5M−8

制約論理型言語のＯＲ並列実行

車谷　博之　　広瀬　正
(株)日立製作所　システム開発研究所

1．はじめに

現在、制約論理プログラミング(ＣＬＰ,Constraint Logic Programming)の研究が、盛んに行われている。我々は、ＣＬＰの持つ制約機能によりPrologより宣言的な記述が可能となることに注目しているが、制約解消処理はProlog処理に比べて多大の計算時間を必要とする。一方、我々は、PrologのＯＲ並列処理系の研究も行っているが[1][2]、ＣＬＰのＯＲ並列実行には、Prologと同様、①全解探索型問題：並列実行による高速化効果と、②単解探索型問題：横型探索時、最初に求まった解を採用した時点で他の探索パスを中止し、処理時間を削減する効果、を期待できる(図１)。そこでＣＬＰのＯＲ並列処理を試みた。この検討に際して、以下の方針を立てた。

(１)推論エンジンのＯＲ並列実行：ＣＬＰの並列実行には、制約解消処理の並列実行と推論エンジンの並列実行がある。文献3)では、有限集合を対象とした制約解消のＯＲ並列処理を行う研究報告がある。本処理系では、先ず推論エンジンの並列実行を試行する。

(２)制約解消処理とＯＲ並列処理の独立した実現：独立に実現することにより、簡潔に処理系を作成する。

(３)従来Prolog処理系の拡張機能として実現:従来処理系の上位互換性の保持、従来処理系からの移行の容易化のため、従来Prolog処理系の拡張機能として、制約機能とＯＲ並列機能を実現する。

これらの方針を満たすため、制約解消処理機能やＯＲ並列実行制御機能をProlog処理系の組込述語(それぞれ、制約述語、並列述語という)として実現する方法を採った。また、ＣＬＰプログラムを解析して制約述語と並列述語を挿入し、Prologプログラムに変換するプリプロセッサを実現する。

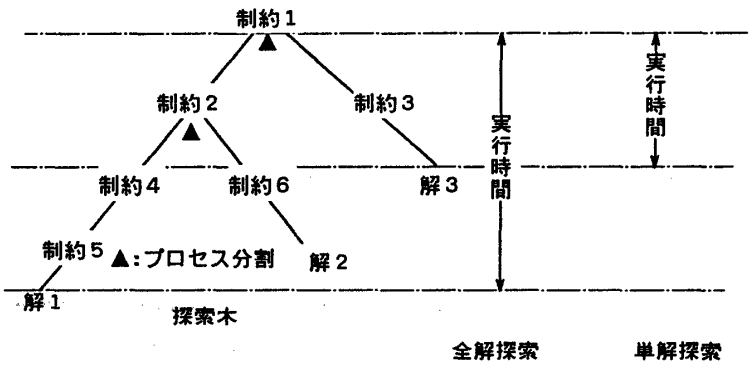

図１．ＣＬＰのＯＲ並列実行効果

2．並列述語

(１)divide述語

プロセスを生成する。

本述語実行プロセスは「divide≡fail」と解釈し、実行を続ける。

生成されたプロセスは「divide≡'!'」と解釈し、実行を続ける。

プロセスはdivideにバックトラックした時点で消滅する。

(２)dfg述語

本述語はプロセスの深さ優先探索の順序(Prolog逐次実行順)を回復し、従来Prolog処理系との互換性を確保するために用いる。

深さ優先順で前のプロセスすべての消滅を待つ。この待合せにより深さ優先探索の順序を回復する。

3．制約述語

数値(分数で表現する有理数、浮動小数点数)の線形等式／不等式に関する制約を実現する。

(１)clp_unify(X,Y,Env):変数X,Yの制約解消を行う。Envは、伝播する制約リストである。

(２)clp_le(X,Y,Env):不等式X≦Yの制約解消を行う。

(３)clp_lt(X,Y,Env):不等式X＜Yの制約解消を行う。

(４)clp_check(Env):制約リストEnvの制約解消を行う。

4．制約述語の実現

(１)等式制約解消方法

ガウスの消去法を次のように用いる。

ステップ１(前進消去)：各制約実行時点で、Expに既に解かれている変数を代入する。そして、方程式を次の形式に変形する。

$f(X)$：　$A_1X_1+A_2X_2+\cdots+A_nX_n=0$　　A_i:定数、X_i:変数。

既に解かれていない変数に注目し、この変数について解き　X = Expとする。

ステップ２(後退代入)：Expが定数式の場合評価し、Xに評価値をユニファイする。既に解かれている変数にこの変数値を代入し、定数式になるか否かを判定する。次にステップ１に行く。

OR-Parallel Execution of Constraint Logic Programming
Hiroyuki KURUMATANI, Tadashi HIROSE
Systems Dev. Lab. Hitachi Ltd.

表１．不等式変数消去条件

項	A_i:0	X_i:Exp_i	f[Exp_i]:0
1	$A_i>0$	$X_i \geqq Exp_i$	f[Exp_i]≦0
2	$A_i<0$	$X_i \leqq Exp_i$	f[Exp_i]≦0
3	$A_i>0$	$X_i \leqq Exp_i$	f[Exp_i]≧0
4	$A_i<0$	$X_i \geqq Exp_i$	f[Exp_i]≧0
5	$A_i>0$	$X_i > Exp_i$	f[Exp_i]<0
6	$A_i<0$	$X_i < Exp_i$	f[Exp_i]<0
7	$A_i>0$	$X_i < Exp_i$	f[Exp_i]>0
8	$A_i<0$	$X_i > Exp_i$	f[Exp_i]>0

(２)不等式制約解消方法

等式解消と協調して、線形不等式を解消する。不等号$\alpha \leqq \beta$の制約解消方法(clp_le述語の実現)は次のように行う。

ステップ１：各制約実行時点で、既に等式制約により解かれている変数を制約式に代入し、次の形式に変形する。

等式　　f(X)：　$A_1X_1+A_2X_2+\cdots+A_nX_n=0$

不等式(≦)　f(X)：　$A_1X_1+A_2X_2+\cdots+A_nX_n \leqq 0$

ステップ２：f(X)に現れる各変数について、既存の不等式制約で変数消去可能なものを解析し消去する。

変数消去可能条件を表１に示す。f[Exp_i]は、f(X)のX_iにExp_iを代入した式である。１項は、$A_i>0$かつ$X_i \geqq Exp_i$の場合、f[Exp_i]≦0が成立することを示している。また、５項は、$A_i>0$かつ$X_i>Exp_i$の場合、f[Exp_i]<0が成立することを示している。(ただし、≦ついては、１，２，５，６項)

f[Exp_i]≦0が成立する表１の１，２項の場合、これらの不等式の各変数に注目して、X≦ExpまたはX≧Expに変形する。また、f[Exp_i]<0が成立する表１の５，６項の場合、これらの不等式の各変数に注目して、X<ExpまたはX>Expに変形する。

ステップ３：X≦Exp、X≧Exp、X<Exp、X>Expにおいて、Expが定数式の場合評価し、これを$Const_i$とする。同一Xについて、境界値の無矛盾性を判定する。判定条件及びその処理を表２に示す。

表２．変数境界無矛盾性判定表

項	条件	処理
1	Const1≦X, X≦Const2, Const2<Const1	「fail」を実行しclp_le述語を失敗させる。
2	Const1<X, X≦Const2, Const2≦Const1	
3	Const1≦X, X<Const2, Const2≦Const1	
4	Const1<X, X<Const2, Const2≦Const1	
5	Const1≦X, X≦Const2, Const1＝Const2	「X=Const2」を実行する。
6	以外	「true」を実行し、次にステップ１に行く。

５．プリプロセッサ

先ず、ＣＬＰプログラムをPrologプログラムに変換する。ＣＬＰでは、Prologのヘッドユニフィケーション、'='、'=<'、'>='、'<'、'>'を制約実行と解釈するので、これを制約述語に変換しPrologプログラムに変換することによりＣＬＰプログラムを実行する。

(a)ヘッドユニフィケーションの変換

「p(X,Y):-….」は「p(α,β,Env):-clp_unify(α,X,Env),clp_unify(β,Y,Env),….」に変換する。α、βは変数とする。ただし、変数パラメタについてはclp_unifyには変換しない。

(b)述語'='の変換

「X=Y」は、「clp_unify(X,Y,Env)」に変換する。

(c)不等式述語'=<'、'>='、'<'、'>'の変換

X=<Y、Y>=Xはclp_le(X,Y,Env)に変換する。X<Y、Y>Xはclp_lt(X,Y,Env)に変換する。

(d)式評価代入述語'is'の変換

「X is Exp」は、「X is Exp,clp_check(Env)」に変換する。評価結果の変数Xへの代入による制約解消を行うため、制約述語clp_checkを呼ぶ。同様に、変数へユニファイが生じる部分の直後に、clp_checkの呼出しを挿入する。

次に、文献2)で述べた方法を用いて、変換したPrologプログラムをＯＲ並列実行するプログラムに変換する。

６．おわりに

制約解消機能とＯＲ並列実行制御機能をPrologの組込述語として実現し、ＣＬＰプログラムを解析して制約述語と並列述語を挿入しPrologプログラムに変換するプリプロセッサを実現することによって、従来Prolog処理系の拡張機能としてＣＬＰのＯＲ並列実行処理系を実現する方法を述べた。

参考文献

1)広瀬、車谷、「並列化Prolog処理系ＬＯＮＬＩ＋、基本設計方針」、第３９回情報処理学会全国大会、pp.1336-1337.

2)車谷、広瀬、「並列化Prolog処理系ＬＯＮＬＩ＋、並列化変換」、第３９回情報処理学会全国大会pp.1338-1339.

3)P.Van Hentenryck,"Parallel Constraint Satisfaction in Logic Programming:Preliminary Results of CHIP within PEPSys",ICLP-'89, pp.165-180.