Prologプログラムの AND-OR 並列実行モデル

6D-7

富樫 敦 †, 田沼 均 ††, 野口 正一 †
(†:東北大学 電気通信研究所) (††:電子技術総合研究所)

1. はじめに

Prologプログラムの実行は, AND-OR木の探索と対応付けられ, 一般には, AND並列性, OR並列性, 並列ユニフィケーションの3つのタイプの並列性が存在する[1]. 本論文では, Prologプログラムを並列に処理するAND-OR並列実行モデルを提案する. 本モデルは, Prologプログラムに内在する並列性を最大限に抽出し, 並列に処理するモデルである.

2. AND-OR 並列実行モデル

本節では, Prologプログラムを並列に実行するモデルについて述べる. モデル化するためにプロセスを用いる.

2.1 センタープロセス

センタープロセスは, 実行モデルにおいて中心となるプロセスである. 1つの実行環境に対して1つのセンタープロセスが割り当てられ, センタープロセス同志は相互に不干渉である. センタープロセスの役割は, プログラムの実行過程の環境を保持することである. 環境は,

$$ENV = <G, E, \Gamma>$$

の3項組である. ここで,

a. $G = \{L_1, \ldots, L_k\}$は, これから解く, または現在解いているゴール, 各L_iは, ゴールリテラルを表す;

b. Eは, 無矛盾性チェックを通過した現段階までに確定している変数への束縛環境;

c. $\Gamma = \{\alpha_1, \ldots, \alpha_k\}$は, 各サーチプロセス$SP_i$によって探索された結果$\alpha_i$の集合であり, α_iはゴールリテラルL_iと統一化可能な節C_{ij}と統一化の際生じる変数への束縛環境E_{ij}の対の集合.

$$\{<C_{i1}, E_{i1}>, \ldots, <C_{im_i}, E_{im_i}>\}$$

から成る. 変数への束縛環境は, 左辺を変数に限定した等式の集合$\{x_1 = t_1, \ldots, x_n = t_n\}$である. 場合によっては, サーチプロセス$SP_i$の進行が遅れて$\alpha_i$が無いときもある.

2.2 サーチプロセス

サーチプロセスは, 1つのゴールリテラルと統一化可能な節を探索するためのプロセスである. 探索すべきゴールが発生すると, 1つのゴールリテラルに対して1つのサーチプロセスがセンタープロセスによって生成される.

本モデルでは, サーチプロセスによってAND並列性を抽出し, 共有変数に関する無矛盾性の保存はセンタープロセスが受け持つ.

2.3 コミュニケーションプロセス

本モデルでは, センタープロセスが全てのサーチプロセスからの報告を待たないで, 報告が適当に溜ったら来た分だけて無矛盾性チェックを行い, 残ったサーチプロセスの分については次回に回す. 無矛盾性チェック時に新たな環境が生成され. 複数の新しいセンタープロセスが生成される場合があるが, そのとき残されたサーチプロセスと分離されたセンタープロセスを結ぶ機構が必要となる. その役割を受け持つのがコミュニケーションプロセスである.

コミュニケーションプロセスは, センタープロセスが新たなセンタープロセスに分離されたときに, 親のセンタープロセスを引き継いで生成される. 以降, このセンタープロセスに向けて来たサーチプロセスからの報告を取り次ぎ, 子のセンタープロセスに送る.

2.4 実行動作

プログラムの実行過程を次に示す. G_0を最初に与えられたゴールとする. 初期設定として, G_0に対応するセンタープロセスCP_0を生成し, その環境を

$$<G_0, \{\}, \{\}>$$

とする.

1. 環境として$<\{L_1, \ldots, L_k\}, E, \Gamma>$を持つセンタープロセスが, k個のサーチプロセス$SP_1, \ldots, SP_k$を生成, 起動する.

2. 起動された各サーチプロセスSP_iは, それぞれ独立かつ並列に対応するゴールリテラルL_iと統一化可能な節を全て捜し出し, センタープロセスにその結果を報告する. 報告結果α_iは, L_iと統一化可能な節C_{ij}と統一化時の変数への束縛環境E_{ij}の対の集合である.

$$\alpha_i = \{<C_{i1}, E_{i1}>, \ldots, <C_{im_i}, E_{im_i}>\}$$

統一化可能な節が無い場合は, そのことをセンタープロセスに報告し, センタープロセスは, 直ちに子の

An AND-OR Parallel Execution Model of Prolog Programs
Atsushi TOGASHI †, Hitoshi TANUMA ††, Shoichi NOGUCHI †
†: Tohoku University ††: Electrotechnical Laboratory

サーチプロセスを殺して自分自身も死滅する.

3. センタープロセスは, サーチプロセスからの報告を受けて無矛盾性チェックを行う. この時点でのセンタープロセスの環境を $<G, E, \{\alpha_1, \ldots, \alpha_n\}>$ と置く. ここで,

$$\alpha_i = \{<C_{i1}, E_{i1}>, \ldots, <C_{im_i}, E_{im_i}>\}.$$

センタープロセスは, サーチプロセスからの報告を直積展開して, $m_1 \times m_2 \times \cdots \times m_k$ 個の独立な束縛環境

$$E_{11} \cup E_{21} \cup \cdots \cup E_1, \ldots, E_{1m_1} \cup E_{2m_2} \cup \cdots \cup E_{km_k}$$

を作り出す. 各 $E_{1i_1} \cup E_{2i_2} \cup \cdots \cup E_{ki_k}$ に束縛環境 E を加えて, それぞれについて無矛盾性チェックを行う.

4. 無矛盾性チェックを通った環境は, 新たなセンタープロセスとして分離, 生成される. 新しいセンタープロセスの環境は,

$$<\{B_1, \ldots, B_n\}, E', \{\}>.$$

ここで, $B_1, \ldots, B_n$ はサーチプロセスが探してきた節 $A :- B_1, \ldots, B_n$ の本体, E' は無矛盾性チェック後に生成された束縛環境である.

無矛盾性チェックを通った環境を全て分離し終わったら, 古いセンタープロセスはコミュニケーションプロセスに変化する. まだ報告の終わっていないサーチプロセスは, このコミュニケーションプロセスを使って新たに生成されたセンタープロセスに結果を報告する.

5. センタープロセスの環境が $<\{\}, E, \Gamma>$ のように表される場合, 即ちそのゴールリテラルがもうない場合は "*success*" で実行を終了する. その他の場合は, 1. からの動作を繰り返す.

このように, 初期設定をした後, 1.~5. の動作を繰り返してプログラムを実行する. 全てのセンタープロセスが死滅, 又は *success* で実行を終了したらプログラムの実行が終了する.

3. シミュレーションと評価結果

本節では, モデルの有効性を調べるために行ったシミュレーションとその評価結果について述べる. 測定項目は, 解が求まるまでの時間と実行中のセンタープロセスの数, 及び全環境数である. シミュレーションを行うに当たり次の仮定を設ける.

a. メモリ管理に関するオーバーヘッドは考えない.

b. プロセスの発生, 死滅, 物理的なプロセッサへの割り付けのための時間の経過, プロセス間通信のためのオーバーヘッドなどは考えない.

c. 評価に使う時間の単位としては, 基本的な統一化にかかる時間を, 即ち文字アトム同志を比較するのに要する時間を単位とする.

d. 無矛盾性チェックは, 全てのサーチプロセスからの報告が終了した時点で行う.

N-Queen についてシミュレーションを行ってみた. OR並列のみの場合と比較して, AND-ORの両並列性を導入すると, *4-Queen* で 2.86 倍, *5-Queen* で 3.1~3.5 倍, *6-Queen* で 3.6~3.9倍の速度向上がみられる. 一般に, *N-Queen* では N が大きくなる程解の探索空間が大きくなるから, 問題が複雑になる程AND並列による速度向上の効果が現れてくるといえる.

N-Queen の他に, ソート, 道順探索, 素数探索, 構文解析についてシミュレーションを行ったが, それぞれのプログラムによりさまざまではあるが, 他のプログラムについても少なからぬ速度の向上が認められた. 同じプログラムの中では, より複雑なもの程AND並列を入れた際の速度向上の効果が大きく, さらに複雑な実用的なプログラムについてはより一層の効果が期待できる.

ただし, 速度の向上とは引き換えに, 本方式では爆発的な環境の増大が起こってしまう. しかしプログラムにもよるが, 多くの場合環境の爆発的な増大は一瞬であり, 極めて短時間で減少してしまう. これはAND並列でそれぞれのリテラルを同時に実行しているために, リテラル同志の干渉等により環境がfailしやすくなっているためである. 従って多くのプログラムでは, この爆発的な環境の増大はその増大した一瞬を何とか防ぐ対策, 例えば環境のためのバッファを大きくとるとかスケジューリングを工夫する等を行うことによって, 対処しうるものである.

4. むすび

本論文では, Prologプログラムを並列に実行するための計算機のモデルを提案し, ソフトウェアシミュレーションによりその有効性を確認した. 今後の課題としては, 更に詳細な設計が必要である. 例えばメモリの構造や管理などの詳細な設計を行う必要がある. その他に, 現在のモデルでは, 逐次型の統一化アルゴリズムを採用したが, 本モデルに合う並列統一化アルゴリズムを考える必要がある.

参考文献

(1) Conery,J.S. , "The AND/OR Process Model for Parallel Interpretation of Logic Programs", Univ. of California Technical Report, June,1983.

(2) Lloyd,W. J. , "Foundation of Logic Programming", Springer-Verlag,1984.

(3) 田沼, 富樫, 野口, PrologプログラムのAND-OR並列実行モデル, 記号処理, 37-2, 1986