで示されるラベルの所へ飛ぶ．

gep [a; b]

a の値が b の値より大きいか等しいとき T，そうでなければ F を値とする．

lep [a; b]

a の値が b の値より小さいか等しいとき T，そうでなければ F を値とする．

4. プログラミング例

ここでは，2.でのべた戦略のうち，優先戦略と制限戦略の代表例についてプログラムの構成例を示し，若干の考察を加える．

4.1 単位節優先戦略

単位節優先戦略は，導出を行うときの親節として，長さの短い節を優先して選ぼうとするもので，そのアルゴリズムは文献 1)，に示されている．**Fig. 3** (次頁参照) にこれを本システム上で記述した例を示すが，アルゴリズムとプログラムの対応に無理がなく，

```
(SEMANTIC-RESOLUTION (LAMBDA (L PORDER) (PROG (M N J I)
      (INTLZ)
 S1   (SETQ J 1)
 S2   (ADDCLIST L (FUNCTION MODEL))
      (SETQ M (GETCLIST 1 NIL))
      (ADDCLIST M 3)
      (ADDCLIST (FACTOR 3 NIL) 3)
      (SETQ N (GETCLIST 2 NIL))
 S3   (SETQ I 0)
 S4   (COND ((MEMBER (QUOTE SUC)(GETCLIST 1 NIL))(GO SUC)))
 S5   (COND ((NULL (GETCLIST 2 NIL))(GO S8) ))
 S6   (CLEARCLIST 4)
      (ADDCLIST 2 4)
      (CLEARCLIST 1) (CLEARCLIST 2)
 P1   (GETPAIR (3 4) P2 S7)
      (GOEV LB)
 P2   (P-RESOLVE PL NIL (FUNCTION LORDER)(FUNCTION LLAST)
                                    P2S P2S P1)
      (GOEV LB)
 P2S  (ADDCLIST CL (FUNCTION MODEL))
      (ADDCLIST 1 5)
      (GO P1)
 S7   (ADD 1 J)
      (GO S4)
 S8   (ADDCLIST 5 6)
      (ADDCLIST 6 7)
 S9   (ADD 1 J)
 S10  (ADDCLIST (FACTOR 6 NIL) 6)
      (ADDCLIST N 8)
      (CLEARCLIST 1)(CLEARCLIST 2)
 P3   (GETPAIR (6 8) P4 S11)
      (GOEV LB)
 P4   (P-RESOLVE PL NIL (FUNCTION LORDER) NIL P4S P4S S11)
      (GOEV LB)
 P4S  (ADDCLIST CL (FUNCTION MODEL))
 S11  (GO S3)
 SUC  (PRINT (QUOTE SUCCESS))
      (RETURN (PRINT *HIST))
)))
```

Fig. 3 An example of the unit preference strategy

かつ容易に記述されていることがわかる.

また, 他の多くの優先戦略も評価関数（この例では補助関数 clength）を中心に修飾することによって, かなり統一的に記述できることがわかる.

4.2 支持集合戦略

与えられた節の集合 S について, S-T は充足可能であるような S の部分集合 T を S に関する支持集合と呼ぶ. 支持集合戦略とは, 導出を行う場合の親節のうち少なくとも一方を T, または T を祖先として持つ節のなかから選ぶというもので, S-T の節同士に関して導出が行われないように制限した制限戦略の一種である. この戦略のプログラムは, **Fig. 4** のように簡潔に記述することができる.

4.3 セマンティック導出

支持集合戦略は, 支持集合が予め知られていて, これを前提にして構成された戦略であったのに対して, セマンティック導出は, ユーザが適当な解釈(モデル)を与え, この解釈のもとでの充足可能性にもとづいて節の集合を分割し, 同じ分割に属する節同士に関する導出を制限しようとするものであり, さらに, 導出によって得られた節も同様の基準によって, どちらかの集合に振り分け, また, 冗長な導出形の生成をさけるためにクラッシュの導出形のみを求める, など制限戦略として徹底した, かつ複雑な戦略と考えられる. **Fig. 5**（次頁参照）にセマンティック導出を記述した例を示す. ここでは, **Fig. 6**（次頁参照）に示すように, 解釈は実行時に関数の形で定義し, 補助関数 model によって, 節の評価を行っている. なお, クラッシュの導出形を求めるためのアルゴリズムは Lee, chang により与えられたもの[8]を使用した.

```
(SET-OF-SUPPORT (LAMBDA (ST SS) (PROG (J)
      (INTLZ)
      (ADDCLIST ST 1)
      (ADDCLIST (FACTOR 1 NIL) 1)
      (ADDCLIST SS 2)
      (ADDCLIST (FACTOR 2 NIL) 2)
 P0   (SETQ J 0)
 P1   (COND ((EQ J 2) (GO FAIL)) )
      (ADD 1 J)
 P2   (GETPAIR (1 J) P2S P1 )
      (GOEV LB)
 P2S  (RESOLVE PL NIL SUC P21 P2 )
      (GOEV LB)
 P21  (ADDCLIST CL 1)
      (ADDCLIST (FACTOR 1 NIL) 1)
      (GO P0)
 SUC  (PRINT (QUOTE SUCCESS) )
      (RETURN (PRINT *HIST) )
 FAIL (RETURN (PRINT (QUOTE FAIL)) )
)))
```

Fig. 4 An example of the set of support strategy

5. おわりに

戦略アルゴリズムを基本的な操作に分解し, これらを比較的少ない関数群によって実現することによって, 従来提案されてきた多くの戦略アルゴリズムを, 容易に, かつ, 統一的に記述することができるプログラミングシステムについてのべた.

本システムの特徴は, セマンティック導出などの比較的複雑な戦略をも記述することができる高い記述能力にあるが, これは他の同種システムには見られない以下の特徴によるところが大きいと考えられる.

① 組み合わせ発生の関数と導出を行う関数を分離することによって, その間での条件テストなどを

```
(UNIT-PREFERENCE (LAMBDA (S) (PROG (L H J)
     (INTLZ)
     (SETQ L S)
     (ADDCLIST L (FUNCTION CLENGTH) )
     (SETQ J 1)
P1   (GETPAIR (1 J) P1S P1F)
     (GOEV LB)
P1F  (COND ((GREATERP J VMAX) (GO NU) )
           (T (ADD 1 J)(GO P1) )       )
P1S  (RESOLVE PL (FUNCTION CLENGTH) SUC P11 P1)
     (GOEV LB)
P11  (ADDCLIST CL (FUNCTION CLENGTH) )
     (SETQ J VL)
     (GO P1)
NU   (SETQ J 2)
P2   (FACTOR J NIL P2S P2F)
     (GOEV LB)
P2F  (COND ((EQ J 2) (ADD 1 J)(GO P2) )
           (T (GO NU1) )               )
P2S  (ADDCLIST CL (FUNCTION CLENGTH) )
     (SETQ J VL)
     (GO P1)
NU1  (SETQ H 2)
     (SUB 1 J)
P3   (GETPAIR (H J) P3S P4)
     (GOEV LB)
P3S  (RESOLVE PL (FUNCTION CLENGTH) SUC P31 P3)
     (GOEV LB)
P31  (ADDCLIST CL (FUNCTION CLENGTH) )
     (SETQ J VL)
     (GO P1)
P4   (COND ((EQ H VMAX) (GO FAIL) )
           ((GEP (PLUS 1 H) J) (ADD H J)(GO P2))
           (T (ADD 1 H)(SUB 1 J)(GO P3))       )
SUC  (PRINT (QUOTE SUCCESS))
     (RETURN (PRINT *HIST))
FAIL (RETURN (PRINT (QUOTE FAIL)))
)))
```

Fig. 5 An example of the semantic resolution

可能にした.

② データ領域への節の標準割り当てに関する記述を陽に表現することを可能にした.

③ 補助関数を導入することによって特に優先戦略を統一的に記述できるようにした.

なお, ここにのべたシステムは, 先に筆者らが構成したもの[9]を, その後の検討にもとづき改良したものである.

最後に, 終始御指導, 御検討頂いた, 電総研長田正情報制御研究室室長, 明治大学向殿政男助教授, また御討論いただいた電総研ロボットグループ各位に感謝の意を表します.

参 考 文 献

1) L. Wos, D. Carson, and G. Robinson, : "The unit preference stratogy in theorem proving", Proc. FJCC, (1964).

2) R. Reboh, B. Raphael, R. A. Yates, R. E. Kling, and C. Velarde, : "Study of automatic theorem-proving programs", SRI tech. note 75, (Nov. 1972).

3) L. Wos, G. A. Robinson, and D. F. Carson, "Efficiency and completeness of the set of support strategy in theorem proving", JACM, Vol. 12, No. 4, pp. 536～541, (Oct. 1965).

4) D. W. Loveland, "A linear format for resolution", Proc. IRIA Symp. Automatic demonstration, Springer-Verlag, N. Y., pp. 147～162.

5) J. R. Slagle, : "Automatic theorem proving with renamable and semantic resolution", JACM Vol. 14, No. 4, pp. 687～697, (Oct. 1967).

6) L. Henschen, R. Overbeek, and L. Wos, : "A theorem proving language for experimentation", Comm. ACM, Vol. 17, No. 6, (June. 1974).

7) J. A. Robinson, : "A review of automatic theorem-proving", Proc. Symp. in App. Math., Amer. Math. Soc., Providence, R. I., (1967).

8) C. L. Chang, and R. C. T. Lee, : "Symbolic logic and mechanical theorem proving", Academic Press, (1973).

9) 山本, 山崎: "Theorem-prover のための実験システムについて", 信学会会国大会予稿, No. 1263. 昭 50.

10) R. B. Stillman, : "The concept of weak substitution in Theoremproving", JACM, Vol. 20, No. 4, (Oct. 1973).

```
DEFINE ((
 (P (LAMBDA (X Y Z)
      (COND ((EQ X (QUOTE A))(EQ Y Z))
            ((EQ Y (QUOTE A))(EQ X Z))
            (T (EQ Z (QUOTE A)) )    ))
 (G (LAMBDA ( X Y)
      (COND ((EQ X Y)(QUOTE A))
            (T (QUOTE B))     ) ))
 (H (LAMBDA (X Y)
      (COND ((EQ X Y)(QUOTE A))
            (T (QUOTE B))     ) ))
 (K (LAMBDA (X) X      ))
))

SEMANTIC-RESOLUTION (
  (   (1 (X Y) ((P (G X Y) X Y)))
      (2 (X Y) ((P X (H X Y) Y)))
      (3 (U V W X Y Z) ((# P U Z W)(# P X V W)
                        (# P X Z V)(P U Z W) ))
      (4 (X) ((# P (K X) X (K X) )))          )
  (   P   )
                )
```

Fig. 6 An example of defining model and executing the semantic resolution

(昭和50年7月1日受付)
(昭和50年10月11日再受付)

訂　正

5月号掲載の山崎正人，山本　明両君の論文「定理の自動証明のためのプログラミングシステムの構成」の中で415ページの Fig. 3 と，416 ページの Fig. 5 の図版が入れ違いでした．お詫びして訂正します．