S14—1

ストリーム並列指向論理型言語の提案

A Proposal of A Stream Parallel-Oriented Logic Programming Language

松原 健二、 相田 仁、 田中 英彦
Kenji Matsubara, Hitoshi Aida, Hidehiko Tanaka

東京大学工学部
University of Tokyo

1. はじめに

論理型プログラムを並列に実行する方法は大別して、OR並列実行とAND並列実行とがある。あるゴールリテラルと単一化可能な定義節が複数ある場合に、単一化を並列に実行することをOR並列実行という。OR並列実行は処理単位間の独立性が高く、実現が比較的容易だと考えられるが、決定的な問題に対しては高い並列度が得られない。AND関係にあるゴールリテラルの単一化を並列に実行することをAND並列実行という。AND並列実行では、ゴールリテラル間で共有されている変数について、並列処理の結果が矛盾していないかどうか検査する必要がある。この無矛盾性検査に要する手間は一般に非常に大きいと考えられる。

ストリーム並列実行とは、共有変数の値をあるゴールリテラルが生成し、別のゴールリテラルがその値を消費するというパイプライン的な実行により、AND関係にあるゴールリテラルを並列に実行する処理方法である。ゴールリテラル間における値の通り道のことをストリームという。ストリーム並列実行は共有変数の値の束縛を制限したAND並列実行の一種と考えることができる。

これまで、CP（Concurrent Prolog）[1]，GHC（Guarded Horn Clauses）[2]といったストリーム並列指向の論理型言語が提案されている。これらは言語のレベルで、ストリームを変数とリストにより実現しており、変数にコンスセルを代入することでストリームへの値の挿入と新しいストリーム用の変数の生成が行なわれる。この方法ではプログラム上にストリームの構造がはっきりと現われてしまい、抽象性の点から好ましくない。また値をひとつストリームに挿入のに、変数とリストをひとつづつ用いるため実行効率においても問題となる。さらに、これらの言語では、ガード・オペレータによりOR関係にある選択肢がひとつに絞られ、OR並列性は大きく制限されてしまう。

以上のような問題から、ここではストリーム型と呼ぶデータ型を導入し、OR並列性を制限せずにストリーム並列を効率よく実行できるように拡張した論理型言語を提案する。

2. 基本言語機能

ここで提案する言語におけるストリームは、AND／OR両関係にあるゴールリテラルからも読み書きすることができる。ストリームは単方向性であり、CPやGHCとは異なり不完全構造の読み書きによる双方向の通信はできない。双方向の通信を行なうためには、ふたつのストリームを明示的に用いればよい。

新たなストリームが生成することをストリームが開くと言い、ストリームへの書き込みが終わることをストリームが閉じると言うことにする。

① ストリーム型

ストリーム型には次の二種類がある。

＜S＞　　＞S＜

Sはストリームにつけられた名前（ID）であり、＜S＞はストリームの入口、＞S＜はストリームの出口を表わす。ストリーム型は入口同士、出口同士の単一化のみが許される。それ以外の単一化（例えば変数とストリーム型）は失敗する。

② ストリームへの挿入

＜S＞！X

ストリームSの入口にXを挿入する。Sの中身は副作用として書き換えられる。Sが既に閉じていれば、実行は失敗する。ストリームを閉じるためには、

＜S＞！　$eos　(end of stream)

を実行すればよい。

③ ストリームからの抽出

＞S＜？X

ストリームSの出口から要素をひとつ取り出して、Xに代入する。副作用として＞S＜は次の要素を指し示す。要素を取り出すことができない場合には実行は中断し、要素が書き込まれるまで待たされる。Sが既に閉じていれば、実行は失敗する。ストリームが閉じているかどうかを調べるには、

＞S＜？　$eos

を実行すればよい。ストリームからの抽出は節のヘッド内に表われてもよい。

④ 逐次AND実行と並列AND実行

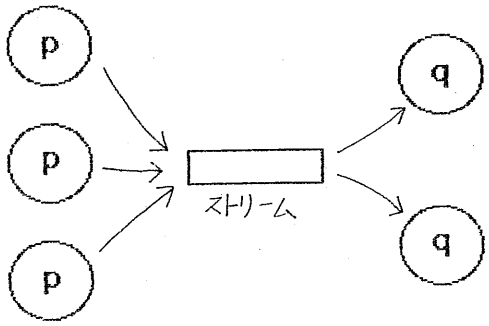
②、③に述べたようにストリームに対する操作は副作用として行なわれるので、この同期関係を制御するためにAND関係にあるゴールリテラルの実行を逐次と並列とに区別する。これはCPやGHCではデータの依存関係とガードにより同期を制御している点と大きく異なる。逐次実行は“，”、並列実行は“／”により表わす。

p，q　　pを実行した後にqを実行する

p／q　　p、qの実行順序は規定しない

並列実行ではストリーム並列しか行なわず、共有変数による通信はできない。ストリーム並列による効果を残す（変数を束縛する）ためには、3章で示すような特別な述語を用いなくてはならない。

ストリームの挿入・抽出の概念図を図1に示す。




```
..., p(<R>)/q(>R<),...

p(<S1>):-...          q(>T1<):-...
p(<S2>):-...          q(>T2<):-...
p(<S3>):-...
```

図1　ストリームの挿入・抽出

以上の機能をOR並列Prolog に付け加えて拡張した論理型言語SPL（Stream-Parallel Logic Programming Language ）を提案する。

3．ストリーム操作述語

ストリーム型に対する操作を実行する述語であり、“$”で始まる。

① $ltos（L，<S>）　(list to stream)
リストLの全要素をストリームSに挿入する。

② $stol（>S<，L）　(stream to list)
ltos の逆で、ストリームSの中身をリストLに変換する。

ストリーム並列により変数に値を束縛するときには、この述語を用いる。他のストリーム並列を行なう述語では、出力用変数を引数とすることはできない。

③ $concat（<S1>，<S2>）
ストリームS1が閉じていなければ、S1にS2を接続する。この実行によりS1は閉じる。

④ $ bag（X，p（X），<S>）
p（X）を満たすXについて全解収集を行ない、解が得られるたびにストリームSに挿入する。Prolog のbagof では、解の収集がすべて終了するまで次の述語を実行することはできないが、この述語は解の収集と他の述語の実行をストリーム並列に実行することができる。CPやGHCでは全解収集を行なうためにはプログラムを変換する必要があるが、本言語では単純な形式で全解収集を行なうことができる。

4．プログラム例

図2にエラトステネスの篩により素数を生成するプログラムを示す。Max以下の素数をストリームRに挿入する。素数をリストとして得たい場合には、

```
..., primes(Max,<R>)/$stol(>R<,L),...
```

とすればLが素数のリストとなる。

5．おわりに

OR並列性を制限することなく、ストリーム並列を効率よく実行できるように拡張した論理型言語SPLを提案した。SPLの処理系をC-prolog で記述し動作を確認した。ストリームに対する操作の言語仕様や並列推論マシン上への実装などについての検討が今後の課題である。

<参考文献>

[1] Shapiro, E. Y. : “A Subset of Concurrent Prolog and Its Interpreter”, TR-003, ICOT, 1983.

[2] K. Ueda : “Guarded Horn Clauses”, Proc. of Logic Programming Conference , ICOT, 1985.

```
primes(Max,<R>):-
        integer(2,Max,<S>)/shift(>S<,<R>).

integer(N,Max,<S>):-
        N < Max, <S> ! N, N1 is N + 1, integer(N1,Max,<S>).
integer(N,Max,<S>):-
        N >= Max, <S> ! $eos.

shift(>I< ? X,<J>):-
        <J> ! X, filter(X,>I<,<J1>)/shift(>J1<,<J>).
shift(>I< ? $eos,<J>):-
        <J> ! $eos.

filter(K,>I< ? X,<J>):-
        not(0 is X mod K), <J> ! X, filter(K,>I<,<J>).
filter(K,>I< ? X,<J>):-
        0 is X mod K, filter(K,>I<,<J>).
filter(K,>I< ? $eos,<J>):-
        <J> ! $eos.
```

図2　素数生成プログラム