論　文

変数を共有する問題の擬似並列処理*

小　川　　均**　田　中　幸　吉**

Abstract

In the problem-reduction approach to automatic problem solving, if several sub-problems share the same variables, then the way in which one problem is solved may affect the solution to the others. In such cases, goal-oriented methods, which assume independency of subgoals, can not be put to practical use.

The authors have developed a pseudo-parallel system for solving problems of the above-mentioned type. The system consists of a scheduler, node micro-actors (NMA) (corresponding to subgoals) and variable micro-actors (VMA) (corresponding to the variables being sent from NMA to NMA). The scheduler controls two kinds of messages passing between one for search, the other for backtracking. Though the NMAs act independently, common NMAs: variables have the same value in each NMA; for NMAs directly converse with VMAs. D-graph deduction, a type of formal deduction using AND/OR graphs, has been implemented in the system.

1. まえがき

問題解決の基本的な動作の一つは，ある問題を幾つかの副題に分け，それらの副題について個々に独立して解決を行うことである[1]．この動作の長所は各副題を独立して並列に解決することにあるが，副題に共通した変数がある場合には問題が生じる．それは各副題を解く際に，共通変数には同じ代入を行わなければならないことであり，独立性が失なわれ，この方法の特長を有効に活用できない．

上の問題を避けるため，いろいろなアプローチがなされてきた．Slagle らは導出原理による定理証明を行う際に，1つの節を共通変数を持たない数個の節に分割し，各節について独立に反ばく (refutation) を求める方法を用いた[2]．これはある程度の効果を上げたが，本質的な問題の解決とはならなかった．一方，Levi と Sirovich のアプローチでは，探索空間を表わすグラフを拡張する時，すなわち，新しい頂点が生成される時，頂点 N の親頂点が共通変数を持つなら，その共通変数を持つ端点のすべてから頂点 N へ有向枝が存在するように拡張しなければならないという制限を適用している[3]．この方法は，各頂点で共通変数に異なる代入をするために間違った解を推論するのを防ぐために，グラフ全体を扱うものであり，個々の副題について独立した推論をしているとは考えられない．また，この方法では，述語論理または節をそのままの形ではなく，論理的に等価な等別の形を用いなくてはならず，表現方法にも問題があると思われる．

Nevins は2つのアプローチを行った[4),5]．これらは，上の2つが宣言的手法であるのに対して，手続き的手法である．各問題に手続きが対応しているので，副題を処理する際の制限は，対応する手続きを呼出す時の条件とみなすことができる．1つのアプローチは，共通変数が同時に出現できる数（普通1個）を制限している．また，ある述語Pと AND 関係にある他の述語すべてについて解決されるまでPについての推論を延期し，得られたデータのすべてについてPを推論するプログラム関数 RECURSIVEX を用いてい

* Pseudo-Parallel Processing for Problems with Common Variables by Hitoshi OGAWA and Kokichi TANAKA (Department of Information & Computer Sciences, Faculty of Engineering Science, Osaka University).

** 大阪大学基礎工学部情報工学科

る．変数が具体化されないままに推論を行うと探索空間が過大に拡張されるような述語，または，再帰的に使用される述語に関数 RECURSIVEX を適用すれば非常に効果的である．この関数の使用は使用者に依存しているため，この方法の効率性，完全性は使用者に依る．他のアプローチは，前者がゴール（問題）からデータ（事実）を見つける backward chaining のみを用いているのに対して，既知のデータから新しいデータを得る forward chaining と backward chaining の両方を用いている．この方法は，他と共通変数を共有する問題を backward に解決する時には，他の問題が forward chaining により解決されていることを条件としている．したがって，backward に処理する時は事実上共通変数は存在しない．

以上4つのアプローチについて述べたが，これらはいずれも単に共通変数の出現を制限する方法であり，最初に述べた，副題を独立に解決する長所を十分には生かしていない．著者らは計算機における知識表現の基本構造として μ-actor を提案した[6),7)]．μ-actor は，Hewitt の actor[8),9)]の主な機能を実現した人工知能用モジュールである．本論では，μ-actor を各副題および共通変数に対して割り当てることにより，各副題が他の副題と独立して処理されることが可能であることを述べる．また，著者らが提案した AND/OR グラフを用いたDグラフ推論[10),11)]に，μ-actor にスケジューラを追加したシステムを適用し，並列処理のシミュレーションを行う．

2. μ-actor と AND/OR グラフ

μ-actor は，擬似日本語を用いた質問応答システムの知識表現の構造として使用され，柔軟で有用な，手続き的に埋め込まれた知識を形成している[6),7)]．本節では μ-actor の構造と動作を簡単に説明し，AND/OR グラフとの対応付けを行う．

2.1 μ-actor の構造と動作

μ-actor は **Fig. 1** に示すように，ボックスで表現し，最上部には μ-actor の名前を書く．図中，メッセージと返答の転送をそれぞれ「⇒」,「→」で示す．広義の意味では返答もメッセージである．

μ-actor の内部は次の3つの部分に分かれる．

（1） **制 御 部**：送られたメッセージと行動情報をパタン照合することにより μ-actor の行動を決定する．そして，その行動の結果が意図されたことを満足するかどうかチェックする．もし，満足すれば返答し，そうでなければ別の行動を決定しようとする．制御部は各 μ-actor に共通したものである．

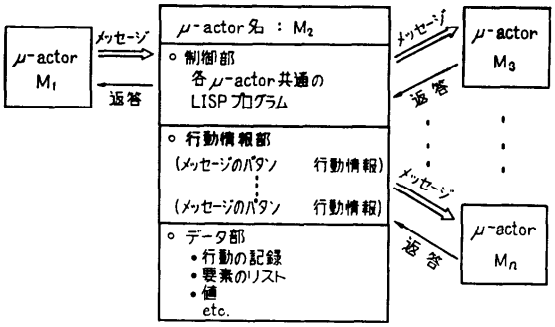


Fig. 1 A micro-actor model corresponding to a word.

（2） **行動情報部**：行動に関する情報を記述してある部分である．μ-actor は受け入れることのできるメッセージのパタンと，これに対応する行動を組にして行動情報部に記憶している．これは事柄や動作を定義する知識に対応している．

（3） **データ部**：データの記憶領域であり，事実を断言する知識に対応する．データ部はその μ-actor の固有値，属性，および行動中に得た情報で必要なものを記憶しておく部分である．

μ-actor においては，プロセジャとデータの構造には区別がなく，それぞれ半ば独立している物体（これを μ-actor と呼ぶ）として存在する．システム全体はこの μ-actor の集合である．各 μ-actor 間では，メッセージを転送するという一種類の行動のみが許され，他の交渉法（他の μ-actor の実行により起こる副作用等）は一切ない．システム全体の動作は，以下のように μ-actor の連鎖的な行動によって遂行される．μ-actor はメッセージが送られることにより動作を起こし，行動情報を参照しながらそれ自身の役割を果たそうとし，必要に応じてメッセージを他の μ-actor へ送る．そして，返答を待ち，返答が得られれば再び行動を続ける．行動が終了するとメッセージを送ってきた μ-actor へ返答を返す．

2.2 AND/OR グラフとの対応付け

問題解決を行う場合，その探索空間のモデル化として AND/OR グラフが用いられる．グラフの頂点と有向枝はそれぞれ問題および副題と操作子に対応している．一般に AND/OR グラフの探索においては，出発点から有向枝の方向に探索し，各頂点を経由して端点に至る道，または，解グラフを得る．AND/OR グラフ中の頂点および共通変数に対応する μ-actor をそれ

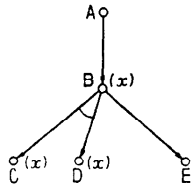


Fig. 2 An example of AND/OR graph.

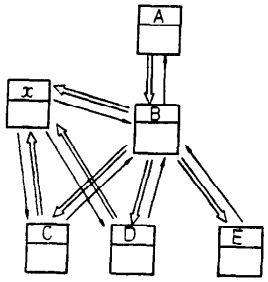


Fig. 3 Micro-actors corresponding to nodes on AND/OR graph in Fig. 2.

ぞれ頂点 μ-actor，変数 μ-actor と呼ぶことにする．これらの μ-actor を使用して **Fig. 2** に示す AND/OR グラフは **Fig. 3** のように表わせる．ここで，頂点 B, C, D が変数 x を共通しているものとする．変数 μ-actor はグラフの探索の際に作成されるが，詳細は後述する．μ-actor システムでは，グラフの探索はメッセージをつぎつぎ転送することにより行われる．メッセージが送られた端点では，その親頂点に返答を返す．返答もつぎつぎ転送され，出発点に必要な返答が返って来た時にグラフ探索を終了する．

以上のようにシステムを構成した場合，μ-actor はリスト処理用言語 LISP で遂行されるので，並列処理が不可能である．したがって，**Fig. 4**(a)のようにスケジューラを用いる．すべての頂点 μ-actor はスケジューラからメッセージを受け取り，スケジューラに返答を返すことにする．すなわち，頂点 μ-actor 間のメッセージの転送はすべてスケジューラを経て行われる．スケジューラはメッセージを first in first out で転送を行う．完全な擬似並列処理を行うためには，Fig. 4 (b) に示す構成になる．コントロールとしては，頂点 μ-actor はメッセージを送ってきた頂点 μ-actor に返答する必要がないので，頂点 μ-actor は擬似コルーチンの形を取る．そして，スケジューラによってメッセージ転送の同期を取ることにより，擬似並列処理を行うことができる．

Fig. 4 (a) に示す構成では，頂点 μ-actor が変数

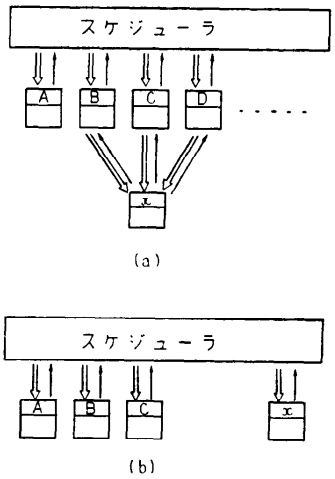


Fig. 4 Pseudo-parallel system.

μ-actor と直接会話を行う．この方法は，以下の理由により Fig. 4 (b) と等価である．本論では，Fig. 4 (a) に示す構成を用いる．

(1) 複数の頂点 μ-actor が同一の変数 μ-actor へメッセージを送る場合を考える．スケジューラはメッセージを first in first out で処理するため，最初に行動を起こした，すなわち，最初にメッセージを送られた頂点 μ-actor のメッセージが最初に変数 μ-actor へ送られる．したがって，頂点 μ-actor から直接変数 μ-actor へメッセージを送っても，メッセージが転送される順序は変わらない．また，同時に同一の変数 μ-actor へメッセージが送られることはない．

(2) メッセージ転送の制御はスケジューラが行っているが，もし，メッセージ転送の順序が変更されても，同じ結果が得られる構成になっている (**3.** 参照)．

(3) Fig. 4 (b) の構成にした場合，変数 μ-actor へ変数値を参照しようとしている頂点 μ-actor では，他の μ-actor から送られて来たメッセージを一時保管しておかなければならない．Fig. 4 (a) では，以上の操作を行う必要がないので，システムの構成は非常に簡単になる．

3. グラフの探索

前章で述べたように擬似並列処理でグラフの探索を行う場合，2種類のメッセージを用いる．1つは，端点を捜すためのもので，探索用メッセージと呼ぶことにする．他は，出発点へ向って送られるもので，バックトレイス用メッセージと呼ぶことにする．これら

は，Fig. 3 において，それぞれメッセージと返答に対応する．本章では，探索，変数 μ-actor，バックトレイスについて述べる．

3.1 探 索

探索は出発点 μ-actor からその子孫頂点 μ-actor へ次々と連鎖的にメッセージを送る動作である．頂点 μ-actor A から B へメッセージが送られたとする．メッセージは (%M SEARCH A NC) の形である．%M はフラグであり，拡張操作が行われるごとに新しく割り当てられる．M は数字で表わされる．フラグは後で述べるように，変数 μ-actor から値を得る時とバックトレイスする時には重要な働きをする．SEARCH はメッセージが探索用であることを示す．A はメッセージを送った頂点 μ-actor 名，NC は送られる引数である．頂点 μ-actor は，グラフ拡張の条件と，その条件が満足された時メッセージを送る頂点 μ-actor 名と引数を組にし，各頂点 μ-actor ごとに ELIST に記憶している．ELIST は **Fig. 5** に示す形で構成される．頂点 μ-actor は送られた引数が条件 i を満足するなら，その時行った操作を引数$_{ij}$ (j=1, …) に適用する．例えば，引数と条件$_i$ をパタン照合，または，単一化したならば，その時行った代入を引数$_{ij}$ に適用する．そして，その結果を頂点 μ-actor$_{ij}$ へ送る．その際，STAC にこの行動を記録しておく．代入を引数$_{ij}$ に適用した結果を SNC$_j$ とする．頂点名$_{ij}$ を SN$_j$ とし，対応するフラグを %N$_j$ と表わす．**Fig. 6** に，STAC の基本データを示す．STAC の第1番目の要素は，(%N$_k$ SN$_k$ SNC$_k$) (k=1, …) のリストで表わされる．この部分には，この STAC を持つ頂点 μ-actor B からメッセージを送った頂点 μ-actor 名とその内容を記憶している．第2番目には，B へメッセージを送った頂点 μ-actor A とその内容を記録する．最後の要素は，バックトレイスの際に付加される解グラフの部分木であり，基本データが作られた時は NIL にする．単一化，パタン照合の時，共通変数の扱いが問題になるが，次節で考察する．

3.2 変数 μ-actor

AND 関係にある頂点 μ-actor が共有している変数には同じ値が代入されなければならない．このことは，頂点 μ-actor の独立した行動の妨げとなる．そこで，転送される引数中に変数が存在する場合，それに対応する変数 μ-actor を作成し，固有の名前を付ける．**2.** で述べたように μ-actor はメッセージ転送以外，他の μ-actor と独立である．したがって，頂点 μ-actor は他と独立して処理が行える．しかも，複数の頂点 μ-actor が同一の変数 μ-actor を参照する場合には，各頂点 μ-actor の共通変数には同一の値が使用される．変数 μ-actor は以下で述べるように，参照される順番が変わっても結果は同様になる．

変数 μ-actor の行動は次の2種類である．(1)送られたデータをデータ部に追加する．(2)与えられたフラグを含むデータを返答する．変数 μ-actor のデータ部には，(F D) の形を基本データとして持つ．F はフラグの集合であり，F の要素は間接的に AND 関係である．D はその変数が持つ値である．ただし，値が定まっていない時は NIL である．変数 μ-actor のデータ部の変化を例を使用して説明する．

［例1］ **Fig. 7**(a)には各頂点 μ-actor の ELIST を示し，その関係を示す μ-actor システム図を **Fig. 7**(b)に示す．X は変数，A, B, C は定数を示す．TN は出発点，FN は端点を表わす．最初に，μ-actor TN から P へメッセージを送る．このとき，引数中に変数が存在するので，これに対応する変数 μ-actor X を作

((条件$_1$ 頂点名$_{11}$ 引数$_{11}$ 頂点名$_{12}$ 引数$_{12}$…)
(条件$_2$ 頂点名$_{21}$ 引数$_{21}$ 頂点名$_{22}$ 引数$_{22}$…)
……
)

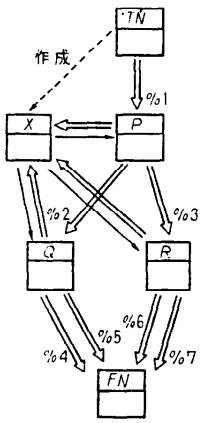
Fig. 5 Structure of ELIST.

(((%N$_1$ SN$_1$ SNC$_1$) (%N$_2$ SN$_2$ SNC$_2$)…)
((%M A NC) Tree)

Fig. 6 Structure of a primitive data of STAC.

TN-ELIST=((NIL *P* (*X*)))
P-ELIST=(((*X*) *Q* (*X*) *R* (*X*)))
Q-ELIST=(((A) *FN* NIL) ((B) *FN* NIL))
R-ELIST=(((B) *FN* NIL) ((C) *FN* NIL))

(a)



(b)

Fig. 7 ELISTs (a) and micro-actors (b) in example 1.

```
((%1) NIL)
    (a)
((%2 %3) NIL)
    (b)
((%4 %3) A)
((%5 %3) B)
    (c)
((%5 %6) B)
    (d)
((%2 %6) B)
((%2 %7) C)
    (e)
((%5 %6) B)
    (f)
```

Fig. 8 Data in variable micro-actor X.

成する（変数 μ-actor には固有の名前が付けられるが，ここでは分かりやすくするために，X と名付ける.）．メッセージのフラグを %1 とすると，X のデータ部は最初 **Fig. 8**(a) のようである．次に，P から Q と R にそれぞれメッセージを送る．この時のフラグをそれぞれ %2, %3 とする．このとき，%2 と %3 は AND 関係だから，X のデータ中の %1 を "%2 %3" に置き換える (**Fig. 8** (b))．次に，Q からの探索を行う場合を考える．Q は X にメッセージを送り %2 を含むデータを尋ねる．変数 X は，%2 において値が定まっていないので，任意の定数と照合することができる．ここでは，変数 X に A または B を代入する場合，FN へ到達する．したがって，フラグをそれぞれ %4, %5 とすると，データ中の %2 と置き換え Fig. 8 (c) のように構成する．その後に，R からの探索をする時，R は X に %3 を含むデータを尋ねる．X は Fig. 8 (c) の2つのデータを返すが，この内，ELIST 中の条件を満足するのは ((%5 %3) B) だけである．この操作のフラグを %6 とすると結局，X のデータは Fig. 8 (d) に示すようになり，X の値は B であることが分かる．

逆に，R からの探索を先に行った場合を考える．上と同様に，変数 X には B または C が代入できるため，それぞれのフラグを %6, %7 とすると，X のデータ部は Fig. 8 (e) に示すようになる．次に，Q からの探索をする時，B のみが ELIST の条件を満足する．この時のフラグを %5 とすると，X のデータ部は Fig. 8 (f) の示すとおりになる．この結果は，Q から先に探索した場合と同じである．

上で述べたように，1つの共通変数に1つの変数 μ-actor が対応している．ところが，複数の共通変数が同時に同じ変数と照合される場合が生じる．たとえば，$(X\ Y\ Z)$ と $(u\ u\ u)$ をパタン照合する場合である．ここで，X, Y, Z は変数 μ-actor，u は変数を表わす．このような場合までも変数 μ-actor によって処理するには，次の2通りが考えられる．

(1) u へ X, Y, Z のうち1つを代入する．この3つの変数 μ-actor のデータ部に，値が定まるまで，互いの変数 μ-actor 名を記憶しておく．そして，これらのデータ（フラグも含む）を変更する場合は，必ず他の変数 μ-actor へそれを知らせるメッセージを送ることにする．

(2) 新しい変数 μ-actor W を作り，u へ代入する．そして，この時のフラグで，X, Y, Z は W と，W は X, Y, Z と値が同じことを記憶しておく．(a) と同様，これらのデータを変更する場合は，X, Y, Z は W へ，W は X, Y, Z へ，そのことを知らせるメッセージを送る．

上の2つの方法を遂行するには，いずれの場合も変数 μ-actor に複雑な処理を行う能力が必要となる．すなわち，大きなプログラムが必要となり，その遂行時間も長くなる．また，変数 μ-actor は比較的数が多くなるのでこの方法はあまり好ましくない．そこで，上に述べた状態の場合，u には X, Y, Z のうち1つを代入しておき，X, Y, Z には何の情報も入れず，従来通り探索を行う．そして次節で述べるように，バックトレイスの際に簡単なパタン照合により正しい解を得る．

3.3 バックトレイス

前節で述べたように，簡単のため探索では共通変数の完全な処理は行わない．したがって，バックトレイスは以下のように行い．正しい代入を抽出する．バックトレイス用メッセージは (%N_0 BACKTRACE SN_0 SNC_0 $Tree_0$) の形をしている．BACKTRACE はこのメッセージがバックトレイス用であることを示している．%N_0, SN_0, SNC_0, $Tree_0$ はそれぞれフラグ，メッセージを送った頂点 μ-actor 名，その引数，その頂点より下にある解グラフの部分木を表わす．頂点 μ-actor は STAC (Fig. 6) の基本データすべてに対して次の操作を行う．第1要素の中に，(%N_0 SN_0 SNC_0) とパタン照合するものがあれば，第1要素の残り，および，第2要素にパタン照合の結果を適用し，Tree に $Tree_0$ を加える．もし，第1要素に何も残っていなければ，すなわち，AND 関係にある子頂点 μ-actor からのメッセージがすべて矛盾しなければ，第2, 3要素を参照して親頂点 μ-actor A へメッセージ (%M BACKTRACE B NC′ Tree′) を送る．B は自分の名

(((%2 *Q* (*X*)) (%3 *R* (*X*))) ((%1 *TN* (*X*))) NIL))
(a)
(((%2 *Q* (*X*)) (%3 *R* (*X*))) ((%1 *TN* (*X*))) NIL))
((%3 *R* (A))) ((%1 *TN* (A)) (*Q* (A) *FN*))
((%3 *R* (B))) ((%1 *TN* (B)) (*Q* (B) *FN*)))
(b)

Fig. 9 STAC of node micro-actor *P*.

前，NC′ は NC にパタン照合の結果を適用したもの，Tree′ は子頂点 μ-actor $SN_1, SN_2, \cdots$ から送られてきた部分木の集合に (B NC′) を加えたものである．

［例2］ 簡単のため，例1と同じ例を変数 μ-actor を使わずに，バックトレイスのみを使用した時の動作を示す．**Fig. 9** (a) には P から Q と R が拡張された時の P の STAC を示す．次に，Q が先に処理されたとすると，X には A または B が代入されるから，P には (%5 BACKTRACE Q (A) (Q (A) FN)) と (%2 BACKTRACE Q (B) (Q (B) FN)) が送られる．それぞれのメッセージについて，上で述べた処理を STAC に行うと，Fig. 9 (b) のようになる．次に，R が処理を行った後，P に送られるメッセージは，

(%3 BACKTRACE R (B) (R (B) FN)) と
(%3 BACKTRACE R (C) (R (C) FN))

である．これらのメッセージと，STAC を比べると，後者のメッセージと第3基本データがパタン照合する．したがって，TN にはメッセージ

(%1 BACKTRACE P (B) (P (B) ((R (B) FN) (Q (B) FN))))

が送られる．

逆に R から先に処理を行っても，同じ結果が得られる．

4. 定理証明システム

擬似並列処理システムをDグラフ推論に適用する．最初に，Dグラフ推論について簡単に説明する．Dグラフの各頂点は，述語およびその否定形，真，偽に対応しており，頂点間の関係は有向枝で表わされる．有向枝には，その始点と終点が表わす述語のそれぞれの n 変数を組にして，ラベル付けする．3つ以上の述語の関係を示す時は，有向枝を連係記号で結び AND 関係を作る．Dグラフ推論は，与えられたDグラフにおいて，真を表わす頂点 TN から偽を表わす頂点 FN へ到達可能であることを，頂点状態，有向枝のラベル条件を使用して調べる推論である．到達可能であることと，Dグラフが表わす節集合が充足不可能であることは等価であり[10]，また，Dグラフ中には最小解が存在することも証明されている[11]．詳細は文献 10), 11) を参照していただきたい．

Dグラフ推論を遂行する場合，探索には **3.1** で述べた操作以外に次の操作が必要である．

(1) **PR (局所反ばく)**：頂点 μ-actor B の否定形を表わす頂点 μ-actor ¬B が B の祖先であり，¬B の STAC 中に，B の引数に包含される第2要素の引数 NC を持つ基本データが存在するなら，PR が可能であるとして，バックトレイスを行う．文献 11) では，PR の際，単一化を行うが，これは最小解が得られることを論理的に保証するためである．ここでは簡単のためパタン照合を行うことにした．

単一化を行う場合は以下の操作をしなくてはならない．もし，NC に代入を行ったなら，¬B の親頂点 μ-actor へフラグと NC の代入結果 NC′ を送る．親頂点 μ-actor では，そのフラグにおいて ¬B と AND 関係にある頂点 μ-actor に NC′ を送り，処理を依頼する．

(2) **併 合**：AからBへメッセージ (%L SEARCH A ANC) が送られたとする．B の STAC 中に，第2要素中の NC が ANC を包含し，そのフラグが %L の祖先でないような基本データが存在すれば，その基本データをコピーし，その第2要素を ((%L A ANC)) に書き換え，STAC に追加する．そして，このメッセージに対する探索は行わない．

(3) **ループの判定**：(2)と同様なメッセージが B に送られたとする．STAC 中に，第2要素中の NC が ANC を包含し，そのフラグが %L の祖先であるような基本データが存在すれば，ループがあると判断し，このメッセージに関する探索を中止する．

その他，バックトレイス用メッセージ中の引数を ELIST に記録しておくことにより，同じ内容の再探索を防いだ．また，スケジューラでは，バックトレイス用メッセージを優先的に転送するように変更し，効率化した．

定理証明プログラムは，FACOM 230/45 S の Σ LISP* 上で遂行されており，文献 1), 2) の例題を解いている．入力データは，頂点 μ-actor 名とその ELIST を対にしたものである．プログラムは，データを参照して頂点 μ-actor を作成した後に，頂点 μ-actor TN から探索を始める．

5. むすび

変数を共有する問題をそれぞれ独立して処理を行う

* LISP 1.5 と同程度のインタプリタ．

ため，μ-actor を使用し，擬似並列処理システムを作成した．システム中，μ-actor は問題を示す頂点および変数に対応し，頂点 μ-actor 間のメッセージ転送はスケジューラにより制御される．頂点 μ-actor はメッセージ転送以外，他と独立に行動し，それらに共通な変数には変数 μ-actor と会話することにより同一の値が代入される．このシステムによりDグラフ推論を遂行した．

終りに，本研究に関して御討論，御助言していただいた大阪大学基礎工学部情報工学科田中研究室諸氏，特に北橋忠宏博士，大学院生横矢直和氏，木島裕二氏，難波秀彰氏，ならびに同学科電子計算機室工藤英夫技官に感謝する．

参考文献

1) N. J. Nilsson: Problem Solving Methods in Artificial Intelligence, p. 255, McGraw-Hill, Inc., New York (1971).
2) J. R. Slagle & D. A. Koniver: Finding Resolution Proofs and Using Duplicate Goals in AND/OR Trees, Information Science, Vol. 3, pp. 315～342 (1971).
3) G. Levi & F. Sirovich: Generalized And/Or Graphs, Artificial Intelligence, Vol. 7, pp. 243～259 (1976).
4) A. J. Nevins: A Relaxation to Splitting in an Automatic Theorem Prover, Artificial Intelligence, Vol. 6, pp. 25～39 (1975).
5) A. J. Nevins: Plane Geometry Theorem Proving Using Forward Chaining, Artificial Intelligence, Vol. 6, pp. 1～23 (1975).
6) 木島，小川，北橋，田中：自然言語処理における知識構造について，信学技報，AL 76-44, pp. 51～58 (1976).
7) H. Ogawa & K. Tanaka: A Structure For the Representation of Knowledge—A Proposal for A Micro-Actor—, 5th IJCAI.
8) C. Hewitt, P. Bishop & R. Steiger: A Universal Modular Actor Formalism for Artificial Intelligence, Proc. 3rd IJCAI, pp. 235～245 (1973).
9) C. Hewitt: Viewing Control Structures as Patterns of Passinge Messages, A. I. Lab. MIT, Working Paper 92 (1976).
10) 小川，北橋，田中：有向 AND/OR グラフを用いた形式的推論とその完全性，信学論 D, Vol. 60-D, No. 11, pp. 913～920 (1977).
11) 小川，北橋，田中：Dグラフ推論と推論過程の最小性，信学論 D, Vol. 60-D, No. 11, pp. 905～912 (1977).

(昭和52年6月30日受付)