数理解析研究所講究録
1093 巻 1999 年 136-141

順序ソートの自動推論と ラベル付けに基づく合流性判定への応用

宮下 大　酒井 正彦　坂部 俊樹
Dai Miyashita　Masahiko Sakai　Toshiki Sakabe
名古屋大学大学院 工学研究科 情報工学専攻
dai@sakabe.nuie.nagoya-u.ac.jp
{sakai,sakabe}@nuie.nagoya-u.ac.jp

本論文は、項書換え系の合流性を証明するシステムについて述べる。証明手法としては、従来の手法の他に外山によって提案されたラベル付けによる方法を用いる。これは、関数記号にラベル付けして得られる関数記号を導入することによりラベル付き項上の項書換え系 R を項書換え系 R' に変換する方法である。新しく得られた R' の導出関係は、ラベルを取り除いた項上の R の導出関係と等しいため、R' の合流性を示すことによって従来の手法が適用できなかった項書換え系 R の合流性を証明できる場合がある。本研究では、与えられた項書換え系に対してそれと矛盾しない順序付型情報を自動的に付加し、その型情報に基づいて関数記号にラベルを割り当てることにより自動的に R' を求める方法を提案する。また、この方法に基づいて試作した合流性判定システムについて述べる。

1　はじめに

項書換え系（TRS）は書き換えのみで計算が進む関数型言語モデルであり、定理自動証明や代数的仕様記述などに利用され、そのさまざまな性質について研究されている [2]。

合流性は TRS の性質の一つで、書換えの順序によらず計算結果が一意になるという意味を持つ。しかし、TRS が合流性を持つかどうかは一般的には決定不能であるため、TRS が合流性を満たす十分条件についてさまざまな条件が提案さてきた。特に停止性を持つことが判明すれば危険対補題 [3] により合流性の判定は決定可能である。しかし、停止性がない TRS の場合にはいくつかの十分条件 [4, 5, 6, 7] が存在するものの十分ではない。

一方、TRS が関数記号の重複がなく 2 つの TRS の和に分解（直和分解）できるとき、それぞれの TRS の合流性を証明できれば元の TRS の合流性が証明されるというモジュラー性 [8] も有効である。すなわち、停止性を壊す要因となる規則を直和分解により分離することによって、既存の手法が適用可能になる。

さらに、直和分解不可能な TRS については、ラベル付け [1] により各々の関数記号をその働きにより複数の関数記号に分離し、導出関係が等価な TRS に変換することができる。それにより、直和分解の手法が適用可能となる。

本論文では、TRS の順序ソート付きシグニチャの推論の結果を用い、関数記号に対する有効なラベルの割当てを自動的に行う方法について議論する。ここでの順序ソート付きシグニチャ推論とは、型の情報を持たない TRS から、それと矛盾しない順序つきのシグニチャのうちで最も情報量の多いシグニチャを自動的に推論する手法で、書換え規則によって定まるソートの順序に関する制約条件に基づいて行われる。ラベル割当ては、推論したシグニチャにおいて、極大なソートを持つ関数記号にそれぞれラベルを付けることによって行われる。最後に、この手続きの計算機上への実装について述べる。

2　準備

ここでは項書換え系の基本的な定義について述べる。詳細については文献 [2] を参照されたい。

定義 1 項書換え系 *(Term Rewriting System:TRS)*

関数記号の集合を F、変数の集合を V とする。項の集合 $T(F,V)$ は F と V から再帰的に定義される。以下では項を s,t などで表す。また $Var(t)$ は、t 中に出現する変数の集合を返す関数とする。文脈 $C[\ ]$ は特別な定数□を一つだけ持つ項であり、$C[\ ]$ 中の□を項 t で置き換えて得られる項を $C[t]$ と表す。代入 θ は写像 $V \to T(F,V)$ である。以下では $\theta(x)$ を $x\theta$ と表すことにする。代入は定義域を $T(F,V)$ に自然に拡張される。

項書換え系 R は、書換え規則と呼ばれる $T(F,V)$ の二項組の集合である。ただし、どの規則 $l \to r$ も $l \notin V$ かつ $Var(l) \supseteq Var(r)$ を満たすものとする。R による書換え関係 $\to_R$: $T(F,V) \times T(F,V)$ は以下

のように帰納的に定義される。

$$s \to_R t \Leftrightarrow \begin{cases} \exists l \to r \in R, \sigma \in Sub, s = l\sigma, t = r\sigma \\ \text{または} \quad s' \to_R t', s = f(\ldots, s', \ldots) \\ \qquad\quad , t = f(\ldots, t', \ldots) \end{cases}$$

項書換え系 R に対して、$t_1 \to_R t_2 \to_R \ldots$ なる無限の書換え系列が存在しないとき、R は停止性を持つという。また、$\overset{*}{\leftarrow} \circ \overset{*}{\to} \subseteq \overset{*}{\to} \circ \overset{*}{\leftarrow}$ を満たすとき、R は合流性を持つという。ここで、$\circ$ は関係の合成を表す。

書換え規則 $l_1 \to r_1, l_2 \to r_2$ について、ある $l_1 = C[l']$ なる文脈 $C[\ ]$、代入 θ_1, θ_2 が存在し、$l'\theta_1 = l_2\theta_2$ となるとき、二項組 $\langle r_1\theta, (C\theta_1)[r_2\theta] \rangle$ を危険対と呼ぶ。各左辺に出現する変数は1度ずつしか現れないとき R は左線形という。危険対がなく左線形であるとき、R は直交であると呼ぶ。

3 ラベル付けによる合流性の解析

本節では合流性解析のために用いるラベル付けについて述べる。

3.1 ラベル付けに基づく項書換え系の変換

関数記号の重複のない TRS の和からなる TRS は、モジュラー性 [8] により、直和分解で得られたそれぞれの TRS に分けて合流性を証明することができる。本手法では与えられた TRS に対して、ラベル付け関数によってそれと導出関係が等価な TRS に変形するものである [1]。

0 を含む自然数の集合 L をラベルと呼ぶ。ラベル付き関数記号の集合ならびにラベル付き変数の集合はは $F^L = \{f^i | f \in F, i \in L\}$, $V^L = \{x^i | x \in V, i \in L\}$ である。

定義 2 *(安定性)* 項に対するラベル付け関数 $\varphi_0 : T(F,V) \times L \to T(F^L, V)$、規則に対するラベル付け関数 $\varphi : R \to R$ が与えられるとき、$TRSR$ が φ に対して安定であるとは、$t \to_R s \Leftrightarrow \varphi_0(t) \to_{\varphi(R)} \varphi_0(s)$ が成り立つことである。

TRSR が φ に対して安定であるとき、R の導出関係と $\varphi(R)$ の導出関係は一対一対応になり、R が合流性を満たすことと $\varphi(R)$ が合流性を満たすことは等価になる。

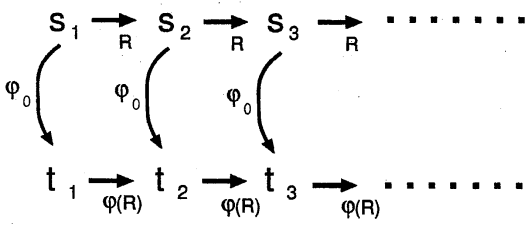

図 3.1: 安定性

すなわち安定性とは、図 3.1のように、ラベル付き規則 $\varphi(R)$ で書き換えた項と、ラベルなし規則 R で書き換えたあとに φ_0 によりラベル付けしたものが同じになることである。

3.2 トップダウンラベル付け

ラベル割当て関数 $\rho : F \to L$ により、各関数記号に対して L の要素の自然数を割り当てる。ラベル付け関数 $\varphi_i : T(F,V) \to T(F \cup F^L, V)$ は、

$$\varphi_i(t) = \begin{cases} x^i \quad (t = x \in V \text{のとき}) \\ f^i(\varphi_i(t_1), \varphi_i(t_2), \ldots, \varphi_i(t_n)) \\ \quad (t = f(t_1, \ldots, t_n), \rho(f) = 0 \text{のとき}) \\ f^j(\varphi_j(t_1), \varphi_j(t_2), \ldots, \varphi_j(t_n)) \\ \quad (t = f(t_1, \ldots, t_n), \rho(f) = j (j \neq 0) \text{のとき}) \end{cases}$$

と定義される。ラベル付き項 $t \in T(F^L, V^L)$ に対して、$\bar{t}$ は、t 中の変数のラベルを取り除いて得られる項を表す。

ラベル付け関数を書換え規則にも拡張し、$\varphi(R) = \{\overline{\varphi_i(l)} \to \overline{\varphi_i(r)} \mid \forall l \to r \in R, i \in L\}$ とする。このようにラベルを付ける手法を、トップダウンラベル付けと呼ぶ。

例 3 *(ラベル付けの例)* ラベル割当て関数を $\rho = \begin{cases} f & \mapsto 1 \\ g & \mapsto 2 \\ h, a & \mapsto 0 \end{cases}$ とする。項 $h(a)$, $g(x)$, $g(f(x))$ に対してそれぞれトップダウンラベル付けを行うと、$\varphi_2(h(a)) = h^2(a^2)$, $\varphi_0(g(x))) = g^2(x^2)$, $\varphi_1(g(f(x))) = g^2(f^1(x^1))$ となる。また、$TRSR = \{h(x) \to a\}$ に対してトップダウンラベル付けを行うと、$\varphi(R) = \{h^0(x) \to a^0, h^1(x) \to a^1, h^2(x) \to a^2\}$ となる。

例 4 *(トップダウンラベル付けでは安定性を満たさない例)* $R = \{g(x) \to g(f(x))\}$ とする。R に対して例 3の ρ を用いてトップダウンラベル付けを行うと、$\varphi(R) = \{g^2(x) \to g^2(f^1(x))\}$ となる。このとき R は φ に対して安定ではない。なぜなら、$g(a) \to_R g(f(a))$ という書換えが行われたとき、図 4のように、ラベル付き規則 $\varphi(R)$ で書き換えた

項と、ラベルなし規則 R で書き換えたあとに φ によりラベル付けしたものが同じならないからである。

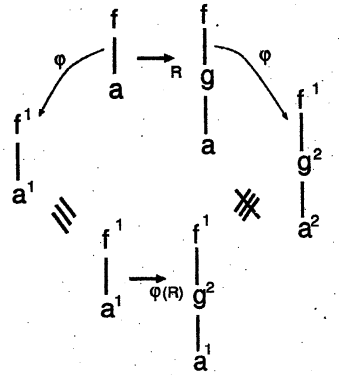

図 *3.2:* 安定性を満たさない例

例4からわかるように、トップダウンラベル付けにおける安定条件は、次の定理が示すように同一規則に出現する変数はすべて同じラベルがつけられることである。

定理 5 *(トップダウンラベル付けに対する安定性)*
項書換え系 R の任意の規則 $l \to r$ について、$\forall k\ \forall x^i, x^j \in (Var(\varphi_k(l) \cup Var(\varphi_k(r))))[i = j]$ が成り立つとする。このとき、R はトップダウンラベル付け関数 φ に対して安定である。

この定理を証明するために補題を用意する。$W(\subseteq V^L)$ は $\forall x^i, x^j \in W[i = j]$ を満たすとき無矛盾であるという。W が無矛盾であるとき、代入 $\sigma : V \mapsto T(F,V)$ を代入 $\sigma' : V \mapsto T(F^L,V)$ に変換する関数 δ を用意する。ここで、
$\begin{cases} x\sigma' = \varphi_j(x\sigma) & x^j \in W\text{のとき} \\ x\sigma' = x & \text{その他のとき} \end{cases}$ とする。

補題 6 $W(\subseteq V^L)$ を無矛盾なラベル付き変数の集合、$\sigma : V \mapsto T(F,V)$ を代入、$\sigma' = \delta(\sigma)$ とする。このとき任意の $i \in L$ について $Var(\varphi_i(s)) \subseteq W$ ならば、$\varphi_i(s\sigma) = \overline{\varphi_i(s)}\sigma'$ が成り立つ。

証明 項の構造に関する帰納法で証明する。
$s = x$ のときは明らかであるので、$t = f(t_1, \ldots, t_n)$ のときを考える。
$\varphi_i(s\sigma)$
$= \varphi_i(f(t_1, \ldots, t_n)\sigma)$
$= \varphi_i(f(t_1\sigma, \ldots, t_n\sigma))$
$= \begin{cases} f^i(\varphi_i(t_1\sigma), \ldots, \varphi_i(t_n\sigma)) & (\rho(f) = 0\text{のとき}) \\ f^j(\varphi_j(t_1\sigma), \ldots, \varphi_j(t_n\sigma)) & (\rho(f) = j \neq 0\text{のとき}) \end{cases}$

- $\rho(f) = 0$ のとき、明らかに $k = 1, \ldots, n$ について $Var(\varphi_i(t_k)) \subseteq W$ であるので帰納法の仮定より $\varphi_i(t_k\sigma) = \overline{\varphi_i(t_k)}\sigma'$ である。

  よって $f^i(\varphi_i(t_1\sigma), \ldots, \varphi_i(t_n\sigma))$
  $= f^i(\overline{\varphi_i(t_1)}\sigma', \ldots, \overline{\varphi_i(t_n)}\sigma')$
  $= f^i(\overline{\varphi_i(t_1)}, \ldots, \overline{\varphi_i(t_n)})\sigma'$
  $= \overline{\varphi^i(f(t_1, \ldots, t_n))}\sigma'$
  $= \overline{\varphi_i(s)}\sigma'$

- $\rho(f) \neq 0$ のときも同様に示される。

□

証明 (定理5の証明) $s \to_R t$ とし、$\varphi_i(s) \to_{\varphi(R)} \varphi_i(t)$ であることを、書換え関係の定義に関する帰納法を用いて証明する。

$s \to_R t$ が $\exists l \to r \in R, \sigma \in Sub, s = l\sigma, t = r\sigma$ であるとき、$Var(\varphi_i(l) \cup \varphi_i(r))$ は無矛盾であり、また $Var(l) \supseteq Var(r)$ より $Var(\varphi_i(l)) \supseteq Var(\varphi_i(r))$ であるので、$Var(\varphi_i(l))$ も無矛盾である。よって $\sigma' = \delta(\sigma)$ とおくと補題6より $\varphi_i(l\sigma) = \overline{\varphi_i(l)}\sigma'$, $\varphi_i(r\sigma) = \overline{\varphi_i(r)}\sigma'$ が成り立つ。$\overline{\varphi_i(l)} \to \overline{\varphi_i(r)} \in \varphi(R)$ であるので、$\varphi_i(s) = \varphi_i(l\sigma) \to_{\varphi(R)} \varphi_i(r\sigma) = \varphi_i(t)$ が示された。

次に、$s = f(\ldots, s', \ldots), t = f(\ldots, t', \ldots)$ かつ $s' \to t'$ を考える。

- $\rho(f) = 0$ のとき、
  $\varphi_i(s) = f^i(\ldots, \varphi_i(s'), \ldots)$ かつ $\varphi_i(t) = f^i(\ldots, \varphi_i(t'), \ldots)$ である。帰納法の仮定により $\varphi_i(s') \to_{\varphi(R)} \varphi_i(t')$ であるから $\varphi_j(s) \to_{\varphi(R)} \varphi_j(t)$ が導かれる。

- $\rho(f) = k \neq 0$ のとき、
  $\varphi_i(s) = f^k(\ldots, \varphi_k(s'), \ldots), \varphi_i(t) = f^k(\ldots, \varphi_k(t'), \ldots)$ である。$\rho(f) = 0$ の場合と同様に $\varphi_i(s) \to_{\varphi(R)} \varphi_i(t)$ が導かれる。

□

4 ラベル付けの自動化

本節では、TRS の順序ソート付きシグニチャ推論を用いて、ラベル割当てを自動的に行う方法について述べる。

4.1 順序ソート付きシグニチャ推論

まず、順序付きシグニチャの定義について述べる。詳細については文献 [10] などを参照されたい。

定義 7 順序ソート付きシグニチャ
S をソートの集合、$<$ を S 上の半順序、F を関数記号の集合、V を変数の集合、$rsort : F \cup V \to S$ を関数記号の返り値のソートあるいは変数のソートを表す関数。$asort : F \times N \to S$ を関数記号の引数のソートを表す関数とする。

このとき、6項組 $\Sigma = (S, <, F, V, rsort, asort)$ を順序ソート付きシグニチャと呼ぶ。以下では $f \in F$, $rsort(f) = s$, $asort(f,1) = s_1$, $\ldots, asort(f,n) = s_n$ のときに $f : s_1 \times \ldots \times s_n \to s$ と表記する。

定義 8 順序ソート付き項

Σ を順序ソート付きシグニチャとする。すべての $s \in S$ について順序ソート付き項の集合 $T_s(F,V) \subseteq T(F,V)$ を以下のように定義する。

1. $x : s$ ならば $x \in T_s(F,V)$
2. $f :\to s_0$ かつ $s_0 \leq s$ ならば $f \in T_s(F,V)$
3. $f : s_1 \times \ldots \times s_n \to s_0, s_0 \leq s$ かつ、すべての i に対して $t_i \in T_{s_i}(F,V)$ ならば $f(t_1, \ldots, t_n) \in T_s(F,V)$

このとき、順序ソート付き項の集合は $T(F,V) = \bigcup_{s \in S} T_s(F,V)$ と定義される。

定義 9 順序ソート付き項書換え系

次の条件をみたすとき項書換え系 R は順序ソート付きシグニチャ Σ に整合するという。R 中の任意の書換え規則 $l \to r$ について、

$$l \in T_s(F,V), r \in T_{s'}(F,V), s \geq s'$$

なるソート s と s' が存在する。このとき R を順序ソート付き項書換え系ともいう。

本研究における、項書換え系の順序ソート付きシグニチャの推論の手法を述べる。それぞれの関数について、項 $t = f(t_1, t_2, \ldots, t_n)$ に対して $rsort(t) = rsort(f)$ と拡張する。

また、$compat(s,t)$ は

$$\begin{cases} \{s \geq rsort(t), s \leq rsort(t)\} & (\text{t が変数のとき}) \\ \{s \geq rsort(t)\} & (\text{その他のとき}) \end{cases}$$

で表される関数である。

まず、与えられた TRS の各々の変数と関数のソートにすべて異なる名前をつける。次に $\mathrm{TRS}(F,R)$ に対して、$(R; \phi; \phi)$ を初期値として、以下の推論規則により $(\phi; \phi; Q)$ となるまで推論する。

$$\begin{cases} ((\{l \to r\} \cup R); T; Q) \\ \Rightarrow (R; \{l, r\} \cup T; compat(rsort(l), r) \cup Q) \quad \cdots(1) \\ (\phi; \{f(t_1, t_2, \ldots, t_m)\} \cup T; Q) \\ \Rightarrow (\phi; T; compat(asort(f,1), t_1) \cup \\ \ldots \cup compat(asort(f,m), t_m) \cup Q) \quad \cdots(2) \end{cases}$$

推論によって得られたソート上の擬順序 Q より定まる同値類を新たにソートとし、同値類間の半順序をソート間の大小関係 $<$ とみなすことによって、順序ソート付きシグニチャ Σ を求めることができる。

例 10 推論の例

$$R = \begin{cases} f(x, g(x)) \to x \\ f(y,y) \to a \end{cases}$$

を考える。推論規則により推論した結果、

$$Q = \left\{ \begin{array}{l} rsort(f) \leq rsort(x), rsort(f) \geq rsort(x), \\ rsort(f) \geq rsort(a) \\ (\text{推論規則 } \textit{(1)} \text{ による}) \\ asort(f,1) \leq rsort(x), asort(f,1) \geq rsort(x), \\ asort(f,2) \geq rsort(g), asort(g,1) \leq rsort(x), \\ asort(g,1) \geq rsort(x), asort(f,1) \leq rsort(y), \\ asort(f,1) \geq rsort(y), asort(f,2) \leq rsort(y), \\ asort(f,2) \geq rsort(y) \\ (\text{推論規則 } \textit{(2)} \text{ による}) \end{array} \right\}$$

となる。シグニチャ Σ は $S = \{s_1, s_2, s_3\} < = \{(s_2, s_1), (s_3, s_1)\}$ $F = \{f : s_1 \times s_1 \to s_1, g : s_1 \to s_2, a :\to s_3\}$ $V = \{x : s_1, y : s_1\}$ となる。

4.2 ラベルの割り当て

次に、推論の結果を、関数記号へのラベル割り当てに適用する2種類の方法について述べる。

順序ソート付きシグニチャ Σ において、極大なソートの集合を S_{max} とする。$\tau : S_{max} \to N$ をソートに対して1から順に番号を割り当てる関数とする。

4.2.1 ラベル付けその1

ラベル割当て関数 ρ を次のように定義する。

$$\rho(f) = \begin{cases} \tau(asort(f,i)) \\ (asort(f,1), \ldots, asort(f,i-1) \notin S_{max} \\ \text{かつ} asort(f,i) \in S_{max} \text{のとき}) \\ 0 \quad (\text{その他のとき}) \end{cases}$$

例 11 *(ラベル付けその1の例)* 例 *10* の R を考える。$S_{max} = \{s_1\}$, $\tau(s_1) = 1$ より、

$$\rho = \begin{cases} f, g \mapsto 1 \\ a \mapsto 0 \end{cases}$$

となる。

4.2.2 ラベル付けその2

ラベル割当て関数 ρ' は、

$$\rho'(f) = \begin{cases} \tau(rsort(f)) & \cdots \quad rsort(f) \in S_{max}\text{のとき} \\ 0 & \cdots \quad \text{その他のとき} \end{cases}$$

例 12 *(ラベル付けその2の例)* 再び例 *10*の R を考える。$s_1 \in S_{max}, \tau(s_1) = 1$ であるため、ラベル付け関数は以下のようになる。

$$\rho' = \begin{cases} f \mapsto 1 \\ g, a \mapsto 0 \end{cases}$$

5 合流性証明システム

5.1 システムの概要

前節のアルゴリズムと、2で述べた従来から広く知られている合流性判定手法を計算機上に実装した。使用言語はC++であり、ソースは約 5000 行程度になった。本システムは、

- 従来の方法による合流性の証明 (sconf)
- ラベル付けによる直和分解 (divide)
- 順序ソート付きシグニチャ推論によるラベル割当て (assign)

の3つのモジュールからなる (図 5.1)。

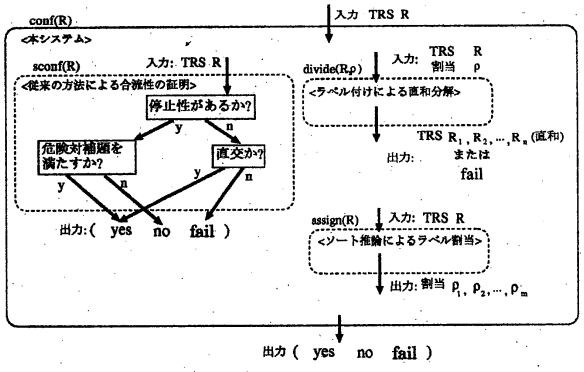

図 5.1: 合流性判定ツール

本システムの実行の流れは以下のようになる。(図 5.2)

1. 従来の方法で合流性が証明できれば、終了する。
2. できなければ、順序ソート付きシグニチャ推論を行い、ラベルの割当てを求める。
3. 2の結果直和分解できるラベル割当てに対して、再帰的にメイン関数を呼び出す。

```
conf(R)=
    switch(sconf(R)){
    case yes : return(yes)
    case no : return(no)
    otherwise : if(∃ρ ∈ assign(R)
                  [[divide(R,ρ) ≠ fail]
                  ∧[∀R' ∈ divide(R,ρ)
                   conf(R') = yes]])
                  return(yes)
    }
```

図 5.2: 合流性判定アルゴリズム

5.2 証明例1

まず以下の TRS を考える。

$$R = \begin{cases} f(x, a(g(x))) \to g(f(x,x)) \\ f(x, g(x)) \to g(f(x,x)) \\ a(x) \to x \\ h(x) \to h(a(h(x))) \end{cases}$$

この TRS は4番目の規則により停止性がなく、1番目と2番目の規則のため直交性をもたない。また、直和分解も不可能である。よって従来の一般的な方法では合流性の判定が困難である。

本システムでは、まず順序ソート付きシグニチャ推論を行う。

```
< rewrite rule >
f(x,a(g(x))) -> g(f(x,x))
f(x,g(x)) -> g(f(x,x))
a(x) -> x
h(x) -> h(a(h(x)))

< label allocation >
auto (a) or manual (m)?
a

< inference >
g : sort6 -> sort1
a : sort2 -> sort2
f : sort6 * sort6 -> sort3
h : sort4 -> sort5
sort6 > sort3
sort2 > sort1
sort2 > sort5
sort6 > sort2
sort6 > sort1
sort3 > sort1
sort4 > sort2

 type1 or type2 (push 1 or 2)
```

図 5.3: 順序ソート付きシグニチャ推論

ここで、$sort1, sort2 \ldots$ は各ソートを表す。$s_1 = rsort(g)$, $s_2 = asort(a,1) = rsort(a)$, $s_3 = rsort(f)$, $s_4 = asort(h,1)$, $s_5 = rsort(h)$, $s_6 = asort(g,1) = asort(f,1) = asort(f,2)$ である。各関数名の右に引数のソートが、$\to$ の右には返り値のソートが表されている。その後に、ソートの大小関係が表されている。

次に、推論結果をラベル付けに適用する。図 5.3 で求めたシグニチャより、極大なソートは $S_{max} = \{s_4, s_6\}$ である。ラベル割り当て関数は、$\rho = \begin{cases} f, g \mapsto 1 \\ h \mapsto 2 \\ a \mapsto 0 \end{cases}$ となる。

```
<result>
f-1(x,a-1(g-1(x))) -> g-1(f-1(x,x))
f-1(x,g-1(x)) -> g-1(f-1(x,x))
a-0(x) -> x
a-1(x) -> x
a-2(x) -> x
h-2(x) -> h-2(a-2(h-2(x)))
```

図 5.4: ラベル付けに対する適用

その結果 $\varphi(R)$ は図 5.4 のようになる。このとき、図中の f-0 は f のラベルが 0 であることを示している。この場合 $\varphi(R)$ は図 5.5 のように 3 つに直和分解される。

```
1
f-1(x,a-1(g-1(x))) -> g-1(f-1(x,x))
f-1(x,g-1(x)) -> g-1(f-1(x,x))
a-1(x) -> x

2
a-0(x) -> x

3
a-2(x) -> x
h-2(x) -> h-2(a-2(h-2(x)))
```

図 5.5: 直和分解

次に、本システムは直和分解したそれぞれの TRS の合流性を従来の手法によって示すことによって $\varphi(R)$ の合流性を示す。その様子を図 5.6 に示す。

```
=1=
f-1(x,a-1(g-1(x))) -> g-1(f-1(x,x))
f-1(x,g-1(x)) -> g-1(f-1(x,x))
a-1(x) -> x

terminating
Critical Pair
(g-1(f-1(x,x)),f-1(x,g-1(x)))
joinable

confluent

=2=
a-0(x) -> x

terminating
Critical Pair dosen't exist

confluent

=3=
a-2(x) -> x
h-2(x) -> h-2(a-2(h-2(x)))

not terminating
left linear
Critical Pair doesn't exist
orthogonal

confluent

Wed@auriga123>
```

図 5.6: 分割証明

その様子を図 5.6 に示す。

6 おわりに

本研究では、トップダウンラベル付けを用いて TRS の順序ソート推論の結果を適用することによって、関数記号に対するラベルの割り当ての自動化を提案した。また、そのアルゴリズムを計算機上に実装し、合流性判定システムを試作した。

文献 [1] では、本論文で用いたラベル付け関数の他にもボトムアップラベル付け関数なども提案されているので、これらのラベル付け関数に有効なラベル付けについても同様な自動化を行っていきたい。また、本手法が適用可能な TRS のクラスについても今後の課題である。

参考文献

[1] 外山芳人、ラベル付けによる項書き換えシステムの解析、LA シンポジューム論文集,46-49,1998.

[2] F.Baader,T.Nipkow:"Term Rewriting and All That", Cambridge Univ. Press,1998.

[3] D.E.Knuth,P.B.Bendix:"Simple word problems in Universal algebra", In J. Leech editor, Computational Problems in Abstrct Algebra, 263-297,1970.

[4] Y.Toyama:"On the Church-Rosser property of term rewriting systems", Technical Report NTT ECL TR 17672, NTT,December 1981.

[5] Y.Toyama:"Commutativity of term rewriting systems", Programming of Future Generation Computers,**2**:393-407,1988.

[6] V.Oostrom:Developing developments,Theoretical Computer Science, **175**159-181,1997

[7] B.Gramlich:"Confluence without termination via parallel critical pairs", Trees in Algebra and Programming - CAAP'96, ,LNCS,**1059**,211-225,1996.

[8] Y.Toyama:"On the Church-Rosser property for the direct sum of term rewriting system", Jounal of ACM, **42**:128-143,1987.

[9] B.K.Rosen:"Tree-manipulating systems and Church-Rosser theorems", Jounal of ACM,**20**:160-187,1973.

[10] U.Waldmann:"Semantics in Order-Sorted Specifications", Theoretical Computer Science,**94**:1-33,1992.