データフロー方式Prologマシンにおける非決定的制御機構

伊藤 徳義　益田 昌直　清水 肇
（財）新世代コンピュータ技術開発機構

Ⅰ．はじめに

並列推論マシンは第５世代コンピュータプロジェクトの長期目標の１つとして位置づけられている．著者等はその１つのアプローチとしてデータフローモデルに基づいた並列Prologマシンの研究を進めている［２］，［３］．Prolog(Programming in Logic)はその基本機能として非決定性を内蔵している．従来の逐次型の処理系においては後戻り（Back-tracking ）制御を用いて，この非決定的探索を実現していた．並列型の処理系においては上記非決定的制御をより自然な形で実現できる．本稿では並列Prologマシンにおける非決定的制御機構について述べる．

Ⅱ．Prologにおける並列性

Prologの実行に際しては以下に示す３つのタイプの並列性が存在する．

－ＯＲ並列性
－ＡＮＤ並列性
－引数間の並列性

このうち，ＯＲ並列実行に関しては，後述するように非決定的ストリームを用いた制御を採用することによって比較的容易に実現できる．一方，ＡＮＤ並列実行に関しては共有変数を有するＡＮＤリテラル間の並列実行を実現しようとすると複雑な無矛盾性チェックのためのオペレーションを必要とするという問題が存在する．従って，共有変数を有するＡＮＤリテラル間は前記ストリームを介してパイプライン的に実行制御するという方式を採用した［３］．もう１つの引数間の並列実行は，リテラルの引数が構造を有するデータである場合，そのサブ構造の統一化(Unification) を並列実行することを意味している．以下，統一化のための基本プリミティブ，及び，統一化処理における非決定的制御のためのプリミティブについて説明する．

Ⅲ．統一化基本プリミティブ

引数間の並列性は，ゴールリテラル及びその定義節の頭部リテラルが与えられたとき，それらの引数がいずれも構造データであるときに，各々のサブ構造間の統一化の処理を並列実行することによって実現できる．ここで問題となるのはゴールリテラル内に同一変数が２度以上現れる場合である．このような変数を共有変数と呼ぶことにする．ゴールリテラルが構造データでありしかもその構造中に共有変数を含む場合，それぞれのサブ構造の統一化処理によって同一共有変数が同一のインスタンスに結合されることが保証されなければならない．

ｐ（Ｘ，Ｘ）のように共有変数を含むゴールリテラルが与えられたとき，ゴール呼出しを行う際ゴールリテラル内に共有変数が存在することを呼出された定義節側に知らせるためにshare オペレータを使用する．このオペレータはその入力オペランド内に未結合変数が含まれていた場合にはそれを共有変数に変更する．

これに対して呼出された定義節側においてはunify オペレータ及びconsistency-check オペレータを用いて引数のサブ構造間の並列実行を実現する．unify オペレータは２つの出力ポートを持っており，入力された２つの引数間の統一化が成功した場合はそれぞれのポートに２つの引数の共通インスタンスと共有変数に対する結合環境を出力し，そうでない場合はいずれのポートにも'fail'を出力する．結合環境は共有変数名及びそのインスタンスのペアを構成要素とするリストで表現され，ゴールリテラルの引数に共有変数を含まない場合は'nil' となる．結合環境は共有変数に対する結合に矛盾がないか否かを調べるconsistency-check オペレータに送られる．

consistency-check オペレータは入力された結合環境を調べ，同一共有変数に対するインスタンスの間の統一化処理を行った後，それらの共有変数に対する新たな結合環境を生成する．この結果は，最終的に，２つのリテラルの間の共通インスタンスに含まれる共有変数を置換するのに使用される．この処理はsubstituteオペレータによって行われる．なお，このオペレータはconsistency-check の結果が'fail'のときは，共通インスタンスとして'fail'を出力する．

これに対して定義節の頭部リテラル側に共有変数が現れる場合は単純である．この場合，各々の引数ごとの統一化によって得られたインスタンスをさらにunify オペレータによって統一化すればよい．この結果はその定義節に本体が存在すれば本体側に渡される．

Ⅳ．ストリームを用いた非決定的制御機構

ＯＲ並列実行における非決定性を実現するために，得られた解（共通インスタンス）の集合を第１図に示すようなストリームで表現する．ストリームはnon-strictな構造デ

ータとして表現され［１］，解の生成プロセスと消費プロセスの間の非同期通信手段を提供する．

このため，構造データを格納する各語には，その語の内容が有効であるか否かを示すR(ready)タグが用意されており，このR タグがオフである語はその語への書込み動作が行われていないことを意味している．このような語を未定義状態にあるといい，読出し命令が実行されるとその命令は，その語に対する書込みが行われるまで待ち状態となる．また後述するように構造データの記憶セル毎に参照数RCが付与されており，構造データの記憶管理のために使用すると共にストリームの管理にも使用する．

ストリームはHeadポインタとTailポインタで表される．Headポインタはストリーム本体がすでに存在すれば，その本体の先頭のセルを指している．そうでなければ未定義状態にある．これに対して，Tailポインタはストリーム本体の最後尾のセルを指している．ストリーム本体はリスト表現されており，そのセルはFirst 部とRest部を有する．First 部には解の１つが格納され，Rest部にはストリームの残りの部分に対するポインタが格納される．

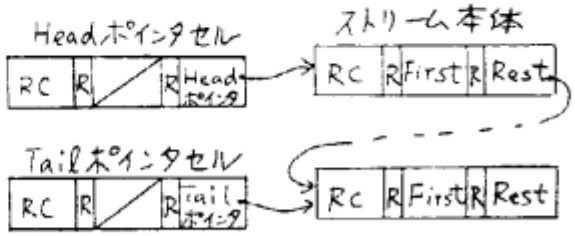
ストリームは第２図に示すように，create-stream ，及び，append-stream の２種類のオペレータによって生成される．create-stream オペレータは前述のHeadポインタ及びTailポインタを格納するための２つのセルを生成する．これらのポインタは第１図に示したように各セルのRest部に格納される．このうちHeadポインタは未定義状態に初期化され，TailポインタはHeadポインタセルのアドレスに初期化される．Headポインタセルのアドレスは解の集合を消費するプロセスへ送られる．これに対してTailポインタセルのアドレスはappend-stream オペレータへ送られ，ストリームを生成するＯＲプロセス間で共有される．

ストリームを生成する各ＯＲプロセスはその実行が終了するとappend-stream オペレータを起動する．append-streamオペレータは解の内容を調べそれが'fail'以外であれば，まず，新しいストリーム本体セルのための領域を割当てる（このセルのアドレスは新Tailポインタとなる）．次に，Tailポインタセルより旧Tailポインタの内容を読出すと同時に新Tailポインタに更新する（この間他のオペレータからの割込みは禁止される）．この後，旧Tailポインタの指すセルのRest部に新Tailポインタを格納し，新Tailポインタの指すセルのFirst 部に解の内容を書込む（このセルのRest部は未定義状態のままにしておく）．解が'fail'のときはappend-stream オペレータは実質的な処理を行わない．即ち，'fail'はストリーム中には現れない．

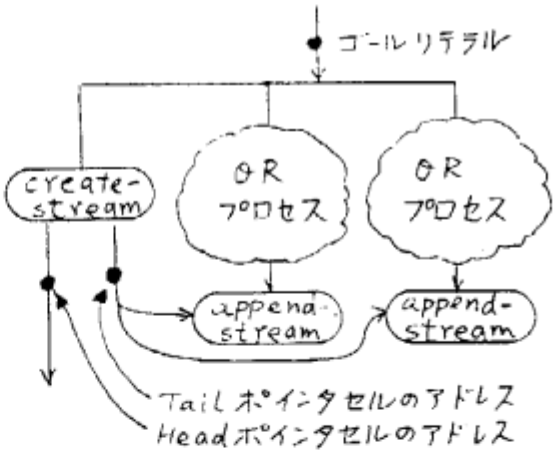
これらの処理の終了後，Tailポインタセルの参照数RCをデクリメントする．参照数は該セルを参照するポインタの数を表しており，この値がゼロに達したとき全てのＯＲプロセスが終了したことを意味する．このとき消費プロセス側に解が尽きたことを知らせるために，Tailポインタで指されるセルのRest部にend-of-stream を書く．

Ｖ．おわりに

データフローモデルに基づいた並列Prologマシン上で，Prologの非決定的制御機能が実現できることを示した．現在，ソフトウェアシミュレータによる上記マシンの評価を行っており［４］，今後さらに詳細なマシンアーキテクチャの検討を進めていく予定である．最後に，本研究の機会を与えていただいた１ＣＯＴ渕研究所長及び日頃有益な助言をいただく村上第１研究室長他の各位に深謝する．



第１図　ストリームの内部表現



第２図　ストリームの生成

＜参考文献＞

［１］ Arvind, et al."An Asynchronous Programming Language and Computing Machine",TR114a,ICS,Univ. of California, Irvine, Dec. 1978.

［２］伊藤　他，"データフロー機構に基づくPrologマシン" 情報処理第２６回全国大会，昭和５８年３月．

［３］ N.Ito, et al."Parallel Prolog Machine Based on the Data Flow Model",Technical Report,ICOT, Aug.1983.

［４］益田　他，"データフロー方式Prologマシンのシミュレーションによる評価"，本予稿集，４Ｐ－４．