社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

強計算性に基づいた単純型項書換え系の停止性証明法

草刈　圭一朗† 　櫻井　敬大†† 　西田　直樹† 　酒井　正彦† 　坂部　俊樹†

†,†† 名古屋大学大学院情報科学研究科
〒 464-8603 名古屋市千種区不老町
E-mail: †{kusakari,nishida,sakai,sakabe}@is.nagoya-u.ac.jp, ††tsakurai@sakabe.i.is.nagoya-u.ac.jp

あらまし　停止性を持つ関数プログラムは評価を安全に実行できるだけでなく，様々な性質を検証する際に帰納法の土台として簡約関係自体を利用することもできる．それゆえに，関数プログラムの停止性証明法は非常に重要である．本論文では，依存対法と呼ばれる再帰構造解析法を用いた停止性証明法を与える．本手法の特長は，型付き λ 計算の停止性証明で導入された強計算性の概念に基づいていることである．強計算性の概念を用いることにより，我々の再帰構造解析は取り扱いの困難な高階変数を事実上取り扱わなくてよい．そのため，非常に強力でかつ効率的な解析法になっている．

キーワード　関数型プログラム，(直接関数渡し) 単純型項書換え系，停止性，強計算性，依存対法．

On Proving Termination of Simply-Typed Term Rewriting Systems Based on Strong Computability

KUSAKARI Keiichirou†, SAKURAI Takahiro††, NISHIDA Naoki†,

SAKAI Masahiko†, and SAKABE Toshiki†

†,†† Nagoya University, Graduate School of Information Science
Furo-cho, Chikusa-ku, Nagoya, 464–8601 Japan
E-mail: †{kusakari,nishida,sakai,sakabe}@is.nagoya-u.ac.jp, ††tsakurai@sakabe.i.is.nagoya-u.ac.jp

Abstract　Terminating functional programs are safe in execution, and valuable because their reduction relations are suitable for induction schema when verifying various properties. Hence it is important to study methods that prove the termination automatically. In this paper, we present such a method based on the recursive structure analysis called the dependency pair. A feature of our method is that it is constructed based on the strong computability, introduced for proving termination of typed λ-calculus. As a result, we can avoid the influence of higher-order variables, which is difficult to analyze theoretically. It makes our method powerful and efficient.

Key words　Functional Program, (Plain Function-Passing) Simply-Typed Term Rewriting System, Termination, Strong Computability, Dependency Pair.

1. はじめに

関数プログラミング言語とは，計算の対象を関数定義の形で宣言的に記述することによりプログラミングを行う言語である．関数プログラミング言語の特長としては参照透明性や高階関数の導入による抽象化などが挙げられる [2]．関数プログラムに操作的意味を与える計算モデルの 1 つに項書換え系 (Term Rewriting System; TRS) がある [13]．例えば，自然数上の加算に対応する項書換え系は以下のように表される．

$$\{+(x,0) \to x,\ +(x,s(y)) \to s(+(x,y))\}$$

一方，TRS は関数プログラムで広く利用されている高階関数を直接取り扱うことができないため，高階関数を直接取り扱える高階書き換え系の研究が近年盛んに行われている．しかし，高階書換え系の研究は理論的興味が優先されてきたため，関数プログラミング言語のモデルとして見た場合には，不必要と思われる高度な表現力を持つ定式化が行われてきた．この反省にもとづき我々は，理論的に取り扱いやすいように制限しながらも関数プログラムに操作的意味を与えるのに十分な表現力を持つ高階の書き換え系である単純型項書換え系 (simply-typed term rewriting system; STRS) を提案した [7]．例えば自然数上の加算と乗算の STRS R_1, R_2 は本論文の記法では以下のよ

うに表現できる．

$$R_1 = \left\{ \begin{array}{rcl} +[(x,0)] & \to & x \\ +[(x,s[y])] & \to & s[+[(x,y)]] \end{array} \right.$$

$$R_2 = R_1 \cup \left\{ \begin{array}{rcl} \times[(x,0)] & \to & 0 \\ \times[(x,s[y])] & \to & +[(\times[(x,y)],x)] \end{array} \right.$$

ここで，自然数を表す型を N とすると，$+, \times$ は共に $N \times N \to N$ の型を持つ．また，代表的な高階関数である (左) 畳み込み関数 $foldl$ は次の STRS R_3 で与えることができる．

$$R_3 = \left\{ \begin{array}{rcl} foldl[f,y,nil] & \to & y \\ foldl[f,y,cons[(x,xs)]] & \to & foldl[f,f[(x,y)],xs] \end{array} \right.$$

自然数リストを表す型を L とし，$foldl$ は型 $(N \times N \to N) \to N \to L \to N$ を持つとする．関数 $foldl$ を用いると，リスト $[x_1,\ldots,x_n]$ の要素の総和 $(x_1 + \cdots + x_n)$ を与える関数 sum と総積 $(x_1 \times \cdots \times x_n)$ を与える関数 $prod$ を次の STRS R_4, R_5 のように与えることができる．

$$R_4 = R_1 \cup R_3 \cup \{sum \to foldl[+,0]\}$$
$$R_5 = R_2 \cup R_3 \cup \{prod \to foldl[\times, s[0]]\}$$

これらの例のように高階関数を用いると高いレベルの抽象化を実現でき，言語の表現力やプログラムの再利用性を高めることができる．

ところで，上述の STRS R_4, R_5 は停止性を持つだろうか？この事実は直感的に正しいと考えられるが，機械的に検証するのはかなり難しい．この困難は高階変数の理論的取り扱いの難しさに起因する．例として，STRS R_3 を考えてみる．多くのプログラマーは $foldl$ の定義は停止性を持つと考えるであろうが，その理由は再帰定義の部分で第 3 引数 (次式の下線部) が減少し，それゆえに有限回の簡約で出力値が求まると考えるためであろう．

$$foldl[f,y,\underline{cons[(x,xs)]}] \to foldl[f,f[(x,y)],\underline{xs}]$$

しかしながら，この考え方は一般には成立しない．実際，次の関数 $foo : N \times N \to N$ を R_3 に追加すると $foldl$ と foo は相互に再帰呼び出しを繰り返し続け停止性を持たない．

$$R_6 = R_3 \cup \{foo[(x,y)] \to foldl[foo,y,cons[(x,nil)]]\}$$

実際，以下のようなループが発生し停止性を持たない．

$$\begin{array}{rl} foo[(x,y)] & \to\ foldl[foo,y,cons[(x,nil)]] \\ & \to\ foldl[foo,\underline{foo[(x,y)]},nil] \end{array}$$

近年，依存対法と呼ばれる TRS の停止性自動証明法が提案された [1]．依存対法は，関数呼び出しの依存関係を解析することにより再帰構造を解析し停止性を証明する手法である．その後，依存対法は高階の系である STRS 上へ拡張されている [7]．例えば，文献 [7] で提案された手法を用いると R_6 の再帰成分は以下の 3 つと解析される．

$$\left\{ \begin{array}{l} \langle foldl^{\#}[f,y,cons[(x,xs)]], foldl^{\#}[f,f[(x,y)],xs]\rangle \end{array} \right\}$$
$$\left\{ \begin{array}{l} \langle foldl^{\#}[f,y,cons[(x,xs)]], f[(x,y)]\rangle \\ \langle foo^{\#}[(x,y)], foldl^{\#}[foo,y,cons[(x,nil)]]\rangle \end{array} \right\}$$
$$\left\{ \begin{array}{l} \langle foldl^{\#}[f,y,cons[(x,xs)]], foldl^{\#}[f,f[(x,y)],xs]\rangle\} \\ \langle foldl^{\#}[f,y,cons[(x,xs)]], f[(x,y)]\rangle \\ \langle foo^{\#}[(x,y)], foldl^{\#}[foo,y,cons[(x,nil)]]\rangle \end{array} \right\}$$

文献 [7] で提案された手法で停止性を検証する際には，$foldl$ の第 3 引数のみの解析では不十分で第 2 引数を無視することが出来ず，取り扱いの困難な高階関数の解析が必要になる．具体的には $foldl[f,y,cons[(x,xs)]] > f[(x,y)]$ となる適切な順序 $>$ を設計する必要が生ずる．再帰構造解析を動的に行う (高階変数 f に代入されうる全ての関数を解析する) ことにより，この問題を解決することは可能ではある．しかしながら，このような動的解析法は自動証明を行う際の計算量を増大させるばかりでなく，関数を追加する度に再解析を行わないといけないため逐次証明手法の適用が困難である．

本研究で提案する停止性証明法も，依存対法に基づいて設計されている．我々の手法の最大の特徴は，既存の成果と違い強計算性と呼ばれる概念に基づき設計されていることである．ここで，強計算性とは直感的には関数としての停止性に対応する概念，すなわち任意の適切な入力に対して関数が停止するという性質のことである．強計算性は型付き λ 計算の停止性証明のために導入された概念である [3], [12]．最初の高階書き換え系への導入は Jouannaud と Rubio により与えられた高階経路順序の設計である [6]．この成果は Raamsdonk によって再定式化され [10]，STRS においても適用可能であることが示されている [8]．さらに，我々は強計算性の概念を依存対法と組み合わせた十分完全性証明法を提案している [11]．

本論文で提案する手法は文献 [11] の手法を全面的に再構築することにより，事実上，高階変数を取り扱わなくて良い停止性証明法である．例えば，本研究の枠組みでは，R_6 の再帰成分は以下の 2 つと解釈される．

$$\{\langle foldl^{\#}[f,y,cons[(x,xs)]], foldl^{\#}[f,f[(x,y)],xs]\rangle\}$$
$$\{\langle foo^{\#}[(x,y)], foo^{\#}[z]\rangle\}$$

このように，理論的に取り扱いが困難な高階変数の解析 (本例では $\langle foldl^{\#}[f,y,cons[(x,xs)]], f[(x,y)]\rangle$ の部分の解析) を必要とせず，また，再帰成分の個数が格段に減少している．それゆえ，我々の手法は非常に効率的でかつ効果的な手法である．

2. 準　　備

本節では文献 [7] に基づき，論文中で必要となる単純型項書換え系に関する諸概念を与える(注1)

関数記号の集合 Σ と変数記号の集合 $\mathcal{V}$ から生成される**項**の全体からなる集合 $\mathcal{T}(\Sigma,\mathcal{V})$ は次のように帰納的に定義される；$a \in \Sigma \cup \mathcal{V}$ かつ $t_1,\ldots,t_n \in \mathcal{T}(\Sigma,\mathcal{V})$ ならば，

(注1)：なお，本論文では文献 [7] では取り扱わなかった直積型の概念を導入し項の記法を少し変更してある ($foldl(f,y,nil)$ を $foldl[f,y,nil]$ に変更)．

$a[t_1,\ldots,t_n] \in \mathcal{T}(\Sigma,\mathcal{V})$．項 $a[\,]$ は単に a と記す．2つの項 s と t が構文的に等しいことを $s \equiv t$ で表す．また，$s \equiv a[s_1,\ldots,s_n]$ とするとき，$s[t_1,\ldots,t_m]$ と書いて項 $a[s_1,\ldots,s_n,t_1,\ldots,t_m]$ を表す．項 $t \equiv a[t_1,\ldots,t_n]$ に対し，$t|_i$ で t_i を記す．項 t に出現する全ての変数の集合を $Var(t)$ で表す．項 $t \equiv a[t_1,\ldots,t_n]$ の根の位置の記号 a を $root(t)$ で記す．

代入は変数から項への関数である．代入 θ の項上への拡張 $\hat{\theta}$ を以下で定義する; $\theta(a[t_1,\ldots,t_n])$ は $a \in \Sigma$ のときには $a[\hat{\theta}(t_1),\ldots,\theta(\hat{t_n})]$ で $a \in \mathcal{V}$ のときには $\theta(a)[\hat{\theta}(t_1),\ldots,\hat{\theta}(t_n)]$ で定義される．以下では，簡便のため θ と $\hat{\theta}$ を同一視する．また，$\theta(t)$ を $t\theta$ で略記する．**文脈**とは穴と呼ばれる特別な関数記号 $\Box$ を 1 つだけ持つ項である．文脈 $C[\,]$ 中の $\Box$ を項 t で置き換えることによって得られる項を $C[t]$ で表す．特に，葉の位置に $\Box$ を持つ文脈を**葉文脈**と呼び，根の位置に $\Box$ を持つ文脈を**根文脈**と呼ぶ．例えば，$s[\Box], foldl[y,\Box]$ は葉文脈で，$\Box[0], \Box[y,nil]$ は根文脈で，$\Box$ は葉文脈であると同時に根文脈でもある．項 t' が項 t の**部分項**であるとは，ある葉文脈 $C[\,]$ が存在して $t \equiv C[t']$ となることである．このとき，$t \geqq_{sub} t'$ と記す．また，項 t の全ての部分項からなる集合を $Sub(t)$ で記す．

空でない**基本型**の集合を $\mathcal{B}$ で表す．基本型 $\mathcal{B}$ から生成される**単純型**の集合 $\mathcal{S}$ は，型構成子 $\rightarrow, \times$ を用いて $\mathcal{S} ::= \mathcal{B} \mid (\mathcal{S} \rightarrow \mathcal{S}) \mid (S_1 \times \cdots \times S_n)$ として定義される．ここで，型構成子 $\rightarrow$ は右結合性を持ち $\times$ より結合が弱いとし，括弧の省略を適宜行う．$\alpha \rightarrow \beta$ の形の単純型を**関数型**と呼び，それ以外の単純型を**非関数型**と呼ぶ．**型関数** τ は $\Sigma \cup \mathcal{V}$ から $\mathcal{S}$ への関数である．**組化構成子**と呼ぶ特別な関数記号 $\mathrm{tp} \in \Sigma$ の存在を仮定する．項 $t \equiv a[t_1,\ldots,t_n] \in T(\Sigma,\mathcal{V})$ が単純型 α を持つとは，各 t_i が単純型 α_i を持ち，$a = \mathrm{tp}$ ならば $\alpha = \alpha_1 \times \cdots \times \alpha_n$ $(n \geqq 2)$，そうでないならば $\tau(a) = \alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow \alpha$ となることである．単純型を持つ項を**単純型項**と呼ぶ．なお項 $\mathrm{tp}[t_1,\ldots,t_n]$ を $(t_1,\ldots,t_n)$ で記すこともある．単純型・基本型・関数型・非関数型を持つ項からなる集合をそれぞれ $\mathcal{T}_\tau(\Sigma,\mathcal{V})$, $\mathcal{T}_{\mathcal{B}}(\Sigma,\mathcal{V})$ $\mathcal{T}_{fun}(\Sigma,\mathcal{V})$, $\mathcal{T}_{nfun}(\Sigma,\mathcal{V})$ で記す．簡便のため，単純型項を単に項と呼ぶこともある．$\mathcal{V}_h$ で高階変数，すなわち，$\alpha \rightarrow \beta$ の形の単純型を持つ変数全てからなる集合を記す．なお本論文中では，代入や文脈は型の整合性を崩さないもののみを扱う．すなわち，単純型項 t に対し $t\theta$ や $C[t]$ のように書いた場合には，$t\theta, C[t]$ は共に単純型を持つと仮定する．

単純型書き換え規則とは $root(l) \in \Sigma$, $Var(l) \supseteqq Var(r)$, $\tau(l) = \tau(r)$ の条件を満たす単純型項の対 (l,r) であり，$l \rightarrow r$ と記す．**単純型項書換え系** (simply-typed term rewriting system; STRS) とは単純型書き換え規則の集合である．STRS R において単純型項 s が t に書き換えられるとは，ある規則 $l \rightarrow r \in R$ と代入 θ と文脈 $C[\,]$ が存在して $s \equiv C[l\theta]$ かつ $t \equiv C[r\theta]$ となることであり，$s \rightarrow_R t$，または単に $s \rightarrow t$ と記す．このとき書き換えられた項 $l\theta$ を**リデックス**と呼ぶ．リデックスを含まない項を**正規形**と呼び，STRS R における正規形の集合と非関数型を持つ正規形の集合をそれぞれ $NF(R), NF_{nfun}(R)$ で記す．特に，$s \rightarrow_R t$ において，最内のリデックス (真部分項にリデックスを持たないリデックス) を書き換えた場合には $s \underset{in}{\longrightarrow}_R t$ と記す．

項 t が STRS R において**停止性**または**強正規性**を持つとは ($SN(t)$ で記す)，t から始まる $\rightarrow_R$ の無限列が存在しないことである．項 t が STRS R において**最内停止性**または**強最内正規性**を持つとは ($SIN(t)$ で記す)，t から始まる $\underset{in}{\longrightarrow}_R$ の無限列が存在しないことである．任意の項 t に対して $SN(t)$ が成立するとき $SN(R)$ と記し，任意の項 t に対して $SIN(t)$ が成立するとき $SIN(R)$ と記す．与えられた STRS R の下で定義される項上の述語 P に対して，$T_P(R) = \{t \mid P(t)\}$ と $T_P^{args}(R) = \{a[t_1,\ldots,t_n] \mid \bigwedge_i P(t_i)\}$ を定義する．

2.1 依存対法

本節では，文献 [7] で STRS 上へ拡張された依存対法の基本成果を説明する．なお，文献 [7] の 7 節で取り扱った STRS は，その規則を基本型に制限していたが，本節を含めた本論文ではそのような制限を用いていない．以下で紹介する成果は文献 [7] の 5 節の成果に対応している．

［定義 2.1］ R を STRS とする．R の規則 $l \rightarrow r$ の左辺 l の根記号 ($root(l)$) を**被定義記号**と呼び，R の被定義記号全体からなる集合を $\mathcal{D}_R$ と記す．また，R の被定義記号でない関数記号を R の**構成子**と呼び，R の構成子全体からなる集合を $\mathcal{C}_R$ で記す．各関数記号 $f \in \mathcal{D}_R$ に対し，**印付記号** $f^\#$ を用意する．また，$t^\#$ で t の根記号を対応する印付記号で置き換えた項を表す．なお，$root(t) \in \mathcal{V}$ の場合は $t^\# \equiv t$ と考える．

［定義 2.2］（[7]） R を STRS とする．$l \rightarrow v \in R$ かつ $root(v) \in \mathcal{D}_R \cup \mathcal{V}_h$ であるとき，対 $\langle l^\#, v^\# \rangle$ を**外依存対**と呼ぶ．空でない葉文脈 $C[\,]$ に対し $l \rightarrow C[v] \in R$ かつ $root(v) \in \mathcal{D}_R \vee (root(v) \in \mathcal{V}_h \wedge v \notin \mathcal{V})$ であるとき，対 $\langle l^\#, v^\# \rangle$ を**内依存対**と呼ぶ．外依存対と内依存対を合わせて**依存対**と呼ぶ．

［定義 2.3］（[7]） 依存対の列 $\langle u_0^\#, v_0^\# \rangle \langle u_1^\#, v_1^\# \rangle \cdots$ が**依存鎖**であるとは，各 i で，代入 θ_i と根文脈 $S_i[\,]$ が存在して以下のいずれか一方が成立することである．

- $\langle u_i^\#, v_i^\# \rangle$ が外依存対かつ
  $\mathcal{T}_{SN}^{args}(R) \ni S_i[v_i\theta_i]^\# \xrightarrow{*}_R S_{i+1}[u_{i+1}\theta_{i+1}]^\# \in T_{SN}^{args}(R)$,
- $\langle u_i^\#, v_i^\# \rangle$ が内依存対かつ
  $\mathcal{T}_{SN}^{args}(R) \ni v_i\theta_i^\# \xrightarrow{*}_R S_{i+1}[u_{i+1}\theta_{i+1}]^\# \in T_{SN}^{args}(R)$.

［命題 2.4］（[7]） STRS R が停止性を持たないことと無限依存鎖の存在は必要十分である．

以下では，本論文の成果と比較しやすいようにこれらの成果を再定式化する．具体的には，根文脈を依存鎖の定義に用いる代りに，依存対の定義に組み入れる．

［定義 2.5］（SN-**依存対**） R を STRS とする．$l \rightarrow v \in R$ かつ $root(v) \in \mathcal{D}_R \cup \mathcal{V}_h$ であるとき，対 $\langle S[l]^\#, S[v]^\# \rangle$ を SN-**外依存対**と呼び，空でない葉文脈 $C[\,]$ に対し $l \rightarrow C[v] \in R$ かつ $root(v) \in \mathcal{D}_R \vee (root(v) \in \mathcal{V}_h \wedge v \notin \mathcal{V})$ であるとき，対 $\langle S[l]^\#, v^\# \rangle$ を SN-**内依存対**と呼ぶ．ここで，$S[\,]$ はフレッシュ変数のみで構成される根文脈 (例えば $\Box[z_1,z_2]$) である．外依存対と内依存対を合わせて SN-**依存対**と呼び，R の SN-依存対の全体を $DP_{SN}(R)$ で記す．

［定義 2.6］（**依存鎖**） P を項の対の集合とし，T_l, T_r を項の集合とする．P の列 $\langle u_0, v_0\rangle\langle u_1, v_1\rangle\cdots$ が (T_l, T_r) 上の $\langle P, \gg\rangle$-依存鎖であるとは，各 i で，ある代入 θ_i が存在して $u_i\theta_i \in T_l$，$v_i\theta_i \in T_r$，$v_i\theta_i \gg^* u_{i+1}\theta_{i+1}$ が成立することである．$T_l = T_r$ のときは単に T_l 上の $\langle P, \gg\rangle$-依存鎖と呼ぶ．

文献 [7] で導入された依存鎖 (定義 2.3) は，$T^{ary s}_{SN}(R)$ 上の $\langle DP_{SN}(R), \to_R\rangle$-依存鎖と同値である．それゆえに，命題 2.4 から次の系が得られる．

［系 2.7］ STRS R が停止性を持たないことと $T^{ary s}_{SN}(R)$ 上の無限 $\langle DP_{SN}(R), \to_R\rangle$-依存鎖の存在は必要十分である．

［例 2.8］ 次で与えられる STRS R_7 を考える．

$$R_3 \cup \left\{ \begin{array}{l} foo[s[i], (x, y)] \to foldl[foo[i], y, cons[(x, nil)]] \\ apply[f] \to f \end{array} \right\}$$

なお，関数 foo の定義は，停止性を持つように R_6 中の定義から少し変形してある．ここで，$foo : N \to N \times N \to N$，$apply : (N \to N) \to N \to N$ であるとする．このとき $DP_{SN}(R_7)$ は以下のようになる．

$$\left\{ \begin{array}{l} \langle foldl^\#[f, y, cons[(x, xs)]], foldl^\#[f, f[(x, y)], xs]\rangle \\ \langle foldl^\#[f, y, cons[(x, xs)]], f[(x, y)]\rangle \\ \langle foo^\#[s[i], (x, y)], foldl^\#[foo[i], y, cons[(x, nil)]]\rangle \\ \langle foo^\#[s[i], (x, y)], foo^\#[i]\rangle \\ \langle apply^\#[f], f\rangle \\ \langle apply^\#[f, z], f[z]\rangle \end{array} \right.$$

文献 [7] では文献 [1] で導入された依存グラフの概念を導入していないため，以下で導入しておく．

［定義 2.9］（**依存グラフ**） P を項の対の集合とし，T_l, T_r を項の集合とする．(T_l, T_r) 上の $\langle P, \gg\rangle$-依存グラフとは有向グラフであり，その点集合は P で，$\langle u_i^\#, v_i^\#\rangle\langle u_j^\#, v_j^\#\rangle$ が (T_l, T_r) 上の $\langle P, \gg\rangle$-依存鎖であるとき弧があると定義される．また，$T_l = T_r$ のときは単に T_l 上の $\langle P, \gg\rangle$-依存グラフと呼ぶ．

［定義 2.10］（**再帰成分**） P を項の対の集合とし，T_l, T_r を項の集合とする．(T_l, T_r) 上の $\langle P, \gg\rangle$-再帰成分 (recursion component) とは，(T_l, T_r) 上の $\langle P, \gg\rangle$-依存グラフの，ある強連結部分グラフ中の点集合の事である．また，$T_l = T_r$ のときは単に T_l 上の $\langle P, \gg\rangle$-再帰成分と呼ぶ．特に，全ての $T^{ary s}_{SN}(R)$ 上の $\langle DP_{SN}(R), \to_R\rangle$-再帰成分からなる集合を $RC_{SN}(R)$ で記す．

［例 2.11］ $RC_{SN}(R_7)$ は以下の 5 個の再帰成分からなる．

$$\left\{ \langle foldl^\#[f, y, cons[(x, xs)]], foldl^\#[f, f[(x, y)], xs]\rangle \right\}$$
$$\left\{ \begin{array}{l} \langle foldl^\#[f, y, cons[(x, xs)]], f[(x, y)]\rangle \\ \langle foo^\#[s[i], (x, y)], foldl^\#[foo[i], y, cons[(x, nil)]]\rangle \end{array} \right\}$$
$$\left\{ \begin{array}{l} \langle foldl^\#[f, y, cons[(x, xs)]], foldl^\#[f, f[(x, y)], xs]\rangle\} \\ \langle foldl^\#[f, y, cons[(x, xs)]], f[(x, y)]\rangle \\ \langle foo^\#[s[i], (x, y)], foldl^\#[foo[i], y, cons[(x, nil)]]\rangle \end{array} \right\}$$
$$\left\{ \langle apply^\#[f], f\rangle \right\}$$
$$\left\{ \langle apply^\#[f, z], f[z]\rangle \right\}$$

［定理 2.12］ R を STRS とする．各 $C \in RC_{SN}(R)$ に対し，ある簡約化対 $(\gtrsim, >)$ が存在して $R \cup C \subseteqq \gtrsim$ かつ $C \cap > \neq \emptyset$ ならば $SN(R)$ が成立．

ここで，擬順序 $\gtrsim$ と狭義の半順序 $>$ の対 $(\gtrsim, >)$ が**簡約化対**であるとは，$\gtrsim$ が文脈と代入に閉じ，$>$ が整礎かつ代入に閉じていることである．簡約化対の設計には切り落とし法が利用される [1], [7]．以下では文献 [7] で STRS 上へ拡張された切り落とし法を紹介する．

［定義 2.13］（**切り落とし法** [7]） 切り落とし関数 π は，各 $f \in \Sigma$ $(\tau(f) = \alpha_1 \to \cdots \to \alpha_n \to \beta)$ を $i_1 < \cdots < i_m \leqq n$ となる正整数のリスト $[i_1, \ldots, i_m]$ に割り当てる関数である．$\pi(f)^{\leqq k}$ で $i_j \leqq k$ を満たす $\pi(f)$ の最大部分リスト $[i_1, \ldots, i_j]$ を表す．項 $a[t_1, \ldots, t_n]$ に対し $\pi(a[t_1, \ldots, t_n])$ は以下で定義される．

$$\left\{ \begin{array}{ll} a[\pi(t_1), \ldots, \pi(t_n)] & \text{if } a \in \mathcal{V} \\ a[\pi(t_{i_1}), \ldots, \pi(t_{i_m})] & \text{if } \pi(a)^{\leqq n} = [i_1, \ldots, i_m] \end{array} \right.$$

狭義の半順序 $>$ に対し，$s \gtrsim_\pi t$ を $\pi(s) \geqq \pi(t)$ で定義し，$s >_\pi t$ を $\pi(s) > \pi(t)$ で定義する．

1 階の TRS においては任意の簡約化順序 $>$(文脈と代入に閉じた整礎な狭義の半順序) に対し $(\gtrsim_\pi, >_\pi)$ は簡約化対になるが [1]，高階の STRS においては，$\gtrsim_\pi$ が簡約化対であるためには適切な制限が必要となる [7], [8] (本論文中の例題は全てこの制限を満たしている)．

3. 強計算性に基づく停止性証明法

本節では強計算性の概念に基づく依存対法を提案する．

［定義 3.1］（**強計算性**） 項 t が強計算性 (strong computability) を持つ ($SC(t)$ で記す) とは，以下のように定義される．

- $\tau(t)$ が非関数型ならば $SN(t)$,
- $\tau(t) = \alpha \to \beta$ ならば，任意の $SC(u)$ かつ $\tau(u) = \alpha$ となる u に対し $SC(t[u])$.

強計算性の概念が単純型の構成に関する帰納的定義によって与えられていることに注意されたい．定義より直ちに次の補題が導かれる．

［補題 3.2］

(1) $SC(t)$ ならば $SN(t)$.
(2) $SC(t)$ かつ $t \xrightarrow{*} t'$ ならば $SC(t')$.
(3) $\bigwedge_i SC(t_i)$ ならば $SC(t_0[t_1, \ldots, t_n])$ も成立.

［補題 3.3］ $SN(R)$ iff $\mathcal{T}_{nfun}(\Sigma, \mathcal{V}) \cap T_{\neg SC}(R) \cap T^{ary s}_{SC}(R) = \emptyset$

［定義 3.4］（SC-**依存対**） R を STRS とする．R の η 拡大 R^η を $\{S[l] \to S[r] \mid l \to r \in R\}$ で定義する．ここで，$S[\,]$ はフレッシュ変数のみで構成される非関数型を持つ根文脈である．ある葉文脈 $C[\,]$ が存在して $l \to C[v] \in R^\eta$ かつ $root(v) \in \mathcal{D}_R$ であるとき，対 $\langle l^\#, S[v^\#]\rangle$ を R の SC-依存対と呼ぶ．ここで，$S[\,]$ はフレッシュ変数のみで構成される非関数型を持つ根文脈である．SC-依存対の全体を $DP_{SC}(R)$ で記す．

［例 3.5］ $DP_{SC}(R_4)$ は以下のようになる．

$$\left\{ \begin{array}{l} \langle +^\#[(x, s[y])], +^\#[(x, y)]\rangle \\ \langle foldl^\#[f, y, cons[(x, xs)]], foldl^\#[f, f[(x, y)], xs]\rangle \\ \langle sum^\#[zs], foldl^\#[+, 0, zs]\rangle \\ \langle sum^\#[zs], +^\#[z]\rangle \end{array} \right.$$

$DP_{SC}(R_7)$ は以下のようになる．

$$\left\{\begin{array}{l}\langle foldl^{\#}[f,y,cons[(x,xs)]], foldl^{\#}[f,f[(x,y)],xs]\rangle\\ \langle foo^{\#}[s[i],(x,y)], foldl^{\#}[foo[i],y,cons[(x,nil)]]\rangle\\ \langle foo^{\#}[s[i],(x,y)], foo^{\#}[i,z]\rangle\end{array}\right.$$

$DP_{SC}(R_7)$ を例 2.8 で与えた $DP_{SN}(R_7)$ と比較すると要素数が 6 個から 3 個に半減し，さらに，取り扱いの困難な右辺が高階変数となっている対

$$\langle foldl^{\#}[f,z,cons[(x,xs)]], f[(z,x)]\rangle$$

が $DP_{SC}(R)$ に含まれていない．また，$DP_{SC}(R)$ の元は全て非関数型項の対となっていることが分かる．

本論文で提案する再帰構造解析法を設計する上での最大の困難は項の構造と型の構造の両立であった．これは，再帰構造解析が項の構造に基づくのに対し，強計算性の概念が型の構造に基づいて与えられているためである．例として項 $s \equiv foldl[f,y]$ と項 $t \equiv foldl[f,y,nil]$ を比較する．ここで項 s の型は $L \to N$，項 t の型は N である．すなわち，単純に構造で比較した場合は，項上での比較では $s < t$，型上での比較では $s > t$ となり両立しない．実際，我々の再帰構造解析法は一般の STRS には適用できず適切な制限が必要となる．このための適切な制限として直接関数渡し STRS の概念を与える．

［定義 3.6］（**直接関数渡し**） STRS R が直接関数渡し (plain function-passing; PFP) であるとは，任意の葉文脈 $C[\,]$ と規則 $l \to C[a'[r_1,\ldots,r_m]] \in R$ に対し，$a' \in V_h$ のときはいつでも，ある k $(0 \leqq k \leqq m)$ と i が存在して $a'[r_1,\ldots,r_k] \equiv l|_i$ となることである．以下では，直接関数渡しである STRS を PFP-STRS で略記する．

本論文で与える全ての STRS の例において，出現する全ての高階変数が左辺の引数として直接出現しているので，これらは全て PFP である．さらに，通常の関数プログラムのほぼ全ては PFP であり，このことは，我々の手法の汎用性を示すものである．

［補題 3.7］ R を PFP-STRS とする．このとき，任意の $t \in \mathcal{T}_{nfun}(\Sigma,\mathcal{V}) \cap T_{\neg SC}(R) \cap T^{args}_{SC}(R)$ に対し，ある $\langle u^{\#},v^{\#}\rangle \in DP_{SC}(R)$ と代入 θ が存在して $t^{\#} \xrightarrow{*} (u\theta)^{\#}$ かつ $u\theta, v\theta \in \mathcal{T}_{nfun}(\Sigma,\mathcal{V}) \cap T_{\neg SC}(R) \cap T^{args}_{SC}(R)$.

［証明］ $\tau(t) \in \mathcal{T}_{nfun}(\Sigma,\mathcal{V})$ より $\neg SN(t)$．補題 3.2(1) より各 i で $SN(t|_i)$．よって，ある $l \to r \in R^{\eta}$ と θ' が存在して $t^{\#} \xrightarrow{*} (l\theta')^{\#}$ かつ $\neg SN(r\theta')$．$\tau(r\theta')$ は非関数型なので $\neg SC(r\theta')$ が成立し，$\{r' \in Sub(r) \mid \neg SC(r'\theta')\} \neq \emptyset$．この集合のサイズ最小の項の 1 つを $r' \equiv a[r_1,\ldots,r_m]$ とする．r' の最小性より各 i で $SC(r_i\theta')$ が成立する．

$a = \mathrm{tp}$ と仮定すると，定義より $SC(r'\theta')$ が成立するので矛盾．よって，$a \neq \mathrm{tp}$．$\neg SC(r'\theta')$ より，$\tau(r'\theta'[v_1,\ldots,v_k])$ が非関数型でかつ $\neg SN(r'\theta'[v_1,\ldots,v_k])$ を満たす $v_1,\ldots,v_k \in T_{SC}(R)$ が存在する．$z_1,\ldots,z_k$ をフレッシュ変数とし，$v \equiv a[r_1,\ldots,r_m,z_1,\ldots,z_k]$ とする．代入 θ を，各 i で $\theta(z_i) = v_i$ かつ，それ以外の場合は $\theta(x) = \theta'(x)$ で定義する．補題 3.2(2) より各 i で $SC(l|_i\theta')$ であるので，$l\theta \in \mathcal{T}_{nfun}(\Sigma,\mathcal{V}) \cap T_{\neg SC}(R) \cap T^{args}_{SC}(R)$ が成立する．また，$v\theta \in \mathcal{T}_{nfun}(\Sigma,\mathcal{V}) \cap T_{\neg SC}(R) \cap T^{args}_{SC}(R)$ も成立する．最後に $a \in \mathcal{D}_R$ を示す．これが示せると $\langle l^{\#},v^{\#}\rangle \in DP_{SC}(R)$ となり題意が成立する．

$a \in \mathcal{V}_h$ と仮定する．各 i で $SC(l|_i\theta)$ が成立し R が PFP であるので，補題 3.2(3) より $SC(a\theta[r_1\theta,\ldots,r_m\theta,v_1,\ldots,v_k])$ となり矛盾．$a \in \mathcal{V} \setminus \mathcal{V}_h$ と仮定する．このとき，$\tau(a)$ が非関数型なので $r'\theta[v_1,\ldots,v_k] \equiv a\theta$．ある i で $a\theta \in Sub(l|_i\theta)$．$SN(l|_i\theta)$ より $SN(a\theta)$ となるので矛盾．$a \in \mathcal{C}_R$ と仮定する．補題 3.2(1) より $r_1\theta,\ldots,r_m\theta,v_1,\ldots,v_k$ は全て停止性を持つので $SN(a[r_1\theta,\ldots,r_m\theta,v_1,\ldots,v_k])$ となり矛盾．以上より $a \in \mathcal{D}_R$ が示された． □

［定理 3.8］ R を PFP-STRS とする．$T^{args}_{SC}(R) \cap \mathcal{T}_{nfun}(\Sigma,\mathcal{V})$ 上の無限 $\langle DP_{SC}(R), \to_R\rangle$-依存鎖が存在しないならば $SN(R)$ が成立．

［証明］ $\neg SN(R)$ と仮定する．補題 3.3 より $\mathcal{T}_{nfun}(\Sigma,\mathcal{V}) \cap T_{\neg SC}(R) \cap T^{args}_{SC}(R) \neq \emptyset$．よって，補題 3.7 を繰り返し適用することにより $T^{args}_{SC}(R) \cap \mathcal{T}_{nfun}(\Sigma,\mathcal{V})$ 上の無限 $\langle DP_{SC}(R), \to_R\rangle$-依存鎖を得ることができ矛盾． □

［系 3.9］ R を PFP-STRS とする．ある簡約化対 $(\gtrsim, >)$ が存在して $R \subseteqq \gtrsim$ かつ $DP_{SC}(R) \subseteqq >$ となるならば $SN(R)$ が成立．

以下では，全ての $T^{args}_{SC}(R)$ 上の $\langle DP_{SC}(R), \to_R\rangle$-再帰成分からなる集合を $RC_{SC}(R)$ で記す．

［系 3.10］ R を PFP-STRS とする．全ての $C \in RC_{SC}(R)$ に対し，ある簡約化対 $(\gtrsim, >)$ が存在して $R \cup C \subseteqq \gtrsim$ かつ $C \cap > \neq \emptyset$ となるならば $SN(R)$ が成立．

［例 3.11］ $RC_{SC}(R_7)$ は以下の 2 個の再帰成分から構成される．

$$\{\langle foldl^{\#}[f,y,cons[(x,xs)]], foldl^{\#}[f,f[(x,y)],xs]\rangle\}$$
$$\{\langle foo^{\#}[s[i],(x,y)], foo^{\#}[i,z]\rangle\}$$

例 2.11 で与えた $RC_{SN}(R_7)$ と比べて格段に少なく単純になっている．

残念ながら系 3.10 だけでは R_7 の停止性は示せない．なぜならば，現時点では $R \subseteqq \gtrsim$ の制約を満たす簡約化対，すなわち $foldl[f,y,nil] \gtrsim y$ と $foldl[f,y,cons[x,xs]] \gtrsim foldl[f,f[(x,y)],xs]$ を同時に満たす簡約化対の設計法が知られていないからである．この困難は Hirokawa と Middeldorp が文献 [5] において，TRS 上で提案した部分項基準の概念を用いることにより解決できる．

［定義 3.12］（**部分項基準**） $C \in RC_{SC}(R)$ とする．任意の $\langle a^{\#}[l_1,\ldots,l_n], a'^{\#}[r_1,\ldots,r_m]\rangle \in C$ について $l_{\phi(a)} >_{sub} r_{\phi(a')}$ を満たす関数 $\phi : \mathcal{D}_R \to N$ が存在するとする．このとき，C は部分項基準 (subterm criterion) を満たすという．

［定理 3.13］ R を PFP-STRS とする．全ての $C \in RC_{SC}(R)$ が部分項基準を満たすならば R は停止性を持つ．

［証明］ STRS においても TRS と同様に $>_{sub} \cdot \to_R \subseteqq \to_R \cdot >_{sub}$ の関係が成立する．よって，定理 3.8 に基づき文献 [5] と同様に証明できる． □

［例 3.14］ 全ての $C \in RC_{SC}(R_7)$ は部分項基準を満たすので，定理 3.13 に基づき R_7 の停止性を示すことができる．実際，$\phi(foldl) = 3$ とすることにより再帰成

分 $\{\langle foldl^{\#}[f, y, cons[(x, xs)]], foldl^{\#}[f, f[(x, y)], xs]\rangle\}$ は部分項基準を満たし，$\phi(foo) = 1$ とすることにより再帰成分 $\{\langle foo^{\#}[s[i], (x, y)], foo^{\#}[i, z]\rangle\}$ は部分項基準を満たす．

4. 強最内計算性に基づく停止性証明法

部分項基準 (定理 3.13) は効率的ではあるが，部分項関係しか比較に利用できないため汎用性が低い．例えば，R_7 を少し変更した次の R_8 を考える．

$$R_3 \cup \left\{ \begin{array}{l} foo[tt, (x, y)] \to foldl[ff, y, cons[(x, nil)]] \\ apply[f] \to f \end{array} \right\}$$

ここで，foo の第 1 引数を見ると，R_7 では $s[i] >_{sub} i$ のように部分項関係を持つが，R_8 では $tt \not>_{sub} ff$ のように部分項関係を持たない．よって，この僅かな変更により定理 3.13 は適用できなくなってしまう．

本節では，文献 [11] で提案された最内停止性と堅固構成子項の概念を利用した停止性証明法を与える．最初に，強最内計算性，すなわち，最内書き換えに関する強計算性を定義 3.1 と同様に与える．

［定義 4.1］（**強最内計算性**） 項 t が強最内計算性 (strong inner-most computability) を持つ ($SIC(t)$ で記す) とは，以下のように定義される．

- $\tau(t)$ が非関数型ならば $SIN(t)$,
- $\tau(t) = \alpha \to \beta$ ならば，任意の $SIC(u)$ かつ $\tau(u) = \alpha$ となる u に対し $SIC(t[u])$.

［定理 4.2］ R を PFP-STRS とする．$(T^{args}_{SIC}(R) \cap NF_{nfun}(R), T^{args}_{SIC}(R) \cap \mathcal{T}_{nfun}(\Sigma, \mathcal{V}))$ 上の無限 $\langle DP_{SC}(R), \underset{in}{\longrightarrow}_R\rangle$-依存鎖が存在しないならば $SN(R)$ が成立．

［証明］ $\neg SIN(R)$ とする．$\to_R$ を $\underset{in}{\longrightarrow}_R$ に置き換えることにより，定理 3.8 と同様にして $T^{args}_{SIC}(R) \cap \mathcal{T}_{nfun}(\Sigma, \mathcal{V})$ 上の無限 $\langle DP_{SC}(R), \underset{in}{\longrightarrow}_R\rangle$-依存鎖の存在が示せる．ここで，ある $l \to r \in R^{\eta}$ に対し $l\theta \underset{in}{\longrightarrow} r\theta$ のときにはいつでも，$l^{\#}\theta \in NF_{nfun}(R)$ であるので題意が示せる． □

以下では，全ての $(T^{args}_{SIC}(R) \cap NF_{nfun}(R),\ T^{args}_{SIC}(R) \cap \mathcal{T}_{nfun}(\Sigma, \mathcal{V}))$ 上の $\langle DP_{SC}(R), \underset{in}{\longrightarrow}_R\rangle$-再帰成分からなる集合を $RC_{SIC}(R)$ で記す．

［定理 4.3］ R を PFP-STRS とする．全ての $C \in RC_{SIC}(R)$ に対し，ある簡約化対 $(\gtrsim, >)$ が存在して $R \cup C \subseteqq \gtrsim$ かつ $C \cap > \neq \emptyset$ となるならば $SIN(R)$ が成立．さらに，R が重なりを持たないならば $SN(R)$ が成立．

［証明］ $SIN(R)$ は定理 4.2 より明らか．R が重なりを持たないときに $SIN(R) \iff SN(R)$ が成立することは，TRS における Gramlich の成果 [4] と，文献 [9] で与えられた単純化項とカリー項の対応関係を組み合わせる事により容易に示せる． □

定理 4.2 では，取り扱う依存対 $\langle u^{\#}, v^{\#}\rangle$ の左辺 $u^{\#}$ の例 $u^{\#}\theta$ が正規形に限定されている ($u^{\#}\theta \in NF_{nfun}(R)$)．この部分が前節の対応する定理 3.8 との最大の違いになっている．この正規形に限定されているという性質のおかげで，定理 4.3 は文献 [11] で提案された堅固の概念を用いた手法と組み合わせることができる．ここで，項 t が**堅固** (firmness) であるとは，t での変数の出現が全て葉の位置であることである．

［系 4.4］ R を PFP-STRS とする．全ての $C \in RC_{SIC}(R)$ に対し，ある切り落とし法に基づく簡約化対 $(\gtrsim_{\pi}, >_{\pi})$ が存在して，$C \subseteqq \gtrsim$ かつ $C \cap > \neq \emptyset$ かつ，任意の $\langle u, a[v_1, \ldots, v_m]\rangle \in DP_{SC}(R)$ と $i \in \pi(a)$ に対し v_i が堅固な構成子項であるとする．このとき，$SIN(R)$ が成立．さらに，R が重なりを持たないならば $SN(R)$ が成立．

系 3.10 や定理 4.3 と比べて $R \subseteqq \gtrsim$ の条件が無くなっているため，容易に R_8 の停止性を示すことができる．実際，$cons[(x, xs)] > xs$ と $tt > ff$ となる簡約化順序の設計は容易であるので (例えば文献 [7] で提案した経路順序)，$\pi(foldl^{\#}) = [3], \pi(foo^{\#}) = [1]$ とする事により系 4.4 が適用できる．

謝辞 本研究は一部，科研費#15500007,#16300005，人工知能研究振興財団，名古屋大学 21 世紀 COE プログラム (社会情報基盤のための音声・映像の知的統合) の補助を受けている．

文　献

[1] T.Arts, J.Giesl, Termination of Term Rewriting Using Dependency Pairs, Theoretical Computer Science, Vol.236, pp.133–178, 2000.
[2] J.Hughes, Why Functional Programming Matters, Computer Journal 32(2): 98–107, 1989.
[3] J.-Y.Girard, Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur, Ph.D. thesis, University of Paris VII, 1972.
[4] B.Gramlich, On Termination and Confluence Properties of Disjoint and Constructor-Sharing Conditional Rewrite Systems, Theoretical Computer Science 165, 97–131, 1996.
[5] N.Hirokawa, A.Middeldorp, Dependency Pairs Revisited, Proc. of the 15th Int. Conf. on Rewriting Techniques and Applications, LNCS 3091, pp.249–268, 2004.
[6] J.-P.Jouannaud, A.Rubio, The Higher-Order Recursive Path Ordering, In IEEE Symposium on Logic in Computer Science, Trento, Italy, 1999.
[7] Kusakari,K., On Proving Termination of Term Rewriting Systems with Higher-Order Variables, IPSJ Transactions on Programming, Vol.42, No.SIG 7 (PRO 11), pp.35–45, 2001.
[8] Kusakari,K., Higher-Order Path Orders based on Computability, IEICE Transactions on Information and Systems, Vol.E87-D, No.2, pp.352–359, 2004.
[9] Kusakari,K., Chiba,Y., Higher-Order Knuth-Bendix Procedure and its Applications, LA-Symposium 2004 (Summer), pp.9.1–9.12, 2004.
[10] F.Raamsdonk, On Termination of Higher-Order Rewriting, In Proc. of 12th Int. Conf. on Rewriting Techniques and Applications, LNCS 2051 (RTA2001), pp.261–275, 2001.
[11] 櫻井 敬大, 引数減少原理に基づいた単純型項書換え系の停止性証明, 名古屋大学工学部 2004 年度卒業論文, 2005.
[12] W.W.Tait, Intensional Interpretation of Functionals of Finite Type, Journal of Symbolic Logic 32, pp.198–212, 1967.
[13] Terese, Term Rewriting Systems, Cambridge Tracts in Theoretical Computer Science, Vol.55, Cambridge University Press, 2003.