(3) 新世代コンピュータのアーキテクチャ

① Concurrent Data Access Architecture

H. Diel (IBM 西独)

VLSI の出現により，より多くの機能をひとつのチップに格納することが可能となった。その結果，計算機設計者にとって，機能のひとつひとつの効率化でなく，複数のプロセッサを並列に稼動させることが重要な問題となった。

CDAA（コンカレント・データ・アクセス・アーキテクチャ）は，高度な並列処理を実現することを目的として設計されたものである。この点においては，CDAA は，近年研究されているデータフローマシン等と同じであるが，CDAAでは単に並列処理を行なうだけではなく並列性を有効に生かし得る応用分野をも対象としている。すなわち，人工知能の分野に対しては back tracking，transaction処理においては BACKOUT /COMMIT 等の機能の実現である。CDAA は，また，従来の計算機の延長線上にあるという特徴を持ち，従来の計算機のために書かれたプログラムをそのまま実行することが可能である。

② Knowledge-based VLSI Routing System — WIREX —

森　啓、光本　圭子、藤田　友之、後藤　敏（日本電気）

AI 手法を用いたVLSI の自動ルーティングシステムを提案する。これまでのシステムは，専門家の助けを必要とするインタラクティブなものであった。しかし，これでは設計に時間がかかる上，誤りも生じやすい。そこで，専門家の知識を規則として知識ベースにたくわえ，従来のＣＡＤシステムと組み合わせることにより，全く自動的にルーティングを行なうシステムを構築した。このシステムは数千ゲートの回路に対して効果が認められた。

システムは従来のＣＡＤデータベースと，専門家の知識を Prolog のプログラムとしてたくわえている知識ベースのふたつのデータベースを持つ。Prolog のインタプリタが与えられた問題に対し，知識ベースの規則に基づいて推論を行ない，ＣＡＤデータベースの手続きを呼び出す。現在，知識ベース内の規則は約 30 であり，この規則を増やすことにより，複雑な回路のルーティングを行なうことができるであろう。

③ Sword 32: A Bytecode Emulating Microprocessor for Object-Oriented Language

鈴木　則久、久保田光一、青木　孝（東京大学）

Sword 32 はSmall talk 80のバイトコードをエミュレートする32ビットのマイクロプロセッサである。

十分なオブジェクトを保持するため，オブジェクトポインタは 32 ビットとした 。ガーベジコレクションにはDeutsch －Bobrow のトランザクション型ＧＣを用いた。メソッド探索を高速化する為にメリッドキャッシュを設け，このキャッシュエントリには，クラス，セレクタ，命令（バイトコード又はマイクロコード）のアドレス，メソッドのアドレスを含ませた。手続き呼び出しの高速化の為のハードウェア支援としては，128 語のスタックと 32 語の汎用レジスタを用意した。これによって最大８個のコンテクストまでスタック上に積める。バイトコードのディスパッチ高速化のために，バイトコードのフェッチ，プロセススイッチ，バイトコードディスパッチテーブルの動的切換えを自動的に行なっている。マイクロコードは水平型で，Alto で用いられている様な，２ステージのパイプラインから成る。

Swordチップは現在，VLSI化の最中であるが，その論理設計やマイクロプログラミングはLISPを用いて短期間の内に完了した。予想性能は 1.4 Mバイトコード／秒であり，Dorado の約５倍の速度である。

④ Hardware Design and Implementation of the Personal Sequential Inference Machine (PSI)

瀧　和男、横田　実、山本　明、西川　宏、内田　俊一 (ICOT)
中島　浩、三石　彰純（三菱電機）

パーソナル逐次型推論マシンPSIは，第5世代コンピュータプロジェクトにおいて，ソフトウェア及びハードウェアの開発のためのツールとして用いられる計算機である。

PSIの機械命令は，論理型プログラミングに基づく核言語KL 0と同レベルに設定されており，その主な処理であるユニフィケーションとバックトラックは，ファームウェアのインタプリタが専用ハードウェアの支援を受けて実行する。このようなハードウェアとして，ディスパッチメモリを用いたマイクロ命令の多重分岐機構，ワークファイルと呼ばれる40ビット×1Kワードの高速ローカルメモリなどがある。ワークファイルの一部はローカルフレームバッファとして用いられ，末尾再帰呼び出しの最適化の効率的な実行が可能である。

PSIの主記憶は，従来のDEC－10 Prologの64倍にあたる16Mワードの最大容量を持ち，大きな応用プログラムの実行を可能にしている。キャッシュメモリは，スタック操作が効率良く行なうことができるように特別に考慮されている。

このほかPSIには，ビットマップディスプレイ，マウスなどの対話性の高い入出力デバイスや，ローカルネットワークへのインターフェースなどが用意されている。

PSIの実装には，入手の容易な高速ショットキーTTLとMOSメモリが用いられ，マシンサイクルは200 nsecである。12枚のCPU基板，最大16枚までのメモリ基板，入出力デバイスのコントローラ，ウィンチエスタディスクドライブ，フロッピィディスクドライブなどが1台の筐体に収められている。

⑤ A Microprogrammed Interpreter for the Personal Sequential Inference Machine

横田　実、山本　明、瀧　和男、西川　宏、内田　俊一 (ICOT)
中島　克人(三菱電機)、三井　正樹(沖電気)

パーソナル逐次型推論マシンPSIは，核言語KL0と同レベルの高水準機械命令を持ち，これをマイクロプログラムで解釈実行する。PSIのマイクロインタプリタは，論理型プログラムの基本的な実行を行なうカーネル，種々の組み込み述語，プロセス管理やメモリ管理などを行なうOSインターフエースより構成される。

カーネル部の基本実行メカニズムはDEC－10 Prologに基づくものだが，実用価値の高いデータ型の追加，実行制御機構の拡張などがなされている。また，手続き呼び出し時に引数をあらかじめ評価しスタック上にコピーしておく引数コピー方式を用い，末尾再帰呼び出しの最適化を可能にしている。

組み込み述語の中には，リモートカット，バインドフックなどの拡張制御機構や，割込処理，メモリ管理などのOS支援機構などが含まれている。

このようにして，PSIのマイクロインタプリタは実行速度30KLIPSの高い性能と，良好なOS支援機能を提供する。カーネル部のマイクロプログラムは約1.5Kステップである。また，約160の組み込み述語が実装され，そのマイクロプログラムは約10Kステップである。

⑥ Design and Implementation of the Relational Database Engine

酒井　浩、岩田　和秀、神谷　茂雄、安部　公朗、田中　哲男(東芝)
柴山　茂樹、村上　国男 (ICOT)

知識ベースマシンの研究に用いる目的で，関係データベースマシンDeltaの開発が進められている。

Deltaは，ローカルエリアネットワークとのインタフエースを行なうインタフエースプロセッサ（IP），コンカレンシー制御などのデータベース管理を行なう制御プロセッサ（CP），関係データベース処理を行なう中心要素である関係データベースエンジン(RDBE)，信頼性等を向上させるメインテナンスプロセッサ（MP），リレーションの格納，取り出し，クラスタリングなどを行なう階層メモリ（HM）より構成される。

主要処理要素であるRDBEは，関係演算，ソート処理，集合演算などのコマンドを受けて，ソータ，マージャ等の専用ハードウェアと，汎用プロセッサにより処理を行なう。ソータは12段のソーティングセルと，結果が正しいかを調べるソーティングチェッカより成り，4096個の長さ16バイトのタプルを43 msecでソートすることができる。マージャはコマンドにより各種の動作モードを指定でき，柔軟なマージ処理が可能である。汎用プロセッサは，複雑な条件によるセレクション，算術演算などの処理を行なう。コマンドをコンパイルすることにより，処理能力の向上が計られている。

⑦ Query Processing Flow on RDBM Delta's Functionally-Distributed Architecture

柴山　茂樹、角田　健男、宮崎　収兄、横田　治夫、村上　国男 (ICOT)

関係データベースマシンDeltaは，インタフエースプロセッサＩＰ，制御プロセッサCP，関係データベースエンジンRDBE，階層メモリHM，メンテナンスプロセッサＭＰより成る機能分散型マルチプロセッサシステムである。ＩＰ，CP，RDBE，MP には同一の汎用ミニコンピュータが用いられ,相互にＩＥＥＥ－488 バスで結合される。HMのコントローラには，これより大きな汎用 プロセッサが用いられ，他のプロセッサと高速チャネルで結合される。

ＩＰは，ローカルエリアネットワークよりホストからのコマンドを受けとり，ＣＰに送る。ＣＰでは，トランザクションごとにコマンドを処理するタスクが作られ，ＲＤＢＥ及びＨＭに対するサブコマンドの生成と送信が行なわれる。ＣＰではこのほか，ディクショナリ及びディレクトリの管理，資源の２フェーズロックによるコンカレンシー制御などが行なわれる。ＲＤＢＥでは比較的単純な関係データベース処理が専用ハードウェアにより高速に実行され，複雑なものは汎用プロセッサで処理される。ＨＭではディスクの制御，データのキャッシング等が行なわれる。

⑧ LPS Algorithms

A. Lowry, S. Taylor, S.J. Stolfo (Columbia Univ., 米国)

DADO のような二進木構造を持つ並列計算機上でＬＰＳ アルゴリズムを用いれば，論理型プログラミング言語を実行することができる。

LPS アルゴリズムは制限されたAND/OR 並列モデルに属している。推論の探索木をできるだけ効率良く小さくまとめるように規則を適用していく。実行はまず抽象証明手続き （Abstract Proof Procedure）なるものを考える。APP ではゴールリテラルを保持，削除，拡張の３通りに処理し，またその処理は単一化（Unification），ジョイン，変数の置換えの３フェーズから成る。

LPS を DADO 上で実行するには，ある部分木の根を制御プロセッサ（CP）とし,その下に位置する処理エレメント（ＰＥ）間の通信を管理する。PE には事実，規則のヘッド，束縛集合（Binding Set）があり，単一化，ジョイン，部分解の生成を行なう。CP には規則のボディがあり，変数の置換，束縛を行なう。

処理速度の向上のために，LPS アルゴリズムの実装に際しては，いくつかの工夫が必要である。その主なものには非同期/同期単一化や，ジョインの実行順序にヒューリスティクスを導入したり，計算機の高並列性を利用して，部分的，分散的にジョインを各計算機に行なわせたり，ゴールリテラルを保持させておく必要等がある。これらをどの程度取り入れるかはオーバーヘッドとトレードオフであり，今後の課題である。

⑨ Performance Estimates for the DADO Machine: A Comparison of TREAT and RETE

D.P. Miranker (Columbia Univ. 米国)

DADO は VLSI マイクロプロセッサを多数，木構造状に接続した高並列計算機であり,プロダクションシステム（PS）の高速実行を目的としている。

PS において，解を少しずつ組み上げて行くのを temporal redundancy（TR）と言い，そうでないものを non TR と言うが，元々の DADO アルゴリズムは non TR に適したものだった。これは各々のプロセッシングエレメント（PE）を3種類のモードに設定して PS を実現するものである。

これに対して，TREAT アルゴリズムでは，まずプロダクションルール（PR）の条件部分を識別ネットワークに翻訳し，そのネットワークにトークンを流して解を生成していく。しかし,途中の状態をすべて記憶する必要が有り，メモリ更新のオーバーヘッドが大きい。

TREAT アルゴリズムでは，RETE と異なり，ルールを満たす条件のみを DADO ツリー中に分散格納する。処理が進行する上で必要な情報は，数多くあるＰＥによって即座に再計算される。この TRE－AT は TR，non TR ともに実現できる。

8ビットプロセッサを 1023 個用いる DADO2 プロットタイプでの計算速度は，RETE の場合が 67 プロダクションサイクル／秒，TREAT が 85 プロダクションサイクル／秒であると予想される。

⑩ Systolic Programming: A Paradigm of Parallel Processing

E. Shapiro (Weizmann Inst. of Science, イスラエル)

脈動アレイが大規模な平行処理に対して有効であることが知られている。これは，均一かつ単純な接続形式を持っているので，拡大性が良いからである。これを，計算機を無限に配置した平面上で考えることにより，より一般的な脈動プログラミングを得ることができる。

無限処理平面は，格子状に配置した計算機群上で，たたみ込みを模倣する事によって実現され，これらの計算機はトーラス上をなす。

この平面では，プログラムの中のデータ構造（配列，ベクタ，木）は，そのまま計算機間の接続として表われる。

処理平面上の計算を記述するために，コンカレント・プロローグに配置表式を導入したものを用いることができる。配置表式は，@ forward 60，@left のようなタートル形式となっており，プログラムの意味とは独立であり，効率のみに影響する。プロセスを配置していく産卵相と脈動アレイとして働く脈動相の2相がプログラムを構成する。

これらの記述による計算機平面の視覚化は，デバックの道具として有効であることがわかった。またこの視覚化により，より洗練されたプログラムを書くことが可能である。ここにあげるH－木，アレクー木などがその例である。この言語のインタプリンタをプロローグで記述し，行列の積，ガウスの消去法などを実際に実現した。

⑪ Restricted AND-Parallelism

D. DeGroot (IBMワトソン研究所、米国)

論理型プログラミング言語の AND 並列処理とは，実行すべきクローズ中のサブゴールを並列に処理することを言う。サブゴールが共有変数を持っているときに，何の制限を付け加えずにこれらのサブゴールを実行したならば，これらのサブゴールが同一の変数に異なった値を結合してしまうという問題が起こる。これに対する解決方法としては一般に2つの方法がある。

ひとつは，変数に何らかの記号をつけることによって，どのサブゴールが変数に値を結合すべきかを示すものである。他の方法は，何の記号も用いずに，動的に各サブゴールの変数の結合状況を調べることによって，サブゴールの実行順序を制御するものである。

後者の方法は，動的な制御を必要とするため大きなオーバーヘッドとなりかねない。従って，コンパイル時にできる限りの前処理を行ない，実行時のオーバーヘッドを軽減することが重要である。本論文ではそのようなひとつの処理方式を示す。この方式では，時として AND 並列に実行可能な可能性を見落とすことがあるが，それは無害なものである。本方式における動的なオーバーヘッドは小さく有効であると考えられる。

⑫ The Architecture of a Parallel Inference Engine —PIE—

元岡　達、田中　英彦、相田　仁、平田　圭二、丸山　勉（東京大学）

本論文では論理型プログラムを高並列に実行するマシンPIE（Parallel Inference Engine）のアーキテクチャとその性能評価を報告する。

PIEでは論理式の導出に忠実なゴール書き換え操作に基づくOR並列処理方式を採用している。並列処理は，推論ユニット（Inference Unit：IU）間で独立性の高いゴールフレーム（Goal Frame：GF）を次々と受け渡しながら進む。IUは定義節メモリ（DM），単一化プロセッセ（UP），メモリモジュール（MM），アクティビティコントローラ（AC）の4つのモジュールから構成される。PIEは，二段のネットワークを用いてIUを数百台結びつける。また，論理型プログラムを効率よく実行するために，様々なメカニズム（ゴール選択，ガードおよびNOT，負荷分散，構造メモリなど）が導入されている。

単一化プロセッセをTTL，ICで構成し，その結果，単一化に必要な時間は1セル当り5～7ステップであることが判った。また，ソフトウェア，シミュレータによりIU256台ではIU1台の約170倍の速度を得ている。今後の拡張として，重復ゴールの除去および集合表現が述べられている。

⑬ A Relational Dataflow Database Machine Based on Hierarchical Ring Network

J.I. Kim, S.R. Maeng, J.W. Cho (KAIST, 韓国)

本論文は，HRDM（Hierarchical Ring−based Relational Dataflow Database Machine）と呼ばれる並列関係データベースのアーキテクチャの提案とその性能評価を報告している。HRDMは一台のBEC（Back End Coutroller）と複数台のEDM（Elementary Database Machine）とS−ringと呼ばれるリングバスで結合した構成をとり，一方，各EDMはCCD又は磁気バブルメモリを用いたMU（Memory Unit）を選択演算等を実行するFP（Filter Processor）をペアとしたモジュール，及びJOINを行なうモジュールを複数台L−ringと呼ばれるリングバスで結合した構成となっている。JOINはJ−ringと呼ばれるリングバスによって結合された複数台のEJP（Elementary Join Processor）によって並列に実行される。マシンの実行制御はデータフロー制御に基づいており，その性能評価の結果，リングバス上のデータ転送負荷の点からリレーションの格納方法としていわゆるAttribute-basedな技法が有効とされている。

⑭ ASSIP-T. A Theorem Proving Machine

W. Dilger, H-A, Schneider (Univ. of Kaiserslautern, 西独)

ＡＳＳＩＰ－Ｔは，一階述語論理の定理の証明を行なう連想プロセッサである。定理証明の方法は，Coxらの，演繹計画法（deduction plan method ）を用いる。この方法では，演繹部と同一化部を別々に行ない，かつ知的バックトラックを実現することができる。

演繹部では，符号の異なる節を結びつけ演繹計画を作る。この結びつきを辺という。全ての節が辺で結ばれ（close），辺で結ばれた節が同一化可能であるとき（correct），証明が完了する。

この段階で辺について，２つのリテラルよりなる束縛項（constraint）ができる。この束縛項と，節の部分式の集合に対して同一化を行なう。（UwC－Unification with Constraint）同一化は，さらに変型段と整列段よりなる。変型段では，式に対して，束縛項を適用することにより，その部分式の束縛項を導く。整列段では，式の包含関係により方向をもつ矢印をひく。このようにしてできた，束縛項から来る双方向性のある矢印と整列段の矢印により，ループがある時，または，異なる述語に対する束縛項がある時は同一化は失敗する。

このような失敗に対して，通常演繹計画の変更を必要とするが，述語の衝突（ＡＴＴＡＣＨ）とループの集合より，最小衝突集合（ｍｃｓ）を作り，それから必要なバックトラックの情報をあつめる事ができる。（知的バックトラック）

ＡＳＳＩＰ－Ｔは，演繹部と，同一化部それぞれ別なデータ構造（演繹計画とＵｗＣ）を持ち，それぞれ別の連想メモリ（ＡＭ１，ＡＭ２）を構成する。この連想メモリは高度なデータを保持するので，連想プロセッサにより近い。

ここでは，具体的なデータ構造と，ＡＳＳＩＰ－Ｔの構性，アルゴリズムについて提示する。

⑮ Parallel Execution of Logic Programs based on Dataflow Concept

長谷川隆三、雨宮　真人（電電公社武蔵野通信研究所）

本論文では，データフローの概念に基づいた論理型言語の新しい並列処理方式について述べる。この処理方式は証明木に忠実に論理型言語の実行を行なうものある。この処理方式は，non−strict な構造データを扱うことのできるリスト処理向きデータフローマシンに実装することを仮定している。

本方式では，eager evaluation と lazy evaluation を用いて効率的なOR並列処理およびAND パイプライン処理を実現している。eager evaluation は，ORプロセスの積極的な導出を行なうために用いられ，lazy evaluation は，ORプロセスの数の爆発的な増加を制御するために用いられている。

ハードウェア資源の量にあわせて，ORプロセス数の制御を行なうために，lazy evaluation とカウンタが用いられる。動的にカウンタの制御ストラテジを変えることによって，探索方法を depth−first および breadth−first と使いわけることが可能である。

既に，本方式のインタプリンタがValid で書かれており，現在開発中のデータフローマシンOFM上に実装される予定である。

⑯ A Data-Driven Model for Parallel Interpretation of Logic Programs

L. Bic (Univ. of California, Irvine, 米国)

本論文では，論理型プログラムを並列処理するモデルについて報告している。このモデルは，データフローマシンの原理に基づき，論理型プログラムの実行をグラフ・テンプレートのパターン・マッチングにより行なう。この実行プロセスは，トークンが非同期的にグラフを動くことにより行なわれる。数多くのプロセッサが，独立性の高いトークンの送信および返信を行なうため，高い並列度が達成できるとしている。

このようなモデルで問題となることは，グラフ，テンプレートを，独立なトークン上にどのように分配すればよいか，そしてこのトークンをどのようにしてグラフ内で動かせばよいかということである。

この問題に対して本論文では，与えられたグラフ・テンプレートを動的に伸長する木構造（spanning tree）に変換することにより，OR並列とAND並列が実行可能であるとしている。
またアクティビティ・ネームのキュエリーなどを並列に処理することが可能であり，高いスループットが得られる。

⑰ EM-3: A Lisp-Based Data-Driven Machine

山口　喜教、戸田　賢二、弓場　敏嗣（電子技術総合研究所）、J.Herath（慶応義塾大学）

LISPベースのデータ駆動型計算機EM－3の試作を行なった。制御機構は，関数評価用に自然な拡張がなされており，パケット通信アーキテクチャにより実装されている。

試作機は，複数の68000を用いたプロセッサエレメントよりなり，それを通信ネットワークにより接続する。ネットワークは，BI－CMOSゲートアレイによるLSIルーターよりなる。

この機械の特徴として，(1)マルチ・プロセッサー実装　(2)パケット記憶用専用ハードウェア　(3)ゲートアレイLSIによる，4 by 4ルーター　(4)関数評価の為の制御機構　(5)分散リスト構造　(6)LISPに似たデータ駆動型言語などがある。

パケット記憶は，256bit長で4096個ある。うち32 bitはハッシュ表に使われる。パケット記憶制御は，ビット・スライス・マイクロプロセッサによる。

並列性は，LISPの引数の同時評価という形で自然に行なわれる。これに，擬似結果を導入することにより，データの発火の条件をゆるめることができる。また擬似結果を含む構造体に関数を適用することもでき，この結果を部分結果と言う。これは，リダクションの機構と似ている。

データ駆動言語EMLISPは，次のような設計思想がある。1）純LISPである。2）大域変数，自由変数の禁止　3）単一代入　4）既に存在するリストの置き換えの禁止　5）ループの禁止

ここでは，n－queen，quicksort，fibonacci数列の3つについて，リストウェア・シミュレーションを行なった。

⑱ The Transputer Implementation of Occam

D. May, R. Shepherd (Inmos, 英国)

トランスピュータは，一対一の通信路により結ばれたプログラマブルVLSI 素子である。Occam は，トランスピュータをチャネルによって通信するプロセスとして記述することができる。

Occam は，代入，チャネル入出力の３つの基本プロセスと，それらを結合する３種のコントラクター，SEQ ，PAR ，ALT よりなる，チャネルは，一対一であり，同期通信を行う。

それぞれのトランスピュータは，６つのレジスタと内部メモリを持つ。６つのレジスタは，作業領域へのポインタ，命令ポインタ，オペランドを形成するためのオペランドレジスタ，評価スタックをなすA，B，Cの３つのレジスタよりなる。命令セットは，１Byte でそれに複数の接頭辞が付く。１Byte の命令はさらに，４bit の機能項と４bit のデータ項よりなる。データ項は，一つの接頭辞ごとに４bit拡長することができる。また間接指定も，接頭辞により可能だが，ADD，EXCLUSIVE OR ，GREATER THAN については，１Byte で間接指定できる。

並列動作に対しては，マイクロプログラム化されたスケジューラーが担当する。プロセスの状態として，active 一実行中，実行待ち inactive 入力可，出力可，時間待ち，がある。

通信には，内部と外部の２種が存在する。内部通信では，入出力と同時に，通信の相手へのプロセスの切換えがおきる。

外部通信では，プロセッサーは，通信を自律した接結系にゆだねる。そしてプロセスはスケジューラーからはずされる。データは，２bit の Header と１bit のStop bit とともに送られ，２bitのAck を生成する。

⑲ Sequential Prolog Machine PEK

田村　直之、和田　耕一、松田　秀雄
金田悠紀夫、前川　禎男（神戸大学）

逐次型Prolog マシンPEKは，Prolog プログラムの高速実行を目指して，現在，開発が進められている実験機である。ＰＥＫは，ユニフィケーションとバックトラックを支援する専用ハードウェアを持ち，マイクロプログラムで書かれたインタプリタにより，逐次型Prolog を効率良く解釈実行する。

ユニフィケーションとバックトラックを効率良く行なうための専用ハードウェアとして，ＰＥＫには，２つのレジスタの値により，16 通りの多重分岐を行なうマッチング回路，グローバルスタック中への書き込み時に，そのアドレスを自動的にトレイリングスタックへプッシュする自動トレイリング回路，バックトラック時に変数を未束縛に戻す作業をサブシーケンサにより，主処理と独立に行なう自動アンドウ回路などがある。

PEKのマイクロ命令は水平型で96ビット24フイールドより成る。マイクロサイクルは可変で120 nsec～400 nsec よりマイクロ命令自身で指定できるため，マイクロアセンブラが適切なマイクロサイクル長を決定する。

簡単なProlog のインタプリタを作り，システムの性能評価を行なったところ，実行速度は約40 KLIPS であった。これは，DEC－10 Prolog コンパイラと同等の性能である。

⑳ Execution of Bagof on the Or-parallel Token machine

A. Ciepielewski, S. Haridi (Royal Inst. of Technology,スウェーデン)

論理型プログラムを効率的に実行するためには，新しいアーキテクチャと，プログラムの実行の新しい制御方式が開発されなくてはならない。本論文では，述語 bagof を並列に実行するために必要な機構について述べ，ついでそれがどのようにして，限られたハードウェア資源より構成されるOR－parallel token machine 上に実装されるかについて述べる。

OR並列処理の環境下において，bagof の実現は，複数存在する解を集めるための手段であるだけでなく，並列処理を制御するための手段でもある。これは，bagof によって，探索木の広がりが制限されるからである。また，bagof はＯＲ並列処理とＡＮＤ並列処理のインタフェースとして用いることができる。

bagof を並列処理下で実現するときの問題点は，bagof の実行の終了をどのようにして検出し，また bagof の実行によって得られた解をどのようにしてマージし，その解を必要とするゴールの環境下に引き渡すかにある。

本論文で述べる bagof の実現方式は，純粋なOR並列処理のみの場合に限らず，AND 並列処理とのインタフェースとしても有効である。