

6 J-3

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

〇近藤 久 栗原 正仁 大内 東
(北海道大学 工学部)

1. はじめに

一般に項書き換えシステム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、Dershowitzによって提案された単純化順序を用いることによって、特定のクラスの項書き換えシステム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仮定するが、本稿の考え方は他の順序を仮定しても成り立つ。

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を決定しなければならない。この半順序を決定する問題は、多くの不必要な探索、矛盾の再発見、不必要な推論を行なう。これらの問題点を避けて効率よい問題解決を実現するためのアーキテクチャとしてATMSが提案されている。

本稿では、ATMSを用いた停止性検証システムの開発について述べる。

2. 項書き換えシステム(Term Rewriting System:TRS)

項書き換えシステムは、 $l \rightarrow r$ の形をしたルール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。書き換えを行うためのルールの選択、ルールを適用するための項の選択は非決定的になされる。このため、TRSの停止性を検証することは非常に難しい問題である。

以下にTRSの例を示す。この例は与えられた論理式を論理的等価な選言標準形に書き換えるものである。

表1

$\neg \neg x \rightarrow x$
$\neg (x \vee y) \rightarrow (\neg x \wedge \neg y)$
$\neg (x \wedge y) \rightarrow (\neg x \vee \neg y)$
$x \wedge (y \vee z) \rightarrow (x \wedge y) \vee (x \wedge z)$
$(y \vee z) \wedge x \rightarrow (y \wedge x) \vee (z \wedge x)$

3. 停止性の検証

TRSの停止性を検証する一般的な方法は存在しないが、特定のクラスのTRSに対して、いくつかの検証法が考案されている。以下では参考文献[1], [2]にもとづきTRSの停止性検証法について述べる。

《定義1》単純化順序(Simplification Ordering) [1]

項の集合を T とする。 T 上の推移的かつ非反射的な関係 $>$ は、次の性質をみたすならば単純化順序である。

- | | |
|---|-------|
| (1) $t > t'$ ならば $f(\dots t \dots) > f(\dots t' \dots)$ | (置換) |
| (2) $f(\dots t \dots) > t$ | (部分項) |
| (3) $f(\dots t \dots) > f(\dots \dots)$ | (削除) |

このとき次のことが成り立つ。

Development of Termination Verifier
for Term Rewriting Systems
based on ATMS and Simplification Ordering
Hisashi KONDOH, Masahito KURIHARA, Azuma OHUCHI
HOKKAIDO University.

【停止定理】(Dershowitz [1])

項の集合 T に関するTRS: $P = \{l_i \rightarrow r_i\}$ $i=1, \dots, n$ は、 l_i の変数に T 中の任意の項の代入に対して、 $l_i > r_i$ $i=1, 2, \dots, n$ となる単純化順序 $>$ が T 上に存在するならば停止性をみたす

単純化順序として辞書式経路順序が知られている。(辞書式経路順序は再帰的経路順序[1]の辞書式順序への拡張である。)

《定義2》辞書式経路順序: $>_{lpo}$

(Lexicographic Path Ordering: [2] 定義はKamin & Lévy, 1980) 演算子の集合上の半順序を $>$ とする。

項を $s = f(s_1, s_2, \dots, s_m)$
 $t = g(t_1, t_2, \dots, t_n)$ とする。

$s >_{lpo} t$ であるのは、以下のいずれかをみたすときに限る。

- (1) $s_i >_{lpo} t$ for some $i=1, 2, \dots, m$
- (2) $f > g$ かつ $s >_{lpo} t_j$ for all $j=1, 2, \dots, n$
- (3) $f = g$ かつ $(s_1, s_2, \dots, s_m) >_{lpo}^* (t_1, t_2, \dots, t_n)$ かつ $s >_{lpo} t_j$ for all $j=2, \dots, n$
($>_{lpo}^*$ は $>_{lpo}$ の辞書式順序への拡張)

【定理2】([2] 証明はKamin & Lévy, 1980)

辞書式経路順序は項を構成する演算子がfixed arityであるとき単純化順序である。

定理2と停止定理より以下の停止性検証法が得られる。

・辞書式経路順序法(LPO-method)

項の集合 T に関するTRS: $P = \{l_i \rightarrow r_i\}$ $i=1, \dots, n$ は、すべてのルールが、 $l_i >_{lpo} r_i$ ($i=1, 2, \dots, n$)となる半順序 $>$ が項を構成する演算子の集合上に存在するならば停止性をみたす。

4. 停止性検証アルゴリズム

ここで述べる停止性検証アルゴリズムはATMSを用いていない、通常のバックトラック法によるアルゴリズムである。(詳細は参考文献[3]を参照)

ルールの集合 R と演算子の集合 F 上の半順序 $>$ が与えられたとき、 $>$ の拡張 $>^*$ により定義される辞書式経路順序 $>_{lpo}^*$ で R の停止性を示すことができるならば、 $>^*$ を返し、さもなければ失敗を告げる手続きを tp とする。

ルールが0本のとき($R = \{\}$)、 $>$ を返す。ルールが n 本のとき、手続き $tp1$ により F 上の半順序 $>$ を $>^1$ に拡張($> \subseteq >^1$)

し、1本目のルールの停止性を示す。さらにこの拡張された $>$ を用い $>$ に拡張($> \subseteq >^1 \subseteq >$)し、再帰的に残りのルールの停止性を示すことができるか検査を行なう。停止性を示すことができたならば $>$ を返す。途中で、あるルールの停止性検査に失敗したときは、最も近くの選択点にバックトラックする。

・半順序の拡張

辞書式経路順序の定義が利用できるようなすでに演算子の関係が定まっている場合は、定義にしたがって停止性を検査する。(定義の(1)は $g > f$ の場合に用いる)

したがって、演算子間に何ら関係が付いていない場合は、半順序を拡張しなければならない。拡張は以下の(1),(2)となる。

(ルールの左側の項の最も外側の演算子を f 、右側の項の最も外側の演算子を g とする)

(I) $f > g$ として、半順序 $>$ の拡張を行ない、辞書式経路順序の定義(2)に従って停止性を検査する。

(II) (I)の拡張での停止性検査が失敗した場合、 $g > f$ として、半順序 $>$ の拡張を行ない、辞書式経路順序の定義(3)に従って停止性を検査する。

なお、ここで述べたアルゴリズムを用いた停止性検証システムは、既に開発済みである。このシステムに対する実験結果は、大会当日発表する。

5. ATMS (Assumption based Truth Maintenance System [4])

ATMSは、問題解決器(problem solver)の推論結果をその根拠とともに保持し、その後の推論を効率よくするために開発された機構である。ATMSを用いることで問題解決における

- (1) 無駄なバックトラック(futile backtracking)
 - (2) 矛盾の再検知(rediscovering contradictions)
 - (3) 推論の再発見(rediscovering inferences)
- を避けることができる。以下では、ATMSを用いた場合の停止性検証システムがたどる探索木を用いて(1)~(3)までの例について述べる。

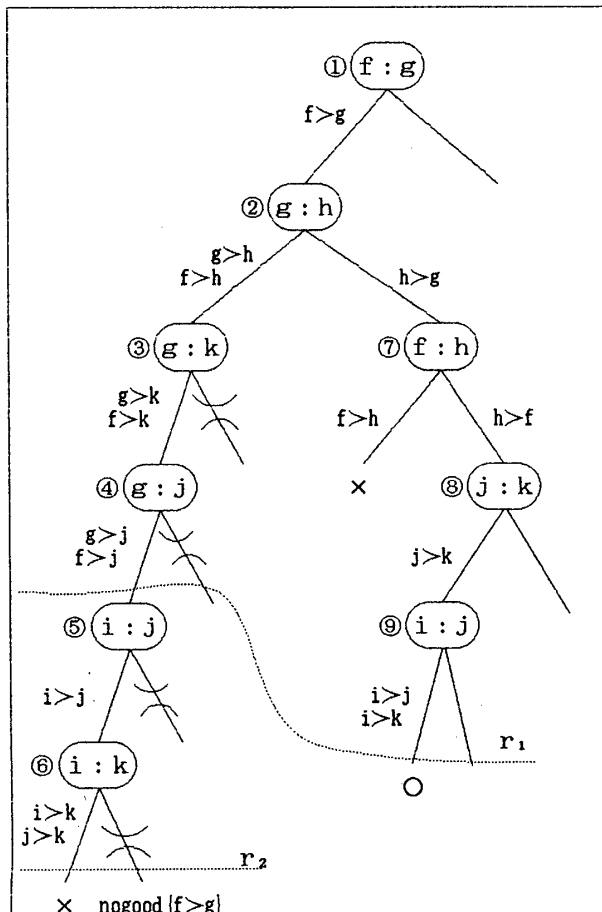


図1. 探索木

図1の例は、演算子の集合 F 上の半順序 $>$ を拡張し3本のルールを持つTRS $R: \{r_1, r_2, r_3\}$ の停止性を検証するための探索木を示している。

$$\begin{cases} r_1: f(g(f(h(j(i(?x)))))) \rightarrow g(f(h(k(j(?x)))))) \\ r_2: i(j(?x)) \rightarrow j(i(k(?x))) \\ r_3: h(?x) \rightarrow f(?x) \end{cases}$$

各枝のラベルは、その枝をたどったときに選択し、拡張した演算子の大小関係を示し、さらに $>$ の推移性による関係も示している。各ノードはそのときの左右の項の最も外側の演算子を比較する選択点を示している。ノード①~⑥までの選択は半順序の拡張法(I)に従って $>$ を拡張し r_1, r_2 の停止性を示すことはできるが r_3 の停止性を示すことはできない。ここで r_3 の停止性を示すことに失敗した原因を解析すると、ノード①での選択 $f > g$ とノード③での選択 $g > h$ による $>$ の推移性からの $f > h$ であることがわかる。ATMSでは $f > h$ をnogoodという矛盾を導く環境(environments)としてデータベースに記憶する。さらに、 $f > h$ だけが r_3 の失敗原因であるから直接ノード⑥からノード②にバックトラックすればよい。したがって、ノード③から⑤の他の選択は刈り込まれる。これは(1)の例に相当し、無駄なバックトラックを避ける。

前記によりノード②での他の選択 $h > g$ を選択しノード⑦にいた。このとき、既にnogoodとして $f > h$ が記憶されていることから $h > f$ を選択することになる。これは(2)の例に相当し、矛盾の再検知を避ける。

これまでの選択とノード⑧、⑨での選択によって r_1 の停止性を示すことができる。さらに、 r_2 は既にノード⑤、⑥での選択 $i > j, j > k$ で停止性を示せることがわかっている。ATMSでは、これをlabelと呼ばれるデータとして記憶している。したがって、 r_1 の停止性を示すための選択の中から $i > j, j > k$ が含まれているため r_2 の停止性を検査する必要がない。これは(3)の例に相当し、推論の再発見を避ける。

上記の例でわかるように、ATMSを用いた停止性検証システムでは通常のバックトラックを用いた場合より効率よく停止性を検証することができる。

6. おわりに

本稿では、通常のバックトラック法を用いた停止性検証アルゴリズム、及びそのアルゴリズムの持つ問題点に対する解決策としてATMSの利用について述べた。ATMSの利用は効率よく停止性を検証するために有用であると考えられる。

ATMSの利用により今回述べた問題点を解決することが可能であるが、単純な(構文的に簡単)TRSに対しては逆にオーバーヘッドが生ずると思われる。

今後の課題は、効率のよいATMSを用いた停止性検証システムを実現し、辞書式経路順序以外の単純化順序についてもシステムにとり入れることが上げられる。

〈参考文献〉

- [1] Nachum Dershowitz: 'Orderings for Term Rewriting Systems' Theoretical Computer Science 17(1982) pp.279-301
- [2] Nachum Dershowitz: 'Termination of Rewriting' J.Symbolic Computation(1987)3, pp.69-116
- [3] 近藤、栗原、大内: '再帰的経路順序により項書き換えシステムの停止性を検査するシステムの開発' 第15回システム、第10回知識工学合同シンポジウム論文集 pp.127-130
- [4] Johan de Kleer: 'An Assumption-based TMS' Artificial Intelligence 28(1986) pp.127-162
- [5] M.Kurihara, I.Kaji: 'Termination of the Direct Sum of Rpo-Terminating Term Rewriting Systems' Transactions of IEICE Vol.E71, No.10(1988) Letter pp.975-977
- [6] Nachum Dershowitz, Zohar Manna: 'Proving Termination with Multiset Orderings' Communications of the ACM 22(1979) pp.456-476
- [7] D.A.Wolfram: 'Forward Checking and Intelligent Backtracking' Information Processing Letters 32(1989) pp.85-87