1E−2

ATMSと単純化順序を用いた項書き換えシステム停止性検証システムの開発 -第2報-†

近藤 久　栗原 正仁　大内 東
北海道大学工学部

1 はじめに

項書き換えシステム(*Term Rewriting System*:TRS)は$l \to r$の形をした書き換え規則(rewrite rule)の集合として記述されたプログラムである。ここでl，rは変数、演算子から構成された項である(rに含まれる変数はlにも含まれる)。一般にTRSの停止性を検証することは決定不能な問題であることが知られているが、単純化順序[1]を用いることによって与えられた特定のクラスのTRSの停止性を検証することが可能である。

本稿では単純化順序として辞書式経路順序を用いるが、辞書式経路順序を用いるためには項を構成している演算子の集合上の半順序$\succ$を決定しなければならない。この決定問題は通常のバックトラック法を用いることによって決定することが可能であるが、多くの無駄なバックトラック、推論の再検知、矛盾の再検知を行なう。筆者らは[6,7]においてこれらの問題点を解決するためATMS[3]の利用を提案した。本稿では、さらにこのATMSを拡張し、新しいnogoodの計算のためにhyperresolutionの利用を示す。また、justificationの計算など、ATMS利用のためのプロトコルを明確にする。

2 TRSの停止性検証

辞書式経路順序(*Lexicographic Path Ordering*:$\succ_{lpo}$)[2]は、項を構成している演算子の集合上の半順序$\succ$を項の集合上の半順序$\succ_{lpo}$に拡張する。このときTRSの全てのルール$l \to r$について$l \succ_{lpo} r$となることを示すことができるならば停止定理[1]より停止性をみたす。したがって、停止性を検証するためには半順序$\succ$を決定しなければならない。[6,7]

3 ATMS

ATMS(*Assumption based TMS*)は、通常のバックトラック法の問題点である
1. 無駄なバックトラック(futile backtracking)
2. 推論の再検知(rediscovering inferences)
3. 矛盾の再検知(rediscovering contradictions)

を効率よく解決するために開発された機構である。[3,4,5]

ATMSを利用することによって全体的な問題解決器(*Overall Problem Solver*:OPS)は"問題領域に関する部分(問題解決器(*Problem Solver*:PS))"と"PSの行なった推論を記録する部分(ATMS)"に明確に分割することができる。

3.1 停止性検証システムにおけるATMSの利用

停止性検証システムではPSが直接、環境(ルールを向き付けることのできる$\succ$の部分集合)を与えるようにした(justification=environment)。このようにするとATMSはjustificationからの環境計算を必要としないため処理効率が上がると考えられる。

3.1.1 探索木ノード

PSが生成する探索木は4種類のノードを含む。
(1)P選択点(選好順序(precedence)の選択):このノードは楕円によって表わされ、楕円中に大小を決定したい演算子を含んでいる。
(2)S選択点(部分項(subterm)の選択):このノードは長方形によって表わされ、部分項の選択を表わす。
(3)境界点:円で表わされ、ルール$n+1$の向き付け開始を表わす。(ルール$1,2,\cdots,n$の向き付け成功)
(4)失敗点:×で表わされ、あるルールの向き付け失敗を表わす。

ここで述べたノードを用いた探索木は4にしめす。

3.1.2 Justificationの計算

ATMSに対して与えるjustificationは次の2種類とした。つまり、
(1) ルールの向き付けに成功した場合のjustification
(2) ルールの向き付けに失敗した場合のjustification
である。(2)のjustificationはnogoodであり、次節で述べる。

(1)の場合、例えば現在の環境が$\{f \succ g, h \succ g, h \succ i\}$であり、向き付けようとしているルールが

$$f(h(x)) \to g(f(i(x)))$$

とすると、このルールの向き付けに用いたのは$\{f \succ g, h \succ i\}$である。したがって、これがこのルールに対するjustificationとなる。

他の例として、上記と同様の環境とし、向き付けようとしているルールが

$$i(h(f(x)) \to h(g(x))$$

とすると、このルールのjustificationは$\{f \succ g\}$である。ここで$\{h \succ i, f \succ g\}$は$\{f \succ g\}$より弱いためjustificationとはならない。これはLPOの定義がルールの左右の項の最も外側の演算子の比較を必要としないためである(この例の場合$h \succ i$)。

3.1.3 Nogoodの計算

nogoodは以下で定義されるhalfgoodを用いて計算される(nogoodはhalfgoodに含まれる)。各ノードからバックトラックする時点で以下のように計算される。
(1)失敗点での計算
失敗に関与した仮説の集合をhalfgoodとする。例えば、上記と同じ環境のもとで向き付けようとしているルールが

$$g(f(i(x))) \to f(h(x))$$

とすると、halfgoodは$\{f \succ g, h \succ i\}$となる。

† Development of Termination Verifier for Term Rewriting Systems based on ATMS and Simplification Orderings -2nd report-
Hisashi KONDOH, Masahito KURIHARA, Azuma OHUCHI
Faculty of Engineering, Hokkaido University

(2)P 選択点での計算
P 選択点では hyperresolution rule を適用し、新しい halfgood を生成して、親ノードに伝播させる。例えば、一方の部分木から halfgood:$\{f \succ g, h \succ g\}$ を受け取り、他方の部分木から halfgood:$\{g \succ f, g \succ i\}$ を受け取ったとする。hyperresolution rule をこの 2 つの halfgood に適用する手続きを hyperresolve(hgd1,hgd2) とする。hyperresolve は 2 つの halfgood の中から互いに反する関係を削除し和をとる。したがって、新しい halfgood は $\{h \succ g, g \succ i\}$ となる。
(3)S 選択点での計算
S 選択では子ノードからの halfgood の集合和をとり、新 halfgood とし、親ノードに伝播する。例えば、現在の環境が $\{i \succ f, i \succ g\}$ とし、向き付けようとしているルールが

$$f(g(x), h(x)) \rightarrow i(i(x))$$

であり、halfgood が $\{h \succ i\}$ とすると fig.1 の実線で囲まれた部分が向き付け失敗の探索木を示すこととなる。したがって、S 選択点では halfgood:$\{i \succ f, i \succ g\}$ と hyperresolution からの halfgood:$\{i \succ f\}$ との集合和をとり、halfgood:$\{i \succ f, i \succ g\}$ となる。
(4) 境界点での計算
境界点では子ノードから伝播された halfgood を nogood とする。PS は nogood を避けるようにバックトラックし (dependency directed backtracking [3])、この nogood を戻り先のノードへ伝播する。fig.1 の例では、nogood:$\{i \succ f, i \succ g\}$ と nogood:$\{h \succ i\}$ が境界点での計算を示している。

3.1.4 ATMS への登録

PS が ATMS へ推論結果を通知するのは以下の場合である。
(1) 各ルールの向き付けに成功したとき (境界点)。
(2) 各ルールの向き付けに失敗してバックトラックする時点 (失敗点)。
(1) 登録手続きを record-inference(r,j) とする。ただし、r は向き付けすることのできたルール、j は justification である。このとき、既に ATMS-database にルール r の ATMS-node が存在するならば ATMS-node γ_r の label に justification を追加する。さもなくば、ルール r の ATMS-node を作成し登録する。
(2)(1) と同様に record-inference を用いる。ただし、r は矛盾を示す⊥、j は justification である。このとき、j は Nogood-database へ登録され、PS は nogood を避けるようにバックトラックする (dependency directed backtracking [3])。これによって、通常のバックトラック法の問題点 1 を避けることができる。

3.1.5 ATMS への問い合わせ

PS が ATMS に問い合わせを行なうのは以下の場合である。
(1) 半順序≻の拡張を行なう各選択点 (P 選択点)。
(2) 次のルールの向き付けへ行く直前 (境界点)。
(1) 半順序の拡張を行なうことで現在の環境が、ある nogood を包含することになるかを検査する。この関数手続きを nogoodp(f,g,≻) とする。ただし、f, g は大小関係を決定しようとしている演算子、≻は現在の環境である。このとき、$\{f \succ g\} \cup \succ$の推移的閉包が、ある nogood を包含するならば、真を返し、さもなくば、偽を返す。真ならば PS はその関係 $(f \succ g)$ を選択しない。これによって、通常のバックトラック法の問題点 3 を避けることができる。
(2) この場合、向き付けしようとしているルールが既にある環境のもとで向き付けに成功している場合があるため、そのルールの ATMS-node の label 中の要素 (environment) が現在の環境に包含されるかどうかを検査する。この関数手続きを orientp(r,≻) とする。ただし、r は向き付けようとしているルール、≻は現在の環境である。このとき、ルール r の ATMS-node の label 中のある環境が現在の環境に包含されるならば、真を返し、さもなくば、偽を返す。真ならば、PS はそのルールの向き付けを行なわなくてよい。これによって通常のバックトラック法の問題点 2 を避けることができる。

4 探索木

ここでは 3 本のルールを持つ TRS の停止性検証の探索木を示す。

$$TRS: \begin{cases} i(h(g(x))) \rightarrow f(g(x), x) \\ f(g(x), h(x)) \rightarrow i(i(x)) \\ i(x) \rightarrow h(x) \end{cases}$$

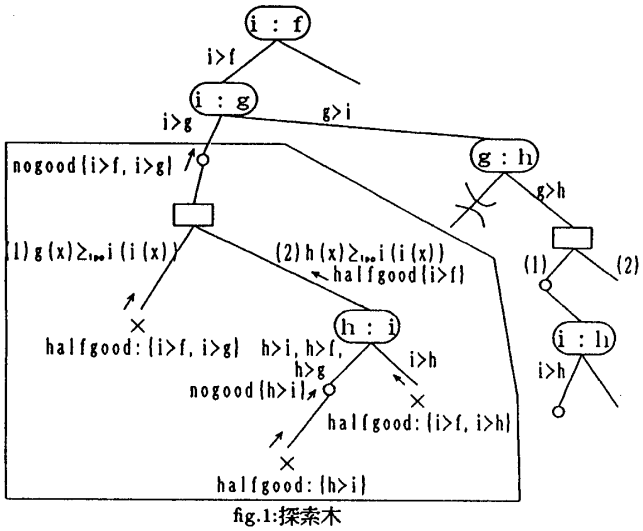

fig.1:探索木

5 おわりに

本稿では TRS の停止性検証に対し ATMS を利用した停止性検証システムの開発について述べた。通常のバックトラック法と比較して ATMS の利用は効率を改善する。今後の課題として、データ構造の工夫によりさらに効率のよい停止性検証システムに改良すること、辞書式経路順序以外の単純化順序を取り入れること、*Knuth-Bendix* 手続きによる完備化への応用が挙げられる。

参考文献

[1] Dershowitz,N., Orderings for term-rewriting systems *Theoretical Computer Science* 17 pp.279-301(1982)

[2] Dershowitz,N., Termination of Rewriting *J.Symbolic Computation* 3 pp.69-116(1987)

[3] de Kleer,J., An Assumption based TMS *Artificial Intelligence* 28 pp.127-162(1986)

[4] de Kleer,J., Extending the ATMS *Artificial Intelligence* 28 pp.163-196(1986)

[5] de Kleer,J., Problem solving with the ATMS *Artificial Intelligence* 28 pp.197-224(1986)

[6] 近藤, 栗原, 大内., "単純化順序を用いた項書き換えシステム停止性検証システムの開発 −ATMS の利用−" 信学技報 Vol.89 No.427 pp.9-16(1990)

[7] 近藤, 栗原, 大内., "ATMS と単純化順序を用いた項書き換えシステム停止性検証システムの開発" 情報処理学会第 40 回全国大会論文集 pp.974-975(1990)