Vol. 27 No. 1 情報処理学会論文誌 Jan. 1986

定理証明プログラムにおける内部構造の一実現法†

山口高平†† 打浪清一†††
手塚慶一††† 角所収††

導出原理に基づく定理証明プログラムの内部構造の実現法には，導出の実行時に節レベルでまったく新しい構造を作り出す Structure Generating (SG) 方式と導出の実行時に束縛だけを作成する Structure Sharing (SS) 方式があるが，現在，SS 方式が効率の良さから定理証明システムにおいて多く採用されており，また，Prolog を代表とする論理プログラミングシステムにおいても，SS 方式は Structure Copying 方式とともに代表的な内部構造の実現法になっている．しかしながら，束縛を頻繁に参照するプログラムにおいては，SS 方式では，効率が悪くなる危険性がある．SS 方式のこの種の欠点は，節を間接的に表現することに起因していると考えられる．そこで本論文では，従来の SG 方式に改良を加え，必要最小限の情報のみを生成する RSG 方式を提案し，効率比較実験により，本方式は，SS 方式と比較してほぼ同程度の効率が得られることを示すとともに，上記の欠点が克服されることを示す．

1. まえがき

定理証明プログラムは，一階述語論理の一標準形である節を用いて表現されるが，一階述語論理は，数学の大部分と日常会話の多くの叙述を表現できる論理体系であるため，プログラム化できる問題領域は広いと言える．また，定理証明プログラムの実行理論は，一階述語論理体系における導出原理に基づく場合が多いが，これは，導出原理が完全性および単純性の両性質を兼ね備える強力な推論規則となっているためである．

しかしながら，与えられた節集合に対して，導出原理をすべての節ペアに適用することは，証明したい定理に関係のない内容まで推論してしまい，実行効率・記憶効率ともに悪くなる．そこで，導出原理の適用範囲（探索空間）を限定し，効率改善を図る手続きとして，制限付導出法および証明戦略が数多く考えられ，そのなかでも，節集合の特別なクラスの一つである Horn 節集合に対して完全性を保持する制限付導出法[1]に興味が集まった．

以上の背景から，導出原理に基づく定理証明システムの作成が始まったわけであるが，項・リテラル・節等のデータ表現（内部構造）が重要な問題となった．

この内部構造の実現方式としては，大きく分けて SG 方式と SS 方式[2]がある．SG 方式（リスト構造が代表的である）というのは，導出の実行時に節レベルでまったく新しい構造を作り出す方式であり，導出時に置換・コピー等の操作により，構造を頻繁にたどる必要があるため実行効率が悪く，また，節記録が導出ごとに新しく生成され，その節記録がリテラル数等に直接比例して大きくなるため記憶効率も悪くなる．

一方，SS 方式というのは，導出の実行時に束縛（変数への代入情報）だけを作成する方式であり，実行時に構造をたどる必要がないため，実行効率が良く，また，束縛を利用して節をリテラル数とは無関係にデータ圧縮して表現できるため，SG 方式に比べて大幅に記憶効率が改善される．以上の理由から，現在，多くの定理証明システムおよび論理プログラミングシステム[3]に SS 方式が採用されている．

しかしながら，SS 方式では実行時に束縛をたどる必要があるため，束縛を頻繁にたどるプログラムにおいては，実行効率が悪くなる危険性がある．

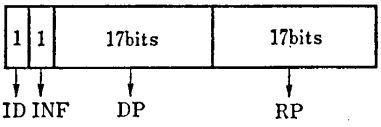
そこで本論文では，従来の SG 方式に改良を加え必要最小限の情報のみを生成する RSG 方式を提案し，そのインプリメンテイションについて述べる．また，効率比較実験により，本方式は，SS 方式と比較してほぼ同程度の効率が得られることを示すとともに，上記の欠点が克服されることを示す．

2. RSG 方式の基本概念

まず，筆者らが以前に開発した定理証明システム SENRI[4] のセル構造（リスト構造）を図 1 に示す．この

† Implementation of Internal Structure in Theorem Proving Programs by TAKAHIRA YAMAGUCHI (Institute of Scientific and Industrial Research, Osaka University), SEIICHI UCHINAMI, YOSHIKAZU TEZUKA (Faculty of Engineering, Osaka University) and OSAMU KAKUSHO (Institute of Scientific and Industrial Research, Osaka University).

†† 大阪大学産業科学研究所
††† 大阪大学工学部通信工学科



ID部 ―→ セルの識別情報をもつ.

セルの種類	値
リストセル	0
データセル	1

INF部 ―→ データセルのとき，データの型情報をもつ.

データの型	値
定　数	0
変　数	1
関数記号	0
述語記号	0
補の述語記号	1

注) 同値のデータの型の区別は，セルの位置情報により行う.

DP部 ―→ データセルのときは，文字データを指すためのポインタをもち，リストセルのときは，下連結のためのポインタをもつ.

RP部 ―→ 右連結のためのポインタをもつ.

図 1 SENRI のセル構造
Fig. 1 Cell structure of SENRI.

リスト構造を基にして，定理証明問題 (TPU EXAMPLE 1-9)[5]の証明過程において発生するリテラルリスト数とリテラルリスト構造数を調査した結果が**表 1** である.

表 1 より，定理の証明過程において発生するリテラルリスト構造数は，リテラルリスト数に比べて非常に少なく，構造的に同等なリテラルリストが，数多く発生していることがわかる.

(定義 1) 同型

点集合を P および辺集合を E とし，グラフ G を $G=(P, E)$ で表現する．二つのグラフ $G_1=(P_1, E_1)$ と $G_2=(P_2, E_2)$ が同型であるとは，すべての $x, y \in P_1$ に対し，$(x, y) \in E_1$ であるときに限り，$(\phi(x), \phi(y)) \in E_2$ となるような 1 対 1 の写像関数 $\phi: P_1 \to P_2$ が存在することである. (定義終)

定義 1 の用語を使えば，従来の SG 方式では，冗長な情報として同型のリテラルリスト構造を生成していたと言える.

次に，リテラルリスト構造の表現法について考察する.

(定理 1) 2 分木構造の同型性

2 分木 T と T′ の節点をおのおの行きがけの順に並べるとき，$u_1 \cdot u_2 \cdots u_n$ および $u_1' \cdot u_2' \cdots u_n'$ となるものとする．次に，u を任意の節点とするとき，

$$l(u)=\begin{cases}1: u \text{ の左の部分木が空でないとき}\\ 0: u \text{ の左の部分木が空であるとき}\end{cases}$$

$$r(u)=\begin{cases}1: u \text{ の右の部分木が空でないとき}\\ 0: u \text{ の右の部分木が空であるとき}\end{cases}$$

と書くことにする.

T と T′ が同型であるのは，以下の条件が満足される場合であり，またその場合に限られる.

$n=n'$

$l(uj)=l(uj')$, $r(uj)=r(uj')$

(for $1 \leqq j \leqq n$)

(証明は，文献 6) の pp. 110, 111 参照)

(定理終)

定理 1 に従って，**図 2** のようにリテラルリスト構造に相当する 2 分木の構造値 (リテラルリスト構造値と呼ぶ) を作成すれば，同型の判定操作をビット列で高速に実行できる.

以上の考察より，定理証明プログラムの各節がもつ情報として，基本的には，リテラルリスト構造値とそ

表 1 証明過程で発生するリテラルリスト数とリテラルリスト構造数
Table 1 The number of literal lists and literal list structures in proof process.

問題＼項目	リテラルリストの数	リテラルリスト構造の数
TPU-1	39	16
TPU-2	244	1
TPU-3	102	9
TPU-4	106	6
TPU-5	33	5
TPU-6	128	11
TPU-7	177	10
TPU-8	79	9
TPU-9	43	11

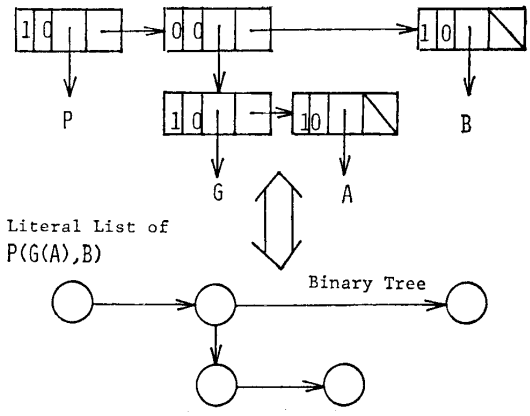


図 2 $P(G(A), B)$ のリテラルリスト構造値
Fig. 2 Literal list structure value of $P(G(A), B)$.

```
< CLAUSE RECORD >::=< (< STRUCTURE VALUE SEQUENCE >) , (< DATA VALUES OF STRUCTURE VALUE SEQUENCE >) >
< STRUCTURE VALUE SEQUENCE >::=< STRUCTURE VALUE >|< STRUCTURE VALUE >,< STRUCTURE VALUE SEQUENCE >
< STRUCTURE VALUE >::= S< UNSIGNED INTEGER >
<DATA VALUES OF STRUCTURE VALUE SEQUENCE >::= (< DATA VALUE SEQUENCE >)|
                                              (< DATA VALUE SEQUENCE >),< DATA VALUES OF STRUCTURE VALUE SEQUENCE >
< DATA VALUE SEQUENCE >::=< DATA VALUE >|< DATA VALUE >,< DATA VALUE SEQUENCE >
< DATA VALUE >::=< INFORMATION VALUE >< IDENTIFIER >
< INFORMATION VALUE >::= 0|1
< IDENTIFIER >::=< ALPHABET >|< ALPHABET >< LETTER SEQUENCE >
< LETTER SEQUENCE >::=< ALPHABET >|< ALPHABET >< LETTER SEQUENCE >|< DIGIT >|< DIGIT >< LETTER SEQUENCE >
< UNSIGNED INTEGER >::=< DIGIT >|< DIGIT >< DIGIT SEQUENCE >
< DIGIT SEQUENCE >::=< DIGIT >|< DIGIT >< DIGIT SEQUENCE >
< ALPHABET >::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
< DIGIT >::= 0|1|2|3|4|5|6|7|8|9
```

図 3 RSG 方式による節記録の表記法
Fig. 3 Notation of clause record by RSG.

れらがもつデータ値のみを考え，これらの情報を高速に処理する方式を RSG (Refined Structure Generating) 方式と名付ける．したがって，RSG 方式による節記録の表示法（外部表現）は，リテラルリスト構造値とそれらがもつデータ値のタップル表現となり，**図 3** にその定義を示すとともに，**図 4** における各節の外部表現を**図 5** に示す．

3. RSG 方式のインプリメンテイション

本章では，リテラルリスト構造値とそれらがもつデータ値のタップル表現で表された節記録を高速に処理する手順について述べる．

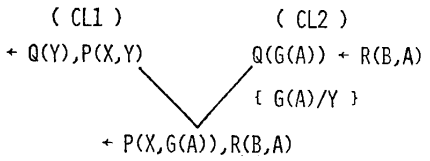


図 4 定理証明プログラムの実行例
Fig. 4 Execution example of theorem proving programs.

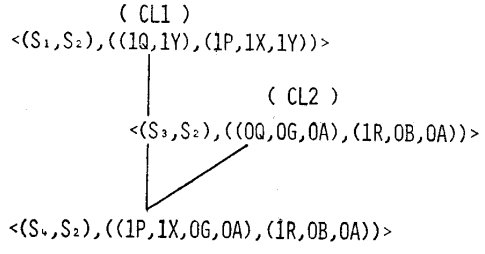


図 5 RSG 方式による節記録の表記例
Fig. 5 Notation example of clause record by RSG.

3.1 節の内部表現

RSG 方式による節記録は，基本的には，リテラルリスト構造値とそれらがもつデータ値のタップル表現であるが，その他の付属情報として，両親節の順位と変数を標準化するための最大指数および出力の制御情報を蓄えている．**図 6** に，節の具体例と RSG 方式によるその外部表現および内部表現を示す．ただし，LS はリテラルリスト構造値を蓄え，LCNT には，その構造値のビット数と保有データ数が蓄えられている．

3.2 単一化計算と導出操作

RSG 方式による単一化計算では，リテラルリスト構造値を利用して，単一化計算を実行するためのペアを高速に抽出することが可能であり，**図 7** にそのアルゴリズムを示す．抽出後，変数にある値が代入されるときは，その度に置換を行うと実行効率が悪くなるので，一時的な束縛を作成し，導出形生成時にこの束縛を利用して，初めて置換を行う．変数に関数が代入されるときは，新しいリテラルリスト構造が発生する可能性があるので，束縛に関数構造値を付加して，導出形生成時に利用する．また，導出形作成時には，親節の被導出リテラルを除いたリテラルを連結し，一時的な束縛を参照しながら，リテラルリスト構造値あるいはデータ値を置き換え，導出形の内部表現を生成する．

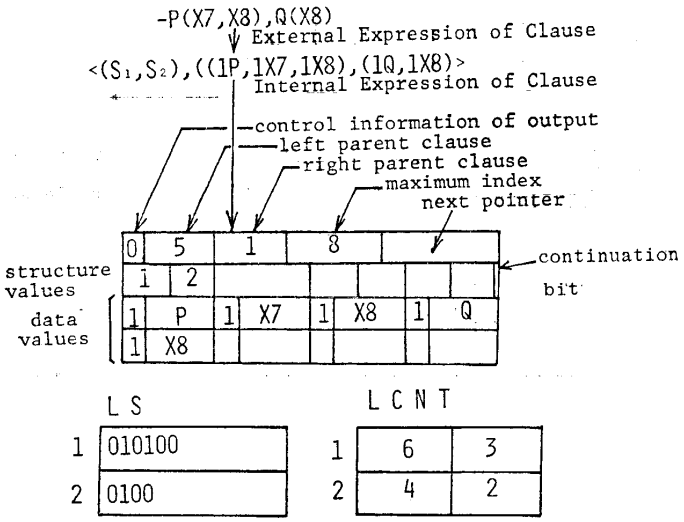


図 6 RSG 方式による節の外部表現と内部表現
Fig. 6 External expression and internal expression of a clause by RSG.

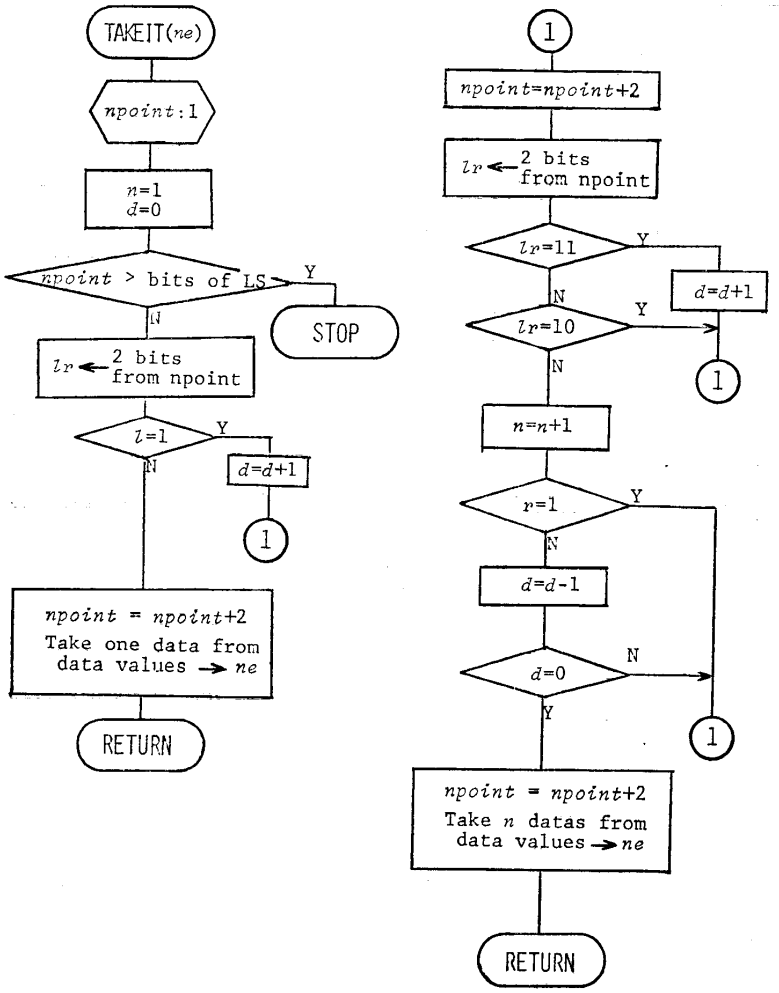


図 7 単一化計算を実行するためのペア抽出アルゴリズム
Fig. 7 Extraction algorithm of a pair to unify.

3.3 RSG 方式による導出過程の具体例

以下，具体例を通して導出過程を説明する．まず，**図 8** の導出過程を考えると，親節の内部表現は，**図 9** のようになる．各親節の第 1 リテラルに注目すると，parent clause (6)の第 1 リテラルでは，リテラルリスト構造値として LS(1)＝010100 を保持しており，LCNT(1,2)＝3 より，3 個のデータ値を保有していることがわかる．また，parent clause (2)の第 1 リテラルでは，リテラルリスト構造値として，LS(3)＝0101100100 を保持しており，LCNT(3,2)＝4 より，4 個のデータ値を保有していることがわかる．

次に，図 7 に示すペア抽出アルゴリズムに従って，**図 10** のように単一化計算を実行するためのペアが順次取り出され，単一化計算時に**図 11** に示すような関数構造値を付加した一時的な束縛が形成される．

最後に，一時的な束縛を参照しながら，親節から被導出リテラルを除いたタップル情報を基にして，**図 12** に示すような導出形の内部表現を生成する．

4. 効率比較実験による RSG 方式の評価

RSG 方式による単一化計算は，構造をたどることなく，また束縛を参照することなく，タップル値を直接操作しながら実行されるため，高速化が期待される．さらに，節記録は，直接的な表現であり，同型のリテラルリスト構造を蓄えないため，データ圧縮度は高いと考えられる．以上のことを定量的に評価するために，実行効率と記憶効率に関して，本方式と SG 方式と SS 方式との効率比較実験を行った．まず，実験システムについて述べ，次に，実験結果およびその検討について述べる．

4.1 実験システムの概要

まず，SG 方式の実験システムは，定理証明システム SENRI[4]を用いた．このシステムでは，導出の実行時に原始的な変数名変換操作を行ったり，変

数にある値が代入される度に新しい構造を生成しているため，頻繁に構造をたどる操作が行われる．ただし，本システムでは，リストのコピー操作が頻繁に発生し，効率に大きな影響を与えている．

次に，SS 方式の実験システムは，Boyer, R. S.[2]らによって提案されたものを参考にして作成しており，節記録は**図 13** に示すようなセル構造で表現される．

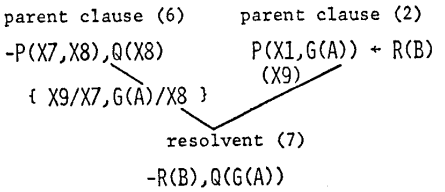


図 8 導出過程の一例
Fig. 8 Example of resolution process.

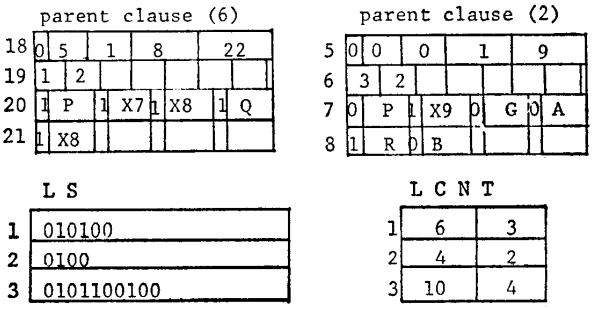


図 9 各親節の内部構造
Fig. 9 Internal structure of each parent clause.

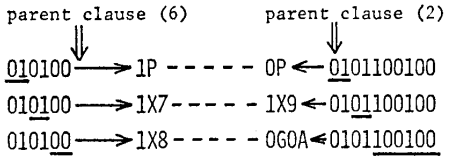


図 10 順次取り出された単一化計算を実行するためのペア
Fig. 10 Pair extracted sequentially for unification.

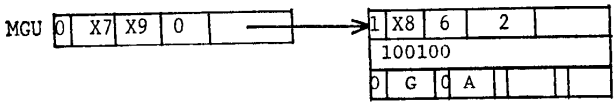


図 11 生成された一時的な束縛
Fig. 11 Generated temporary binding.

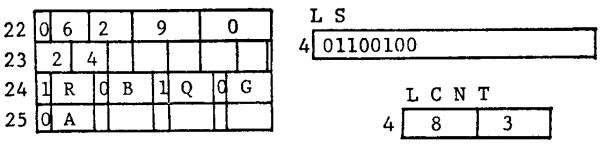
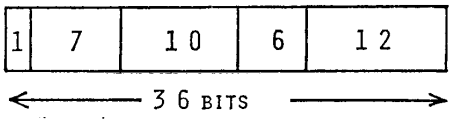


図 12 生成された導出形の内部表現
Fig. 12 Generated internal expression of resolvent.



1ST FIELD : Whether this clause is a factor or not
2ND FIELD : The order of the (right) parent clause
3RD FIELD : The maximum index
4TH FIELD : The number of literals
5TH FIELD : The pointer to list structure or binding environments

図 13 SS 方式を内部構造とする実験システムのセル構造
Fig. 13 Cell structure of experimental system with SS.

また，束縛環境は，連想リストに似た形式で蓄えられており，逐次的に探索される．このシステムでは，導出の実行時に束縛を参照しながら単一化計算を実行し，新しい束縛の生成および導出形の記録生成が主な操作となる．

最後に，RSG 方式の実験システムは，3 章で述べた内容に沿って作成されており，導出の実行時には，単一化計算を実行するためのペア抽出と一時的な束縛の生成および導出形の記録生成が主な操作となる．

以上の三つの実験システムにおいては，制御構造，および記憶領域の管理法等，他の環境はまったく同じである．また，実験用問題としては，SORTING, PARSING, 4-QUEEN を用いた．

4.2 実験結果と検討

表 2 に実行効率の比較，**表 3** に生成された導出形に関する記憶領域の比較を示す．

まず，実行効率については，SG 方式は，導出の実行時に構造を頻繁にたどったり，置換操作を繰り返し行うため最悪である．SS 方式は，導出の実行時に束縛を参照・生成するわけだが，この実行コストは，上記の実行コストに比べてかなり小さいため，実行効率は，大幅に改善されている．最後に，RSG 方式は，ペアの抽出および導出形生成コストが，小さく抑えられているため，SS 方式よりさらに効率は改善されている．本実験では，RSG 方式は平均して，SG 方式よりも 13.0 倍効率が改善され，SS 方式と比較しても 1.48 倍効率が改善されている．

次に，記憶効率については，SG 方式は，節記憶がリテラル等に直接比例して大きくなるため最悪である．SS 方式は，束縛を蓄える必要があるが，1 セルで一つの節（導出形）を表現できるため，大幅にデータ圧縮されている（三つの問題を実行するのに必要な

表 2 各内部構造の実行効率
Table 2 Execution efficiency of each internal structure.

(msec)

Problem \ Internal Structure	Structure Generating	Structure Sharing	Refined Structure Generating
Sorting	1677	246	134
Parsing	6750	606	458
4-Queen	18892	2054	1615

Machine: ACOS system 1000 model 40

表 3 各内部構造の記憶効率
Table 3 Memory efficiency of each internal structure.

(words)

Problem \ Internal Structure	Structure Generating	Structure Sharing	Refined Structure Generating
Sorting	567	187	155
Parsing	930	216	243
4-Queen	1241	374	399

記憶領域で比較すれば，SG 方式より 3.52 倍改善されている)．RSG 方式は，束縛および同型のリテラルリストを蓄える必要がなく，またリテラルリストが1セルで表現できるため，SS 方式とほぼ同程度になっている（上記と同様な方法で比較すれば，SG 方式より 3.44 倍改善されている)．

最後に，RSG 方式の他の特徴としては，定理証明プログラムの実行中に，無限ループを回避するのに有効な戦略である「二つの節が等しいかどうかを判定する処理」(subsumed clause[5]) の判定の一つと考えられる）が，リテラルリスト構造値とデータ値を表すブロック領域の同等性として扱えるため，高速に処理できることが挙げられる．

5. むすび

本論文で提案した RSG 方式は，他の定理証明プログラムの内部構造の実現法と比較して，以下の特徴をもっている．

（1） 導出の実行時に，単一化計算を行うペアを高速に抽出しながら，単一化操作を実行し，一時的な束縛を作成することにより，置換操作を一度だけで済ませているため，実行効率は向上する．また，束縛を頻繁に参照する定理証明プログラムにおいても，本方式は束縛を参照しないため実行効率は低下しないと考えられる．

（2） 節記録は，束縛環境を蓄える必要がなく，また同型のリテラルリスト構造を蓄える必要がないため，記憶効率は改善される．

（3） 二つの節の同等性の判定が高速に処理できる．

以上のことから，RSG 方式は，定理証明プログラムにおける新しい置換型の内部構造の実現法として位置付けられると考えられる．

謝辞 実験において多大なご協力をいただいた本学卒業生石川淳士君に感謝します．

参考文献

1) Kuehner, D.: Some Special Purpose Resolution Systems, *Machine Intelligence*, Vol. 7, pp. 117-128 (1972).
2) Boyer, R. S. and Moore, J. S.: The Sharing of Structure in Theorem Proving Programs, *Machine Intelligence*, Vol. 7, pp. 101-116 (1972).
3) Warren, D. H. D. and Pereira, L. M.: Prolog—The Language and Its Implementation Compared with Lisp, ACM, Proc. of the Symposium on Artificial Intelligence and Programming Language, pp. 109-115 (1977).
4) 山口，西岡，打浪，手塚：定理証明システム SENRI の構成，情報処理学会論文誌，Vol. 25, No. 1, pp. 46-58 (1984).
5) Chang, C. L. and Lee, R. C. T.: *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York (1973).
6) クヌース（米田信夫，筧 捷彦訳）：基本算法/情報構造，サイエンス社，東京 (1978).
7) Bruynooghe, M.: The Memory Management of PROLOG Implementations, Proc. of Logic Programming Workshop, pp. 12-20 (1980).
8) ファルツ，J. L.（間野浩太郎監訳）：電子計算機データ構造論，マグロウヒル好学社，東京 (1980).
9) 佐藤泰介：導出原理による定理証明，情報処理，Vol. 22, No. 11, pp. 1024-1036 (1981).

(昭和 59 年 1 月 23 日受付)
(昭和 60 年 7 月 18 日採録)