知識工学と人工知能 53 - 2
(1 9 8 7. 7. 1 0)

並行論理型言語の実行における分散型ユニフィケーション

結縁祥治*・山崎進**・堂下修司**
吉田幹***

* 名古屋大学工学部、** 京都大学工学部、*** 日本ＩＢＭ

ホーン節集合における導出の過程を計算とみなす論理型言語の非決定的な計算機構に並行処理を導入した並行論理型言語が提案されている。本稿では、並行論理型言語の実行モデルが持つ並列分散性に着目し、導出法に基づく正負リテラルの消去の際に行われるユニフィケーションを分散実行することにより、並行処理を実現する並行論理プログラムの実行意味を示す。これを分散型ユニフィケーションと呼ぶ。並行論理型言語では、論理を基礎とした計算の枠組みの中に並行処理の機構が埋め込まれ、この並行処理の同期をとるための機能として、変数束縛の制限を表す標記（annotation）とガード機構とが導入されている。これらは、分散型ユニフィケーションにおいて個々のユニフィケーションの同期制御として扱われる。この分散型ユニフィケーションに基づく処理系は、同期通信を行いながら計算を進めていく分散処理システムとして、入力プログラムを実行する。ここでの分散処理システムにおけるユニフィケーションの同期制御をoccam 風の言語によって具体的に記述し、並行論理型言語の実行モデルにおける分散処理を明らかにする。

Distributed Unification on Execution of Concurrent Logic Programming Languages

Shoji YUEN*, Susumu YAMASAKI**, Shuji DOSHITA**,
and Mikio YOSHIDA***

* Faculty of Engineering, Nagoya University
Furo-cho, Chikusa-ku, Nagoya-shi, Aichi, 464 Japan

** Faculty of Engineering, Kyoto University

*** IBM Japan Ltd., Tokyo Research Laboratory

Concurrent logic programming languages have been proposed, where concurrency is introduced as the implementation of nondeterminism in a logic program. We propose an execution semantics called 'distributed unification' which implements concurrency by distributing unifications, with respect to the model of the languages. As the synchronization mechanisms for controlling concurrency, the annotations for variables are introduced to restrict binding variables and create suspending-states in unification and the guard-mechanism to select one out of OR-selections. These mechanisms are treated as synchronization of each unification. A system based upon the distributed unification executes an input logic program as a distributed communicating system. We describe its synchronization control with a 'occam'-like language to explicate distributed processing in the execution model of a concurrent logic programming language.

1. はじめに

抽象度の高いプログラミング言語の直接実行という目的において、一階述語論理のサブセットであるホーン節集合における導出の過程を計算とみなす論理型言語処理系の研究が最近盛んに行われている。論理型言語のプログラム（以下、論理プログラムという）の実行において特徴的なのは、論理の枠組みにおける導出の過程で、非決定性（nondeterminism）が埋め込まれていることである。このため、論理プログラムでは、その非決定的な実行手続きをプログラム中で陽に表現することなく、その解を求めることができる。

論理型言語の処理系は、非決定的な処理を決定的な計算機の上で実行するための機構を備えている。Concurrent Prolog[2]やGHC[3]をはじめとする並行論理型言語処理系では、非決定的な処理に対して並行処理を導入し、同一節内の負リテラルに対する導出法の同時適用と、ガード機構とによってこれを実現している。並行論理プログラムでは、並行実行の意味は陽に表現されず、論理の枠組みにおける導出法の処理の中に埋め込まれている。しかし、これだけでは並行プログラミング言語としては不十分で、同期制御とそれに伴う通信のための基本的な機能が必要となる[3]。この機能として、導出法のユニフィケーションの際に変数束縛を制限することによって正負リテラルの消去に対する同期制御が導入されている。このとき、変数に対する代入の適用が通信とみなされる。

本稿では、並行論理型言語の実行モデルが並列分散処理の意味を持つことに着目して、分散実行により並行処理を実現する実行意味として、分散型ユニフィケーションを提示する。ここでは、並行論理プログラムは、分散処理システムを記述していると見なす。導出法に分散処理が埋め込まれているという意味で、並行論理プログラムは抽象度の高いレベルで分散処理を記述しているといえる。

ここでの計算の領域として、ユニファイア木[4]を採用する。ユニファイア木は、プログラム実行中に現れる変数に対する代入を原理的に保持しているため、並行論理プログラムの実際の振舞いの表現に適した空間と言える。この木構造の与える空間において、並行論理プログラムの計算は、導出に使われた正負リテラルの変数に対する項の代入（substitution）の結合演算と見なすことができる。代入の結合演算の持つ性質から、この領域上では計算は部分的に分散して行っていくことができることが保証されている。

分散型ユニフィケーションにおける手続きは、代入を生成する手続きと、代入をシステムの持つ状態（これも代入の形で与えられる）に結合する手続きとに分けられる。いずれも、それぞれの節毎に分散された制御に基づいて、計算が進行するが、前者の計算手続きで扱うデータが局所的であるのに対し、後者の計算手続きではシステムの状態を規定している大域的なデータを扱う。これは、論理プログラムの処理において、一回のユニフィケーションで変数が完全にインスタンシェートされないという性質によっている。従って、ユニフィケーションが終了した後でも、変数に対してインスタンシェートが行われる可能性がある。このため、分散型ユニフィケーションに基づく分散処理では、強結合の分散処理システムを仮定している。

まず、並行論理プログラムにおいて、分散された制御のもとにユニフィケーションを行っていく分散型ユニフィケーションについて述べる。次に、分散型ユニフィケーションに基づく処理系によって、入力プログラムから実現される分散処理システムとして、入力節に対応する計算単位が相互に同期通信を行いながら計算を進めるシステムを示す。

2. 並行論理型言語

以下の議論において節、リテラル、アトム、項、変数などの基本的な定義は、文献[1]のそれらに基づくものとする。

対象とする並行論理型言語は、Horn節集合に次のような拡張を加えたものとする。

1) 入力確定節はガードを持つ次のような節である。

H :- G1,...,Gm | B1,...,Bn (m≧0,n≧0)

ここで、H,G1,...,Gm,B1,...,Bnはアトムであり、合意記号 :- の左辺のアトムを正リテラル、右辺のアトムを負リテラルと呼ぶ。Hを<u>ヘッド</u>、G1,..,Gmを<u>ガード</u>、B1,..,Bnを<u>ボディ</u>と呼ぶ。特に、m=0のときは<u>コミットオペレータ</u> | を省略する。さらに、m =n=0でアサーション（assertion）を表し、このとき、合意記号も省略する。コミットオペレータの実行意味はConcurrent Prolog のそれと同じである[2]。

2) ゴール節は、負リテラルのみから構成され、ガードを持たない。

:- B1,...,Bn

3) 述語引数は全てモードづけがされており、ユニフィケーションの際に、述語引数の項に含まれる変数に対して行われる束縛に関して制限を与えている。それらのモードを表示するキャラクタとその実行意味は次の３種類である。

i) ?（入力モード）

負リテラルの入力モードづけされた述語引数の項に含まれる変数は、ユニフィケーションの際に変数以外の項とは束縛されない。

ii) !（出力モード）

正リテラルの出力モードづけされた述語引数の項に含まれる変数は、ユニフィケーションの際に変数以外の項とは束縛されない。

iii)*（双方向モード）

この項に含まれる変数に関して束縛制限はない。■

これらのモードは、次のようなモード宣言によって、プログラム中の同じ述語記号を持つ全てのアトムに対して宣言する。

mode <述語記号>(<モードを表すキャラクタ(?,!,*)のリスト>)

4) 副作用をもつ組み込み述語が存在する。

１個の並行論理プログラムは、０個以上のガード付き確定節、１個のゴール節、そして、これらに含まれるすべての述語記号に対する１通りのモード宣言とからなる。

対象とする並行論理型言語をこのようなものとした理由は、一般の論理プログラムに対して、並行論理プログラムが持つ本質的な違いは、上記のようなシンタックスとその実行意味によって表現できると考えたためである。実際、これはConcurrent PrologやGHCにおいて、それぞれの述語について、変数に対する束縛制限を陽に表現したものとなっている。ただし、言語仕様としての安全性は保証されていない。例えば、つぎのようなプログラムは、変数 Xがインスタンシェートされないので永久に待機状態となる。

```
mode p(?).
```

```
:-p(X).
p(a).
```

ここでは、与えられた並行論理プログラムの振舞いについてのみ問題とし、そのプログラムが正しいプログラムであるかどうかについては問題としない。また、実際の処理系の実現性の関連から、実現されているそれぞれの言語によってガード機構において変数についての束縛条件が異なるが、ここでは、それが並行論理型言語を特徴づける要素であるとは考えていない。

3．分散型ユニフィケーション

本節では、まず、並行論理プログラムの実行を分散されたユニフィケーションとして説明し、この分散型ユニフィケーションをユニファイア木[4] の与える空間で意味づけする。

3．1 論理プログラムの実行における並列性

一般に論理プログラムはその実行意味において、3種類の並列性を持つことが知られている[7]。 並行論理プログラムでは、以下に示すような意味となる。

(1) 述語引数間の並列性

導出法による正負リテラルの消去において、p(t1,...,tm)を正リテラル、n(s1,...,sm)を負リテラルとするとき、 (s1,t1),...,(sm,tm)のユニフィケーションを行う際の並列性。（ここで、p、nは述語記号、t1,..,tm、s1,..,smは、項である。）

(2) 負リテラルの消去における並列性（AND並列）

導出法の適用において、一つの節

H :- G1,...,Gm | B1,...,Bn (m≧0,n≧0)

に含まれる負リテラルのリスト

G1,...,Gm または B1,...,Bn

の消去を行う際の並列性。

(3) 正リテラルの消去における並列性（OR並列）

一個の負リテラルとユニファイできる正リテラルの消去

H :- ...,N,... （Nは、ガードまたはボディに含まれる負リテラル）

```
┌──────┐
├ H1 :- G11,...,G1m1 | B11,...,B1n.
|          . . .
└ Hk :- Gk1,...,Gkmk | Bk1,...,Bknk
```

を行う際の並列性。

3．2 モード付きユニフィケーション

ユニフィケーションは項のペアに対する操作である。ここでの並行論理プログラムでは、これら項のすべてに対してモードづけがされている。宣言の方法から、ユニフィケーションは次の3種類となる。ここで、tnを負リテラルに含まれている項、tpを正リテラルに含まれている項とする。

(1) 入力モードとして宣言されている述語引数

tpに含まれている変数に対し、tnに含まれている項が束縛される。tpの変数のうち、束縛されないものがあれば待機する。このとき、(tn→tp)と書く。

(2) 出力モードとして宣言されている述語引数

tnに含まれている変数に対し、tpに含まれている項が束縛される。tnの変数のうち、束縛されないものがあれば待機する。このとき、(tp→tn)と書く。

(3) 双方向モードとして宣言されている述語引数

待機状態は存在せず、通常のユニフィケーションを行う。このとき、(tn↔tp)と書く。

ここにおいて、モード付きユニフィケーションは、成功終了、失敗終了、および待機状態の3種類の状態をとる。待機状態は、変数に対するインスタンシエーションが行われることにより、成功終了、失敗終了、待機状態のうちのいずれかの状態に遷移する。

＜例＞

ここで、f,a,bを関数記号、Xを変数とする。

(f(a)→f(X)) 成功終了 ユニファイア{a/X}

(f(X)↔f(a)) 成功終了 ユニファイア{a/X}

(f(a)→f(b)) 失敗終了

(f(X)→f(a)) 待機状態

この待機状態はX が aにインスタンシエートされることで成功終了し、そのユニファイアはε（空代入）となる。

3．3 リテラルの消去手続き

前節で述べたモード付きユニフィケーションに基づいて、基本的計算メカニズムである導出法におけるリテラルの消去ほ、次のように定義される。

正負リテラルのペア(p(t1,..,tm),n(s1,...,sm))の導出法による消去は、項のペア (t1,s1),...,(tm,sm)に対するユニフィケーションがすべて成功終了したときに行われる。

3．4 モード付きユニフィケーションの分散制御

一つの節は、その節が持つ述語引数に関するユニフィケーションを制御する機構を持つ。各節は、他の節とユニフィケーションによる相互作用を行いながら、計算を進めていく。並行論理プログラムの実行において、この同期制御のための手続きは、プログラムの持つ並列性と対応して次のような3種類となる。

(1) 正負リテラルの消去操作

導出に用いる正負リテラルに対し、リテラルの消去手続きを起動する。このとき、正リテラルの属する節に対する消去操作を起動する。この節がコミットされたとき、正負リテラルのペアの消去操作は成功終了して、この正負リテラルは、消去される。

(2) 節の消去操作

最初に、ガード内の各々の負リテラルに対し、(1) の消去手続きを起動する。ガード内の負リテラルが全て消去されたとき、この節はコミットされたという。コミットされた節は、ボディの各々のリテラルに対し、消去手続きを起動する。そして、ガードとボディ内の全てのリテラルが消去されたとき、節の消去操作は成功終了する。

(3) ガード機構による節の組合せの選択

ある負リテラルと消去できる正リテラルが複数個存在するとき、ガード及びボディの負リテラルの消去手続きの組合せは複数になる。このとき、節のガードおよびボディの消去は、一番目にすべてコミットされた組合せを一通りだけ選択し、他の選択の計算は中止する。一通りの組合せが選択されたとき、ガード機構による選択は成功終了する。

(1)および(2)のメカニズムは明らかに、個々の節に対して分散させることができる。(3) のメカニズムも、図3.1のように各節間に親子関係があるものとし、その選択は、親が行うものであるとすることによって、個々の節に対して分散させることができる。この性質は、ガード機構によってORの選択をただ一つに限定することに基

づいている。したがって、一般のHorn節集合のプログラムで完全性を保証するような実行では、ORの管理はプログラム全体を見渡す（すなわち、環境をコピーする）グローバルな手続きが必要となってくる。（図 3.2）この点、並行論理プログラムでは、個々の節における無矛盾性をチェックする手続きがあればよく、そのようなグローバルな手続きは必要ない。

ここでのORの管理はローカルに行われ、その中でコミットされた選択のみが外部の環境に公開される。この環境を次に述べるユニファイア木の与える空間における代入として定義する。

３．５　ユニファイア木[4] に基づく意味

ユニファイア木は、論理プログラムの計算過程を表現した木構造である。この木構造は、 AND木に対応し、入力節に対応するノードと導出法による正負リテラルの消去に対応するアークとから構成されている。ノードには、正負リテラルのmgu、 アークには正負リテラル間で共通した変数が存在しないように入力節の変数の名前替えを行うリネーミング代入が割り当てられる。これらの代入を、その要素のユニファイア因子と呼ぶ。（図 3.3）木構造自身の持つユニファイア因子は、それを構成する要素のユニファイア因子の結合[4] によって規定される。未定義でないユニファイア因子を持つ木を正則であるといい、正則なユニファイア木は正当な計算過程を表現していることが示されている。

並行論理プログラムの実行はユニファイア木の与える空間では、図 3.4のように、木構造を根の方から、並行して構成していくことである。ガード機構によって、構成される木の数は１個に限定される。木構造のユニファイア因子が結合で定義されていることから、この木の構成は分散した計算によって得られた部分的な結果をあとで結合していって構わない。このことから、この木構造によって、分散型ユニフィケーションの振舞いを表現することができる。

３．６　代入の結合演算における同期制御

ユニファイア木に基づく実行は、ユニファイア木を構成しながら、その木構造に対して定義されたユニファイア因子を計算していくことである。ここで、説明のため次のような代入の２項演算である結合演算[4]を導入する。

２個の代入σとλが無矛盾で、σとλの結合をθとするとき、σ＋λ＝θと書く。σとλが矛盾するとき、σ＋λは、特別な代入Υ（矛盾代入と呼ぶ）とし、σ＋λ＝Υとなる。ここで、任意の代入σに対し、σ＋Υ＝Υ＋σ＝Υ。

特に、任意の代入σに対し、ε＋σ＝σ＋ε＝σ（εは空代入）。また、結合演算では結合則が成立するので、３個以上の代入の結合演算は括弧を省略する。

並行論理プログラムの実行におけるユニフィケーションは、その実行により述語引数のモードに従って、成功終了、失敗終了、および、待機状態のいずれかに遷移する。項の間でユニフィケーションを行うことは、代入の結合演算を行うことにつぎのように対応する。

正負リテラル間のモードづけされた項のペアのユニフィケーション（t1→t2）について、t1'、t2'をそれぞれt1、t2に対応して入力節に出現する項であるとする。ここに、t1=t1'θ1、t2=t2'θ2となるような無矛盾な代入

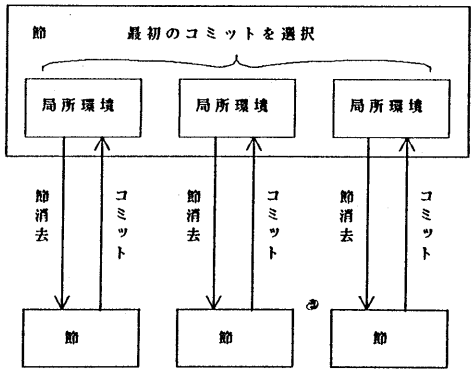

図３．１　コミットによる節の選択

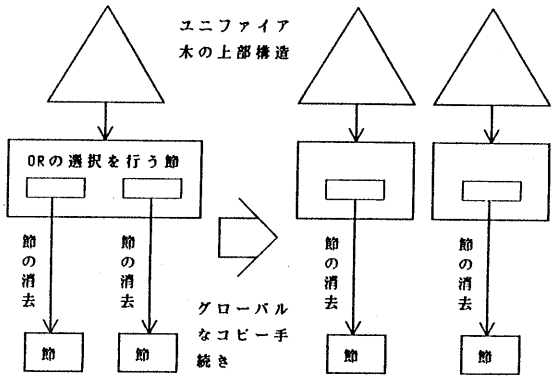

図３．２　環境のコピーを行うORの選択

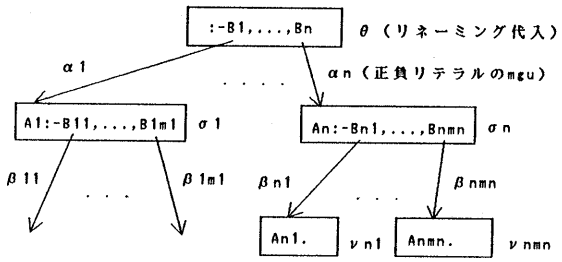

図３．３　ユニファイア木

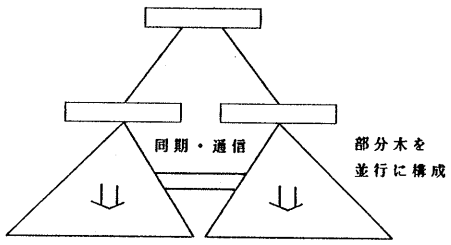

図３．４　ユニファイア木上での並行論理プログラムの実行

θ1、θ2が存在して、mgu(t1,t2)=(θ1＋θ2＋mgu(t1',t2'))↾(var(t1)∪var(t2))。（var(t)は項tに出現する