ICOT Technical Report: TR-433

TR-433

第五世代コンピュータプロジェクトにおける知識ベースシステムの研究開発

伊藤　英則、物井　秀俊、柴山　茂樹(東芝)、
宮崎　収兄(沖)、横田　治夫(富士通)、小長谷明彦(日電)

November, 1988

©1988, ICOT

ICOT
Mita Kokusai Bldg. 21F
1-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

第五世代コンピュータプロジェクトにおける知識ベースシステムの研究開発

伊藤英則、物井秀俊、柴山茂樹*、宮崎収兄†、横田治夫‡、小長谷明彦§
新世代コンピュータ技術開発機構 (ICOT)
研究所 第3研究室

概要

並列論理型プログラミングを研究開発のパラダイムとする第五世代コンピュータシステム (FGCS) の構成要素には並列推論サブシステムと知識ベースサブシステムがある。ここでは知識ベースサブシステムの研究開発の現状について報告する。なお、これらのサブシステムは並列論理型言語を中核として後期に統融合する。中期目標は、このための知識ベースサブシステム実験機の開発を通してその基本技術を明らかにし、種々の技術的課題を検証することである。知識ベースサブシステムのモデルとして分散および並列知識ベース等、複数のモデルを採用した。これらの開発で得られた技術およびサブシステムは、本プロジェクトの後期に完成するFGCSプロトタイプに統融合する。

1 はじめに

本プロジェクトでは、論理型プログラミングと並列処理をパラダイムとして効率的な知識情報処理のための第五世代コンピュータシステム (FGCS) プロトタイプの開発を目指している。このプロトタイプシステム実現のために、中期では並列推論サブシステムと知識ベースサブシステムの2つのサブシステムを開発した。後期では、この2つのサブシステムを並列論理型核言語 GHC (Guarded Horn Clauses) [Ueda 85] を中核として、第五世代コンピュータのプロトタイプとして統融合して行く計画である [Takewaki 87][Itoh H 88]。

本稿では、第五世代コンピュータプロジェクトにおける、知識ベースサブシステムの中期研究開発の現状について報告する。ここで述べる知識ベースサブシステムは、並列推論サブシステム上の複数の AI 向き応用プログラムが知識ベースを共有しアクセスする際に、知識ベースへの処理要求 (知識ベースの構築、検索および管理等) を効率よく実行するためのシステムとして位置付けられる。このような、知識ベースシステムでは、大量の知識 (オブジェクト) を効率よく処理するため、オブジェクト検索機能やトランザクション管理機能など、従来のデータベースシステムにおいて開発されている機能をほとんど備える必要がある。さらに、応用プログラムは複雑な構造の知識表現データを処理対象として、かつその処理のためにも知識 (メタ)

*株式会社 東芝 総合研究所
†沖電気工業株式会社 総合システム研究所
‡株式会社 富士通研究所 川崎研究所
§日本電気株式会社 C&C システム研究所

を使用するので、知識ベースシステム自体にも、従来のデータベースシステムよりも高度な知識処理機能とユーザインタフェース機能が要求される。

プロジェクトの前期3年では、知識ベースサブシステム研究の第1段階として、関係データベースシステム *Delta* を開発した[Murakami 83][Kakuta 85]。この開発では大量の知識(事実に関する知識)を効率良く処理するシステムのアーキテクチャについて、種々の経験を積み、その評価を得た[Itoh H 87][岩田 87][Itoh F 88]。また、論理型プログラミング言語 Prolog と関係データベースの統合化に向けたアルゴリズムの研究を行った[Kunifuji 82][Yokota84][Yokota 86a]。

現在実施中の中期4年では、知識ベースサブシステム実験機の開発を通して前述の機能を提供すること目指した。すなわち、複雑な知識表現データを直接扱うことができ、そのとき高度な推論等の処理が行え、また、論理型プログラミング言語に基づくユーザプログラムに対して使い易いインタフェースの提供することを目指した。このため、知識ベースサブシステムも推論機能を基本に先ずは構築すべきであると考え、前期に開発された逐次型推論マシンを研究開発の道具および環境に利用することとし、並列化に向けた以下のモデルを順次採用し、追加した。研究開発の効率化のため先発モデルと後発モデル間での技術流通および後発モデルへその蓄積を図り実施することとした。

- 第1のモデルでは、推論マシンを用いて、かつ、そのマシン上で動く論理型プログラミング言語を拡張して、高度な知識ベースに関する機能を実現することとした。本モデルによる知識ベース機能の有効性を実証するため、大量の知識を持つ実用的な知識ベースを構築しその管理システムを開発した。

  具体的には、高性能逐次推論プロセッサと大容量メインメモリを持つ、*CHI* (Cooperative High Performance Sequential Inference Machine) を高度な知識ベース管理機能を持たせることにより拡張して、知識ベースシステムを開発した。

- 第2のモデルでは、分散環境での知識ベースにおける効率的な分散協調問題解決の技術確立のために、ローカルエリアネットワーク (LAN) に接続された複数の逐次型推論マシン *PSI* (Personal Sequential Inference Machine) を用いて分散知識ベースの実験システムを開発することとした。

  このモデルは、分散知識ベースの管理と問合せの処理等は演繹機能を基本として実現していることより、このシステムを分散演繹データベースとも呼べる。また、問合せ処理にプログラム変換、部分計算、定理証明等の手法を取り入れその効率化を図った。また、否定表現を許した層状知識ベースにおける新しい検索手法を提案し、それを実証した。

- 第3のモデルでは、知識ベース処理の並列化による高速化を狙い、8個の要素プロセッサと比較的大容量の記憶機構からなる並列知識ベースシステムを開発することとした。

  具体的には、記憶機構として管理単位をページとしアクセス競合が無いマルチポートページメモリ方式を採用した。また、8個の要素プロセッサ上に並列処理制御ソフトウェアを開発した。各々の要素プロセッサはポートを介してページメモリに並列にアクセスできる。さらに、単一化検索を基本とする論理型の問い合わせ言語を設計開発した。これにより、*PSI* からこの実験システムに効率的なアクセスを可能とした。なお知識の格納構造として関係データモデルの拡張である関係型知識ベースモデルを提案した。

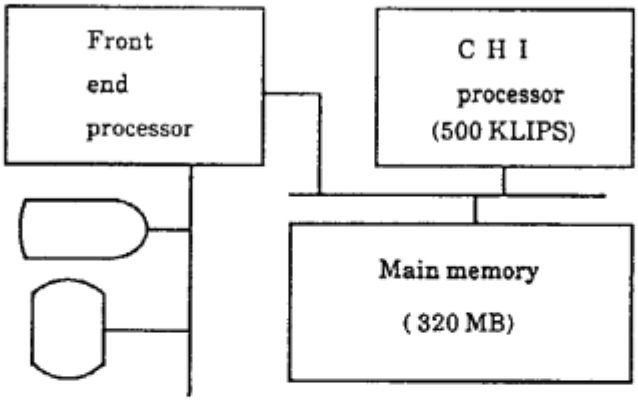

図 1: *CHI* ハードウェア構成
(*CHI* Hardware system configuration)

- 最後のモデルは、並列論理型核言語 GHC と並列知識ベースシステム間の有機的結合インタフェースを確立するために、優良優先検索を実行する並列プロダクションシステムを実証プログラムの一つとして選び、これに適合する知識ベース検索機能を GHC に追加し、その評価を実施した。

以降では、上記の 4 つの各実験システムの詳細について述べる。まず、2 章では *CHI* マシン上に開発した知識ベースシステム、3 章では *PSI* を使った分散知識ベースシステム、4 章では並列知識ベースシステム、5 章では並列論理型プログラミング言語による知識ベースインタフェースシステムについて、最後の 6 章はまとめと後期に向けた展望について述べる。

2 逐次推論マシンを拡張した知識ベースサブシステム

本章では、*CHI* マシン [Habata 87] 上に開発した高性能知識ベースシステムについて述べる。このシステムは、前期に開発した知識の管理技法等を踏まえ、さらに、複雑な知識ベースの検索・管理を高級化効率化するメカニズムの研究を目的として開発した。システムの特徴は、実用に耐えうる性能とメインメモリ容量を備えている点と、複数プロセス環境に適合するように多重名前空間 (multiple name space) 機能を拡張し、多重 - 多重名前空間 (multiple-multiple name space) 機能を導入した点である。

2.1 システムの概要

CHI は、FGCS プロジェクトで開発した推論マシンのひとつで、実用的な大規模論理型プログラムを効率良く実行できるように設計されている。*CHI* のハードウェア構成を図 1 に示す。高性能プロセッサ (ベンチマークプログラムで 500 KLIPS) と大容量のメインメモリ (320 MB) が、入出力操作用のフロントエンドプロセッサに接続された構成を持つ。

知識ベースシステムは、カーネル層、言語処理層、アプリケーション層の 3 層から成る (図 2)。カーネル層は、多重処理やリモート入出力操作の基本機能を提供する [Konagaya 87]。言語処理層は、SUPLOG (Prolog の方言で多重名前空間をサポートする) に対し、対話型の完全なプログラミング環境を提供する [新 88]。アプリケーション層は、DNA 塩基配列マッチング [Doolittle 86] や機械翻訳システムなどの特定の領域に対し、特別の推論ルールとファクト

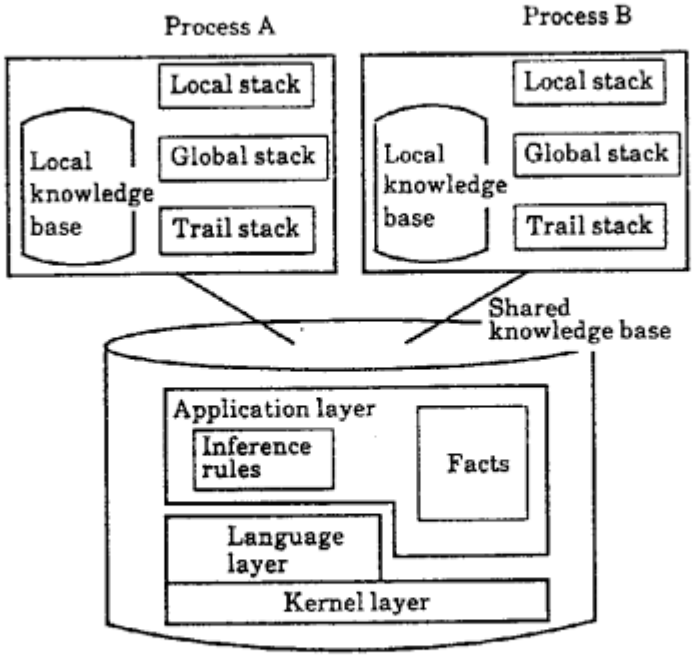

図 2: *CHI* ソフトウェア構成
(*CHI* Software system configuration)

を提供する。全プロセスが知識ベースを共有するが、多重 - 多重名前空間 (multiple-multiple name space) 機能を導入したことにより、各プロセスは、専用の実行環境 (ローカルスタック、グローバルスタック、トレールスタックとローカル知識ベース) で論理型プログラムを実行できる。ユーザの観点からは、システムプログラムにアプリケーションプログラムがロードされていれば、*CHI* は Prolog マシンというよりも、むしろ、対象領域指向の知識ベースマシンのように見える。

CHI では、単一化 (ユニフィケーション)、バックトラッキング、クローズインデキシングのための特別のハードウェア、ならびに高度のコンパイラ最適化により、高い性能を実現している [Habata 87]。コンパイラ最適化を利用するため、述語を「動的述語」(その定義を動的に変えられる述語) と「静的述語」(その定義を動的に変えられない述語) に分割した。これにより、一般的に知識ベースの大部分を占める静的述語について述語呼出しオーバーヘッドを除去できた。動的述語をインタープリタによって実行するとボトルネックが生じ易いため、アサート (assert) 時に節 (clause) をコンパイルする動的コンパイル法 (インクレメンタルコンパイリング手法) を導入した [小長谷 88]。その結果、*CHI*は、静的述語実行時の 3 倍程度の時間内で、動的述語を実行できるようになった。

大容量メモリ (320MB) の採用により、メモリ常駐型の知識ベースシステムを実現できた。知識ベースシステムでは、大量の知識データと多数の推論ルールが必要になる。たとえば、DNA 配列マッチングシステムでは膨大な DNA データ (2000 万 塩基) が、機械翻訳システムでは大型の翻訳辞書 (10 万 語) がそれぞれ必要である。このような、実用的な知識ベースシステムで最も時間を必要とする処理は、大量の知識ベースの検索である。従来のコンピュータシステムでは一般的には、ディスクアクセス時間がデータ検索時間のかなりの割合を占めているといえる。*CHI* はメモリ常駐方式の知識ベースシステムであるのでディスクアクセスの必要が無い。

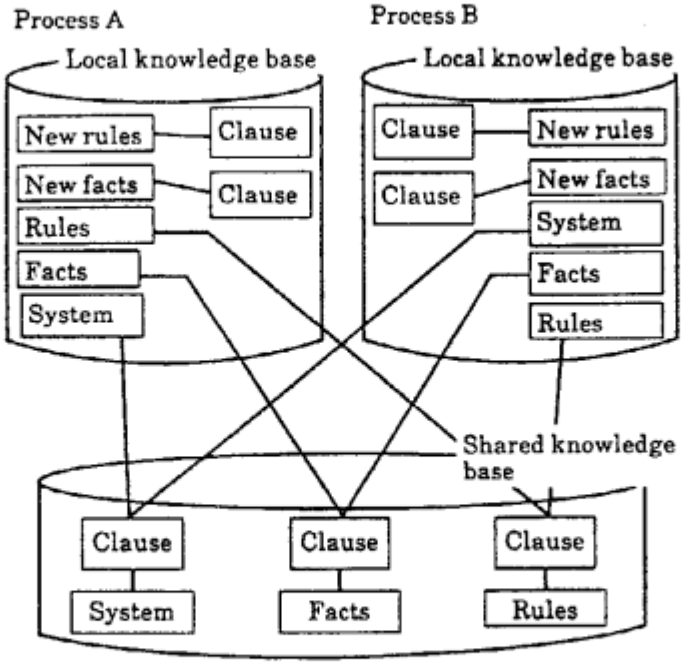

図 3: 多重 - 多重名前空間
(Multiple-multiple name spaces)

プロセス間の名前の衝突を防止し階層型知識ベースを表現するために、多重 - 多重名前空間機能を導入した。これはプロセスの生成時に名前空間をコピーすることにより、プロセス間の名前の衝突を防止する機能である。名前空間をコピーすることにより、プロセス同士で節データを共有しながら、個々のプロセスはそれぞれ独自の名前空間にアクセスできるようになる。また、多重 - 多重名前空間内でのカプセル化 (encapsulation) メカニズム、性質の継承メカニズム、述語のシャドーイング (shadowing) メカニズムを実現した。カプセル化メカニズムは、同じ名前を知識ベース内で、様々に使えるようにする機能である。継承メカニズムは、共有節を効率良く定義することを可能にする。シャドーイングメカニズムは、継承の際に発生する名前の衝突を解決する。これらは、非単調継承を表現する際にも便利なメカニズムである。

次節以下では、知識ベースシステムで重要な役割を果たす多重 - 多重名前空間について詳しく述べる。

2.2 多重 - 多重名前空間機能

前節で述べたように多重 - 多重名前空間機能を使うと、マルチプロセス環境で共有知識ベースをうまく実現できる。ここで解決すべき問題は、名前の無矛盾性保証とプロセス間の名前の衝突回避であり、これらは協調問題解決の分野で特に重要である。各プロセスは、知識ベース内の要素 (節) を動的に更新できるが、このとき、プロセス間で名前の衝突が起きる。この問題はロックメカニズムでも解決できるが、更新のスケジューリング問題は依然残る。すなわち、プログラムの結果がプロセススケジューリングの影響で変わる可能性がある。

プロセス間の名前の衝突問題を解決するために、名前空間コピー方式を採用した。この方式の特徴は、プロセスが生成されると各プロセスは共有知識ベースの名前テーブルをコピーすることと、コピーされた名前テーブルがローカル知識ベースに置かれるため、各プロセスがどの名前空間を変更しても他のプロセスには影響しない点である。図 3 にこの方式の例を示す。

この例では各プロセスが3つの共有名前空間(system, rule, fact)と2つのローカル名前空間(new-rule と new-fact)を持つ。なお、メールボックス方式を採用して、プロセス間通信を実現している。

さらに、一般的にはほとんどの知識ベースは静的データであることから、先ず知識ベースを共有知識ベースとローカル知識ベースの2つに分割し、共有知識ベースには全システムプログラムと静的知識を入れ、ローカル知識ベースにはプロセス自身のプログラムと動的知識を入れた。ここに、共有知識ベースを更新すると知識ベースアクセスの非決定性問題が生じるので共有知識ベースの更新は禁止することにした。

プロセススケジューリングは、プログラムが知識ベースをいくら動的に変更しても、プログラム実行に影響しない。ローカル知識ベースは、プロセスの停止時か消滅時に除去される。

2.3 階層型知識ベース

多重-多重名前空間機能を使うと、カプセル化機能、継承機能、述語のシャドーイング機能をサポートする階層型知識ベースもうまく表現できる。この3つの機能によりフレーム的階層知識が表現可能になる。

カプセル化機能 カプセル化機能により、内部で使う述語を外部から隠すことができる。この機能により、名前の衝突を低減し信頼性を向上させることができる。たとえば、コナガヤさんの預金口座に関する知識は次のように記述できる。

```
:- in_package(konagaya).
:- export withdraw/2, deposit/2.
:- dynamic account/1.

   withdraw(Amount,New_balance) :-
        retract(account(Balance)),
        New_balance is Balance - Amount,
       (New_balance >= 0
            -> assert(account(New_balance));
        print("Not enough balance!"),
        assert(account(Balance))).

   deposit(Amount,New_balance) :-
        retract(account(Balance)),
        New_balance is Balance + Amount,
        assert(account(New_balance)).
```

この例では、述語 account/1 はコナガヤさんの残高を記録するためだけに使われている。したがって、他人がこの残高に直接アクセスできないように、 in-package 述語により残高をカプセル化している。

継承機能 継承機能は、フレーム理論やスクリプトなどの階層知識表現方法を強化する。知識ベースにおける名前方式の継承機能の利点のひとつは、ルールの階層構造をファクトの階層構造と同じように構築できる点である。つまり、従来の AI ツールよりも柔軟にかつ強力に、ルールセットとデータセットを混在させられることである。

たとえば、鳥のクラス(bird)は、「鳥は羽を2枚持つ」とか「鳥は飛べる」といった鳥の一般的属性を持つことができる。これらのルールは次のように表現できる。

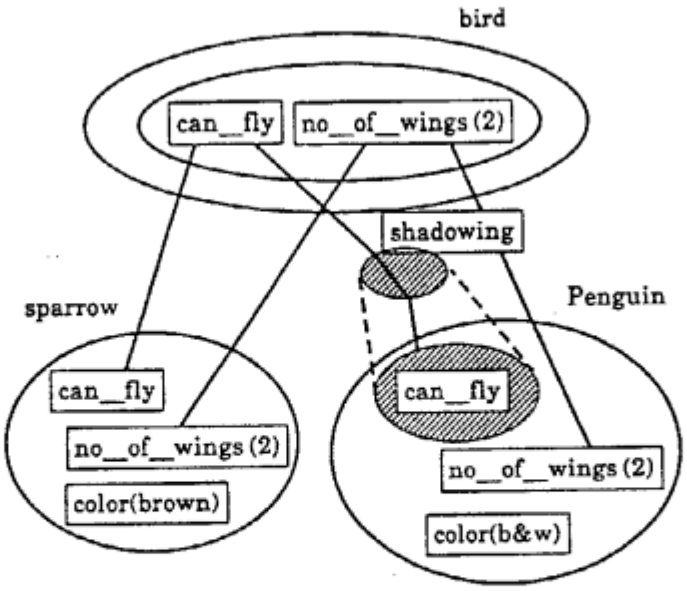

図 4: 知識ベースの継承
(Inheritance in knowledge bases)

```
:- in_package(bird).
:- external wings/1, canfly/0.

   wings(2).
   canfly.
```

スズメのクラス (sparrow) は、鳥のクラスを継承するものとして定義できる。

```
:- in_package(sparrow,[$use(bird)]).
:- external color/1.

   color(brown).
```

シャドーイング機能　シャドーイング機能を使うと、上位クラスから下位クラスへ述語が継承されないように述語を隠すことができる。一種の非単調な性質の継承は、この機能を使って表現できる。たとえば、ペンギンのクラス (penguin) を鳥のクラスを継承するものとして定義し、その際に述語 canfly/0 を隠すことができる。

```
:- in_package(penguin,[$use(bird)]).
:- external color/1.
:- shadowing canfly/0.

   color(b & w).
   canfly :- fail.
```

なお、非単調な性質の並列継承問題をモデル化し、その並列処理アルゴリズムの提案と GHC によるプログラムの作成・評価を行った [毛受 88a] [毛受 88b]。今後の応用プログラムの要素として発展することが期待される。

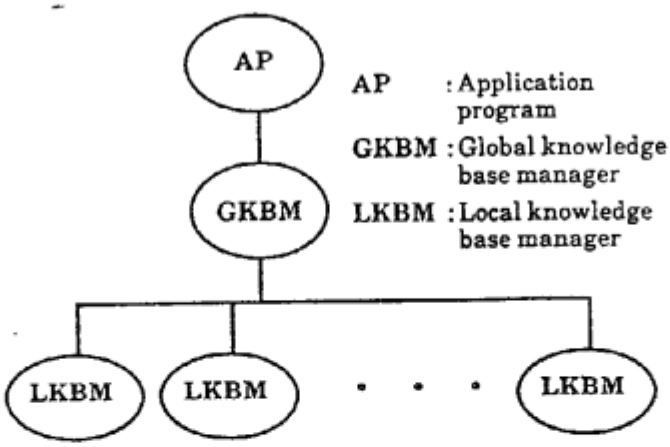

図5: *PHI* システムの論理構成
(Logical configuration of the *PHI* system)

以上この章では、多重 - 多重名前空間機能を使うと、プロセス間で知識ベースを共有可能にできることを述べた。さらにこのシステムは、履歴に依存するデータ構造、および、オブジェクト構造を導入すれば、知識ベースシステムをオブジェクト指向ベースに拡張でき、節 (clause) をオブジェクト間の拘束条件 (constraint) の定義に使うこともできるが、ここではその詳細については割愛した。

3 分散知識ベースサブシステム

様々な知識ベース間の協調と分散環境における知識ベースの処理は、将来の知識情報処理システムにとって重要である。分散知識ベースサブシステムの研究開発の狙いは、前章で述べた知識ベースシステムの研究開発で得られた結果を踏まえて分散知識ベースにおける分散協調問題解決技法の検証を実施し、かつ、並列知識ベースモデルへその結果を反映させることである。

本研究では 演繹機能をシステムの基本機能として採用したことより、このシステムは分散演繹データベース (DDDB: distributed deductive database) システムとも呼べる。

3.1 システムの概要

ここでの DDDB は、ルールの集まりである内包データベース (IDB: intensional database) と、ファクトの集まりである外延データベース (EDB: extensional database) から成る。

DDDB システムの主な技術的課題は次の通りである。

- 分散演繹問合せ処理
- 分散データベース更新 / 管理
- 論理型プログラミング言語と DDDB システム間のインタフェース
- 演繹データベースを効率良く処理するための専用プロセッサのアーキテクチャ

良く知られているように、EDB の具象単位節 (ground unit clause) は関係データベースのタプルとみなせることができるので、本システムでは、下層は関係データベース管理システム

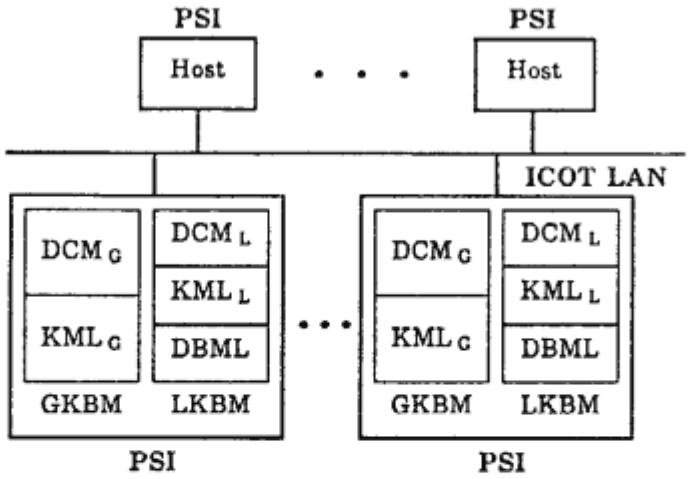

図 6: *PHI* システムの物理構成
(Physical configuration of the *PHI* system)

で EDB を扱い、上層は IDB を扱う 2 層構造を採用した。さらに、分散環境をサポートするため、演繹データベースシステムを、グローバル知識ベースマネージャ (GKBM: global knowledge base manager) とローカル知識ベースマネージャ (LKBM: local knowledge base manager) から構成した。

上の技術的課題を検討するため、*PHI* (Predicate logic based Hierarchical knowledge management) 実験システムを開発した。このシステムでは、図 5 に示すように、一つの GKBM と一つまたは複数の LKBM を、各ユーザまたは各アプリケーションプログラムに動的に割り当てる。

PHI は物理的には、図 6 に示すように LAN で結合された複数の*PSI* から構成される。各サイトの*PSI* は GKBM と LKBM を一つずつ持つ。また、重ね合せ符号法に基づいてインデックスを処理するアクセラレータを備えた専用プロセッサを、*PSI* 用の付加装置として実験試作した (図 7)。

3.2 分散演繹データベース

主な特徴 *PHI*は、多数のサイトに分散した演繹データベースから構成される。問合せはゴールアトムで示され、解は問合せの具象例 (ground instance) の集合である。この具象例インスタンスは、演繹データベースの節集合と論理的帰結 (logical consequence) である。*PHI*の主な機能は次の通りである。

- データベースは、本体に負のリテラルを許すように拡張された関数記号のない節の集合である。
- データ操作は、論理データ言語によって実行される。
- 問合せ処理戦略は、問合せ変換と動的最適化機能を持つボトムアップ戦略である。
- コンカレンシー制御は、2 相ロック (two phase lock) 方法によって実行される。

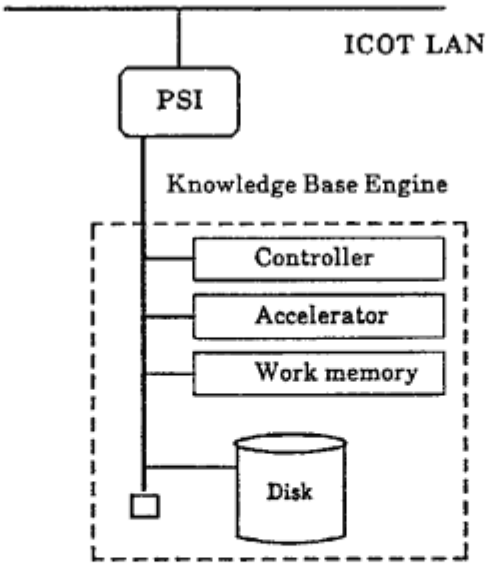

図 7: 知識ベース演算エンジン
(Knowledge base engine)

- 障害回復は、2相コミット (two phase commit) 方法によって実行される。
- セキュリティ管理は、パスワードとデータカタログを使って実現される。

最後の3つの機能で使われるアルゴリスムは、従来の分散関係データベース用に開発されたものと同様である。

PHI のインタフェースは、Prolog や ESP (Extended Self-contained Prolog) などの論理プログラミング言語に組み込めるようにした。*PHI* は問合せに対する解を集合として計算するが、ホスト言語の逐次実行に合わせ解を一部分ずつユーザプログラムに返す。ユーザプログラムにバックトラックが発生した場合、システムは代わりの解を返す。

分散問合せ処理　DDDB システムでは中間結果の転送にかかる通信コストを、適切な転送方向を決めることによって、削減することが重要である。たとえば、システムが2つの中間結果を結合 (join) する際は小さいほうを転送する。転送方向の決め方には、静的最適化戦略と動的最適化戦略の2通りがある。静的最適化戦略は、中間結果のサイズを実際の処理実行以前に予測して転送方向を決める。動的最適化戦略は、処理中に実際の中間結果のサイズを比較することによって転送方向を動的に決める。

PHI では動的最適化戦略を採用した。これは後述する再帰的問合せにおいては、中間結果のサイズを予測するのが困難なためである。この戦略の採用で中間結果のサイズの予測処理コストは無くなるが、中間結果のサイズの比較に要する通信オーバーヘッドは増大する [高杉 87]。しかし、この通信オーバーヘッド増大の問題は、*ICOT-LAN* にブロードキャスト通信機能があるため、*PHI* では重大な問題とはならない [Taguchi 84]。

再帰的問合せ処理　再帰的問合せ処理戦略は、トップダウン戦略とボトムアップ戦略に分類される。トップダウン戦略は、通常の Prolog の処理系と基本的に同様である。ボトムアップ戦略は、EDB 内の関係から出発して、中間結果を繰り返し生成することによって解を計算す

る。ここでは、EDBが大量であることを想定しているので、ボトムアップ戦略を採用した。ただし、ボトムアップ戦略には以下のような問題点がある。

1. データベースの最小不動点の全要素を計算するため、不要な計算量が大きくなる。

2. 繰り返し手順による冗長な計算を大量に行うことになる。

最初の問題の解決には、問合せ変換手順を使う。この手順は、最小不動点が小さい別の形式に問合せを変換する。変換の際、解の集合は変わらない。2番目の問題の解決には、差分計算法 (differential computation technique) [Balbin 87] を使う。

問合せ変換 節集合をそれと等価の節集合に変換するのに、ホーン節変換 (HCT: Horn clause transformations) [宮崎 88][Miyazaki 88][Sakama 87] という問合せ変換手順を使う。以下の3種類のHCTを提案した。不要な情報がデータベースから削除されるため、新しいデータベースの最小エルブランモデルは、元データベースのものよりも小さくなる。3種類のHCTを次に簡単に説明する。

HCT/P (部分評価によるHCT):

HCT/Pは、プログラム変換用に開発された部分評価 (partial evaluation) 法を基本としている。これは、論理的帰結を得るために、導出を用いる変換手順である。(演繹された) 関係の関係代数表現を関係記号に代入する時の手順を、一般化したものと考えることができる。

HCT/R (制約子によるHCT):

これは、制約子 (restrictor) という新しい述語を使って、包摂 (subsumption) に基づく元の節集合の論理的帰結を生成することにより、変換を行う手順である。HCT/Rは、マジックセット法 [Bancilhon 86] と同様な変換結果を与える。

HCT/S (具象代入によるHCT):

これは、節の変数に具象項を代入して論理的帰結を得ることにより、変換を行う手順である。推移閉包演算中に定数を移動する手順を、一般化したものである。

否定の処理 *PHI* システムでは、節の本体に負リテラルを許している。このため、

- データベースの意味を定義するには、構文的制約を加える必要がある、
- データベースの問合せ処理の効率化は、ホーン節データベースの場合より難しくなる、

という問題が生じる。

PHI では、層状 (stratified) データベースだけに制限する [Apt 88]。層状データベースでは、節の集合が階層的に分割されていて、各述語の定義節はその本体で自分自身の否定を (直接または間接的に) 呼ばないように制限されている [Apt 88][Van Gelder 86]。

層状データベースに対する問合せ評価方法は、ホーン節データベースの場合と同様、トップダウン戦略とボトムアップ戦略の二通りの方法がある。トップダウン戦略の問合せ評価方法が、最近いくつか提案された [Seki 88] [Kemp-Topor 88]。これらの方法ではボトムアップ戦略の特徴の一部を、トップダウンアルゴリスムに組み込んでいる。たとえば [Seki 88] は、

OLDTNF 導出という問合せ評価方法を提案した。これは OLDT 導出 (Ordered Linear Resolution with Tabulation)[Tamaki 86] を基本とし、否定を ' 失敗による否定 '(negation as failure) として扱う方法であり、データベースの応用として充分な層状プログラムのクラスを規定して、それにたいして正当性を示した。

PHI における層状データベースのボトムアップ問合せ処理は、基本的にはホーン節データベースの問合せ処理と同じである。*PHI*は、HCT を使ってまず問合せを等価な形式に変換し、次いで結果を一層ずつ計算する。ただし、HCT を無条件に使うと、層状でない結果を得ることがある。HCT/P と HCT/S はデータベースの層構造を保存するため、層状データベースでもそのまま適用できる。HCT/R は、層状データベースを非層状データベースに変換する場合があり、HCT/R を一般的には適用できない。

3.3 演繹データベース用の重ね合せ符号法

ボトムアップ戦略を採用する演繹データベースシステムでは、選択演算、結合演算、集合演算、集合比較などの演算が頻繁に実行される。演繹データベースが関係データベースと大きく違う点は、集合演算と集合比較の使用頻度が高いことである。重ね合せ符号 (SCW: superimposed code) 法の概念は、元々はテキスト処理用に提案されたものであるが、EDB と IDB の両方を効率良く処理する統一的アプローチを提供するものと期待される [Wada 88][Morita 88]。重ね合せ符号法の知識ベース演算エンジンへの適用性を検証した。

EDB に関する重ね合せ符号法　関係データベースでは、属性へのインデックスを使って、EDB のタプルへのアクセスを効率化する。問合せの条件に少数の属性しか頻繁に使わない場合は、インデックスの設計はやさしい。これはビジネスアプリケーションの場合に一般的である。演繹データベースでは属性をより統一的に扱うことが必要である。このような目的には、重ね合せ符号法に基づくインデックス方式が有望である。インデックスは次のようにして得る (図 8)。

1. 各キーの属性の値を BCW (binary coded word) というコードにハッシュする。
2. タプルに関する全 BCW の論理和を取り SCW を得る (SCW は元のタプルよりかなり小さい)。

このインデックスを使った検索は、次のように実行する。

1. 問合せの各キー属性の値を分割して BCW を得る。
2. BCW の論理和を取り問合せマスク Q を得る。
3. 各 SCW が $(Q \wedge SCW = Q)$ を満足する場合は、SCW をチェックする。インデックスに対応するタプルが問合せ条件を満足する場合は、SCW はこの条件を満足する。

再帰的問合せの処理に必要な集合演算と集合比較も、SCW インデックスを用いて実行できる。SCW インデックスを使い、対応するタプルの対が同じである可能性があるインデックスの対を作る。SCW は元のタプルよりかなり小さいため、SCW インデックスで前処理すれば性能を改善できる。重ね合せ符号法の利点は次の通りである。

- キー属性が多くある場合は、インデックス全体のサイズは他のインデックス方式より小さい。演繹データベースでは全属性がキーの可能性がある。

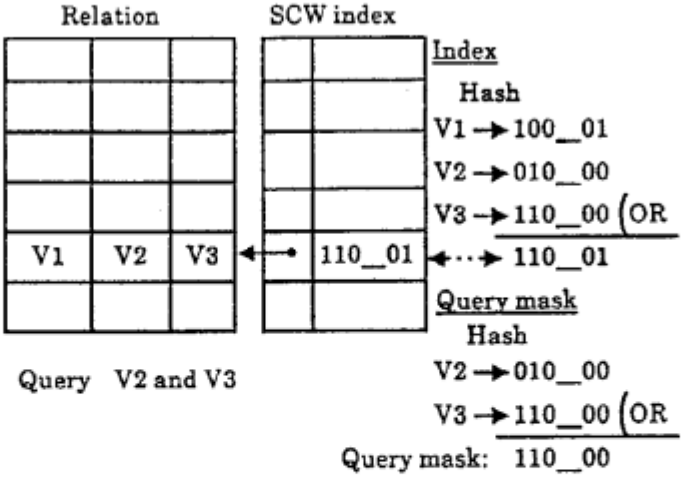

図 8: SCW の例
(Example of SCW)

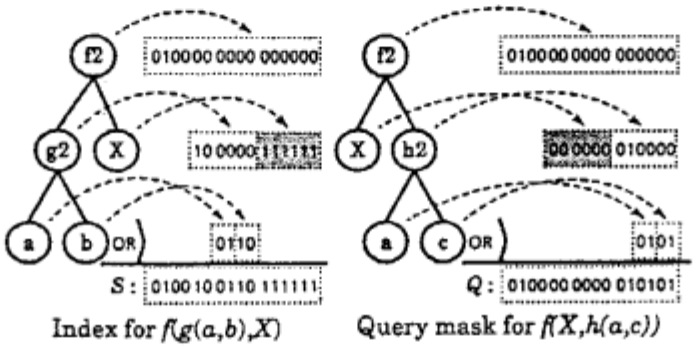

図 9: SSCW の例
(Example of SSCW)

- 2 個以上の属性が問合せ中で指定されている場合は、性能は向上する。
- インデックスの構造が単純なため、インデックス処理は簡単に並列実行できる。

つぎに、重ね合せ符号法の欠点を以下にあげる。

- 通常は全体的なインデックスの走査が必要である。インデックスが小さい場合でもこのための時間がかかる。
- 範囲指定の条件の検索を効率良く処理できない。

最初の問題は、専用ハードウェアか並列処理アーキテクチャ、あるいはこの 2つの組み合わせにより解決できる。実験システムではインデックス処理用の専用ハードウェアを用いた。重ね合せ符号法は、項に対しても拡張でき、これを SSCW(structured superimposed code words) と名付けた。SSCW の概念を図 9 に例示するが [Morita 88]、ここではその詳細は割愛する。

さらに、実験的なソフトウェア開発支援応用プログラムを、*PHI* システムの機能・性能を検証するために実験試作した。その他、分散知識ベースにおける論理型言語によるシソーラス

管理手法 [Oba 87] について基礎的実証研究の成果を得たが、ここでは同様にその詳細は割愛する。

4 並列知識ベースサブシステム

本章では並列知識ベースモデルに基づく知識ベースシステムについて述べる。前章で述べた分散知識ベースモデルは、複数の推論マシン*PSI*がLANで接続された環境での分散協調問題解決技法を研究開発の対象とした。ここでは、これらの結果を踏まえて、密に結合された複数の要素プロセッサが共有メモリ機構を有した環境での、並列協調問題解決技法を研究開発の対象とする。

4.1 システムの概要

本研究では、並列モデル知識ベースシステムの開発を狙いとして、複数のプロセッサとマルチポートページメモリ (MPPM: multiport page-memory) を持つ実験ハードウェアシステム (*Mu-X*) を試作実現する。

*Mu-X*では、知識ベースのモデルとして、[Yokota 86b] で提案された項関係モデルを採用した。すなわち、論理型プログラミング言語と知識ベース間のギャップを埋めることができる一つの候補として項関係モデルを採用した。項関係モデルを用いると、論理型プログラミング言語の基本的データ構造である項の集合を関係として格納することができ、さらにこれらの項の集合を単一化と関係演算 (relational calculus) を基本演算とする新しいタイプの検索処理が可能となる。このモデルは大量の知識に広くて浅い推論処理を繰り返して全解を求める処理に向いているといえる。さらに、この概念に従った、単一化に基づく問い合わせ言語を設計開発した [Monoi 88b]。これは*PSI*上のプログラムと*Mu-X*上の項関係知識ベース間のアクセス言語の基本となる。これに基づく実験機の試作を通じて並列処理方式を検証することがこのモデルの主要課題である。

中期では*Mu-X* は複数の*PSI* マシンに対するバックエンドとした。 *PSI* と*Mu-X* 間の通信を司るプロセスをESP[Chikayama 84] のクラスとして記述し、*PSI* プログラミング環境に追加した。このクラスは、 *PSI* のプログラミング環境からユーザが*Mu-X* と通信するメソッド (述語) を提供する。すなわち、ユーザプログラムからのメソッドコール (通常は述語 `retrieve`) で指定されたメッセージを*Mu-X* に転送する。

Mu-X の役割は、 ESP の述語 `retrieve` が示す問合せを実行するためのバックエンドシステムである。なお、いうまでも無く検索を高速化するために並列処理を採用した。

4.2 ハードウェア構成

Mu-X は、マルチポートページメモリを持つマルチプロセッサ構成である (図 10)。*Mu-X* のアーキテクチャは、[Yokota 86b] と [Morita 86] が示す知識ベースシステムのアーキテクチャに従っている。このアーキテクチャに関するシミュレーション [Sakai 88][Monoi 88a] により、知識ベースに関する処理種別と粒度と並列制御効果度の予備的研究を重ねた。その結果、知識ベースシステム内の要素プロセッサ (PE: Processing element) には中期研究開発では処理の柔軟性と今後の発展性を考慮して先ず汎用マイクロプロセッサを使用し各種実験評価することとした。また、 GHC を基本とする知識ベース向き要素プロセッサの VLSI 化については並列推論サブシステムの研究成果を流用する方が効率的であると判断し後期の研究開発課題とした。

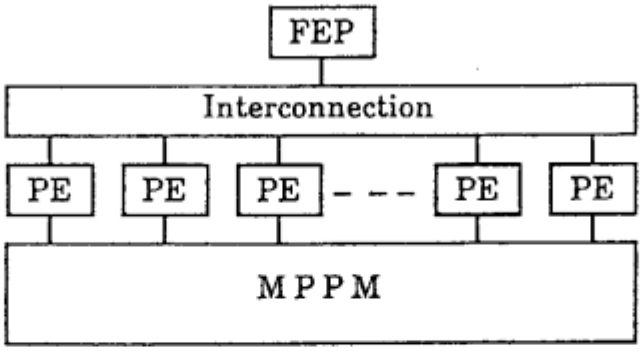

図 10: 並列知識ベースシステムのハードウェア構成
(Hardware configuration of the parallel knowledge base system)

表 1: ハードウェアの仕様
(Hardware specifications)

要素プロセッサ数	8
要素プロセッサ CPU	MC68020 (12.5MHz)
要素プロセッサメモリ容量	2MB
マルチポートページメモリ	8 ポート
	64MB (512 バイト / ページ)
	5MB/s(ポート転送速度)

ページ単位で競合のないアクセスが可能なマルチポートページメモリは、作業用に知識ベースの内容を一時的に記憶できる [Tanaka 84b]。マルチポートページメモリは、複数のメモリバンク (memory bank)、ポートとメモリバンクの結合組み合せ状態を周期的に切り換えるネットワーク、各ポートに接続されたポートコントローラ、およびメインコントローラから構成される。各ポートから任意のページへのアクセスは、ネットワーク内でポートとメモリバンクの組み合せを周期的に切り換えることと、メモリバンクの該当部分を適切に読み書きすることによって実現される。マルチポートページメモリを採用した理由は、大量の知識に対して問合せ処理を並列に実行する場合、複数の PE が競合無くメモリをアクセスできるようにするためである。見方を変えれば、マルチポートページメモリはメモリのバンド幅をメモリバンク数の倍数 (通常はポート数) まで高めたことになる。

Mu-X で試作したマルチポートページメモリはポート数が 8 で容量は 64MB である。ハードウェア全体の仕様を表 1 に示す。

4.3 制御ソフトウェアの特徴

Mu-X の制御ソフトウェア開発の目標は、知識ベース処理の分野で並列処理技術を追求することである。並列データベースシステムの研究は従来から進められてきおり、*GAMMA* [DeWitt 86]、*Grace* [Kitsure 82][Kitsure 83]、*MPDC* [Tanaka 84a]、*MDBS* [Demurjian86] など数多い。*Mu-X* の特徴は次の通りである。

- 簡潔さと柔軟さ

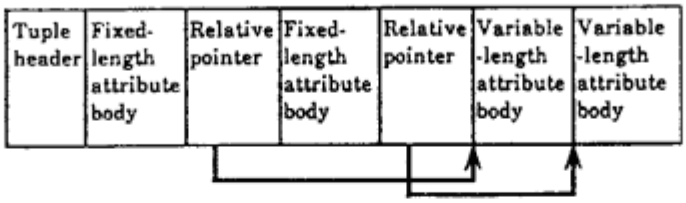

図 11 可変長レコードの表現法式
(Representation of variable-length records)

たとえば *Grace* や *MPDC* は、ハードウェア要素の種類が多く、またこのことから制御ソフトウェアが比較的複雑になる傾向にあるといえる。 *Mu-X*はこれらに比べると、比較的単純な並列処理で済む。プログラミングが必要なハードウェア要素は、処理の中核であるPEと、フロントエンドプロセッサ(FEP)の2種類である。しかもFEPで実現する機能は非常に単純で済む。 *Mu-X* は実験機であるため後で演算を変更したり追加する要求が出てくることが予想され、システムソフトウェアはこれらの変更要求に対応できるように設計された。

- 項データタイプのサポート

関係型知識ベースのサポートのために、(1)項データタイプ処理系、および、 (2)項データに対する新しい演算処理系を開発した。これらのため、基本的データ構造として、タグ付きデータと可変長レコードをサポートする。これは、項関係モデルではProlog のようにアトミックなデータと構造体の扱いを許すため、可変長レコード(図11)をサポートした。

4.4 並列処理

マルチポートページメモリの特性 マルチポートページメモリはページ方式のメモリシステムで、ページ転送制御ブロック (PTCB: page transfer control block) によって起動される。特性は次の通りである。

- アクセスはページ単位である。
- PE(ポート)間でアクセス競合は起こらない。
- アクセス時間は約1m秒程度である。

実験システムでは、512バイトを8個のメモリバンクに均等に水平方向に割り当てて、1ページの物理ページとしている。論理ページ(2KB)は、この物理ページ4ページで構成される。 各メモリバンクに割り当てられる64バイト(512/8バイト)の転送時間は約100μsである。このことから、8個のポートで並列に論理1ページを転送する時間は平均して450μs($4 \times 100 + 100/2$) かかる。なお、100/2はポーリングオーバーヘッドである(ポーリング時間を 100μsとしている)。また、ソフトウェア処理は約500μsかかる。以上を合計すると、論理ページ1ページの転送時間は約1msである。

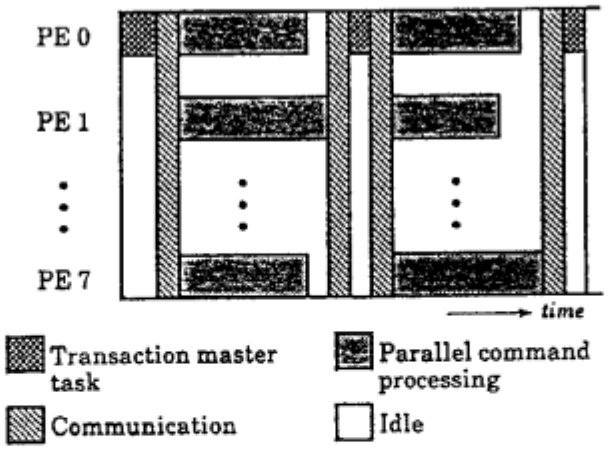

図 12: 並列処理タイミングダイアグラム
(A parallel processing diagram)

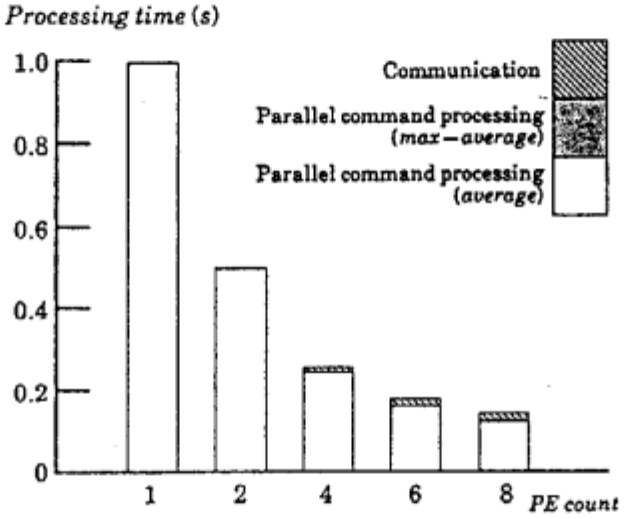

図 13: 選択演算の性能
(Performance of the selection operation)

動的制御プロセッサ割り当て　制御ソフトウェアを、中央集権的な特定の制御プロセッサ上に置く方式ではなく、トランザクション単位で動的に任意の PE が制御プロセッサの役割を果たす方式とした。すなわち、*PSI* でトランザクションを発生すると、アイドル状態の PE がそのトランザクションのマスタ (トランザクションマスタ) に指名される。

トランザクションのコンパイルとその分割並列実行　トランザクションマスタは、そのトランザクションの問合せのコンパイル、並列コマンドの生成、応答生成を担当する。並列コマンド実行は、複数の PE (トランザクションマスタの PE が含まれる場合もある) のタスクである。このときで、(1) 複数トランザクションと (2) 並列コマンド実行に対し並列処理が実行される。並列コマンド実行時のタイミングダイアグラムを、図 12 に示す。この図では、PE0 がトランザクションマスタに指名されている例である。

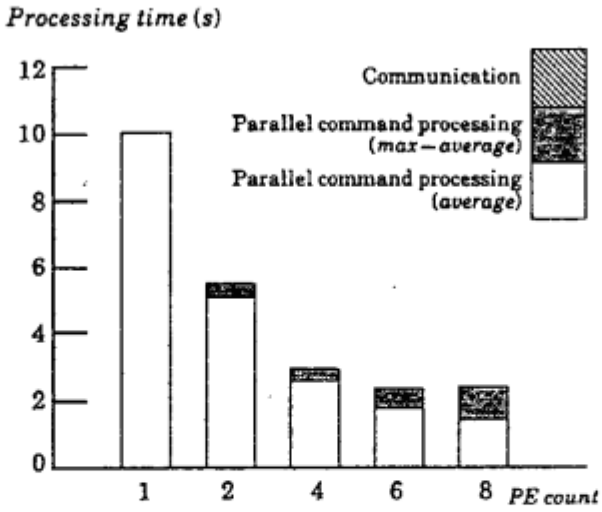

図 14: 結合演算の性能
(Performance of the join operation)

4.5 評価

これまでに予備的な性能評価を行った。この評価の目的は、ハードウェアの基本速度と並列処理方法の効率についてのデータを得ることである。

使用した問合せは選択演算と結合演算である。選択用の問合せは、1600 タプルの関係 (サイズは 500 KB) から 111 個のタプルを選択する。結合演算は、15KB の 111 タプルの関係 (前の選択演算の結果) と 215 タプルの関係 (サイズは 20 KB) の間で実行した。結合演算の実行にはネステッドループアルゴリズムを使った。結果は 37 タプルである。各タプルが可変長である点と、問合せは図 12 が示すように並列処理される点に注意されたい。

選択演算の評価結果を図 13 に示す。処理時間の合計は、並列コマンド実行の時間にほぼ等しい。並列実行のオーバーヘッド (この場合は通信時間) は、プロセッサ数が 6 に達するまでは認められなかった。プロセッサ数が 6 の場合でもオーバーヘッドは極めて少ない。

結合演算の結果を図 14 に示す。選択演算の場合とは対照的に、処理時間の合計はプロセッサ数が 6 になると飽和する。結合演算の場合も、並列コマンドの実行効果は十分にある。ただし、オーバーヘッドはプロセッサの増加とともに増大する。オーバーヘッドの原因は、PE の処理時間にバラツキがあるためである。通信時間が目立たないのは、絶対処理時間が選択演算の場合の約 10 倍となっているためである。

以上の現象は、PE の処理時間を選択演算と結合演算の場合について比較すれば明らかになる (図 15)。結合演算の場合にバラツキが見られるのは、PE 間で均等に分割できないためである。

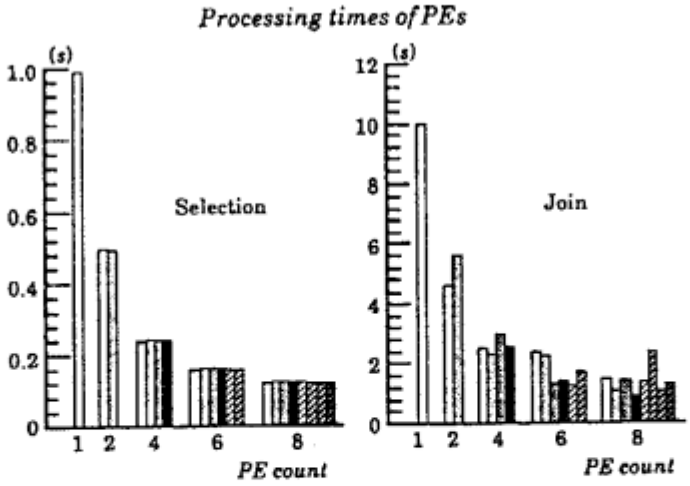

図 15: 処理時間の比較
(Comparison of processing times)

5 GHC と並列知識ベースサブシステム間のインタフェース

知識ベースサブシステムは、大量の知識の中から必要なものを迅速に取り出せなければならない。また、多様な知識を一様に取り扱えなければならない。さらに、取り出した知識を効率良く処理しなければならない。並列論理型プログラミング言語と知識ベース処理の専用システムとを有機的に結合することは、アプリケーションプログラムを充実させる有望な方法となる。本章では、並列論理型プログラミング言語と知識ベースシステムを結び付けるインタフェースについて述べる。この研究開発と前章で述べた並列知識ベースシステムとは後期ではさらに関係を深めて進める計画である。

5.1 システムの概要

単一化検索 (RBU: Retrieval-By-Unification) 演算は、知識ベース処理の専用システムとして提案された [Yokota 86b]。RBU 演算は、各種の知識を操作する関係演算と単一化演算を基礎としている。個々の知識は項で表される。項はそれ自身に変数を含む構造体であり、知識ベースは項の集まりである。 RBU システムでは、項を単一化可能関係により検索して同時に単一化を実行する。ここでは、GHC のストリーム入出力処理特性に合わせた 2 つの単一化関係代数演算処理、単一化制約ストリーム (urs: unification-restriction-stream) と単一化結合ストリーム (ujs: unification-join-stream) を新たに導入した。この他、従来からある検索系演算 (union, projection, join, selection など) と更新演算 (insert、 delete 等) もシステムの実証上必要であることより追加実現した。

GHC は 'committed choice' セマンティクスを持つ並列論理型プログラミング言語で FGCS プロジェクトの核言語として採用された。並列プロセス処理とプロセス間通信のためのストリームとを効率良く処理できる。しかし、複数の解を持つような知識ベースの検索には不向きである。これは GHC の変数は解が 1 度しか割り当てられないためである。また、GHC には知識ベースに置かれるようなグローバル情報を扱うことも困難であり、並列更新中に知識ベースの無矛盾性を保証する適当な方法がない。しかしながら、GHC で書かれた並列問題解決システムが項関係の検索・更新用の RBU コマンド

を出せれば、GHCでも知識ベースの処理ができることになる。同様にして並列演算の無矛盾性をチェックすることもできる。このように、GHCとRBUの有機的結合は処理対象および機能の幅を拡大することに役立つ。

5.2 並列検索

GHCとRBUの有機的結合の有効性を検証するための応用プログラムの一候補として並列プロダクション(ルールベース)システムを採用した。プロダクションシステムの基本概念は、状態遷移のためのプロダクションルールを初期状態から適用して、終了条件を満足する目標状態に達することである。プロダクションルールの適用により、ひとつの状態から複数の状態が生成される。そのため状態遷移は探索木を構成する。プロダクションシステムの目標は、探索木をたどることにより初期状態から目標状態に至る経路を得ることである。

プロダクションシステムの並列化は、プロダクションシステムが費やす膨大な時間を削減するためのひとつの方法である[Gupta 87]。並列処理は、たとえば探索木を並列にたどることによって実現される。この場合は、個々の状態から複数の新しい状態が並列に生成される。メモリとプロセッサ数には制限があるため、特別の探索方法を使う必要がある。一つの例として、最良優先(best first)探索[Barr 81]がある。この方法は、現在の状態に関する状態評価を使って、その時点で探索木の中で最良(best)の評価値を持つ状態が選び出されて、新しい状態になる。ただし、この方法は集中管理のため、最良の評価値を見つけることはそこにボトルネックを生じさせる。並列処理を効率化するには、制御を分散する必要がある。ここでは優良優先(better first)探索と名付けた新しい探索方法を提案した。この方法では、探索木の一部(部分木)だけを対象にして、最良の評価値を持つ状態を探す。この値は木全体については最良でない場合もあり得るので優良と名付けた。

並列の優良優先探索を実現するプロセス構成として木構造を使う。この木構造はプロダクションシステムがたどる探索木とは直接には関係していない。プロセス木には3種類のノード(プロセス)がある。根ノード、葉ノード、分枝ノードである。プロダクションは葉ノードで実行される。プロダクションの優先度はその評価値に基づき分枝ノードで制御される。システムの管理(たとえばユーザインタフェースなど)は根ノードで実行される。プロセス構成と探索木を図16に示す。

プロセス木のノードは再帰的に呼び出されるGHC節から生成される永久(perpetual)プロセスを使って実現される。プロセスの振る舞いは節の引数の変数に束縛されたストリームによって制御される。ストリームは、プロセスのメッセージとして処理される。この構成は第1章で述べた知識ベースシステムの並列モデルに適している。このシステムでは多数のプロセッサと共有記憶が一つのクラスタを構成する。各葉プロセスはプロセッサに置かれることを想定している(図17)。

5.3 GHCインタフェース

プロダクションは、個々の知識を知識ベースから取り出すことと、プロダクションルールに基づいて知識ベースを更新することによって実行される。知識ベースは、並列プロダクションプロセスにとってグローバル状態である。GHCでグローバル状態を実

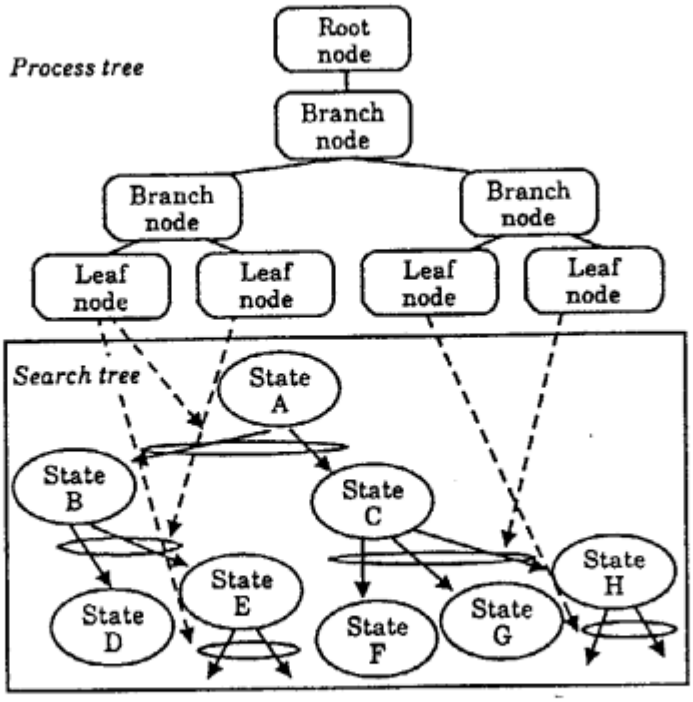

図 16: プロセスの構成と探索木
(Process configuration and a search tree)

現するために各節の引数として共通ストリームを使って永久プロセスを作成した場合でも、永久プロセス間のグローバル状態を扱えないし、知識ベースを効率的に検索・更新することもできない。GHC の変数には値を 1 度しか割り当てられないため、GHC で実現される単一化を複数の候補を持つ知識の探索に使うことはできない。GHC の変数に知識が一旦束縛 (bind) されてしまうと、その束縛内容は変更できないからである。

知識ベースを処理する専用システムに GHC を接続すれば、並列プロダクションシステムを構築できる。ここでの個々の知識は、GHC と同様に 1 階論理で定義された項である。したがって、構文の変形は必要とはしない。RBU は項の集合を項関係として記憶する。この項関係は、並列更新中の知識ベースの無矛盾性の保証に使われる。GHC に RBU の仕様を可能にするために特殊述語 rbu(C) を備えている。知識ベースの検索・更新用のコマンドはストリーム引数Cに束縛される。たとえば、

$$\texttt{C=[urs(tr1,[1],p(a,\$(1)),[1],X),ujs(tr1,[2],tr2,[1],[3],Y),}\cdots\texttt{]}.$$

最初のコマンド文 urs(tr1,[1],p(a,$(1)),[1],X)は、項関係tr1の最初の属性を対象に条件 p(a,$(1)) と単一化可能な項を探し、最初の属性を結果として導出する。結果は変数 X に束縛されたストリームとして戻される。

$$\texttt{X} = [\texttt{p(a,g(\$(2)))},\texttt{p(a,g(b))},\cdots].$$

第 2 のコマンド文 ujs(tr1,[2],tr2,[1],[3],Y) は、 tr1の第 2 の属性と tr2 の第 1 の属性を対象にユニフィケーション可能な項を探し、第 3 の属性を結果として導出する。結果は、変数 Y に束縛されたストリームにして戻される。

$$\texttt{Y} = [\texttt{q(\$(10),c)},\cdots]$$

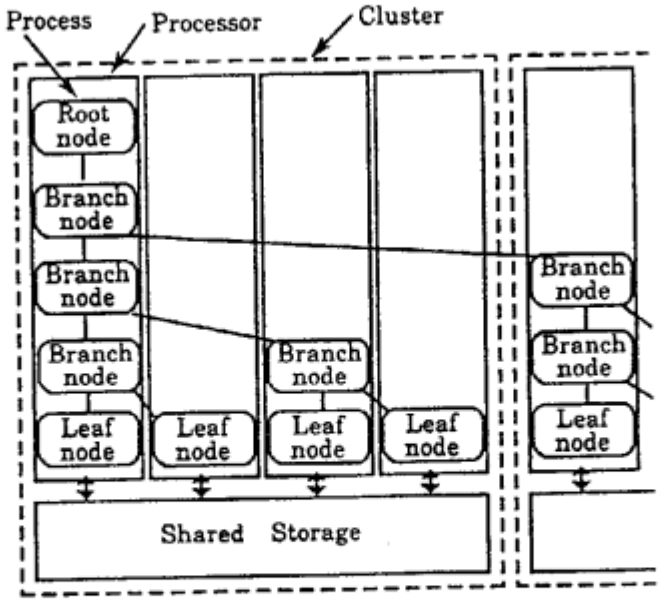

図 17: 並列モデル上の実現
(Implementation on the parallel model)

ここで、特殊関数記号 $ はコマンド文と結果の中の変数を表す。$ の導入は、GHC 変数は知識検索に使えないため検索用の変数を表示するのに特別の記号が必要であるからである。これらの変数は RBU では知識に束縛されるが GHC では束縛されない。これは Prolog システムの 'set of' 述語中のテンプレート述語に現れる無束縛変数に相当する。

5.4 RBU の実現

これまでに様々な検索速度改善方法が提案された。ひとつには専用ハードウェアを使う方法で例えば [Morita 86][Yokota 86b] が提案した単一化エンジンがある。[Ohmori 87] は節インデックス用のハッシュベクター (hash vector) を提案した。項用のスーパーインポーズドコードワードとこれを操作する専用エンジンを [Wada 88] が提案した。ここでは、項集合を単一化とバックトラッキングによって検索するためにインデックス方式を採用した。検索されるべき項は探索条件とユニフィケーション可能ならば互いに一部が似ている。バックトラッキングを効率化するためにはこれらの項をインデックスの近くに置くことが必要である。'trie' は同じ要素を共有する 1 種の木構造 [Knuth 73] でこの条件を満たす。項集合に対する 'trie' の例を図 18 に示す。

単一化のコストはオブジェクト項の要素間の比較の数に比例する。'trie' を使うと単一化を実行する際の比較の数が減る。たとえば、図 18 に示す項集合を対象に条件 p(f(a,b), h(c)) と可能な項を探す場合を考えてみる。'trie' 構造を使えば要素 p は 1 回だけの比較でよいが、使わない場合は 4 回の比較が必要になる。この条件と単一化可能な項全部を探す場合の比較の回数は、'trie' 構造を使えば 10 回だが使わないと 18 回必要である。

ハッシュテーブルは多くの種類の項を項関係に記憶する際に 'trie' 構造の前に使う

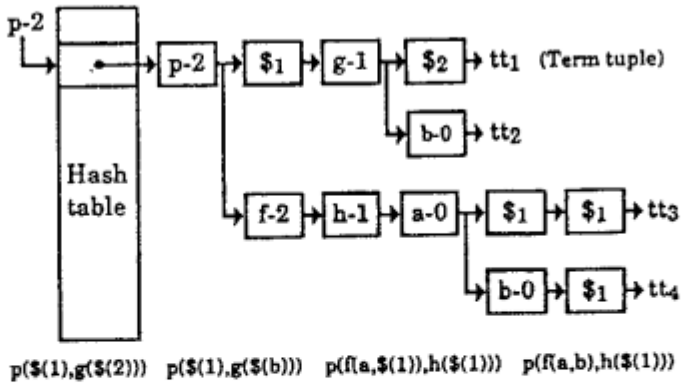

図 18: ハッシュと trie 構造を持つタプルインデックス
(Tuple index with hashing and trie structure)

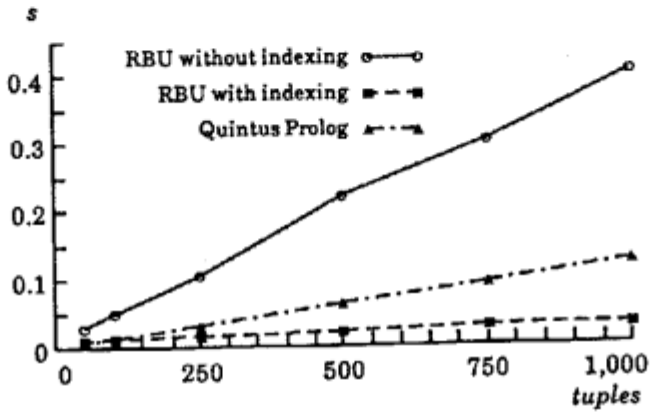

図 19: 検索速度の比較
(Comparison of search speeds)

(図 18)。項の最初の要素はハッシュキーとして使う。'trie' 構造はハッシュの競合解消とみなすことができる。

RBU プロトタイプの探索・更新速度を Quintus-Prolog インタープリタのものと比較した。Prolog コンパイラは 'assert' 述語と 'retract' 述語をサポートしないため (すなわち、Prolog コンパイラは知識ベースを更新できないため)、コンパイラについては調べていない。Prolog インタープリタと 'urs' の検索速度を、インデックスのある場合とない場合について比較した (図 19)。インデックスのない 'urs' は Prolog 節探索より約 4 倍時間がかかる。この探索時間はインデックスのない場合、Prolog でも 'urs' でもタプル数とともに増加する。これに対し、インデックスのある 'urs' の探索時間はタプル数が増加してもほとんど増加しない。タプル数が 1000 の場合、探索時間は Prolog 節探索の約 1/4 である。これはインデックスの効果である。

タプル挿入速度を 2 つのシステムについて比較した (図 20)。RBU を使うタプル挿入は、Prolog の consult 演算の約 1/6 の時間しか要さない。項関係についてインデックスを作成するためのオーバーヘッドは挿入時間の約 1/10 である。

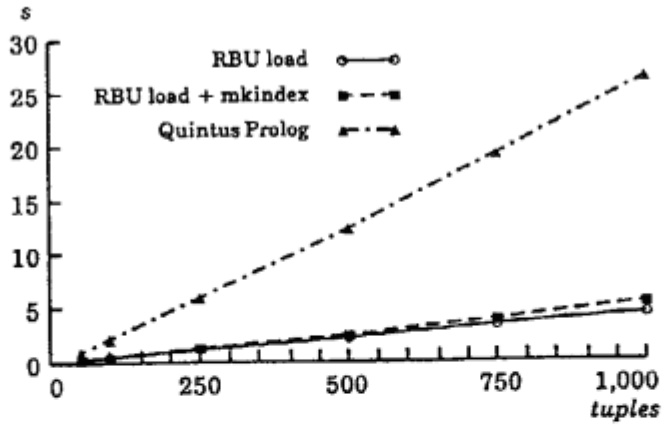

図 20: 挿入速度の比較
(Insert speed comparison)

6 結論

本論文では、FGCS プロジェクトで実施されている知識ベースサブシステムの研究開発の現状を報告した。特に中期で開発した、以下の 4 つのモデルに基づく知識ベースシステムについて報告した。

(1) *CHI* マシン上に開発した知識ベースシステム

CHI マシン上の知識ベースシステムは、多重処理環境に対し超高性能知識検索メカニズム、実用的なメインメモリ方式知識ベース、階層型節データベースを提供した。階層型知識表現を採用し、非単調性質継承機能および多重 - 多重名前空間機能を開発し、プロセス間の名前の衝突が起きない効率的知識ベースを実現した。

(2) 演繹データベースに基づく分散知識ベースシステム

ICOT-LAN で接続された*PSI*マシンを使って、分散演繹データベースシステムを開発した。このシステムの問合せ処理方法は、問合せ変換手法と組み合わされたボトムアップアプローチに基づいている。分散問合せの処理には、動的な最適化手法を適用している。また、インデックス処理用の専用ハードウェアを、知識ベース処理の効率化のための重ね合わせ符号法に基づいて試作した。

(3) 並列知識ベースシステム

8 個の要素プロセッサとマルチポートページメモリから構成される実験用ハードウェアと知識ベース管理ソフトウェアからなるシステムを開発した。このシステムは、項集合を格納していて、項関係演算の並列処理により効率的に検索・処理できる。このシステムは中期段階では*PSI* マシンに接続され、単一化に基づく強力な問合わせ言語を*PSI* とのインタフェースとして開発した。 なお、これまでの研究成果を踏まえて、後期では GHC を基本とする知識ベース向き VLSI 要素プロセッサを設計開発する計画である。

(4) 並列論理型プログラミング言語による知識ベースインタフェースシステム

並列論理型プログラミング言語インタフェースを知識ベースシステムに導入することを提案した。並列知識ベースシステムで採用した単一化検索と*GHC*との効率的組み合わせが可能かを検証するため、並列プロダクションシステムを一つの応用

プログラムとしてインタフェースシステムを試作したことにより、後期に実施する並列推論サブシステムとの融合手法の見通しを得た。

その他、本稿では詳細を述べなかったが、知識ベース内で連想検索を可能とする代表元による知識管理技法 [Sakama 88]、また、従来のキーワード検索とは異なる知識の意味検索を可能とする知識の抽象化表現技法 [Koguchi 88] 等の研究成果を得た。これらの技法は並列知識ベースシステム上の実証プログラムに取り入れられることが期待される。

最初のモデルと二番目のモデルは第五世代コンピュータのプロトタイプ開発のためのマイルストンとして位置付けられ、ここで提案し検証した様々な技術は、第五世代コンピュータのプロトタイプの研究開発に活用する。また、三番目と最後のモデルは並列推論サブシステムと統融合化に向けた後期研究開発の重点項目の一つとして位置付けられる。

謝辞

日頃熱心にご討論された第三研究室の方々、KBM WG 委員の方々に感謝するとともに、ここで述べたシステムの開発にご尽力頂いた関連各社の方々に感謝いたします。

参考文献

[Apt 88] Apt, K.R., Blair, H.A. and Walker, A., "Toward A Theory of Declarative Knowledge", Minker (ed.), in *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann Publishers, 1988

[Balbin 87] Balbin, I. and Ramamohanarao, K., "A Generalization of the Differential Approach to Recursive Query Evaluation", J. Logic Programming, Vol.4 No.3, 1987

[Bancilhon 86] Bancilhon, F., Maier, D., Sagiv, Y. and Ullman, J.D., "Magic Sets and Other Strange Ways to Implement Logic Programs" 5th ACM PODS, 1986

[Barr 81] Barr, A. and Feigenbaum, E. A., in *The Handbook of Artificial Intelligence*, 1, William Kaufmann, Inc. 1981

[Chikayama 84] Chikayama, T., "Unique Features of ESP", in *Proc. Int. Conf. Fifth Generation Computer Systems*, pp.292-298, 1984

[Demurjian 86] Demurjian, S.A. and Hsiao D.K., "A Multibackend Database System for Performance Gains, Capacity Growth and Hardware Upgrade", in *Proc. Int. Conf. on Data Engineering*, pp.542-554, 1986

[DeWitt 86] DeWitt, D.J., Gerber, R.H., Graefe, G., Heytens, M.L., Kumar, K.B. and Muralikrishna, M., "GAMMA - A High Performance Dataflow Database Machine", in *Proc. 12th Int. Conf. Very Large Databases*, pp.228-237, 1986

[Doolittle 86] Doolittle, R. F., "Of Urfs and Orfs, A Primer on How to Analyze Derived Amino Acid Sequences", University Science Books, Mill Valley, CA, 1986

[Goto 87] Goto, A., "Parallel Inference Machine Research in FGCS Project", in *Proc. of the US-Japan AI Symposium 87*, pp. 21-36, 1987

[Gupta 87] Gupta, A., in *Parallelism in Production Systems*, Morgan Kaufmann Publishers, Inc.,1987

[Habata 87] Habata, S., Nakazaki, R., Konagaya, A., Atarashi, A. and Umemura, M., "Co-operative High Performance Sequential Inference Machine: CHI", in *Proc. ICCD'87*, New York, 1987

[Itoh F 88] Itoh, F., Shimakawa, K., Togo, K., Matsuda, S., Itoh. H. and Oba, M., "Design, Implementation, and Evaluation of a Relational Database Engine for Variable Length Records", in *Database Machines and Knowledge Base Machines*, Kluwer Academic Publishers, pp.269-282, 1988

[Itoh H 87] Itoh, H., Abe, M., Sakama, C. and Mitomo, Y., "Parallel Control Techniques for Dedicated Relational Database Engines", in *Proc. 3rd Int. Conf. Data Engineering*, pp.208-215, 1987

[Itoh H 88] Itoh, H., Takewaki, T. and Yokota, H., "Knowledge Base Machine Based in Parallel Kernel Language", in eds. Kitsuregawa and Tanaka, in *Database Machines and Knowledge Base Machines*, Kluwer Academic Publishers, 1988

[Kakuta 85] Kakuta, T., Miyazaki, N., Shibayama, S., Yokota, H. and Murakami, K., "The Design and Implementation of Relational Database Machine Delta", in *Proc. Int. Workshop on Database machines '85*, 1985

[Kemp-Topor 88] Kemp, B.D. and Topor, W.R., "Completeness of a Top-down Query Evaluation Procedure for Stratified Databases", Dept. of Computer Science, Univ. of Melbourne, Technical Report, 1988, also in *Proc. 5th Int. Conf. and Symp. on Logic Programming*

[Kitsure 82] Kitsuregawa, M., Tanaka, M. and Moto-oka, T., "Relational Algebra Machine GRACE", *Lecture Notes in Computer Science*, Springer-Werlag, pp.191-214, 1982

[Kitsure 83] Kitsuregawa, M.,Tanaka, M. and Moto-oka, T., "Application of Hash to a Data Base Machine and Its Architecture", in *New Generation Computing*, OHMSHA, 1, 1983

[Knuth 73] Knuth, D. E., "The Art of Computer Programming", 3, Sorting and Searching, Addison-Wesley, 1973

[koguchi 88] Koguchi, T., Kondo, H., Oba, M. and Itoh, H.,"Knowledge Representation with Abstractive Layers for Information Retrieval", in *Proc. Int. Conf. Fifth Generation Computer Systems*, 1988

[Konagaya 87] Konagaya, A., Nakazaki, R. and Umemura, M., "A Co-operative Programming Environment for a Back-end Type Sequential Inference Machine CHI", in *Proc. Int. Workshop on Parallel Algorithms and Architectures*, East Germany, pp.25-30, 1987

[Kunifuji 82] Kunifuji, S. and Yokota, H., "Prolog and Relational Database for Fifth Generation Computer Systems", in *Proc. Workshop on Logical Bases for Data Bases*, Gallaire, et al.(eds.), ONERA-CERT, 1982

[Miyazaki 88] Miyazaki, N., Haniuda, H., Yokota, K. and Itoh, H., "Query Transformations in Deductive Databases", ICOT-TR 377, 1988

[Monoi 88a] Monoi, H., Morita, Y., Itoh, H., Sakai, H. and Shibayama, S., "Parallel Control Technique and Performance of an MPPM Knowledge Base Machine Architecture", in *Proc. 4th Int. Conf. Data Engineering*, pp.210-217, 1988

[Monoi 88b] Monoi, H., Morita, Y., Itoh, H., Takewaki, T., Sakai, H. and Shibayama, S., "Unification-Based Query Language for Relational Knowledge Bases and its Parallel Execution", in *Proc. Int. Conf. Fifth Generation Computer Systems*, 1988

[Morita 86] Morita, Y., Yokota, H., Nishida, K. and Itoh, H., "Retrieval-By-Unification Operation on a Relational Knowledge Base", in *Proc. of 12th Int. Conf. on Very Large Databases*, pp. 52-59, 1986

[Morita 88] Morita, Y., Itoh, H. and Nakase, A., "An Indexing Scheme for Terms using Structural Superimposed Code Words", ICOT TR-383, 1988

[Murakami 83] Murakami, K., Kakuta, T., Miyazaki, N., Shibayama, S. and Yokota, H., "Relational Database Machine: First Step to a Knowledge Base Machine", in *Proc. 10th int. symp. Computer Architecture*, pp.423-426, 1983

[Oba 87] Oba, M. and Itoh, H., "Basic Method for Mutual Utilization of distributed Personal Knowledge Base", in *Proc. IFIP WG 10.3 Working Conf. on Distributed Processing*, pp.587-598, 1988

[Ohmori 87] Ohmori, T. and Tanaka, H. "An Algebraic Deductive Database Managing a Mass of Rule Clauses", in *Proc. of 5th Int. Workshop on Database Machines*, pp. 291-304, 1987

[Sakai 88] Sakai, H., Shibayama, S., Monoi, H., Morita, Y. and Itoh, H., "A Simulation Study of a Knowledge Base Machine Architecture", in *Database Machines and Knowledge Base Machines*, Kluwer Academic Publishers, pp.585-598, 1988

[Sakama 87] Sakama, C. and Itoh, H., "Partial Evaluation of Queries in Deductive Databases", Workshop on Partial Evaluation and Mixed Computation, 1987

[Sakama 88] Sakama, C. and Itoh, H., "Handling Knowledge by Its Representative", in *Proc. of Int. Conf. on Exispert Database Systems*, 1988, Morgan Kaufmann Publishers, 1988

[Seki 88] Seki, H. and Itoh, H., "A Query Evaluation Method for Stratified Programs under the Extended CWA", ICOT Technical Report TR-337, 1988, also in *Proc. 5th Int. Conf. and Symp. Logic Programming*

[Shibayama 87] Shibayama, S., Sakai, H., Monoi, H., Morita, Y. and Itoh, H., "*Mu-X*: An Experimental Knowledge Base Machine with Unification-Based Retrieval Capability", in *Proc. France-Japan Artificial Intelligence and Computer Science Symposium 87*, pp.343-357, 1987

[Taguchi 84] Taguchi, A., Miyazaki, N., Yamamoto, A., Kitakami, H., Kaneko, K. and Murakami, K., "INI: Internal Network in the ICOT Programming Laboratory and its Future", in *Proc. of 7th ICCC*, 1984

[Takewaki 86] Takewaki, T. and Itoh, H., "Parallel Control Techniques for Retrieval Processes in the Parallel Logic Programming Language and their Evaluation", ICOT TR-255, 1987

[Tamaki 86] Tamaki, H. and Sato, T., "OLD Resolution with Tabulation", in *Proc. of 3rd ICLP*, 1986

[Tanaka 84a] Tanaka, Y., "MPDC: Massive Parallel Architecture for Very Large Databases", in *Proc. Int. Conf. Fifth Generation Computer Systems*, pp.113-137, 1984

[Tanaka 84b] Tanaka, Y., "A Multiport Page-Memory Architecture and A Multiport Disk-Cache System", in *New Generation Computing*, OHMSHA, 2, pp.241-260, 1984

[Ueda 85] Ueda, K., "Guarded Horn Clauses", in *Logic Programming '85*, E. Wada (ed)., Lecture Notes in Computer Science 221, Springer-Verlag, 1986

[Van Gelder 86] Van Gelder, A., "Negation as Failure Using Tight Derivations for General Logic Programs", in *Proc. 1986 Symp. on Logic Programming*, IEEE Computer Society, pp. 127-138, 1986, also to appear in *Journal of Logic Programming*

[Wada 88] Wada, M., Morita, Y., Yamazaki, H., Yamashita, S., Miyazaki, N. and Itoh, H., "A Superimposed Code Scheme for Deductive Databases", in eds. Kitsuregawa and Tanaka, in *Database Machines and Knowledge Base Machines*, Kluwer Academic Publishers, 1988

[Yokota 84] Yokota, H., Kunifuji, S., Kakuta, T., Miyazaki, N., Shibayama, S. and Murakami, K., "An Enhanced Inference Mechanism for Generating Relational Algebra Queries", in *Proc. 3rd ACM SIGACT-SIGMOD Symp. Principles of Database Systems*, pp.229-238, 1984

[Yokota 86a] Yokota, H., Sakai, K. and Itoh, H., "Deductive Database System Based on Unit Resolution", in *Proc. 2nd Int. Conf. Data Engineering*, pp.228-235, 1986

[Yokota 86b] Yokota, H. and Itoh, H., "A Model and an Architecture for a Relational Knowledge Base", in *Proc. 13th Int. Symp. Computer Architecture*, pp.2-9, 1986

[新 88] 新, 柳田, 小長谷, "SUPLOG マニュアル", 1988

[小長谷 88] 小長谷, "高速 Prolog インタプリタの構築とその評価", 情報処理学会記号処理研究会 46-4, 1988

[岩田 88] 岩田, 柴山, 酒井, 伊藤, 村上, "関係データベース処理エンジンのソータの試作と評価", 情報処理学会論文誌, Vol.28, No.7, 1987

[高杉 87] 高杉, 羽生田, 宮崎, 伊藤, "KBMS PHI における分散問い合わせ処理方式", 情報処理学会マルチメディアと分散処理研究会 34-9, 1987

[宮崎 88] 宮崎, 羽生田, 伊藤, "ホーン節変換: 演繹データベースにおける部分評価の応用", 情報処理学会論文誌, Vol.29, No.1, 1988

[毛受 88a] 毛受, 森田, 伊藤, "例外のある性質継承に関する並列アルゴリズム", 第 36 回情報処理学会全国大会, 6p-8, 1988

[毛受 88b] 毛受, 伊藤, 森田, "制約付き性質継承の並列アルゴリズム", 第 2 回人工知能学会全国大会, 4-3, 1988