MGTP: 並行論理型言語 KL1 によるモデル生成型定理証明系

長谷川 隆三[†] 藤田 博[††]

一階述語論理のための定理証明系を，モデル生成法に基づき並行論理型言語 KL1 で効率良く実現する方法について述べる．モデル生成法においては，値域限定という条件を満たす節集合を対象とした場合，出現検査付き単一化が不要であり，照合操作で十分となる．その結果，入力節中の変数を直接 KL1 変数で表現することができ，入力節に対する単一化を KL1 節の頭部単一化として実行できるなど，KL1 言語処理系を活用した効率の良い実装が可能となった．しかしながら，モデル生成法で支配的な計算量を占める連言照合手続きを効率良く実現することはそれほど容易ではない．我々は，連言照合手続きに含まれる冗長計算を除くため，2 つの方式を考案して比較検討した．両方式ともコーディング手法として KL1 に特化されたものであるが，一般的に前向き推論系の効率的実装技術として KL1 以外の実装言語を用いた場合にも有効な技術である．

MGTP: A Model Generation Theorem Prover in the Concurrent Logic Programming Language KL1

RYUZO HASEGAWA[†] and HIROSHI FUJITA[††]

This paper presents a novel technique to implement model-generation based theorem provers for first-order logic in the concurrent logic programming language KL1. The model generation method makes it possible to use only matching in place of full-unification with occurs-check if the given clause set satisfies the condition called range-restrictedness. As a consequence, a logical variable in the clause set can be represented with a KL1 variable; unification can be executed directly by the head-unification of a KL1 clause; and many of the KL1 system features can be made available for efficient execution of the theorem provers built upon it. In addition, this paper presents two kinds of methods to eliminate redundant computations in conjunctive matching which would be a primary cause of significant speed-down of the model-generation based theorem provers.

1. まえがき

論理プログラミングが新しい計算パラダイムとして登場した後，第5世代コンピュータプロジェクトにおいて「論理」と「並列」がメインテーマに謳われ，並行論理型言語 KL1[7]が核言語として開発された．

論理プログラミングは，ホーン論理式とその証明手続きがプログラムの手続きとその実行に対応づけられるという発見に基づいている．このパラダイムを具現化した Prolog は，プログラミング言語としての実用に耐え得るために多くの工夫がなされた結果，現在ではかなり効率良い処理系が利用可能になっている．

Prolog プログラムの実行自体が定理証明の一部と見なせること，およびその処理系の効率が高いことを利用して，本来の一階述語論理の定理証明を効率良く実現するというアイデアが生じるのは自然であろう．実際，Prolog をベースに，Stickel は PTTP[4]，Manthy と Bry は SATCHMO[3] を開発した．

Prolog ではプログラミング言語としての実用性が追求された結果，単一化における出現検査，非ホーン節あるいは古典論理的否定の扱い，探索の完全性など，定理証明の用途においては欠かせない事項が犠牲にされている．これに対し PTTP は，出現検査付きの単一化手続きを別に用意し，非ホーン節については対偶をとることによってホーン節化する，さらに iterative

† 九州大学工学部電子工学科
Department of Electronics, Kyushu University
†† 三菱電機（株）先端技術総合研究所
Mitsubishi Electric Corporation, Advanced Technology R&D Center

deepening探索によって完全探索を実現する，などの工夫を行った．

さて，KL1は論理プログラミング言語に基礎をおく一方，並行プログラムの記述を容易にすることを目的に開発された．KL1で記述されたプログラムは同期に関するバグが発生しにくいなどの特徴があり，並行プロセスとプロセス間通信の概念に基づく並列処理システムを効率良く実現するのに優れた言語である．

しかしながら，KL1は論理的観点からはPrologよりさらに後退したものになっている．KL1節はコミット演算子という非論理的要素を含んでおり，純粋なホーン節ではない．そして，あるKL1節本体での単一化の失敗はプログラム全体の実行の失敗を意味し，失敗前の状態への巻き戻しと再試行が許されないため，探索に関する完全性が失われている．

ここで我々は，MantheyとBryによる定理証明系SATCHMO[3]に注目した．SATCHMOは，PTTPと同じくPrologを利用しており，極めて簡潔に実現されていることが特徴的である．ただ，PTTPがPrologと同様にゴール（負節）から始まるトップダウン実行に基づいているのに対し，SATCHMOは基本的にはファクト（正節）から始まるボトムアップ実行に基づいている点が大きな違いである．この証明方式を我々はモデル生成法と呼ぶことにする．

モデル生成法によると，値域限定という条件を満たす節集合を対象とした場合，出現検査付き単一化が不要であり，照合操作で十分となる．これは，KL1にとってはPrologの場合にも増して都合の良い性質である．入力節中の変数を直接KL1変数で表現することができ，入力節に対する単一化をKL1節の頭部単一化として高速に実行できるからである．こうしてKL1処理系を活用して定理証明系を効率良く実装する道が開けた．

モデル生成法では連言照合と呼ばれる過程が支配的な計算量を占めており，これを安易に実装すると冗長性が含まれ，顕著な性能劣化の原因となる．KL1の特長を自然に活かした並行プロセスによる実装を考えると，非ホーン節に対応する場合分け推論の際に環境の複製が必要になり，連言照合の冗長性回避をはるかに上まわるオーバヘッドをともなうことが簡単な考察でわかる．冗長計算の除去における要点は，共通計算の記憶と再利用ということであるが，KL1によるコーディングスタイルとの相性は良いとはいえないのである．これに関し，我々は先にRAMS（RAMified Stacks）法と呼ばれるコーディング方式を提案した[2]．本論文では，さらにMERC（Multiple Entry Repeated Combination）法と呼ばれるコーディング方式を示し，両者を比較検討した．

本論文では一階述語論理とKL1の関係にふれた後，モデル生成法の原理を述べ，KL1によるモデル生成型定理証明系の簡潔な実装法を示す．次いで，冗長計算を避けるための2つの方式，RAMS法とMERC法について説明し，その効果を定量的に評価する．

2. 一階述語論理とKL1

並行論理型言語KL1のプログラムは，ガード付きホーン節と呼ばれる節形式（KL1節と呼ぶ）の集合によって表現される．また，KL1の実行系は定理証明法のひとつであるレゾリューション法に基づいている．したがって，一階述語論理式がKL1節として直接表現され，KL1の実行系が健全かつ完全であるならば，KL1で直接一階述語論理の定理証明が可能なはずである．

しかしながら，任意の一階述語論理式（とりわけ非ホーン節）をKL1節で直接的に表現するのは困難であるし，KL1の実行系も一階述語論理の証明系としては完全でも健全でもない．健全でないのは，単一化において出現検査を省略しているからであり，完全でないのは，節選択の際，複数の可能性があってもコミッテッドチョイスによってそのうち1つのみが選ばれ，他は捨てられてしまうからである．

このように，KL1処理系を直接一階述語論理の定理証明系として扱うには無理があるので，メタプログラミング的なアプローチを取らざるをえないが，KL1に受け継がれている論理型言語としての特性を可能な限り活用することが望ましい．

入力節中の論理変数については，(1) KL1の基礎項で表現する，あるいは (2) KL1変数で直接表現する，という2つのアプローチが考えられる．基礎項表現によれば，対象レベルの変数と実装レベルの変数が厳密に区別されるので，意味論的な観点から好都合である．しかし，変数管理手続きをKL1プログラムで実現することになるため，実行時のオーバヘッドが大きくなり，効率上耐え難い方法といえる．

これに対して，KL1変数による直接表現をとった場合，変数管理の多くをKL1処理系に委ねることができ，オーバヘッドの問題は軽減される．

しかしながら，並行言語としての仕様上，KL1変数は同期の機構に関与しているため，ある時点におけるKL1変数の束縛状態についてユーザプログラムから任意に判定/操作することが許されてない．また，その結果，KL1変数を含む項の複製を作ることもでき

ない☆.

こうした問題を解決し，KL1 の論理型言語としての特性を活かしつつ，健全かつ完全な一階述語論理の定理証明系を実現するため，我々はモデル生成法と呼ばれる証明方式を採用した.

3. モデル生成法

任意の一階述語論理式は，常に充足可能性に関して同値な節形式の論理式に変換可能である.

1本の節は正負リテラルを任意の順で選言結合したものであるが，ここでは節 $\neg A_1 \vee \ldots \vee \neg A_n \vee B_1 \vee \ldots \vee B_m$ を次のような含意式の形で表現する.

$$A_1, \ldots, A_n \rightarrow B_1; \ldots; B_m$$

ここで，$A_i (1 \leq i \leq n), B_j (1 \leq j \leq m)$ はアトム，"," は論理積結合子，";" は論理和結合子，"$\rightarrow$" は含意結合子である．"$\rightarrow$" の左辺（のアトム）を前件（リテラル），右辺（のアトム）を後件（リテラル）という．前件が空の場合 $A_0 = \mathit{true}$ と置き正節，後件が空の場合 $B_0 = \mathit{false}$ と置き負節と呼ぶ.

節 $A_1, \ldots, A_n \rightarrow B_1; \ldots; B_m$ において，

$$\bigcup_{i=1}^{n} var(A_i) \supseteq \bigcup_{j=1}^{m} var(B_j)$$

が成り立つとき，この節は値域限定（*range-restrictedness*）の条件を満たすといわれる．ここで，$var(F)$ は式 F に現れるすべての変数の集合である.

節集合 $\mathcal{S}$ に値域限定の条件を満たさない節が含まれる場合には，充足性について $\mathcal{S}$ と同値で，かつ，すべての節が値域限定の条件を満たすような節集合に変換することができる☆☆．しかし，単一化が必須となる無限領域を扱う場合は，このような変換をしても無意味であるので，本論文では対象としない.

モデル生成法は，与えられた入力節（MG 節と呼ぶ）集合 $\mathcal{S}$ に対するモデル，すなわち $\mathcal{S}$ を充足する基礎アトムの集合を，次のような手続きに従って構成的に求めようとするものである．以下では，構成途中のモデルを M で表し，これをモデル候補と呼ぶ．また，モデル候補の集合を $\mathcal{M}$ で表す.

(1) $\mathcal{M}$ の要素として，初期モデル候補 $M_0 = \emptyset$ を設定する.

(2) $\mathcal{M}$ のある要素 M に対し，後述のモデル拡張規則あるいはモデル棄却規則のいずれかが適用可能なとき，その規則を適用して $\mathcal{M}$ を更新する.

(3) 上記2のステップを可能な限り繰り返す.

(4) $\mathcal{M}$ のすべての要素 M に対して，いずれの規則も適用できなくなったとき，手続きは終了する.

この手続きが終了したとき，$\mathcal{M}$ が空になっていれば，$\mathcal{S}$ にはモデルが存在しない（すなわち，充足不能である）ことが結論できる．さもなければ，すべての要素 $M \in \mathcal{M}$ がいずれも $\mathcal{S}$ のモデルとなっている.

ここで，モデル拡張規則とモデル棄却規則は以下のとおりである.

- モデル拡張規則：モデル候補 M に対して，非負節 $A_1, \ldots, A_n \rightarrow B_1; \ldots; B_m$ ならびに，ある基礎代入 σ が存在して，$\sigma A_i \ (1 \leq i \leq n)$ がすべて M で充足され，かつ $\sigma B_j \ (1 \leq j \leq m)$ のいずれも M で充足されないならば，M を m 個のモデル候補 $M \cup \{\sigma B_j\} \ (1 \leq i \leq m)$ で置き換えて拡張（場合分け）する.
- モデル棄却規則：モデル候補 M に対して，負節 $A_1, \ldots, A_n \rightarrow \mathit{false}$ ならびに，ある基礎代入 σ が存在して，$\sigma A_i \ (1 \leq i \leq n)$ がすべて M で充足されるならば，M を棄却する.

ここで，前件 $A_1, \ldots, A_n$ が M で充足されるような σ を求める操作を連言照合と呼ぶ.

例として次の節集合（問題Sと呼ぶ）を考えよう.

C_1 : $\mathit{true} \rightarrow p(a,a); q(b)$.

C_2 : $r(X, f(X)) \rightarrow \mathit{false}$.

C_3 : $p(X,X), p(X,Y) \rightarrow r(X, f(Y))$.

C_4 : $q(X) \rightarrow p(f(X), f(X))$.

モデル生成法による問題Sの証明木を**図1**に示す．この証明木において，$\emptyset$ でラベル付けされた根は初期モデル候補（$M_0 = \emptyset$），$p(a,a)$ などでラベル付けされた節点はモデル候補の拡張に用いられたアトム，枝の分岐はモデル候補の場合分け，$\perp$ でラベル付けされた葉は根からその葉に至る枝上のすべてのアトムを要素とするモデル候補が棄却されたこと，をそれぞれ表している.

この証明木はモデル生成手続きに従い，以下のようにして作られる．C_1 に関してモデル拡張規則を適用することにより，M_0 は $M_1 = \{p(a,a)\}$ と $M_2 = \{q(b)\}$ に場合分けされる．M_1 は C_3 によって $M_3 = \{p(a,a), r(a, f(a))\}$ に拡張される．M_3 は C_2 に関してモデル棄却規則が適用できるので棄却される．一方，M_2 は C_4 によって $M_4 = \{q(b), p(f(b), f(b))\}$ に，M_4 は C_3 によって

☆ Prolog では var（あるいは nonvar）組み込み述語によって，変数束縛状態の判定が可能で，項の複製も copy_term という組み込み述語を用いたり，これと同等のものを var 述語を用いてユーザが定義することが可能である.

☆☆ Herbrand 領域を生成する dom 述語を導入する．詳しくは 3) を参照.

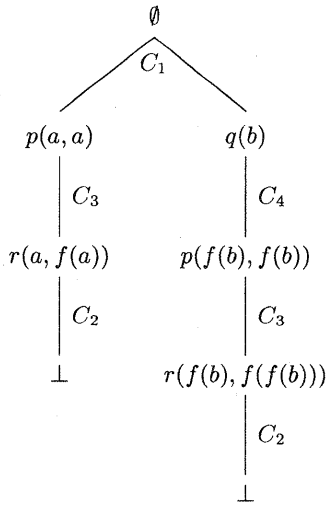


図1 問題Sの証明図
Fig. 1 Proof Tree for Problem S.

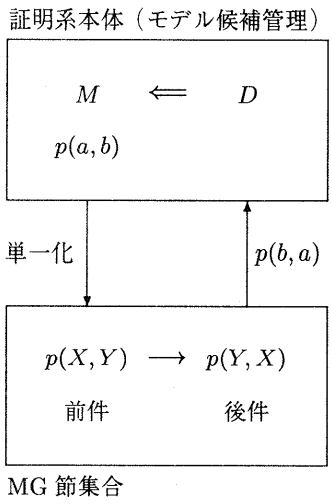


図2 証明系のKL1による実現
Fig. 2 KL1 implementation of the prover.

$M_5 = \{q(b), p(f(b),f(b)), r(f(b),f(f(b)))\}$ に拡張され，M_5 は C_2 によって棄却される．こうしてモデル候補集合が空となるので，問題Sは充足不能であることが証明される．

4. モデル生成型定理証明系 MGTP

4.1 MGTP の基本構成

モデル生成法の推論過程で動的に生成されるのは，モデル候補要素としてのアトムのみである．このアトムは，値域限定条件を満たすMG節集合を対象とする場合，常に変数を含まない基礎アトムとなる．節の前件の充足可能性テスト（連言照合）では，MG節の前件リテラルとモデル候補の要素であるアトムの単一化が行われる．論理変数はモデル候補要素には現れずMG節にのみ現れるから，この場合出現検査付きの単一化は不要であり，常に前件リテラル側の変数を束縛する一方向の単一化，すなわち照合操作で十分である．

さて，KL1ゴールとKL1節の頭部間の頭部単一化は，並行動作における同期機構を簡潔に実現する目的で，KL1節側の変数しか束縛を許さない一方向の単一化になっている．したがって，KL1で実装する場合は，MG節をKL1節で表現し，連言照合はKL1節の頭部単一化を利用するのが簡便かつ効果的と考えられる．これにより，MG節の複製にともなう新変数の導入もKL1実行系に委ねることができる．

そこで，モデル生成型定理証明系MGTPをKL1で実現する際，**図2** のような基本構成をとるのが適切である．すなわち，モデル候補 M を管理する証明系本体とMG節集合を表現するKL1節の部分とを分離し，図中のMG節 $C_n : p(X,Y) \rightarrow p(Y,X)$ は，以下のようなKL1節で表現する☆．

```
c(n,p(X,Y),R):-true|R=p(Y,X).
```

これにより，節の前件の充足可能性テストにおける単一化は，たとえば証明系本体が M からアトム $p(a,b)$ を選び出し，MG節集合部のKL1節を呼んで，その頭部で表現された前件リテラル $p(X,Y)$ に照合させる，というふうに実現することができる．

ここで，$p(X,Y)$ と $p(a,b)$ の照合が成功すると，後件インスタンス $p(b,a)$ が証明系本体に返される．$p(b,a)$ が M によって充足されなければ M を拡張すべきアトムの候補となるので，証明系本体でいったんモデル拡張候補集合 D に格納する☆☆．

4.2 KL1 による MG 節集合の表現

前件が2個以上のリテラルからなる一般の節の場合，たとえば，$C_n : p(X,X), q(X,Z) \rightarrow p(X,Z)$ のようなMG節についても，

```
c(n,(p(X,X),q(X,Z)),R):-true|R=p(X,Z).
```

のような1本のKL1節で表現することができよう．この場合，連言照合のたびにモデル候補中からアトムの対 $\langle A_i, A_j \rangle$ を作ることにすれば前件全体を一度に照合することができる．しかしこの方法には冗長性が含まれている．たとえば $\langle p(a,b), A_j \rangle$ の対はすでに第1リテラルにおいて $p(X,X)$ との照合で失敗することが明らかであるにもかかわらず，すべての A_j につい

☆ 実際には，照合が失敗する場合のために最後にotherwise節が置かれる．

☆☆ 後件の充足性テストは，テスト回数削減のため後件がモデル拡張候補として選ばれる直前に行ってもよい．

```
c(1,true,[],R):-true|R=(p(a,a);q(b)).
c(2,r(X,f(X)),1:[X],R):-true|R=false.
c(3,p(X,X),[],R):-true|R=(1:[X]).
c(3,p(X,Y),1:[X],R):-true|R=r(X,f(Y)).
c(4,q(X),[],R):-true|R=p(f(X),f(X)).
otherwise.
c(_,_,_,R):-true|R=fail.
```

図3 問題S のKL1 節表現
Fig. 3 KL1 representation of Problem S.

て無駄に連言照合を実行してしまうからである.

このような冗長性を省くためには, 連言照合を前件リテラル個別に実行できるように, 1本のMG 節を複数のKL1 節に分離して表現すればよい. ただし, この場合には前件リテラル間の共有変数の同一性をいかに保つかという問題が生じる. これを解決するには, たとえば,

```
c(n,p(X,X),[],R):-true|R=(1:[X]).
c(n,q(X,Z),1:[X],R):-true|R=p(X,Z).
```

のような2本のKL1 節で表現すればよい. ここでは, X が第1リテラルと第2リテラルおよび後件との共有変数であり, 第2リテラルに対応する `c/4` の第3引数で $1:[X]$ を受けることにより, 共有変数 X の同一性を表現している.

この表現法において, 連言照合は次のような手順で行われる. まず, モデル候補中からアトム A_i を1つ選び, 1本めの `c/4` 節を呼び出し, 第1リテラル $p(X,X)$ と照合を行う. 照合が成功したなら, そのときに限り次のアトム A_j を選び, 前の照合で束縛された変数 X とともに2本めの `c/4` 節を呼び, 第2リテラル $q(X,Z)$ の照合を行う.

この方式に基づき, 前出の問題S のMG 節集合をKL1 節集合で表現した実例を**図3** に示す.

$c(N,A,V,R)$ において, N は節番号, A は前件リテラル, V はこの前件リテラルの番号と束縛済み共有変数のリスト, R は照合が成功した際証明系本体に返される値である.

問題S のMG 節 C_3 が2本のKL1 節で表現されていることに注意. C_3 に対する連言照合は以下のように進められる. まず, 証明系本体はモデル候補から1つのアトム A_1 を選び, ゴール $c(3,A_1,[],R_1)$ を起動して C_3 の最初のリテラル $p(X,X)$ を A_1 と照合する. もしこの照合が失敗すれば, 最後のKL1 節により $R_1 = fail$ という結果が返される. 照合が成功すれば, C_3 に対応する1本目のKL1 節によって $R_1 = (1:[X])$ という結果が返され, 証明系本体は C_3 の次のリテラル $p(X,Y)$ の照合を行うためにモデル候補から別のアトム A_2 を選び, ゴール $c(3,A_2,1:[X],R_2)$ を起動する. 第3引数の $1:[X]$ によって第1リテラルで束縛された共有変数 X の値が伝えられる. この X の束縛状態のもとで, $p(X,Y)$ が A_2 と照合するかどうかテストされ, 照合に成功すると $R_2 = r(X,f(Y))$ という結果が返される.

4.3 証明系本体の実現

証明系本体は, **図4** および**図5** に示すKL1 プログラムとして実現することができる.

メインループの `mgtp/5` は, モデル拡張候補集合 D, モデル候補 M, 負節番号のリスト Cn, 非負節番号のリスト Cg を受け, 結果を Res に返す. ここで, モデル拡張候補集合とは, モデル候補 M を拡張するべく節集合 Cg の後件から導かれた, 変数を含まない(基礎)正節の集合のことである. D が空の場合, MG 節集合は M で充足可能(M がモデル)であるという結果を得てメインループを終了する. D が空でない場合, `pickup/3` によりモデル拡張候補 $Delta$ を D から1つ選び, これに対し, `subsTest/3` により M のもとでの充足性を調べる. もし $Delta$ が M で充足されているならば, モデル拡張候補集合の残り $D1$ について再びメインループに入る. そうでなければ, $Delta$ が選言である場合, `caseSplit/6` により場合分けを行う. $Delta$ がアトムである場合, M はこれを含むように拡張される. 拡張されたモデル候補 $[Delta|M]$ に対して, 負節について `cjm/4` を呼び, モデル棄却規則の適用可能性を調べる. もしモデル棄却規則が適用できれば, MG 節集合は M のもとで充足不能である(M はモデルになりえない)という結果を得てメインループを終了する. モデル棄却規則が適用できない場合, 拡張されたモデル候補 $[Delta|M]$ に対して, 非負節について `cjm/4` を呼び, モデル拡張規則の前半, すなわち前件の連言照合を適用し, 前件が M で充足された非負節の後件からなる集合を得て, これを新たなモデル拡張候補集合 New とする. これを `addNew/3` によってモデル拡張候補集合の残り $D1$ に加え $NewD$ として再びメインループに入る.

連言照合手続き `cjm/4` は, 節番号 N のリストとモデル候補 M を受け, リスト中のすべての節について連言照合 `cjm1/6` を行い, 結果を差分リスト $Rh - Rt$ に詰めて返す. 1本のMG 節に対する連言照合手続き `cjm1/6` では, 各前件リテラルに対し, M 中のすべてのアトムについて連言照合を行う. このとき, MG 節集合のKL1 表現 `c/4` が呼ばれる. 当該リテラルについて M 中のアトム A による `c/4` での照合結果が

```
mgtp(D,M,Cn,Cg,Res):-true|
 empty(D,E),
 (E=yes->
   Res=sat;
  otherwise;true->
   pickup(D,Delta,D1),
   subsTest(Delta,M,S),
   (S=yes->mgtp(D1,M,Cn,Cg,Res);
    S=no,Delta=(_;_)->
     caseSplit(Delta,D1,M,Cn,Cg,Res);
    otherwise;true->
     cjm(Cn,[Delta|M],F,[]),
     (F=[false|_]->
       Res=unsat;
      otherwise;true->
       cjm(Cg,[Delta|M],New,[]),
       addNew(New,D1,NewD),
       mgtp(NewD,[Delta|M],Cn,Cg,Res)))).

caseSplit((A;B),D,M,Cn,Cg,Res):-true|
 caseSplit(A,D,M,Cn,Cg,R1),
 (R1=sat->Res=sat;
  otherwise;true->
   caseSplit(B,D,M,Cn,Cg,Res)).
otherwise.
caseSplit(A,D,M,Cn,Cg,Res):-true|
 addNew([A],D,NewD),
 mgtp(NewD,M,Cn,Cg,Res).

subsTest((A;B),M,S):-true|
 subsTest(A,M,S1),
 (S1=yes->S=yes;
  otherwise;true->subsTest(B,M,S)).
otherwise.
subsTest(A,[A|_],S):-true|S=yes.
otherwise.
subsTest(A,[_|M],S):-true|subsTest(A,M,S).
subsTest(_,[],S):-true|S=No.
```

図4 基本MGTP証明系
Fig. 4 Basic MGTP.

$fail$ の場合，このリテラル–アトム対の照合結果は空である．共有変数情報 $(\_:\_)$ が返された場合，引き続くリテラルについての連言照合を進める．それ以外の場合，前件全体の連言照合が成功して後件が返されているので，これを $Rh - Rm$ の差分リストに詰めて返す．当該リテラルについて M 中のほかのアトムによる連言照合結果は，差分リスト $Rm - Rt$ に詰めて返される．

```
cjm([N|Cs],M,Rh,Rt):-true|
 cjm1(N,M,[],M,Rh,Rm),
 cjm(Cs,M,Rm,Rt).
cjm([],_,Rh,Rt):-true|Rh=Rt.

cjm1(N,[A|M],V,Mh,Rh,Rt):-true|
 problem:c(N,A,V,R),
 (R=fail->Rh=Rm;
  R=(_:_)->cjm1(N,Mh,R,Mh,Rh,Rm);
  otherwise;true->Rh=[R|Rm]),
 cjm1(N,M,V,Mh,Rm,Rt).
cjm1(_,[],_,_,Rh,Rt):-true|Rh=Rt.
```

図5 基本MGTP証明系の連言照合部
Fig. 5 Conjunctive matching part of MGTP.

5. 連言照合における冗長性除去

5.1 連言照合の冗長性

基本MGTPの連言照合には重複計算が含まれている．簡単のために，前件リテラルが2個のMG節 $A_1, A_2 \rightarrow B$ を考える．図4の `mgtp/5` の中の`cjm/4` の呼び出しでリテラルの対 $\langle A_1, A_2 \rangle$ に対して，アトムの対が $[Delta|M]$ のアトム集合から組み合わされて連言照合が行われる．そのアトム対の総数は，$(1+|M|)^2$ である．次に，`mgtp/5` の再帰呼び出しが起こると，`cjm/4` の呼び出しで同じリテラル対 $\langle A_1, A_2 \rangle$ に対して，アトム対が $[Delta', Delta|M]$ のアトム集合から組み合わされて連言照合が行われる．そのアトム対の総数は，$(1+(1+|M|))^2$ である．ところが，このうち $[Delta|M]$ から組み合わせたアトム対 $(1+|M|)^2$ 組分についてはすでに照合済みであり，これを再び繰り返す☆のはまったく無駄である．ここでは，$\langle Delta', [Delta|M] \rangle$, $\langle [Delta|M], Delta' \rangle$, $\langle Delta', Delta' \rangle$ の合計 $(1+2(1+|M|))$ 個だけが新たに連言照合が必要なアトム対である．

SATCHMOにおいても，1つのアトムによってモデルが拡張されるたびに拡張されたモデル候補の全要素について連言照合があらためて実行されるので，上と同じような重複計算が生じている．

ハイパーレゾリューションをサポートする汎用定理証明系OTTER[5]では，節

☆ これをステージ間冗長性という．

$A_1, A_2, \ldots, A_n \rightarrow C_1; C_2; \ldots; C_m$

に対して，まず，モデル拡張候補 Δ をある A_i に照合させる．その後，残ったリテラル

$\{A_1, A_2, \ldots, A_n\} - \{A_i\}$

のそれぞれに対して $M \cup \{\Delta\}$ から取り出したアトムを照合させる．これによれば，A_i のすべてのリテラルが M のアトムと照合されるという重複は除去できる．しかし，前件リテラル数が2以上の場合，重複計算の可能性が残っている．すなわち，対 $\langle A_i, A_j \rangle (i \neq j)$ に対し，対の集合 $\langle \{\Delta\}, M \cup \{\Delta\} \rangle$ と $\langle M \cup \{\Delta\}, \{\Delta\} \rangle$ からのアトムを照合させるので，$\langle \Delta, \Delta \rangle$ の分が重複するのである．以下に述べる MERC 法は，OTTER の方法をより精密化したもので，モデル生成法の値域限定条件を利用して上の $\Delta - \Delta$ 冗長性を除去している．

OTTER 方式や MERC 法では前件全体としての連言照合の重複は除去するが，前件リテラル数が2以上の場合，部分的にはまだ重複計算の可能性がある．RAMS 法は，リテラルごとの照合結果を記憶することによりこの重複を完全に除去しようとするものである．RAMS 法の実現にあたっては，RETE ネットワークに類似したプロセス指向のコーディングが自然と考えられるが，非ホーン節によって引き起こされる場合分けに対し，1）モデル候補中のアトムに色をつけ，どの場合分けに対応したモデル候補に属しているのかを識別する，あるいは，2）プロセスネットワーク全体を場合分けが起こるたびに複製する，といった対策が必要となる．いずれも大きなオーバヘッドをともなうので，実際的とはいえない．MERC や RAMS では，LIFO 型の記憶機構を利用することにより，場合分けを実現する際の記憶共有の問題を解決している．

5.2 RAMS 法

RAMS 法は，前件の照合の履歴を記憶する機構を設けることにより，連言照合における重複計算を避けるものである．**図6** にこの方法を図示している．

一般に，MG 節の前件 $A_1, \ldots, A_n$ の各リテラル A_i につき1つのインスタンススタック S_i を割り当てる．各 S_i には $A_1, \ldots, A_i$ のモデル候補 M との照合結果（成功した場合の変数束縛情報）が蓄えられる．S_1 には A_1 の照合結果が格納され，A_i $(i > 1)$ の照合では，S_{i-1} に蓄えられた $A_1, \ldots, A_{i-1}$ の各照合結果のもとで，A_i 自体と M との照合を行う．以下，A_i $(i > 1)$ に対するこの照合演算結果を $S_{i-1} \circ M$ と表す．

各インスタンススタック S_i は，前ステージまでに積まれた部分 S_i^{k-1} と，現在のステージにおいて積まれる部分 δS_i^k との2つの部分に分けられる．ここで，指標 k はメインループのステージ番号を表す．また，

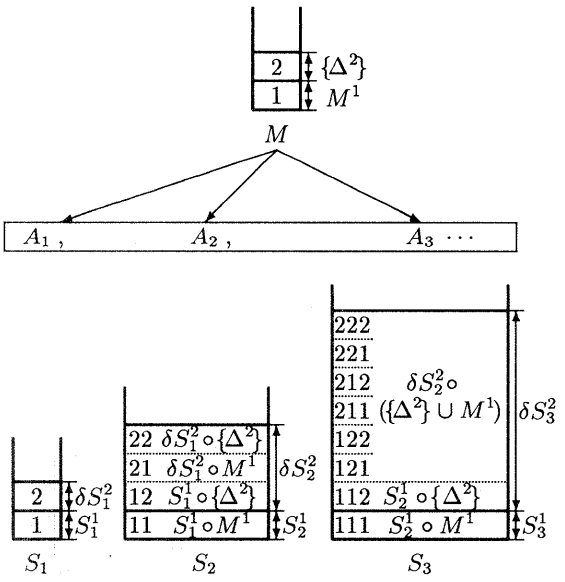


図6 インスタンススタック
Fig. 6 Instance stacks.

前ステージまでのモデル候補 M^{k-1} を拡張するアトムをモデル拡張アトムと呼び，Δ^k で表す．

A_1 における作業 T_1^k は，A_1 と Δ^k との照合であり，その結果が S_1 に積まれる．各リテラル A_i $(2 \leq i \leq n)$ における作業 T_i^k は，連言照合の後，次のようにスタックの更新を行う．

$$\delta S_i^k := \delta S_{i-1}^k \circ (\{\Delta^k\} \cup M^{k-1}) \cup S_{i-1}^{k-1} \circ \{\Delta^k\}$$
$$S_i^k := S_i^{k-1} \cup \delta S_i^k$$

ただし，A_n における作業 T_n^k は連言照合の最後であり，照合結果に対応する後件が得られた後は $A_1, \ldots, A_n$ の照合計算結果自体をスタックに積む必要がない．したがって，最後のリテラル A_n に対するスタック S_n も実際に割り付ける必要はない．

こうして，作業列 $T_1^k, \ldots, T_n^k$ をこの順に実行することにより，前件 $A_1, \ldots, A_n$ の連言照合を重複なしに実現することができる．図6は，2ステージ目で，M にアトム1が積まれ，Δ^2 がアトム2であるという状況を表している．まず，A_1 と Δ^2 の照合が成功し，その結果（図中，簡単のため2と表記）が δS_1^2 として S_1 に積まれる．この δS_1^2 のもとで A_2 と Δ^2 の照合が成功し，その結果（図中22で表す）が δS_2^2 の1つとして積まれている．

RAMS 法はスタックを用いているので，非ホーン節に対しても容易に対応できる．それは，モデル候補 M ならびに各インスタンススタック S_i は，ボトムからトップまでのアトムの集合で表現されており，その

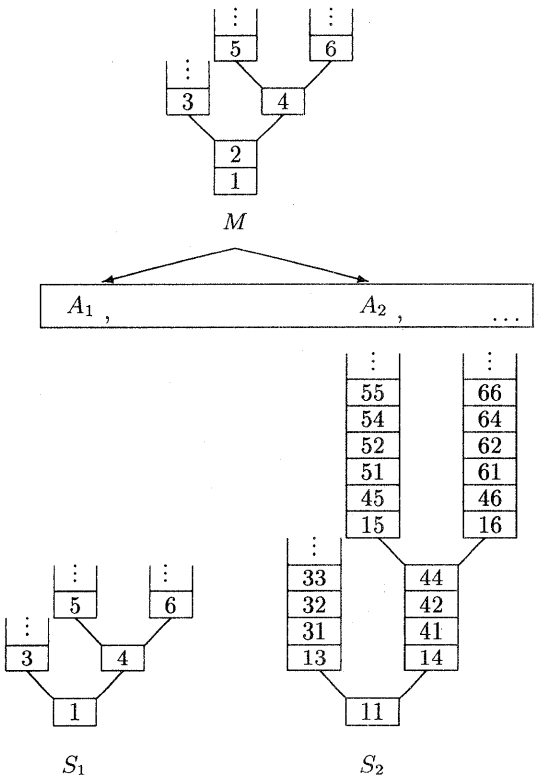


図7 分岐スタック
Fig. 7 Ramified stacks.

拡張はスタックトップにおいてのみ行われ，連言照合時にこれを参照する際もスタックトップからのみ行われるからである．非ホーン節によって場合分けが起こるときにはスタックトップが複数生じるが，各場合のモデル候補中でアトムを参照する際には，それぞれのスタックトップからボトム方向にたどる道筋が一意であり，場合分けの履歴を調べる必要がない．このようにして，非ホーンのMG節を含む一般の場合には図7に示すような分岐するスタックが生成される．本図では，第1リテラル A_1 と M 中のアトム2の照合は失敗するものと仮定しているので，この結果は S_1 には積まれていない．

RAMS法に基づいて，基本MGTP証明系の連言照合部を修正したものを図8に示す☆．

5.3 MERC法

MERC法では，一般にMG節の前件 $A_1,\ldots,A_n$ に対し，M と Δ に対する連言照合を図9に示すような組合せで行う．本図の第1行に示すパタンは，A_1 に Δ を，$A_2,\ldots,A_n$ に M 中のアトムを照合させることを表す．ここで，$A_1,\ldots,A_n$ の少なくとも1つは Δ と照合するような組合せのみを選ぶようにし，$A_1,\ldots,A_n$ のすべてが M に含まれるアトムと照合されるような組合せを排除すれば，ステージ間冗長性が避けられるわけである．

以下では，Δ と照合させるリテラルを入口リテラルと呼び，M 中のアトムを照合させるリテラルを後続リテラルと呼ぶことにする．連言照合は，まず入口リテラルに Δ を照合させ，これが成功した場合に後

```
cjm([N:L|Cs],Cs1,[Delta|M],Rh,Rt):-true|
 Cs1=[N:L1|Cs2],
 cjm1(N,L,L1,Delta,M,[[]],[],Rh,R1),
 cjm(Cs,Cs2,[Delta|M],R1,Rt).
cjm([],Cs1,_,_,Rh,Rt):-true|Cs1=[],Rh=Rt.

cjm1(N,[Sn|Ls],Ls1,Delta,M,Sp,Dp,Rh,Rt):-
 true|
 Ls1=[NewSn|Ls2],
 cjm2(N,[Delta|M],Dp,NewSn,S1,Dn,D1),
 cjm2(N,[Delta],Sp,S1,Sn,D1,[]),
 cjm1(N,Ls,Ls2,Delta,M,Sn,Dn,Rh,Rt).
cjm1(N,[],Ls1,Delta,M,Sp,Dp,Rh,Rt):-true|
 Ls1=[],
 cjm2(N,[Delta|M],Dp,Rh,R1,_,_),
 cjm2(N,[Delta],Sp,R1,Rt,_,_).

cjm2(N,As,[V|Vs],Sh,St,Dh,Dt):-true|
 cjm3(N,As,V,Sh,S1,Dh,D1),
 cjm2(N,As,Vs,S1,St,D1,Dt).
cjm2(_,_,[],Sh,St,Dh,Dt):-true|
 Sh=St,Dh=Dt.

cjm3(N,[A|As],V,Sh,St,Dh,Dt):-true|
 problem:c(N,A,V,R),
 (R=fail->Sh=S1,Dh=D1;
  R=(_:_)->Sh=[R|S1],Dh=[R|D1];
  otherwise;true->Sh=[R|S1]),
 cjm3(N,As,V,S1,St,D1,Dt).
cjm3(_,[],_,Sh,St,Dh,Dt):-true|
 Sh=St,Dh=Dt.
```

図8 連言照合部のRAMS版
Fig. 8 RAMS version for CJM.

☆ cjm/5の第1引数（Cs）は，節番号とインスタンススタックの対（N:L）のリストであり，第2引数（Cs1）は，Csを変更したものが返される．これに応じて，mgtp/5の中のcjm/4をcjm/5へ，また，mgtp/5の再帰呼び出しの中の第3，4引数にcjm/5の第2引数で返された値を与えるように変更する．

	A_1	A_2	A_3	...	A_n
(イ)	Δ	M	M	...	M
	M	Δ	M	...	M
	M	M	Δ	...	M
			...		
	M	M	M	...	Δ
(ロ)	Δ	Δ	M	...	M
	M	Δ	Δ	...	M
			...		
	Δ	Δ	Δ	...	Δ

図9 MERC 法による連言照合
Fig. 9 CJM by MERC.

続リテラルを M 中のアトムで照合させるように進める．しかしながら，前節までに示した KL1 節表現ではリテラル順序が 1 つに固定されており，任意のリテラルを入口リテラルとすることができない．したがって，MERC 法においては 1 本の MG 節に対し，入口リテラルの異なる取り方ごとにそれぞれ別個の KL1 節として表現する必要がある．

図 9 中 (イ) に属する，入口リテラルが 1 つのパタンについては，各パタンにつき，$A_1,\ldots,A_n$ の順序を入れ替えて入口リテラルを先頭とする 1 本の MG 節を考え，これに対応する KL1 節を用意する．一方，図中 (ロ) に属する，複数の前件リテラルが入口リテラルとなるパタンについては，KL1 節を省ける場合がある．たとえば，2 つのリテラル A_j，A_k が同時に Δ と照合するのは，これらが単一化可能な場合 ($A_j = A_k$ と表す)，すなわちファクタリング可能な場合に限られる．したがって，図中 (ロ) に属するパタンのうち，単一化不可能な入口リテラル対を含むものについては KL1 節を用意しなくてよい．一方，単一化可能な複数の入口リテラル $A_{i1},\ldots,A_{il}$ については，ファクタリングを行った結果のリテラル A_r ($A_{i1} = A_{i2} = \ldots = A_{il}$) を計算し，これを 1 つの新たな入口リテラルとした KL1 節を用意することにする．

このようにして，問題 S の KL1 節表現を作ると**図 10** のようになる．ここで，MG 節 C_3 において `c3_1`，`c3_2` のように 2 つの入口リテラルごとに別個の KL1 節が用意されている．また，`c3_1_2` は，2 つの前件リテラルをファクタリングしたものに対応している．MERC 法を採用する場合，基本 MGTP 証明系の連言照合部は一部**図 11** のように修正される．非ホーン節への対応は，RAMS 法と同様に，M を分岐スタックで実現することによりなされる．

```
c(c1,true,      [],R):-true|R=(p(a,a);q(b)).
c(c2,r(X,f(X)),[],R):-true|R=false.
c(c3_1,p(X,X),   [],R):-true|R=(1:[X]).
c(c3_1,p(X,Y),1:[X],R):-true|R=r(X,f(Y)).
c(c3_2,p(X,Y),   [],R):-true|R=(1:[X,Y]).
c(c3_2,p(X,X),1:[X,Y],R):-true|R=r(X,f(Y)).
c(c3_1_2,p(X,X), [],R):-true|R=r(X,f(X)).
c(c4,q(X),      [],R):-true|R=p(f(X),f(X)).
otherwise.
c(_,_,_,R):-true|R=fail.
```

図 10 問題 S の MERC 版 KL1 表現
Fig. 10 KL1 representation in MERC version for problem S.

```
cjm([N|Cs],[Delta|M],Rh,Rt):-true|
 problem:c(N,Delta,[],R),
 (R=fail->Rh=Rm;
  R=(_:_)->cjm1(N,M,R,M,Rh,Rm);
  otherwise;true->Rh=[R|Rm]),
 cjm(Cs,[Delta|M],Rm,Rt).
cjm([],_,Rh,Rt):-true|Rh=Rt.
```

図 11 連言照合部の MERC 版
Fig. 11 MERC version for CJM.

6. 議　　論

RAMS 法では前件リテラル $A_1,\ldots,A_{i-1}$ に対するモデル候補 M との照合結果を記憶し，その記憶をもとに A_i と最新のアトムである Δ の照合を行うのに対し，MERC 法では $A_1,\ldots,A_{i-1}$ に対する照合を再計算するという冗長性がある．これを M-M 冗長性と呼ぶ．さらに，MERC 法では 1 本の MG 節を入口リテラルの数だけ複製しているので，上記の冗長性は複製分だけ重複して現れる．

しかしながら，MERC 法では入口リテラル A_i と Δ との照合を優先し，後続リテラル $A_1,\ldots,A_{i-1}$ と M 中のアトムとの照合を後に行うのに対し，RAMS 法では，与えられた MG 節の前件リテラルの並びの順序によって照合順序が固定されており，Δ との照合が優先されるということはない．その結果，MERC 法では Δ による入口リテラルの照合が失敗したときには後続リテラルの照合が省略されるが，RAMS 法では後続リテラルの照合で成功する組合せのすべてについて入口リテラルと Δ との無駄な照合を繰り返す可能性がある．これを Δ 失敗冗長性と呼ぶ．

また，メモリ消費に関しては，RAMS 法では証明

実行時にインスタンススタックの分だけ余計にメモリを消費するのに対し，MERC 法では入口リテラルの数の分だけ節の複製を必要とする．

ステージ間冗長性，M-M 冗長性，ならびに Δ 失敗冗長性，さらにメモリ消費特性が，証明系の全体性能にどのように影響するかは，対象の問題に強く依存しており，RAMS 法と MERC 法の優劣は明確でない．加えて，RAMS 法と MERC 法では，連言照合で取り出されるモデル候補中のアトムの組合せ順序が異なるので，生成されるモデル拡張候補アトム集合の並べ換えを行わない限り，一般には Δ の選ばれる順序が異なり証明も異なってくる．

なお，本論文では 2 本以上の MG 節で共通の前件リテラルを含む場合に生じる，連言照合の節間冗長性については議論しない．

7. 評　　価

本節では，連言照合における冗長計算の有無がいかに MGTP の全体性能に影響を与えるかを，定理証明系のベンチマーク用例題集 TPTP ライブラリ[6] を用いて検証する．

一般に，得られる証明は推論方式自体や戦略によって変わり，異なる証明系の比較評価は難しい．本実験では，これらの影響をなくし，連言照合の効率特性のみを調べるため，比較対象の各 Naive 版（基本 MGTP），RAMS 版，MERC 版とも共通に以下の戦略を用いた．

(1) 図 4 の **addNew/3** におけるモデル拡張候補集合 D への New の追加/取り出しはキュー（FIFO）を用いる．
(2) **pickup/3** では *Delta* として単位節を優先して選択する．
(3) 項の重み付けによる並び換え（ソーティング）戦略や削除戦略は用いない．

したがって，証明の違いは連言照合における照合順序の差異のみから生じる．そこで本評価実験では，MERC 版と RAMS 版に対し，Naive 版と証明を一致させるために New の要素の整列を行っている．

表 1 に，Naive 版，RAMS 版，MERC 版 MGTP の計測結果を示す．本表中，Clauses 欄は MG 節の本数と KL1 節の本数（括弧内は MERC 版の KL1 節本数）を表し，Proof 欄の Br は反駁証明木の分枝数，Dp は枝の長さ（棄却されたモデル候補の大きさ）の平均値を表す．また，CJM 欄の値は連言照合における KL1 節 **c/4** の呼び出し総数で，RAMS の括弧内はインスタンススタックに要したセルの総数を表す．Time 欄の値は SPARC10-30 上の KL1 処理系 KLIC[1] による CPU 時間を表す．ただし，証明一致のための整列に要した時間は除いてある．また，“TO” は 1 時間以内に解が得られなかったことを，“-” は測定不能を表す．

KL1 節の本数は Naive 版と RAMS 版では常に同数だが，MERC 版の場合 SYN006-1 を除いて 2 割から 3.5 倍位までに増加している☆．証明木は三者とも完全に一致しており，LAT，PLA，GRP はホーン問題なので，棄却モデル候補数は 1 個（分枝数 Br が 1）である．そのほかは非ホーン節を含む問題で，棄却モデル候補数は数十から数十万である．また，棄却モデル候補 1 個あたりの大きさ（アトム数）は，ホーン問題では数十から千数百，非ホーン問題では数十の規模である．

CJM の回数については，Naive 版と RAMS 版では 1 桁から 2 桁以上の開きがあり，RAMS 版による大幅な削減効果が見られる．RAMS 版と MERC 版を比べると，CJM 回数は SYN006-1 で同じ，PLA001-1，PLA003-1，SYN009-1 で RAMS 版の方が 3 割から 2 倍弱増加しているが，そのほかの問題では逆に MERC 版の方が 1 割から 3 倍近く増加している．全体の傾向としては，RAMS 版の方がおおむね CJM 回数は少なくて済んでいる．しかし，RAMS 版では SYN006-1 を除き，インスタンススタックのために他の版より数十から数十万セルのメモリを余分に要している．

CPU 時間で見ると，Naive 版とそれ以外の RAMS 版や MERC 版との間には 10 倍から数千倍以上の開きがあるが，RAMS 版と MERC 版は大差はなく，相互の CPU 時間比は問題により 1 割から 6 割程度の範囲で変動している．RAMS 版では，CJM 回数が MERC 版より少ないにもかかわらず CPU 時間が長くなっている場合があるが，これはインスタンススタック操作に関するオーバヘッドのためと思われる．

以上の結果から，まずステージ間冗長性による無駄な連言照合を除くだけで全体性能を著しく改善できることがわかる．次に，RAMS 版と MERC 版の差異を分析するために，PLA001-1 においては MERC が，PRV009-1 では逆に RAMS の方が，いずれも倍近く速くなっていることに注目する．

PLA001-1 で前件リテラルが 2 個以上の節は，

$$at(A,B,C,D), next\_to(A,E) \rightarrow \ldots$$

という異なる述語の 2 個の前件リテラルを持つ節のみである．ここで，第 2 リテラル $next\_to(A,E)$ に対し，

☆ SYN009-1 を除き，値域限定条件を満たすよう dom 述語を導入している．

表 1 MGTP 各版の計測結果
Table 1 Evaluation Results.

Problem	Clauses			Proof		CJM				Time (sec)		
	MG	KL1	(merc)	Br	Dp	Naive	RAMS	(stack)	MERC	Naive	RAMS	MERC
LAT005-1	31	52	(182)	1	395	-	8065k	(20386)	11669k	TO	223.2	183.2
LAT005-2	31	46	(135)	1	248	-	1908k	(7664)	2453k	TO	28.2	30.0
MSC006-1	6	10	(17)	612	24	1482k	115k	(2401)	125k	19.8	1.5	1.4
PLA001-1	16	18	(22)	1	1378	-	3789k	(2734)	1922k	TO	41.6	25.8
PLA003-1	11	20	(42)	1	89	513k	14k	(152)	9k	832.3	0.3	0.3
PUZ012-1	18	23	(33)	165	27	606k	31k	(332)	40k	5.5	0.4	0.4
PUZ025-1	24	47	(108)	90	22	424k	34k	(773)	68k	4.4	0.4	0.6
SYN006-1	7	9	(9)	96	84	328k	4k	(0)	4k	41.3	0.2	0.2
SYN009-1	7	9	(22)	19683	12	2918k	246k	(12)	187k	49.6	4.6	3.5
SYN015-2	26	36	(62)	196	43	17057k	687k	(7521)	737k	603.0	8.9	7.8
SYN036-3	36	63	(123)	516	33	15171k	886k	(12548)	1314k	197.1	12.6	12.7
SYN037-1	36	63	(123)	358	33	9053k	535k	(7473)	832k	120.4	7.8	8.0
GRP028-1	4	12	(33)	1	1834	-	64835k	(35345)	83559k	TO	1050.3	971.0
PRV009-1	9	19	(64)	266240	31	-	35307k	(314110)	81122k	TO	498.1	818.2

Δ として $at(\_,\_,\_,\_)$ を照合させると失敗する．したがって，RAMS 版では Δ 失敗冗長性のために MERC 版よりも余計な CJM を行う結果，証明時間が長くなると考えられる．

一方，PRV009-1 においては，

$$le(m,A), lt(A,i), lt(j,B), le(B,n) \rightarrow \ldots$$
$$le(m,A), le(A,B), le(B,j) \rightarrow \ldots$$
$$le(i,A), le(A,B), le(B,n) \rightarrow \ldots$$

という 3 個以上の前件リテラルを持つ節が 3 本ある．このように前件リテラル数 3 以上の節が多く含まれ，かつ 1 本の節の前件に同じ述語が複数出現するような問題では，Δ 失敗冗長性よりも M-M 冗長性の影響の方が優勢となるため，MERC 版の方が RAMS 版よりも証明時間が長くなると考えられる☆．

以上見てきたとおり，RAMS 版においては Δ 失敗冗長性とインスタンススタックにともなうオーバヘッド，MERC 版においては M-M 冗長性というそれぞれに固有の性能劣化の原因がある．M-M 冗長性が大きくかつ記憶量の増大が許容される場合には RAMS が有利であり，逆に Δ 失敗冗長性が大きいか記憶量の増大が許容できない場合には MERC が有利である．

8. ま と め

モデル生成型定理証明器 MGTP を並行論理型言語 KL1 で実現した．モデル生成法の採用により，値域限定を満たす節集合に対して出現検査付き単一化が不要となり照合操作で十分となることを利用し，KL1 の特長を活かして一階述語論理の効率良い定理証明系を実現することが可能となった．KL1 による実現のメリットは次のとおりである．

- 入力節中の論理変数が KL1 変数で直接表現される．
- 入力節の単一化は KL1 節の頭部単一化として実行される．
- 入力節の複製時に必要な新変数は，KL1 節の呼び出しメカニズムから自動的に得られる．

さらに，モデル生成法における連言照合を効率的に行うため，RAMS，MERC の 2 つの方式を考案した．いずれの方式も連言照合のステージ間冗長性を除去することによって MGTP の実行効率を顕著に改善できる．しかし，RAMS には Δ 失敗冗長性，MERC には M-M 冗長性という互いに性質の異なる冗長性が残っており，それらの影響は問題によって異なる．

本稿では SPARC10-30 上での性能のみを示したが，MGTP は容易に並列化可能である．queen 問題や pegeon hole 問題など，分岐数が大で均衡した証明木が得られる非ホーン問題に対しては PIM/m-256PE 上でほぼ線形の台数効果が得られている．

謝辞 本研究に関して，有益な討論および参考データをいただいた ICOT の越村三幸氏，計算機上での計測の労をとっていただいた九州ジェービーエー・山鹿英樹氏に感謝いたします．また，本研究の機会とご支援をいただいた ICOT の淵一博前研究所長（現東大教授），内田俊一研究所長に深く感謝いたします．

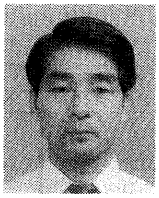
参 考 文 献

1) Chikayama, T., Fujise, T. and Sekita, D.: A Portable and Efficient Implementation of KL1, *Proc. 6th Int. Symp. on Programming Language Implementation and Logic Programming* (1994).
2) Fujita, H. and Hasegawa, R.: A Model Generation Theorem Prover in KL1 Using a Ramified-Stack Algorithm, *Proc. 8th ICLP*, pp.535–548

☆ RAMS 版において，PRV009-1 の CPU 時間は PLA001-1 の場合の約 10 倍であるのに対し，インスタンススタックの量は 100 倍以上になっていることから，PRV009-1 では Δ 失敗の頻度が小さいことが推察される

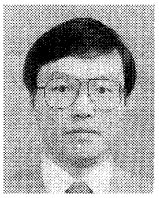
(1991).
3) Manthey, R. and Bry, F.: SATCHMO: A Theorem Prover Implemented in Prolog, *Proc. 9th CADE,* pp.415–434 (1988).
4) Stickel, M. E.: A Prolog Technology Theorem Prover: Implementation by an Extended Prolog Compiler, *Journal of Automated Reasoning,* Vol.4, pp.353–380 (1988).
5) McCune, W.: OTTER 2.0 Users Guide, Technical Report ANL-90/9, Argonne National Laboratory (1990).
6) Suttner, C., Sutcliffe, G. and Yemenis, T.: The TPTP Problem Library, *Proc.12th CADE,* pp.252–266 (1993).
7) Ueda, K. and Chikayama, T.: Design of the Kernel Language for the Parallel Inference Machine, *The Computer Journal,* Vol.33, No.6, pp.494–500 (1990).

(平成6年 8 月4日受付)
(平成7年10月5日採録)



長谷川隆三（正会員）

1949年生．1972年九州大学工学部通信工学科卒業．1974年九州大学大学院工学研究科通信工学専攻修士課程修了．同年 日本電信電話公社入社．同社武蔵野電気通信研究所勤務．1987年–1995年（財）新世代コンピュータ技術開発機構へ出向．現在，九州大学工学部教授．工学博士．ポリプロセッサシステム，データフローマシン，関数型言語，論理プログラミングおよび定理証明に関する研究に従事．電子情報通信学会会員．



藤田　博（正会員）

1955年生．1978年東京大学理学部物理学科卒業．1980年同大学院情報工学修士課程修了．同年三菱電機（株）入社．同社先端技術総合研究所勤務．1986年–1990年（財）新世代コンピュータ技術開発機構に出向．工学博士．論理プログラミング，部分計算および定理証明に関する研究に従事．日本ソフトウェア科学会，ACM各会員．