融合型プログラミングに関する一検討

7Q-1

鶴巻 宏治
NTTヒューマンインタフェース研究所

1．はじめに

従来から我々が開発を進めてきた新ＥＬＩＳ[1]の言語仕様・処理系はＥＬＩＳ－１[2]上のプログラミング言語ＴＡＯの特徴であるマルチパラダイムを継承している。マルチパラダイム言語の開発効率は検証済みであり、特に論理型プログラミングにおいては、Ｌｉｓｐの関数や特殊型式を活用することによる実行効率の向上も著しい。

本稿では、論理型プログラミングパラダイムの基本計算機構に関して、特に他のパラダイムとの融合の局面における新しい特徴について検討する。

2．融合型プログラミングの利点

論理型プログラミングにおいてしばしばＬｉｓｐの機能を効果的に活用できるが、大きく分けて次の２つの場合に分類できる。

(1).繰返しや非局所的脱出など強力な制御構造が利用できる

(2).リスト操作、数値演算、述語関数など豊富な高速大容量の関数が利用できる

これらを利用することにより、論理型プログラミングの宿命である再帰的定義をプログラムから大幅に削減することができる。さらに、多くのＬｉｓｐプリミティヴがμプログラムで実装されていることとあわせて、実行効率の画期的な向上が見込める。以下では、それぞれの局面における問題点と解決法を述べる。

3．Ｌｉｓｐ制御構造中のカット

論理型プログラミングでの論理述語の定義にＬｉｓｐの制御構造を使えることはプログラムの開発効率、実行効率の両面で有益である。しかし、パラダイムの融合によって強力な記述力が得られたとしても、その代償としてプログラムの読みやすさが損われることがあってはならない。端的な例としては、Ｐｒｏｌｏｇにおいてさえ理解しづらいとされるカット(！)の挙動が、融合型プログラミングにおいてさらに理解しにくくなりかねない、という問題がある。

新システムでは、カット(！)の意味をできるだけ直観的に理解しやすいように明確に規定した。Ｐｒｏｌｏｇのカットはカット・オペレータを含む述語全体の選択肢を捨てる。たとえ(！)が選言(；)の中にあっ

```
(assert (maximum _x _y)          ..... 第1節
        (== _y ,(mapcon #'(lambda (x &aux _y0 _y1 _y2)
                            (&and (== ,x ((_ . _y0)(_ . _y1)(_ . _y2) . _))
                                  (< _y0 _y1) (> _y1 _y2) ! (list (cadr x))) ) _x))
        _y )
(assert (maximum _x _x))         ..... 第2節
```

図1．論理述語maximumの定義

```
(goal (maximum ((0 . 1)(1 . 3)(2 . 0)(3 . 1)(4 . 2)(5 . 2)(6 . 4)(7 . 2)) _x))
_x = ((1 . 3) (6 . 4)) ;
no
```

図2．論理述語maximumの実行例

A consideration about fusion of programming paradigms
Kouji TSURUMAKI
NTT Human Interface Laboratories

ても同様である。従って、融合型プログラミングにおいても「カットはそれ自身を文脈的に含む論理述語の定義節の選択からカットまでの実行を決定的にし、すべての選択肢を捨てる」と規定した。

繰返しの制御構造の一例としてマッピング関数mapconを使った例を示す。図１は２次元座標上の座標列からすべての極大点を求め、１つもないときは座標列自身を返す論理述語maximumの定義である。極大点が１つでも見つかったら４行目のカット(！)によって第２節の選択肢は捨てられる。図２に示す入力が与えられると、２つの極大点{(1,3),(6,4)}が得られ、その後強制的にバックトラックさせると失敗する。

４．関数計算との融合

論理型プログラミングでもリストを操作したり数値演算を行った結果を扱いたいことはよくある。そのようなときＬｉｓｐ関数で適当なものがあればそれを用いた方が実行効率がよい。Ｌｉｓｐ関数を実行した結果を論理述語の引数としたいときは、論理述語の目的の引数の位置にそのＬｉｓｐ関数の呼び出し形式にコンマを冠した式を書くことができる(図３(a)参照)。コンマがつけられたＬｉｓｐ形式はユニフィケーションの直前に評価される。コンマを用いることによって補助的な論理変数をプログラム中から削減することができるため、読みやすいプログラムを書くことができる。

新システムでは、この機能をさらに一歩進めて、より多くの場面で積極的にコンマのついた形式を使えるようにする。具体的には、論理述語定義の頭部引数に使えれば、さらに補助的な論理変数及び関数==を用いたユニフィケーションをプログラム上から減らすことができる。

図３(a)は論理風の階乗計算プログラムである。同じ階乗計算のプログラムを図３(b)のように書けると、論理型プログラミングとしてはより自然な記述であるといえる。但し従来はこのプログラムは正しく動かなかった。通常の呼び出し形式（fact 1 _a）と（2'）とのヘッド・ユニフィケーションの時に、論理変数 _a と コンマ形式 ,(* _n _x) との同一化が試みられるが、後者の中の変数 _x がこの時点で未代入の状態なので乗算形式は評価できない。図３(b)のプログラムを正しく動かすためには、

・図３(a)のプログラムに変換する
・遅延評価機構[3,4]を導入する

いずれかの必要がある。新システムでは、インタプリタ実行においては遅延評価機構の導入を検討している。図３(b)のプログラムでは(2')の本体中の副目標（fact , (1- _n) _x）の実行が成功して変数 _x に値がバインドされた時点で _a と ,(* _n _x) との同一化が行われる。論理型プログラミング言語単体としては[3,4]ほど強力な枠組ではないが、融合型のプログラミングを行う場合には便利な機能である。

```
(assertz (fact 0 1))    .........(1)
(assertz (fact _n _z)   .........(2)
         (fact ,(1- _n) _x)
         (== _z ,(* _n _x)) )
```

図３(a)

```
(assertz (fact 0 1))    .........(1')
(assertz (fact _n ,(* _n _x)) ..(2')
         (fact ,(1- _n) _x))
```

図３(b)

５．まとめ

新ＥＬＩＳにおける論理型プログラミングパラダイムの基本計算機構に関する特徴について検討した。新ＥＬＩＳの言語仕様・処理系には従来のＴＡＯ/ＥＬＩＳ上でのプログラミングの経験が反映されている。今後さらに実行管理などの効率向上を追及するべく評価検討を進めたい。

[参考文献]
[1] 鈴木他,"新ELISシステム概念",信学全大,1989.
[2] Takeuchi,I.et al.,"A List Processing Language TAO with Multiple Programming Paradigms",New Generation Computing,1986.
[3] 中島秀之,"知識表現とProlog/KR",産業図書,1984.
[4] 戸村他,"TDProlog:項記述可能な拡張Prolog",日本ソフトウェア科学会第２回大会,1985.