

逐次型推論マシンCHI-IIの性能評価

幅田 伸一、小長谷 明彦、新 淳、横田 実
日本電気(株) C&Cシステム研究所

本稿では、通産省の第5世代計算機プロジェクトの一環として研究開発して来た逐次型推論マシンCHI-IIの性能評価結果について報告する。

CHI-IIは述語論理型言語を使用した知識ベース・システムの高実行を目指しており、大容量メモリと高速インタプリタを特徴とする。インクリメンタル・コンパイル方式を採用することで、インタプリタの高速化を実現した。現在、約100メガバイト規模のDNA配列検索知識ベース・システムをCHI-II上に構築中である。性能評価は、ECRC (European Computer-industry Research Center) が提案するベンチマークプログラムを使用した。ベンチマークプログラムを使用した性能評価では、述語呼び出し、ユニフィケーション、バックトラック処理などの基本機能の処理速度と9個の簡単なPrologプログラムの実行速度を求めた。評価の結果、コンパイルしたコードの実行性能で、Sun4/280上のQuintus Prolog 2.3版の1.5倍、インタプリタの場合は6倍の処理速度であることが判った。また、インクリメンタル・コンパイル方式は、汎用計算機上でも効果があることを確認した。

Performance Evaluation of Sequential Inference Machine CHI-II

S. Habata, A. Konagaya, A. Atarashi and M. Yokota

C&C Systems Research Laboratories, NEC Corporation
4-1-1 Miyazaki Miyamae-ku, Kawasaki 213, Japan

A Co-operative High performance sequential Inference (CHI-II) machine has been developed within the Fifth Generation Computer Systems (FGCS) project in Japan. This paper describes the performance evaluation of CHI-II.

CHI-II provides the large main memory capacity and a fast interpreter for large knowledge-base systems. The benchmark program measurement characterizes the CHI-II architecture from the viewpoints of principle operations, such as predicate call, unification, backtracking, and simple Prolog program execution. The result of measurement proves the effectiveness of the CHI-II architectural support and incremental compilation. The average performance of compiled-programs is 1.5 times faster than that of Quintus Prolog on a Sun4/280 workstation in spite of the disadvantage of machine clock cycles: 200 nano seconds versus 66 nano seconds. The incremental compilation enables the CHI-II interpreter to execute 6 times faster than the Quintus Prolog interpreter. It can be also applied to the Prolog implementation on conventional machine.

1. はじめに

通産省の第5世代計算機プロジェクトの一環として、逐次型推論マシンCHI-II (Co-operative High Performance Sequential Inference Machine) を研究開発して来た。^{[1][2]} CHI-IIは、述語論理型言語を使用した知識ベースの高速実行を目指しており、大容量実メモリと高速インタプリタを特徴とする。現在、DNA配列検索知識ベース・システムKNOAをCHI-II上で構築中である。本稿では、ベンチマークプログラムを使用した性能評価結果について報告する。性能評価は、ECRC (European Computer-industry Research Center) が提案しているベンチマーク・プログラムを使用した。性能評価の結果、コンパイルしたコードの実行性能でSun4/280上のQuintus Prolog 2.3版の1.5倍、インタプリタの場合は6倍の結果を得た。

2. アーキテクチャ

CHI-IIのシステム構成を図1に示す。遠隔手続き呼び出し機能を用意することで、CHI上のプロセスがホストマシンの機能を自由に使用できる環境を実現した。ファイル・システム、入出力装置の操作、グラフィック機能などを遠隔手続き呼び出し機能で実現している。述語論理型言語実行系のアーキテクチャは、D.H.D.Warrenが提案したWAM (Warren Abstract Machine)^[3]を基に、大規模知識ベース・システムの高速実行を目指し、以下の拡張を行った。

- 多重プロセス環境の実現
- 割込み/トラップの実現
- 多重名前空間の導入
- インタプリタの高速化

多重プロセス環境 CHIシステムでは述語論理型言語を実行する為に、4種の独立したメモリ領域を使用する。プロセスは、述語論理型言語実行用として、そのプロセス専用領域を4個与えられる。この他に、CHI上の全てのプロセスが共有する領域が4種ある。論理メモリ空間を図2に示す。全てのプロセスが共有する4種の領域は以下のものである。

- システム情報領域： 例外/割込処理用ベクター・テーブル、組込述語管理テーブル、入出力用バッファを格納する。
- システム・コード領域： オペレーティング・システム、インタプリタなどの実行コードを格納する。
- 共有データ領域： システム・パッケージなどを格納する。

システム・オブジェクト領域： オブジェクトの生成/消去に必要なメモリを提供する。

上記4領域の他に、プロセスが述語論理型言語の実行に使用する4種の領域がある。

トレイル・スタック領域： undo情報を格納する。

ローカル・スタック領域： 環境フレーム、選択点フレームを格納する。

グローバル・スタック領域： リスト、構造体などを格納する。

ヒープ領域： ユーザ・パッケージ、ユーザ・プログラムの実行コードなどを格納する。

CHI-IIシステムが使用する論理アドレスは、図3に示す様に、プロセス識別番号、領域番号と領域内アドレスの3フィールドからなる。

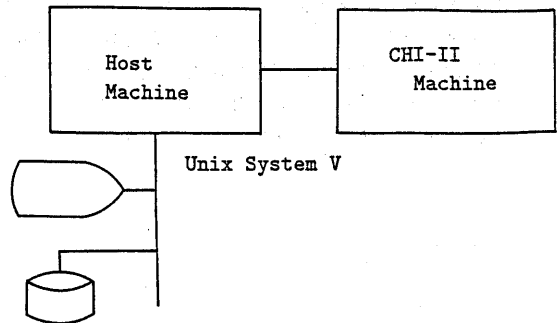


図1 CHI-IIシステムの構成

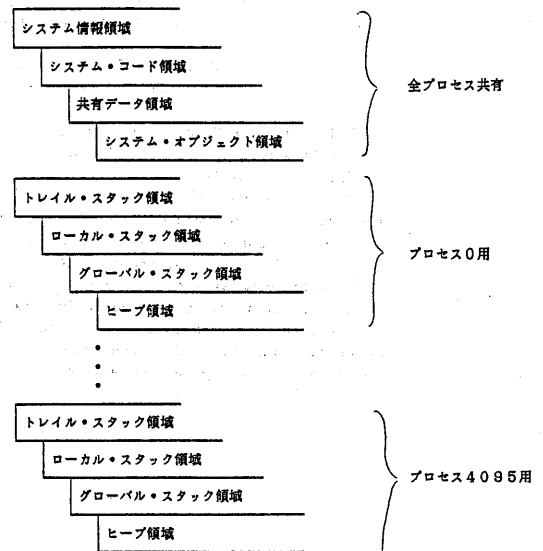


図2 論理メモリ空間の構成

Process No.	Area No.	Inner Address
-------------	----------	---------------

図3 論理アドレスの形式

割り込み/トラップ 割り込み/トラップとして、スタック伸長に伴うメモリの動的割付け、入出力割込み、バインドフック・トラップ等がある。割り込み/トラップは、機械語命令と機械語命令の間でトラップ処理を行う機械語命令レベル・トラップと述語と述語の間でトラップ処理を行う述語レベル・トラップがある。割り込み/トラップ要因の検査による処理速度の低下を最小に抑える為、割り込み/トラップの検出をハードウェア化した。トラップ処理の起動タイミングが機械語命令レベル・トラップと述語レベル・トラップで異なる為、割り込み/トラップに要求レベルを設定し、機械語命令に割り込み/トラップ要求に対する受けレベルを設定し、起動タイミングの選択を可能にした。割り込み/トラップの要求レベルと機械語命令の割り込み/トラップ受けレベルを使用した割り込み/トラップ処理起動の流れを図4に示す。機械語命令レベル・トラップは要求レベルが全ての機械語命令の受けレベルより高い為、要求を出した時に実行していた機械語命令の処理が終了すると直ちにトラップ処理を開始する。一方、述語レベル・トラップは要求レベルが"2"と低い為、call, execute など受けレベルが低い機械語命令の実行直前まで、トラップ処理の起動が抑制される。

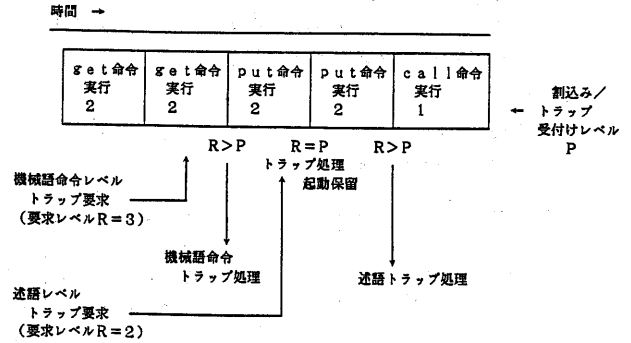


図4 割り込み/トラップ処理の起動タイミング

表1. インタプリタのために追加した機械語命令

多重名前空間 モジュール間でのシンボル名の衝突を回避する為、シンボル名を管理する一種のテーブルであるパッケージを使用した多重名前空間を導入した。パッケージには、全てのプロセスが共有するシンボル名を管理するシステム・パッケージと、プロセス固有のシンボル名を管理するユーザ・パッケージがある。ユーザ・パッケージは、1つのプロセスの中で複数個使用でき、プロセス間でのシンボル名の衝突回避の他に、プロセス内のモジュール間でのシンボル名の衝突回避にも有効である。システム・パッケージは、全てのプロセスが共有する共有データ領域に格納する。ユーザ・パッケージは、各プロセスのヒープ領域に格納する。

機械語命令名
ctry
cretry
ctrust
itry
iretry

インタプリタの高速化 クローズ・データベース機能を使用した知識ベース・システムの実行を考えると、コンパイルしたコードの実行速度ではなく、インタプリタの実行速度が重要となる。インタプリタを高速化する為、インクリメンタル・コンパイル方式を採用し、インタプリタ用機械語命令を追加した。追加した機械語命令を表1に示す。追加した機械語命令は、WAMの実行制御命令 (try, retry, trust) に実行コードとソースコードを管理する機能を付加している。インタプリタが使用する実行コードの形式を図5に示す。

ディスクリプタ
述 語 名
制 御 情 報
ctry_me_else
コンパイルしたコード
ソース・イメージ

図5 インタプリタが使用する実行コード

3. ハードウェア上の特徴

CHI-IIマシンの構成を図6に示す。ホストマシンとの通信機能を提供するホスト・インターフェース・ユニット(HIU)、メモリシステムとプロセッサから成る。メモリシステムは320メガバイトの大容量実メモリを備え、大量のワーキング・メモリを必要とする応用プログラム及び大規模クローズ・データベースの高速実行を可能とする。実メモリの管理はページング方式で行う。

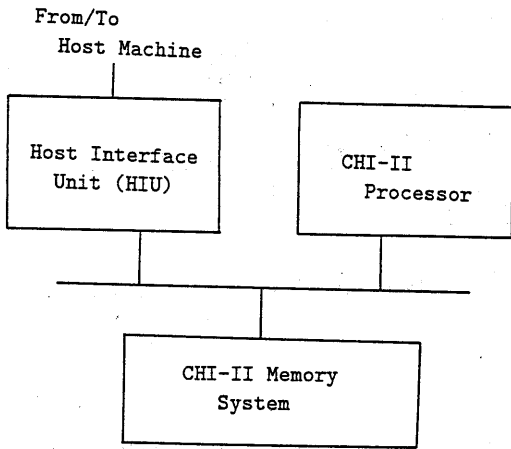
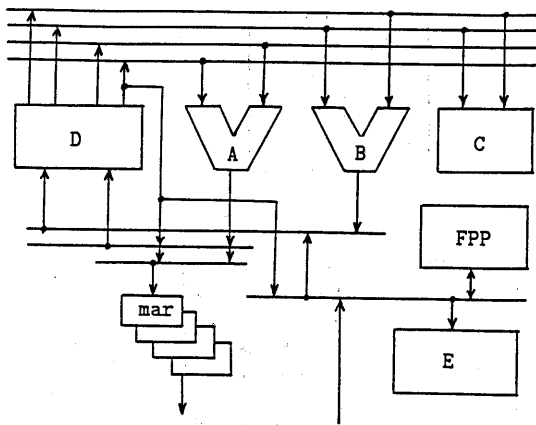


図6 CHI-IIマシンの構成



Cache Memory &
CHI-II Memory System

- A : ALU for address calculation
- B : ALU for data computation
- C : argument comparator
- D : multiport register file
- E : instruction prefetcher
- FPP : floating point processor
- mar : memory address registers

図7 CHI-IIプロセッサ EXU部の構成

プロセッサ EXU部の構成を図7に示す。マイクロプログラム制御方式のプロセッサで、水平型マイクロ命令を採用し、マイクロ操作の並列実行による高速化に重点を置いている。ハードウェア設計上の留意点を以下に示す。

- 機械語命令のパイプライン処理
- マイクロ操作の並列処理
- タグの生成/変更処理
- タグの検査処理
- 条件分岐の高速化
- レジスタの構成

機械語命令のパイプライン処理 機械語命令の処理は、読み出し、解読、実行の3段のパイプラインで行う。解読段では、オペレーション・コードの解読、レジスタ番号/ディスプレイメントの切り出しの他に、割り込み/トラップの検出、無条件分岐命令における分岐先番地の計算、unify系命令の実行モードの判定などを行い、実行段の処理負荷を軽減する。解読段で行う割り込み/トラップの検出処理は、処理待ちの割り込み/トラップの要求レベルと解読段で処理中の機械語命令の割り込み/トラップ受付レベルを比較する。unify系命令はreadモードとwriteモードの2つの動作モードを持ち、実行時に動作モードを決定する。この動作モードの決定を解読段で行う。CHI-IIの機械語命令コードの形式を図8に示す。オペレーション・コードを10ビット長とし、組み込み述語の機械語命令化、複合命令導入等による機械語命令の個数増加を可能にした。現在、235個の機械語命令が存在する。

tag	OPcode	Ri	Rj	Disp 8
tag	OPcode	Ri	Rj	Immed 8
tag	OPcode	Ri	Rj	
tag	OPcode	Ri	Rj	Rk
tag	OPcode	Ri		Rel 16
tag	OPcode	Ri		Immed 16
tag	OPcode	Ri		Disp 12
tag	OPcode	Disp 8		Rel 16
tag	OPcode	Disp 8		
tag	OPcode			

図8 機械語命令コードの形式

マイクロ操作の並列処理 78ビット長の水平型マイクロ命令を採用し、マイクロ操作の並列処理による高速化を目指した。マイクロ操作の並列処理の為、2個の算術演算器と多ポート（読み出しポート4個、書き込みポート2個）のレジスタファイルを採用し、スタック・ポインタの更新とデータの比較を同一ステップ内で実行可能にした。マイクロ操作の並列処理機能は、WAMが提案していた命令セットでは十分発揮されない為、インデクシング系とリスト/構造体のユニフィケーション系命令に複合命令を追加し^[4]、処理の高速化を行った。

タグの生成/変更処理 実行時のデータタイプ検査を高速化する為、タグアーキテクチャを採用した。CHI-IIのデータは図9に示す様に8ビットのタグ・フィールドと32ビットのバリュウ・フィールドから成る。データ・タイプの変更などタグ・フィールドの操作を高速化する為、タグの生成/変更用ハードウェアを用意した。タグの生成/変更機能は、メモリへのデータ書き込みパスとレジスタ間のデータ転送パスに用意した。機能としては、タグ値の読み出し、マイクロ即値を使用したタグ部への書き込み、レジスタの値のタグ部への書き込みがある。メモリへのデータ書き込みパスを使用する場合、書き込みデータのタグ部にマイクロ即値を書き込むことが可能である。この機能は、ローカル・スタック上の環境フレーム内に局所変数を生成する時、変数同士の束縛で変数間のチェーンを作る時などに使用する。レジスタ間のデータ転送パスを使用する場合、マイクロ即値を使用したタグ部への書き込み、タグ部の読み出し、レジスタの値のタグ部への書き込みが可能である。マイクロ即値を使用したタグ値の書き込みは、リスト、構造体、変数へのポインタを引数レジスタに設定する時に使用する。タグ値の読み出し、レジスタの値のタグ部への書き込みは、タグ操作用組込述語で使用する。

タグの検査処理 実行時のデータタイプ検査はユニフィケーション処理、組込述語の中で必要となる。データタイプ検査を高速化する為、タグ部を使用した多方向分岐機能をマイクロ操作に用意した。タグ部による多方向分岐マイクロ操作は2種類ある。第1の機能は、1つのデータのタグ値による多方向分岐機能である。第2の機能は、2つのデータのタグ値とバリュウ値の組合せによる多方向分岐機能である。各マイクロ操作は、高速大容量SRAMに登録した分岐パターンに従い、図10に示す様な多方向分岐または次の機械語命令のマイクロルーチンへの分岐を実行する。登録できる分岐パターン数は、1つのデータのタグ値を使用する多方向分岐では最

大128パターン、2つのデータを使用する多方向分岐では最大4パターンである。

条件分岐の高速化 述語論理型言語の実行をマイクロプログラム・レベルでみると、条件分岐操作の実行頻度が高い。ユニフィケーション処理におけるデータタイプの検査の他に、ユニフィケーションで変数に値を束縛する時に、バックトラック処理に備え、束縛前の値をトレイル・スタックに保存するかを調べる検査がある。また、バックトラック処理では、束縛変数を束縛前の状態に戻すundo処理を行うかを調べる検査などがある。CHI-IIのマシン・サイクル時間決定において、条件分岐操作の扱いが重要な検討項目であった。2つの案があり、第1案は、演算操作とその結果による分岐操作を別のマイクロステップで実行する方式である。第2案は、演算操作とその結果による分岐操作を同一マイクロステップで実行する方式である。マシンのクロック時間検討の結果、第2案は、マシンのクロック時間が第1案の1.5倍位に成るが、マイクロステップ数減少の効果により、全体の処理時間は第1案より短いと判断した。

第2案採用によるマイクロステップ数減少の効果を高める為、条件分岐操作と、算術演算器を使用したポインタの更新、メモリ・アクセスなどの同時実行を可能とした。

レジスタの構成 多ポート、64個のレジスタから成るレジスタファイルを採用し、述語引数などのデータ及び述語論理型言語の実行に必要な制御情報をレジスタ常駐にした。引数レジスタとして32個のレジスタを割り当てた。述語論理型言語の実行に必要な制御情報を保持する為に22個のレジスタを割り当てた。制御情報のレジスタ常駐化によりレジスタとメモリ間のデータ転送を減らした。

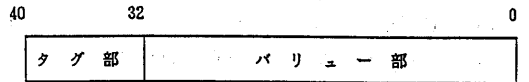


図9 内部データ形式

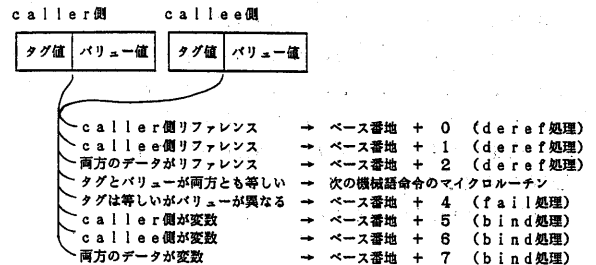


図10 タグ値を使用した多方向分岐パターン

メモリアドレス・レジスタを4個用意した。この結果、リストまたは構造体データを処理する時の構造体ポインタ (SP) をメモリアドレス・レジスタに常駐でき、図11に示す様に、unify系命令のマイクロステップ数を減らした。

4. 性能評価

処理性能を評価する為、ECRC (European Computer-industry Research Center) が提案するベンチマークプログラムを使用した。ECRCのベンチマークプログラムは、述語論理型言語の基本機能を評価するプログラム群とどの様な処理系でも実行できる簡単なプログラム群から成り、基本機能の処理性能評価の他、応用プログラムを実行する場合に期待できる処理性能の指標を得ることができる。

基本機能の評価として、以下の16個の基本機能を取り上げている。

述語呼び出し処理速度

引数3個の選択点フレーム生成処理速度

引数無しを選択点フレーム生成処理速度

深いバックトラック処理速度

浅いバックトラック処理速度

変数3個の環境フレーム生成処理速度

変数無し環境フレーム生成処理速度

候補節が21個ある時の処理速度 (インデクシング機能)

リストのユニフィケーションによる生成処理速度

リストのユニフィケーションによるマッチング処理速度

構造体のユニフィケーションによる生成処理速度

構造体のユニフィケーションによるマッチング処理速度

ディリファレンス処理速度

カット処理速度

評価では、コンパイルしたコードの実行速度とインタプリタを使用した場合の実行速度を測定した。参考として、コンパイルしたコードの実行速度については、Sun4/280上のQuintus 2.3版を測定した。インタプリタについては、Sun4/280上のQuintus 2.3版、VAX8650上のC-Prolog、VAX8650上に構築したCHIと同方式の処理系の実行速度を測定した。

測定した結果を表2に示す。

5. 考察

コンパイルしたコードの実行速度の測定結果について、Quintusを1とした場合のCHI-IIの実行速度を図12に示す。基本処理の実行速度では、QuintusがCHI-IIより早い基本機能が存在する。しかし、簡単な応用プログラムの実行速度では、CHI-IIが早いことが判る。特に、fibonacci, differenでその差が開いている。fibonacci, differenにおける差は、index機能の差と考えられる。fibonacci, differenにおけるswitch系命令の実行頻度を図13に示す。switch系命令の実行頻度が高く、index機能の差が全体の速度に大きな影響を与えているのが判る。

表2. ECRCベンチマークプログラム実行結果

[単位: KLIPS]

ベンチ マーク プログラム 名	CHI-II		C-Prolog VAX8650	quintus ver.2.3 Sun4/280		CHI方 式の処理 系 VAX8650
	コンパ行	インタプリタ		コンパ行	インタプリタ	
	boresea	2487	327	20.0	767	10.5
opoint	333	63	10.9	121	1.6	26.3
cpoint0ar	400	103	20.0	105	4.7	31.7
baktrak1	156	75	6.4	138	6.3	10.7
baktrak2	200	133	21.6	213	24.0	24.7
envir	194	87	11.3	227	7.3	15.9
envir0ar	621	230	22.1	299	9.6	22.4
index	64	22	3.7	27	4.8	8.5
const_lst	107	73	12.6	47	7.1	11.1
match_lst	154	78	6.6	130	6.4	10.9
const_str	84	62	12.8	47	7.2	10.3
match_str	112	65	6.8	85	6.3	10.8
wth_n_str	4	3	0.5	3.2	0.7	0.4
gen_unif	0.7	0.7	0.5	1.4	1.2	0.5
deref	6	6	1.5	2.6	2.4	1.7
cuttest	234	98	14.4	220	6.9	24.2
fibonacci	147	75	5.7	59	5.5	12.6
map	85	18	4.3	41	4.8	9.9
mham	92	41	4.7	75	6.5	9.3
mutest	92	40	6.5	66	5.8	8.4
qs	122	48	6.1	78	5.6	9.7
qu	153	46	5.4	120	6.6	9.9
query	73	30	2.7	65	5.7	6.5
differen	54	22	4.0	14	4.5	7.8
diff	108	46	6.0	69	5.3	10.1

```

get_structure : marlにSPポインタをセット
unify_atom   : marlを使用し、構造体の要素を読み出す
               marlをインクリメント
unify_variable : marlを使用し、構造体の要素を読み出す
               marlをインクリメント
unify_value  : marlを使用し、構造体の要素を読み出す
               marlをインクリメント

```

図11 SPポインタのMAR常駐化による unify系命令の高速化

gen_unifは、変数に束縛しているネストした構造体同士のユニフィケーション処理である。この種の処理の発生は非常に少ないと考えられる為、CHI-IIではトラップ処理にした。この為、gen_unifの処理がQuintusより悪くなっている。

バックトラック処理の速度を求めるdback, sbackの実行速度の測定結果は、QuintusとCHI-IIが同程度であるが、fibonacci, differenで見られたindex機能の差により応用プログラムの実行においてCHI-IIが有利と考えられる。

汎用機上の処理系は中間コードに対応するルーチン呼び出して実行するThreaded-code方式を採用している為、図12からハードウェアシステムの性能(プロセッサとメモリスステムの性能)比較することは難しい。しかし、derefは図14に示したポインタ・チェーンをたどるディリファレンス処理の

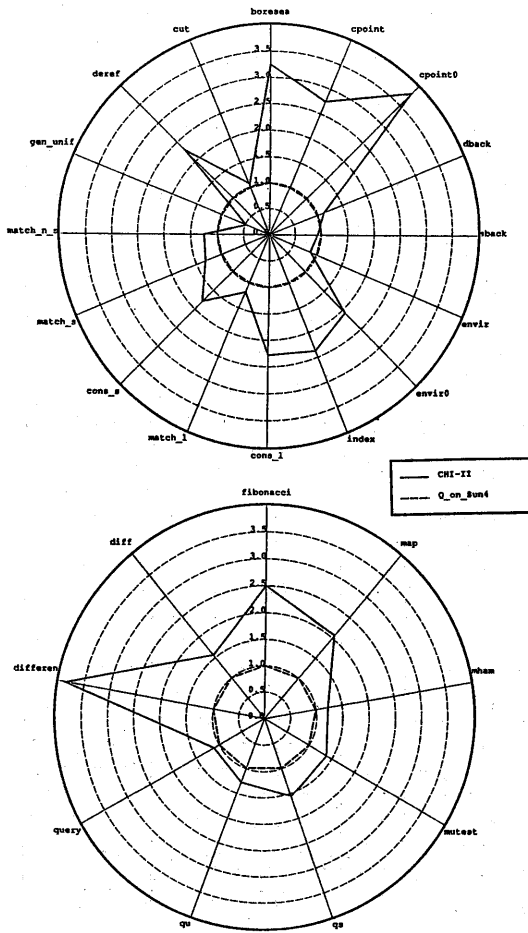


図12 コンパイルしたコードの実行速度比較

速度を求めるベンチマークプログラムで、中間コード呼び出しの時間が含まれないと考えることができる。derefについて、SUN3/160, SUN3/260, SUN4/280, CHI-IIで実行速度を求めた結果を図15に示す。この結果は、ハードウェアシステムの性能を判断する参考になると考えられる。

インタプリタの実行速度測定結果について、Quintusを1とした場合のCHI-II、C-Prolog、VAX上のCHI方式の処理系の実行速度を図16に示す。VAX上のCHI方式の処理系がQuintus, C-Prologより早いことからCHIで採用している処理方式が優れていることが判る。簡単な応用プログラムの実行速度で比較すると、CHI-IIはQuintusの約6倍の処理速度を実現していることが判る。

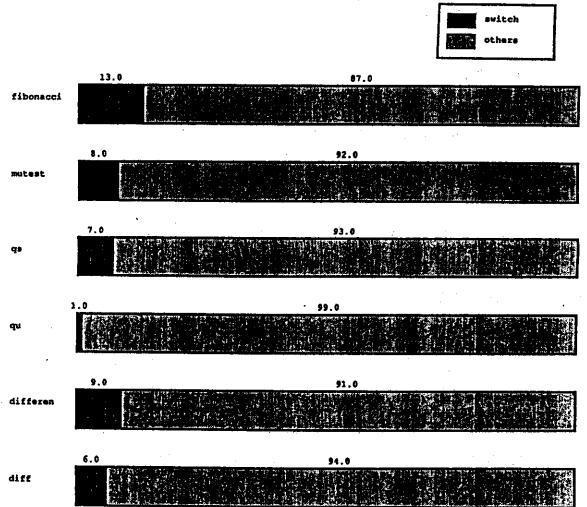


図13 fibonacci, differen における switch系命令の実行頻度

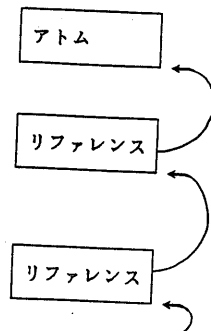


図14 deref が求めるポインタ・チェーン

6. まとめ

述語論理型言語を使用した知識ベースの高速実行を目指して研究開発して来た逐次型推論マシンCHI-I-IIの性能評価を行った結果、応用プログラムの実行において、コンパイルしたコードの実行速度でQuintusの約1.5倍、インタプリタの性能でQuintusの約6倍であることを確認した。知識ベースシステムでは、クローズデータベース機能を使用し、assert/retractによるクローズの追加/削除操作を行う為、インタプリタの性能差は大きな意味を持つ。現在、DNA配列検索知識ベースシステムをCHI-II上に構築中で、約100メガバイトのデータをCHI-II上に格納する予定である。

謝辞： 最後に、本研究の機会を与えて下さったICOT第4研究室の内田室長、日本電気 C&Cシステム研究所の石黒所長、同所 コンピュータ・システム研究部の大野部長に深謝致します。

参考文献：

- [1] R.Nakazaki, et al. "Design of a High-speed Prolog Machine (HPM)", Proc. of the 12th International Symposium on Computer Architecture, June 1985
- [2] S.Habata, et al. "Co-operative high Performance Sequential Inference Machine:CHI", Proc. of ICCD'87, 1987
- [3] D.H.D.Warren, "An Abstract Prolog Instruction Set", Tech. report 309, Artificial Intelligence Center, SRI International, 1983
- [4] 幅田 他、「逐次型推論マシン：CHI」、記号処理研究会 42-1, 1987. 9
- [5] 小長谷 他、「高速Prologインタプリタの構築法とその評価について」、記号処理研究会 46-4, 1988

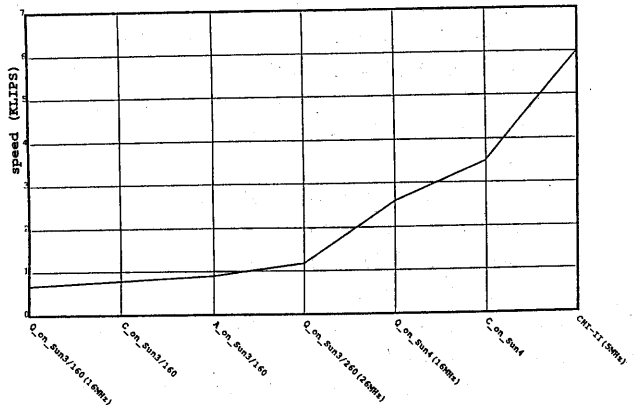


図15 derefの実行速度比較

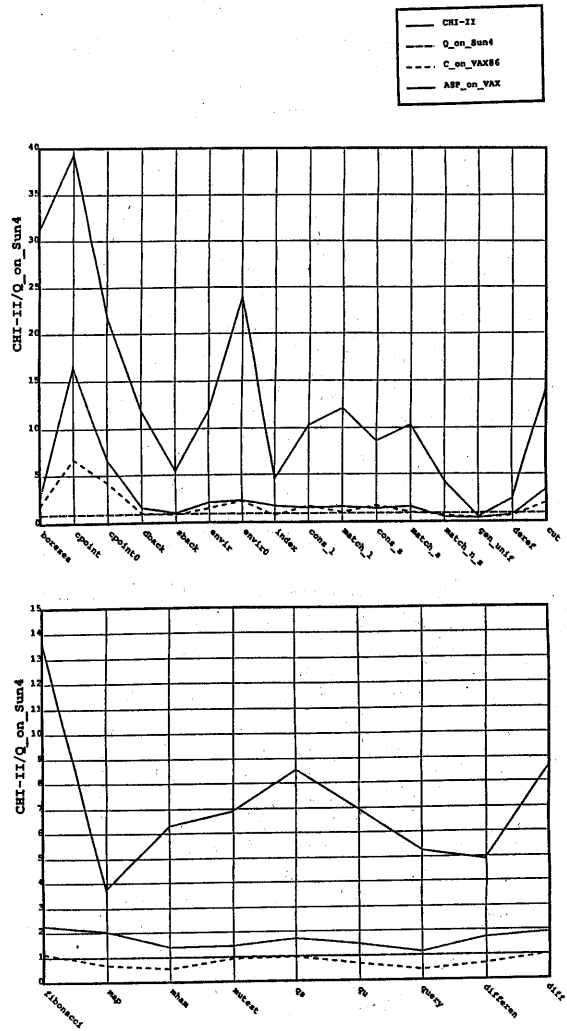


図16 インタプリタの実行速度比較