Multi-PSI 上の並列 ATMS

中島 誠† 太田 好彦‡ 井上 克巳‡
†(財) 日本情報処理開発協会 ‡(財) 新世代コンピュータ技術開発機構

1 はじめに

従来提案されている ATMS (Assumption-based Truth Maintenance System) [1] の高速化は，効率的な知識システム実現のために重要である．近年，並列化によるいくつかの高速化 ATMS が提案されている [2, 3, 5, 6, 7]．これらの中にあってここで述べる疎結合型並列計算機 Multi-PSI 上に開発した並列 ATMS の特徴は，入力されるデータの数だけのプロセスを生成する並列アルゴリズムを採用している点にある．この方式により，入力データの数に対して生成するプロセス数の爆発を防ぐことができる．さらに，これらの生成されたプロセスを複数のプロセッサへ割り当てることが，データの分配の問題に帰着されるため，比較的簡単に行なえると考えられる．以下，本システムについて述べる．

2 ATMS

ATMS[1] は，仮説集合と命題論理のホーン節 (理由付け (*Justification*) と呼ぶ) の集合を入力とし，アトム (データ (*Datum*) と呼ぶ) の真偽の状態 (あるいは信念の状態) を管理し，出力する．理由付けとして入力されるホーン節は次に示すいずれかの形式である．

$$a_1, \cdots, a_n \Rightarrow b.$$
$$a_1, \cdots, a_n \Rightarrow \bot.$$

ここに，$a_i (i = 1, \cdots, n; n \geq 0)$ 及び b は命題論理のアトム，記号 $\bot$ は偽 (false) を表し，記号 $\Rightarrow$ の左の各アトムを前件，右を後件と呼ぶ．

仮説の集合の部分集合を環境と呼ぶ．また，全ての理由付けの集合を J とすると，J とある環境から $\bot$ が導かれるときその環境を矛盾環境と呼び，J とある環境から $\bot$ が導けないときその環境を無矛盾な環境と呼ぶ．あるノードが信じられているとは (IN 状態と呼ぶ)，ある無矛盾な環境 E があって J と E からノードのデータが導かれることである．IN 状態のノードは，後から追加される知識によってそのノードが信じられていない状態 (OUT 状態と呼ぶ) になる可能性を持っている．ノードのラベルはそのノードが信じられている極小の環境の集合である．

3 並列 ATMS

以下，並列 ATMS の構造および機能の概略を示す．なお詳細については [4] を参照のこと．

3.1 構造

並列 ATMS では，外部から与えられたデータに対応して以下のデータ構造を有するプロセスを生成し，これによってノードを表現する：

γ_{Datum}: < データ，ラベル，*Justifications*, *Consequents*, *Attention*>.

Justificatinns: このデータを後件とする理由付けの前件ノードへメッセージを送信する出力ストリーム (これを $J-$ ストリームと呼ぶ) の集まり．

Consequents: このデータを前件に持つ理由付けの後件ノードへメッセージを送信する出力ストリーム (これを $C-$ ストリームと呼ぶ) の集まり．

Attention: 偽に対応するノードから送られる矛盾環境の削除命令を受信する入力ストリーム (これを $A-$ ストリームと呼ぶ)．

データおよびラベルは 2 節において定義されたものと同じである．$J-$ ストリームは理由付けに基づいてノードのラベルを計算する際に，その理由付けの前件ノードからラベルを収集するメッセージを送信するためのストリームである．さらに，$C-$ ストリームはラベルの更新が起こった場合に，後件ノードに対して，そのラベルの更新命令を伝える．*Justifications*, *Consequents* は，ノードを表すプロセスへのストリームの集まりとなっている．

また，偽に対応するノードは，極小の矛盾環境 (他の矛盾環境を包含しない矛盾環境，以下単に矛盾環境と呼ぶ) の集合をそのラベルとして保持し，自身以外の全てのノードへ $A-$ ストリームを通じて矛盾環境を伝達する．他のノードの *Attention* はその矛盾環境を受信するためのストリームである．

並列 ATMS では，基本的にノードのみをプロセスとして表現しているので入力されたデータの数のプロセスが生成される．図 1 に理由付けの例とそれによって構築されるプロセスのネットワークを示す．

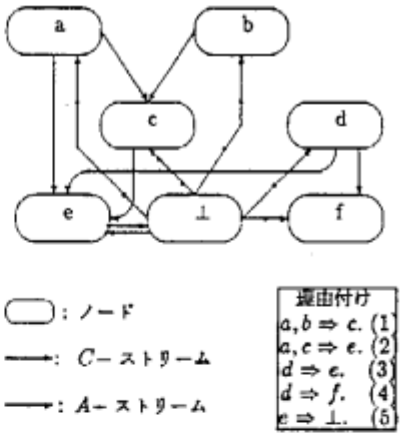

図 1: 本並列 *ATMS* によって生成されるプロセス・ネットワークの例

3.2 並列化機能

本並列 ATMS では以下のような処理を並列に実行することが可能である．

1. 外部から ATMS に与えられる複数の命令の並列実行

   個々のノードは互いに独立しており，同時生成が可能である．また，それぞれのノードに対する理由付けの追加，信念の状態の問い合わせは，それぞれに対して対応する命令を送ることで実行される．したがって，複数のノードのそれぞれに対する理由付けの

Impremementation of Parallel ATMS on Multi-PSI
Makoto NAKASHIMA† Yoshihiko OHTA‡ Katsumi INOUE‡
†JIPDEC ‡ICOT

追加も並列に実行できる．さらに仮説の追加と理由付けの追加が混在するような場合も並列に実行できる．ただしデータの問い合わせについては，それ以前に ATMS に与えられた命令が全て処理された後行なわれる．これにより，問い合わせを行なった時点でのラベルの正当性が保証される．

2. 理由付けに伴うラベル更新の並列実行

ラベル更新とは具体的に以下の処理を行なうことである．この中で (a),(b) および (c) の処理を並列に実行することができる．

ある理由付け $a_1, \cdots, a_i, \cdots, a_n \Rightarrow x \ (1 \leq i \leq n; n \geq 0)$ が与えられた時，その理由付けの後件ノード x を表すプロセス (γ_x と示す) は，その前件ノード γ_{a_i} への $J-$ ストリームを有しているので，γ_{a_i} に対してそのラベル L_i の問い合わせる (a)．これを受信した γ_{a_i} は自身のラベルを返す．次に γ_x は，収集したラベルと現在の自身のラベルの内容をもとに，新たに自身のラベルを求める．ここで，$x \neq \perp$ であるならば，この x を理由付けの前件に持つ全てのノードに対して $C-$ ストリームを介してラベルの更新命令を送る．それを受けたノードは，それぞれが持つ理由付けに基づいてラベルの更新を始める (b)．$x = \perp$ ならば，新しく追加された矛盾環境を $A-$ ストリームに送信する．$A-$ ストリームを介して矛盾環境を受信したプロセスは，自分自身のラベルの要素で矛盾環境を含むものを全て削除する (c)．

また，後件が相異なる複数の理由付けが与えられた時には，各々並列にそれぞれの後件ノードに対応するプロセスにおいて，上記の手順を実行することができる．

4 実測結果

並列 ATMS の実行性能を評価するために N-Queens 問題を取り上げた．以下 N=8 の場合を例に解法例を述べる．

まず，8×8 の盤上の 1 つの桝目に 1 つのクィーン (以下 Q) をおくことを仮説とし，次に二つの Q が互いにとり合う位置にある組合せを矛盾環境とする．そして，以下のような理由付けを導入する．

$$Q(i,j) \Rightarrow Q_i.$$
$$Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_{7j}, Q_{8k} \Rightarrow S_{jk}.$$
$$(1 \leq i \leq 6, 1 \leq j \leq 8, 1 \leq k \leq 8)$$

8-Queens の解はこの S_{jk} のラベルの和集合で示される．

ここで生成されるノードを使用できるプロセッサ (以下 PE) に静的に割り当て，PE の数を 1 から 64 まで段階的に変更して，実行を行なった結果を図 2 に示す．これによると 1 台のみの PE で実行を行なった場合に対して，8-Queens(10-Queens) の場合 8 台の時の実行速度は約 6.5 倍 (7.8 倍)，64 台の時は約 23 倍 (40 倍) 早くなることがわかる．

[3] と [7] では，Multi-PSI 上で稼働する ATMS について，それぞれ 5PE のときに逐次処理の 2 倍，16PE のときに 4 倍の高速化が観測されている．前者は，1 つの理由付けに伴うラベル計算の並列アルゴリズムを提案しており，後者では，各 ATMS ノードをプロセスで表現し，複数の理由付けの集まりに伴うラベル計算タスクを 1 つの PE に割り当てることで上記の結果を得ている．また，[2] でインプリメントされた ATMS は，13-Queens で 70 倍の高速化がなされたと報告している．しかしながら，この基本的なアルゴリズムは，入力された仮説の数の指数オーダのプロセッサが必要になる．そこで，極大の無矛盾な環境をプロセッサに割り当てる方法も提案されているが，これもまた，最悪の場合において指数オーダのプロセッサが必要になる．さらに，共有メモリ型並列マシン上でインプリメントされた ATMS についてもその研究成果が報告されている．[6] では，14PE 上での 8-Queens の解決において約 6.5 倍の高速化が観測され，[5] の ATMS では，4PE で 3.5 倍の高速化が観測されている．しかしながら，共有メモリ型マルチ・プロセッサでは，PE 数をあまり増やせない小規模並列のアーキテクチャなので，それ以上の高速化はあまり期待できないと考えられる．

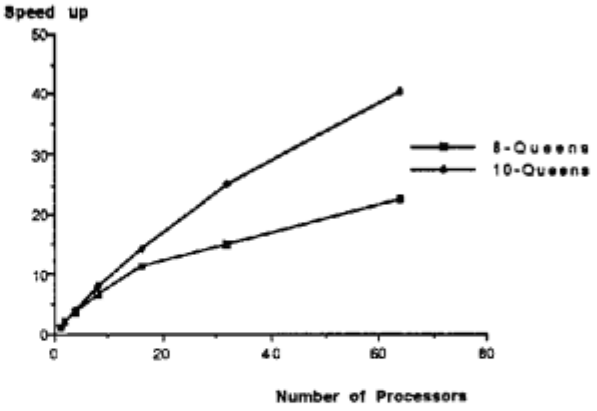

図 2: N-Queens 測定結果

5 まとめ

並列 ATMS の特徴は，ノードをプロセスによって表すことで，その生成数を現実的な数に押えることができる点にあり，同時に ATMS の処理の基本となるラベル更新の処理をそれぞれのノードが独自に行なうことで，ノードを一括管理する方法に比べて並列化の効果を期待できることにある．しかしながら，この方法によって並列処理の効果を得るためには，一つのノードへ負荷が集中することを避けるように，4節に示したような理由付けの戦略が必要となることを付記しておく．

今後の課題としては，使用できる PE へ順番に割り当てている負荷分散の仕方をノードの依存関係に従ったものにするということが挙げられる．

参考文献

[1] deKleer, J. : "An Assumption-based TMS", *Artif. Intell.* , Vol. 28, pp. 127–162(1986).

[2] Dixon, M., de Kleer, J. : "Massively Parallel Assumption-based Truth Maintenance", *Proc. of the AAAI-88*, pp. 199-204(1988).

[3] Harada, T. and Mizoguchi, F. : "Pararell Constraint Satisfaction by Paralleling ATMS", *Proc. of the PRICAI*, pp. 462-455(1990).

[4] 中島 誠, 太田好彦, 井上 克巳: "KL1 による並列 ATMS", ICOT Technical Reports(1991).

[5] 奥乃 博: "網: 新しい ATMS の処理系とその共有メモリ型マルチプロセッサ上での並列処理", 人工知能学会誌, Vol. 5, No. 3, pp. 79-88(1990).

[6] Rothberg, E. and Gupta, A. : "Experiences Implementeting a Parallel ATMS on a Shared Memory Multiprocessor", *Proc. of the IJCAI*, pp. 199-205(1989).

[7] 和田 真一: "疎結合型並列マシンにおける ATMS の並列化方式について", 人工知能学会全国大会 (第 4 回) 論文集, pp. 261-264(1990).