ATMSのラベル遅延更新アルゴリズム

5C-5

西岡 真吾　溝口 理一郎
大阪大学産業科学研究所

1 はじめに

解法をアルゴリズムとして定式化することが困難な問題解決にとっては，探索が本質的である．仮説推論は仮説的な状況を明示的に取り扱うことにより，探索を効率良く制御することができる．de Kleerが提唱したATMS (Assumption-based Truth Maintenance System)[1]は，仮説推論を現在までのところ最も適切に定式化した枠組である．ATMSは多重世界の扱い方に基づいて平行型と巡回型に大きく2つに分類することができる．前者は特定の世界に注目することなく全ての世界を一括して取り扱う．後者は単一の世界に注目して個々の世界を巡回し，問題解決は巡回中の世界のみで行う．de KleerによるDDB-Guided ATMS[2]は巡回型として分類することができる．

ATMSベースの問題解決システムの性能が良いといえるための前提の一つに，ATMSを用いることによるオーバーヘッドよりも，ATMSを用いることによって回避できるようになった探索のパスの負荷が大きいということがあげられる．設計型のエキスパートシステムなどといった，問題解決器が消費するCPU時間が極めて大きい問題では，この前提が成立するため，ATMSを用いることが有利であると言える．しかし，この前提は多くの問題解決にとって必ずしも成立するものではない．例えば，音声理解システムにおいては，問題解決器の一つのコンシューマが消費するCPU時間は少なく，ATMSの消費するCPU時間が全実行時間の半分を越えるようなこともまれではない．このような問題においてはATMS自身の性能がシステム全体の性能を左右するといえる．特に，ATMSはその実行時間の大半をラベル更新に費やしていることから，効率的なラベル再計算のアルゴリズムを与えることは問題解決システム全体の性能向上にとっても重要である．

本稿では，ATMSにおけるラベルの取り扱いに着目することにより，効率的なラベル更新のためのアルゴリズムを提案する．

2 ATMSにおけるラベルの取り扱い

ATMSは，ノードの状態を“in”，“out”の2つに分類する．つまり，並行型のATMSでは，空のラベルを持つノードを“out”，空でないラベルを持つノードを“in”と定義する．また，巡回型のATMSにおいては，ラベルに現れる環境のうち，現在の環境(current environment, 以後CEと略記)の部分集合であるような環境が1つもなければ“out”，少なくとも1つ存在すれば“in”であると定義する．また，並行型，巡回型のどちらのシステムにおいても，あるノードのラベルに現れる環境の数は，現時点までの推論で得られているそのノードの解釈の数よりは多くならず，また現在の環境で少なくとも1つの解釈が存在すれば，ラベルに現れる環境の数は0になることもない．言い換えれば，ATMSは，あるノードに対する解釈が少なくとも1つあるかどうかを管理しており，そのノードに対する解釈が何通りあるかは問題としていない．

ところで，多くの問題解決において個々のデータの解釈は複数存在する．例えば音声理解システムでは，半数以上のノードについてそれぞれのラベルに2つ以上の環境が現れる場合もある．このような場合，従来のATMSは全てのラベルを“完全”(complete)に保っていたため，空であるかどうかを判定するという目的から見れば不必要であるともいえる余分なラベル計算を行わざるを得なかった．

de Kleerが[3]において示したアルゴリズムでは，ラベルは常に“完全”，“健全”(sound)，“最小”(minimal)，“無矛盾”(consistent)に保たれる．しかし，これら4つの条件のうち“完全”は，先述の考察から，そのラベルが空でなければ無意味である．なぜならば，少なくともCEの部分集合であるような環境を少なくとも1つ含んでいるようなラベルを持つノードは，たとえそのラベルが“完全”でなくとも“in”であると断言することができるからであり，はじめに述べた「ノードが“in”であるか“out”であるか」という判定にとってはそれで十分だからである．

見かけが“out”(ラベルが空)であるようなノードのラベルが“完全”でなかったとすれば，そのノードのラベルを更新することによって，新たに環境が付加され，その結果ノードの状態が“in”に変化することがあるので，“空”のラベルが“完全”でないということは許されない．以上のことがらを考慮すれば，現在の状態が“in”であるノードに対しては，さしあたってラベルの更新を行う必要がないと言える．また，矛盾データのラベルは常に空に保たれるため，そのラベルは常に“完全”でなければならない．また，全ての矛盾を発見するという必要性からも矛盾データのラベルは“完全”であるべきである．

以上をまとめると，ATMSにおいては全てのノードのラベルを常に“完全”に保つ必要はない．そして，ラベルを更新する必要が生じるまではそのラベルの更新を行わないことによって，多くの無駄なラベル計算を避けることができる．そして，それはノードの解釈が複数考えられるような，多くの問題解決にとって有利に働く．(しかし，典型的な探索問題である8-queenなどをATMSベースの問題解決システムで記述すると，各ノードのラベルに現れるラベルの数は高々1個となる．オーバーヘッドを考えると，このような問題ではここに示した方法はむしろ不利である．)

3 アルゴリズム

de KleerによるラベルのアルゴリズムではATMSでは，新しい支持理由が与えられると即座にラベルの更新が始まる．それに対し，本アルゴリズムでは，新しい支持理由が与えられた際にはその支持理由によって支持されたノードに新たな環境が付加される可能性があること(すなわちラベルが完全ではないという情報)のみを記録する．また，あるノードのラベルが完全でなければそのノードを親にもつようなノードのラベルもまたは完全でないということが推測される．したがって，ラベルが完全でないという情報は支持理由を辿って伝播していく．

また，2節において行なった考察から，ラベルが完全でないという情報がラベルが空であるようなノードに伝播して来た時には，そ

Lazy Updating Algorithm of ATMS Labels
Shingo NISHIOKA　Riichiro MIZOGUCHI
The Instutute of Scientific and Industrial Research, Osaka Univ.

のノードに対してラベルの更新を行う必要が生じる．このノードのラベルの更新は，このノードを支持している親ノードに対して再帰的にラベルの更新を要求することにより，このノードを支持しているノードのラベルを完全にしてから自分自身のラベルを更新する．

以下にアルゴリズムの概略を示す．

アルゴリズム propagate($(x_1,\ldots,x_k \Rightarrow n)$)

1. [n のラベルが既に不完全であれば何もしない]
   if ($n.Incomplete$) return.
2. [n のラベルが完全でかつ空でなければ不完全さを伝播させる]
   if ($n.Incomplete$&¬empty($n.Label$))
   $n.Incomplete \leftarrow True.$
   foreach J in $n.consequents$ do
   propagate(J)
   return
3. [n のラベルが完全でかつ空であれば更新を行う]
   $n.Incomplete \leftarrow True.$
   force-update(n).
   return.

アルゴリズム force-update(n)

1. [ラベルが既に完全であれば何もしない]
   if (¬$n.Incomplete$) return.
2. [アップデートを行う]
   foreach J in $n.antecedents$ do
   foreach a in $J.antecedents$ do
   force-update(a)
   L = minimamize(Cartesian-product($J.antecedents$))
   Delete every environment from L
   which is a superset of some environment of $n.label$.
   Delete every environment from $n.label$
   which is a superset of some element of L.
   Add every remainning environment of L to $n.label$.
   if (this is the last iteration of loop) $n.Incomplete \leftarrow False.$
   if (¬empty(L)) return.

アルゴリズム force-complete-update(n)

force-update に似ているが，全ての親ノードに対して force-complete-update を呼び出す点が異なる．これは，nogood ノードのラベル更新のために呼び出される．

ここでは，ラベルのインクリメンタルなアップデートは示さなかったが，実際のインプリメントでは，ノード a のラベルの支持理由 $I_{(a,J)}$ を記録することにより，インクリメンタルなアップデートを行う．増加分を記録する必要があることは，ラベルの再計算を即座に行わなかった代償である，もしくは計算時間を記憶空間に転嫁していると考えることができる．

また，同じく示していないが，nogood ノードのラベルを更新した際には，新たに発見された矛盾を矛盾データベースに登録する作業も必要である．

4 その他の技法

巡回型のシステムにおいては，ノードの "in" と "out" を判定するためにラベルに現れる環境と，CE をひとつひとつ比較しなければならない．しかし，ラベルを CE の部分集合であるようなものと，そうでないようなものの 2 つに分類して持っておけば，"in", "out" の判断は容易になる．ここでは，CE の部分集合であるような環境からなるラベルを "in Label", 他方を "out Label" と呼ぶことにする．この方法は de Kleer による従来のラベルアップデートアルゴリズムでは環境遷移等にかかるオーバーヘッドが "in", "out" の判断が容易になったことによって得られる利益と相殺するためにあまり用いられていない．しかし，3節で提案したアルゴリズムにおいては，この方法は有効である．

すなわち，ラベルが 2 つに分かれているために，2 つのノードのラベルの直積を計算する際に，"in Label"×"in Label", "in Label"×"out Label", "out Label"×"in Label", "out Label"×"out Label", の 4 回の計算を行うことになるが，"in Label" 同士の直積は，全ての要素が CE の部分集合であるようなものとなり，それ以外のものは全ての要素が CE の部分集合とならない．"out Label" にいくら新しい環境が付け加わったとしても，そのノードの現在の "in", "out" の状態は変化しないので，先のアルゴリズム中の force-update において再計算する必要があるのは，"in Label" のみである．したがって，"out Label" に現れる環境は，環境遷移が起こり "in Label" に移動しない限りラベル再計算で用いられることはない．これは，単純に考えてラベル計算が $\frac{1}{4}$ になることを意味する．

先にも述べたように，ATMS の負荷の大半がラベル更新に伴うものであることから，この手法も ATMS の高速化に十分貢献すると思われる．

5 評価

de Kleer によるアルゴリズムを用いて実現した ATMS と 3節で示したアルゴリズムを用いた ATMS を Symbolics 3620 上で Common Lisp を用いて作成し，比較実験を行った．問題解決器として，音声理解システムを用いた．

従来版，改良版共に ATMS が消費する CPU 時間は問題解決全体における CPU 時間の 50~60%であった．従来のアルゴリズムを用いた ATMS は問題解決全体にかかる時間が平均 429 秒であったのに対し，改良版では平均 292 秒となり，約 30%の効率改善となった．

また，全てのノードのラベルに現れる環境のうち，問題解決終了時の CE の部分集合であるような環境の数は別のノードに現れている同じ環境を重複して数えて，従来版で 644 個，改良版では 413 個であり，約 35%減少した．同じく終了時に，従来版ではラベルに現れる CE の部分集合であるような環境の数の最大値は 10 であったのに対し，改良版ではその数は 4 であった．これらのことから，後まわしにした計算が，ついに問題解決終了まで不必要である確率はそれほど小さくないものと予想することができる．

6 おわりに

以上，ATMS ラベルの遅延更新アルゴリズムを提案し，音声理解システムを問題解決器とした例題において，その有効性を検証した．今後は 4節で述べた "in Label", "out Label" の 2 つのラベルを用いるアルゴリズムをインプリメントし，その有効性を検証していく予定である．

参考文献

[1] de Kleer, J., "An Assumption-based Truth Maintenance System", *Artificial Intelligence*, 28, pp. 127-162 (1986).

[2] de Kleer, J., "Back to Backtracking : Controlling the ATMS", *Proc. of AAAI '86*, pp. 910-917 (1986).

[3] de Kleer, J., "A General Labeling Algorithm for Assumption-Based Truth Maintenance", *Proc. of AAAI '88*, pp. 188-192 (1988).