TM-0119

情報処理学会第31回全国大会
発表論文集

ICOT および関連機関

September, 1985

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

9V-5

prologによる定理証明機の試作

南　俊朗
(富士通・国際研)

1．はじめに

Prolog言語を使用した一階述語論理の定理証明機を試作した［Fujitsu 85］ので報告する。prologによる定理証明機の例として順序付き線形法によるもの［Mukai 83］，鍵付き法によるもの［Fujitsu 84］等がある。本証明機は結合グラフ法［Kowalski 79］，［Sickel 76］に基づいたものである。結合グラフ法はリテラルをノードとし使われる可能性のあるリテラル間をリンクで結んだグラフの形で節集合を表現し，導出に利用しようというものであり，あらかじめ分解導出に利用できるか否かを調べておくため，実際の導出時にチェックする手間を大幅に省くことが期待できる。また何らかの基準によってリンクの一部分を取り去ることで解探索時の戦略の大枠を予め設定することもできる。本証明機はSickelに従って結合グラフ上を解を求めて移動する方式を採用した。

以下、2節で証明機の概要を述べ，3節でprolog言語の特徴を活かした実現上の工夫例を紹介する。4節で鍵付き証明機との簡単な比較を行う。5節で本証明機に対するまとめと今後の課題について述べる。

2．証明機の処理概要

本証明機の処理は大きく次の4ステップに分けられる。

(1)節形式変換部
　与えられた前提，結論を節形式に変換する。

(2)結合グラフ作成部
　節形式にリンク情報を付加し結合グラフを作成する。

(3)解探索部
　結合グラフを探索し解の木を求める。

(4)導出部
　解探索の木より実際に分解導出を行う。

処理の過程の例を次に示す。

［例］

与えられた問題

前提条件：$\forall x(x\in A\Rightarrow x\in B)\Rightarrow A\subseteq B$

$\forall x(\sim x\in\phi)$

結論：$\forall M(\phi\subseteq M)$

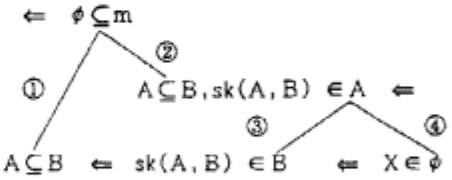
(1)の結果

$A\subseteq B, sk(A,B)\in A \Leftarrow$

$A\subseteq B \Leftarrow sk(A,B)$

$\Leftarrow X\in\phi$

(2)の結果



(3)の結果

(4)の結果

$\Leftarrow \phi\in m \qquad A\subseteq B, sk(A,B)\in A$

$sk(\phi,m)\in\phi\Leftarrow \qquad \Leftarrow X\in\phi$

3．実現上の工夫

prolog 言語の持つ特長を活かした例を2つ紹介する。

（i）単一化機構

prolog は単一化機構を利用した部分データを使えるのが大きな特長である。節形式変換部のスコーレム関数の生成で利用している。閉式 $\forall x\exists y\,\phi(x,y)$ をスコーレム化すると $\phi(x,sk(x))$ が得られる。∃で束縛された変数yはその外側から∀で束縛している変数xを引数とする関数で置き換えられている。閉式でない $\forall x\exists y\,\phi(x,y,z)$ の場合φの中に現れる自由変数．は一番外側から∀で束縛されているものと考えられる。すなわち $\forall z\forall x\exists y\,\phi(x,y,z)$ という式であると考えてスコーレム化を行うため式 $\phi(x,sk'(x,z),z)$ を得る。外側から内側へと再帰的にスコーレム化を行う手続きはφを処理するとき式の中に同様な自由変数が含まれているか分からないため通常は予め式をスキャンし自由変数を取り出し式を閉式とした後スコーレム化を行う。本証明機では自由変数に対応するスコーレム関数の引数部を不定にしたままスコーレム化を行っている。上記のφの例ではφの中のyはsk(x,...) という関数で置き換えておく。節形式変換部はスコーレム化と同時に自由変数の拾い出しを行っており，全体の変換処理後確定した自由変数のリストをスコーレム関数の未確定部分に一斉に代入することで最終的なスコーレム化を実現している。

/

（ii）後戻り機構

prologは特殊化した論理系の定理証明機であるため，後戻り機構などの定理証明機として必要な機能がシステムに初めから備えつけられているため，定理証明機の解探索時に必要な後戻りを自然な形で実現できる。前述の例の場合，たとえば初め①のリンクによる単一化を行うと次のゴールはsk(φ,m) ∈m となる。ところがリンク③を使おうとしてもゴールが行き先のsk(A,B)∈A の形になっていないため③は使えないことになる。また③以外のリンクも存在しないため現在のゴールを反駁することはできない。そのため以前の選択点に戻り別のリンクを探すことになる。この後戻りの制御はprologの後戻り機構をそのまま利用できるためprologによる解探索のプログラムは非常に簡単なものとなる。

4．健付き定理証明機との比較

本証明機は健付き定理証明機［Fujitsu 84］と類似の使い方になっており節形式変換部等の共通部分は改良版となっている。証明部分は方式の違いのため異なっている。節形式変換部の改良点は健付き法は①自由変数を見つけ全称束縛して閉式にする②⇒，⇐，⇔を除き，￢を内側へ移動する③変数名をつけかえる④限量子を外側へ移す⑤スコーレム化⑥節形式への変換と，式を部分式に分解する操作が5ステップ必要であるのに対し本証明機は変換ステップと部分形であるスコーレム関数の引数への自由変数リストの代入ステップ等により節形式に変換しているため部分式分解操作は1ステップで済んでいる。その結果結合グラフ法の処理時間は健付き法と比べて，簡単な式に対しては70%～100%，複雑な式に対して30% ～60% と高速化されている。しかし結合グラフの作成に大きな時間が必要なため解探索の準備段階全体としては50% 程度余計に時間がかかっている。健付き法の解探索法は節を構成するリテラルに番号を付けておき，番号の小さいリテラル間で分解導出を行うという制限戦略であり，結合グラフ法の考えかたと原理的な優劣の比較はできない。実際，両者に同一問題を解かせてみたところ問題により一方には解けて他方には解けないということが起こった。健付き法が横型探索であり現在の結合グラフ法証明機が縦型探索であることより，健付き法が優れている問題のタイプには組み合わせ可能な節の組みは多くあるが実は少数の組み合わせで解が求まる問題がある。逆に結合グラフ法が得意とする問題は表面的に見た節の組み合わせは多いがその中の特定の組み合わせだけが解につながるようなものである。

両者に共通の弱点として偶然に左右されやすいことがある。それぞれ番号付け，結合リンクの順序により解探索の効率が大きく左右されてしまう。

5．おわりに

結合グラフを利用した定理証明機を試作した。prologの処理系の特長である単一化機構，後戻り機構をうまく利用することでプログラミングの手間，効率が大きく改善できる。健付き法の証明機と比べると一長一短である。節形式変換部は高速化された。結合グラフ法はリンクの付け方をどのように制限するかによって解探索の戦略の大枠を表現することができる。探索時のリンクの選択の際の戦略と結びつけることでより一層知的な定理証明機が実現できるものと考えられる。

以上の考察より今後の改良点として，

（i）何等かの横型探索機構を実現すること。

（ii）結合グラフ作成時の戦略表現，解探索時の戦略表現を組み合わせて総合的にすぐれた戦略を探すこと。

等が考えられる。

6．謝辞

本研究は，第5世代コンピュータ・プロジェクトの一環として行われたものである。本研究にあたって国際研の北川会長，榎本所長，竹島室長，加藤研究員の援助に感謝します。

参考文献

［Chang 73］ C.L.Chang, R.C.T.Lee:Symbolic Logic and Mechanical Theorem Proving, Academic Press,1973.

［Fujitsu 84］ 富士通㈱：58年度電子計算機基本技術開発再委託成果報告,5G核言語第1版の検証用ソフトウェア詳細仕様書,Part Ⅲ 一階述語論理の定理証明機,1984.

［Fujitsu 85］ 富士通㈱：59年度電子計算機基本技術開発再委託成果報告,5G核言語第1版の検証用ソフトウェア詳細仕様書(第2版) 及び試験評価データ,Part Ⅱ 定理証明機,1985.

［Kowalski 79］ R.Kowalski:Logic for Problem Solving, Artificial Intelligence Series 7, North Holland,1979.

［Mukai 83］ K.Mukai, K.Furukawa:An Ordered Linear Resolution Theorem Proving Program in Prolog,ICOT Technical Report,TM-0027,Sep. 1983.

［Sickel 76］ S.Sickel:A Search Technique for Clause Interconnectivity Graphs,IEEE Trans. on Computers, Vol. C-25,No.8,Aug. 1976.

Proofreading Japanese Word Usage and Proper Nouns by Computer

44-5

Satoru Ishii

Information and Communication Systems Laboratory
TOSHIBA Corporation, Kawasaki, Japan

1. INTRODUCTION

A Japanese Proofreader System for Japanese text has been constructed. [4] [1] [2] [3] This report describes knowledge used and the experimental results. The system is written in DEC10 Prolog on the DEC2060.

2. WORD USAGE CORRECTION

Proofreading to correct word usage was conducted in three steps.

In the first step, input data was converted to a list format, in which each component of the list corresponds to one character in JIS KANJI codes.

The second step is the proofreading step. In this step, a sentence input in the list format is divided into words, which are output in list format. Each component of the output list is information regarding one word.

The knowledge in use was derived from a dictionary[5], Common KANJI Table [7], and the stylebook being used at the Asahi Shimbun Publishing Co. [6]. [3] Proofreading is performed by recursively seeking the predicate for distinguishing words by means of the longest-match method.

In the third step, the proofreading result is indicated. Knowledge based upon the Common KANJI Table and the stylebook used at the Asahi Shimbun Publishing Co. was used.[3]

The result of executing the program is as follow. Sentence examples were chosen as much as possible from data on the newspaper articles during the period from July 1 to 10, 1984, received from the Asahi Shimbun Publishing Co. However, they were not the same as the actual sentence which appeared in the paper.

In this experiment, because of limitations in the processing system, about 10,000 lines of knowledge, that is, about one tenth of the entire volume was used.

Figure 1 shows the result of analyzing the sentence "足下から雨が降る。" (ASHIMOTO KARA AME GA FURU: It rains from the footstep.) The system responds at two points. First, it indicates that "足元" should be used. Second, it handled the noun "雨" (AME: rain) as an undefined word.

3. CORRECTING PROPER NOUNS

The following describes an example

```
INPUT SENTENCE      足下から雨が降る。

RESULT OF PROOFREADING
WORD, PART OF SPEECH, STEM, INFLECTION, ERROR?, REWORD?, TITLE NO

足下    NOUN      足下          --      REWORD      730
    BASE TO REWORD
    SOURCE
        STYLEBOOK, WORD USAGE      245  PAGE
        COMMON KANJI TABLE         8  PAGE
    PARAPHRASE
        足元
から    POSTPOSITIONAL      から          --      --      0
雨    NOUN(ESTIMATED)      雨          --      --      --
が    POSTPOSITIONAL      が          --      --      0
降る    VERB      降      SHUSHI      --      --      40977
。    PUNCTUATION      。          --      --      0
```

Fig. 1 Proofreading for Word Usage

This work was carried out as a part of the Fifth Generation Computer Project while the author was at the Institute for New Generation Computer Technology (ICOT). The author wishes to thank ICOT for the opportunity to pursue this research and permission to present this report.

of proofreading articles to report personnel changes concerning directors working at companies listed on the stock exchange.[2] Also, an example of handling KANJI characters, which are not designated as Common KANJI, is described.

The following knowledge was used in the experiment.

Listed companies knowledge concerns the names of all companies listed on the stock exchange. It was created using the database of the Toyo Keizai Shinposha.[8] There are about 4400 lines.

Knowledge concerning directors of listed companies includes data on all directors of all listed companies. It was also created, based on the database of the Toyo Keizai Shinposha. There are about 32,000 lines.

KANJI knowledge is knowledge concerning Common KANJI and the ways they are read.[3] There are about 4100 lines.

Owing to restrictions in the processing system, only about one tenth of the knowledge was used in the experiment. Moreover, a short program to analyze only articles in the specific format used for personnel changes was used.

The example sentence was extracted from newspaper articles mentioned in Section 2. However, as noted before, it is not exactly the same as the actual sentence which appeared in the paper.

The result of the example is shown in figure 2. "十条製紙", the name of the company, cannot be found when the system is executed normally. This is because the name of the company is included in the knowledge only as "十條製紙" and "十条紙".

However, when KANJI other than Common KANJI are handled by a special procedure, the name of the company can be retrieved, as shown in the figure. This operation is valid for symbols other than KANJI, such as GETA symbols.

4. CONCLUSION

Knowledge used in the proofreading texts to check word usage and proper nouns and the results have been described. Due to limitation in the processing system, experiments were made with about one tenth of the entire volume of data, and the system was proven effective.

The author would like to express sincere gratitude to the Asahi Shimbun Publishing Co. and the Toyo Keizai Shinposha, who kindly offered their data and approved of the use of their material. Also, thanks are due to the Sharp Corp. for their help in inputting and editing the knowledge.

REFERENCES

[1] Ishii,S. : "Study of Proofreading Techniques Used at a Japanese Newspaper"、第２８回全国大会講演論文集（Ⅱ）2M-7、pp.1205～1206、情報処理学会（1984）

[2] Ishii,S. : "Study of a Newspaper's Treatment of Proper Names"、第２９回全国大会講演論文集（Ⅱ）4J-5、pp.1449～1450、情報処理学会（1984）

[3] Ishii, S. : "Knowledge for Proofreading Japanese Word Usage"、第３０回全国大会講演文集（Ⅱ）3G-3、pp.1635～1636、情報処理学会（1985）

[4] 石井 暁："計算機による用語・固有名詞の校正"、TR-124、（または、"Proofreading Word Usage and Proper Nouns by Computer"、TR-125）、ICOT（1985）

[5] 久松他："角川国語辞典" 63版、角川書店（1982）

[6] 朝日新聞社用語幹事編："朝日新聞の用語の手びき" 第18版、朝日新聞社（1984）

[7] 大蔵省印刷局編："常用漢字表"、大蔵省印刷局（1982）

[8] 東洋経済新報社編："役員四季報１９８５年版"、東洋経済新報社（1984）

```
INPUT SENTENCE      ◇十条製紙（６月３０日）専務（常務）遠藤健一郎

COMPANY CANDIDATE      十条製紙
DATE CANDIDATE       ６月３０日
NEW POST CANDIDATE       専務
OLD POST CANDIDATE       常務
NAME CANDIDATE       遠藤健一郎

COMPANY NAME EXISTS     ON THE MATERIAL    十條製紙      CODE NO.   3863
NEW POST EXISTS
OLD POST EXISTS
NAME EXISTS
POST OK
```

Fig. 2 Proofreading for Proper Nouns

並列推論マシンＰＩＭ－Ｒの ハードウェア・シミュレータ －システム構成－

1C-2

杉江　衛，米山　貢，坂部俊文，岩崎正明，吉住誠一（日立）
麻生盛敏，清水　肇，尾内理紀夫（ＩＣＯＴ）

１．はじめに

第五世代コンピュータ・プロジェクトでは、知識情報処理システム研究開発の一環として、そのハードウェアである推論マシンの研究を進めている。我々は、そのアーキテクチャの検証を目的として、並列推論マシンＰＩＭ－Ｒ[1]のハードウェア・シミュレータを試作した。本稿では、そのシステム構成について述べる。

２．システム構成

本ハードウェア・シミュレータの開発にあたっては、各種の評価が可能となるようにシステムの柔軟性を念頭に置いた[2]。我々はMC68000搭載のシングル・ボード・コンピュータ(SBC)を用いてＰＩＭ－ＲのInference Moduleを構成することにし、各種モジュールはSBC上のソフトウェアで実現して、変更改良に容易に追従できるようにした。

図1に試作したハードウェア・シミュレータの概念的構成を示す。ＰＩＭ－Ｒのアーキテクチャに従って、推論モジュール（IM: Inference Module)、ネットワークおよび構造体メモリ・モジュール　(NSM:Network and Structure Memory Module)、制御モジュール(CM:Control Module)の3つのモジュールから構成されている。

これらのモジュールの主な機能は次のとおりである。CMは言語処理系から成り、プログラムのコンパイル、IMの起動と結果の出力を行なう。NSMはCM-IM間の交信データのバッファ(CMバッファ)、IM-IM間の交信データのバッファ(IMバッファ)を行ない、長い構造体データの格納も受け持つ。IMは5つの処理系と3つの格納領域からなる。以下にその機能を記す。

（ａ）Nucleus

①CMからの指令割り込みに対する処理。

②PPU(Process Pool Unit)処理系の起動。

（ｂ）PPU処理系

①プロセスの生成、消滅およびReady, Wait,Suspendの状態遷移の管理。

②Ready Processからのゴールの切り出しと該ゴールのUU (Unification Unit)処理系への送信。

③NET処理系へのパケットの送信。

（ｃ）UU処理系

①ゴールと該当クローズのユニフィケーションによるリダクション。

（ｄ）NET処理系

①送信先のIMに対応したIMバッファへのパケットの書き込み。

②IMバッファ内のパケットの待ち行列管理。

（ｅ）SMU処理系

①長い構造体データのユニフィケーション。

（ｆ）Process Pool

①プロセスに関係したデータを格納する。これらのデータには、プロセスの状態、親プロセス、変数に関するデータが含まれる。

（ｇ）Clause Pool

①ソースプログラムを格納する。

（ｈ）Message Board

①プロセス間の同期用のチャネルに関するデータを格納する。

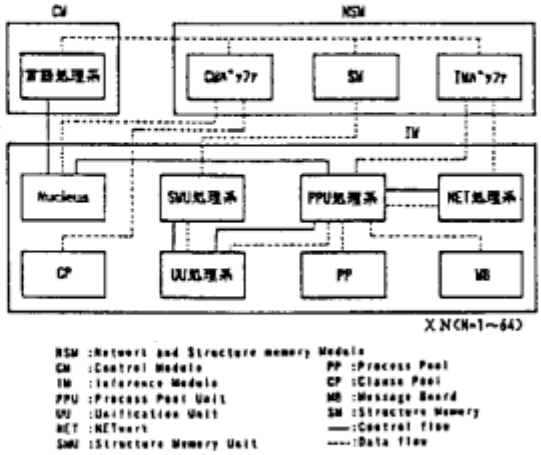


図1　ハードウェア・シミュレータの概念構成

図2にシミュレータのハードウェア構成を示す。IMはSBCとそのバスに接続されたローカル・メモリから構成される。NSMは8台のSBCがアクセス可能な共有メモリからなり、CMはVAX11/780からなる。NSMに対応する共有メモリは、共有バスに接続される。この共有バスには、共有メモリの他に、GPIBインタフェイスとSBC8台分のIMボードインタフェイスが接続され、CM-IM間、IM-NSM間の交信を実現する。VAX11/780はGPIBインタフェイスを介して共有メモリにアクセスし、かつ、目的のSBCに指令を送る。各SBCはIMボードインタフェイスを介して共有メモリにアクセスする。バスアービタは共有バスの所有権を制御することによって、8台のSBCとVAX11/780の共有メモリへのアクセスを可能にする。

NSMに共有メモリを用いたのは、各種のネットワークのシミュレーションを可能にするためである。IMボードインタフェイスは各SBCにローカル・メモリを所有させるために導入した。ローカル・メモリと共有メモリはアドレス空間を分割して実現し、共有メモリのアドレス空間に対して、SBCバスを共有バスに接続する。各SBCのアドレス・マップを図3に示す。ローカル・メモリは最大10MB、共有メモリは最大4MBの容量を持ち得る。試作したハードウェア・シミュレータでは、共有メモリに2MB、ローカル・メモリに30MB(2MB～6MB/SBC)を実装した。

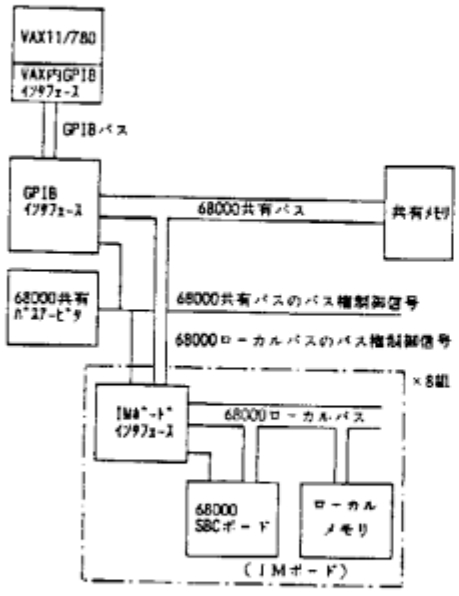


図2 シミュレータのハードウェア構成

3. 動作概要

CMは、まず、全IMをリセットする。ユーザ・ファイルより、プログラム、シミュレーション用パラメータおよび質問を読み込んで、前2者については、全IMのローカル・メモリにブロードキャストし、質問は特定の IM(#0) のバッファに書き込む。そして、これらの処理完了後、全IMに起動をかける。IMでは、Nucleusがパワー・オン・リセット又はCMからのリセットによって起動され、Clause Poolの先頭アドレス等を初期設定した後、CMからの起動を待つ。CMから起動されると、Nucleusは制御をPPU処理系に移す。PPU処理系は言語指定パラメータを判定して、Prolog処理モジュール、Concurrent Prolog処理モジュールのうちの一方を呼び出す。Prolog処理モジュール、Concurrent Prolog処理モジュールは、他PPUからのパケット処理ルーチンを呼び出して、リダクションを行なう。

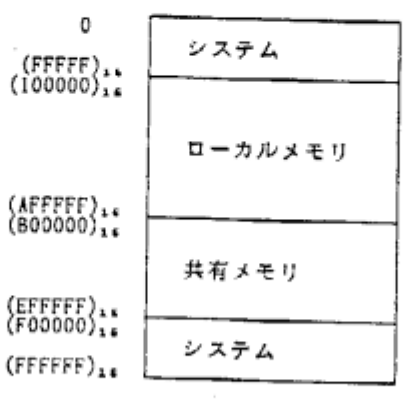


図3 SBC のアドレス・マップ

4. おわりに

今回の試作では、8台のSBCを実装し、各SBCは1台のIMをシミュレートする単純な形をとった。現在、SBC台数を16に拡張し、各SBC内に複数のタスクをインプリメントするように拡張をはかっている。これによってIMが100台規模のＰＩＭ－Ｒのシミュレーションを実行し、アーキテクチャの評価を行なう予定である。最後に、日頃御指導いただいた ICOT 村上 国男 前第1研究室長 および内田 俊一 第4研究室長に深謝する。

なお、本研究は第五世代計算機プロジェクトの一環として、ICOTの委託で行なった。

参考文献

1) 尾内 他；“リダクション方式並列推論マシンＰＩＭ－Ｒのアーキテクチャ”, The Logic Programming Conf. '85.2.1(1985)

2) 尾内 他；“並列推論マシンＰＩＭ－Ｒのハードウェア・シミュレータ -開発思想-”, 本大会予稿１Ｃ－１

並列推論マシンＰＩＭ－Ｒの ハードウェア・シミュレータ －制御プログラムの方式－

岩崎正明，杉江　衛，米山　貢，坂部俊文，吉住誠一（日立）
麻生盛敏，清水　肇，尾内理紀夫（ＩＣＯＴ）

１．はじめに

並列推論マシンPIM-R[1]のハードウェア・シミュレータを開発した。　開発にあたってはシステムの柔軟性を重視した。PIM-RのIM(Inference Module)には、MC68000を搭載したシングル・ボード・コンピュータ（SBC)を用い、各モジュールはソフトウェアで実現した。

本稿ではハードウェア・シミュレータの制御プログラムの方式について述べる。

２．全体構成

本シミュレータには、PrologとConcurrent Prolog処理系をインプリメントした。　本シミュレータは、　共有メモリを介した８台のSBCとホスト・マシンVAX-11/780で構成している[2]。　PPU(Process Pool Unit)、UU(Unification Unit)、NET(Network)等のPIM-Rの構成要素は、SBC上のソフトウェアとして実現し、Ｃ言語を用いて記述した。　又、シミュレータ内部の処理時間を計測するシミュレーション・クロックも ソフトウェアで実現している。　Prolog処理系はOR並列に処理を行ない、　Concurrent Prolog処理系はAND並列に処理を行なう。

３．PPU処理系

PPU処理系は、(1) 他PPUからのパケット処理ルーチン[PP_PKT]、(2) UUからのパケット処理ルーチン[PU_PKT]、　(3) レディ・プロセス処理ルーチン[RDYPRC]から構成する。

PP_PKTは、受信パケットの種類によって、reduction処理、子プロセス成功/失敗処理、activate処理、　MB(Message Board)read/write処理の何れかを行なう。

PU_PKTは、UUからのユニフィケーション処理結果パケットの種類によって、子プロセス生成処理、　子プロセス成功/失敗処理、suspend処理の何れかを行なう。　OR並列処理を行なうProlog処理系では、新たに生成された子プロセスの幾つかを他IMへ分配し、残りを自IM内のRPQ (Ready Process Queue)に登録する。

RDYPRCは、 RPQから取り出したプロセスの状態によって、reduction処理、retry処理、commit処理を行なう。　AND並列処理を行なうConcurrent Prolog処理系では、reduction処理においてAND関係の子ゴールをreduction要求パケットとして、自IMを含む複数のプロセッサへ分配する。

Concurrent Prolog処理系では、 commit処理のオーバーヘッド低減の為に、OR兄弟関係のプロセスは同一IM内に存在させ、OFCB(OR Fork Control Block)で管理している。　又、各プロセスは、PCB(Process Control Block)によって管理する。図1にOFCBとPCBの形式を示す。　OFCB内には、commit tag、 OR fork数等の制御情報を格納する。 PCB内には、プロセス状態フラグ、 親プロセス・ポインタ、AND fork数等の制御情報を保持する。

Control Block Type
Status
Commit Tag
OR Fork Count

▲OFCB

Control Block Type
Process Status
Flags
Simulation Clock
Reduction Level
Next　　PCB Pointer
Previous PCB Pointer
PTB pointer
Parrent Process Pointer
Parrent Process IM Number
OFCB Pointer
AND Fork Count

▲PCB

図１．　ＯＦＣＢとＰＣＢの形式

4. UU処理系

UU処理系は、PPU処理系から送られたパケットを入力としてユニフィケーション処理を行ない、結果パケットをPPU側へ返す。

ユニフィケーション処理では、structure copy方式の採用によるコピー処理増加に対処する為、変数の参照テーブルを用い、同一変数に束縛されたリスト等の重複コピーを避けている。しかしながら、今回のインプリメントではコピー処理が全処理の約70%を占めており、データ構造の改良等によるコピー処理の低減が必要である。

チャネル変数に対する処理は次の様に行なう。図2、3に例を示す。親側のチャネル変数に非変数が束縛される場合、この束縛を親側へ隠蔽する為に親側のMB (Message Board)識別情報を子側へコピーする。これによって後のコミット処理成功時のMBへの値書き込みが可能となる。逆に図3に示す様に、子側のチャネル変数に非変数が束縛される場合、このチャネル変数のチャネル属性は消滅させる。

又、子側の変数と親側のチャネル変数が同一化される場合、子側へチャネル属性をinheritする為に、MB識別情報を格納するセルを確保する。

5. NET処理系

本シミュレータでは、種々のネットワーク構成のシミュレーションが可能な様に、共有メモリによってIM間通信を実現している。即ち、各IM毎に共有メモリ上に入力バッファを設けている。この入力バッファ内では、受信パケットはシミュレーション・クロックの順序に従いqueue管理される。

6. おわりに

今回の試作により、データ・コピー量の低減が高速化に不可欠な事が明確となった。今後、構造体データの部分的な共有化を検討する予定である。また、MBを介したmessage passing等の並列プロセス間の同期機構をインプリメントしたが、guard内でのチャネル変数への値束縛の隠蔽、チャネル属性inheritのための動的なMBの確保等の処理が予想以上に重いことが明らかになった。

最後に、日頃御指導いただいた ICOT 村上国男 前第1研究室長 および 内田 俊一 第4研究室長に深謝する。

なお、本研究は第五世代計算機プロジェクトの一環として、ICOTの委託で行なった。

参考文献

1) 尾内 他; "リダクション方式並列推論マシンPIM-Rのアーキテクチャ", The Logic Programming Conf.'85,2.1(1985)

2) 杉江 他; "並列推論マシンPIM-R のハードウェア・シミュレータ -システム構成-", 本大会予稿, IC-2

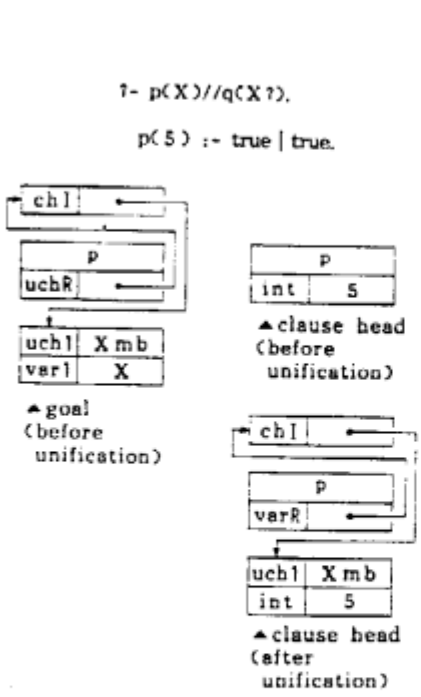


図2. 束縛隠蔽の為の処理例

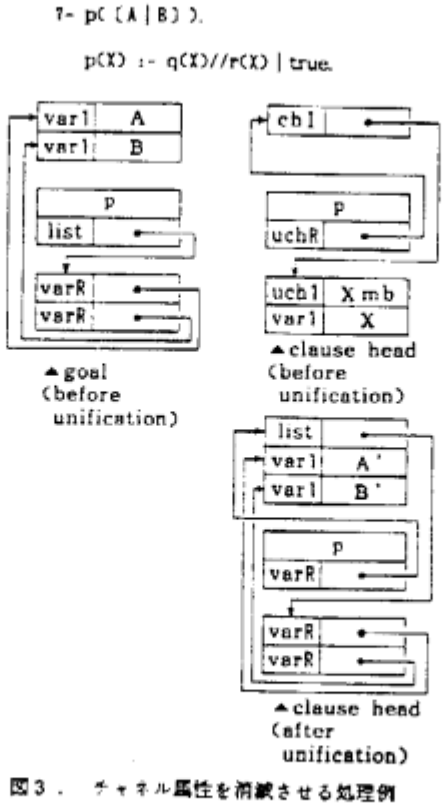


図3. チャネル属性を消滅させる処理例

並列推論マシンＰＩＭ－Ｒの ハードウェア・シミュレータ －ハードウェアの方式－

米山　貢，杉江　衛，坂部俊文，岩崎正明，吉住誠一（日立）
麻生盛敏，清水　肇，尾内理紀夫（ＩＣＯＴ）

１．はじめに

並列推論マシンＰＩＭ－Ｒ［1］［2］のハードウェア・シミュレータを開発した［3］．開発にあたっては、柔軟なシミュレーションに堪える構成を考慮した．ＰＩＭ－ＲのControl ModuleにVAX11/780を、Inference ModuleにMC68000搭載のシングルボード・マイクロコンピュータ（以下SBCと略す）を、Network Moduleに共有メモリを用いることにし、専用ハードウェアによってこれらを結合する構成とした．

本稿では、専用ハードウェアの方式について報告する．

２．ハードウェア

インタフェイスの方式を検討し、次に示す4項目を定めた．

（1）VAX11/780からSBCへの指令に関するインタフェイスの方式は、総てVAX11/780からSBCへの割込み方式とした．

（2）SBCからVAX11/780への処理要求に関するインタフェイスの方式は、総てVAX11/780による状態監視方式とした．

（3）VAX11/780とSBC間のデータ転送に関するインタフェイスの方式は、総てDMA方式とした．また、DMA転送は、DMA制御用LSI（日立HM68450）を使用して行なうことにした．

（4）SBC間のデータ転送に関するインタフェイスの方式は、総てPIO方式とした．

上記のインタフェイス方式に従って専用ハードウェアを開発した．図1に開発した専用ハードウェアの論理ブロック構成の概要を示す．開発したハードウェアは、GPIBインタフェイスブロックとIMボードインタフェイスブロックの2論理ブロック構成とした．GPIBインタフェイスブロックは、GPIB制御論理、68000共有バス制御論理、全SBCからアクセスの行なわれるレジスタやDMA制御用LSI、及びDMA制御用LSIのバス権制御論理で構成した．IMボードインタフェイスは、各SBC毎に必要となるバス権制御論理、割込み制御論理、メモリプロテクション制御論理、及び68000ローカルバス制御論理で構成した．

GPIBインタフェイスブロックを構成するレジスタの仕様を表1に示す．これらのレジスタはVAX11/780-SBC間のインタフェイス動作を行なう場合に使用する．

VAX11/780からSBCへのインタフェイス指令に関しては表2に示す8種類を設けた．これらの指令はIMボードインタフェイスブロックの割込み制御論理に送出され、ここで各SBCに対する割込みが生成される．

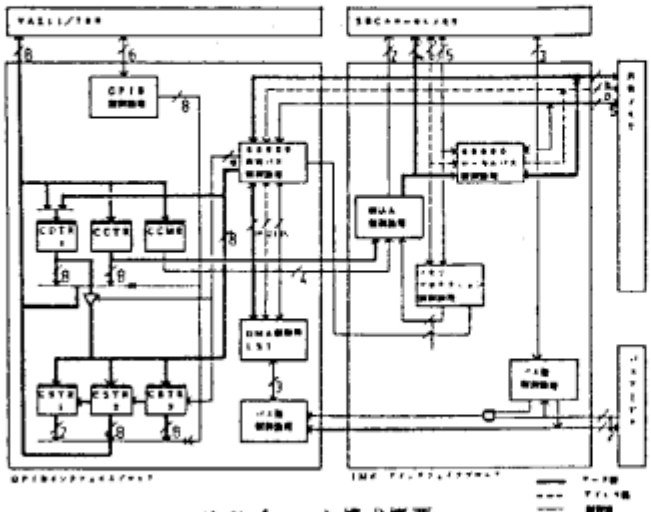


図1　論理ブロック構成概要

表1　レジスタ仕様

レジスタ名	略称	ビット幅（ビット）	仕　様
データレジスタ1	CDTR1	8	データ転送に使用する．
データレジスタ2	CDTR2	8	DMA制御情報転送に使用する．
ステータスレジスタ1	CSTR1	8	SBC68kのステータス確認に使用する．
ステータスレジスタ2	CSTR2	8	データ転送要求を出しているSBC68kの識別に使用する．
ステータスレジスタ3	CSTR3	8	SBC68kの処理完了確認に使用する．
コントロールレジスタ	CCTR	8	VAX11/780が指令を送るSBC68kの指定に使用する．
コマンドレジスタ	CCMR	8	VAX11/780の指令を格納する．

GPIB制御論理は、VAX11/780からレジスタへのデータの書込み、レジスタからVAX11/780へのデータの読出しの制御を行なう。

68000共有バス制御論理は、レジスタやDMA制御用LSIの内部レジスタの値の68000共有バスへの送出、68000共有バス上のデータのレジスタやDMA制御用LSIへの送出、レジスタやDMA制御用LSIによるデータ取込み/送出のための制御信号の生成、などを行なう。

68000ローカルバス制御論理は、SBCのアドレス信号23本、データ信号16本、制御信号4本の68000共有バスへの送出、68000共有バスのデータ信号、制御信号1本のSBCへの送出の制御を行なう。

バス権制御論理は、SBCやDMA制御用LSIとバスアービタとのバス権制御信号3本の送受信制御を行なう。

メモリプロテクション論理は、SBCのプログラムによるアクセス禁止領域へのアクセスの検出、および検出時の割込み制御論理への割込み信号の送出を行なう。

開発した専用ハードウェアは、図1のGPIBインタフェイスブロックおよびIMボードインタフェイスブロックのみであり、これらを、標準基板(IC最大80個搭載可能)4種18枚で実現した。なお、これ以外の部分については、市販品を使用して組み立てた。表3に、

表2 VAX11/780→SBC インタフェイス指令

VAX11/780の指令	内 容
共有メモリ ライト	VAX11/780から共有メモリへのデータ転送
共有メモリ リード	共有メモリからVAX11/780へのデータ転送
ローカルメモリセット	共有メモリからSBC68kのローカルメモリへのデータ転送
共有メモリセット	SBC68kのローカルメモリから共有メモリへのデータ転送
評価用データ要求	SBC68kの評価用データ格納エリアから共有メモリへのデータ転送
ローカルメモリダンプ要求	SBC68kのローカルメモリから共有メモリへのデータ転送
イニシャライズ要求	SBC68kのテーブル類の初期化
強制終了要求	SBC68kのリダクション処理の強制終了

表3 ハードウェア諸元

基板名	IC数	主な搭載論理
GPIBIF1基板	58	CDTR1 CSTR1 CCTR CCMR
GPIBIF2基板	63	CSTR2 CSTR3 DMAコントローラ
CONECT 基板	14	バスドライバ
ISMBDIF基板	56	割り込み制御

設計、製作した専用ハードウェアの諸元を示す。

GPIBインタフェイスブロックは、GPIBIF1基板、GPIBIF2基板の2枚の基板により実現した。

GPIBIFI基板は、CDTR1、CSTR1、CCTR、CCMRの4レジスタおよびその制御論理を搭載している。CDTR1は、CDTR2としても使用する。

GPIBIF2基板は、CSTR1、CSTR2の2レジスタおよびその制御論理、DMA制御論理およびバス権制御論理を搭載している。

IMボードインタフェイスブロックは、CONECT基板、ISMBDIF基板の2枚の基板により実現した。

CONECT基板は、共有メモリ実装のマザーボードとSBC実装のマザーボード間で送受信する信号のドライバのICを搭載している。

ISMBDIF基板は、SBCへの割込制御論理、共有メモリ・アクセス制御論理、SBCのバス権制御論理を搭載している。

3. おわりに

リダクションの概念に基づく並列推論マシンＰＩＭ－Ｒのハードウェア・シミュレータを試作した。

ハードウェアの開発量は、標準基板4種18枚、約680ICである。

最後に、本研究の推進にあたって日頃御指導いただいた ICOT 村上 国男 前第1研究室長 および 内田 俊一 第4研究室長に深謝する。

なお、本研究は第五世代計算機プロジェクトの一環として、ICOTの委託で行なった。

参考文献

1) 尾内 他 ;"並列推論マシンＰＩＭ－Ｒのアーキテクチャ",情報処理学会第30回全国大会6C-6(1985)

2) 尾内 他 ; "リダクション方式並列推論マシンＰＩＭ－Ｒのアーキテクチャ",The Logic Programming Conf. '85,2.1(1985)

3) 杉江 他 ;"並列推論マシンＰＩＭ－Ｒのハードウェア・シミュレータ －システム構成－",情報処理学会第31回全国大会1C-2(1985)

並列推論マシンＰＩＭ－Ｒの ハードウェア・シミュレータ －評価方式－

杉江　衛，米山　貢，坂部俊文，岩崎正明，吉住誠一（日立）
麻生盛敏，清水　肇，尾内理紀夫（ＩＣＯＴ）

１．はじめに

第五世代コンピュータ・プロジェクトの一環として、ＰＩＭ－Ｒ[1]のハードウェア・シミュレータを試作した[2]－[5]．本ハードウェア・シミュレータ開発はＰＩＭ－Ｒのアーキテクチャ検証を目的とし、①ＰＩＭ－Ｒ詳細構造のシミュレーション、②シミュレーション速度の向上、③並列環境の実現　等を図っている．本稿では、評価方式および評価結果について報告する．

２．評価方式

Prolog および Concurrent Prolog のテスト・プログラムを走行させ、アーキテクチャの評価を行なった．

並列推論マシンでは、プロセスの分配が重要になる．シミュレーションにおいては、これに関連したネットワークとプロセス分配方式に焦点を合わせ、ネットワーク・スループットの台数効果への影響、プロセス分配方式のプロセッサ・エレメント稼働率分布への影響を調べた．また、プロセッサ・エレメント間の交信量の低減のための基礎データとして、ネットワークを通過するパケットの統計的データも収集した．

本ハードウェア・シミュレータは本格的なベンチマークによるＰＩＭ－Ｒのアーキテクチャの評価を目指しており、シミュレーションの高速化は不可欠である．そこで、本ハードウェア・シミュレータではイベント駆動方式を採用し、シミュレーション実行時におけるアイドル時間の省略を図った．

タイマに関して、システム全体を統一的に計時するTOD (Time of Day Clock)は有しておらず、各IM内にソフトウェア・タイマを設けている．タイマは、ある機能単位の処理を実行する毎に、予めパラメータとして指定された値を加えることによって更新される．他のIMに対してパケットを送信する際には、タイマの値に、パラメータ指定されたネットワーク遅延時間を加えた値を、到着時刻として送出パケットに付加しておく．送り先のIMでは、パケットの取り出しの際に、この到着時刻と自己のタイマの値を比較することによって、計時を制御する．

３．シミュレーション条件

（１）転送パケットの衝突のないネットワークを仮定する．

（２）各IMが有する入出力バッファは十分に大きいとし、ハードウェア・シミュレータの物理的バッファの容量オーバーによる待ち時間は計数しない．

（３）Prologではルールに対してユニフィケーションが成功したときに、Concurrent PrologではAND-fork時に、新しい子プロセスを各IMに分散させるが、ネットワーク方式によって、次の分配方式を採用した．

鎖状ネットワーク：自IM→右IM→左IM→自IM→……

メッシュ状ネットワーク [7]：自IM→東IM→南IM→西IM→北IM→自IM→……

４．評価結果

図1，2，3に、6-Queensの場合のIM台数効果、IM稼働率を示す．図1からは、メッシュ方式ネットワークによって、IM台数≦8では、処理性能が、台数に応じて改善されていることがわかる．この結果は、ＰＩＭ－ＲがPrologプログラムに内在する並列性を引き出していることを示していると考えられる．また、図1は、鎖方式ネットワークでは、5～6台で処理性能向上に飽和が見られることも示している．その原因は、図2に見られるとおり、各IMの稼働率の低下にある．一方、メッシュ方式では、図3が示すように、全IMの平均稼働率も高く、稼働率分布も小さい．図2からは、IM#0，#1，#2，#7の4個のIMで稼働率が高いことがわかる．このことから、鎖方式ネットワークでは、～4個のIM鎖について、プロセス分配を均質化できると考えることができる．2×2および2×4のメッシュ構造で、処理性能向上が得られたのは、このためと考

えられる。

表1に Concurrent Prolog版 Quick-sort プログラムをベンチマークとした場合のネットワーク通過パケットに関するデータを示す。表1からは、IM台数が増すにつれて、ネットワークを通過するパケット数が増していくことがわかる。これは、IM台数の増加によって、message board関連のパケットが増加することによる。実行したQuick-Sortプログラムの総リダクション回数は1387であるので、IMが8台の場合、約0.6リダクションに1回の割合でパケットの転送が生じることになり、しかも、転送パケットの約90%がmessage board関連のパケットである。このことから、Concurrent Prolog処理系では、message boardアクセスのためのネットワーク・スループットの確保が重要になることがわかる。

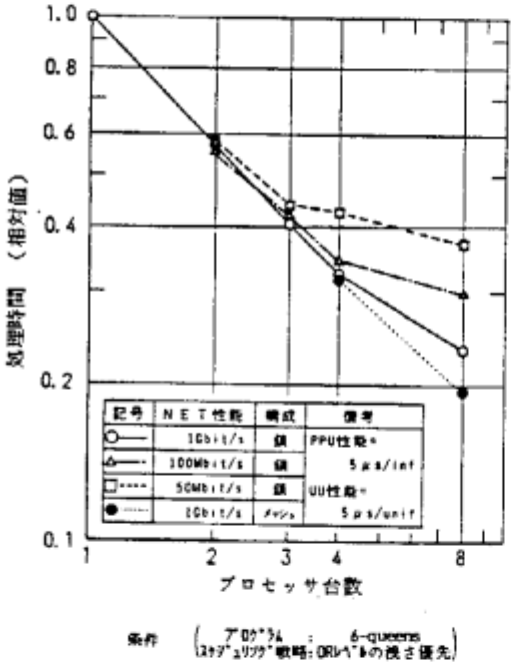
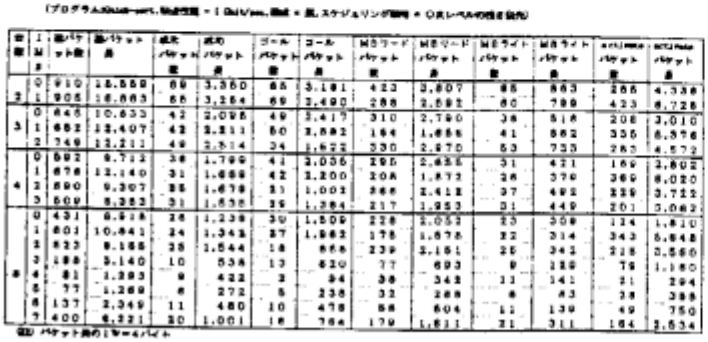


図1 台数効果

表1 送出パケット・データ



5. おわりに

本稿では、6-Queens, Quick-sortプログラムによるＰＩＭ－Ｒの評価について報告した。今後、ベンチマークを増して、ＰＩＭ－Ｒの評価を続ける予定である。最後に、日頃御指導いただいた ICOT 村上 国男 前第1研究室長 および 内田 俊一 第4研究室長に深謝する。

なお、本研究は第五世代計算機プロジェクトの一環として、ICOTの委託で行なった。

参考文献

1)尾内 他；“リダクション方式並列推論マシンＰＩＭ－Ｒのアーキテクチャ”, The Logic Programming Conf. '85,2.1,(1985)

2)-5) 本大会予稿1C-1～1C-4 ”並列推論マシンＰＩＭ－Ｒのハードウェア・シミュレータ -開発思想- ～ -ハードウェアの方式- ”

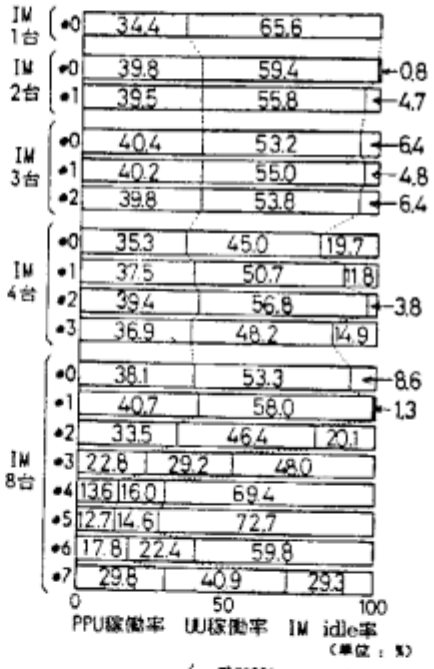


図2 IM 稼働率比較

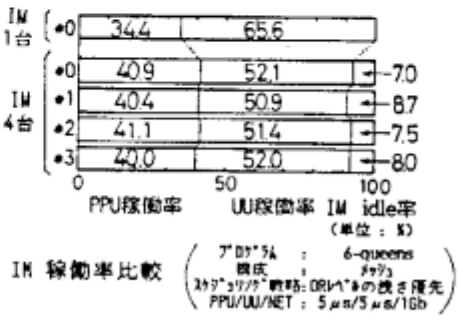


図3 IM 稼働率比較

ＡＮＤ並列型Ｐｒｏｌｏｇにおける ストリーム処理の最適化について

伊藤　慎義　　来住　晶介　　久野　英治　　六沢　一昭
（ＩＣＯＴ）　　（　沖　　電　　気　）

１．　はじめに

GHC(Guarded Horn Clauses)[1]に代表されるＡＮＤ並列型Prologは，ＡＮＤ並列に実行されるプロセス間でガード機構を用いたインタラクティブな通信手段を提供しており，ストリーム並列処理を実現することができる．しかし，このような言語ではプロセス間の通信は論理変数に対する統一化によって行われるために，ＯＲ並列型Prologにおけるストリーム通信処理に比して，処理オーバヘッドが大きい[2]．本稿では，このような処理の効率的な実現手法について述べる．基本的な考え方は，ストリームの詳細な構造やインプリメンテーションをプログラム中から隠蔽し，ストリーム処理をマシンのプリミティブとして実現することにより，論理型言語の枠組を壊すことなくストリームの効率的な実現手段を提供することにある．

２．　ストリーム並列処理

ＡＮＤ並列型Prologにおいては，ゴールから起動されるプロセス群はゴール間で共有される変数を介してメッセージ通信を行う．このとき，起動されたプロセスをコルーチンとして実現でき，これらのプロセス間で通信されるメッセージ系列はストリームとして実現される．

ストリームは任意の要素系列からなる構造データとして表現され，要素系列を生成する生産者プロセスと，それらを消費する消費者プロセスの間の非同期通信手段を提供する．即ち，生産者プロセスは新しい要素を生成する度にストリーム中に格納し，消費者プロセスは格納された要素をストリームから次々に取出す．

図１は以下のゴールが与えられたときに，２つのサブゴール間で通信されるストリームの様子を示す．

```
?- p(X),q(X).
```

同図において，サブゴールp(X)から起動されるプロセスはメッセージ列xi(i=1,2,...)を生成し，q(X)から起動されるプロセスはメッセージ列を消費する．メッセージの渡される方向はプログラム中で指定する．ＧＨＣでは，メッセージ方向をガード機構を用いて制御する．即ち，プログラムを構成するすべての節をガード節として表現しており，節のガードにおいて，親のサブゴールから渡された変数に非変数項（又は親から渡された他の変数）を結合する統一化は禁止される．このような統一化を行おうとすると，他プロセスによって変数に値が結合されるまで（変数を表すセルに値が書込まれるまで）待ち状態になる．

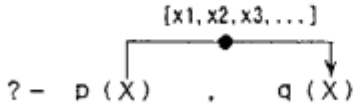


図１．　ストリームによるプロセス間通信

図１に示した処理は以下のようなガード節で表現される．

```
p(Y) :- ... | ..., Y=[E|Z],p(Z).
q(W) :- W=[U|V],... | ..., q(V).
```

この例においては，ストリームはリストとして表現されており，サブゴールq(X)から呼出される節ｑのガード（コミットオペレータ "|" の左辺）において，与えられたゴール中の変数Ｘにリストセル[U|V]を結合する統一化が試みられる．これによってＸの値（この場合はリスト）が読出され，先頭要素Ｕと残りの要素列Ｖに分解される．即ち，節ｑはストリームの消費者として動作する．一方，サブゴールp(X)から呼出される節ｐは変数ＸとリストセルE|Z]との統一化をその本体（コミットオペレータの右辺）で行う．これによって変数Ｘにリストが結合され，節ｐはストリームの生産者として動作する．

３．　ストリーム処理の効率化

上述の例ではストリームをリストとして表現しており，図２に示すような構造が生成される．即ち，ストリーム要素を１つ生成する度に変数セル及びリストセルの割当てを行わなければならない．このようなセルは消費者プロセスが要素の取出しを完了するとただちにガーベジとなる．従って，メモリの割当て及び解放が頻繁に行われその管理オーバヘッドが大きくなる．

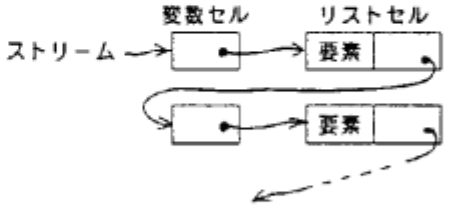


図２．ストリーム（リスト表現）の構造

以下，このようなオーバヘッドを減少するためのストリームの効率的な実現法をいくつかの例をもとに説明する．

（１）　単純な生産者／消費者問題

上述のプログラムは以下のように書き直すことができる．

```
?- create-stream(T,H),p(T),q(H).
p(T) :- ... | ..., stream(T,E,T1),p(T1).
q(H) :- stream(H,E,H1),... | ...,q(H1).
```

ここで注目すべきことは，ストリームの詳細な構造や実現法はプログラムから隠蔽されており，ストリーム処理は以下に示すようなcreate-stream 及びstream述語（ストリーム操作述語）を介して行われることである．

```
create-stream(X,Y) :- true | X=Y.
stream(X,V,Y) :- true | X=[V|Y].
```

従って，このストリーム操作述語をシステムの組込み述語としてインプリメントすることにより，効率良いストリーム処理が実現できる．以下，このような実現法について述べる．

ゴール中のcreate-stream 組込み述語は適当な大きさのバッファ領域を割当て，その先頭ポインタを変数H及びTのインスタンスとして返す（図３（ａ））．一方，ゴールから起動された節で実行されるstream組込み述語は以下の操作を行う．節pの本体で実行されるstream述語は変数Tに結合されたポインタの指す語に要素Eを書込み，インクリメントしたポインタを変数T1に結合する．節qのガードで実行されるstream述語は変数Hに結合されたポインタの指す語の内容を読出し変数Eに結合して，インクリメントしたポインタを変数H1に結合する（同図（ｂ））．

このようなインプリメンテーションによって，プログラム上は論理型言語の枠組を壊すことなく，ストリーム処理を効率良く行える．即ち，ストリーム要素の書込みや読出しの度にメモリセルの割当てや解放を行う必要はなく，バッファが一杯になったときのみ，メモリ管理を行えばよい．

（２） 非決定的マージ

このようなストリーム処理の実現法によって，複数のプロセスから生成されるストリームを非決定的に一本のストリームにマージする処理も効率良く制御できる．今，以下のゴールを考える．

```
?- merge(X,Y,Z),p1(X),p2(Y),q(Z).
```

この例で，述語p1及びp2がそれぞれストリームＸ及びＹを生成し，述語q はそれらをマージした結果Ｚを消費するものとする．純論理型インタプリタではこのmerge 述語を以下のように定義する[1]．

```
merge([],Y,Z) :- true | Z=Y.
merge(X,[],Z) :- true | Z=X.
merge([H|X],Y,Z) :- true | Z=[H|W],merge(Y,X,W).
merge(X,[H|Y],Z) :- true | Z=[H|W],merge(Y,X,W).
```

これに対して，ストリーム操作組込み述語を用いてこのゴールを以下のように書き直すことができる．

```
?-create-shared-stream(X,Y,Z),p1(X),p2(Y),q(Z).
```

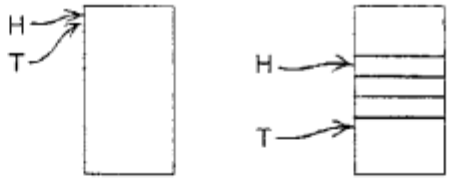
述語create-shared-streamは，図４に示すように２つの生産者間で共有されるストリームを生成する．同図における STP(Stream Tail Pointer)はストリームの現在の最後尾を指しており，各生産者プロセスはストリーム要素が生成される度にSTP の指す語に要素を書込みSTP の内容をインクリメントする（このときSTP の参照から更新の間の処理は不可分のオペレーションである）．これによって，STP はバッファ中の次の要素の書込まれる語を指す．

（３） 有限長バッファ通信

有限長バッファ通信は，生産者プロセスが必要以上にストリーム要素を生成するのを抑止するのに使用される．これを実現するのは，create-stream 述語でバッファサイズＮを指定し，長さＮの循環バッファを生成するようにすればよい[3]．このとき，生産者側がＮ個以上の要素をストリームに書込むのを抑止するために，生産者側と消費者側で共有されるセマフォを生成し，その初期地をＮに設定する．生産者プロセスはこの共有セマフォに対しＰオペテーションを実行するし，消費者プロセスはＶオペレーションを実行する．

４． おわりに

ＡＮＤ並列型Prologにおける効率良いストリーム並列処理の実現法について述べた．今後，試作を進めているデータフロー方式並列推論マシン[2] 上でその有効性等の評価を行っていく予定である．最後に，御討論いただいたICOT第一研究室及び第四研究室の各位に深謝する．



（ａ）初期状態　　（ｂ）要素格納後

図３． ストリームの構造

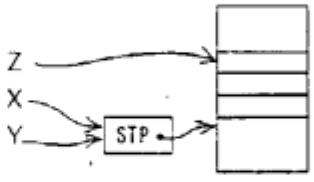


図４． 共有ストリームの構造

<参考文献>

[1] Ueda,K., "Guarded Horn Clauses," ICOT TR-103, June, 1985.

[2] Ito,N., et al., "The Dataflow-based Parallel Inference Machine To support Two Basic Language in KL1," IFIP TC-10 Working Conference on 5th Generation Computer Architecture, Manchester, July, 1985.

[3] 尾内，麻生『並列推論マシンにおけるGuard と入力Annotationの制御機構』情報処理第27回全国大会予稿集．

拡張 Concurrent Prologの試作*

田中二郎** 横森 貴 岸下 誠
（富士通 国際研） （富士通SSL）

1. Extended Concurrent Prolog

筆者らは、1984年の核言語第1版概念仕様書[Furukawa 84] に基づき、 AND並列及びOR並列機能、集合抽象化機能、メタ推論機能などを持つExtended Concurrent Prolog (ECP) Interpreterの試作を試みた[Fujitsu 85]。Shapiro の Concurrent Prolog (CP) Interpreter [Shapiro 83] では、AND 関係のゴールのみをscheduling Queueに入れていたが、我々はそれを拡張し、プログラム実行中に現れるすべてのAND 関係及びOR関係が、一つのscheduling queueのなかに入るようにした。我々は、この方式を'AND-OR Queuing'と命名した。この方式により、ECP の諸機能を統一的な方法でimplement することが可能となった。本稿では、特にscheduling queueに関連した部分の基本的インプリメント技法の解説を行なう。

2. AND並列及びOR並列機能

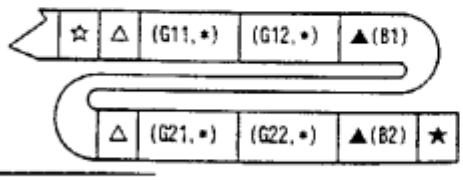
AND並列機能とは、論理的にAND で結ばれたゴールを並列に評価する機能である。AND 並列は、scheduling queue の先頭のゴールをとり出し、リダクションを行ない、その結果をqueue の末尾に登録することにより実現される。この AND並列機能については、すでにShapiro のInterpreter で実現されていた。

一方、OR並列機能とは、ゴールとユニファイ可能なヘッドを持つ節が複数個あるとき、これらの節についてガード部の実行を並列に行なう機能である。我々は考察の末、OR並列を2種類のマーカーで囲み、scheduling queueの中に格納することにした。

例えば、queue の先頭から取り出されたゴールがPであり、ゴールPに対する候補節が

$$P_1 \ :- \ G_{11}, G_{12} \ | \ B_1$$
$$P_2 \ :- \ G_{21}, G_{22} \ | \ B_2$$

であるとき、scheduling queueの末尾には以下のように格納される。



ここでOR-Claueses 全体（OR関係）は☆と★で囲まれている。定義節のガード部分（AND 関係）はそれぞれ△と▲に囲まれて表されている。• は、一つ一つのプロセスの属している世界がグローバル・データベースの世界であることを示している。また、ここでは省略しているが、各マーカーには引数として、例えばある節がコミットされたか等の制御情報が入る。

このscheduling queueは、queue からマーカーが取り出されたときは以下のような処理をおこなう。即ち、☆が取り出されたときには、いずれかのガードが既にコミットされているかを調べ、そうであればコミットされた節のボディ部をOR-Clauses全体に置換える。△が取り出されたときには、scheduling queueの先頭が▲であるか、即ちガードが空であるかを調べ、空であればそのガードをコミットする。

3. 集合抽象化機能

集合抽象化機能とは、並列環境下におけるPure Prolog的全解探索機能の実現のために導入された機能である。こうした集合抽象化機能を実現する述語として、以下のような形式のeager-enumerate とlazy-enumerateの2つが提案されている[Fujitsu 84]。

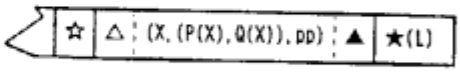
```
eager-enumerate({X | Goals}, L)
lazy-enumerate ({X | Goals}, L)
```

ここでGoals は、Pure Prolog の世界で定義されている述語のゴール列である。この2つのenumerate は、Pure Prolog の世界でGoals を解き、それを満たすようなX の集合をリストの形でL に入れる。

集合抽象化機能の例として、scheduling queueから次のようなゴールが呼出されたケースを考える。

```
eager-enumerate({X | P(X), Q(X)}, L)
```

このとき、scheduling queueの末尾には、つぎのように格納されると考えられる。



ここで、ふたたび2種類のマーカーが現れていることに留意されたい。☆と★は全体を囲んでおり、解を集めそれを外部に公開する役目をする。△▲の挟む部分は、解の1つを計算する役目をする。ここでも簡単化してあるが、実際には、各マーカーには、コントロール情報や、解を集めて

* 本研究は、第5世代コンピュータ・プロジェクトの一環として行われたものである。

**1985年 6月より、ICOT、第1研究室。

くるための補助情報が、引数として入る。

△▲がqueue から取り出された時の処理を詳しく説明すると以下のようになる。

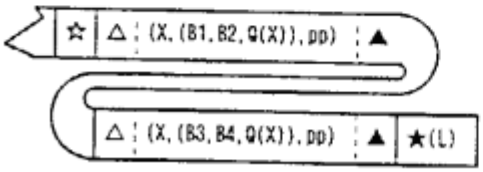
①△と▲の間が空であれば、それは成功を意味するので、そのときのX をL に登録する。

②△と▲の間に通常のゴールがあれば、△の次のゴールにつき、その定義節を捜し出し、ヘッド・ユニフィケィションの可能なものが複数個あるときには核分裂を起こす。

核分裂については、解りにくいので例で説明する。前の例において、P がPure Prolog の世界で以下のように定義されていると仮定する。

```
pp((P(X)  ←  B1, B2)).
pp((P(X)  ←  B3, B4)).
```

P(X)の定義節は２つあるので、このとき△と▲の間は核分裂を起こし、最終的には以下のようなものがscheduling queueの末尾に格納される。



こうして核分裂を次々に起こしながら、最終的に全解が求まることになる。lazy-enumerateについても、★における解の集め方コントロール情報の渡し方が異なるだけで、全く同じ原理で処理できることになる。

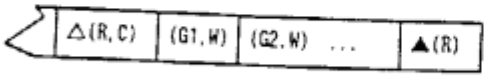
4. メタ推論機能

メタ推論機能とは、与えられたゴールを、ある世界(World) の中で定義されている知識を用いて解く機能である[Furukawa 84] 。このメタ推論機能を実現するプリミティブとして、以下のような形式の、述語simulateを用意する。

simulate(World, Goals, Result, Control)

ここでWorld は与えられる世界名、Goals は解かれるべきゴール列のリスト、Resultはゴールを与えられた世界の知識を使って解いた時の結果、Control はメタ推論機能を一時止めたり再開したりするコントロール情報を入れるストリームである。

scheduling queueからsimulate述語が呼出された場合、今Goals がG1, ... ,Gn であるとすれば、scheduling queueの末尾には、つぎのような形で格納されると考えられる。



上の図で、マーカー△と▲がまた使われていることに留意されたい。引数のR はResult、C はControl 、W はWorld 名をそれぞれ表わしている。△がqueue から取り出された時の処理を簡単にまとめると以下のようになる。

①△の引数Resultに、既にfailがセットされているときは、△から▲までをすべて取除く。

②queue の先頭が▲である、即ち△と▲の間が空であれば、Resultにsuccess がセットされる。

③△の引数Control が[... stop | 変数] であるときは、△以降のゴールにリダクションを行なうことなく、対応する▲までのゴール列をqueue の末尾につける。

④△の引数Control が[... cont | 変数] もしくは変数であるときは、単にマーカー△をqueue の末尾につけ、次にすすむ。

5. まとめ

以上、簡単にICP における拡張論機能の実現のための骨組みについて説明した。より詳細については、本稿のfull paperにあたる[Tanaka 85] を参考にされたい。我々はAND-OR-Queuingを提案したが、従来、全く違った機能と考えられていたOR並列機能、集合抽象化機能、メタ推論機能などが１つの骨組みの中で説明できることは驚くべきことである。

さて、AND-OR-Queuing方式の適用範囲であるが、むろん適用範囲はConcurrent Prolog に限らない。新しいKL1 の中核と想定されているGHC [Ueda 85] においてもその方式は有効であると言えよう。

6. 謝辞

本研究は、第５世代コンピュータ・プロジェクトの一環として行なわれたものである。本研究にあたっていろいろ協力を戴いた、ICOT第１研究室の竹内氏、上田氏、古川室長、国際研の北川会長、横本所長、富士通SSL の吉井氏に感謝する。

[参考文献]

[Fujitsu 84]富士通: 58年度電子計算機基礎技術開発再委託成果報告書、5G核言語第１版の検証用ソフトウェア・詳細仕様書 Part II、1984.

[Fujitsu 85]富士通: 59年度電子計算機基礎技術開発再委託成果報告書、5G核言語第１版の検証用ソフトウェア・詳細仕様書（第２版）及び試験評価データ Part I 、1985.

[Furukawa 84] 古川、渕一、竹内、上田: 核言語第１版概念仕様書、ICOT、1984年 3月.

[Shapiro 83]E.Shapiro: A Subset of Concurrent Prolog and Its Interpreter、ICOT Technical Report 、TR-003、1983.

[Tanaka 85] 田中、横森、岸下: AND-OR-Queuing in Extended Concurrent Prolog、The Logic Programming Conference '85、新世代コンピュータ技術開発機構、1985年 7月.

[Ueda 85] K.Ueda: Guarded Horn Clauses、ICOT Technical Report 、TR-103、1985.

PSI のガーベッジコレクタ

3C-3

西川 宏
（新世代コンピュータ技術開発機構）

池田 守宏
（三菱電機）

1. はじめに

逐次型推論マシンPSI は，第五世代コンピュータプロジェクトの一環として，論理型言語を効率よく実行するために開発された，高級言語マシンである[1]． 本報告では，PSI 上のガーベッジコレクションの概要を述べる．

2. アドレス変換とメモリ管理

PSI の32ビットの論理アドレスは，上位 8ビットのエリア番号と，下位24ビットのエリア内のアドレスにわけられる．エリア内アドレスは1K語単位のページに分割され，論理アドレスから物理アドレスへの変換には，エリア毎に用意されたページ変換テーブルが用いられる．

各々のエリアのページ変換テーブルの大きさは，ファームウェアのページ割当ルーチンにより管理される；プログラムの実行にともない新しい論理ページへのセルの割りつけが必要となると，ページ変換テーブル中の対応するエントリが空であれば，ページ割当ルーチンにより，ファームウェアで管理されている自由物理ページプールから物理ページが取りだされ，対応するエントリに登録される．

ただしPSI では仮想記憶を採用していないので，自由物理ページプールが空の状態でさらに物理ページを取りだそうとした場合，PSI のマイクロインタプリタは割り出し（トラップ）のメカニズムにより，物理ページ不足の情報を対応するトラップハンドラに通知し，

(a) 各プロセスの状態を調べ，もしスタックとして使用されたエリアが縮んでいるときには，不必要となった物理ページを自由物理ページテーブルに返却する；

(b) もし(a) の操作で充分な物理ページが回収できないときには，マイクロコードで記述されたガーベッジコレクタをよびだす；という作業がなされる．

このように，PSI 内では，まずページ単位のG.C.がおこなわれ，それでも充分なページが回収できないとき，本格的なG.C.をおこなうことにしている．

3. ガーベッジコレクタ

PSI にはside-effect により状態が変化するオブジェクトの格納領域としてヒープ領域がサポートされている．この領域はバックトラックによって縮まないので，生成されたオブジェクトはG.C.でのみ解放される．PSI 内のヒープ上のオブジェクトは複数のプロセスから共有されているのが一般的であり，もしこれが移動されると，それを参照している全てのプロセスにもその情報をつたえる必要があり，ヒープ領域のG.C.をおこなうためには全プロセスのスタック中の状態を調べなければならない．

さらにKL0 の実行では，各プロセスのグローバルスタックのみならず，コントロール，ローカルスタック上にもゴミが生成される（例えば遠隔カットにより）ので，この領域もG.C.をする必要がある．

したがってG.C.をおこなうときは，すべてのエリアを同時におこなったほうがよい；PSI 内に存在するプロセスについて一括G.C.をするのが得策である．

各種スタック中のゴミでないセルは，コンパクションをおこなってゼロアドレス側に移動させるが，セル間のアドレスの大小関係は保ったままおこなう必要がある．この性質はBaker[2]に代表されるコピーG.C.との相性がよくない．またこの方法ではプログラムが実際に使用できるメモリ空間は半分になってしまう．PSI のように，仮想記憶システムでないマシンではこれは欠点となる．

このような理由によりPSI のガーベッジコレクタはマーク& スイープ型として実現されている．図 1に示されるように，マーキングではプログラムの解釈／実行に必要となとなる 4つのスタックとヒープ中の生きているセルをプロセス毎にマークすることがおこなわれ，コンパクションでは，各エリア毎にゼロアドレス側にマーク付けられたセルを移動することがおこなわれる．

またすべてのメモリセルには２ビットのフラグがG.C.のために付与されている．

3.1 マーキングフェーズ

マーキングは生きているセルをマークすることでおこなわれる．ローカル／グローバルスタック中に出現した構造

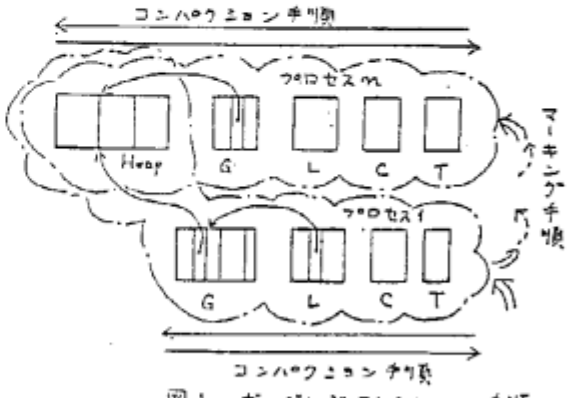


図1． ガーベッジコレクションの手順

データは，それを格納している変数セルがルートとなり，さらにマーキングがおこなわれる．構造データのマーク付けには，それがネストする毎に，その戻りセルアドレスと未処理セル数をスタックに積む方法をとっている．このための領域はファームウェア用に用意された主記憶上の作業領域をつかっている．ちなみにヒープ領域に格納されたオブジェクトで必要なものは，必ずあるプロセスのローカル／グローバルスタック中のセルから参照されるので，ゴミでないヒープ中のオブジェクトはすべてマークできる．

PSI ではクローズの実行途中の情報はコントロールスタック中に，変数セルはローカル／グローバルスタックのいずれかに，それぞれフレーム単位で生成されるので，マーキングはフレーム単位におこなわれる．

生きているコントロールフレームが検出できれば，対応するローカルおよびグローバルフレームはコントロールフレーム中の情報からわかる．生きているコントロールフレームをマークするためには，コントロールフレーム中に格納されている親(parent)チェーンと後戻り(backtrack) チェーンがつかわれる．

コントロールフレームのサイズは固定であり，対応するローカルフレームのサイズは，連続したコントロールフレームに格納されたローカルスタックベースの差として算出可能である．一方，グローバルフレームのサイズは対応するコントロールフレームがテイルリカージョンにより消滅している場合もあり，コントロールフレーム中の情報からは決定できないので，フレームの先頭にそのサイズを格納している．

マーキングを始めるにあたり，個々のエリアのセル数はテーブルに格納され，セルがマークされる毎に，対応するテーブルエントリのカウントをひとつ減じる．これにより，マーキング終了時には各エリアのゴミ数が得られる．

3.2 コンパクションフェーズ

PSI ではゴミセルの回収にスライディング方式をとっている．この結果，各エリア内のセルがゼロアドレス側に移動し，そのエリアで以前に使用されていた論理ページに対応する物理ページを自由物理ページプールへ返却することが可能となる．

DEC-10 PROLOG[3]では各グローバルフレームごとにコンパクション後の移動量をしめす語を持っており，セルの内容がグローバルスタックへのポインタである場合，コンパクション後のセルアドレスを計算するために，常にリニアサーチをして移動量を格納したセルを捜しだしている．

PSI ではこれを避けるために，同一セルへの参照をしているすべてのセルを，被参照セルをルートとして，一重書きでそれを参照していたセルを結ぶチェーン構造に変換し，被参照セルの移動通知をチェーンをたどり各参照セルにしらせ，その後元の構造にもどすHorris[4] の方式を採用している(図 2参照)．

ただしPSI では複数のエリアが存在し，かつエリア間のポインタが存在するので，エリア番号を含めた32ビットの論理アドレスに基づいてコンパクションをおこなっている．つまり各エリアのスイープ手順は，エリア#255からエリア#1へとおこなわれ，つぎに，その逆順でおこなわれる．

最初のスイープでは自分より上方(スイープ方向と同一) を指しているアップワードポインタであるセルをコンパクション後のアドレスにつけ換る作業がなされる．この補正に必要な未スイープ領域のゴミ数は，そのエリアのゴミ数から，スイープした領域のゴミ数と引くことでえられる．図 2のセルa:の処理ではその数をもとに，参照セルb:,c:のポインタの値をコンパクション後の値につけかえている．

つぎのスイープではエリアのゼロアドレス側からエリアの先頭へ向かって処理がおこなわれる．このときダウンワードポインタ(セルの内容が自身より下方をさす) の処理が，アップワードスイープと同様の方法でなされ，エリア内セルのコンパクション処理もおこなわれる．

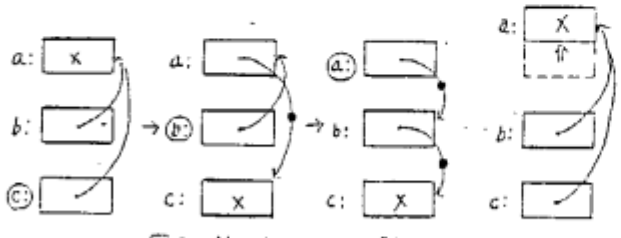


図 2. Morris のアルゴリズム

4. おわりに

ガーベッジコレクタの容量は約 1.5K ステップであり，現PSI 上では，4Mセルの主記憶のガーベッジコレクションには約40秒要している．今後は方式上のことも踏え，いろいろ評価をおこなっていくことを考えている．最後にG.C.のコーディングとデバッグに協力いただいた久津間氏及び関係者各位に感謝します．

参考文献

[1]Yokota,H et al:A Microprogrammed Interpreter for The Personal Sequential Inference Machine,Proc. of FGCS'84 ,1984.

[2]Baker,H.G:List processing in real time on a serial computer,Commun.A.C.M 21,4,1978.

[3]Warren,D.H.D:Implementing Prolog-Compling Predicate Logic Program Vol.1-2,D.A.I Research Report,No.39-40,Dept. of A.I.,Univ. of Edinburgh,1977.

[4]Morris,F.L :A time- and space-efficient garbage compaction algorithm ,Commun.A.C.M 21,8,1978

株分け処理方式の評価

2C-5

久門耕一　板敷晃弘　佐藤健　中村直人　増沢秀穂　相馬行雄
（富士通株式会社）

1．はじめに

我々は、先に"株分け方式"と呼ぶ新しい並列推論方式を発表し[1]、ワークステーションによるシミュレーション結果を発表した[2]。今回、その経験を基に要素プロセッサ16台によるハードウェアシミュレータを作成し、これによるデータを得たので報告する。

2．ハードウェアシミュレータ（HWS）の目的

前回発表を行った、ワークステーション上での動作実験用シミュレータ（WSS）によれば、株分け方式の通信回数は少ない事が分かった。又、逐次型処理系を並列型処理系に変更した事によるオーバヘッドは、6%以下であった。

しかし、WSSでの処理系実現上の都合から、通信のオーバヘッドが大きく、全体の処理量が小さい時や、PE台数を多くして通信一回あたりの仕事量が小さくなると、相対的に通信のオーバヘッドが大きくなり処理系の性能が低下する事が問題であった。特に、あるPEが仕事を終了し、他のPEからの仕事の転送を待っている時間は、システムとして無駄な時間でありこの時間を短縮する為にハードウェアによる援助がシステムの高速化のためには重要であると考えた。

WSSでは、PEが4台程度で台数効果に頭打ちの傾向がみられたが、多数のPEを接続し実行速度を向上させる並列推論処理を実現するためには通信を高速化する事で、目的とする性能が挙げられることを確認する必要がある。

そこで、今回作成したHWSでは、通信の高速化を念頭に置いて通信専用のネットワークを2種用意した。特にPEが仕事をせずに空いている時間を小さくする為に、各プロセッサに対し排他制御高速化用のネットワークを設計した。

3．システム構成

今回試作したハードウェアシミュレータ（HWS）の構成を図1に示す。処理を行う要素プロセッサ(PE)は MC68010 を使ったマイクロプロセッサを用いた。16台中の1台は、ユーザとのインタフェースを取るための管理用として用い推論動作は行わない。各PEには、株分け方式に基づく並列処理系が実装されている。

PE間を結合するためのネットワークとして

①PEに対する排他制御を高速化する為のリング状のネットワーク

②PE間で高速に仕事を転送する為の多段ネットワーク

を作成した。

ネットワークの諸元を表1に示す。

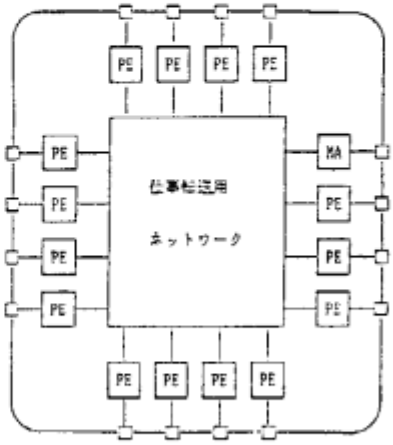


図1　システム構成

・多段ネットワーク

4×4　スイッチ　4行2列の多段ネットワーク

8ビット並列DMAデータ転送

400KB/S(ハードウェア) 50KB/S (ソフトウェア込み)

・リングネットワーク

制御情報4ビット　付加情報8ビット　並列転送

クロック2MHz　・周回時間最大8μsec

表1．　ネットワークの仕様

リングネットワークは、並列推論を行っている各PEの状態（推論中、暇）を示すパケットが常時回っており、推論実行中のPEが実行中の仕事が分割（株分け）可である時、リングネットワークアダプタ(RNA) を通じて暇なPEのパケットを拾い排他的に仕事を暇なPEに転送する。

多段ネットワークは、PEとDMA により接続されており、仕事転送等の比較的大容量のデータ転送を行う。

4．試作システムの評価

試作システムの評価には、N-QUEEN 問題を用い、PE内で実行時に行われている処理の解析、HWSで2種類のネットワークを作成した事に対する効果とPEの台数とNの値を変えた場合の実行時間、並びに各PEによる通信量の総数を調べた。

(1) 台数効果

PE台数に対する台数効果を図2に示す。

WSSとHWSを比較すると、8-QUEEN を4台のPEで実行した場合、WSSでは単体性能の 2.8倍（理想値の70%）であったが、本HWSでは、3.9 倍（同 97 %）と向上

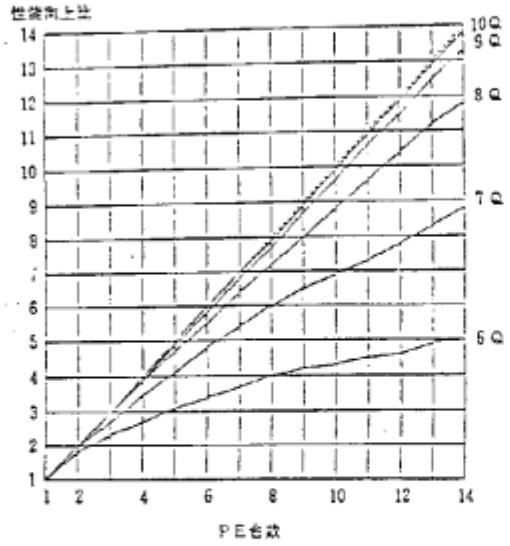


図2　PE台数に対する台数効果

している。又、探索木の大きい10-QUEENでは、PEが14台の時にも、13.8倍（同98%）と理想値に近い値が得られた。

(2) 通信回数

PE台数とN-QUEEN　のNの値による通信回数の変化を図3に示す。

一般にN-QUENEN では、処理量はNに対して指数的に増加するが、図3に見られるように、枝分け方式では通信量はPE台数とNの値双方に対して一次的に増加している。この事は、枝分け方式が探索木の大きな問題に対して、一回の通信により渡される仕事が大きくなるという事を示している。又、推論を行っている最中の通信量の移り変わりを図4に示す。この図より、6-QUEEN では推論開始時から終了時まで常時通信を同程度行っているが、10-QUEEN　では推論開始終了間際に通信総数の6割の通信を行っている事が分かる。しかし、このように頻繁に通信を行うのは、推論開始直後で未だ各PEが仕事をもらっていない時と、終了直前の殆んどのPEが大きな仕事を持っていない時と考えられる。

(3) PE内の処理内容

ある1台のPEが推論処理を行う時、内部でどのような処理が行われているかを示すのが図5である。図5では、7QUEEN をPE 13 台で実行した場合、ある1台のPEが行った処理のうちわけを示す。この図より、約3/4 の時間がプロログの処理に、残り1/4 の時間が並列処理の為に行った処理である事が分かる。この事は、7-QUEEN PE 13 台の台数効果が理想値の65% にとどまる一因であると考えられる。

5．まとめ

枝分け方式によるハードウェアシミュレータを製作し、データ収集を通じて、データ転送、排他制御の高速化を行う事で、高速な並列推論マシンが製作出来る事が分かった。しかし、PEの処理でデータ転送を行う為のオーバヘッドがまだ大きく改善の余地が残っている事が分かった。

尚、この研究は第5世代計算機に関する研究の一環として、ＩＣＯＴからの委託により行われたものである。

参考文献

[1] 久門ほか「並列推論処理システム－改良型節単位処理方式」第30回情報処理学会全国大会 7C-8 PP.217-218

[2] 板敷ほか「並列推論処理システム－改良型節単位処理方式の実験」第30回情報処理学会全国大会 7C-7 PP.215-216

[3] 相馬ほか "A NEW PARALLEL INFERENCE MECHANISM BASED ON SEQUENTIAL PROCESSING" IFIP TC-10 Working Conference On Fifth Generation Conputer Architecture, July 15th-18th, 1985 (To be published)

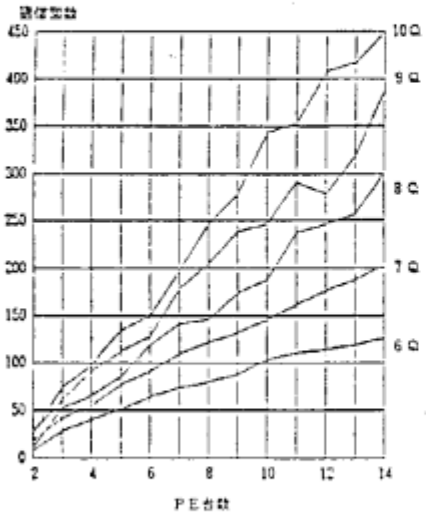


図3　PE台数に対する通信回数

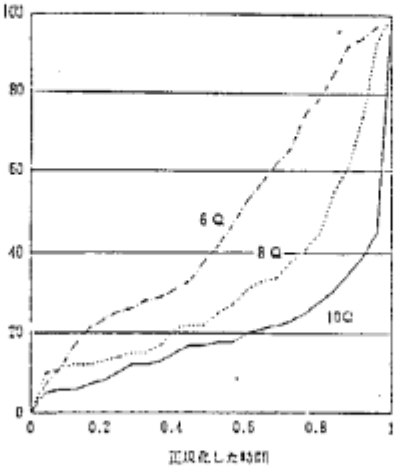


図4　実行中の通信回数の推移

図5　PE内の実行時間比率

述語論理に基づく文章理解実験システム

平井　章博、坂田野　陽一、(*)新田　義彦
（（株）日立製作所　システム開発研究所、(*)基礎研究所）

1．はじめに

計算機による自然言語処理の研究開発の活発化に伴い、文章の意味理解処理の必要性が高まっている。そこで、我々は、文章理解方式開発の第一歩として、一階述語論理式(Horn節)を意味表現として用いた文章理解の実験システムを、論理型言語を使用して作成した。本実験システムは、小学校国語教科書の文章の要約を理解し、その内容に関する質問の回答を生成するシステムである。以下、本システムにおける意味表現の枠組み、意味表現生成方式を中心に、システムの概要を述べる。

2．意味表現

意味表現の枠組みとしては、意味ネットワーク、フレーム等があるが、一階述語論理の(1)semantics,syntaxの簡明さ(2)対応するprogramming言語（論理型言語）の存在、を考慮し、一階述語論理式(Horn節)を本システムの意味表現として用いることとした。本システムの意味表現の基本となる述語の形式は、次の通りである。

述語記号(時間index,md(ﾓｰﾀﾞﾙ,ｱｽﾍﾟｸﾄ,程度,場所,連用修飾),理由,
格要素1,･･･,格要素n)

ここで、時間indexは、事象の生起時間を示すインデックス、程度、場所、連用修飾は、日本語文中の自由格要素等を述語論理の形式に納めるための項、理由は、推論の根拠を保持するための項である。意味表現法の詳細を以下、項目別に述べる。

(a)対象物の表現

名詞類の指示する対象物の意味表現は、識別ラベルと呼ぶ定数（述語論理における定数）を割り当てることにより行う。これにより、skolem化が同時に行われる。識別ラベルの性質は、述語によって表す。

(b)時間表現

時間解析に要求される厳密性を緩和するために、時間モデルとして、格子的時間モデル[杉原 74]を採用した。時間関係を表現する最もprimitiveな述語としてt_o(時間index1,時間index2)（時間index2の方が後という意味）を設定し、これを基としたいくつかの時間推論の機構を用意した。

上記の意味表現法では、例えば、"ごん(obj1)は兵十(obj2)に栗(obj3)を届けた。"という文の意味は、

todokeru(t1,md([],w,[],[],[]),[],obj1,
obj2,obj3)

という論理式で表現される。ただし、t1は時間index、wは完了相を表すアスペクト・コードである。

2．システム構成

本実験システムの構成を図1に示す。全体の処理は、以下の順で行われる。

(1)格解析

漢字かな混じりの入力文章の形態素解析、格[平井 84]構造解析を行い、格関係グラフを生成する。格関係グラフとは、格構造を表すグラフで、図2に示すように、属性群を有するノードを格コード・ラベル付きの有向枝によって連結したものである。

(2)意味表現生成

格関係グラフを基に意味表現(論理式)を生成する。

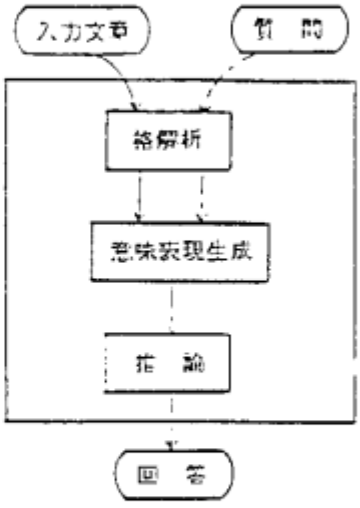


図1　システムの構成

詳細は後述する。

(3)推論

ユーザから質問文が入力されると、(1),(2)の処理により、質問文に対応する論理式が得られる。その論理式を一階述語論理のproverにかけることにより、入力文章の意味表現（論理式）と常識として蓄えてある論理式を基にした推論を行わせ、その結果得られる情報（変数のユニフィケーションの結果も含む）より回答文を生成する。

3．意味表現生成法

基本的には、各用言の辞書に登録されている条件付きtemplateを利用して行う。条件付きtemplateとは、用言の受け格に対する条件と論理式の"鋳型"となるtemplateの対で、その概念は、図3の通りである。具体的な処理手順を以下に示す。

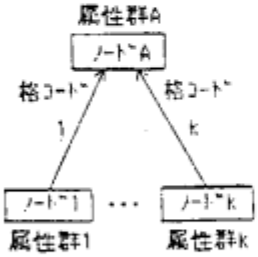


図2 格関係グラフ

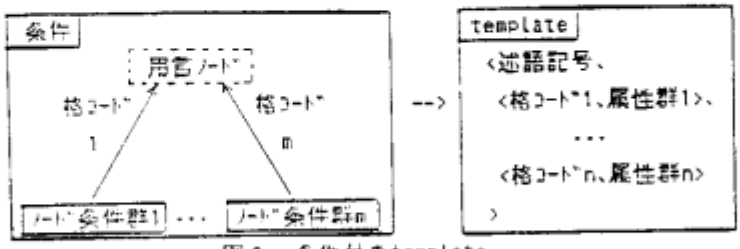


図3 条件付きtemplate

(1)templateの選択：注目している格関係グラフと整合する条件を持つ条件付きtemplateを探索し、そのtemplateの獲得を行う。

(2)時間indexの割り当て：文の出現順、時間関係を表す副詞、接続詞等を手掛かりとして時間indexを割り当てる。

(3)アスペクト、モーダル情報等の獲得：格関係グラフ中の用言ノードの属性群より、アスペクト、モーダル等の情報を得る。

(4)識別ラベルの割り当て：格関係グラフ中の名詞類ノードの指示対象が、概念であるか、個体であるかを判断し、この情報を基に、名詞類ノードの指示対象と既出の識別ラベルとの同定を試みる。同定成功の場合は、同定された識別ラベルを、同定不可の場合は、新たに生成した識別ラベルを注目している指示対象に割り当てる。

(5)論理式（意味表現）の表明：前段までで得られた情報をまとめ上げ、templateに示されている述語記号を持ち、格要素の項としてtemplate中の各格コードに対応する識別ラベルを有する論理式の表明を行う。

(6)概念指示に対する処理：格関係グラフ中の名詞類ノードの指示対象が概念の場合、項がその概念に属することを調べる述語を負のリテラルとして持つ論理式も表明する。

(7)前向き推論：必然性の無い因果関係の認知のために、原因と結果に対応する事象の成立の証明から、両事象間の因果関係の存在を論理的帰結として導く推論を前向き推論により行い、その結果を論理式として表明する。

4．まとめ

小学校国語教科書に掲載されている"ごんぎつね"を要約し、これに、本実験システムを適用したところ、穴埋め的質問（誰が、何を、等）、登場人物の心理状態や理由を問う質問に答えさせることができた。本実験システムの作成を通じ、述語論理式の形式の意味表現は、(1)モジュール性が高く、常識の記述が容易、(2)範囲を限定すれば、日本語文の格関係ネットワークからの意味表現生成が容易、等の長所を持つことが判明した。

なお、本研究の一部は、第5世代コンピュータ・プロジェクトの一環としてＩＣＯＴからの委託により行ったものである。

参考文献

[杉原 74] 杉原丈夫：時間の論理、早稲田大学出版部、1974

[平井 84] 平井秀博、新田義彦、他："日本語辞書シソーラスにおける格体系について" 情報処理学会第29回全国大会、4J-7、1984

7N−6

談話理解システムDUALS第2版の構想

向井国昭，田中裕一，安川秀樹*1)，三吉秀夫，平川秀樹*2)，瀧塚孝志，横井俊夫
（財団法人　新世代コンピュータ技術開発機構）
*1) 現在　松下電器産業株式会社
*2) 現在　株式会社東芝　総合研究所

1．はじめに

DUALS(Discourse Understanding Aimed at Logic-based Systems) [1,2] は談話理解の計算モデルを構築することを目的とした談話理解実験システムである。これは DCG, GPSG, LFG 等の構文論や状況意味論(situation semantics) [3,4] などの新しい言語理論の成果を取り入れて，談話の構造を明らかにしようとするものである。また自然言語処理だけでなく，知識表現，問題解決等，知識情報処理の要素技術を統合的に組込んだシステムである。既に第1版は完成し，現在はさらに状況意味論的要素を取り込んだ形の第2版を作るべく検討を行っている。第1版では小学校三年生用の国語読解力問題を対象としたが，第2版では代名詞・省略の同定，焦点の移動管理の各処理を充実させ，小学校6年程度の文章理解を目指す。ここでは第2版の構想を述べる。

2．状況意味論

状況意味論では，現実の世界とその集合論的モデルである抽象的世界を考える。現実の世界を構成する現実の状況は人間の知覚の対象となり得るものである。一方，抽象的状況は発話に対して，その解釈を与えるものである。

2.1 文の意味

状況意味論の基本的な考えでは叙述的な文（発話）の意味は，発話がなされた状況と発話で描かれた状況との関係であるととらえる。つまり発話状況d(descourse situation)，話者のコネクションc(speaker connections) 及び記述状況e(described situation) というものを導入することにより文（発話）αの意味は，

$$d, c[\alpha]e$$

と表現される。発話状況は「発話が誰によって，いつ，どこでなされたか」というような情報を記述する状況である。話者のコネクションは話者が発話によって何を参照しているのかを与える関数である。

2.2 タイプ，不定項

状況中のオブジェクトに関する不変要素としてタイプ(type)がある。オブジェクトの基本タイプ(basic type)には次のものがある：

IND	個体のタイプ
RSIT	実状況のタイプ
SET	集合のタイプ
SIT	状況のタイプ
LOC	ロケーションのタイプ
POL	極性（真偽値）のタイプ
RELn	n 項関係のタイプ

タイプのインスタンスとして不定項(indeterminate) を導入する。各基本タイプTに対して，不定項

$$T1, T2, T3, \cdots$$

が存在し，それはタイプTのオブジェクトにアンカー(anchor)される。アンカーとは基本不定項のある集合を定義域とする関数σで，σ（Tn）が定義されるならば，それがタイプTのオブジェクトになっているようなものである。またパラメータと条件Cを持つようなタイプの対する不定項，複合不定項(complex indeterminate)

$$[X \mid C]$$

が存在する。

3．アプローチ

DUALS の第1版は開発の時間的な制約もあったため，談話理解に於ける本質的な問題を強力なヒューリスティクスにより回避している部分が多い。第2版は第1版の色々な問題点をふまえて，次のような点に重点を置いて，状況意味論の枠組みからアプローチしたい。

①「約束する，命令する，質問する」などの発話行為は状況間のコンストレイントとして記述する。

②コンストレイントに記述される状況にメンタルステートのモデルを構築する。このモデル上へ「信じる，疑う，知る，見る」などの態度(attitude)に関する基本動詞の意味記述を行う。メンタルステートの表現についてはBarwise とPerry 自身により提案されているのでそれを参考にする。

③発話行為の定義にはロケーション（時間・空間）が密接に関係している。第1版ではロケーションを全く扱わなかった。それをどう扱うかは第2版の大

きな課題である。

われわれは，状況意味論に従って談話理解の計算モデルをタイプとコンストレイントを用いてフォーマルに定義する。

次のような条件を満たす状況の列 $H_0, H_1, \ldots, H_n$ と関係の列 $\delta_1, \ldots, \delta_n$ のことを談話理解という。ここでは簡単のために状況をメンタル・ステートと同一視する。

$$H_0 \xrightarrow{\delta_1} H_1 \xrightarrow{\delta_2} \cdots\cdots \xrightarrow{\delta_n} H_n$$

[条件 1]

δ_i (H_{i-1}, H_i) である。

[条件 2]

δ_i は発話 α_{i-1} に応じて文法規則とレキシカル項目から構成的(compositionally) に決まるコンストレイントである。

[条件3]

各メンタル・ステート H_i は以下の情報の総体である。

belief の集合
knowledge の集合
want の集合
intention の集合

[条件 4]

発話行為論は次のような形式のコンストレイント群として記述する。

$$E_1 \Rightarrow E_2$$

E_1, E_2 はイベント型であり，E_1 は E_2 をinvolve すると読む。

4. CIL

CIL(Complex Indeterminate Language) [5] は状況意味論に基いた談話理解システムの記述言語となるPrologの拡張言語である。2.1で述べた複合不定項(Complex Indeterminate) を導入している。不定項の導入に対応して拡張された単一化処理(unification) を基本計算機能として持つ。またCIL はコンストレイント型の言語を志向しているためfreeze機能を組込んでいる。これにより，変数の束縛に関係した条件によりゴールをサスペンドしたり，解除したりできる。すなわちCIL は次の等式で表現されるようなプログラミング言語である。

CIL = Prolog
+ 不定項
+ freeze

CIL による非常に簡単な発話行為に関するプログラム及び実行例を次に示す。これは「王様が『寒い。』と言った」状況から導かれる状況を求めるプログラムである。

```
test(L)<-discourse([[_,(say,king,cold),1)], L).

discourse(X, X)<- final(X).
discourse(X, Y)<-
        speech_act_plan(Plan),
        apply_plan(Plan, X, Z),
        discourse(Z, Y).

apply_plan([], X, X).
apply_plan([P|R], X, Y)<-
        involve(P, E1, E2),
        subset(E1, X),
        union(E2, X, Z),
        apply_plan(R, Z, Y).

member(X, [X|_]).
member(X, [_|Y])<-member(X, Y).

union([], X, X).
union([X|Y], Z, U)<-(member(X, Z)=>union(Y, Z, U),
                     union(Y, [X|Z], U)          ).

subset([], _).
subset([X|Y], Z)<-member(X, Z), subset(Y, Z).

final([(_, final, 1)|_]).
final([_|R])<-final(R).

speech_act_plan([request, response]).

involve(request,
        [(L, (say, king, cold), 1)],
        [(L, (request, king, servant, to_close), 1)]).
involve(response,
        [(L, (request, king, servant, TO_DO), 1)],
        [(L, (TO_DO, servant),1),(_,final,1)]).

**********************************************************

| ?- test(0).

0 = [(_772,final,1),
     (_76,(to_close,servant),1),
     (_76,(request,king,servant,to_close),1),
     (_76,(say,king,cold),1)]

yes
```

第２版は逐次型推論マシンPSI 上で開発予定である。

[参考文献]

[1] 三吉秀夫，その他： 状況意味論に基く談話理解システムDUALS －そのインプリメンテーション，自然言語研究会50-7, 1985.

[2] 田中裕一，その他： 状況意味論に基く談話理解システムDUALS －その基本原理，自然言語研究会50-6, 1985.

[3] J. Berwise, J. Perry: Situations and Attitudes, MIT Press, 1985.

[4] J. Berwise: Lectures on Situation Semantics, CSLI, Winter Quarter, 1984.

[5] K. Mukai: Horn Clause Logic with Parameterized Types for Situation Semantics Programming, ICOT Technical Report, TR-101, 1985.

並列推論マシンＰＩＭ－Ｒの ハードウェア・シミュレータ －開発思想－

尾内理紀夫　　杉江衛　　吉住誠一
（財）新世代コンピュータ技術開発機構　（株）日立製作所

1．はじめに

第五世代コンピュータ・プロジェクトでは述語論理型言語をベースにした知識情報処理システムの研究開発を行なっている．述語論理の基本操作は推論であるから，そのハードウェアは推論マシンと呼ばれ，また，推論が基本操作であるから，このマシンは並列動作が基本となる．

並列推論方式あるいは述語論理型言語として，現在いくつかのものが提案されているが[Moto 84],[Ito 84],[Clar 84],[Shap 83]，我々は，ICOTが核言語第１版(KL1(84))のベース言語として位置づけているPrologとConcurrent Prolog[Shap 83] を対象言語として選択し，並列推論方式としては，reduction 概念に基づくresolvent 生成過程の並列実行を基本動作とし，PrologをOR並列に，Concurrent PrologをAND 並列に処理する方式を採用した並列推論マシンPIM-R (Reduction-based Parallel Inference Machine)のアーキテクチャを提案し[Onai 85],研究開発を行なってきた．

本稿では，PIM-R シミュレーションの経緯と，ハードウェア・シミュレータ開発の目的と条件について述べる．

2．PIM-R シミュレーションの経緯

PIM-R は，並行プロセス間チャネル用のリモートアクセス可能な分散化共有メモリ(Message Board) の導入[Onai 84],効率的packet分配のためのIntelligent Network Node の導入,Ground Instanceである長い構造体データ格納のための構造体メモリの導入[Hasu 85a]をアーキテクチャ上の特徴としている．　また，PIM-R はstructure-copy方式を採用しているが，copyおよびそれに伴う処理量の低減およびnetwork 通過packet数とpacket幅の低減のためreverse compaction, only-reducible-goal-copy,独特のprocess 構成法を採用している[Shim 85a]．

我々は，このPIM-R アーキテクチャの提案と共に，アーキテクチャ検証の第一歩として，PIM-R の原理的動作確認のためにPrologで記述されたソフトウェア・シミュレータを開発し，各種データを収集した．その結果，PIM-R が Prolog,Concurrent Prologに内在する並列性を引き出すこと，Message Board のための専用Controllerの導入効果，Intelligent Network Nodeによる子プロセスの動的分配効果，Process Pool Controller におけるdead処理／fork downパケット生成及び転送処理の休止効果，個々のプロセスが持つreduction レベル（プロセス木の根からの深さ）による疑似depth-first 実行の効果等を確認した[Onai 85]．しかし，このソフトウェア・シミュレータは原理的動作確認を目的としており，データ構造等PIM-R を更に詳細にシミュレートすることが難しく，また，シミュレーションに時間がかかるため，次節に述べる開発目的と条件にかなうハードウェア・シミュレータを開発することにした．

3．ハードウェア・シミュレータ開発の目的と条件

3.1　開発目的

以下の４点を開発の主目的とした．

①PIM-R 詳細構造のシミュレーション
②シミュレーション速度の向上
③十分なメモリ空間
④並列環境の実現

①～③はProlog版ソフトウェア・シミュレータでは実現が難しい点である．特に④の並列環境の実現を目的の一つにしたのは，並列環境ゆえに生起する各種の問題（デッドロック等）に関するデータ，あるいは，並列環境のためのソフトウェア・ユーティリティ（デバッガ等）構築のための各種データは，逐次マシン上ではなく，物理的に異なるプロセッサ上で収集する必要があるからである．

3.2　開発条件

早期開発と柔軟性を条件とした．なぜならば，PIM-R のデータ収集／評価，そして，そのフィードバックに基づくPIM-R の改良，そしてまた，データ収集／評価というサイクルに耐えるためには，ハードウェア・シミュレータは柔軟でなければならない．また，Prolog版ソフトウェア・シミュレータによるPIM-R の原理的動作確認に引き続き，ハードウェア・シミュレータによるデータ収集をすることが望ましく，早期開発を目ざす必要があった．

このように，早期開発と柔軟性実現のため，我々は MC68000 搭載のシングルボードマイクロコンピュータ（以後ＳＢＣと略す）を用いて，PIM-R のInference Moduleを実現することとした．そして，PIM-R の各種機能モジュール（PPU,UU,etc.)は，ソフトウェアでインプリメントし，PIM-R 機能モジュールの変更改良に容易に追従できるよう

にした．　また，PIM-R のネットワークは，メッシュ構造を出発点としているが，各種応用分野あるいはKL1 の変更，改良によっては，異なるネットワーク構造をも検討してみる必要がある．そこで，ハードウェア・シミュレータのネットワークは，固定的なハードウェア構成ではなく，柔軟な共有メモリによる構成とし，その上で各種ネットワークをシミュレートすることとした．

実際に並列環境を実現するするためにＳＢＣを複数台(当初８台，現在16台に拡張中）配列し，ＳＢＣと共有メモリとを結ぶ共有バスには，ハード化したバスアービタを導入した．　共有メモリへのアクセス競合は，このバスアービタと試作したインタフェース回路により解決されるので，各ＳＢＣは，互いに他を気にすることなく，共有メモリにアクセスが可能となっている．また，本ハードウェア・シミュレータ上で，各種応用テストプログラムを走行できるようにするため，各ＳＢＣには，十分なローカルメモリを付けることとし，10M バイトまで実装可能とした．

また，ＳＢＣ群と共有メモリのスーパーバイザとしてVAX11/780 を導入した．　VAX11/780 と（ＳＢＣ＋共有メモリ）系との接続にはGPIBを用いたが，これも，インタフェースが簡単で，VAX11/780 用のインタフェースが市販されており，早期開発の条件にかなったからである．

4．おわりに

このようにしてリダクション方式並列推論マシンPIM-R のハードウェア・シミュレータを開発した結果，Prolog版ソフトウェア・シミュレータによる原理的動作確認に引き続き，ハードウェア・シミュレータによるデータ収集を開始することができた．　本ハードウェア・シミュレータは，PIM-R の詳細構造シミュレーションおよび複数種類（二種類のメッシュ構造と鎖状構造）のネットワークシミュレーションが可能であり，テストプログラムとしてQuicksort，N-Queensプログラムを選択し，シミュレートした結果，PIM-R がプログラムに内在する並列性を引き出すことを確認した[Sugi 85a]，[Iwas 85]，[Yone 85]，[Sugi 85b]．

このように，前記開発条件を満たし，目的①～④を達成するハードウェア・シミュレータを開発することができた．

現在これを用いてシミュレーションを行なっており，今後は，本シミュレータ上でのソフトウェア・ユーティリティの検討を行なう予定である．

なお，一方，シミュレーション規模の向上（Inference Module16～64台以上）を主目的に，PIM-R の詳細構造シミュレーションが可能な高速ソフトウェア・シミュレータを作成し，データを収集中である[Hasu 85b]，[Shim 85b]，[Mats 85]．

最後に，御理解いただいた渕一博ICOT研究所長，千葉常世日立中研第８部長，御指導，御討論いただいた東京大学元岡達教授（プロジェクト推進委員長），田中英彦助教授（ワーキンググループ１主査）ならびに関係各位に深謝する．

〈参考文献〉

[Clar 84] Clark,K.L. and Gregory,S.,"PARLOG:Parallel Programming in Logic", Research Report DOC 84/4, Dept. of Computing, Imperial College London, 1984.

[Ito 84]Ito,N. and Masuda,K., "Parallel Inference Machine Based on the Data Flow Model",International Workshop on High Level Computer Architecture 84,1984.

[Iwas 85] 岩崎他,"並列推論マシンPIM-R のハードウェア・シミュレータ -制御プログラムの方式-" 本予稿集,1C-3,1985

[Hasu 85a]益田他，“並列推論マシンPIM-R の構造体メモリの一構成法”，情報処理第30回全国大会，1985

[Hasu 85b]益田他，“並列推論マシンPIM-R の詳細ソフトウェア・シミュレーション -構造体メモリ導入効果- ”本予稿集，1C-8,1985

[Mats 85] 松本他,"並列推論マシンPIM-R の詳細ソフトウェア・シミュレーション-Concurrent Prolog解析-"本予稿集，1C-7,1985

[Moto 84] Moto-oka,T.,Tanaka,H. et al.,"The Architecture of a Parallel Inference Engine -PIE-", Proc. of Int.Conf.on FGCS 1984,ICOT, 1984.

[Onai 84] 尾内，麻生，“並列環境におけるConcurrent Prolog の実現法”，情報処理第29回全国大会，1984.

[Onai 85] 尾内，清水，麻生，益田，松本，“リダクション方式並列推論マシンPIM-R のアーキテクチャ”，Logic Programming Conference '85 東京，1985

[Shap 83] Shapiro,E.Y.,"A subset of Concurrent Prolog and Its Interpreter", ICOT, TR-003, 1983.

[Shim 85a]清水他，“並列推論マシンPIM-R のプロセス内部表現”，情報処理第30回全国大会，1985

[Shim 85b]清水他,"並列推論マシンPIM-R の詳細ソフトウェア・シミュレーション -OR 並列Prolog解析 -" 本予稿集，1C-6,1985

[Sugi 85a]杉江他,"並列推論マシンPIM-R のハードウェア・シミュレータ -システム構成-"本予稿集，1C-2，1985

[Sugi 85b]杉江他，“並列推論マシンPIM-R のハードウェア・シミュレータ －評価方式－”本予稿集，1C-5，1985

[Yone 85] 米山他，“並列推論マシンPIM-R のハードウェア・シミュレータ －ハードウェアの方式－”本予稿集，1C-4，1985

ＰＳＩの性能評価（３）

SC-2

―― マイクロインタプリタの動特性 ――

中島　克人　　横田　実　　山本　明
（三菱電機）　（新世代コンピュータ技術開発機構）　（沖電気）

１．はじめに

逐次型推論マシンＰＳＩのハードウェア，ファームウェアの開発が一段落し、現在それらの評価を行なっている。性能評価(1)[1]では、ベンチマーク・プログラムを用いた基本性能測定を行なった。性能評価(2)[2]では、実行マイクロ命令解析によるハードウェア使用効率，キャシュメモリのヒット率等の評価を行なった。本報告では、ＰＳＩのシステム記述言語ＥＳＰで書かれた応用プログラムを対象として、プログラム動特性と、その実行に要するインタプリタのステップ数解析、インタプリタの使用しているバッファの効果等の評価結果について述べる。

２．測定対象プログラム

FGCS'84 のＰＳＩデモンストレーションに用いたものの中から、ゲーム・プログラムの代表として「8-PUZZLE」、比較的決定的(deterministic) なプログラムとしてビット・マップ・ディスプレイの制御の一部を行なう「WINDOW」、エキスパート・システムの例として「HARMONIZER」（入力曲は“赤とんぼ”）の３つを選んだ[3]。

３．プログラム動特性と実行ステップ数比率

表１に３つのプログラムのバックトラック率（失敗回数／ヘッド・ユニフィケーション回数または組込述語呼出し回数）を示す。また、シャロウ・バックトラックと関係が深いクローズ・インデックシングの比率も併せて示す。

ユーザ定義述語のバックトラック率は、プログラムにより大きく異なる。ただし、決定的なプログラムと思われたWINDOWでも１４．１%と以外に高い。その理由は、いわゆるプログラム内の条件判定処理がオールタネイト・クローズを並べてのヘッド・ユニフィケーションで実現されているからである。組込述語のバックトラック率は、ユーザ定義述語に比していずれも低くなっている。バックトラック率とクローズ・インデックシングの間には、特別な関係は見当たらなかった。

Backtrack 率　(%)	8-PUZZLE	WINDOW	HARMONIZER
ユーザ定義述語	３８．９	１４．１	４４．２
組込述語	２．６	６．１	１７．３
Shallow Backtrack	６１．１	５５．２	７９．０
Deep Backtrack	３８．９	２．０	１４．７
OR Backtrack	０．０	４２．８	６．３
Total	７．８	７．９	３０．６
Clause Index 率 *	２２．４	１０．２	１１．１

*: Clause Indexing 回数／ユーザ定義述語呼出し回数

表１　バックトラック率

図１にWINDOWとHARMONIZERの述語呼出し回数の種類別比率および、その処理内容別のマイクロ命令実行ステップ数比率を示す。いずれも、ユーザ定義述語のための基本制御に要する実行ステップ数(Control.Unification) が、その呼出し回数に比べ大きくなっている。特に、HARMONIZERでは、構造体引数のユニフィケーションが頻繁で、かつ、バックトラック率も高いため、"Unify" に多くのステップを費やしている。ただし、「バックトラックのための環境の復帰や変数の“ＵＮＤＯ”」(Back)の比率は予想したほど大きくはなかった。これは、トレールされていた語数が少なかった事と、バックトラック時に引数フレームが再利用できる「引数コピー方式」[4]の効果によると思われる。

組込述語に関しては、その呼出し準備(B-Call)や、引数の取出し・格納(Get-Arg) 等の共通処理が大きな比率を占め、組込述語の本来の処理は、乗・除算等を含めたとしても、かなり小さい比率でしかないことがわかった。

	Cache	Buffer	8-PUZZLE	WINDOW	HARMONIZER
①	YES	YES	１００．０	１００．０	１００．０
②	YES	NO	９９．６	９９．４	９９．６
③	NO	YES	１３５．５	１４２．８	１４８．６
④	NO	NO	１３５．５	１４２．１	１４７．９

表２　トレール・バッファの効果（実行時間比）

４．バッファの効果

ＰＳＩのインタプリタではローカル・スタックとトレール・スタックに対し、スクラッチパッド・メモリ上にバッファを設けており、メモリ・アクセス頻度の軽減を図っている。しかしながら、ＰＳＩは８Ｋ語ｘ２セットのキャッシュ・メモリを備えており、当初からバッファの効果については興味のあるところであった。シミュレータによるキャッシュ・ヒット率の測定結果は、ローカル・スタックでは９７．４～９９．４%、トレール・スタックでは９７．６～９８．７%といずれも高率であった。

表２はトレール・バッファを使用しない様にインタプリタを修正した実験の結果である。現インタプリタによる各プログラムの実行時間（①）を１００とした時間比を見ると、バッファを廃止した場合（②）の方が性能は向上している。また、キャッシュ・メモリを切り離した場合（③，④）においても、バッファがない場合（④）の方が僅かに良い結果となっている。これらの原因は、

(1)トレール・バッファをサポートするための余分なマイクロ命令ステップがかなり削減された。

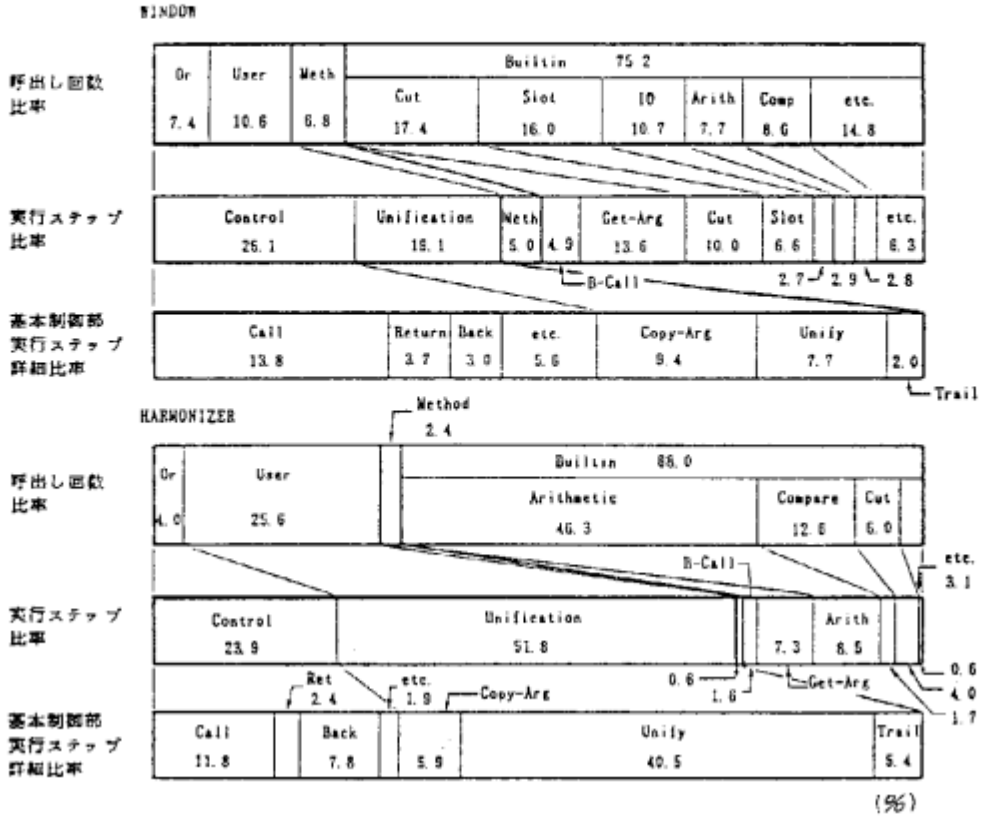


図1　述語呼出し回数と実行ステップ比率

(2)トレールのアクセス頻度がもともと低い（全メモリ・アクセスの4%程度）。
(3)キャッシュ・メモリのヒット率が非常に高い。
という事である。

ローカル・バッファに関しては、「引数コピー方式」やＴＲＯ(Tail Recursion Optimization) と密接した関係にあり、インタプリタを修正して実験するという事が簡単ではない。そこで、ローカル・バッファのアクセス頻度などの基本的なデータを計測し、それらを基に、スタック上で「引数コピー方式」を行なう場合の実行時間を試算した。

この場合も、バッファ廃止による性能低下は殆どなく、プログラムによっては、トレール・バッファと同様に向上するものもあったが、いずれにせよ1%未満であった。キャッシュ・メモリを切り離した場合の試算では、バッファ廃止による性能低下は2～3%程度となった。これらの理由もトレール・バッファと同様で、
(1)「引数コピー方式」のための2面のバッファの切換えに要するマイクロ命令ステップが削減できる。
(2)ローカルのアクセス頻度は高い（25%程度）が、全実行ステップから見たメモリ・アクセスのオーバヘッドは、なお小さい。
(3)キャッシュ・メモリのヒット率が非常に高い。
等である。

5．おわりに

3つの応用プログラムを用い、各種の定量的データを収集した結果、ＰＳＩのインタプリタ性能向上のためには、ユーザ定義述語呼出しの基本制御と、組込述語の共通処理に要するステップ数の削減が重要である事、そして、トレール・バッファは不要である事が分かった。

また、ローカル・バッファに関しても、ＴＲＯの実現には有用であるとしても、1面のバッファで済ます等の方法で、バッファ操作のために生じるマイクロ・ステップのオーバヘッドを削減しなければ、本当のバッファの効果は期待できないことも分かった。

参考文献

(1) 西川 他 "ＰＳＩの性能評価(1)" 第30回情処全国大会
(2) 中島 他 "ＰＳＩの性能評価(2)" 第30回情処全国大会
(3) 山本 他 "論理型言語ＥＳＰのプログラム特性評価" Proc. of Logic Programming Conf. '85
(4) 横田 他 "A Microprogrammed Interpreter for the Personal Sequential Inference Machine" Proc. of FCCS'84

並列推論マシンPIM-R の詳細ソフトウェア・シミュレーション —構造体メモリ導入効果—

益田 嘉直　清水 肇　尾内 理紀夫
（財団法人 新世代コンピュータ技術開発機構）

1．はじめに

並列推論マシンにおける構造体データの共有方式や処理方式に関する研究は高速な並列推論マシンを開発する上で極めて重要である。

本稿では、当機構で検討及び評価を進めている並列推論マシンPIM-R の構造体メモリ[1][2]に関して、Occam で記述した詳細ソフトウェア・シミュレーションによって得られた結果をもとにその導入効果について考察を行う。

2．構造体メモリ(Structure Memory)の特徴

Structure Memory Module(以下SMM と呼ぶ）はIM-SMM Network を介して多数台の Inference Module （以下IMと呼ぶ）と接続され、SMM 内に格納される一定の条件を満たすリストやベクタ等の構造体データは多数台のIMより共有される。現段階では、数台のIMに対して、同一の内容を持つSMM を１台ずつ接続して行き全体システムを構成する方法によりメモリアクセス競合等の集中化を回避する方法を考えている。

現在のPIM-R の構造体メモリの主な特徴としては以下のことが挙げられる[1][2]。

(1) Ground Instance レベルの共有

基本的には未定義変数を含まないリストやベクタ等の構造体データをSMM 上の専用メモリに格納することにより部分的に構造体データを共有する方式を採用する。この方式では読出しだけを行い書き換えの生じないground instance の部分だけを共有するので、共有度は低いが次々と発生するプロセス間の独立性を高く保つことが出来るという特徴がある。

(2) 構造体データのハイブリッドな格納形式

現段階では、プログラムのcompile 時にclause中の構造体データのうちプログラマによって指示された、SMM に格納すべきground instance 部分の切り分けを行い、それをSMM へ格納する。この時、図１．に示す様にリストに対してはリスト形式で、ベクタ等の構造体データに対しては一次元配列のレコード形式でそれぞれ専用のメモリに格納する。これらSMM に格納された構造体データの読出しはBlock 単位に行う。即ち、リストの場合にはList Data Memory中のcar 部とcdr 部の２セル（List Block と呼ぶ）がBlock 転送され、Unification Unit （以下UUと呼ぶ）内のバッファメモリに読出され、ベクタの場合にはVector Address Table 内のアドレスで指示されたVector Data Memory内のベクタの一まとまり（Vector Blockと呼ぶ）がBlock 転送されUU内のバッファメモリに読出される。

(3) 実行効率を考えた構造体データの格納

構造体データのground instance の部分を格納する方法としては、実行時において効率よく構造体データが処理されることを考慮し、compile 時に部分的に同一の内容を複数のポインタにより多重参照されるようにする等、最適化を試みて格納している。

(4) 引数間のLazy Unification

SMM に格納されている構造体データの参照を必要とするunification に対しては不必要な引数間unification やSMM への参照を避けるために、SMM を参照する引数間のunification を後回しにするlazy unificationを行っている。

3．SMM の導入効果

現在、筆者等はデータ内部形式等がPIM-R の詳細構造を反映している詳細ソフトウェア・シミュレータ[3][4]を開発しながら，各種データを収集中である。本シミュレータの実行はVAX11 上で行なわれる。

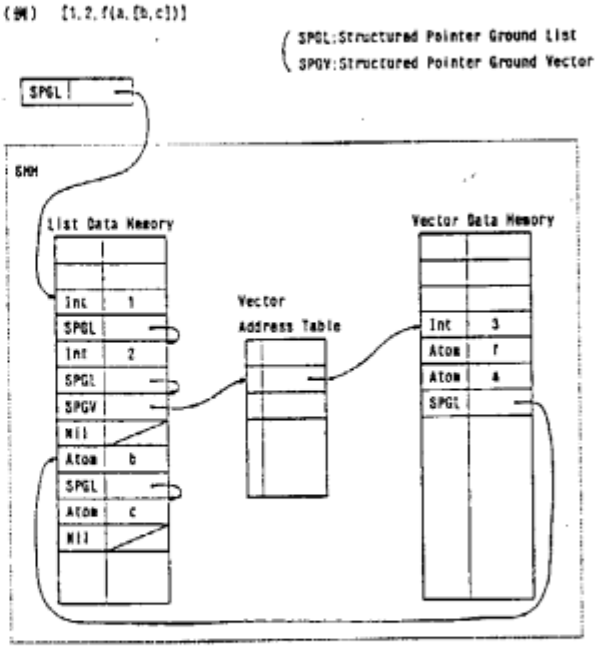


図１．構造体データの格納例

<シミュレーション条件>

①PPC(Process Pool Controller)は新しく生成されたプロセスをReady Process キューの最後尾につなぎキューの先頭から順にUUへ転送する。

②組込み述語の解がreturnした時，逆順compactionによりPTB(Process Template Block) 語長が長くならない場合は直接overwrite する。

③IN 2台の場合の分配法則は、自分→隣→自分→…の繰り返しとし、 4台以上の場合は、自分→東→南→自分→…の繰り返しとする。

評価プログラムとしては、OR並列Prologで記述した形態素解析、DCG 、数式簡単化(Equiv2)、quicksort(50要素)等を選定し各種データを収集中であるが、それらをもとにここではSHM の導入効果、特にストラクチャエリアの減少効果及びLazy Unificationの効果について考察を行う。

(1) ストラクチャエリアの減少効果

表1．に示すように、①②③の場合にはPTB 語長をSHM 導入により20～30% (A/B) 、Clause Pool 共有化の下でもSHM 導入により30～45% (C/D) 短縮できる。更にOR fork 、true return パケットもSHM 導入により10～45% (A/B) 、Clause Pool 共有化の下でもSHM 導入により20～50%(C/D) 短縮できる。なお、Clause Pool 共有化とSHM 導入の両者を併せることによりPTB 語長を50～60% (C/B) 、OR fork 、true return パケットも50～60% (C/B) 短縮できる。ここでOR fork パケットに比べてtrue return パケットはストラクチャエリアの占める割合が一般に大きくなるので若干大きな効果がでている。また、形態素解析、DCG 、数式簡単化のようにgroundにおちた構造体データを多数ポインタにより持ち回ることができるプログラムの場合にはかなりの効果が期待できるが、quicksort のように新しいリストを次々に生成して行くものについては十分な効果が得られないことが考察される。

(2) Lazy Unificationの効果

形態素解析プログラムでは表2．のようにLazy Unificationの導入により約10% の効果が得られた。通常、goalまたはclauseのどちらか一方のn番目の引数がSHM を参照していて、その次以降の引数間unification が失敗する可能性がある場合Lazy Unificationの効果が期待できるが、形態素解析ではこのような場合が幾分あることを示している。この他にも数式簡単化で僅かながらLazy Unificationの効果が得られたが、DCG やquicksort のようにLazy Unificationの効果が全然みられない場合もある。

表2．Lazy Unificationの効果

形態素解析	lazy(L)	normal(N)	(N-L)/N(%)
SHM read要求回数	822	908	9.5
SHM read return 総長	2772	2956	6.2

4．おわりに

上記のようにPIM-R のSHM の導入効果について考察を行ったが、今後も引き続き詳細ソフトウェア・シミュレータにより並列推論マシンにおける構造体メモリに関する処理方式の検討及び評価を進める予定である。

最後に、日頃ご指導をいただいた内田俊一第４研究室長はじめ並列推論マシングループ諸氏に感謝の意を表する。

〈参考文献〉

[1] 尾内他“リダクション方式並列推論マシンPIM-R のアーキテクチャ” Logic Programming Conf.'85,Tokyo

[2] 益田他“並列推論マシンPIM-R の構造体メモリの一構成法” 情報処理第３０回全国大会、6C-5,1985.3

[3] 清水他“並列推論マシンPIM-R の詳細ソフトウェアシミュレーション－OR並列Prolog解析－” 本予稿集 1C-6

[4] 松本他“並列推論マシンPIM-R の詳細ソフトウェアシミュレーション－Concurrent Prolog 解析－” 同 1C-7

表1．ストラクチャエリアの減少効果

評価プログラム		パケット種類	平均語長 (Clause Pool非共有) SMM 使用:A	SMM 未使用:B	A/B(%)	平均語長 (Clause Pool共有) SMM 使用:C	SMM 未使用:D	C/D(%)
①	形態素解析プログラム	PTB	44.2	64.2	68.8	23.6	43.7	54.1
		OR-fork	35.8	47.4	75.5	17.9	29.7	60.2
		true return	32.3	52.6	61.4	24.0	44.3	54.2
②	DCG プログラム	PTB	29.3	37.9	77.3	14.5	23.1	62.9
		OR-fork	28.3	34.6	81.3	13.2	19.7	66.8
		true return	51.3	86.0	59.7	42.8	77.4	55.3
③	数式簡単化プログラム (Equiv2)	PTB	35.3	43.4	81.3	20.4	28.5	71.5
		OR-fork	35.9	40.8	88.0	20.5	25.4	80.9
		true return	54.5	98.5	55.3	46.9	90.9	51.6
④	quicksort プログラム (50 要素)	PTB	85.7	94.2	91.0	61.7	74.2	83.1
		OR-fork	63.3	73.0	86.7	42.7	52.4	81.6
		true return	132.7	144.5	91.8	122.7	134.5	91.2

並列推論マシンPIM-R の詳細ソフトウエア・シミュレーション
—OR並列Prolog解析—

清水 肇　松本 明　尾内 理紀夫
(財団法人 新世代コンピュータ技術開発機構)

1. はじめに

我々は，OR並列Prolog(以下Prologと略す)，及び，Concurrent Prolog を実行するリダクション方式並列推論マシン:PIM-Rの検討を行ない，また，基本アーキテクチャの確認のためにDEC 2060/VAX-11 上にDEC-10 Prolog/C-Prologで記述したソフトウェア・シミュレータを開発し，各種データを収集してきた [1]。

そして，新たに，VAX11 上に，シミュレーション速度と規模の向上を主目的とし，データ内部形式等の詳細構造を正確に反映した詳細ソフトウェア・シミュレータを並行プロセス記述機能と通信記述機能を持つ並列言語Occam を用いて作成している。

本稿では，Prologプログラム用のPIM-R 詳細ソフトウェア・シミュレータの構成とPrologプログラムのシミュレーション結果（構造体メモリ導入効果は[3])について述べる(Concurrent Prologプログラムのシミュレーション結果は[2])。

2. シミュレータの構成

PIM-R のProlog用詳細ソフトウェア・シミュレータは，図 1に示すようにInference Module(IM)と，Structure Memory Module(SMM)という二種類のModuleとそれらを結ぶ Network から構成される。

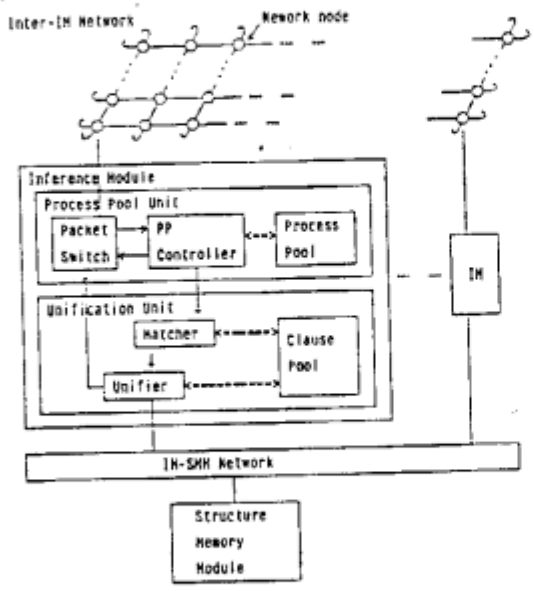


図 1. シミュレータ構成図

IMは，clauseを格納するClause Pool(CP) を用いたclauseの第一引数による絞込み(Matcher部) や，実際のUnification と組込み述語の実行(Unifier部) を行なうUnication Unit(UU)と，パケットの制御(Packet Switch部)やprocess を格納するProcess Pool(PP)を用いてprocessの生成，更新，消滅などの処理(Process Pool Controller部:PPC) を行なうProcess Pool Unit(PPU)からなる。

SMM は，ある一定の条件を満足した（現在はprogram 中でgroundに落ちている）リストやベクタ等の構造体を格納する。

他のIMへ新しい子process の分配(OR forkパケット) や，その子process の成功情報(true returnパケット)，消滅情報(fork downパケット) は，メッシュ型Inter-IM Networkにより転送される。

SMM に格納されている構造体の読み出し要求(SMM read要求パケット) やその結果(SMM read returnパケット) は，バスを仮定したIM-SMM Networkにより転送される。

ここで，process とは，goal列の制御情報を格納するProcess Control Block(PCB)，PCBの数を管理するProcess Life Block(PLB) ，及び，PCB と対になっていてgoal列のテンプレートを格納するProcess Template Block(PTB) から構成され，論理的にひとまとまりとなって同一IMに割り付けられている。

また，clauseは，ヘッダー，各変数のバインド情報を格納する変数エリア，goal列（リテラル列）を格納するリテラルヘッダー，リテラルエリア，及び，リストやベクタ等の構造体データを格納するストラクチャエリアからなり，PTB やOR fork パケット，true return パケットはclauseに準じた構成をとっている[1] 。

子process が成功した時，PTB のコピーをとり，変数をバインドし，不要になったリテラルを抜きさるが，この時リテラルは逆順に格納されており，また，構造体データは，ストラクチャエリアの先頭番地からのdisplacementでさされているため，不要になったリテラルをリテラルエリアの後から抜き，ストラクチャエリアを前詰めするだけでよく，ポインタのはりかえは必要としない。この手法を逆順コンパクションとよぶ。

なお，シミュレーション中のdeadlockをさけるため，Inter-IM Networkの各Network NodeとPPU にそれぞれ Queue を有している。

3．シミュレーション条件

①Hatcher,Unifier 等は，Occam のPAR コンストラクトを用いて実現している。

②PPC は新たに生成されたPCB をReady Process Queue (RPQ) の最後尾に繋ぎ，RPQ の先頭に格納されているPCB に対応するPTB からgoalを切り出しUUへ転送する。

③組込み述語の解がreturnし，逆順コンパクションによりPTB 語長が長くならない場合，新しいPTB をもとの位置に直接overwrite する。

④unit clause 以外のclauseとのunification に成功した時に，新しい子process を分配する(OR forkパケット）が，その分配方式は，IH 2台の場合は，自分→隣→自分→…の繰り返しとし，4台以上の場合は，自分→東→南→自分→…の繰り返しとする静的な手法をとる（図 2）。

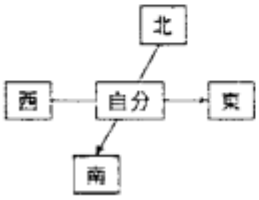


図 2

4．シミュレーション結果

(1) プロセスの分配とNetwork traffic

6queens をシミュレーションした時(IH 9 台）の各IH内process 数とネットワーク上のOR fork パケット数を図 3に示す。本シミュレーションのプロセスの分配方式は静的であるため，最大process 数 288(IH 5)と最小process 数 179(IH 0)とでは約 1.6倍の開きがある。今後process 数をバランスさせるためにNetwork nodeがPPU のバッファ長を調べ，バッファ長のより短いIHへOR forkさせる方式[1] の効果を調べる予定である。

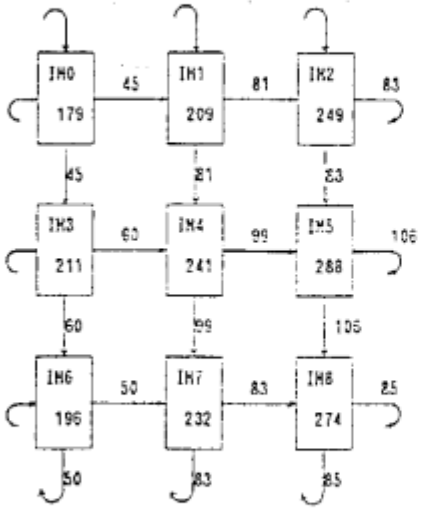


図 3. process 数とOR fork パケット数の分布

(2) Clause Pool 共有化効果

IH 2台の時，6Queens 実行時のtrue return, OR forkパケット及びPTB に関する各エリア長の収集データを表1に示す。但し，リテラル長＝リテラルヘッダー長＋リテラルエリア長とする。 ここで，Clause Pool をPPUからもアクセス可能とすれば，リテラルヘッダー，リテラルエリアをPTB の中に置く必要はなくなり，PTB語長をリテラル長分の48% が短縮でき，PPU におけるprocess 生成，更新，UUでのunification に伴うcopy量を減少することができる。 更に，true return パケット，OR forkパケットもそれぞれ54%,32% 短縮でき，Network traffic がその分減少する。

表 1

	PTB	OR fork パケット	true return パケット
平均語長(W)	44.8	43.7	30.9
平均ヘッダー長	3	4	4
平均変数エリア長	10.0	7.9	7.8
平均リテラル長(L)	21.7	23.4	9.9
平均ストラクチャエリア長	10.1	8.4	9.1
L/W (%)	48.4	53.5	32.0
W-L	23.1	20.3	21.0

(3) 組込み述語

現在組込み述語は全てgoalとしてUUへ送り処理しているが，unification を必要としない組込み述語のgoal中に占める割合は高い(6queensの場合44%)ので，これらはUUへgoalとして送らず，PPU 内で処理する方式を検討中である。

5．おわりに

並列推論マシンPIM-R 上でのPrologの詳細シミュレータの構成とその結果について述べた。今後はシミュレーション結果をもとにPIM-R の改良をしていく予定である

最後に、御協力いただいた（株）日立製作所　杉江衛、米山貢、岩崎正明各氏，御指導いただいた内田俊一　第4研究室長，ならびに関係各位に深謝する。

［参考文献］

[1] 尾内ら，“リダクション方式並列推論マシンPIM-R のアーキテクチャ”，Logic Programming Conf. '85

[2] 松本ら，“並列推論マシンPIM-R の詳細ソフトウェア・シミュレーション　－Concurrent Prolog 解析－”，本全国大会予稿集1C-7

[3] 益田ら，“並列推論マシンPIM-R の詳細ソフトウェア・シミュレーション　－構造体メモリ導入効果－”本全国大会予稿集1C-8

3C-6

高性能PROLOGマシン（CHI）のファームウェア

丸山裕之* 中嶋良成** 福田伸一** 小長谷明彦** 島津秀雄** 梅村 隆**

* 日本電気技術情報システム開発㈱ ** 日本電気株式会社

1．はじめに

現在開発中の逐次型推論マシン（CHI）[1]のファームウェアに関してその構成と機能、またプロセス管理と割込処理方法について報告する。

2．ファームウェアの構成

CHIでは、より高速な処理を行なうため7フィールドからなる1語80ビット長の水平型マイクロプログラム方式を採用している。その結果、メモリアクセスとスタックポインタの更新、データの比較処理などを並列に処理可能となっている。ファームウェアの構成を図1に示す。これらの規模は約10K語であり、その内訳はユニフィケーション用基本命令2K語、組込命令6K語、その他拡張処理部2K語となっている。

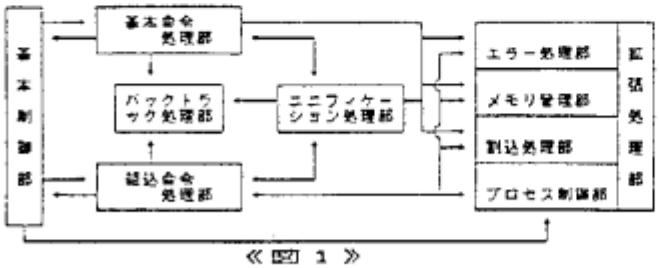
・基本制御部

機械語の逐次実行の制御と入出力などの割り込みの検出処理を行なう。これらの制御は、ハードウェアとの連係により実現している。

・基本命令処理部

述語の引数単位でのユニフィケーションや述語呼び出し等の基本的な実行の制御を行なう命令列を処理するモジュールである。CHIではコンパイラによる最適化を最大に生かすように基本命令を設定しているが、さらに処理速度を向上させるためにマイクロプログラム制御での命令先取りを行なっている。

・ユニフィケーション処理部



《図 1》

ユニフィケーションを行なうモジュールで，呼び出し側引数と呼び出された側の引数の2つのデータタイプによる多方向分岐機能により基本命令処理部と組込命令処理部から呼ばれる。さらにリストなどの入れ子構造をした構造体のユニフィケーション処理は、本モジュールを再帰的に呼び出すことによって行なう。

・バックトラック処理部

ユニフィケーション処理に失敗した時に、実行環境をその述語呼び出し以前の状態に戻す処理を行なう。

・組込命令処理部

CHIでは、システムプログラミングを可能にするためのデータ操作命令をはじめとして以下に示すような約110個の組込命令を用意している。

- ・拡張制御命令（level_cut , catch&throw bindhook , signal,on_backtrack）
- ・データオブジェクト生成命令
- ・属性検査命令　・構造体データ操作命令
- ・算術論理演算命令　・ロカティブ操作命令
- ・入出力命令　・プロセス管理命令

・拡張処理部

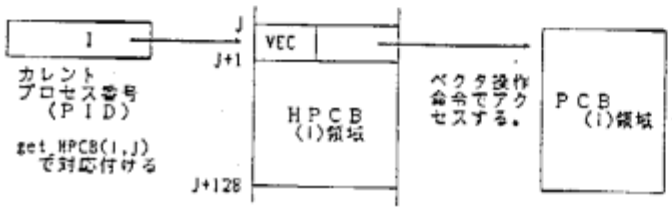
エラー処理や、組込命令からの拡張処理ハンドラの起動、バインドフック及びシグナルハンドラ（後述）の起動、入出力割込処理、メモリ管理、プロセス制御、トレース実行サポートなどの各種割込処理を行なう。

3．プロセス管理と割込処理

3．1　マルチプロセス管理

CHIでは、マルチプロセス機能[2]を導入し、そのためのハードウェア資源の管理などプロセス制御の多くの部分をファームウェアでサポートしている。各プロセスはローカル、グローバル、トレイルの各

スタックに独自のアドレス空間を持ち、その他のシステム領域、コード領域、データ領域、ヒープ領域をプロセス間で共有する。その結果実メモリ容量との関連からＣＨＩでは、最大３４０個までのプロセスを同時に取り扱うことが可能である。またこのためにＨＰＣＢ、ＰＣＢと呼ばれるプロセス制御用の情報をそれぞれシステム領域とヒープ領域にプロセス毎に持っている。（図２）　ＨＰＣＢは、ファームウェアが参照するプロセス切換え時のハードウェア情報や各プロセスの例外処理プロセス番号、プロセス間インターフェイス情報などを保持する１２８語固定長の領域で、ＰＣＢは、ソフトウェアが参照、操作するプロセス情報を格納する可変長の領域でヒープ領域にベクタとして確保される。



《図２》

３．２　マルチプロセスサポート機能

ファームウェアでサポートしているプロセス管理機能として以下のものがある。

・　プロセス切換え

指定されたプロセス番号のプロセスに実行環境を切換える機能で、レジスタのスワップ（２５個）やアドレス空間の切換え、キャッシュメモリの選択的クリア処理を行なう。

・　他プロセスのデータオブジェクト操作機能

指定した他のプロセスのスタック情報を読み書きすることができる。例えば、エラー処理プロセスがエラーを発生したプロセスの状態を調べる時などに必要な機能である。

・　非同期なプロセス間通信

ＣＨＩでは、非同期なプロセス間通信手段としてシグナルを用意している。各プロセスは、ＨＰＣＢ領域に３２個までのシグナルハンドラを定義し、その各々のエントリに対応したシグナルフラグを他プロセスが立てることにより非同期にそのシグナルハンドラを実行する。ＯＳではシグナルハンドラを定義するための述語を用意している。なお、シグナルフラグのチェックは述語呼び出し時に行なっている。

３．３　割込処理

ＣＨＩでは、各種の割込検出処理によって通常シーケンスの実行速度を犠牲にしないためその多くをハードウェア化している。割込検出のタイミングは以下のとおりである。

①機械語単位

・入出力割り込み

・トレース実行処理

②述語呼び出し単位

・メモリの動的確保

・バインドフックハンドラの起動

・シグナルハンドラの起動

③例外事項発生時

・エラー処理

・組込命令の拡張処理

また処理の方法としては、各割込処理プロセスに任せる方法と割込を発生したプロセスの内部で処理する方法があり、前者としては入出力、エラー、トレース、メモリ要求などで、後者には組込命令の拡張処理、バインドフックハンドラ、シグナルハンドラの処理が含まれる。

４．おわりに

現在ＣＨＩ上では、ＯＳをはじめとするシステムプログラムの開発を継続中であるが、そこから得られた経験をもとにファームウェアの機能をさらに拡充すると同時に処理速度をより向上させるために最適化を行なっていく予定である。

なお、本研究は第５世代コンピュータプロジェクトの一環として行なわれた。

参考文献

1．R.Nakazaki,et.al "Design of a High-speed Prolog Machine" '85 Int. Symp. on Computer Architecture

2．島津秀雄他 "高性能ＰＲＯＬＯＧマシン：ＨＰＭ－　オペレーティングシステム" 情報処理学会第３０回全国大会論文集

高性能Ｐｒｏｌｏｇマシン ＣＨＩ における Ｐｒｏｌｏｇプログラムのコンパイル方式

2-5

小長谷明彦＊　阪野正子＊＊　梅村　護＊

＊　日本電気株式会社　　＊＊　関西日本電気ソフトウェア（株）

1．はじめに

近年、知識処理用言語としてPrologが知識表現の容易さ、自然さ、明白さの観点から注目を集めているが、より実用的な問題を解くために処理性能の向上、メモリ使用効率の向上が大きな課題となっている。性能を向上させる方式としては処理系の一部をファームウェア、ハードウェア化する方式と共に、重要かつ有効な手法としてコンパイラによる最適化がある。実際Ｐｒｏｌｏｇマシン ＣＨＩの処理性能（appendで280KLIPS[1]）の実現にあたってはハードウェア性能とともにコンパイラによる最適化の効果が大きな役割を果たしている。

本稿ではＣＨＩで採用したWarrenのコンパイル方式の中心となるインデクシングならびに終端再帰呼び出しの最適化（ＴＲＯ）において、原論文[2]では明らかにされていないカットオペレータを含む節のインデクシングとＴＲＯのための変数分類について、実現上の問題点とＣＨＩにおける解決法を述べる。

2．インデクシング

インデクシングは入力引数と節との対応関係をコンパイル時に解析し、冗長なバックトラックを除去する技法である。例えば、ａ，ｂ，ｃをそれぞれ引数とする３つの節は入力引数がａ，ｂ，ｃのときはそれぞれ第１節、第２節、第３節のみを実行し、入力引数が変数のときは第１節から逐次的に実行するようにし、それ以外のときは直ちに失敗するようにコンパイルすることにより、冗長な選択点の生成とバックトラックを避けることができる。Warren方式の特徴の一つはこのインデクシングを引数タイプについても適用した点にある。例えば、以下のappendプログラム

```
append([],X,X).
append([X|Y],Z,[X|YZ]):-append(Y,Z,YZ).
```

は、第１入力引数を変数、リスト、構造体および定数（[]を含む）の４種に分類し、変数なら両方の節を実行し、リストなら第２節のみを実行し、構造体ならば述語呼び出しを直ちに失敗させ、定数なら第１節のみを実行するようにコンパイルされる。

このように、インデクシングを用いると入力引数が変数でないときは述語呼び出しを関数呼び出しと同程度の手間で実現でき、さらに、入力引数が変数のときは非決定的に動作するようにコンパイルすることが可能である。

次に、カットオペレータを含む節のインデクシングを考える。例えば図１のプログラムの場合、第２節のカットオペレータは述語ｐの選択点をカットするために使用されるが、入力引数がｂのときはインデクシングにより実行可能な節が一つしか選ばれない。ところが、Warren方式では述語が一つの節から構成されているときは選択点が生成されない。このことは入力引数が変数のときとｂのときとでカットオペレータの働きを変えねばならないことを示している。この問題の解決法としては以下の３つが考えられる。①制御スタック（ローカルスタック）上に手続き呼び出し毎にエントリ・ポイントを生成し、エントリ・ポイントまでの選択点をカットする。②第２節のコンパイルコードを２種類用意し、選択点を生成する場合としない場合で使い分ける。③述語呼び出しのレベルでカットの範囲を指定するレベルカットを用いる。ＣＨＩでは①は再帰呼び出し時のメモリ消費量が増加すること、②はコンパイラのオーバーヘッドが無視できないことから③の方式を採用している。

```
p（a）：－q（Y，Y），r．
p（b）：－！．
p（c）：－r．
```

図１　カットを含むプログラム例

図１のプログラムのＣＨＩにおける展開例を図２に示す。switch_on_term(A0)命令は第１入力引数の引数タイプに応じて４way に分岐する命令を表わす。第１入力引数が変数のときはtry_me_else(1,$3),retry_me_else($5),trust_me_else_fail命令により第１節、第２節、第３節をバックトラックしながら逐次的に実行する。また、第１入力引数がアトムのときはswitch_on_atomic(A0,4)命令によりさらに４way に分岐し、第１入力引数がａ，ｂ，ｃのときはそれぞれ第１節、第２節、第３節のみを実行する。第２節のコンパイル結果に現われるcut_relative(0)命令はその命令の述語呼び出しレベルからレジスタA0に与えられたレベル（この場合は０）だけ上位にある親ゴールの述語呼び出しより後に生成された選択点をカットする命令を表わす。

レベルカットは通常の手続きカットに比べると実現のオーバーヘッドは大きいが、

```
p：－q，（r，！；s）．
```

のようにＯＲ内にカットオペレータがあってもＯＲゴールを独立した別手続きに展開できるという利点を持つ。

3．終端再帰呼び出しの最適化（ＴＲＯ）

ＬＩＳＰ等の関数型言語では最後に自分自身を呼び出しているような再帰呼び出しはループ構造に展開でき、処理性能とメモリ使用効率を向上できることが知られている。Ｐｒｏｌｏｇでも同様な最適化が可能である。ただし、Ｐｒｏｌｏｇの場合必ずしも最後に自分自身を呼び出している必要はなく、述語が呼ばれてからその述語の最後のゴールを実行するまでに選択点が一つも生成されていなければＴＲＯが適用され、その述語が使用していた制御情報や変数の領域を解放することができる。これによりユニット節やappendの第２節のように本体部に一つのゴールしか持たない節は、選択点を生成しないかぎり制御スタックを伸ばさずに実行させることが可能である。ただし、ＰｒｏｌｏｇでＴＲＯを採用したときには、‘解放する変数領域に値領域を持つ変数の大域化’を行なう必要が生じる。例えば、

ｐ：－ｑ（Ｘ），ｒ（Ｘ，Ｙ）．

ではＹは解放する変数領域に値領域を持つことはできない。また、Ｘもｑ（Ｘ）の呼び出しで具象化されないかぎり解放したい変数領域に値領域が残り、値領域を解放すると‘宙吊りポインタ’ができてしまう。

Warren方式ではさらにＴＲＯを一般化し、

```
p:-q(X,Y,Z),r(Y,Z),s(Z).
```

におけるＹのように本体部の残りのゴールで使用されない変数についても選択点が生成されていないときは変数領域の解放を行なっており、この場合にも大域化の必要が生ずる。

Warren方式ではこのためにグローバル・スタック上に値領域を持つ変数（本体部の一時変数）と、最後に出現したゴールで値が具象化されていないときはグローバルスタック上に値領域を取る変数（非安全変数）を用意している。問題は変数をどのカテゴリに分類するかにある。例えば、Warrenの分類では

ｐ（ａ）：－ｑ（Ｙ，Ｙ），ｒ．

のＹは非安全変数に分類されるが、この場合は最初のＹの出現に対応する命令が解放したい変数領域へのポインタを生成してしまうため次のＹを大域化するための命令（put_unsafe_value）に展開しても宙吊りポインタができてしまう。ＣＨＩでは以下に示す分類基準を採用し、Ｙを一時変数に分類することにより、この問題を解決している。

一時変数：　１ゴールにしか現われない変数（ただし、ヘッドゴールは本体部の第一ゴールに含める）。

非安全変数：最初の出現がヘッドゴール、最終ゴール、構造体の中でない変数。

安全変数：　それ以外の変数。

上記例の場合、Ｙを安全変数に分類し、変数領域の解放を次のゴールで行なう方法も考えられるが、その場合はゴールｑ（Ｙ，Ｙ）の中で参照ポインタがＹの値領域を指さないことを保証する必要がある。

4．　おわりに

Warren方式のコンパイル技法の中心となるインデクシングとＴＲＯについて、実現上の問題点を明らかにし、ＣＨＩでの実現法を述べた。今後は本稿で述べたコンパイル方式について、実行速度、メモリ使用容量、コンパイル時間等の観点から定量的な評価を行なってゆく予定である。

本研究は、第五世代コンピュータプロジェクトの一環として行なわれた。本研究の機会を与えて下さった新世代コンピュータ技術開発機構の方々に深謝いたします。

参考文献

[1] Nakazaki et al. :"Design of a High-Speed Prolog Machine"Intnl.Symp. on Comp. Archi. 1985

[2] Warren :"An Abstract Prolog Instruction Set" TR309, AI Center, SRI International, 1983

```
p/1:  switch_on_term(A0)
      jump($1)          %変数
      fail(0)           %リスト
      fail(0)           %構造体
      jump($7)          %定数
$1:   try_me_else(1,$3)
$2:   allocate              %  p（
      get_constant(A0,a)    %  a）：－ｑ（
      put_temporary_variable(A0,A0) %  Y
      put_temporary_value(A1,A0)  %  , Y）
      call(0,q/2)                 %  ,
      deallocate
      execute(r/1)                %  r．
$3:   retry_me_else($5)
$4:   get_constant(A0,b)      %ｐ（ｂ）：－
      put_short_integer(A0,0) %
      cut_relative(0)         %  !
      proceed                 %  .
$5:   trust_me_else_fail
$6:   get_constant(A0,c)      %ｐ（ｃ）
      proceed                 %  .
$7:   switch_on_atomic(A0,4)
      fail(0)   %  other types
      jump($2)  %  in case of a
      jump($4)  %  in case of b
      jump($6)  %  in case of c
      fail(0)   %  other atoms
```

図２　ＣＨＩにおけるコンパイル例

高性能ＰＲＯＬＯＧマシン（ＣＨＩ）の マシン・デバッグ支援環境

幅田 伸一[*]，中崎 良成[*]，島津 秀雄[*]，貝谷 和人[**]
（[*]日本電気株式会社）（[**]株式会社エヌジェーケー）

はじめに　逐次型推論マシン”ＣＨＩ”［1］は、Ｐｒｏｌｏｇで記述したプログラムを高速実行するＰｒｏｌｏｇ専用マシンである。短期間での開発を実現するために、ハードウェア（以下、Ｈｗと略す）とファームウェア（以下、Ｆｗと略す）の同時開発、実機とシミュレータの実行結果比較による自動ベリファイなどが重要となる［2］。本稿では、ＣＨＩ開発期間短縮に貢献した開発支援システムの構成及び、特徴について報告する。

目　的　ＣＨＩ開発支援システムは、Ｈｗ開発者、Ｆｗ開発者によるマイクロ命令の動作確認、マクロ命令の動作確認、更に、テスト用ＰＲＯＬＯＧプログラムの動作確認を行なう際の実機制御、シミュレータ操作を行なう。実機の制御機能としては、Ｈｗ資源アクセスとクロック制御機能がある。ＣＨＩ開発にはＨｗ開発者、Ｆｗ開発者など多くの人間が係わり、ＣＨＩ開発支援システムの利用者毎に作業内容が異なる。Ｈｗ開発者は、

・Ｈｗ資源の動作確認
・マイクロ命令の動作確認
・マイクロ命令列の動作確認

を行なう。一方、Ｆｗ開発者は、

・マクロ命令の動作確認
・マクロ命令列の動作確認
・テスト用ＰＲＯＬＯＧプログラムの動作確認

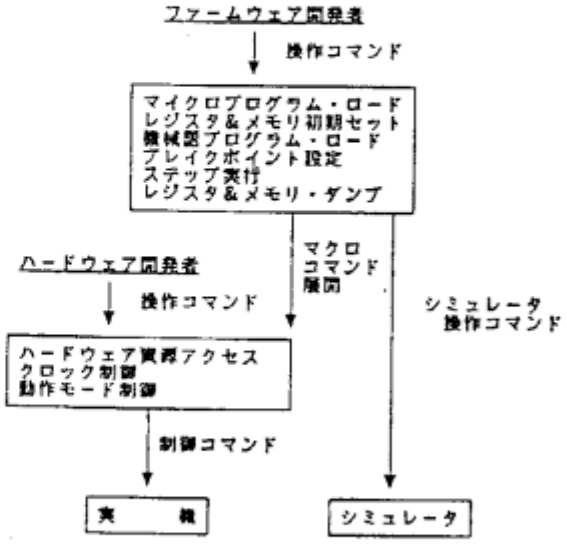
を行なう。このため、ＣＨＩ開発支援システムの利用者毎に操作コマンドの機能を設定する必要がある。

ＣＨＩ開発支援システムは、特にＨｗ開発用操作コマンドとＦｗ開発用操作コマンドの機能レベルの差を埋め、利用者に応じた操作コマンド・インタフェースを提供し、各デバッグ・フェーズの時間短縮と共に、Ｈｗデバッグ・フェーズからＦｗデバッグ・フェーズへの移行をスムーズにする。

機　能　開発期間短縮のために、以下の機能を備える。

1）レジスタ転送レベルのシミュレータによるマイクロプログラム開発
2）実機制御用デバッガで試験プログラム実行によるＨｗ基本動作確認
3）ベリファイアによるシミュレータと実機制御用デバッガのステップ動作結果比較
4）Ｆｗ開発者にシミュレータと同レベルの実機操作コマンド・インタフェースを提供

Ｈｗの開発と並行してマイクロプログラム開発を可能にするものが、（1）の機能である。Ｈｗ基本機能の動作確認における異常箇所検出の自動化が（2）の機能である。（3）の機能は、実機にマイクロプログラムを搭載後の動作確認で、異常箇所検出を自動化するベリファイ機能である。（4）は、Ｆｗ開発者に実機とシミュレータの操作を同レベルのコマンドで行なう環境を提供して、実機上でのマイクロプログラム・デバッグを容易にする機能である。



第1図　ユーザインタフェース

構　成　　逐次型推論マシン"ＣＨＩ"は、ＶＡＸ１１／７５０を入出力装置制御、コンソールプロセッサとして使用している。ＣＨＩ開発支援システムは、ＶＡＸ１１／７５０上に構築し、Ｈｗ開発者とＦｗ開発者のデバッグ作業における実機操作を支援する。ＣＨＩ開発支援システムの構成は以下のとうりである。

１）シミュレータ
２）デバッガ
３）マイクロベリファイア

シミュレータ　　Ｆｗ開発者が、実機を使用せずにマイクロプログラム開発を行なうためのレジスタ・トランスファ・レベルのシミュレータである。マイクロアセンブラ＆リンカにより生成したオブジェクトを実行する。ステップ実行、ブレイクポイント設定、レジスタ＆メモリ・ダンプが可能である。

レジスタ＆メモリの初期セット、繰り返し操作を自動化するため、コマンドファイル機能を備える。

デバッガ　　実機制御、試験プログラム実行機能を提供する。２つの実機デバッグ手段がある。１つは、インタラクティブ・モードで、端末から実機制御コマンドを入力しながら基本機能の動作確認を行なう手段である。もう１つは、検査自動化の一環として、試験プログラムを実行することにより複数の基本機能の動作確認をまとめて行なうオートベリファイ・モードである。

オートベリファイ・モードでは、テストデータを扱うテストデータ・ファイル、試験プログラムを扱う試験プログラム・ファイル、実行後の期待値データを扱うアイディアルデータ・ファイルの３種類のファイルを使用する。試験プログラムは、コマンドファイルを拡張したもので、インタラクティブ・モードで使用可能な実機制御コマンド、実行制御命令(FOR　NEXT)、条件分岐命令(JUMP ON NOT EQUAL)が使用できる。

ベリファイア　　Ｈｗ基本機能の動作試験が終了した実機は、Ｆｗ開発者が受け取り、実機上でマイクロプログラムのデバッグを行なう。ベリファイアは、Ｆｗ開発者に、シミュレータと同じ操作コマンドを使用した実機上でのデバッグ作業を可能とする。さらに、実機とシミュレータの実行結果を比較することにより異常箇所の検出を自動化する。

第１のシミュレータと同じ実機操作インタフェースを実現するために、ベリファイアにデバッガをアクセスするためのマクロコマンドを導入する。さらに、Ｈｗ開発者がデバッガを使用してＨｗ資源をアクセスする時に使用する名称とＦｗ開発者がシミュレータを使用してＨｗ資源をアクセスする時に使用する名称の対応を管理するテーブルを導入する。

第２の実機とシミュレータの実行結果比較を実現するために、シミュレータのデータに対応した実機のデータをアクセスする手順を管理するテーブルを導入する。さらに、データフォーマットの違いを変換するフォーマット変換テーブルを使用する。

ベリファイアは、Ｆｗ開発者がシミュレータによるマイクロプログラム開発時に端末から入力した操作コマンド列を試験プログラムとして使用する。この試験プログラムは、実機とシミュレータの初期セット、マイクロプログラムの実行制御、各ステップ実行後の実行結果比較を行なう。

まとめ　　シミュレータによるマイクロプログラム開発、デバッガが提供する試験プログラムによるＨｗ基本機能の動作確認、ベリファイアによるシミュレータと実機の実行結果比較により、デバッグ作業を効率化できた。

又、実機操作インタフェースを階層化した結果、Ｈｗ開発者からＦｗ開発者へのデバッグ作業の引き継ぎが迅速に行なえ、開発期間短縮に貢献している。

謝　辞　　本研究は、第５世代コンピュータプロジェクトの一環として行なわれた。本研究の機会を与えて下さった、新世代コンピュータ技術開発機構の方々に深謝いたします。

参考文献

[1] Nakazaki et al.:"Design of a High-Speed Prolog Machine"　Computer Architecture,1985

[2] G.Staas:"TDL:A Hardware/Microcode Test Language Interpreter"　Microprogramming Workshop,1984

並列推論マシンPIM-R の 詳細ソフトウェア・シミュレーション —Concurrent Prolog 解析—

松本 明　清水 聰　尾内 理紀夫
（財団法人 新世代コンピュータ技術開発機構）

1. はじめに

PIM-R(Reduction based Parallel Inference Machine)は、リダクション概念に基づく並列推論マシンである。並列型マシンでは、言語が重要な要素となるが、PIM-Rでは、 当機構が核言語第一版 KL1(84)のベース言語として位置付けてきたOR並列型のPrologと、 AND並列型のConcurrent Prolog 、及び KL1(85)のベース言語であるGuarded Horn Clauses(GHC) を対象言語としている。我々は、既に PIM-Rの基本アーキテクチャを確認するため、DEC 2060/VAX-11 上にDEC-10 Prolog/C-Prologで記述したソフトウェア・シミュレータを開発し、各種データを収集してきた[Onai 85] 。そして、現在新たに、VAX-11上に、シミュレーション速度と規模の向上、及び詳細構造のシミュレーションを主目的として並列型プログラミング言語Occam を用いた詳細ソフトウェア・シミュレータを作成している。 本稿では、PIM-R でのConcurrent Prologの並列実行方式、AND 並列型Concurrent Prolog用詳細シミュレータの構成、シミュレーション結果について報告する。

2. Concurrent Prolog の並列実行方式

PIM-R では、Concurrent Prolog の AND関係にあるゴールは、別の要素プロセッサ(Inference Module:IM)(図1) にフォークして並列に実行するが、OR関係にあるゴールは同一IM内でパイプライン的に実行する。次に例を示す。

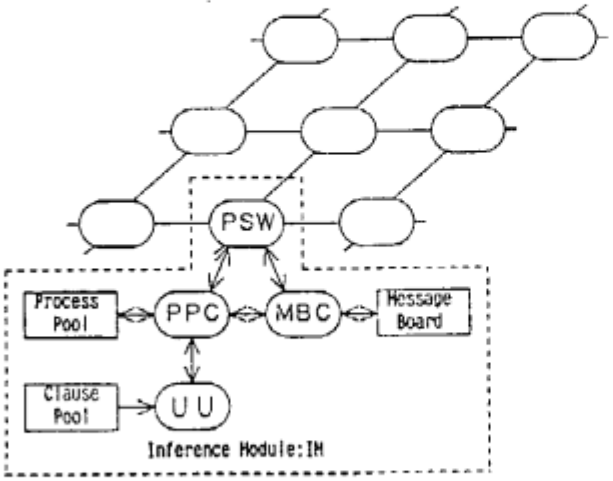


図1． PIM-R詳細ソフトウェア・シミュレータの構成

（例） ゴール　　p1.
クローズ群　p1:-gi | bi. (1≦ i≦m)

この例で、p1のユニフィケーションを行うと m個のOR関係にある子プロセス（兄弟プロセス）が生成されるが、これらのプロセスは、同一IM内でパイプライン的に処理を行い、別IMにフォークすることはしない方式を取っている。これは、ガード部(gi)の実行に成功したプロセスは、コミット・オペレータ("|")を越える時に、既に他の兄弟プロセスが，ガード部の実行に成功しているか否かを調べる必要があるが、このために、ネットワーク転送が発生するのは得策ではないと判断したからである。

3. シミュレータの構成

Concurrent Prolog 用詳細シミュレータは、図1に示すように多数のIMと、それらを2次元メッシュ状に接続するネットワークから構成され、 現時点でStructure Memory Module は、導入していない。 PIM-Rでは、1つのIM内の要素内並列を考えているため、UU,PPC,MBC、及び PSWの4個のサブ・ユニットは、それぞれ並列動作可能である。これらは Occamの PARコンストラクトを用いて実現してあり、各ユニットは、以下に示す機能をそれぞれ分担する。

① Unification Unit (UU)

クローズを格納するClause Pool (CP)を用い、クローズの第一引数による絞り込み(Matcher部) と、実際のユニフィケーションと組込み述語の実行(Unifier部) 等を行う。

② Process Pool Controller (PPC)

プロセスを格納するProcess Pool (PP) を用い、プロセス生成処理、プロセス更新処理、プロセス消滅処理等を行う。

③ Message Board Controller (MBC)

並列AND 関係にあるゴールが通信に使用するチャネル変数、及びサスペンドしたプロセス・リストを格納するMessage Board (MB)を用いて、MBへの書き込み、読み出し、サスペンドしているプロセスのアクティベイト要求等の制御を行う。

④ Packet Switch (PSW)

ネットワークとIMの接合点であり、IM間を渡るデータ・パケットの流れを制御する。

各ユニット間のデータ転送は、全て Occamチャネルを介したパケット転送をしている。また、デッドロックを回避するため、適宜ユニット間にキューを設けている。なお、本詳細シミュレータは、PIM-R の内部データ形式、及び転送されるパケットの形式を正確にシミュレートしている。

4. 詳細シミュレーション

4.1. シミュレーション条件

① IHの台数は、N^2、但し、N＝1，2，3，4 とし、IH間のネットワークは、N^2のメッシュとする。

② PPC は、新しく生成されたプロセスを、レディー・プロセス・キューの最後尾につなぎ、キューの先頭から順にUUへ転送する（疑似breadth first)。

③ AND フォーク時の他IHへのゴール分配法則は、IHが２台の時は、自分→隣→自分…、IHが４台以上の時は、自分→東→南→自分…、の繰返しとする。

4.2. シミュレーション結果

① Message Board(MB)関連パケット

IH間を渡るネットワーク転送には、他IHへの ANDフォーク要求、子プロセス成功・失敗パケットの他に、Concurrent Prolog のチャネル変数が自IHのIH内にない場合の書込み、読出し、及び、サスペンドしているプロセスのアクティベイト要求等のMB関連パケットがある。表１に、Quick Sort (50要素) をシミュレートした時のパケット数、転送語長とIH台数の関係を示す。これによれば、MB関連転送パケットは、１パケット当りのデータ語長は短いが、IH間全パケットの内に占める割合が極めて高いことが解る。

② Clause Pool共有化効果

PIM-R は、処理の独立性を高くし並列度を上げるため、リダクション要求、及びリダクション成功パケット等は、リテラル部をコピーする方式を採用していた。しかし、UUからしかアクセスできないClause Pool を PPC からもアクセスできるように変更すれば、クローズのテンプレートを共有できるため、プロセス・テンプレート・ブロック(PTB) 語長をQuick Sortの例では、約29% 程短縮でき、PPC におけるプロセス生成、更新、UUでのユニフィケーションに伴うコピー量も減少することができることが判明した。

表１．IH台数によるパケット数と転送語長

IH台数	1	2	4	9	16
自IH内パケット	8106個 166258語	6773 161267	6127 158849	5694 157162	5683 157123
IH間パケット	0 0	1535 11110	2247 15709	2674 17261	2681 17106
IH間パケットの内MB関連パケット	0	1333	1979	2412	2423
	0	4991	7409	9096	9135

③ Concurrent Prologと GHC化効果

Concurrent Prolog と GHCは、共に AND並列を基本にしているが、 GHCはConcurrent Prolog から多環境と読出し専用注釈(Read Only Annotation)を取り除いた形となっている。つまり、 GHCでは、チャネルのためのローカル環境を各PTB が持つ必要はない。従って、Concurrent Prolog による50要素のQuick Sortでは、

平均 PTB語長 ……	37.7語
平均チャネル情報（MBへのポインタとローカル環境）語長 ……	8.3語

であるが、 GHCの場合には、ローカル環境が不要なので、その分、約22% PTB 語長を短くできる。しかし、GHC では、述語を呼び出した時に、各クローズのガード部を実行している間は、親から観測できるバインディングを生成できないので、そのバインディング（ユニフィケーション）は、ボディー側へ移す必要がある。例えば、第二引数から出力するQuick Sortの qsort述語の場合、Concurrent Prolog では、

```
qsort([],[]):-true | true.
```

であるが、GHC では、

```
qsort([],X):-true | X=[].
```

のようになる。

つまり、この例では新たな変数X のために変数エリアが１語、 X=[]の内部表現用にリテラル・エリアが４語増加する。 このように、GHC では Concurrent Prologに比べ、 変数エリアとリテラル・エリアが増加するので、ローカル環境を持たなくても良いというGHC の長所を生かすためにも、Clause Pool をPPC からもアクセス可能なように共有化する必要がある。

5. おわりに

並列推論マシンPIM-R 上での Concurrent Prologの並列実行方式、詳細ソフトウェア・シミュレータの構成とシミュレーション結果について述べた。今後は、シミュレーション結果をもとに、 Clause Pool共有化をはじめとする PIM-Rの改良を行っていく予定である。最後に、御討論いただいた（株）日立製作所　杉江衛、米山貴、岩崎正明の各氏，御指導，御討論いただいた内田俊一第４研究室長に謝辞する。

〈参考文献〉

[Onai 85] 尾内、沼水、麻生、益田、松本、“リダクション方式並列推論マシン PIM-Rのアーキテクチャ”, Logic Programming Conference '85, Tokyo, 1985.7.

90－1

計算機研究者用ネットワーク-JUNET-

村井純(東工大) 松方純(東大) 田中啓介(上智大) 長谷部紀元(図書館情報大) 徳田英幸(CMU)
桜井清蔵(慶応大) 浅見徹(KDD) 奥乃博(NTT) 高木茂行(ICOT)

1．はじめに

各種計算機技術の発達にともない、計算機が提供する研究環境の有効性は向上している。そのため、計算機ネットワークは、研究者間の研究情報の交換によって研究環境を向上させる有効な手段となりえる。計算機の研究開発に携わる研究者間の情報交換を実現するための研究開発用ネットワークは、米国国防総省のARPAネットワーク[Hinden83]を中心として、NSFのCSNet[Denning83]、IBM社のメインフレームの結合からはじまったBITNET、UNIX利用者のためのUSENET[Horton83]などいくつかのものが実在する。これらは、いずれも大学、公的研究機関、民間研究機関、民間開発機関を結合し、電子郵便、ファイル転送などの応用プログラムを利用することにより、研究開発上の情報交換を可能とした研究者の研究環境を実現するために用いられている。

JUNETは、我が国における研究開発ネットワークを実現するための課題を、実際的に実験する基盤となることを目的としたネットワークである。JUNETの実験は1984年10月から開始されており、電子メイル、電子ニュースによる情報交換を行ないながら、研究開発ネットワークの運用管理、名前管理、アドレス変換、ゲートウェイ機能、分散データベース、国際間ネットワーク結合などに関する研究開発と実験が行なわれている。

2．JUNETの目的

国内の大学や研究開発機関が内外の各種大学、研究機関と研究情報を交換することにより研究環境の質的向上をはかる目的のネットワークを構築するためには、実際に多くの研究者が利用することができ、このようなネットワークにともなう多くの課題に関する研究、開発、実験の基盤となり得る実験ネットワークが必要である。JUNETは、このような研究、実験ネットワークとしての役割を目的としたネットワークである。

3．JUNETの結合

JUNETは、実験参加サイトの計算システム、または、ローカルエリアネットワークのゲートウェイシステムを計算機用電話回線を用いて結合することにより実現している。したがって、2サイト間の物理的結合は、自動発信機能付きモデムと自動受信機能付きモデムが1組必要となる。発信側のサイトは、一定間隔、または、要求の発生時に受信サイトを呼出し、定められた上位レベルのプロトコルを用いて通信を行なう。

現在JUNETでは、電話回線を利用したシリアル回線上のプロトコルとしてベル研究所のUUCP[Nowitz78]を利用している。UUCPは、UNIX†間の通信機能で、ファイル転送と遠隔コマンド実行の機能を提供し、1回の結合の際にスプールされている相互の通信要求が解決される（図1）。

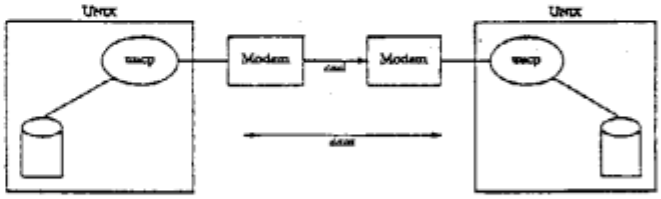


図1．JUNETの基本結合

4．JUNETの運用

JUNETの運用は、各参加サイトの管理者が、そのサイトの利用者を登録することにより行なわれている。登録されている利用者から発生された通信要求は、上記の通信手順に従い、隣接サイトに引き渡される。引き渡されたサイトでは、あらかじめ定められた利用権利に従い、その要求を処理する。必要であれば、その要求は更に次の隣接サイトに引き渡されることもある。

JUNET全体の運用費用は、隣接サイトとの通信のために発信側が負担する電話料金のみで、この他の費用は一切無料である。

5．JUNETの基礎機能

現在、JUNETで運用されている基礎機能は電子メイル[Allman82]と電子ニュース[Horton84]の2種類である。電子メイルは、利用者間のメッセージ交換を提供する機能で、電子ニュースは、利用者が随時参照することのできる電子掲示板の機能である。

これらの機能は、いずれも上記のUUCPプロトコルのもつファイル転送と遠隔コマンド実行の機能を組み合わせることによって実現されている。

6．JUNETにおける実験

(1)ネットワークの運用管理

研究者間の情報交換を円滑に、効率良く実現するためには、実験的運用による記録を検討し、今後の運用に活用することが必要となる。また、現在では管理者間の直接的な情報交換で行なわれている管理情報の交換を自動的に処理する方向で検討することも必要である。現在収集、検討している管理情報には、次のようなものがある。

・通信量と通信時期

シリアル回線を用いた通信では、通信時に回線を占有する。したがって、通信量が増加すると回線のとりあいが起こり、スケジュール管理の考慮が必要となる。

・通信時のホスト負荷

通信時には、通信そのものに関する負荷に加えて、通信の要求か

†UNIXは、米国におけるAT&Tベル研究所の商標である。

ら発生する処理の負荷が存在する。必要に応じて、このような処理を遅延して行なうことにより過負荷の状態を避けることができる。

・各基礎機能の利用状況

電子メイル、電子ニュースなどの基礎機能の利用状況では、電子メイルの利用者数に比べて、電子ニュースの利用者、特に、電子ニュースの発信者が少ないことがあげられる。この原因として、宣伝と演出の不足、システムの使いにくさ、日本語処理環境の貧弱さ、国民性などいくつか考えられている。技術的な面での改善をはかっていくなかで、検討が必要な項目である。

・スプールされる情報量

結合の状態と、通信量に依存して、スプールされる情報量が変化する。また、電子ニュースのシステムには、ネットワーク間結合を介して、全世界から記事が到着する。これらの情報を効率良く通信、格納、待避、廃棄するためには、現在用いられている圧縮の技術と、格納方法を検討する必要がある。

(2)日本語メッセージの取り扱い

いくつかの通信用ソフトウェアにおいては、コード上の制約のために日本語コードが使用できないものがある。JUNETでは、通信用の日本語コードとしてJIS6226のコードを利用することを定め、それを利用できるように各ソフトウェアを作成、改造している。

(3)名前管理

分散環境においては資源の名前管理が重要な問題となる。JUNETでは、Oppen[Oppen83]やCrocker[Crocker82]らによって提案されている階層的な名前の定義域、ドメイン、の概念に基づいた名前管理を実現している。特にJUNETのような結合方式では、物理的結合の状態は安定していなく、変化が激しい。このような環境では、物理的な結合や最適な経路を利用者から隠蔽することのできる名前管理の必要性が高い。

現在のJUNETの機能は、メッセージ交換が中心であるので、利用者の識別に関してこの名前管理を行ない、電子メイルなどの宛先の表現に効果をあげている。

(4)ネットワーク間結合

JUNETのような研究情報交換用のネットワークは、他の同様なネットワークとネットワーク間結合を行ない、情報の相互乗り入れを行なうことにより研究的価値は高まる。そのために、JUNETでは、USENETをはじめとするいくつかの国際ネットワークとのネットワーク間結合の実験を行なっている。

このためには、高価な国際通信を効率良く利用するための利用の制限、ネットワーク間で異なる、利用規則、アドレス表現、形式既定、コードなどの項目に関するチェックや変換を、ゲートウェイサイトを中心に処理できなくてはならない。JUNETでは、このような処理を実現するためのソフトウェア、データベースを接続されているネットワークの管理者との議論に基づき実現している。

7. 今後の課題

JUNETの実験は実際的な運用に基づいて行なわれているので、結合システムの増加と、それにともなう利用者の増加にしたがい、多くの課題が提供されてきている。まず、本実験のために設計開発された名前管理のシステムを含めて、動的な物理状況の変化に伴う通信システム全体の自動的な対応が必要である。このため、分散データベースの技術を応用した管理用データの自動更新、参照、検証の機能に関する設計開発が急務である。

また、JUNETが接続しているローカルエリアネットワークは、Ethernetによる単純な結合が殆どであるが、この他の環境に対応するためには、ゲートウェイ機能を一般的に設定できる方向での解決が必要である。

実際の利用者環境としては、研究情報の交換方法に対するより効果的な機能を考慮しなくてはならない。これは、現在利用している電子ニュースや、電子メイルの機能を改善すると同時に、新しい機能による研究情報の交換方法の開発によって実現する。研究者にとって一般的に必要である文献に関する情報の交換を効率的に実現するためのシステムに関する検討などが現在進められている。

8. まとめ

本論文執筆時現在、JUNETには、約20サイト、70システムが結合されている。ドメインの概念にもとづいた名前管理、アドレス制御、経路制御、ネットワーク間結合などに関する技術の開発により効率的な管理運用に成功している。また、国内外の研究機関との研究情報の交換を可能にする環境の提供により、研究面での効果も報告されている。各研究者が、各研究分野においてのJUNETの利用により多くの未解決な課題が生じてきている。このような問題を解決する方向での積極的な利用と、分散環境に関する実験基盤としての研究開発に使用することにより、JUNETは研究環境を向上するための有効な実験となると考えられる。

謝辞　本実験の開始、運用にあたっては、実験の中心となっている東京工業大学の池辺淑教授、木村泉教授、前野年紀助教授、米沢明憲助教授、米澤直樹助教授、東京大学の元岡達教授、和田英一教授、石田晴久教授、慶応義塾大学の斎藤信男助教授、所真理雄助教授、電総研の島田俊夫氏をはじめとする各先生方とJUNETの実験参加各サイトの管理者、利用者の方々から貴重な御協力、御助言、激励をいただいた。これらの方々に深く感謝する。

参考文献

[Kleinrock] R.Kleinrock, et al., "The DARPA Internet Interconnection: Heterogeneous Computer Networks with Gateways," *IEEE Computer*, vol. 16, no. 9, pp.38-48, Sep. 1983.

[Quarterman] J.Quarterman, et al., "History and Overview of CSNET," *SIGCOMM '82* pp.128-144, Mar. 1982.

[Horton] M.R.Horton, "Usenet: The Network News," *Appendix, USENIX, Vol.2, No.2*, pp.10-12, Aug. 1983.

[Nowitz] D.A.Nowitz and M.E.Lesk, "A Dial-up Network of UNIX systems," *User Programmer's Manual*, Bell Labs, Aug. 1978.

[Allman] E.Allman, "Sendmail - An Internetwork Mail Router DRAFT," *User Programmer's Manual*, CSRG U.C. Berkeley, Jul. 1983.

[Horton] M.R.Horton, "Standard for Interchange of of USENET Messages," *User Programmer's Manual*, CSRG U.C. Berkeley, Jul. 1983.

[Oppen] D.C.Oppen and Y.K.Dalal, "The Clearinghouse: A Decentralized Agent for Locating Named Objects in a Distributed Environment," *ACM TOIS*, vol.1, no.2, pp.230-253, Jul. 1983.

[Crocker] D.H.Crocker, "Standard for the Format of ARPA Internet Text Messages," *RFC 822*, Univ. of Delaware, Aug. 1982.