前向き推論による並列オブジェクト指向論理プログラミング*

6J－5

田中　将光† 山田　昌寛‡　　中村　克彦¶
東京電機大学大学院理工学研究科§　　東京電機大学理工学部‖

1　はじめに

論理プログラムにはSLD融合にもとづく後ろ向き推論による計算方式のほかに単位融合にもとづく前向き推論による計算方式があることが知られている．これまでの論理プログラムを代表するProlog言語による直列計算および，Parlog, GHCなどの言語による並列計算はどちらもSLDまたは後向き計算にもとづくゴール駆動型計算モデルを採用しているが，前向き推論にもとづくアプローチは一般的な論理プログラミングとしてはあまり発展していない．しかし，後向き計算の方式は，多くの言語を持つ配列や連想記憶の使用などのデータ構造がないため，大量のデータの集合の処理を効率よく行いにくい．また，ゴール駆動型の並列計算モデルは，共通変数の代入による同期に強く依存しているため論理的な意味の多くを失っているなどの問題点がある．

われわれはデータ駆動型の前向き推論にもとづく方法によってこのような問題点を解決できるようなシステムを実現できる計算モデルを提案し，このための並列論理型言語Monologを開発している．本報告では論理型言語Monologの概要とオブジェクト指向プログラミングについて述べる．

*Object-Oriented Parallel Logic Programing Based on Forward Deduction

†Masamitsu TANAKA

‡Masanori YAMADA

§Graduate School of Science and Engineering

¶Katsuhiko NAKAMURA

‖The Faculty of Science and Engineering, Tokyo Denki University

2　Monolog言語

Monolog言語の入力と出力は単位節の集合であり，プログラムはホーン節の規則からなる．規則は単位融合によって単位節から単位節を生成し，並列オブジェクト指向型計算のオブジェクト（または，アクター）として扱われ，オブジェクトを生成するクラスの定義ともなる。単位節はオブジェクト間のメッセージとみなすことができる．

このモデルはPrologと同様な閉鎖型の計算システムのほか，ParlogやGHCと同様に外部と対話的に応答を継続する解放型のシステムともなる．

2.1　規則とオブジェクト

各$p_i(\overline{S_i})$, $q_j(\overline{T_j})$を原子式（アトム）とするとき，

$$p_1(\overline{S_1}),\cdots,p_n(\overline{S_n}) \rightarrow q_1(\overline{T_1}),\cdots,q_n(\overline{T_n}),$$

$(n,m \geq 1)$の形式をもつ．直感的には，左辺のすべてのアトムが作業記憶内の単位節との融合に成功したとき，右辺アトムの単位が生成される．このような各融合プロセスは非同期的にかつ並列的に行われる．

この規則は図1のようなオブジェクトとしての働きをもつ．単位節$p_i(\overline{U_i})$が規則のアトム$p_i(\overline{S_i})$と統一化を行うとその結果がアトム用の環境変数(E_i)に格納され，共通変数に対する代入の結果がこの規則の左辺の他のアトムの結果と統一化可能であるとき新しい単位節$q_1(\overline{T_1})\theta,\cdots,q_m(\overline{T_m})\theta$が生成されることになる．ここでの$\theta$は各環境の代入の合成である．

2.2　C++へのコンパイル

Monologプログラムはコンパイラによって C++プログラムに変換される．このとき，各規則につい

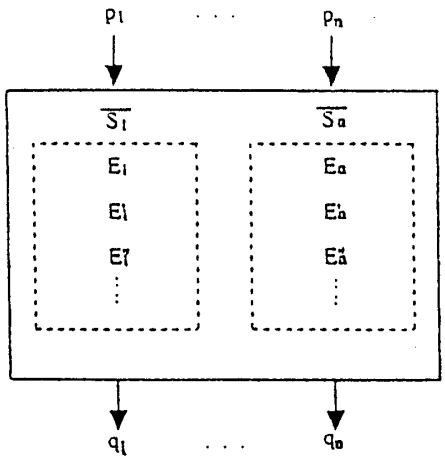

図 1: オブジェクトとしての規則

て 1 つのクラスが生成される．環境変数領域 (E_i) はプライベートなデータとして宣言され，規則オブジェクトの行う処理は C++のパブリックな関数として定義される．

環境変数領域は Monolog プログラムに記述された規則の左辺のアトムと規則とオブジェクトが受け取った単位節の単一化に成功するたびに更新される．左辺の全部のアトムが満足されるためには，共通変数の値が等しい変数環境の組合せを探索する必要がある．このため，われわれは項，リストをハッシュ (monocopy)・リストを用いることによって共通変数の比較の効率化をおこなっている．ハッシュ・リストでは同一のリスト構造を示すポインタは唯一なので，二つのリスト構造の同一性の決定をリストのポインタの比較で行えるため，単一化を高速化できるほか，変数の値から変数を求める連想演算にも使用できる．

3 プログラム例

図 2 に Erastothenes のふるいによる素数発生プログラムを示す．入力単位節`$get(max(M))`によって求めたい素数の最大値 M を与えると，出力単位節`#put(prime(X))`によって発生された素数が出力される．素数は述語 `sift/4` によってふるいにかけられる．このふるいは各素数の候補の数を含む単位節 g/3 に対して 1 度しか使われる必要がないため Monolog の制御機能`#excl` を用いて演算を制限している．また，`$(X-1)` などのような項は，単一化の際にこの算術式の値が用いられることを表している．さらに，変数の値の条件を与えることなどのために，`#mod(X,P)` のように Prolog 言語の組み込み述語を使用した条件を変数の後に付加している．

```
#get(max(M))  -> generate(2,M).
generate(X,Max),#(X=<Max)
   -> generate($X+1,Max),g(1,$X-1,X).
g(N,1,X)
   -> sift(N,2,1,X),#put(prime(X)).
#excl sift(N,K,M,P),g(N,K#K>2,X),
#mod(X,P)=/=0
   -> sift(N,$K+1,$M+1,P),g($N+1,M,X).
#excl sift(N,K,M,P),g(N,K#K>2,X),
#mod(X,P)=:=0
   -> sift(N,$K+1,M,P).
```

図 2 素数発生プログラム

4 まとめ

Monolog では単位節によって表される複数のデータの組み合わせによって直接融合を行うので，多数のデータ集合から記号的推論によって何らかの結論を引き出すような処理を効率よく行うことができる．現在，Monolog の制御機能の検討，各種テストプログラムによる評価，C++言語へのコンパイラの作成を行っている．さらに現在は直列実行であるがネットワーク上で動作する並列処理システムのインプリメントを行う予定である．

参考文献

[1] Nakamura. K, Parallel Logic Programming Based on Forward Deduction, 未発表.