7B−2

リフレクションを導入した並列論理型言語 RKL1 の実装と評価

竹内 剛　武田正之
東京理科大学大学院 理工学研究科 情報科学専攻

1 はじめに

並列論理型言語 KL1 にリフレクション [1] を導入し，KL1 プログラムの記述性を向上したプログラミング言語が RKL1 である [2]．リフレクションを導入することにより，ベースレベルとメタレベルのプログラムを分割することが可能であり，メタレベルのプログラムを独立したモジュールへ分割し，プログラムの再利用を可能にする．RKL1 ではリフレクションをボディ部に制限したため，ベースレベルとメタレベルに分割したプログラムを合成した KL1 プログラムへコンパイルでき，効率良く実行できる．

2 並列論理型言語 RKL1

RKL1 ではメタボディとメタプロセスグループという 2 つの概念から成るリフレクション機構を導入している．

2.1 メタボディ

メタボディはボディ部の実行の制御やカスタマイズをおこなうメタレベルの記述であり，拡張した KL1 で記述する．次のメタボディ add_x/1 は，元のボディのゴールに加えて，ゴール x/1 を実行するためのメタレベルのプログラムである．

```
:- metamodule msample.
add_x(V) :- x(V), $execbody.
```

$execbody はベースレベルのボディ部を実行するための記法である．メタボディ add_x/1 を用いる場合には，ベースレベルのプログラムにおいて対象となる節の前に %reflect で始まる注釈行を記述する．

```
%reflect msample:add_x(X).
p(X) :- q1(X) | r1(X), s1(X).
p(X) :- q2(X) | r2(X), s2(X).
```

リフレクションの指定が注釈行中であるため，リフレクションを導入していない普通の KL1 言語処理系でベースレベルのみを実行することも可能である．このように分割して記述されたベースレベルとメタレベルのプログラムを RKL1 コンパイラが次の KL1 プログラムへコンパイルする．

```
p(X) :- q1(X) | p_1_1_add_x_1(X).
p(X) :- q2(X) | r2(X), s2(X).
p_1_1_add_x_1(X) :-
                x(X), r1(X), s1(X).
```

メタボディではベースレベルのボディ部の変数の参照／具体化／置換，ゴールの追加／削除等が可能である．

2.2 メタプロセスグループ

メタボディの実行をメタプロセスと呼ぶ．メタプロセスグループはメタプロセス間で共有資源を扱うために導入された概念であり，これに属するメタプロセスはグループコントローラによって統括される．共有資源の管理はメタプロセス群がグループコントローラと通信することで実現する (図 1)．次のプログラムは負荷分散を行なうメタレベルのプログラムである．

```
:- metamodule msample2.
init(*strm) :- demander(*strm).
assign(*strm) :-
     *strm << N, $execbody@node(N).
```

init/1 でグループコントローラ demander/1 が起動され *strm がメタプロセスとの通信路となる．assign/1 ではグループコントローラから実行すべきノード番号を受け取り，ベースレベルのボディ部をそのノードで実行する．

Implemntation and Evaluation of RKL1 : Parallel Logic Programming Language Incorporated a Reflective Architecture
Tsuyoshi Takeuchi and Masayuki Takeda
Science University of Tokyo

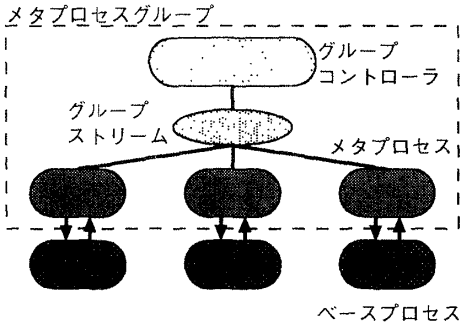

図 1: メタプロセスグループ

このメタレベルのプログラムを次のベースレベルのプログラムに適用する.

```
%defgrp msample2:init(*ns).
main :- gen(X), foo(X).
%reflect msample2:assign(*ns).
foo([H|T]) :- bar(H), foo(T).
foo([]) :- true.
```

%defgrp はメタプロセスグループを用いるための宣言であり, %defgrp と %reflect の行にある *ns は同一の通信路を示している. ここでは foo/1 を分散して実行するように指定している.

RKL1 コンパイラでは, メタプロセスとグループコントローラが通信を行なうための通信路をベースレベルのプログラムに追加する. 以下にコンパイル後の KL1 プログラムを示す.

```
main :- gen(X), foo(X, Ns),
    merge(Ns, Ns1), demander(Ns1).
foo([H|T], Ns) :- Ns = {Ns1, Ns2},
    Ns1=[N],foo_b(H,T,Ns2)@node(N).
foo([], Ns) :- Ns = [].
foo_b(H,T,Ns) :- bar(H),foo(T,Ns).
```

3 RKL1 コンパイラの概要

RKL1 コンパイラは KL1 で記述されており, ベースレベルとメタレベルの 2 つのプログラムから KL1 プログラムを生成する. RKL1 コンパイラは図 2 に示す構成になっている.

メタレベルとベースレベルのプログラムはそれぞれ別のファイルから読み込まれる. 始めに, メタプロセスとグループコントローラの通信経路を調べるために, ベースレベルのプログラムのゴールの呼び出し関係を示したゴール呼び出しテーブルを生成する. 次に, メタレベルのプログラムと

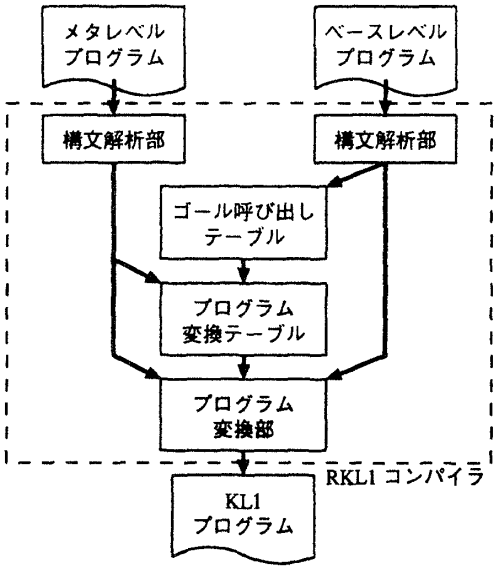

図 2: RKL1 コンパイラの構成図

ゴール呼び出しテーブルから変更が必要なベースレベルのプログラムの節を選びだし, どのように変更するかをプログラム変換テーブルに登録する. プログラム変換部では, プログラム変換テーブルを参照してベースレベルのプログラムの変更を行ない, KL1 プログラムを出力する.

4 おわりに

RKL1 の有用性を評価するため, いくつかの負荷分散のアルゴリズムをメタレベルで作成し評価実験を行なった. リフレクションを導入した並列論理型言語 RKL1 を利用することによって, ベースレベルとメタレベルのプログラムを分割し, メタレベルのプログラムをモジュール化することが可能である. したがって, メタレベルのプログラムを再利用することが可能になり, KL1 のソフトウェアの生産性を向上させる.

参考文献

[1] Maes,P.: *Issues In Computational Reflection*, In Meta-Level Architectures and Reflection, pp.21-35, North-holland, 1988.

[2] Takahashi, T. and Takeda, M.: *An Efficient Implementation of Reflection in KL1*, FGCS'94 Workshop on Parallel Logic Programming, Institute for New Generation Computer Technology, pp.17-26 Dec. 1994.