TR-236

PSIへのコンパイラ向きProlog命令の試験実装と評価

中島　浩、中島克人、瀧　和男

March, 1987

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

Abstract

An experimental implementation of a compiler target Prolog instruction set and an optimizing compiler have been made for the Prolog machine PSI, which has a microprogrammed interpreter originally. And the language execution mechanism have also been improved. The instruction set is based on that proposed by D.H.D. Warren and some extensions are introduced. Execution performance have been measured, that is, the execution speed was 2 to 3 times faster than that of the original interpreter. The maximum inference speed was 114.5k LIPS. Average execution speed of the Prolog instruction was approximately 0.6 MIPS.

「情報処理学会論文誌」第28巻 第10号別刷 昭和62年10月発行

PSI へのコンパイラ向き Prolog 命令の試験実装と評価

中島 浩 中島克人 瀧 和男

社団法人 情報処理学会

Vol. 28 No. 10　　情報処理学会論文誌　　Oct. 1987

ショートノート

PSI へのコンパイラ向き Prolog 命令の試験実装と評価†

中島　浩††　中島克人†††　瀧　和男†††

マイクロプログラムによるインタプリタ方式を採用している Prolog 専用マシン PSI に，コンパイラでの最適化を重視した機械語命令セットを試験実装し，合わせて処理方式についても改良を加え，性能評価を行った．機械語命令には，D.H.D. Warren の提案した Prolog 命令セット[9]を拡張して使用した．性能は PSI のオリジナル処理系に比べて 1.74 倍から 3.36 倍となり，最大推論速度 114.5 kLIPS を得ることができた．また機械語命令の平均実行速度は，約 0.6 MIPS となった．

1. まえがき

われわれは，Prolog を拡張した述語論理型言語の高速実行と，メモリの大容量化や対話型入出力装置の強化などの実用面での機能向上を目的に，パーソナル逐次型推論マシン PSI を日本の第5世代コンピュータプロジェクトの一環として開発し[3)~7)]評価を行ってきた[1),2)]．

PSI は命令実行方式として，プログラムのマシン内部表現をマイクロプログラムで記述されたインタプリタにより解釈実行する方式をとっており，その性能は DEC-10 Prolog のコンパイラ版[11)]を DEC-2060 上で実行した場合とほぼ同等の 30 kLIPS (Logical Inference Per Second：1秒間の推論回数) を実現している．

PSI では主に設計上の見通しの良さからマイクロプログラムによるインタプリタ方式を採用したが，DEC-10 Prolog のコンパイラ版との比較評価から，複雑なユニフィケーションやバックトラックなどの動的処理の大きいプログラムでは PSI の方が速いが，コンパイラによる最適化のかかりやすいプログラムでは DEC-10 Prolog の方が速くなること，また PSI では実行制御情報の管理に多くのステップ数を費やしていることが確かめられた[1)]．このことは，PSI においてもコンパイラでの最適化処理を重視した命令コード体系を導入することによりさらに速度向上の可能性があること，また高速化のためには実行制御情報管理のステップ数削減が有効であることを示している．

われわれは，これらのことを確認し PSI の改良に関する指針を得るという立場から，PSI の上にコンパイラでの最適化に適した機械語命令を試験実装し，処理方式についても改良を行い，いくつかの評価用プログラムを用いて性能評価を行った．本論文ではその結果について報告する．

2. 処理系の方式

Prolog プログラムはコンパイラにより Prolog 向きの機械語命令列へ変換される．機械語命令はマイクロプログラムにより実行されるが，命令コードを主にタグ部分に割り付けたため PSI のタグディスパッチ機能を用いて1ステップでマイクロプログラムの処理番地へ分岐させている．機械語には，D.H.D. Warren により提案された仮想マシンの命令セット[9),10)]に拡張を施したものを使用した．その特徴としては，

(1) プログラム中の変数，述語や構造体の名前，アトムなどの構成要素は，それぞれがほぼ1個の機械語命令に変換される．

(2) 述語呼出し時の引数をレジスタ渡しにするとともに，変数の一部（一時変数）をレジスタに割り付け，コンパイラで割付けの最適化をすることによりユニフィケーション系の命令の数を削減する．

(3) 前項の機能をテイルリカージョンの最適化[12)]に利用する．

(4) 構造体コピー法を用いる．

(5) 述語呼出しから復帰後の実行継続に必要な情報 (environment) とバックトラック時の再試行に必要な情報 (choicepoint) を分離してスタックに積む

† An Experimental Implementation of a Compiler Target Prolog Instruction Set for the PSI Machine and Its Evaluation by HIROSHI NAKASHIMA (Information Processing Department, Information Systems and Electronics Development Laboratory, Mitsubishi Electric Corporation), KATSUTO NAKAJIMA and KAZUO TAKI (Fourth Research Laboratory, Research Center, Institute for New Generation Computer Technology).

†† 三菱電機(株)情報電子研究所情報処理開発部
††† (財)新世代コンピュータ技術開発機構研究所第4研究室

ことにより environment の情報量を少なくし，それらを実行する命令を各々必要なときにだけコンパイラで生成する．

（6） カット機能の実装とPSIに特有の拡張実行制御機能[8]を包含する．

などがあげられる．これらのうち(1)〜(5)については文献[9]の思想を踏襲し，実用上必要な(6)を追加した上で，PSI上での試験実装を試みた．

またPSIハードウェアの活用と方式の見直しにより次のような高速化を図った．

（1） マイクロプログラム制御による命令の先読みを行い，命令フェッチのオーバヘッドを削減する．

（2） ローカルスタックとコントロールスタックを共通化し，スタックの本数を4本から3本に減らすことにより実行制御情報を削減する．

（3） スタック伸長時などに実メモリが割当て済みか否かをファームウェアでテストしていたものを省略し，ステップ数を短縮する．かわりに，未割当て領域へのアクセスにより発生するトラップを利用し，実メモリの割当てを行う．

これらの中でマイクロプログラム制御による命令の先読みが可能となったのは，PSIのハードウェア資源のうちメモリインタフェースのデータレジスタとアドレスレジスタが2組ありそのうち1組を命令レジスタ，プログラムカウンタとして固定的に使用できたこと，およびこれらのレジスタにタグディスパッチの機能と自動増加の機能が備わっていたこと[3),4)]による．命令の先読みはマイクロプログラムの各命令実行ルーチンの中で並行して行うため，命令フェッチと解析のための専用ルーチンというものは存在しない．

なお，割込みやガベージ・コレクションなど，本格的なシステムに必要な機能に対する配慮はすべて盛り込まれており，試験システムで測定された性能は，実用システムにおいても，オーバヘッドなしに実現可能である．

3. Prolog 向き機械語命令

機械語命令には，以下に示すように，文献[9]をもとに一部拡張を施したものを用意した．

（1） get 系命令：主にクローズヘッドの引数に対応する命令で，引数レジスタに与えられた引数とのユニフィケーションを行う．組込述語の出力引数部分のユニフィケーションにも使用した．

（2） put 系命令：ボディゴール中の引数に対応する命令で，引数をレジスタにロードする．

（3） unify 系命令：構造体あるいはリスト中の変数に対応し，既存の構造体要素とのユニフィケーションまたは新しい構造体要素の作成を行う．

```
get_variable_p Ai,Yn
get_variable_t Ai,Xn
get_builtin_variable Ai,Yn
get_value_p Ai,Yn
get_value_t Ai,Xn
get_atom Ai,C
get_integer Ai,C
get_constant Ai,C
get_nil Ai
get_vector Ai,Arity
get_list Ai

unify_void N
unify_variable_p Yn
unify_variable_t Xn
unify_local_variable
unify_value_p Yn
unify_value_t Xn
unify_local_value_p Yn
unify_local_value_t Xn
unify_atom C
unify_integer C
unify_constant C
unify_nil

try_me_else L,N
try L,N
retry_me_else L
retry L
trust_me_else Fail
trust L
switch_on_term Ai,Lv,Lc,Ll,Ls
switch_on_constant Ai,N,Table
switch_on_hash_constant Ai,Key,Table
switch_on_structure Ai,N,Table
switch_on_hash_structure Ai,Key,Table

put_variable_p Ai,Yn
put_variable_t Ai,Xn
put_value_p Ai,Yn
put_value_t Ai,Xn
put_local_value Ai,Yn
put_unsafe_value Ai,Yn
put_constant Ai,C
put_nil Ai
put_vector Ai,Arity
put_list Ai

cut_me
cut_allocated_me
cut_me_and_proceed
cut_normal
cut_and_proceed
cut_me_and_fail
cut_and_fail

allocate N
call Proc,N
execute Proc,N
execute_with_deallocate Proc,N
proceed
proceed_with_deallocate
```

図1 Prolog 向き機械語命令一覧
Fig. 1 Prolog instruction repertory.

(4) procedual 命令：述語の呼出し，復帰，environment の作成，除去の制御を行う．効率化のため機能を複合させた命令を追加した．

(5) indexing 命令：述語を作り上げている複数のクローズをつなぎ合わせ呼出し順を決める命令とクローズインデキシングを行う命令がある．choice point もこれらの命令の一部により作成される．クローズインデキシングを行う命令は，ハッシングによるもののほかに，順次探索によるものを追加した．

(6) cut 命令：効率向上のため複数の種類を用意した．各々の命令が使用される場合のクローズの例を図2に示す．

(7) built-in 命令：組込述語はオペランドにレジスタが指定できるよう作りなおし，評価用プログラムが必要とするもの約35種を用意した．

図1に組込述語を除く命令の一覧を示す．

```
cut_me :
   p :- !,q.     or  p :- bl,!,q.

cut_allocated_me :
   p :- !,q,r.   or  p :- bl,!,q,bl.

cut_me_and_proceed :
   p :- !.       or  p :- bl,!.

cut_normal :
   p :- q,!,r.   or  p :- q,!,bl.

cut_and_proceed :
   p :- q,!.

cut_me_and_fail :
   p :- !,fail.  or  p :- bl,!,fail.

cut_and_fail :
   p :- q,!,fail.

      ----- bl : built-in predicate
```

図2 カット命令と対応するクローズの例
Fig. 2 Usage of the cut instructions.

4. 最適化方式

本システムのコンパイラでは主に次の3種類の最適化を実施した．

(1) 永久変数と一時変数を判別して一時変数はレジスタに割り付け，永久変数が存在するときだけ environment を作る．

(2) 一時変数と引数用のレジスタ割付けを最適化し，get_variable, put_value の対を削除する．

(3) 第一引数によるクローズインデキシング処理を行い成功時には choice point 作成命令をスキップするようコード生成する．

試作したコンパイラでは，文献[8]に提案されているものと同等の出力コードを得ることができた．例えば，同文献にあり本稿でも評価に使用した quick sort では，deallocate と execute の二つの命令が，execute_with_deallocate という複合命令に置きかわっているほかは等価となっている．

5. 性能評価

評価用プログラムとして第1回 prolog コンテスト[13]の課題プログラムの一部と，8-puzzle と harmonizer という2つの応用プログラムを用いて実行時間を測定した．表1には本処理系のほかに比較のため2つの処理系の測定結果を示す．new compiler と記したのは本処理系であり，実行時間と LIPS (Logical Inference per Second：1秒間の推論回数) 値を示す．interpreter とは PSI 上に元から実装されている処理系であり，DEC-10 Prolog と記したのは同名の処理系のコンパイラ版を DEC-2060 上で実行した場合である．いずれも，実行時間ならびに本処理系との実行時間比を示す．

表中の(1)から(6)はコンテストの課題プログラムでいずれもリスト処理を中心とした単純なものであ

表1 評価用プログラムの実行時間ならびに他処理系との比較
Table 1 Execution time of bench-mark programs: new compiler compared with the original interpreter and DEC-10 Prolog.

Programs	New Compiler		Interpreter		DEC-10 Prolog	
	msec	kLIPS†	msec	ratio‡	msec	ratio‡
(1) nreverse (30)	4.35	*114.7*	14.6	3.36	9.5	2.18
(2) quick sort (50)	6.73	*89.5*	16.1	2.39	14.6	2.17
(3) tree traversing	28.95	*73.3*	52.2	1.80	61.1	2.11
(4) 8 queens (1)	48.10	*119.5*	105	2.18	97.5	2.02
(5) reverse function	22.31	*58.3*	38.9	1.74	41.7	1.86
(6) slow reverse (4)	2.58	*82.6*	6.1	2.38	5.4	2.10
(7) 8-puzzle	2442	—	6544	2.68	—	—
(8) harmonizer-1	276	—	657	2.34	1040	3.77
(9) harmonizer-2	784	—	1879	2.39	2670	3.41
(10) harmonizer-3	9846	—	24119	2.48	31390	3.19

†Including built-in predicate calls as logical inferences.
‡Exection time ratio (time of each system/time of new compiler).

る．8-puzzle はゲームのプログラムで組込述語の実行頻度が高い特徴を持ちプログラムサイズはソースプログラムで 100 行程度のもの，harmonizer はメロディーからハーモニーを合成する一種の音楽に関する専門家システムで引数個数が多いことと構造体データを多く扱う特徴を持ち，プログラムサイズはソースプログラムで 700 行程度である．

表から本処理系は，PSI 上に元から実装されているインタプリタに比べて 1.74 倍から 3.36 倍高速であり，推論速度は nreverse で 114.5 kLIPS を達成していることが分かる．インタプリタとの実行時間比を見ると(1)の nreverse のようなコンパイラでの最適化がかかりやすいプログラムにおいて，本処理系が特に高速になっていることが読み取れる．一方 DEC-10 Prolog との実行時間比を見ると，本システムは DEC-10 Prolog に比べて 1.86 倍から 3.77 倍の性能を持つことが分かる．また(1)から(4)および(6)のプログラムでは比の値がほぼ一定であることから，本処理系で採用した最適化方式と DEC-10 Prolog コンパイラでの最適化方式が近いものであることが推測される．(8)以下のプログラムで実行時間比が大きいのは，前述の harmonizer のプログラム特性が影響しているためと考えられる．

実行時間のほかに，機械語命令の実行回数，マイクロプログラムの全実行ステップ数を(7)，(8)のプログラムについて測定した．(7)では命令実行回数の合計は 1,614,249 回，マイクロ命令のステップ数は 12,210,000 ステップであり，機械語命令 1 個当たりの平均マイクロ命令ステップ数は 7.56 ステップ，命令実行速度は 0.66 MIPS (Million Instruction per Second) となった．(8)ではそれぞれ，163,975 回，1,380,000 ステップ，8.42 ステップ／命令，0.59 MIPS となった．

1 命令当たり平均マイクロステップ数が約 8 ステップであることから，命令の先読みの効果に関する荒い見積りができる．すなわち先読みをやめると 1 命令当たりの平均マイクロステップ数は約 9 ステップとなり 12.5% 程度の性能低下となることが予想される．

6. あとがき

PSI へのコンパイラ向き機械語命令の試験実装と性能評価を行った．性能は PSI のオリジナル処理系に比べて 1.74 倍から 3.36 倍，DEC-10 Prolog のコンパイラ版に比べて 1.86 倍から 3.77 倍となり，推論速度は nreverse プログラムで 114.5 kLIPS となった．性能の向上は，コンパイラによる最適な命令コードの生成と，命令の先読みや処理方式の見直しの両方によっている．また機械語命令 1 個当たりの平均マイクロ命令ステップ数は約 8 ステップ，機械語命令の実行速度は 0.6 MIPS 前後という値を得た．

今後はこれらの測定結果を PSI の改良版の設計に役立ててゆく予定である．

最後に研究の機会を与えてくださった(財)新世代コンピュータ技術開発機構研究所の渕　一博所長，内田俊一室長に感謝します．

参考文献

1) Taki, K., Nakajima, K., Nakashima, H. and Ikeda, M.: Performance and Architectural Evaluation of the PSI Machine, *Proc. of 2nd International Conference on Architectural Support for Programming Languages and Operating Systems* (1988), to appear.
2) Nakajima, K., Nakashima, H., Yokota, M., Taki, K., Uchida, S., Nishikawa, H., Yamamoto, A. and Mitsui, M.: Evaluation of PSI Microinterpreter, *Proc. of COMPCON Spring 86*, pp. 173-177 (1986).
3) 瀧　和男，横田　実，山本　明，西川　宏，内田俊一：パーソナル逐次型推論マシン PSI のハードウェア設計，*Proc. of the Logic Programming Conference '84*, No. 8-1, Tokyo (1984).
4) Taki, K., Yokota, M., Yamamoto, A., Nishikawa, H., Uchida, S., Nakashima, H. and Mitsuishi, A.: Hardware Design and Implementation of the Personal Sequential Inference Machine (PSI), *Proc. of the International Conference on Fifth Generation Computer Systems 1984*, pp. 398-409, Tokyo (1984).
5) Yokota, M., Yamamoto, A., Taki, K., Nishikawa, H. and Uchida, S.: The Design and Implementation of a Personal Sequential Inference Machine PSI, *New Generation Computing*, Vol. 1, No. 2, pp. 125-144, Ohmsha, Tokyo (1983).
6) Yokota, M., Yamamoto, A., Taki, K., Nishikawa, H., Uchida, S., Nakajima, K. and Mitsui, M.: A Microprogrammed Interpreter for the Personal Sequential Inference Machine, *Proc. of the International Conference on Fifth Generation Computer Systems 1984*, pp. 410-418, Tokyo (1984).
7) 山本　明，横田　実，西川　宏，瀧　和男，内田俊一：逐次型推論マシン ϕ のマイクロインタプリタ，情報処理学会研究会資料，記号処理 29-7 (1984).

8) 高木茂行，近山 隆，横田 実，服部 隆：拡張頭部構造の Prolog への導入，第 26 回情報処理学会全国大会論文集，No. 4 D-11 (1983).
9) Warren, D. H. D.: An Abstract Prolog Instruction Set, SRI Technical Note 309 (Oct. 1983).
10) Tick, E. and Warren, D. H. D.: Towards a Pipelined Prolog Processor, *Proc. of 1984 International Symposium on Logic Programming*, Atlantic City (Feb. 1984).
11) Bowen, D. L.: DEC System-10 Prolog User's Manual, Dept. of Artificial Intelligence, Univ. of Edinburgh (1983).
12) Warren, D. H. D.: An Improved Prolog Implementation which Optimizes Tail Recursion, *Proc. of the Logic Programming Workshop*, Hungary (July 1980).
13) Okuno, H.: The Report of the Third Lisp Contest and the First Prolog Contest, 情報処理学会研究会資料，記号処理 33-4 (1985).

(昭和 62 年 2 月 27 日受付)
(昭和 62 年 7 月 9 日採録)

中島 浩（正会員）

昭和 31 年 5 月生．昭和 54 年京都大学工学部情報工学科卒業．昭和 56 年同大学院修士課程修了．同年三菱電機(株)入社．同社情報電子研究所において，推論マシンのアーキテクチャの研究開発に従事．論理型言語の実行方式および並列プロセッサのアーキテクチャに興味を持つ．

中島 克人（正会員）

昭和 28 年生．昭和 52 年京都大学工学部電気第 2 工学科卒業．昭和 54 年同大学院修士課程修了．同年三菱電機(株)入社．以来，情報電子研究所にて汎用・専用計算機の開発に従事．昭和 57 年より第五世代コンピュータ・プロジェクトに参画し，逐次型推論マシンの開発・評価に従事．昭和 60 年より(財)新世代コンピュータ技術開発機構に出向．現在，逐次型推論マシンの改良，並列型推論マシンの研究・開発に従事．VLSI 向きプロセッサ，並列マシン・アーキテクチャ等に興味を持つ．

瀧 和男（正会員）

昭和 27 年生．昭和 51 年神戸大学工学部電子工学科卒業．昭和 54 年同大学院修士課程システム工学修了．工学博士．同年(株)日立製作所入社．同社大みか工場にて制御用計算機システムの設計に従事．昭和 57 年(財)新世代コンピュータ技術開発機構に出向．以来逐次型および並列型推論マシンの研究開発に従事．並列マシンのアーキテクチャ，並列プログラミングなどに興味を持つ．電子情報通信学会，IEEE 各会員．