オブジェクト指向論理型言語における実行制御

6E−7

秋山 友子　鳥居 悟　小野 越夫
㈱ 富士通研究所

【1】はじめに

近年，コンピュータは単なる計算の道具としてだけではなく，より人間の社会活動に近い処理をする為の道具として利用される様になりつつある．その結果，知識情報処理が脚光を浴びる様になった．しかし，大多数のプログラム作成者は，依然としてコンピュータが大量のデータ群を高速に演算することを重要視し，知識処理システムを実用的に使用する段階には到っていない．

省みるに，人間の社会活動は計算のみ，もしくは知識処理のみによって成り立っているわけではない．従って，実用的なシステム構築には従来の様な計算機能もしくは知識処理機能といった単一機能だけでなく，これらの両方を一つのシステムで扱えることが必要である．

更に，深刻さを深めつつあるソフトウェア危機を打開する為には，一つ一つのプログラムパーツの再利用性を高めることを考えなくてはならない．

我々はこれらの問題を解決すべくプロローグをベースとした処理系を試作した．

本稿では，この処理系の実用的プログラミングのサポートの為の実行制御機能及びC言語インタフェース機能について述べる．

【2】実用的プログラミングへのアプローチ

プロローグは，知識処理には非常に強力であるが，反面，単純計算や，実行順序の制御を必要とする処理には弱い．

我々は，この弱点を補う様なプロローグ処理系を試作している．その特徴を以下に述べる．

① プロローグの組み込み述語として様々な実行制御を提供する．
② C言語とのインタフェースを提供する．
③ オブジェクト指向を導入する．

①により手続き的処理を強化しプロローグの欠点を補う．また，②によりCでインプリメントされた既存プログラムの再利用性を高める．これは，同時にC言語の持つ移植性の良さをも取り込むことになる．さらに，③によりプロローグプログラムの再利用性を高めることを可能にし，また，目的に応じたプロローグとC言語の簡単な使い分けを可能にする．これにより，実用的なプログラム作成効率が大きく向上すると思われる．

【3】プロローグにおける実行制御

本処理系が実行制御機構として，サポートする組み込み述語を以下に示す．

①ｃａｔｃｈ＆ｔｈｒｏｗ
catch(Id,Goal) / throw(Id)
→ グローバルＥＸＩＴを実現する
②メタコール
meta_call(Goal)
→ 動的な述語呼び出しを可能にする
③バインドフック
bind_hook(Var,Goal)
→ 未定義変数に値が代入された時に，あらかじめフックしておいたゴールを起動する
④カット
cut
→ 述語のオルタナティブを刈り取る
⑤フェイル
fail
→ 意図的に実行を失敗させる
⑥例外処理フック
exception _hook(Id,Goal) / exception(Id)
→ 例外事項が発生した時に，あらかじめ与えておいたゴールを起動し，デフォルトの例外処理を行わない

【4】C言語インタフェース

本処理系はプロローグとC言語とのインタフェースとして，以下の機能をサポートする．

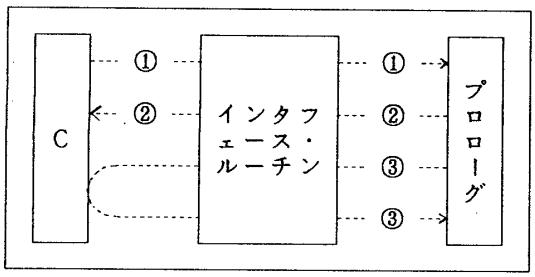
①Cからプロローグを呼び出すコールイン機能
②プロローグからCを呼び出すコールアウト機能
③プロローグから呼び出したCの中でさらにプロローグを呼び出すコールバック機能

（①と③は一見似ているが，副作用をもつデータの扱いや，引数の受け渡しの面で扱いが異なる）

これらの機能の実現に必要な処理はシステムの提供するCインタフェースルーチンが一括管理している．（図1参照）

以下では，上記三機能について詳述する．なお，Warrenの中間コード(1)についての一般的な知識があることを前提とする．

Execution Control System on Logic Based Object Oriented Language
Yuko Akiyama, Satoru Torii, Eetsuo Ono
FUJITSU LABOLATORIES LTD.



①　コールイン　　②　コールアウト
③　コールバック

図1　インタフェースルーチンのイメージ

4.1　C→プロローグ

Cからプロローグを呼び出す為に，C関数call_prologを提供する．

call_prologはCからのプロローグ呼び出しの為に，述語のアドレスをサーチしたり，述語の引数値を引数レジスタに設定する，といった処理を行う．

4.2　プロローグ→C

プロローグからのC呼び出しの為にCクラスというCとのインタフェース用のオブジェクトの枠組みを用意する．Cクラス内のメソッドのボディ部では一つのC関数と複数の組み込み述語を呼び出すことができる．

例を図2.1，図2.2に示す．図2.1はCクラスにおけるメソッド定義の例であり，図2.2はCクラスから呼び出されているC関数の例である．

```
c_class  c_std_io
    :
  :c_write_atom( Obj, Atom ) :-
      write_atom( Atom ) ;
    :
```

図2.1 Cクラスの定義

```
write _atom( atom )
ATOM   atom;
 {
     char   *symbol ;

     get_symbol( atom, &symbol ) ;
     printf( "%s", symbol ) ;

     return 1 ;
  }
```

図2.2 C関数の定義

また，Cクラスのオブジェクト内に1ワード（データの単位）の他言語領域をもち，そこにはプロローグのタグを持たないCデータを保持できることとする．これにより，Cの世界で扱いたいアドレスなどをCクラスのオブジェクトという皮を被せてプロローグの世界でも扱うことを可能にしている．

4.3　C言語呼び出しにおける値の上書き

本処理系においては，未定義変数はすべてフェイル時に未定義に戻されることとしている．従って，C関数の中で上書きされたデータに関しても，それがユーザ定義のC関数の時は，システム側でフェイル時に未定義変数を元に戻す為の処理をしておく必要がある．（ユーザ定義のC関数の中で値の代入時にこの処理をさせると，プログラムが見づらく，かつ再利用がしづらくなる為）

また，既にバインドされている変数に対しては，C関数の中での上書きを許さないこととしている．未定義変数に値がバインドされていた場合，バインドされた値が，プロローグのデータ型として正しいかどうかをチェックし，トレイルスタックにバインドされた変数のアドレスを退避する．また，未定義以外の変数の値が書き変わっていた場合は例外を発生する．

4.4　C関数の戻り値

本処理系では，プロローグからC関数を呼び出す場合，C関数からの戻り値を判断し，戻り値が負数であれば，呼び出した述語（実際はメソッド）をフェイルさせる．
戻り値が，正数もしくは0であれば，メソッドを決定的に成功させる．

C関数の戻り値自体はプロローグに渡さない様にする（値のやりとりは引数を経由させる）

【5】まとめ

プロローグベースの言語における，実用的なプログラム作成のサポートを目標とした言語処理系の実行方式を紹介した．この処理系は，
①組み込み述語として実行制御を持つことにより手続き的処理を強化する
②C言語インタフェースの導入により，C言語の既存プログラムの利用を可能にし，プログラムの移植性を高める
③オブジェクト指向を導入することにより，プロローグのモジュール化を促進させ，プロローグプログラムの再利用性を高める
という点で実用プログラミングに適していると言える．

今後は，Cインタフェースにおける引数モードの導入などの具体的な使い勝手の充実と，よりグローバルな視点からの実用プログラミング作成環境とを併せて検討していきたい．

【6】参考文献

(1)　D. Warren :
AN ABSTRUCT PROLOG INSTRUCTION SET
SRI Technical Notes 309