TR-291

並列論理型核言語に基づく知識ベースマシン

伊藤英則

August, 1987

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191~5
Telex ICOT J32964

人工知能学会誌原稿
8/17 ICOT TR-291
15104

並列論理型核言語に基づく知識ベースマシン

伊藤 英則

(財) 新世代コンピュータ技術開発機構
Institute for New Generation Computer Technology (ICOT)
東京都港区三田1-4-28 三田国際ビル21F

あらまし

知識は変数を含む論理構造体である項(term)で表現されたものとする．知識ベースマシンの一つのモデルとして，項ベースマシンを研究・開発している．この項ベースマシンは，並列論理型核言語に基づいた，並列推論機構および項ベース蓄積・検索機構から構成される．ここでは並列論理型核言語処理系がもつデータストリーム制御・並列制御機能を活かし，全解収集機能を補強した項ベースの単一化検索機構に重点を置いて述べる．

1．はじめに

新世代コンピュータ技術開発機構では知識情報処理と並列アーキテクチャコンピュータをロジックプログラミングで融合するパラダイム[(1)]からその研究開発を行っている．

このため先ず始めに，従来のコンピュータの機械語に相当する核言語をロジックプログラミングにより定義して，それをシステムの中核に据える．しかもこの核言語は並列型である．つぎに，この並列型核言語を基にして下にはハードウェアシステムを，上にはソフトウェアシステムを構築する．ハードウェアシステムは並列推論サブシステムと知識ベースサブシステムからなる．ソフトウェアシステムは並列オペレーティングシステムと知識情報処理のための基本応用・実証ソフトウェアからなる．

知識ベースマシンの定まった明確な定義は未だ存在していない．ここでは以下を前提にした知識ベースマシンについて述べる．

知識ベースマシンは知識ベース機構と並列推論機構を兼ね備えるものとする．また処理効率の観点から，応用プログラムが走行する並列推論機構と知識ベース機構とは密結合しているものとする．さらに，これら2つの機構は単一のオペレーティングシステムの下

で制御される．

一階述語論理で用いられる変数を持つ論理的構造体である項(term)を知識とする．応用プログラム間で共有される知識の集合を知識ベースとする．また，項の意味については知識ベース管理プログラムで扱い，知識ベース機構ではこれを扱わない．

この理由は以下である．これまでに応用分野指向の知識表現言語が種々開発されているが，項はこれらのプリミティブ要素である．応用分野指向の知識表現言語から項への変換またその逆変換処理は容易といえる．知識を共有するハードウェア機構の知識表現の中間言語を，項とすれば種々の応用プログラムへの汎用性が保証できる[(2)]．

以上を前提にするので，これ以降の議論は知識を項と読み替えることができる．以下に知識ベースモデルとそのモデル上の操作・演算，および，並列論理型核言語に基づく知識ベースマシンへの要求機能と知識ベースマシンモデルを中心にして述べる．

2．知識ベースモデルと単一化検索

文献(7) で知識ベースモデルとして関係型項ベースモデルと，その上での単一化による検索のための基本演算を定義した．ここではこれらの概要と，その基本演算を効率化するための項の整理技法について述べる．なお，このモデルは大量知識をもつマシン用の下位基盤のモデルであり，応用プログラムまでの上位モデルを包含するものではない．

2.1 関係型項ベースモデル

新しい知識ベースモデルとして横田はルール集合とファクト集合を融合して持つ，関係型項ベースモデル[(7)] ①を提案した（注）．すなわち，項$t_1, \ldots, t_n$が関係Rにあることを以下のように定義した．

$$R(t_1, \ldots, t_n) \qquad ---①$$

これは関係型データモデルのデータを，項にまで拡張したものである（蛇足ながら前述のファクトはt_iで表現される）．この拡張を最も単純に解釈すれば，項の格納の仕方・検索の仕方・項間の関連付け等（従来のビュー(view)に相当）もまた項で表現できることになる．この解釈による能力の実現でも充分であるが，もっと極端な解釈もできる．例えば，Rを，t_1がPrologプログラムの節(clause)のヘッド，$t_2, \cdots, t_n$がそのボデ

（注）これまでに，ファクト集合を関係型データモデルととらえ汎用コンピュータを基礎として，関係データベースマシンDELTA[(3)(4)]を開発した．また，ルール集合とファクト集合に分離して双方を持つ演繹データベースマシンPHI[(5)(6)]を，逐次型推論マシン(SIM;Sequential Inference Machine)を基礎にして開発している．

ィーの関係としてとらえれば，Prologプログラムの節を知識として扱えることになる．この解釈では，知識とプログラムの区別をしなくて良いことになる．どのレベルの解釈でシステム化するかについては適用する応用による．以降，この知識ベースモデルを並列論理型核言語による並列知識ベースマシンPKBM[6][8]で扱うことについて述べる．

2.2 単一化検索

2.1 で述べた関係型項モデル上で検索のための基本演算について述べる．
関係型データモデル上で定義される演算の基本はデータの等価性チェックであった．関係型項モデル上でも項の等価性チェックをその演算の基本とすることもできるが，それでは，項のパターン整合によるフィルタリング処理ができる程度に留ってしまう．そこでここでは項の単一化演算にまで拡張した．これを基本にして検索することを単一化検索(RBU:Retrieval By Unification)と呼んだ．

制約関係演算と単一化演算を融合して単一化制約演算を，また結合関係演算と単一化演算を融合して単一化結合演算を定義した[7][9]．

ここで，図１に単一化制約演算と，図２に単一化結合演算の簡単な例を示す．

図１．単一化制約演算の例
図２．単一化結合演算の例

この基本演算を用いて，知識ベースを前述のPrologプログラムであると解釈した場合の検索アルゴリズムを図３に示す．

図３．単一化検索アルゴリズム

つぎに，この単一化検索アルゴリズムを用いて図４に示す先祖親子関係知識から "a" の子孫 "X" を求めることができる．その単一化検索過程の一部を図５に示す．

図４．先祖関係知識
図５．単一化検索過程

単一化検索アルゴリズムで単一化関係演算を繰返して最終解まで求めらる（図８で後述）．実際の応用では途中結果の解を推論機構に渡す処理の段階を最適化すれば，システム全体として処理効率を上げることができる．しかも，その最適化は個々の応用毎に定めることになる．このときは，知識ベース機構は解の候補を例えば十分の一から千分の一に絞るだけに留めればよく，知識ベース機構内では多くの知識に浅い推論を繰返し適用することが処理パターンの特徴となる．

2.3 項の順序付け，索引付け

大量の項の単一化検索演算を効率的に処理するためには，項の順序付け，索引付けを行うことが重要である．大量データについてはこれらは既に確立されている．ここではこれらを項に拡張したこれまでに提案した事項の概要を述べる．

2.3.1 項の順序付け

単一化結合演算では二つの項集合間の全ての組み合わせを試みることが要求される．ここでデータの場合と同様に，項がある条件で整列していれば組み合わせの数を減らすことができる．この条件に，置換におけるジェネラリティーの概念を導入した[7][9]．このジェネラリティーは②を満たす半順序関係である．

$$t_2 > t_1 \text{ iff } {}^{\exists}\sigma,\ t_2\,\sigma = t_1 \quad (\sigma\text{；置換}) \quad ---②$$

第３章で後述するように，項をストリームとして処理するには，このジェネラリティーを保存しながらさらに全順序付けを行う必要がある．すなわち，項が木で表現できることに注目して，その木を線形表現し，その文字列を辞書式に順序付ける．

木の線形化表現には，家族順とレベル順がある．これらの単一化結合演算における比較評価の詳細については文献（18）を参照されたい（文献（18）では木を線形化したものをさらにトライ(trie)化して組み合わせの数を減らすことと，項の格納スペースを縮めることの効果の評価も行っている）．

2.3.2 項の索引付け

従来のデータの索引付けにおける重ね合わせ方式の概要は以下であった．ファイルＦのレコードをR_iとする．R_iの幾つかのキーワードに或る関数（ハッシュ関数）を施してその２進符号語(BCW) を作成する．或るレコードR_iに対して，その全てのキーワードのBCW をビットごとにOR演算したものを重ね合わせ符号語(SCW; Superimposed Code Word) S_iと呼ぶ．つぎに，上と同一関数で検索キーワードのBCW を作成する．また，検索キーワードが複数個あれば同様にそのSCW を作成する．これをクェリマスクＱとする．このとき，③によってS_iからふるい落とされたレコードの中から，本来検索したいレコードを選び出す．

$$Q \wedge S_i = Q \quad (\wedge\text{:and演算}) \quad ---③$$

これを森田は項の重ね合わせとその単一化検索に拡張した．その概要について述べる．

重ね合わせ符号を用いた項の検索については，主にPrologの処理系の高速化の技法として研究されてきている．森田は，検索される側のレコード項に含まれる変数に対しては

1 …,(または0 …) と符号化し，検索する側のクェリ項に含まれる変数に対しては0 …,(または1 …) とする新しい方法を提案した．項を木表現したときの節・葉にあたる記号と変数は，その位置が深くなればなるほどその符号長を短くして重ね合わせる符号化方式である．これをSSCW(Structural SCW)[10][11]と名付けた．この方式により，適当なハッシュ関数を定めれば単一化からみた一つの連想サーチを可能とすることもできる．

BCW 長，ハッシュ関数，単一化率，および，項の絞り込み率についての評価および他の方式との比較は文献（12）を参照されたい．

3．並列論理型核言語の処理系と単一化検索の結合

まず，並列論理型核言語のシンタックス，セマンティックスとその処理系について述べる．つぎに第２章で述べた，単一化検索処理と並列論理型核言語の処理系を結合する方法について述べる．

3.1 ＧＨＣ(Guarded Horn Clauses)

上田は並列論理型核言語としてGHC を提案した[13]．以下にそのシンタックス，セマンティックスとその処理系の概要を，以降の議論に必要な部分に絞って述べる．

3.1.1 ＧＨＣのシンタックス

GHC のシンタックスは以下である．

$$H :- G_1, \cdots, G_n \mid B_1, \cdots, B_m . \qquad ---④$$

| はガード記号であり，| の左側がガード部，右側がボディー部である．他の記号は通常のPrologで使用されているものである．

3.1.2 ＧＨＣのセマンティックスとその処理系

GHC の場合もPrologの場合と同様に，与えられたゴール節から空節を導き出す．このときのルールも極めて単純で，以下である（正確には文献（13）を参照されたい）．

(1) 呼び出し側のガード部と変数とは単一化できない．
(2) ガード部が許可されるまではボディー部とガード部の変数は単一化できない．
(3) 同じヘッドがある場合は最初にガード部が許可された節のみ処理を続行する．
　そのとき，その他の候補であった同じヘッドを持つ節は消去する．
(4) ボディー部はAND-並列で実行する．

ここに，(3) はドントケア非決定性 (don't care nondeterministic)処理である．

つぎに，GHC による簡単なプログラムとその処理系を図６に例示する．ここにp(X)，q(X) は共有変数Ｘを持ったプロセスである．このプログラムは共有変数Ｘを介して，デ

ータストリーム［a｜Y］が流れる．p(X) はデータストリームジェネレータ，q(X) はデータストリームコンシューマであり，双方はデータストリームの送受に同期をとる．

図６に示したように，GHC 処理系は，与えられたプログラムを並列ゴールプール管理・スケジューラ・ガードテスト・ボディー実行のサイクル順に実行する．

図６．GHC プログラム例とその処理系

3.2　ＧＨＣの処理系と単一化検索の結合

GHC の処理系と単一化検索の結合方法について述べる．

GHC の特徴の一つであるデータストリームによるプロセスの入出力処理と大量の項関係演算処理を論理的に結合できれば，後述する並列推論機構と知識ベース検索機構とからなる知識ベースマシンの構築が容易であり処理の効率化が図れる．ただし，GHC は処理系の単純さから並列推論処理の性能を追求するためにドントケア非決定性処理を採用していて，単一化検索で要求される全部の解を収集する処理とは相矛盾する．何等かの手立てが必要である．

このために，GHC の処理系に組み込み述語を取り入れてこの矛盾を解決した(8)．図７に組み込み述語 "rbu" を持つプログラム例を示す．ここにプロセス "loop" はコマンドジェネレータであり，"rbu" はコマンドコンシューマである．プロセス "solve"は "rbu" と "loop" で全ての解を収集するまで消滅しない．第一番の "loop" 節は終了条件であり，第二番目の "loop" 節は "rbu" からの答をスタックする．また，最後の "loop" 節は与えられたKBとGOALから単一化検索を行ない，その時点の中間結果Ｓとこれまでの結果Ｌをマージして新中間結果Ｎとする．また，Ｓはプロセス "loop" とプロセス "rbu" 間のResultがデータストリームとして流れるチャネルでもある．図４で示す先祖親子関係知識KBを例にして，a の子孫を検索する "rbu" の内部処理を図８に示す．"unification-restriction" は図３のアルゴリズムを実行する．特に，[1,2],[1,3] は図５-cのプロジェクションを意味する．図７のプログラムのコマンドストリーム C_1 を受けながら，S_1 から順に解を求めて行く．この場合，Ｒ＝[]である S_2 と S_5 が解であり，図７のResultにスタックされる．また，S_8 と S_9 ＝[]であり，第一番目の "loop" 節の終了条件を満たし計算が終了する．なお，ここでは説明の単純化のために単一化制約演算だけを用いた．

ここではGOALから単純にトップダウンに検索したが，トップダウンとボトムアップ検索を組み合わせた効率的アルゴリズムの開発も盛んである EX(14)(15)(16)(17)．

図７．組み込み述語 "rbu" とＧＨＣの結合

図８．"rbu" 単一化検索例

4．ＲＢＵ専用装置の構成

これまでの議論は知識が主記憶上にあるか外部記憶上にあるかの区別はなかった．これから以降の議論は知識が外部記憶上にあることを前提にする．特に，大量の項が外部記憶に蓄積されているか，または多量のデータに混じって項が蓄積されている場合を想定する．このように大量に蓄積された知識の単一化関係演算による検索は，大部分が単純演算の繰返しである．この繰返し演算部分を専用装置でオンザフライ処理すれば，システム全体の効率化が図れる．このために，第３章で述べた組み込み述語 "rbu " の専用ハードウェアを実現する．この専用装置を単一化検索専用装置(RBU専用装置) [9] と呼び，並列推論機構と外部記憶機構の間に複数台設定する．並列推論機構と RBU専用装置の入出力インターフェイスは第３章で述べた推論プロセスと検索プロセス間のそれと同様にデータストリーム形式にできる．

RBU 専用装置の構成概念を図９に示す．項ソートユニット（２個），ペア生成ユニット，単一化ユニットから構成する．項ソートユニットは2.3.1 で述べた全順序に従い項を整列させる．ペア生成ユニットは項ソートユニットからの出力を入力として，項のペアを作成し，単一化ユニットに出力する．ただし，ここでの項のペアは可能な限り単一化可能性のあるものにだけ絞られる．単一化ユニットは項のペアにたいして単一化操作を実行する．これらのユニット全体でデータストリームをパイプライン処理実行[18]する．

各ユニットは，DELTA で開発した関係演算専用装置[3][19] の構成とのアナロジーとしてみることができる．項ソートユニットはデータのソートユニット，ペア生成ユニットはマージャ，単一化ユニットは関係演算器にそれぞれ相当する．

項ソートユニットでも2 way-sort/mergeアルゴリズム[20]が使える．またこのアルゴリズムにより項の整列も入力量ｎに対してオーダｎの処理時間を保証できる（ただし，ｎ≦ソートユニット内レジスタ容量）．データの場合は，2 way-sort/mergeアルゴリズムによりマージ処理はマージャで同様にオーダｎの処理時間を保証できたが，ペア生成ユニットでは項のペア生成は置換によるジェネラリティーが半順序であるので最悪オーダ n^2 かかる．単一化ユニットでは最汎単一子(most general unifier)を求めてから単一化を実行することにより，無駄な項の流れを最少にして処理の効率化を図れる[18]．

図９　ＲＢＵ専用装置の構成

5．ＲＢＵ専用装置のＧＨＣによる並列制御

充分大きな粒度の項関係演算は，それを分割して並列に処理すれば効率的である．そのためにRBU 専用装置を複数台備える．すなわち，第３章で述べた "rbu " コマンドの処理を第４章で述べた複数台のRBU 専用装置で並列実行する．なお，GHC で定義されるデータストリームによって１つ１つのRBU 専用装置へ起動がかかり，並列処理が実行される．

一つの "rbu" コマンドが指定する演算対象を，複数台のRBU 専用装置に割り付けるには下の三つの方法がある．

[1] 演算対象をすべて一つの専用装置に割り付ける．
[2] その時点で空いている専用装置に分割して割り付ける．
[3] 常に固定数（例えば専用装置台数）に分割して割り付ける．

[1] は分割処理が不必要であり，演算対象が比較的小さくコマンド件数が多いときに有利である．[3] はその逆であり，分割処理が必要であり，演算対象が比較的大きくコマンド件数が少ないときに有利である．[2] は分割処理と，特に専用装置の現在のステータス（空／稼動中）の監視が必要である．コマンド件数が多いときは[1] に近く，コマンド件数が少ないときは[3] に近くなる(21)(22)．ステータス監視処理負荷が小さければ，[2] が効率的な部分は[1],[3] の中間に存在する(21)．図10にステータス監視機能をもつ方法[2] のGHC によるプログラムを例示する．

図10．ＧＨＣによる専用装置並列制御プログラム

これらの三つの方法の内どれがその時点で最適かは，そのときの "rbu" コマンド種別，処理すべき対象の粒度，専用装置の稼動状況と，その制御方式自身の処理量によって決められる．特に，"rbu" コマンド種別により専用装置の並列使用形態が異なる．例えば，大量の項のソート処理はデータの場合と同様に多段フェーズが必要である．第 iフェーズで専用装置を n_i 台使用したとすれば，第 i+1フェーズでは n_i の半分を使用する．また，単一化制約演算系では処理フェーズは１段で済むので可能な限り多くの専用装置を用いた方が有利である．単一化結合演算は項のソート処理後に実行した方が得策である（これらの詳細評価については文献（22）を参照されたい）．

このような最適化を考慮した制御をメタコントロールと呼ぶ．このメタコントロールもGHC により記述できる．そのプログラム例を図11に示す．図10の'REscheduler' を図11のプログラムで置き変えれば，RBU 専用装置のメタレベルまでの並列制御が可能である．

図11．ＧＨＣによるメタコントロールプログラム

6．知識ベースマシンモデル

知識ベースマシン(KBM) の概念モデルを図12に示す．

知識ベースマシンは並列推論機構と知識ベース機構から構築される．上段のネットワーク(NW)とプロセスエレメント(PE)のクラスタ(CL)は並列推論マシン(PIM) として研究開発されている(23)．知識ベース機構は複数のRBU 演算専用装置(RE)とそれを接続する下段

のNWおよびグローバル共有記憶機構(GSM) から構成する．GSM は各CLからアクセスされる．また，各CL内にローカル共有記憶機構(LSM) がある．LSM とGSM により階層化共有記憶機構を構成する[24]．上段と下段のNWは物理的構成としては同一である．

REはCLから第４章で述べたRBU 検索コマンドを，NWを介して受ける．なお，RE台数はCL台数の約1/10を設定する．NWはRBU 検索コマンドを動的にREに割り付ける．第６章で述べたように，これにはGHC の並列制御機能を活かして実行する．NWに接続している，CLとREは単一の並列オペレーティングシステム(POS) によってプラグマティックに制御される．特に，REはPOS のKBM クラスとして並列制御される．

REの内部構成・機能については第５章で詳述した．再述するがCLとの入出力はデータストリーム形式である．

GSM は幾つかのメモリバンクをもっている．各バンクはリード・ライト専用ポートを持つ．GSM では論理的ページを各バンクにまたがらせて蓄積・管理するので，同一ページが複数のポートから同時にアクセスされ得る[25]．共有記憶機構への多重アクセスメカニズムは並列プロセッサの環境では特に重要である．ページサイズ，バンク／ポート数，リード・ライト速度，検索対象の知識粒度の相関の評価については文献 (26)(27) を参照されたい．

図12．知識ベースマシンの構成概念図

7．おわりに

項を知識として，前半ではまず，マシン用の下位基盤モデルとして関係型項ベースモデルを提案し，このモデル上で単一化演算と関係演算を結合したRBU 演算について述べた．また，AND 並列・論理型核言語としてGHC が開発されていることも紹介した．このGHC の下でRBU 演算を可能とするための方策について第４節で詳述したが，以下にそれを要約する．

GHC.RBU 双方ともに一階述語論理を基礎としていることと，以下の二つの点でGHC の下でRBU 演算処理を行うことは親和性がよい．

（１）RBU 演算専用装置の入出力処理に，GHC で定義されている
　　プロセスのデータストリーム型入出力機能がそのまま使える．
（２）大量の項関係演算処理の並列実行にGHC の並列処理系が使える．

ただし，条件を満足する全部の知識を収集することが要求される知識ベース検索処理と，GHC のドントケア非決定性処理（ある時点で一つの解が求まると同時にその他の候補の解の探索は全て放棄する）とは親和性がない．このために，同一条件で検索処理が続行できるようにする組み込み述語を，GHC 処理系に付加してRBU 演算で全知識の収集を可能

とした．

後半では，GHC に基づいた，並列推論機構および複数のRBU 演算専用装置をもつ知識ベース機構から構成される並列知識ベースマシン(PKBM)モデルについて述べた．現在は，この並列推論機構および知識ベースの蓄積・検索機構の実験機の開発とその構成要素の部分的評価を行っている．またさらに，システム全体の評価のために，大量知識の検索を必要とする組み合わせ・協調問題の応用プログラムを作成している．

謝　辞

知識ベースマシンの研究開発にあたって有益なご示唆を頂いた渕所長に感謝します．また，ここで引用したプログラム例[8] は元ICOTに在籍された横田（富士通㈱)，武脇（㈱東芝))氏によるものを使用させていただいた．ここに感謝します．また，日頃熱心なご討論に参加されている第一・第三研究室の諸氏，特に，田中，上田両氏，および，ＫＢＭワーキンググループと関連株式会社の皆様方に感謝します．

参考文献

(1) Fuchi. "Revisiting Original Philosophy of Fifth Generation Computer Systems Project." Proc. of the International Conference on Fifth Generation Computer Systems.ICOT.1984

(2) 横田, 伊藤 : "知識ベースシステム" 計測自動制御学会誌 vol.25.No.4.1986

(3) 岩田, 柴山, 酒井, 伊藤, 村上 ; "関係代数専用装置の評価" 情処学会誌 Vol.28 No.7. 1987

(4) Murakami.Shibayama. et al.. "A Relational Database Machine : First Step to Knowledge Base Machine." Proc. of 10th Annual International Symposium on Computer Architecture.1983

(5) 伊藤, 森田, 大場, 山崎 ; "KBMS PHI(1) 分散知識ベースシステムのシステム構成方式" 情処学会 第32回大会 3. 1986

(6) Itoh: "Research and Development of Knowledge Base Systems at ICOT" The 12th International Conference on VLDB. Aug. 1986.

(7) Yokota.Itoh; "A Model and an Architecture for a Relational Knowledge Base." The 13th International Symposium on Computer Architecture.p.2-9 June 1986. IEEE.

(8) Itoh.Takewaki.Yokota: "Knowledge Base Machine Based on Parallel kernel Language." The 5th International Workshop on Database Machines. Oct. 1987

(9) Morita.Yokota.Itoh. et al.; "Retrieval-by-Unification Operation on a Relational Knowledge Base Model." The 12th International Conference on VLDB. Aug. 1986.

(10) 森田, 和田, 伊藤 ; "スーパインポーズドコードを用いた構造体の検索方式" 情処学会 第33回大会 10. 1986

(11) Wada. Morita. Miyazaki. Itoh. el al.; "Performance Evaluation of Superimposed Code Sceheme for Relational Operations." The 5th International Workshop on Database Machines. Oct. 1987

(12) 森田, 物井, 中瀬, 柴山; "構造体のインデックス方式に関する一考察" 情処学会 第35回大会 10. 1987

(13) Ueda. "Guarded Horn Clauses." Logic Programming'85.E.Wada(ED). Lecture Notes in Computer Science 221. Springer-Verlag. 1986

(14) Yokota. Sakai. Itoh: "Deductive Database System Based on Unit Resolution" The Second Data Engineering Computer International Conference. pp.228-235 Feb. 1986. IEEE.

(15) Banclion. et al.. "An Amateur's Introdution to Recursive Query Processing Strategies." ACM SIGMOD '86. pp. 16-52. 1986

(16) Ceri.et al.. "Translation and Optimization of Logic Queries : The Algebraic Approach." VLDB'86. pp.395-402. 1986

(17) Miyazaki. Itoh; "Restricted Least Fixed Points and Recursive Query Processing Strategies." ICOT TR-183. 1987. ICOT

(18) Morita. Oguro.Itoh; "Performance Evaluation of a Unification Engine for a Knowledge Base Machine." ICOT TR-225.1987.(または，情報処理学会研資データベースシステム57-4. 1.1987)

(19) Itoh. Ohba. et al.; "Design and Evaluation of Retrieval Database Engine for Variable Length Records"
The 5th International Workshop on Database Machines. Oct. 1987

(20) Todd; "Algorithm and Hardware for Merge Sort Using Multiple Processors." IBM Jounal of Research and Development. 22. 1978

(21) Itoh. Abe. Sakama. Mitomo; "Parallel Control Techniques for Dedicated Relational Database Engines."
The Third International Conference on Data Engineering. Feb. 1987

(22) 武脇，伊藤；"並列論理型言語による検索処理プロセスの並列制御とその評価" 人工知能学会 第1回全国大会 (1987.7)

(23) Goto. Uchida; "Toward A High Performance Parallel Inference Machine." ICOT TR-201. 1986

(24) Itoh. Uchida; "Parallel Inference Machine and Knowledge Base Machine." Computers and Comunications Technology Toward 2000. IEEE Region 10 Conference. Aug. 1987. TENCON '87. ICOT TR-234

(25) Tanaka; "A Multiport Page-memory Architecture and A Multi-port Disk-Cache System." New Generation Computing. Vol. 2. pp.241-260. OHMSHA. Feb. 1984

(26) Monoi. Morita. Itoh; "Parallel Control Technique and Performance of a MPPM Knowledge Base Machine." ICOT TR-284. (または，情処学会研資データベースシステム60-4. 7.1987)

(27) Sakai. Shibayama. Monoi. Itoh; "Knowledge Base Machine using Multi-port Page Memory."
The 5th International Workshop on Database Machines. Oct. 1987

$$\sigma_{f(a,x)\diamond 1}\left\{\begin{array}{c}[\ 1\ ,\ 2\ ]\\ (f(a,y),g(y,z))\\ (f(b,z),g(a,x))\\ (f(w,w),g(c,w))\end{array}\right\} = \left\{\begin{array}{c}[\ 1'\ ,\ 2'\ ]\\ (f(a,x),g(x,z))\\ (f(a,a),g(c,a))\end{array}\right\}$$

◇：単一化制約

図1.　単一化制約演算の例

$$
\begin{array}{c}[\ 1\ ,\ 2\ ] \\ \left\{\begin{array}{l}(f(a,y),g(x,y)) \\ (f(b,z),g(a,a)) \\ (f(v,v),g(b,y))\end{array}\right\}\end{array}
\underset{2\diamond 1'}{\bowtie}
\begin{array}{c}[\ 1'\ ,\ 2'\ ] \\ \left\{\begin{array}{l}(g(a,z),h(z,b)) \\ (g(b,w),g(y,c))\end{array}\right\}\end{array}
=
\begin{array}{c}[\ 1''\ ,\ 2''\ ,\ 3''\ ] \\ \left\{\begin{array}{l}(f(a,z),g(a,z),h(z,b)) \\ (f(b,a),g(a,a),h(a,b)) \\ (f(a,w),g(b,w),h(y,c)) \\ (f(v,v),g(b,w),h(y,c))\end{array}\right\}\end{array}
$$

図2.　単一化結合演算の例

$(R : \text{result}) \leftarrow \varnothing$

$K_0 \leftarrow \sigma_{\text{goal} \diamond \text{head}} (KB : \text{term relation})$

$i \leftarrow 0$

while $K_i \neq \varnothing$ do

begin

$R \leftarrow (K_i\text{のbody}が[\,]\text{の}K_i\text{のhead})UR$

$K_{i+1} \leftarrow \pi_{K_i\text{のhead}, KB\text{のbody}} (K_{i\,\text{body}} \overset{\bowtie}{\diamond}_{\text{head}} KB)$

$i \leftarrow i + 1$

end (但し、πは通常の射影)

図3. 単一化検索アルゴリズムの例

KB	head	body
	[an (A , B) | Tail]	[pa (A , B) | Tail]
	[an (A , B) | Tail]	[pa (A , C) , an (C , B) | Tail]
	[pa (a , b) | Tail]	Tail
	[pa (b , c) | Tail]	Tail

図4.　先祖・親子関係知識

K_0	head	body
	[an (a , x)]	[pa (a , x) | Tail]
	[an (a , x)]	[pa (a , C) , an (C , x) | Tail]

図5-a.　単一化制約演算例 ($K_0 = \sigma_{\text{an (a , x)} \diamond \text{head}}$ KB)

k_1	1	2	3
	[an (a , b)]	[pa (a , b)]	Tail
	[an (a , x)]	[pa (a , b) , an (b,x)]	[an (b , x) | Tail]

図5-b.　単一化 結合演算例　$k_1 = K_0$　　$\underset{\text{body} \diamond \text{head}}{\bowtie}$　　KB

K_1	1	3
	[an (a , b)]	Tail
	[an (a , x)]	[an (b, x) | Tail]

図5-c ~~プロジェクション~~ 射影 例 ($K_1 = \pi_{1,2,3, \rightarrow 1,3}[k_1]$)

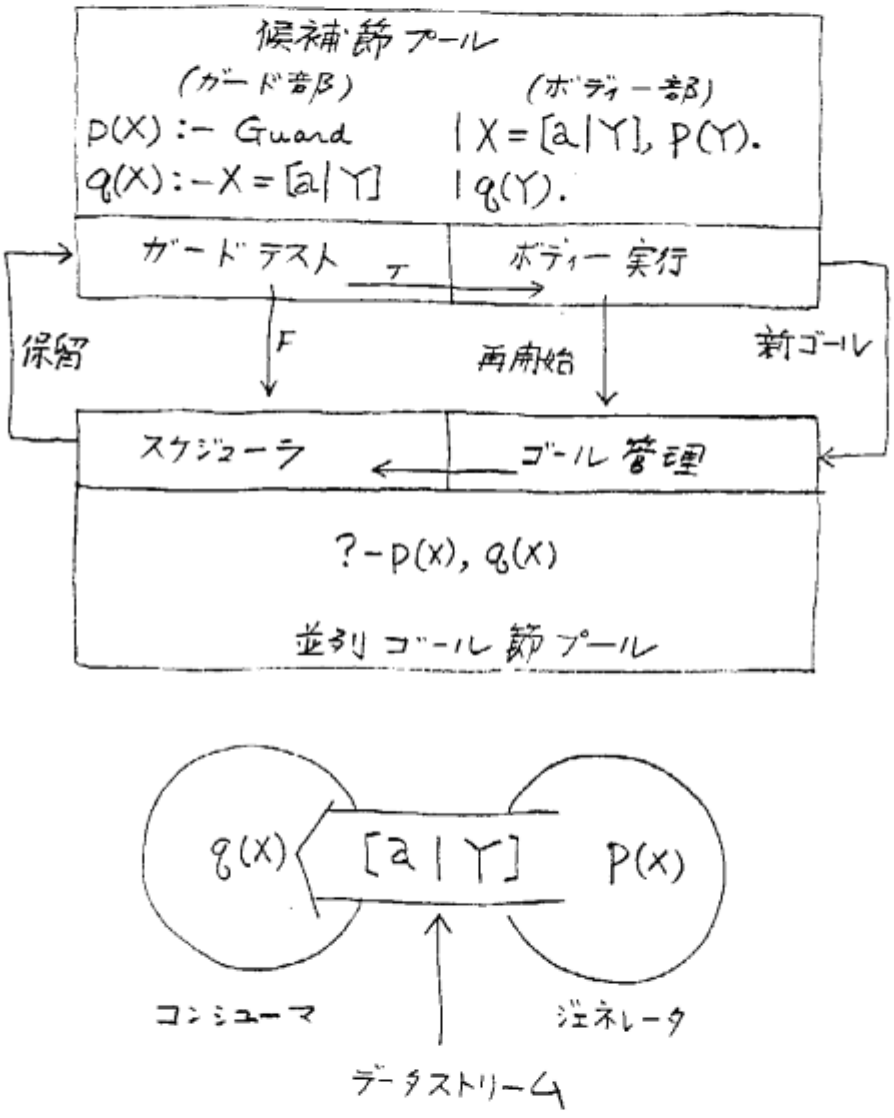


図 6.　GHC 処理系と簡単なプログラム例

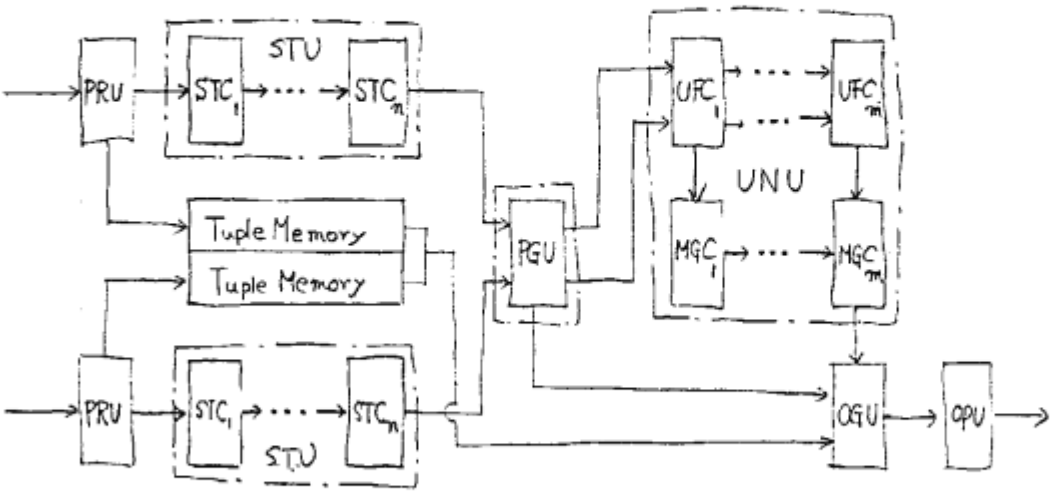
```
solve(KB,Goal,Result) :- true |
        loop(KB,cmd(C1,C2),[[[Goal],[Goal]]],Result),
        rbu(cmd(C1,C2)).

loop(KB,cmd(C1,C2),[]   ,X) :- true  | X = [], C1=C2.
loop(KB,CMD       ,[[G,R]|L],X) :- R = [] |  X = [G|Y],
        loop(KB,CMD,L,Y).
loop(KB,cmd(C1,C3),[[G,R]|L],X) :- R \= [] |
   C1 = [unification_restriction(KB,[1=G,2=R],[1,3],S)|C2],
        merge(L,S,N),
        loop(KB,cmd(C2,C3),N,X).
```

図7. 組み込み述語 rbu と GHC の結合.

```
unification_restriction(kb, [1=[an(a,X)],2=[an(a,X)]],[1,3],S1)
  S1 = [[[an(a,X)],[pa(a,X)]],[[an(a,X)],[pa(a,C),an(C,X)]]]
unification_restriction(kb, [1=[an(a,X)],2=[pa(a,X)]],[1,3],S2)
  S2 = [[[an(a,b)],[]]]
unification_restriction(kb, [1=[an(a,X)],2=[pa(a,C),an(C,X)]],[1,3],S3)
  S3 = [[[an(a,X)],[an(b,X)]]]
unification_restriction(kb,   [1=[an(a,X)],2=[an(b,X)]],[1,3],S4)
  S4 = [[[an(a,X)],[pa(b,X)]],[[an(a,X)],[pa(b,C),an(C,X)]]]
unification_restriction(kb, [1=[an(a,X)],2=[pa(b,X)]],[1,3],S5)
  S5 = [[[an(a,c)],[]]]
unification_restriction(kb, [1=[an(a,X)],2=[pa(b,C),an(C,X)]],[1,3],S6)
  S6 = [[[an(a,X)],[an(c,X)]]]
unification_restriction(kb, [1=[an(a,X)],2=[an(c,X)]],[1,3],S7)
  S7 = [[[an(a,X)],[pa(c,X)]],[[an(a,X)],[pa(c,C),an(C,X)]]]
unification_restriction(kb, [1=[an(a,X)],2=[pa(c,X)]],[1,3],S8)
  S8 = []
unification_restriction(kb, [1=[an(a,X)],2=[pa(c,C),an(C,X)]],[1,3],S9)
  S9 = []
```

図8. rbu コマンド 単一化検索列

16



PRU：前処理ユニット　PGU：ペア生成ユニット　OGU：出力項生成ユニット
STC：ソートセル　MGC：最汎単一化処理セル　OPU：出力処理ユニット
STU：ソートユニット　UNU：単一化ユニット

図9. RBU専用装置の構成概念図

```
% Top level
'REscheduler'([C|T],Stream) :- true | availableREs(Stream, NewStream,Free)
                                     divide(Free,[C|T],NewStream).
'REscheduler'([]   ,Stream) :- true | closeStream(Stream).

availableREs(St, NS,Free) :- true | inspect(St,NS,Ins), checking_free(Ins,Free).

% Inspection of REs status
inspect([]                  ,New,Ins) :- true | New=[], Ins=[].
inspect([stream(N,St)|Rest],New,Ins) :- true | New=[stream(N,SR)|NR],
       Ins=[(N,State)|IR], St=[ins(State)|SR], inspect(Rest,NR,IR).

divide([] ,C    ,St) :- true     | 'REscheduler'(C,St).        % All REs are busy.
divide(REs,[C|T],St) :- REs \=[] | 'REscheduler'(T,NS),
      data_division(C, REs, SubC), send_out(REs,SubC,St,NS).   % send out C to free REs

% 'RE' manages status of REs, SendToRE sends C to Nth RE.
'RE'(N,[term  |Command]) :- true | Command=[].                 % termination
'RE'(N,[ins(C)|Command]) :- true | C=free, 'RE'(N,Command).    % inspection of status
'RE'(N,[cmd(C)|Command]) :- true | sendToRE(N,C,Response),     % retrieval command
      response(Response,Command,Next), 'RE'(N,Next).

response(end,Command     ,N) :- true | Command=N.              % process ends
response(R  ,[ins(C)|Cmd],N) :- true | C=busy,response(R,Cmd,N).% response of status.
```

図10. GHCによるRBU専用装置並列制御プログラム(例)

```
meta_control([]                    , ST, REs) :- true | colseStream(ST).
meta_control([cmd(C,Type,Size)|Rest], ST, REs) :- true |
            availableREs(ST, NewST, Free),
            strategy(C,Type,Size,Free, Method),
            solve(Method, REs, Free, cmd(C), IntST, NewST),
            meta_control(REs, Rest,NewST).

solve(method1, REs, Free, Cmd, ST, NewST) :- true |
            selectRE(Free, RE), send_out([RE],[Cmd], ST, NewST).
solve(method2, REs, Free, Cmd, ST, NewST) :- true |
            data_division(Cmd, Free, SubCmd), send_out(SubCmd,Free, ST, NewST).
solve(method3, REs, Free, Cmd, ST, NewST) :- true |
            data_division(Cmd, REs, SubCmd), send_out(SubCmd, Free, ST, NewSt).
```

図11. GHCによるメタコントローラプログラム(例)

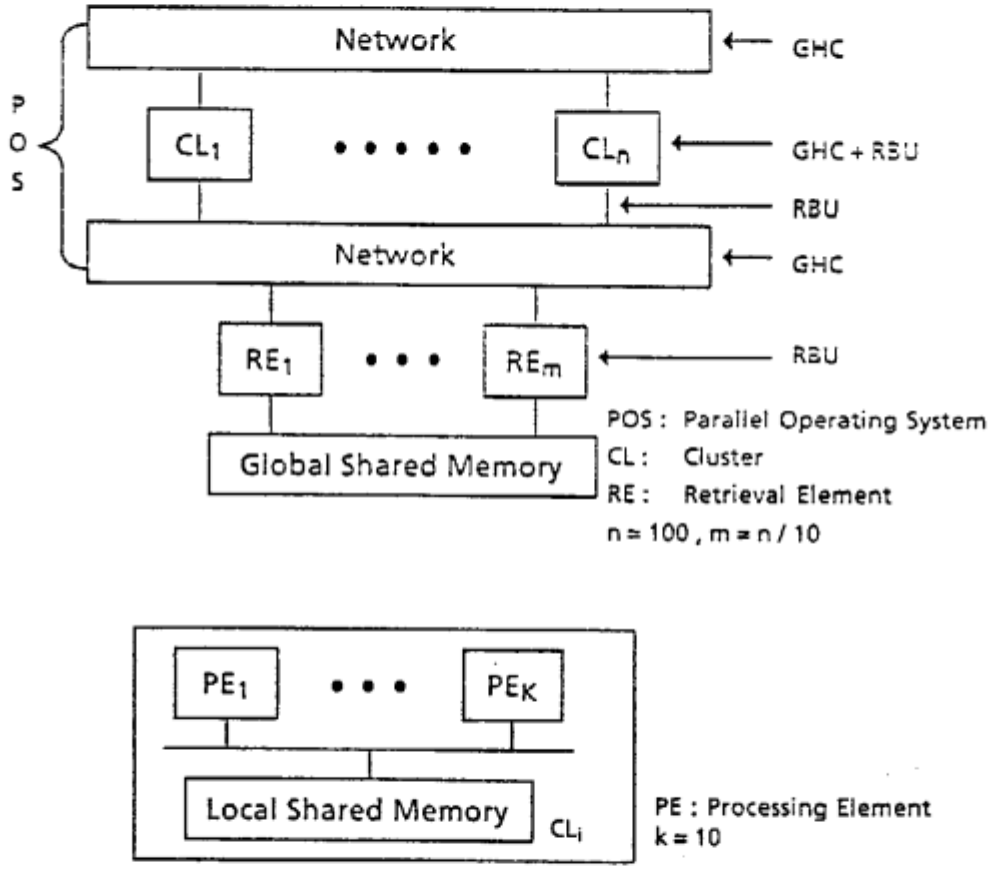


図12 知識ベースマシンの構成概念図