5J-8 マルチパラダイム言語処理系MCによるプログラム開発

松野 年宏，市川 眞一（ファコム・ハイタック㈱） 井田 哲雄（筑波大・電子情報工学系）

1. はじめに

本稿では，われわれの開発したLisp-Prolog 融合処理系MC（Meta Computing environment）の概要を述べるとともに，MCによるアプリケーションプログラム開発を通じて得た，融合の効果についての評価を行う．

計算機によって処理される問題が拡大し，またそれらの複合問題の解決が要求されるにつれ，それを記述するためのプログラミング言語に要求される機能は多様化してきている．この要請に対する解決には，強力な新言語を開発する，あるいは，既存のプログラミング言語に適切なインターフェイスを設ける，という2つのアプローチが考えられる．MCでは，既存ソフトウェア資産，およびプログラミングノウハウの継承という現実的な配慮から後者のアプローチをとり，記号処理分野で広く利用されているLispとPrologの融合を図った．Lisp-Prolog 融合処理系自体は決して新しいものではないが，その効果についての評価は未だに定まっていない．われわれは，MCによる実際的アプリケーションの開発により，Lisp-Prolog 融合処理系が単なる研究的興味の対象ではなく，プログラム開発の効率化に有効であるとの評価を得た．

2 Lisp-Prolog 融合の意義

①Lispの特徴
- 制御構造を記述する手段が豊富
- マクロ機能による言語仕様の拡張が可能
- 局所的環境（局所変数，局所関数）の定義が可能

②Prologの特徴
- パターンマッチング機能
- 宣言的なプログラム記述が可能
- バックトラック機能

③共通の特徴
- リスト処理機能
- 再帰的プログラムを簡潔に記述可能
- インタプリタによる対話的処理が可能

LispとPrologはともに記号処理に適した言語と言われるが，それは，この2つの言語が，ともにリストという柔軟な可変長のデータを扱う機構を備えていること，および再帰的なプログラムが簡潔に記述できる点にあると考えられる．再帰的プログラムは再帰的な構造をもつリストと良く調和し，プログラムの記述量を減らす効果をもつ．

一方で，上に記したようにLispとPrologはそれぞれ独自の特徴をもっている．とりわけ，両者の決定的な相違は，Lispが問題の解を得るための手順を全て書き下してやらなければならないのに対し，Prologでは，宣言的なルールから自動的に解を探索する機構を備えている点にある．このことは，必ずしもPrologがLispよりも優れていることを意味しない．手続きの明確な問題に対しては，宣言的なプログラムを記述することはかえって煩わしいことだからである．手続きの記述には，Lispのマクロが適している．

多くの実際的な問題は，宣言的なルールで表現できる部分と手続き的に処理をする部分が混在している．このような場合，LispとPrologの役割分担によって，問題解決のための記述が容易になると考えられる．

3. MCのLisp-Prolog インターフェイス

3.1 方針

われわれは，Lisp-Prolog 融合処理系を構築する上で，次の目標を設定した．

①LispとPrologが対等であること
②両者で共有できるデータの種類をできるだけ多くすること
③インターフェイスの方法が直観的でわかりやすいこと
④融合によるLispとPrologの個々の処理系の性能低下を最小限にとどめること

①については，PrologをLispの中に埋め込むのではなく，LispとPrologそれぞれ別のトップレベルループを用意し，モードの切り換えにより，LispあるいはPrologを選択できるようにした．また，各シンボルはLisp関数とProlog述語の定義を同時にもつことができる．②については，Lispで扱う任意のデータはPrologの変数のinstantiation の対象となり得るようにした．③は，Lispの値の代入あるいは束縛によるデータの受け渡しとPrologのユニフィケーションによるデータの受け渡しを自然な形で融合することであるが，現段階では，基本的な操作関数（あるいは述語）を利用者に提供しているだけである．④については，LispによりPrologを実現する方法はさけ，LispとPrologの共用のメモリ管理機構の上に個々の言語処理系をインプリメントする戦略をとった．

3.2 関数，述語の定義

PrologはLispとの親和性を良くするために，Lispと同じS式表現を用いている．これによって，LispとPrologを，関数あるいは述語単位で同一テキストに記述することができる．

(1)関数定義

COMMON LISP と同じ．

(2)述語定義

(DEFPREDICATE 《述語名》
{《patameter-list》 (《goals》)}*)

Programming by MC (Meta Computing environment)
Toshihiro MATSUNO*, Shinichi ICHIKAWA*, Tetsuo IDA **
*FACOM-HITAC Ltd., ** University of Tsukuba

3.3 呼び出し形式

(1)Lisp ⇨ Prolog

①(?- 《variable-list》《goals》)
《goals》をPrologとして評価し，それが成功したならば，最初の解について《variable-list》中の変数がユニファイされた結果が多値として返される．失敗したならば，シンボル*FAIL*が返される．
②(?all 《variable-list》《goals》)
全ての解をリストにして返す．
③(?any n 《variable-list》《goals》)
n 番目の解を返す．

(2)Prolog ⇨ Lisp

①(IS 《term》《form》)
《form》がLispとして評価され，その主値が《term》とユニファイされる．
②(ARE《term-list》《form》)
《form》がLispとして評価され，それが返す多値の個々が《term-list》中の項に順にユニファイされる．

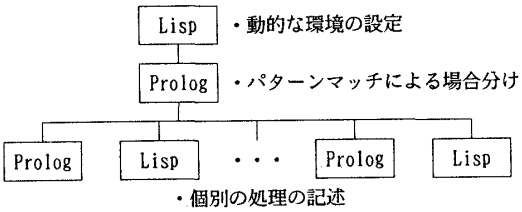
4. MCによるプログラム開発例

4.1 MC/LISP コンパイラのMCによる記述

プログラムの構造としては，下図のようにmainプログラムをLispにより記述し，内部中間形式の生成とコードの生成の実質的な部分をPrologで，末端の手続きのいくつかをLispでそれぞれ記述している．LispとPrologのそれぞれの記述部分は次のようである．

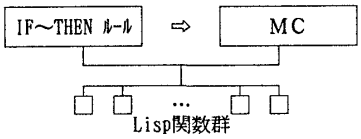
①Lisp ； I/O,繰り返し処理，動的な環境の設定・参照
②Prolog ； プログラム変換，構文解析

LispとPrologの構成比は，Lisp関数（またはマクロ）53，Prolog述語97とPrologの記述部分が多い．これは，コンパイラの処理では，パターンマッチングが中心であり，これにはPrologのユニフィケーションを用いると記述が容易となるからである．この問題はPrologに比較的適したものであるが，Prologでは記述しにくい部分をLispによって補う形となっている．とくに，コンパイラ内部で用いる定数等の，あるスコープ内でのみ有効な動的環境の設定・参照では，Lispのspecial 変数を用いるのが，Prologのassertとretract の組合せよりもはるかに容易である．また，全リスト要素に対する共通の操作は，Prologで再帰的プログラムを記述するよりも，Lispのmap 関数や，doによる繰り返しで記述するほうが直観的でわかり易い場合が多い．



4.2 アプリ開発におけるｴｷｽﾊﾟｰﾄｼｽﾃﾑ構築ツールとの比較

エキスパートシステム構築ツール（ESHELL）にて開発された化学反応式の合成プログラムのうち，IF～THENルール部分をMCにより書き換え，両者の比較を行った．



(1)記述性

知識と推論機構の分離あるいは，知識表現の定形化というエキスパートシステム構築ツールの特徴は，決定的な問題の記述のためには必要ではない．記号処理の決定的問題の効率的な解決という観点からは，宣言的なルール記述による探索手順の省略が重要である．IF～THEN型ルールとPrologの述語は，ともに宣言的なプログラムの記述手段を提供するものとして比較の対象となる．

宣言的ルールの記述手段として両者を比較した場合，全解の探索を行う記述では，IF～THEN型で表現すると探索のための手続きをループ構造で記述する必要があるのに対し，Prologを用いることにより宣言的に記述にすることができる．また，ルールの条件部が，排他的なものと，そうでないものが混在する箇所では，PrologとLispを併用することにより，条件部の記述が簡潔になる．

(2)記述量

・ESHELL 1130 line （内部表現のLispソースの規模）
1270 line （browser の出力）
・MC 1110 line

(3)性能

エキスパートシステムの知識は手続きではなくデータとして扱われ，推論エンジンがそのデータを解釈することになるため，LispやPrologで規則を手続きとして記述しコンパイルした場合に比べ，処理性能は劣ると考えられる．現時点ではPrologコンパイラが未完のため，実測は行っていない．

5. まとめ

Lisp-Prolog 融合処理系MCを用いたアプリケーション開発による，LispとPrologを融合の効果の評価を行った．そこでは，Prologによる宣言的な記述とLispによる手続きの記述により，問題のプログラム化が容易になることが示された．

参考文献

[1] T. Ida, T. Matsuno, A. Nakamura; Implementing Lisp and Prolog on a common abstract machine: A practical approach to combining functional and logic programming language, Proceedinds of the IFIP TC10/WG10.1 workshop on concepts and characteristics of declarative systems, Oct. 1988
[2] 松野，井田；マルチパラダイム環境MC上のLispコンパイラの実現，情報処理学会第36回全国大会論文集，1988
[3] 松野，井田；LISP/MC コンパイラの型推論，情報処理学会第37回全国大会論文集，1988
[4] 松野，井田；マルチパラダイム処理系MC上のCommon Lispの実現，情報処理学会記号処理研究会，1988