並列推論マシンＰＩＭ

－ＫＬ１の抽象命令仕様とコンパイラ－

木村 康則　近山 隆　久門 耕一　中島 浩
ＩＣＯＴ　ＩＣＯＴ　富士通（株）　三菱電機（株）

１．はじめに

第５世代コンピュータの研究開発プロジェクトでは，並列推論マシンＰＩＭの研究を進めている．

ＰＩＭ上の並列言語ＫＬ１は，並列論理型言語ＧＨＣ［１］に基づいた言語であり，以下に示す３つの階層から構成されている．ＫＬ１－Ｕは，ユーザ言語であり最上位に位置するものである．ＫＬ１－Ｃは，言語の基本枠組みを規定したもので，制限されたＧＨＣ（フラットＧＨＣ，以下ＦＧＨＣ）に対応する．ＫＬ１－Ｂは，ＫＬ１の機械語に対応するものである．

一方，ＰＩＭマシンのアーキテクチャも現在設計中であり，ＩＣＯＴでは，ＰＳＩ－ＩＩを核に結合したシステム（ＭＰＳＩシステム）の開発も行っている．そこでまず，両者に共通のマシン独立の命令セットを決め（抽象命令セットと呼ぶ），これを用いて検討，評価を行いながらアーキテクチャ，命令の詳細化を行っていくことにした．

本稿では，このＫＬ１－Ｂの抽象命令セットとＫＬ１－Ｃから抽象命令セットへのコンパイラについて説明する．

なお，逐次処理系に関しては，［２］を参照されたい．

２．基本方針

ＫＬ１－Ｂ抽象命令を設定するにあたっての基本方針及び想定した抽象マシンの概要は以下の通りである．

（１）ＰＩＭのＰＥの抽象マシンとしては，タグ付データを格納する引数レジスタを多数持ち，リダクション対象のゴール引数は，この引数レジスタ上にキャッシュされた後ユニフィケーションが行われる様なマシンを仮定する．すなわち変数のユニフィケーションに関しては，ＷＡＭ［３］に似たコードを出す．

（２）ＫＬ１ではPrologの様なバックトラックが無い代わりにゴール実行の中断／再開がある．そこで，ゴールの実引数，コード，制御情報はゴールレコードと言う領域をメモリ上にとり，ここに格納することにする．また，実行可能ゴールは，レディキューと呼ぶキュー（スタック）で管理し，コンパイラが制御命令としてレディキューを操作する命令を出す．

（３）各ゴールに対する候補節群は，基本的にプログラマの書いた順に上から下へ，一つの節内では左から右へ実行が進むようにコード生成を行う．すなわち，１つのゴールに対する実行は逐次である．

（４）ゴールの実行が中断したときに行う処理をする命令を陽に出し，この命令で中断の原因となった変数にゴールをフックする（non-busy waiting）．このため，受動部の実行中に中断の原因となった変数を一時確保しておくために作業用のスタックを用いる．

（５）中断していたゴールの実行が再開される時は，コードの最初（すべての候補節の先頭）から行なわれる．

３．命令セット

ＫＬ１－Ｂの抽象命令セットは，大きく別けて以下の５種に分類できる．

- passive unification 命令
- active unification 命令
- 引数準備命令
- 実行制御命令
- 組込述語

３．１ passive unification 命令

wait_variable	Ai, Aj	read_variable	Ai
wait_value	Ai, Lab	read_value	Ai, Lab
wait_constant	C, Ai, Lab	read_constant	C, Lab
wait_list	Ai, Lab		

受動部におけるunification を行う命令群である．WAM では get_variable/value, unify_variable/value などの命令が出ていた．しかし，ＫＬ１の受動部では，呼出し側に値を送出することはないし，値が未定義のときサスペンドする可能性がある．そこで，新しい名前の命令群を定義し，機能を明確にした．

命令中で，Lab は，この命令で実行が中断したときに次に試みるべき候補節の番地を示す．また，read_XXX命令は，構造体（リスト）の要素の passive unification を行う．

３．２ active unification 命令

get_variable	Ai, Aj	unify_variable	Ai
get_value	Ai, Aj	unify_value	Ai
get_constant	C, Ai	unify_constant	C
get_list	Ai		

能動部におけるunification 命令である．unification の結果，値が決まることにより，サスペンドしていたゴールが起きることがある．また，ＫＬ１では，active unification 以外では構造体のユニフィケーションにおける"mode"は無い．

一方，マルチプロセッサ上での実現の場合，変数のロックなどの観点から"write mode 決めうち"方式による命令も考えている［４］．

３．３ 引数準備命令

put_variable	Ai, Aj	set_variable	Ai, Gi
put_value	Ai, Aj	set_value	Ai, Gi

Parallel Inference Machine : PIM
-- An Abstract KL1 Instruction Set and Its Compiler --

Yasunori KIMURA, Takashi CHIKAYAMA, Kouichi KUMON, Hiroshi NAKASHIMA
ICOT ICOT Fujitsu Ltd. Mitsubishi Electoric Corp.

```
put_constant  C, Ai      set_constant  C, Gi
put_list      Ai         set_list      Gi
write_variable  Ai
write_value     Ai
write_constant  C
```

ボディゴールをforkするときの引数を準備するための命令である。格納先がメモリか，レジスタかによってset_XXXと put_XXX の違いがある。また，write_XXX 命令は，構造体をメモリ上に新たに作るときの要素をセットする命令である。

3．4 実行制御命令

```
try_me_else     Lab
create_goal     Goal/Arity
enqueue_goal    Goal/Arity
execute         Goal/Arity
proceed
suspend         Goal/Arity
```

create_goal，enqueue_goal は，forkするゴールの領域を獲得し，レディキューに繋ぐ命令である。proceed は，或るゴールの実行が成功したのち，新たなゴールをレディキューからとりだして実行を始める命令である。suspend は，どの候補節もコミットしなかった場合の処理を行う命令である。try_me_else のLab は，wait_XXX 命令のそれと同様で，次候補節のアドレスを示す。この命令は，語長の関係で wait_XXX 命令に Lab が入らないときのことを考慮して設けられた。

3．5 組込述語

```
integer   Ai, Lab        wait      Ai, Lab
equal     Ai, Aj, Lab    greater   Ai, Aj, Lab
less      Ai, Aj, Lab    add       Ai, Aj, Ak
multiply  Ai, Aj, Ak     divide    Ai, Aj, Ak
modulo    Ai, Aj, Ak
                :
```

受動部における組込述語の実行を行う命令である。実行によりサスペンドする可能性のあるものと，そうでないものがある。また，入力引数のタイプチェックを行う命令も含む。各々の組込述語が呼ばれるときには，その入力引数は，値が決まっているものとする。従って，組込述語のなかで値を待ってサスペンドすることはない。

4．コンパイラ

このコンパイラは，実験用であるため，移植性，改良の容易性を考慮してPrologで記述した。DEC-10 PROLOG 互換の処理系上で稼働している。図1，図2にＫＬ１のサンプルプログラムとそのコンパイル結果を示す。

```
foo([X|Y], A) :- X > 0, add(X, A, Z) |
                        bar(Z, A), foo(Y, A).    ①
foo([X|Y], A) :- true | bar(X, A), foo(Y, A).    ②
```

図１．サンプルプログラム

```
foo/2:  wait_list       a1, foo/2/1       (1)
        read_variable   a3                (2)
        read_variable   a5                (3)
        put_constant    0, a4             (4)
        integer         a3, foo/2/1       (5)
        greater         a3, a4, foo/2/1   (6)
        integer         a2, foo/2/1       (7)
        add             a3, a2, a1        (8)
        create_goal     foo/2             (9)
        set_value       a5, g1            (10)
        set_value       a2, g2            (11)
        enqueue_goal    foo/2             (12)
        execute         bar/2             (13)
foo/2/1:wait_list       a1, foo/2/2       (14)
        read_variable   a1                (15)
        read_variable   a3                (16)
        create_goal     foo/2             (17)
        set_value       a3, g1            (18)
        set_value       a2, g2            (19)
        enqueue_goal    foo/2             (20)
        execute         bar/2             (21)
foo/2/2:suspend         foo/2             (22)
```

図２．コンパイル結果

図２で(1)～(13) が図１の①のコードに，(14)～(21) が②のコードに対応する。(22) が中断処理命令である。

引数レジスタ経由で渡された実引数は，コミットするまで壊さない事にしているので，(2)，(3)では別のレジスタを使っているが，(15)ではこれ以後サスペンドすることはないのでレジスタを破壊的に使っている。(8) の出力変数用のa1 も同様である。(5) は，a3 をデレファレンスし，その結果をa3 に置き，結果が整数であればそのまま次命令へ，さもなくば 次クローズ（foo/2/1 で示される）へ分岐する命令である。(9)～(12)，(17)～(20) でゴール foo/2 をフォークしている。現在は，コミット後の最左のゴールをそのまま実行し，残りのゴールをフォークしている。また，(12)～(13)，(20)～(21) の間にはそれぞれ

```
put_value  a1, a1
put_value  a2, a2
```

という命令列が省略されている。コンパイラで，レジスタ割付の最適化を行ったことによる効果である。

5．おわりに

ＫＬ１の抽象命令セットとそのコンパイラについて報告した。現在この抽象命令セットを用いて，ＰＥ設計および命令の詳細化を行っているところである。今後は，インデキシング命令や，並列実行命令の設計を行う予定である。

＜参考文献＞

［１］K. Ueda : GUARDED HORN CLAUSES, LPC '85, pp.225 1985-6, JAPAN

［２］久門他：並列推論マシンＰＩＭ 本大会2P-2

［３］D.H.D.Warren : An Abstract Prolog Instruction Set. Technical Note 309 SRI International

［４］清水他：並列推論マシンＰＩＭ 本大会2P-3