解説B

非ノイマン型コンピュータ [II・完]

田中 英彦

田中英彦：正員　東京大学工学部電気工学科
Non-von Neumann Computers [II·finish]. By Hidehiko TANAKA, Member (Faculty of Engineering, The University of Tokyo,Tokyo).

前回では非ノイマン型コンピュータの意味，計算のモデルについて紹介し，更に具体的なマシンとしてデータフローマシンの機構と研究について解説した．

今回はその続きとして，リダクションマシンとその他の非ノイマン型マシンを取り上げ，それらの機構と研究状況について紹介する．

5. リダクションマシン

5.1 概　　要

リダクションマシンの提案は，1971 年の Klaus Berkling が最初[18]といわれている．すなわち，問題が式の形で与えられたとき，その部分的な文字列を次々と規則に従って書き換えてゆく．その書換えをリダクション (reduction) とよび，それを行ってゆくことを計算とみなすものである．

例えば図 10 にその一例を示す．四角で囲まれたものが定義（いわゆるプログラム）部分で，例えば b：(4) は，「b と名付けられたデータの値は 4 である」ことを表し，$i1$：(+b1) は，「b に 1 を加えた値を $i1$ とする」ことを表している．下方に（ ）で囲まれているのが解くべき問題である．a の値に対する要求がまず起り，定義内に求めにゆき，a の定義がコピーされて a と置き換えられる．次に，新しく置き換わった式 (…(*$i1$ c)…) を解こうとして $i1$，c に対する要求が生じ，またそれに対する置換が行われる．(*52) となったところで，5×2 の計算が実行され 10 に置き換わって終了する．

以上がリダクションマシンの原理であり，その操作

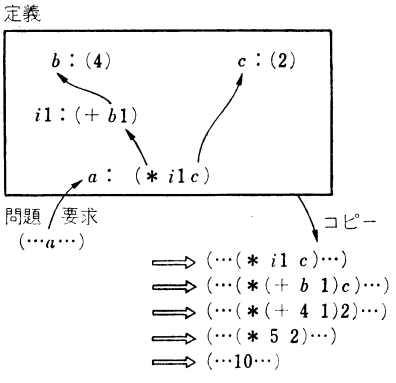


図 10　項書換えの例

は従って，置換え可能な部分（これをリデックス：redex とよぶ）の認識と，その部分の変換である．置換可能な部分が複数同時点で見出されたとき，どの順序でそれを実行するか，また，置換を実際にコピーして行うか，ポインタで済ますか等，幾つかの選択項目があり，それに従って幾つかのモデルが提案されている．

リダクションマシンは，その動作が直接的で理解しやすく，マシンと言語間のいわゆるセマンティックギャップが少ない．更に，文字列の各所で並列に書換えが可能で（図 10 では並列に実行している），マシンをうまく構成することにより von-Neumann ボトルネックを避け得る可能性がある．

リダクションマシンは元々，Lisp 等に代表される関数型言語を対象として考えられてきたが，述語論理に基づく論理型言語に対しても同種の適用が可能であり，幾つかの提案がある．

5.2 リダクションの戦略[19]

文字列があったとき，一般に複数のリデックスが存在するが,その内どれから書き換えるかが問題となる．これをリダクション戦略という．次々と書き換えていってリデックスが無くなったときに計算は終了するが，その最終形を正規形という．

戦略として考慮すべきことは，その正しさと効率である．すなわち，文字列が正規形を持てば，必ずそれに行きつくことができるものが望ましいし，また，正規形を求めるまでの書換え回数が最小でありたい．これらに対し今までにわかっていることは，次のとおりである．

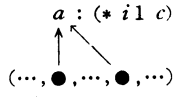
もし文字列が正規形を持てば：

- ●最外リダクションによって必ず正規形が得られる．
- ●書換え回数最小の方式は，コピー無し最外リダクションで実現できる．

すなわち，外側のリデックスを常に書き換える最外選択とし，書換えのときにコピーをせず，ポインタを用いれば書換え回数は減少する．例えば，前図 10 において，書き換えるべき項が，

$$(\cdots, a, \cdots, a, \cdots)$$

のような形をしているとき，単純に a の定義を文字列内にコピーすると，同じ操作を 2 箇所で重複して行うことになるが，ポインタを用いて



という構造を作れば，そのような重複は避けることができる（これをグラフリダクションとよび，前述のコピー方式をストリングリダクションとよぶ）．

5.3 関数型言語リダクションマシン

Lisp 等の関数型言語を，直接リダクションによって実行するマシンの研究例を表 3 に示す．これらの内の代表例として North Carolina Cellular Tree Machine[20] と，ALICE[21] について簡単に述べる．

まず前者であるが，提案されているマシンアーキテクチャは図 11 のようなものである．2 進木構造をしており，葉部分はメモリ，途中のノードは計算と通信をつかさどるプロセッサである．プログラムは J. Backus の提案した関数型言語 FP の記法にそって書かれ 1 列の文字ストリングで表されるが，各葉部分に 1 シンボルずつ（何重の括弧内のシンボルであるかを与えるネストの深さ情報と共に）置かれる．式内にはオペレータ部分とオペランド部分とがあり，それに対応した通信が木上でローカルに，また並列に実行されて式の書換えが進行する．

次に ALICE におけるプログラム（階乗計算）の例を図 12 に示す．これは HOPE とよばれる言語に基づいて記述したものであり，その意味は次のとおりである．第 1 行目は，Factorial とよぶ関数の宣言で，整

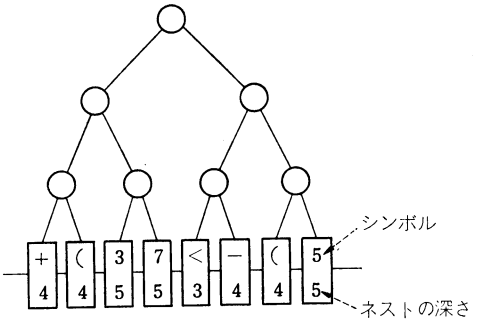


図 11 セルラートリーマシンの構造

表 3 関数型リダクションマシン研究例

マシン名	研究所	言語	特徴
GMD RM	GMD	関数型言語	ストリングリダクション，最外，1971 提案，1978 稼動
Newcastle-RM	U. of Newcastle Upon Tyne	関数型言語	ストリングリダクション，最内，線状構造，1980 提案
North Carolina Cellular Tree Machine	U. of North Carolina	FP	2 進木状セルラマシン，最内，1979 提案
AMPS	U. of Utah	FGL	グラフリダクション，最外，木構造，1978 提案
S-K RM	U. of Kent	Combinator	グラフリダクション，最左最外，1979 提案
SKIM RM	Cambidge U.	Combinator+α	グラフリダクション，最外，1980 発表，稼動中
ARM	NEC	Combinator	最内，データ駆動，1981 提案
ALICE	Imperial Colledge of London	ALICE CTL, HOPE	グラフリダクション，1981 発表，プロトタイプ作成中

RM : Reduction Machine

```
Factorial : integer→integer
  Factorial(n)⇐FactB(0,n)
FactB : integer×integer→integer
  FactB(i,i)⇐i
  FactB(i,i+1)⇐i+1
  FactB(i,j)⇐FactB(i,mid)*FactB(mid,j)
  where mid=Integer divide(i+j,2)
```

図 12 階乗を求めるプログラム

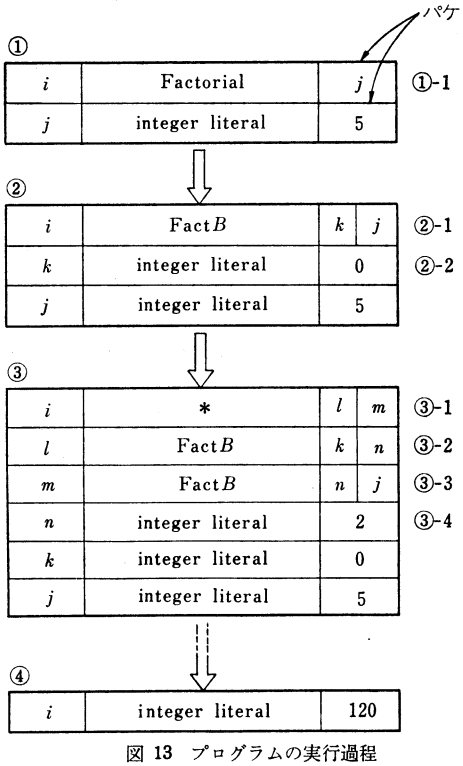


図 13 プログラムの実行過程

数を与えて整数を出力するものであることを述べている．2行目は置換を表し，Factorial (n) という形の式が与えられればそれを右辺の式で置換することの指示である．第6行目がこのアルゴリズムの中味であるが，Fact$B$ (i, j) という式が与えられ($i \neq j$, 且つ $j \neq i+1$)れば，i と j の中間にある値 mid を用いて，Fact$B$ (i, j) を右辺の二つの Fact$B$ の乗算に置換することを述べている．

これに対応する機械語のリダクション過程を図 13 に示す．① の内リデックスは①-1 であり，それが②-1，②-2 に置換され，次に②-1 がリデックスとなって，③-1～4 と変換される．そして最終的に，④ のような結果 (5!=120) が得られる．このようなグラフリダクションを実行するマシンアーキテクチャとして

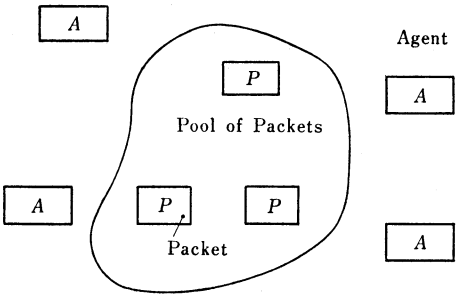


図 14 ALICE の概念アーキテクチャ

は，図 14 のようなものが考えられており，現在試作中である．図中，パケットプールには図 13 のようなパケットが置かれ，周りの各 agent は，リデックスとなっているパケットを取り出しては書換えを実行するプロセッサであり，各々独立にリデックスのチェックおよび書換えを行うことで並列に処理が進行する．

5.4 論理型リダクションマシン

1階述語論理に基礎を置く論理型プログラミング言語を直接アーキテクチャから支援するマシンが，ここで述べる論理型リダクションマシンである．この言語の代表は PROLOG であるが，それを用いて親子関係のプログラムを記述したのが図 15 である．ここに，① は，「X が Y の祖父（母）であることの条件は，X が Z の親であり，且つ Z が Y の親であることである」ことを記述したもの（parent と parent の間の，は AND を意味する），② は，taro が satoru の親である（③ 以下も同じ）ことを述べたものである．このようなプログラム定義が与えられたとき，質問「kazuo の祖父（母）はだれですか？」が生じたとする．これを ?-grand-parent (X, kazuo). と書くが，解を得るための手続きは，この質問と同じ述語名 (grand-parent) を持つ定義をプログラム内で捜し，この場合は，① において $X=X$, $Y=$kazuo とすれば，記号 :- の左辺は一致するから，その条件下で① を書き直せば，その右辺は

?-parent (X, Z), parent (Z, kazuo). ⑥

となる．すなわち，最初の質問に答える代りにこの質

```
grand-parent(X,Y) :- parent(X,Z),
                     parent(Z,Y).   ①
parent(taro, satoru).               ②
parent(taro, yoshiko).              ③
parent(yoshiko, kazuo).             ④
parent(jiro, kazuo).                ⑤
```

図 15 親子関係の PROLOG による記述

表 4　論理型リダクションマシンの研究例

マシン名	研究所	内容
PSI	ICOT	逐次型推論マシン，1983 稼動，OS (SIMPOS) 1984 稼動
PEK	神戸大	逐次型 Prolog マシン，1983 発表，1984 稼動
Pipelined Prolog Processor	SRI	パイプライン方式逐次型 Prolog マシン，1984 発表
AND/OR プロセスモデル	U. California Irvine	論理型プログラムの並列実行，1981 発表
PIE	東大	並列 Prolog マシン，1982 発表，Logic と Control の分離
PIM-R	ICOT＋日立	OR 並列 (Prolog)，AND 並列 (CP)，1983 発表，8×PE 試作中
OR-Parallel Token Machine	Royal Institute of Technology	OR 並列マシン，1983 発表
プロセスグラフモデル	電総研	データフロー制御による OR 並列処理，1983 発表
株分けモデル	富士通研＋ICOT	逐次マシンの並列実行，16×PE の試作済 (1984)
並列リダクションモデル	京大	OR 並列＋ストリーム並列，1984 発表

問に答えればよい．続いて，⑥ の前半の条件 parent (X, Z) を満たす節をプログラム内で捜し② からまず

X=taro, Z=satoru

が得られる．これを⑥ の後半の条件に合うか否か代入してみると parent (satoru, kazuo) となるが，この形の節はプログラム内に無いので，これは成立しないとみなす．そこで前に戻り，前半の条件を満たす別解を捜せば，③ より，X=taro, Z=yoshiko が得られる．これをまた後半の条件に代入すれば，parent (yoshiko, kazuo) となり，④ にそれが存在するので成立し，従って問題の解 X=taro (Z=yoshiko) が得られる．

以上のような処理が，PROLOG の処理であるが，これは，前の関数型と同様，質問文を次々と置き換えてゆく操作とみなすことができ，従ってリダクションマシンの一種である．もちろん PROLOG の処理系は，通常のノイマン型コンピュータで作ることができ広く使われているが，ここで取り上げたいのはそれらではなく，アーキテクチャから直接その実行を支援するようなマシンである．

このようなマシンの研究は，ここ数年始まったばかりであり実用に供されているものは PSI[27] 以外ほとんど存在しない．従って研究例のみを表 4 に示す．大きく分けると，逐次型マシンと並列マシンとになる．逐次型マシンは前例で述べたように，候補解を一つ一つ求めてはチェックすることを繰り返すもので，表では，PSI[27]，pipelined prolog processor[28]，PEK[29] 等がこれに相当する．アーキテクチャからみれば，タグ付の語形式を採用し，スタックアクセスの高速化，条件分岐機構の強化等が特徴的である．

並列マシンでは，OR 並列を中心とし，パイプライン制御を取り入れるものが多い．すなわち，前例でいうと，質問の parent (X, Z) とマッチするプログラム定義（複数存在する）のチェックを並列に実行することと，それから得られた条件（X=taro, Z=satoru 等）

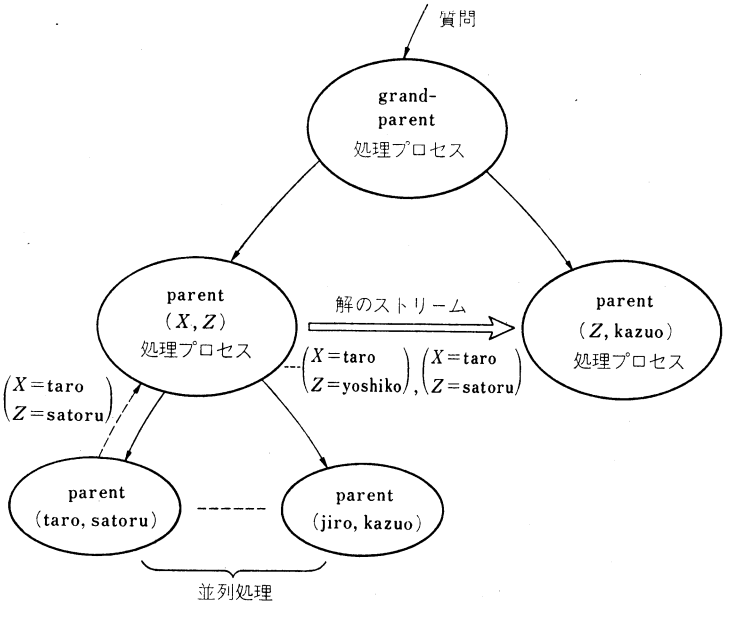


図 16　PROLOG の並列処理

を次々と直列に質問の後半処理（parent (Z, kazuo)）に渡して，次の処理をパイプライン処理すること等をアーキテクチャ上で支援することがその中心である（図 16）．

実際の並列処理モデルとしては，質問の記憶内表現形式，処理の単位，負荷分配手法等によってさまざまなモデルを考えることができ，表 4 の後半に示すような研究が行われている[80]~[86]．

6. その他のマシン

その他の非ノイマン型コンピュータと考えられるものとして，記憶装置の高機能化につながる次のようなアーキテクチャがある．

（1） タグアーキテクチャ

記憶データそれ自身に，データ型等の種別を付けるもので，命令とデータの独立性が高くなり，命令コードが単純になる．高級言語マシンでよく用いられ，前述の prolog マシンのみならず，Lisp マシンで広く用いられている．

（2） 動的記憶管理と構造記憶

プログラム処理中に動的に記憶セルを要求して複雑なデータ構造を作るような処理では，不必要になったデータの記憶セルを次々と回収して空きセルとし，新たなセル要求に備えるための処理[87]（ごみ集め：garbage collection）が大切である．これにより有限の記憶領域を無限に見せることができるが，この機能を記憶装置自体に持たせる試みがある．更に，構造データに対する基本的な処理（例えば，Lisp の car, cdr, cons）を記憶装置内に持たせ，プロセッサからはその指令だけを与えて結果を求める方式がある．このような記憶を構造記憶とよんでいる．

（3） 連想マシン[88]

更に記憶を高機能化し，記憶内容（の一部または全体）を与えてデータ（残りの内容とか存在）にアクセスする連想記憶とすることが考えられる．並列動作が基本なので高速性が期待できるほか，非数値処理に向いた機能であって，VLSI との整合性が高い．Goodyear Aerospace 社の連想プロセッサ STARAN は代表的なものであり，256 語×256 bit/語のメモリを 256 台のプロセッサ要素で並列処理する構造となっている．更に，データベースへの問合せ処理の大部分を連想記憶を用いて実現する提案[89] もあり，かなりの性能向上が期待されている．

（4） 意味ネットワークマシン

知識表現の 1 方法として，個々の事物をノードで表し，それらの間を種々のリンク（Is-a Instance-of 等）で結んだネットワークとして相互の関係を表現する手法がある．これを意味ネットワークというが，それを網状に結合された数多くのプロセッサ要素網で直接実現し，その網上に“処理要求”を走らせて知識処理を実現する試み[40] がある．10^6 台程の要素結合が想定されているが，これは前述の連想記憶と同じく，メモリとプロセッサを集中せず多数のノードに分散させることにより von-Neumann ボトルネックを避ける試みであり，VLSI に向いたアーキテクチャとして興味深い．

7. おわりに

コンピュータ構造はその出現以来，年々改良を受け現在の形となってきたが，基本構造はほとんど変っていない．これは，蓄積プログラム型逐次制御方式という非常に優れた枠組を全体として上回る方式がなかなか見つからないことも確かであるが，何としても現実的には従来型コンピュータ技術の完成度，膨大なソフトウェアの蓄積が大きな壁になっている．しかし，VLSI 技術の発達によってハードウェアはコストの制約から自由になり，種々の試みができる余裕が出てきた．また，関数型/論理型言語，データフローマシン，リダクションマシン等，新しい枠組がそろいつつある．Neumann 型コンピュータの限界を打ち破り，コンピュータ処理能力を飛躍的に向上させるためには，まだまだ多くの問題が残っている．今後の研究に期待したい．（完）

文　献

（1） G.J. Myers : “Advances in Computer Architecture”, John Wiley & Sons (1978).

（2） P.C. Treleaven, D.R. Brownbridge and R.P. Hopkins : “Data-driven and demand-driven computer architecture”, Comput. Surv. **14**, 1, pp. 93-143 (March 1982).

（3） 相磯秀夫，飯塚　肇，元岡　達，田中英彦：“計算機アーキテクチャ”，岩波講座（昭 57）．

（4） Arvind, et al. : “An asynchronous programming language and computing machine”, California Irvine TR 114 A (1978).

（5） Arvind, M.L. Dertouzos and R.A. Iannucci : “A multiprocessor emulation facility”, TR 302, LCS, mIT (Sept. 1983).

（6） W.B. Ackerman and J.B. Dennis : “VAL-a voclue oriented algorithmic language, preliminary reference manual”, CSG, TR-218, MIT (June 1979).

（7） J.B. Dennis and D.P. Misunas : “A computer

architecture for highly parallel signal processing", Proc. NCC, AFIPS, pp. 402-409 (1974).
(8) C.L. Hankin, P.E. Osman and J.A. Sharp : "A data flow model of computation", Dept. of CS, West-field College (Univ. of London) (March 1978).
(9) 雨宮，長谷川，小野："データフローマシン用高級言語 Valid"，研実報，**33**, 6, pp. 1167-1181 (昭 59-06).
(10) N. Takahashi and M. Amamiya : "A data flow processor array system-design and analysis", Proc. 10th Intl. Symp. Computer Architecture, pp. 243-250 (June 1983).
(11) 雨宮，長谷川："リスト構造を並列処理するデータフローマシンの方式とその言語"，日経エレクトロニクス，pp. 219-258 (昭 59-10-08).
(12) Y. Yamaguchi, K. Tada and T. Yuba : "A performance evaluation of a lisp-based data driven machine (EM-3)", Proc. 10th Intl. Symp. Computer Architecture, pp. 363-369 (June 1983).
(13) 弓場，山口，島田："データ駆動マシン用 Lisp とその中間言語"，第7回情処学全大，7D-7 (昭 57-03).
(14) T. Shimada, K. Hiraki and K. Nishida : "An architecture of a data flow machine and its evaluation", COMPCON '84 Spring, pp. 486-490 (Feb. 1984).
(15) T. Suzuki, K. Kurihara, H. Tanaka and T. Moto-Oka : "Procedure level data flow processing on dynamic structure multimicroprocessors", JIP, **5**, 1, pp. 11-16 (March 1982).
(16) M. Kishi, H. Yasuhara and Y. Kawamura : "DDDP: A distributed data driven processor", Proc. 10th Intl. Symp. Computer Architecture, pp. 236-242 (June 1983).
(17) 伊藤，尾内，益田，清水："データフロー方式の Prolog Machine", Proc. Logic Programming Conference '83 (March 1983).
(18) K.J. Berkling : "A computing machine based on tree structures", IEEE Trans. Comput., **C-20**, 4, pp. 404-418 (Jan. 1971).
(19) 二木，外出："項書き換え型計算モデルとその応用"，情報処理，**24**, 2, pp. 133-146 (昭 58-02).
(20) G.A. Mago : "A cellular computer architecture for functional programming", Proc. IEEE COMPCON, 80, pp. 179-187 (Feb. 1980).
(21) J. Darlington and M. Reeve : "ALICE multiprocessor reduction machine for the parallel evaluation of applicative languages", Proc. 1981 Conf. on Functional Programming Language and Machine Architecture.
(22) P.C. Treleaven and G.F. Mole : "A multi-processor reduetion machine for user-defined reduction languages", Proc. 1980 Symp. on Computer Architecture (May 1980).
(23) R.M. Keller, et al. : "A loosely coupled applicative multiprocessing system", AFIPS, NCC, pp. 861-870 (1978).
(24) D.A. Turner : "A new implementation technique for applicative languages", Soft. Practice and Experience, 9, pp. 31-49 (Sept. 1979).
(25) T.J.W. Clark, et al. : "SKIM-the S.K.I reduction machine", Proc. LISP-80 Conf., pp. 128-135 (Aug. 1980).
(26) 小長谷，山本："リダクションマシンの構想について"，信学技報，**EC 81-33** (1981).
(27) K. Taki, M. Yokota, et al. : "Hardware design and implementation of the personal sequential inference machine (PSI)", Proc. Intl. Conf. on Fifth Generation Computer Systems (FGCS), pp. 398-409 (Nov. 1984).
(28) E. Tick and D.H.D. Warren : "Toward a pipelined prolog processor", 1984 Intl. Symp. on Logic Programming (Feb. 1984).
(29) N. Tamura, K. Wada, et al. : "Sequential prolog machine PEK", Proc. Intl. FGCS, pp. 542-550 (Nov. 1984).
(30) J.S. Conery : "The AND/OR process model for parallel interpretation of logic programs", UC Irvine, TR-204 (June 1983).
(31) T. Moto-Oka, H. Tanaka, et al. : "The architicture of a parallel inference engine-PIE", Proc. Intl. FGCS, pp. 479-488 (Nov. 1984).
(32) S. Haridi and A. Ciepielewski : "An OR-parallel token machine", Royal Inst. Tech., TRITA-CS-8303 (1983).
(33) 尾内，麻生，ほか："並列推論マシン PIM-R，アーキテクチャとソフトウェアシミュレーション"，ICOT-TR-077 (昭 59).
(34) 久門，板敷，ほか："並列推論処理システム一株分け方式"，第30回情処学全大 (昭 60-03).
(35) 梅山："論理プログラムの為の OR 並列モデル"，Proc. Logic Programming Conf. '83 (1983).
(36) 八田，柴山，萩原："並列リダクションモデルに基づく Prolog マシンのハードウェア構成"，第30回情処学全大 (昭 60-03).
(37) 日比野："ガーベッジコレクションとそのハードウェア"，情報処理，**23**, 8, pp. 730-741 (昭 57-08).
(38) 市川，上林："連想プロセッサ"，信学誌，**64**, 6～7, pp. 606-614, pp. 736-742 (昭 56-06～07).
(39) E.J. Oliver, et al. : "RELACS; A relational associative computer system", Proc. 5th Workshop on Computer Architecture for Non-Numeric Processing, pp. 108-114 (1980).
(40) D.W. Hillis : "The connection machine—computer architecture for the new wave", MIT AI-Memo, No. 646 (1981).
(41) 曽和，上村："試作データフローコンピュータのトークンメモリの構造と連想処理"，信学技報，**EC 83-17** (1983).
(42) 中西："LISP入門—システムとプログラミング"，第2版，近代科学社 (昭 55).
(43) E.Y. Shapiro : "A subset of concurrent prolog and its interpreter", ICOT Technical Report TR-003 (1983).