ＰＳＩ－ＩＩにおけるＫＬＯ実行順序系組込述語の実現方式

6B-6

中島　浩，立野　裕和，木村　康則
（三菱電機）　（ＩＣＯＴ）

１．はじめに

ＰＳＩ－Ⅱの機械語であるＫＬＯは，Prologにはないいくつかの特殊な実行順序制御機能を持っている．これらの機能は，Prologの単純な制御構造を補うものであり，大規模なプログラムの開発には極めて有用である．その反面，動的な述語呼出しなど，コンパイル時には予測できない動作を伴なうため，その効率的な実現は必ずしも容易ではない．本稿では，主な実行順序制御機能をまず説明し，それらの実現方式について，ＰＳＩとＰＳＩ－Ⅱの対比や，システム性能との関わりなどを含めて述べる．

２．ＫＬＯの実行順序制御機能

(1) bind-hook ：　組込述語bind-hook(X,Goals)は，変数X （フック変数）に値がバインドされたときに，呼び出されるゴール列Goals を定義する．例えば，

```
generate([X|L]):-genreate-element(X),
                 bind-hook(L,generate(L)).
check([X|L]):-check-element(X),
              check(L).
```

のような，lazy-generator/eagar-checkerのプログラムでは，述語check がその引数にリスト・セルをユニファイして，チェックすべき要素を要求するたびに，述語generateが呼び出される．

(2) exception ：　ＫＬＯの組込述語の多くは，その入力引数に対して，制限条件（データ型，値の範囲など）を持っている．この制限条件に反するような引数が入力されたとき，組込述語は例外（exception ）の種類に応じたハンドラ述語（exception handler ）に置換えられる．ハンドラへは，組込述語の識別子，例外の詳細な内容を示すコード，及び組込述語の引数が，引数として渡されるので，例外に応じた処理を行なうことができる．例えば，ほとんどの例外はバグにより発生するため，トレーサを呼出すことによりプログラマにプログラムの続行／中断などを問合わせるようなハンドラが，システムによりあらかじめ用意されている．また，プログラマが例外処理を自ら記述したい場合は，組込述語exception-hook(Exception,Handler) を用いて，特定の種類の例外Exception に対するハンドラHandler を定義することができる．例えば，

```
...,exception-hook('integer-overflow',
                   bignum-calc/8),...
bignum-calc(-,add,X,Y,Z,-,-,-):-
                   add-bignum(X,Y,Z).
```

のようにexception を用いて多倍長演算を記述することもできる．

(3) on-backtrack ：　組込述語on-backtrack(Goals,Id)は，バックトラックの際に実行され，カットされないゴール列Goals を定義する．即ち，

```
...,on-backtrack(Goals,Id),...
```

は，

```
...,(true ; Goals,fail),...
```

とほぼ同様であるが，カットによっても選択肢が除去されない点が異なる．例えば，

```
set-slot-and-trail(Obj,Slot,Value):-
      Old=Obj!Slot,
      Obj!Slot:=Value,
      on-backtrack(undo-slot(Obj,Slot,Old),-)
undo-slot(Obj,Slot,Old):-Obj!Slot:=Old.
```

のように，on-backtrackを用いて破壊的代入のundo処理を行なうことができる．なお，組込述語on-backtrackの引数Idには，生成した選択肢の識別子が返される．この値を用いて，組込述語reset-on-backtrack(Id)により，選択肢を除去することができる．

(4) remote-cut ：　組込述語absolute-cut(Level) は，述語の呼出しレベルLevel 以降に生成された選択肢を全て除去する．例えば，

```
top-level(Obj):-level(Lev),
                Obj!level:=Lev+1,
                goal(Obj).
top-level(Obj):-abort(Obj).
goal(Obj):-...
      ....
error(Obj):-absolute-cut(Obj!level),fail.
```

のように，例外発生時にabsolute-cutを用いて不要な選択肢を除去して，直接特定の選択肢にバックトラックすることができる．

３．処理方式

実行順序制御機能は，ソフトウェアで実現するのが不可能（または極めて困難）な機能であり，ＫＬＯを処理するマイクロプログラムが実行時に特殊な処理を行うことが必要である．但し，これらの機能は実行される頻度が小さい例外的処理であるので，それ自信の処理を高速化することよりも，通常の処理の速度への悪影響をできるだけ小さくすることが重要である．ＰＳＩ－Ⅱではこの考え方に基づいて，処理方式を工夫している．

(1) bind-hook ：　組込述語bind-hook は，ゴール列を呼出す命令列へのポインタと，引数ベクタへのポインタなどからなる制御ブロックをグローバル・スタックに作り，フック変数には制御ブロックへのポインタを格納する．フック変数へのバインドを行なうと，この制御ブロ

Implementation of KL0 Sequence Control Functions in PSI-II
Hiroshi Nakashima (Mitsubishi Electric Corp.)
Hirokazu Tateno (Mitsubishi Electric Corp.)
Yasunori Kimura (ICOT)

ックに従ってゴール列が呼出される．さて，Prologの重要な最適化手法の1つに，first goal optimization がある．これは，最初のゴールを呼出すまでは，述語の実行環境が破壊されないことを利用し，ヘッドに出現した変数の最初のゴールへ引渡しを，スタックを経由せずにレジスタなどの一時的な記憶領域を用いて行なうものである．さて，ヘッド・ユニフィケーションでフック変数に値をバインドすると，最初のゴールが呼出される前に（ヘッド・ユニフィケーションが全て終了した時点で）変数にフックされているゴール列が呼出されてしまい，実行環境が破壊される．そこで，ＰＳＩでは以下の方法をとっていた．

①最初のゴール呼出してあることを示すフラグを持つ．このフラグは述語呼出し時にセットされ，ユニット・クローズが成功したときにリセットされる．

②ゴール引数の引渡しの際，①のフラグの状態により，レジスタ（スタック・バッファ）経由かスタック経由かを，実行時に決定する．

③ヘッド・ユニフィケーション終了後，フック変数に値をバインドしたか否かを調べ，バインドしていればスタック・バッファを退避し，ゴールを呼出す．①のフラグはゴールの実行完了時にリセットされているので，「最初」のゴールの引数は，スタック経由で引渡される．

この方法には，次のようなオーバヘッドがあった．

a)引数の引渡し方法を，実行時に決定すること

b)ヘッド・ユニフィケーション終了時に，フック変数へのバインドの有無を調べること

そこで，ＰＳＩ－Ⅱでは以下のような方法を採用した．

①ゴール引数の引渡し方法（即ちtemporary/permanent variableの区別）はコンパイル時に行なう．

②フック変数に値をバインドした場合はトラップ・フラグをセットする．このトラップは一般の割込要因とマージされ，命令の切れ目でマイクロプログラムの割込処理ルーチンへの分岐を引起こす．

③割込処理ルーチンはトラップを認識し，次の命令がヘッド・ユニフィケーション命令（ｇｅｔ，ｕｎｉｆｙ）であれば元のフローに戻り，そうでなければゴール呼出しの処理を行なう．

④ゴール呼出し処理では，ローカル・スタックに特殊なenvironment を作り，引数レジスタを退避する．

⑤コンパイラはbind-hook 組込述語の引数Goals を呼出す命令列の最後に，end-of-bind-hook-handlerなる命令を付加する．この命令は，④で作ったenvironmentを用いて元の実行環境を復元する．

この方法では，フック変数へのバインドが行われた時の処理が若干遅くなるものの，通常の処理へのオーバヘッドは全く無い．

(2) exception：　組込述語は実行環境を破壊しないので，temporary/permanent variableの区別においては，引続く通常のゴール呼出しの一部であると見なすことができる．従って，例外が発生し組込述語が（一般のゴールである）ハンドラ述語に置換えられた時には，(1) で述べたフックされたゴールの呼出しと同じ処理が成される．また，組込述語ではその出力値を単に指定されたレジスタに代入し，出力引数とのユニフィケーションは組込述語の後のｇｅｔ命令で行なうので，例外発生時の出力先レジスタの値はなんら意味がない．そこで，グローバル・スタックに変数セルを生成し，出力先レジスタにそのセルへのポインタを格納することにより，ハンドラが値を戻すことができるようにしている．

(3) on-backtrack：　on-backtrack組込述語により生成される選択肢は，カットによっても除去されないという性質があるので，通常の選択肢とは異なる取り扱いが必要である．そこで，以下の方法を用いている．

①on-backtrackはゴール列を呼出す命令列へのポインタと引数ベクタへのポインタなどからなる制御ブロックをグローバル・スタックに生成する．また，制御ブロックへのポインタをトレイル・スタックにプッシュする．

②バックトラック時のundo処理において，トレイル・スタック中に制御ブロックへのポインタがあれば，制御ブロックにしたがってゴール列を呼出す．

③カット処理でトレイル・スタックの選別除去を行なう際には，制御ブロックへのポインタは除去しない．

この方法では，undo処理及びトレイル・スタックの選別除去の際に，トレイル・スタックのエントリが制御ブロックへのポインタであるか否かを判別する必要があるが，制御ブロックへのポインタのタグを特殊なものとすることにより，オーバヘッド無く判別できるようにしている．

(4) remote-cut：　remote-cutのためには，述語の呼出しレベルの管理が必要であり，以下の手間は回避することができない．

a)述語呼出しの際の呼出しレベル・カウンタの増加．

b)カウンタのenvironment への退避と復旧．

c)カウンタのchoice point への退避と復旧．

さて，remote-cut処理では，指定されたレベルでのゴール呼出し以後に生成された選択肢を除去する（指定された値以上のレベルを持つ選択肢ではない）．従って，あるchoice pointの除去の可否の判定には，レベルだけではなく述語の呼出し関係の情報が必要である．environmentの連鎖はこの関係を表したものであるが，ＴＲＯにより情報の一部が失われている．ＰＳＩでは，environmentの一部とchoice pointを組合わせた制御フレームによって実行制御を行ない，そのフレーム中にＴＲＯによっても失われないような呼出し関係の情報を保持していた．ＰＳＩ－Ⅱでは，この情報の保持のための手間を省くために，以下の方法を用いている．

①指定されたレベルより小さくかつ最大のレベルを持つenvironment （ＡＥ：ancestor environmentと呼ぶ）をＥから始まる連鎖の中から見つける．

②あるchoice pointが指定されたレベル以上であり，かつＡＥに対応するクローズへのcontinuation pointが実行中の述語と同じであるときに，そのchoice pointを除去する．

この方法では，呼出し関係の判定には，environment 及びchoice pointに格納されたＥとＣＰのみを用いるので，余分な情報を保持する必要がない．

4．おわりに

ＫＬＯの重要な特質である実行順序制御機能を，ＰＳＩでの処理方式を見直すことにより，最小限のオーバヘッドで実現することができた．