5S−6

項書き換えシステムの合流性

張 磊　伊藤英則
名古屋工業大学

1. はじめに

項書き換えシステムについての研究が盛んに行なわれている。とくに、合流性に関するKnuth-Bendixアルゴリズムの研究が活発に行なわれている。本研究では、そのアルゴリズムの実行系を作ることを試みる。

2. 準備

項書き換えシステム（TRS）は、書き換え規則の集合である。書き換え規則（rewriting rule）は、二つの項を記号===>で結んだものである。項を書き換え規則のから、より簡単なものへ変形することを簡約（reduction）するといい、MをNに簡約することをM→Nで表す。書き換え規則が適用出来る部分をリデックス（redex）という。さらに簡約出来ない項は正規形（normal form）といい、M↓と表す。無限の簡約のないTRSは停止性（termination property）を持つという。あるTRSに対して、ある項がリデックスを一つ以上持っている場合は、二通りの簡約があり得る。もし、このような項が常に最終的結果が同じ正規形になるなら、このTRSは合流性（Confluency property）を持つという。TRSは停止性と合流性を同時に持つならば、完備（complete）なTRSであるという。

完備なTRSは、Knuth-Bendixアルゴリズムによって、ある程度実現可能である。

3. Knuth-Bendixアルゴリズム

ここで述べる手続きの基本思想は、「二つの書き換え規則の左辺同士を照合して、どちらの規則によっても簡約されるような要素を作り出し、それぞれの規則で簡約する。その結果のどの対もさらに簡約化して同じになるなら、簡約系が完備である」という事実に基づいている。もし、簡約化しても同じにならない対があれば、それらを新しい簡約規則として追加していって、簡約系を逐次的に完備化していく。
手続きは以下のように示している。

Confluency property of Term Rewriting Systems
Lei Zhang, Hidenori Itoh
Nagoya Institute of Technology

初期値：等式の有限集合 E, 整礎順序 $>$
$E_0 := E; R_0 = \emptyset$; $i := 0; p := 0;$
Loop
while $E_i \neq \emptyset$ do
　E_i から等式 $M = N$ を選ぶ
　それぞれの正規形 $M\downarrow, N\downarrow$ を求める
　if $M\downarrow = N\downarrow$ then
　　$E_{i+1} := E_i - \{M = N\};$
　　$R_{i+1} := R_i; i := i + 1;$
　else if $(M\downarrow > N\downarrow) or (N\downarrow > M\downarrow)$ then
　　benig
　　if $(M\downarrow > N\downarrow)$ then $\lambda := M\downarrow, \rho := N\downarrow$
　　else $\lambda := N\downarrow, \rho := M\downarrow;$
　　endif ;
　　λ_k は規則 $\lambda \to \rho$ によって、λ'_k に簡約される
　　$E_{i+1} := E_i - \{M = N\}$
　　$\cup\{\lambda'_k = \rho_k \mid k : \lambda_k \to \rho_k \in R_i, k \in K\};$
　　$p = p + 1;$
　　$R_{i+1} := \{j : \lambda_j \to \rho'_j \mid j : \lambda_j \to \rho_j \in R_i, j \notin K\}$
　　$\cup\{p : \lambda \to \rho\};$
　　ここで、ρ'_j は $R_i \cup \{\lambda \to \rho\}$ を用いて，
　　求めた ρ_j の正規形である。
　　$i := i + 1$
　　endbegin;
　else exitloop (failure)
　endif
endwhile;
危険対の計算：もし、R_i でのすべての規則がマークされたら
exitloop(R_i *is canonical*)
　　そうではないと、R_i でのアンマークされた規則を選び
　　ラベル k を付け、この規則とラベル $i(k > i)$ の規則との
　　間のすべての危険対を E_{i+1} に入れる。
　　規則 k はいまマークされる以外は、$R_{i+1} = R_i$
　　$i := i + 1$
endloop ■

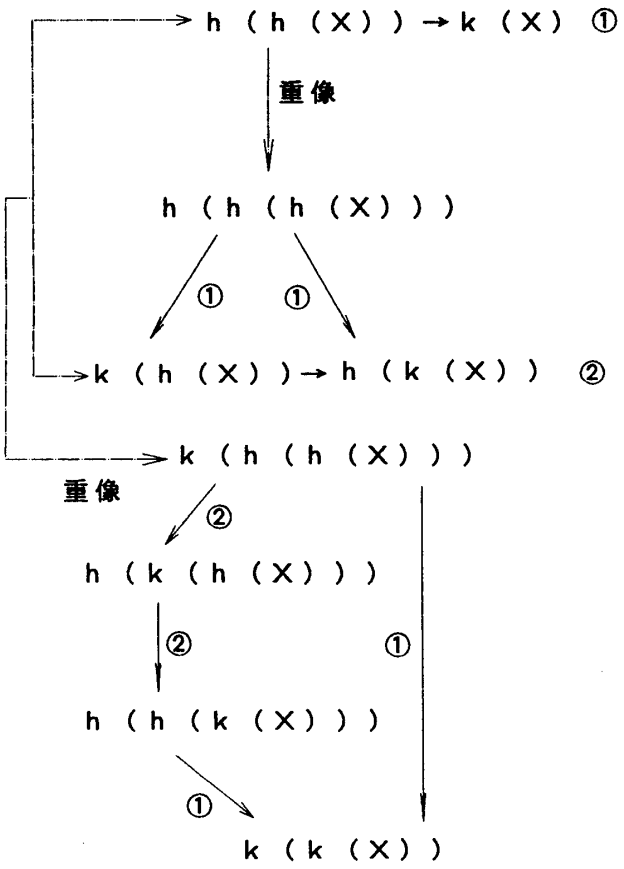
4. prologによる実行系

本論文はprologでワン・ルール・システムを処理するKBプログラムを実現した。

以下に一例をあげる。

ワン・ルール・システム: $h(h(X)) = k(X)$

このシステムの左辺は自分自身がh(h(X))という重像を持っているから(5)、二通りの置き換えを行なうことができ、発散する危険対 $k(h(X))$ と $h(k(X))$ が出る。これを規則にして、システムに追加することによって、元のシステムは二つの規則のシステムになる。引続き、この二つの規則の間には発散する危険対があるかどうかを調べる必要が発生する。調べによって、発散する危険対が存在しなかったので、なお、二番目の規則は自分自身が重なりになっていないから、このシステムが完備なシステムとなる。この過程が下図によって、示される。



合流していく過程

次は、prologの実行系によって、このシステムを実行してみよう。

?- $knuthbendix(h(h(X)) = k(X))$.

★★★★★ System is complete ★★★★★:

$rule((h(h(\_1228))' ===>' k(\_1228))) : -!.$

$rule((k(h(\_1244))' ===>' h(k(\_1244)))) : -!.$

これは、たまたまうまくいった例であるが、一般には、発散する危険対がどんどん出てきて、無限のループに落ちてしまい、あるいは、危険対は順序づけができなくて、失敗してしまう場合が多い。

5. おわりに

今後、種々のTRSシステム（線形、非線形）処理の実行系を作成する。また、危険対の順序づけに関しても調査する。この実行系は出てきた危険対に対して、辞書順序によって、順序をつける。しかし、これだけでうまくいかない場合があるので、他の順序づけ方法も考えないとならない。いままで、提案されたいろんな順序が(1),(2),(3)を参照されたい。

この手続きは失敗したり、無限ループに落ちたりする可能性があるので、厳格に半アルゴリズムといえよう。その故に手続きの改良に関する研究も活発に行なっている。新しい関数記号の導入とか、AC一完備化とか、無向完備化とか研究されている(1)。これらの結果をプログラムにして、いろいろなパラメータを換えて、調べてみるのは今後の課題である。

参考文献

(1) 坂井 公： Knuth-Bendixの完備化手続きとその応用, コンピュータソフトウェア,Vol.4,No.1(1987),pp2-22.

(2) 大須賀昭彦: 項書き換えシステムと完備化手続き, bit,1990,Vol.20,No.4,pp59-67.

(3) 二木厚吉, 外山芳人: 項書き換え型計算モデルとその応用, 情報処理,Vol.24,No.2(1983),pp.133-146.

(4) Huet,G.: A Complete Proof of Correctness of the Knuth-Bendix Completion Algorithm,J.Comput.Syst. Sci.,Vol.23,No.1(1981),pp.11-21.

(5) Huet,G.: Confluent reductions:abstract properties and applications to term rewriting systems, J.ACM,Vol.27,No.4(1980),pp.797-821.

(6) L.Sterling,E.Shapiro:Prologの技芸、松田利夫 訳. 共立出版.