〈一般研究課題〉　生活環境の充実のための高信頼プログラムの
効率的実行を行う計算戦略に関する研究
助 成 研 究 者　愛知県立大学　　粕谷英人

生活環境の充実のための高信頼プログラムの効率的実行を行う計算戦略に関する研究

粕谷　英人
(愛知県立大学)

A study on computing strategy for efficient execution of reliable program

Hideto KASUYA
(Aichi Prefectural University)

Abstract：

It is known that the set of all redexes for a left-linear term rewriting system is recognizable by a tree automaton, which means that we can construct a tree automaton that accepts redexes. This paper extends this result to Nipkow's higher-order rewrite systems, in which every left-hand side is a linear fully-extended pattern. Needed redexes whose reduction yield is a normalizing strategy are not decidable in higher-order rewrite systems as well as in term rewriting. We present a construction of tree automata that recognize the instances of linear higher-order patterns. We give a construction of ground tree tranducers that recognize rewrite relation of a higher-order rewrite system whose rules are linear. From these results, we show that the strong sequential strategy and the NV-sequential strategy of higher order rewrite systems are decidable.

1　はじめに

関数型プログラムでは評価の順番によっては解が得られたり無限に評価し続けたりするため，解が存在するなら必ずそれを求められる正規化戦略の発見が望まれる。これまでに，遅延評価など評価戦略が研究されて来ているものの，まだ不十分である。一方で，項書換え系については正規化戦略の研究が進んでいる[7][9]。頭必須書換え系は項の先頭部分がこれ以上書き換えることができない頭正規形を求める書換え戦略であり，Durand とMiddeldorp はこれが決定可能であることを示し

た[6]。しかしながら，項書換え系では関数をデータとして扱う高階の概念が扱えないため，これらの結果を関数型プログラムに適用できない。そこで，項書換え系での正規化戦略の結果を，高階の概念を導入した高階書換え系に拡張したい。

我々は，Nipkow の高階書換え系[10] において頭必須書換え戦略が正規化戦略であることを示した[8]。Middeldorp らの項書換え系における頭必須性の決定性の結果を高階書換え系に拡張するためには，高階パターンの具体項を受理する木オートマトンが必要である。これまでに，文献[2] で□-木オートマトンを利用してλ項のマッチング問題が決定可能であることは判明している。しかしながら，これに用いられている手法では，パターンとその具体項の候補から□-木オートマトンを構成しているため，残念ながら我々の目的には利用できない。また，□-木オートマトンでは四階までの問題しか扱えない一方で，一階の方法を通常のλ項の表現を用いて拡張することによりパターンの具体項を受理する木オートマトンを構成することは困難である。なぜなら，パターンに出現する束縛変数を区別する必要があるが，α 等価性のために状態数を有限に抑えられないからである。

そこで本論文では，de Bruijn 記法[5] を用い，高階パターンを一階の項で表現するとともに，与えられたパターンの具体項の集合，ならびに，Nipkow の高階項書換え系により書換え可能な項の集合が，それぞれ木オートマトンにより認識可能であることを示す。さらに，高階書換えの近似を用いて正規化戦略の決定可能性の十分条件を与える。

2　高階書換え系

Sを基本型の集合とするとき，関数を表す記号→とSから定められる型の集合$\mathcal{T}_s$は次を満たす最小の集合である。

$$\tau_s \supseteq S$$
$$\tau_s \supseteq \{\alpha \to \alpha' | \alpha, \alpha' \in \tau_s\}$$

型αを持つ型付き変数の集合を$\mathcal{X}_\alpha$，型αを持つ型付き関数記号の集合を$\mathcal{F}_\alpha$で表す。また，型付き変数全体の集合を$\mathcal{X} = \cup_{\alpha \in \mathcal{T}_S} \mathcal{X}_\alpha$，型付き関数記号全体の集合を$\mathcal{F} = \cup_{\alpha \in \mathcal{T}_S} \mathcal{F}_\alpha$とする。単純型λ項は以下の推論規則により定義される。

$$\frac{x \in \mathcal{X}_\alpha}{x : \alpha} \quad \frac{f \in \mathcal{F}_\alpha}{f : \alpha} \quad \frac{s : \alpha \to \alpha' \; t : \alpha}{(st) : \alpha'} \quad \frac{x : \alpha \; s : \alpha'}{(\lambda x.s) : \alpha \to \alpha'}$$

$t : \alpha$が推論されるとき，tを型αを持つ単純型λ項といい，その全ての集合を$\mathcal{T}_\alpha$で表し，$\cup_{\alpha \in \mathcal{T}_S} \mathcal{T}_\alpha$を$\mathcal{T}$で表す。単純型λ項を高階項，λ項もしくは単に項という。通常と同様に自由変数と束縛変数の概念を用い，項t中のすべての束縛変数の集合を$BV(t)$，自由変数の集合を$FV(t)$と書く。また，$FV(t) \cup BV(t)$を$Var(t)$で表す。自由変数を含まない高階項を基礎項と呼ぶ。項tの束縛変数の名前替えによって項sが得られるとき，s とt は同値であると言い，$s \equiv t$ と書く。以下では，基本的にX, Y, Z を自由変数として，x, y, z を束縛変数として用いる。また，$\vec{x}$ を用いて，$x_1 x_2 \cdots x_m$ $(m \geq 0)$を表す。

変数をそれと同一の型の高階項に移す写像$\sigma : \mathcal{X} \mapsto \mathcal{T}$は，その定義域$Dom(\sigma) = \{X \mid X \neq \sigma(X)\}$が有限であるとき代入という。また，$Dom(\sigma) = \{X_1, \cdots, X_n\}$，$\sigma(X_i) = t_i$ とするとき，σを$\{X_1 \mapsto t_1, \cdots, X_n \mapsto t_n\}$とも書く。$W$を変数の集合とするとき，$\sigma$の定義域を$W$に制限して得られる代入を$\sigma|_W$により表し，$W$の補集合に制限して得られる代入を$\sigma|_{\overline{W}}$により表す。すなわち，$\sigma|_W = \{X \mapsto \sigma(X) \mid X \in$

$(W \cap Dom(\sigma))\}$，$\sigma|_{\overline{W}}\sigma = \{X \mapsto \sigma(X) \mid X \in (Dom(\sigma) - W)\}$，である。また，$FV(\sigma) = \cup_{X \in Dom(\sigma)} FV(X\sigma)$と定義する。代入は，以下のように高階項から高階項への写像σ'に自然に拡張される。

1. $X \in \mathcal{X}$ のとき，$\sigma'(X) \equiv \sigma(X)$
2. $f \in \mathcal{F}$ のとき，$\sigma'(f) \equiv f$
3. $t = (t_1 t_2)$ のとき，$\sigma'((t_1 t_2)) \equiv (\sigma'(t_1)\sigma'(t_2))$
4. $t \equiv \lambda x.t_1$ かつ $x \notin FV(\sigma)$ のとき，$\sigma'(\lambda x.t_1) \equiv \lambda x.\sigma'|_{\overline{\{x\}}}(t_1)$

以下ではσ'とσを同一視し，$\sigma'(t)$を$t\sigma$と書く。

β 簡約は，$((\lambda x{:}s)t)$を$s\{x \mapsto t\}$に置き換える操作である。また，sが型$\alpha \to \alpha'$の項，$x \notin Var(s)$ を型の変数とするとき，η拡大はsを$\lambda x.(sx)$ に置き換える操作である。β 簡約できない高階項を β 正規形，η拡大できない高階項をη拡大形という。また，β 簡約，η拡大のどちらもできない高階項を$\beta\eta$拡大正規形といい，これを正規化項と呼ぶ。tの正規化項を$t\downarrow$と書く。どの高階項も唯一の正規化項を持ち，η拡大形の集合はβ 簡約について閉じていることが知られている。どの正規化項tも$\lambda x_1 \cdots x_m.(\cdots(at_1)\cdots t_n)$の形式($m, n \geq 0$, $a \in F \cup V$)で表すことができる。以下では，この正規化項を$\lambda x_1 \cdots x_m.a(t_1, \cdots, t_n)$ と書くことにする。このとき，$top(t) = a$と定義する。なお，$a(t_1, \cdots, t_n)$ は基本型である。

$\lambda x_1 \cdots x_m.a(t_1, \cdots, t_n)$ の形式の表現に基づいて正規化項の位置を定義する。次章以降での取り扱いを簡単にするため，λxの部分には位置を割り振らない。正規化項の位置は自然数の列であり，$t = \lambda\vec{x}.a(t_1, \cdots, t_n)$の位置の集合は$Pos(t) = \{\varepsilon\} \cup \{i \cdot p \mid 1 \leq i \leq n, p \in Pos(t_i)\}$ で定められる。位置p, q, r について，$pq = r$のとき$p \leq r$と書く。特に$q \neq \varepsilon$，すなわち$p \neq r$ のとき$p < r$と書く。tの位置pの部分項$t|_p$を次のように定める。

$$(\lambda\vec{x}.a(t_1, \ldots, t_n))|_p \equiv \begin{cases} a(t_1, \ldots, t_n) & \cdots\ p = \varepsilon \text{ のとき} \\ t_i|_q & \cdots\ p = iq \text{ のとき} \end{cases}$$

$t|_p$が項tの自由変数である位置$p \in Pos(t)$の集合を$Pos_x(t_i)$ で表す。また，tの部分項$t|_p$を同一の基本型の項uで置き換えて得られる項$t[u]_p$ を次のように定める。

$$(\lambda\vec{x}.a(t_1, \ldots, t_n))[u]_p \equiv \begin{cases} \lambda\vec{x}.u & \cdots\ p = \varepsilon \text{ のとき} \\ \lambda\vec{x}.a(\ldots, t_i[u]_q, \ldots) & \cdots\ p = iq \text{ のとき} \end{cases}$$

正規化項s の位置$p \in Pos(s)$で束縛される変数の集合$BV_p(s)$を以下のように定義する。

$$BV_p(\lambda x_1 \cdots x_m.a(t_1, \ldots, t_n)) = \begin{cases} \emptyset & \cdots p = \varepsilon \text{ のとき} \\ \{x_1, \ldots, x_m\} \cup \bigcup_i (BV_q(t_i)) & \cdots p = iq \text{ のとき} \end{cases}$$

tをそのη正規形が変数ではない正規化項とする。tの任意の部分項$F(u_1, \cdots, u_n)(F \in FV(t))$ について$u_1\downarrow_\eta, \cdots, u_n\downarrow_\eta$がそれぞれ異なる束縛変数のとき$t$をパターンという。$\alpha$ を基本型，$l : \alpha$ をパターン，$r : \alpha$ を正規化項で$FV(l) \subseteq FV(r)$を満たすとき，$l \triangleright r : \alpha$ を型α の高階書換え規則と呼ぶ。高階書換え規則の集合を高階書換え系という。

Rを高階書換え系，$l \triangleright r \in R$，θ を代入とするとき，$l\theta\downarrow$をリデックスと呼ぶ。書換え規則$l \triangleright r \in R$，代入σ，位置pについて，$s \equiv s[l\sigma\downarrow]_p$, $t \equiv s[r\sigma\downarrow]_p$ のとき，sはtへ簡約されるといい，$s \to_R t$もしくは単に$s \to t$と書く。また，$p = \varepsilon$ のとき$s \xrightarrow{\varepsilon} t$，$p > \varepsilon$ のとき$s \xrightarrow{>\varepsilon} t$ と書く。なお，書換え規則は基本型であるので，sと項tは正規化項である。

→の反射推移閉包を→*と書く。tから始まる無限系列$t \equiv t_0 \to t_1 \to \cdots$が存在するとき，$t$は無限簡約列を持つという。無限簡約列を持つ項が存在しないとき，→は停止性を持つという。$\to_R$が停止性を持つとき，高階書換え系Rは停止性を持つという。書換え系列に名前Aをつけて$A : t_0 \to t_1 \to \cdots \to t_n$ と書くことができる。

$l \to r$ と$l' \to r'$を書換え規則とする。このとき，代入 σ，σ'，位置$p \in Pos_{\mathcal{X}}(l')$が存在して，$l\sigma\downarrow = l'|_p(\sigma'|_{\overline{BV_p(l')}})\downarrow$を満たす（ただし，書換え規則が同一のときには，$p \neq \varepsilon$とする）とき，これらの書換え規則は重なるという。高階書き換え系Rに重なる規則を持つとき，Rは重なりを持つという。重なりを持たない高階書き換え系は，どの規則も左線形であるとき，直交するという。

3 de Bruijn 記法

λ 項は，等価性により複数の表現が存在するため扱いづらい。そこで，文献[1] における高階書換え系の表現と同様に束縛変数の置き換えによって等しくなる項の表現を唯一に定めるde Bruijn 記法[5] を使用して，これを一階項$\mathcal{T}(\mathcal{F} \cup \mathcal{N} \cup \{\lambda, @\})$ として表し，db 項と呼ぶ。この記法ではλ項の自由変数と束縛変数を数字で表し，数字nはその変数がn個外側のλで束縛されることを，または，その変数に対応するが存在しないときには自由変数を表す。なお，arity(@) = 2, arity(λ) = 1 であり，@ とλ以外のすべての関数記号$\mathcal{F} \cup \mathcal{N}$ のアリティは0 である。以下ではλ項をP, M, Nを使って表し，db 項をp, s, t を使って表す。

定義1（db 項） Mをλ項とする。Mを表す*db*項は$(\phi(M))[[x_1 \mapsto 1]] \cdots [[x_n \mapsto n]]$として与えられる。ここで$x_1$, ..., x_n はMの自由変数である。

$$\begin{array}{lcl} \phi(x) & \equiv & x \\ \phi(MN) & \equiv & @(\phi(M), \phi(N)) \\ \phi(\lambda x.M) & \equiv & \lambda(\phi(M)[\![x \mapsto 1]\!]) \end{array}$$

ここで[[ ]] は*de Bruijn* 記法のための次のような変数の置き換えを表している。

$$\begin{array}{lcl} x[\![y \mapsto n]\!] & \equiv & \begin{cases} n & x = y \text{ のとき} \\ x & \text{それ以外} \end{cases} \\ m[\![y \mapsto n]\!] & \equiv & m \\ (@(s,t))[\![y \mapsto n]\!] & \equiv & @(s[\![y \mapsto n]\!], t[\![y \mapsto n]\!]) \\ (\lambda.(s))[\![y \mapsto n]\!] & \equiv & \lambda(s[\![y \mapsto n+1]\!]) \end{array}$$ □

次の例のようにλ項はdb 項に変換される。

$$\begin{array}{lcl} \phi(\lambda xy.xy) & \equiv & \lambda(\lambda(@(2,1))) \\ (\phi(\lambda x.f(yx)x))[\![y \mapsto 1]\!] & \equiv & \lambda(@(@(f, @(2,1)),1)) \end{array}$$

ここで$\lambda x.f(yx)x$ の自由変数yはdb 項への変換で自由変数1 が割り当てられている。yはλxの内側に出現するため，得られたdb 項$\lambda(@(@(f, @(2, 1)), 1))$では2 として出現している。

定義2（db 項の代入） *db* 項s の自由変数n_1, n_2, ...それぞれに*db* 項t_1, t_2, ...を代入する$s\sigma$を次のように定義する*[1]*。ここで$\sigma = \{\!\{n_1 \mapsto t_1, n_2 \mapsto t_2, \ldots\}\!\}$とする。

$$\begin{array}{lcl} (@(s_1, s_2))\sigma & \equiv & @(s_1\sigma, s_2\sigma) \\ (\lambda(s_1))\sigma & \equiv & \lambda(s_1\tilde{\sigma}) \\ m\sigma & \equiv & \begin{cases} t & m \mapsto t \in \sigma \text{ のとき} \\ m & m \mapsto t \notin \sigma \text{ のとき} \end{cases} \end{array}$$

このとき$\tilde{\sigma} = \{\!\{n_1+1 \mapsto \mathcal{U}_0^1(t_1), n_2+1 \mapsto \mathcal{U}_0^1(t_2), \ldots\}\!\}$ とし, $\mathcal{U}_i^n(\ )$ は以下で定義されるアップデート関数である。

$$\begin{array}{lcl} \mathcal{U}_i^n(@(s_1,s_2)) & \equiv & @(\mathcal{U}_i^n(s_1), \mathcal{U}_i^n(s_2)) \\ \mathcal{U}_i^n(\lambda(s_1)) & \equiv & \lambda(\mathcal{U}_{i+1}^n(s_1)) \\ \mathcal{U}_i^n(m) & \equiv & \begin{cases} m+n & m > i \text{ のとき} \\ m & m \leq i \text{ のとき} \end{cases} \end{array}$$

□

直観的には$\mathcal{U}_0^1(s)$ はsの自由変数に1 を加えて得られるdb 項を表す。

λ適用によるβ変換は$@(\lambda(s),t) \rightarrow_\beta \mathcal{U}_0^{-1}(s\{\!\{1 \mapsto \mathcal{U}_0^1(t)\}\!\})$で表される。例えば，$\lambda$項の$\beta$変換$\lambda z.((\lambda xy.xz)(\lambda x.z)) \rightarrow \lambda z.(\lambda y.((\lambda x.z)z))$に対応したdb 項の$\beta$変換は次のようになる。

$$\begin{array}{lcl} \lambda(@(\lambda(\lambda(@(2,3))),\lambda(2))) & \rightarrow_\beta & \lambda(\mathcal{U}_0^{-1}((\lambda(@(2,3)))\{\!\{1 \mapsto \mathcal{U}_0^1(\lambda(2))\}\!\})) \\ & \equiv & \lambda(\lambda(@(\lambda(3),2))) \end{array}$$

η拡大はsがλを先頭に持たず，かつ基本型でないときにsを$\lambda(@(\mathcal{U}_0^1(s),1)$に置き換える操作である。

β簡約できないdb 項をβ正規形，η拡大できないdb 項をη拡大形という。また，β簡約，η拡大のどちらもできないdb 項を$\beta\eta$拡大正規形といい，これを正規化項と呼ぶ。sの正規化項を$s\downarrow$と書く。どのdb項も唯一の正規化項を持つ。db 項の計算は$\beta\eta$拡大正規形の上で議論を行うことを前提とする。そのため，λ項ではη拡大形はβ変換について閉じていることがわかっているので，β変換のみに注意して議論を行う。

db 項の代入は次の性質を持つ。

補題1 $s \equiv @(\cdots@(l,n_1),\cdots,n_m)$ かつ$l > max\{n_1,\ldots,n_m\}$であるとする。このとき，任意のdb 項$t$に対して$t \equiv s\{\!\{l \mapsto s'\}\!\}\downarrow$かつ$n_1$から$n_m$のいずれも自由変数に持たないdb 項$s'$が存在する。

証明 $t' \equiv t\{\!\{n_1 \mapsto 1,\ldots,n_m \mapsto m\}\!\}$とし, $s' \equiv \lambda(\cdots\lambda(t')\cdots)$ととればよい。 □

4 高階パターンと木オートマトン

4.1 木オートマトン

$\mathcal{V}$を変数の可算無限集合，$\mathcal{F}$を関数記号の有限集合とする。$\forall f \in \mathcal{F}$ はアリティを持ち，$arity(f)$で表される。このとき$\mathcal{V}$の変数と$\mathcal{F}$ の関数記号から構成される一階項(単に項ともいう) の集合を$\mathcal{T}(\mathcal{F},\mathcal{V})$で表す。自然数列$u$で表される項$t$の位置について$t|_u$により位置$u$での部分項を表す。文脈$C$の位置$u$の穴を項$t$で置き換えて得られる項を$C[t]_u$で表す。$u$は省略することがある。二つの項が同一であることを$\equiv$を使って表す。

定義3（木オートマトン） 木オートマトンは$A = \langle Q,\mathcal{F},Q_f,\delta\rangle$で表され，$Q$は状態の有限集合，$\mathcal{F}$はシグニチャで$Q \cap \mathcal{F} = \emptyset$ を満たし，Q_f $(\subseteq Q)$ は最終状態の集合である。δ は遷移規則の集合を表し，遷移規則は次の形をしている。

$f(q_1,\cdots,q_n) \rightarrow q$ ただし$q_1,\cdots,q_n, q \in Q, f \in \mathcal{F}, arity(f) = n$ □

特に左辺が同一な遷移規則が複数存在しないとき，決定的であるという。ここでC[ ] を文脈としたとき，木オートマトンの$\mathcal{T}(Q \cup \mathcal{F})$ 上の遷移関係$\rightarrow_A$ は次のように定義される。

$$f(q_1,\ldots,q_n) \rightarrow q \in \delta \text{ のとき, } C[f(q_1,\ldots,q_n)] \rightarrow_A C[q]$$

$\rightarrow_A^*$ は$\rightarrow_A$ の反射的推移的閉包を表す。

項tが木オートマトンAに受理される，つまり$t \in L(A)$ となるのは遷移関係$\rightarrow_A$ によって表される

次のときである。

$$\forall t \in \mathcal{T}(\mathcal{F}),\ t \in L(A) \qquad iff \qquad t \rightarrow_A^* q \in Q_f$$

4.2 高階パターンの受理

高階パターンの具体項を受理する木オートマトンを構築する方法を具体的に与える。sをそのη正規形が変数ではない正規化項とする。自由変数n を持つsの任意の基本型の部分項@(…@(@(n, t_1), t_2), …, t_m)について$t_1\downarrow_\eta$, …, $t_m\downarrow_\eta$がそれぞれ異なる束縛変数のときsをパターンという。また，自由変数nの引数にnが出現する位置でのスコープにおけるすべての束縛変数を持つとき，そのパターンはfully-extendedであるという。db 項で表されるfully-extended パターンpの全ての出現を$Occ(p)$と表す。ただし，自由変数が左端に現れる場合には，その項がfully-extended である出現のみOcc(p)に含まれる。

$$\begin{array}{lcll} Occ(a) & = & \{\varepsilon\} & a \in \mathcal{F} \cup \mathcal{N} \text{ のとき} \\ Occ(@(s,t)) & = & \begin{cases} \{\varepsilon\} & hd(@(s,t)) \text{ が自由変数のとき} \\ \{\varepsilon\} \cup \{1u \mid u \in Occ(s)\} \cup \{2u \mid u \in Occ(t)\} & \text{それ以外} \end{cases} & \\ Occ(\lambda(s)) & = & \{\varepsilon\} \cup \{1u \mid u \in Occ(s)\} & \end{array}$$

ここで，$hd(p)$は以下のように定義され，db 項pにおいて，@以外で最も左にある関数記号を表す。

$$\begin{array}{lcll} hd(a) & = & a & a \in \mathcal{F} \cup \mathcal{N} \text{ のとき} \\ hd(@(s,t)) & = & hd(s) & \\ hd(\lambda(s)) & = & \lambda & \end{array}$$

高階パターンpの具体項$p\theta\downarrow$を受理するために，木オートマトン$A(p) = \langle Q,\mathcal{F},Q_f,\delta\rangle$を構築する。ただし，$A(p)$ によって受理される具体項は，変数部分を表す自然数が，sと0によってコーディングされているものとする。pをdb 項，Qを状態の集合Q=$\{q_\perp, q_\varepsilon, q_1,...\}$，$\mathcal{F}$をシグニチャ，最終状態の集合$Q_f = \{q_\varepsilon\}$ とし，kをdb 項pに現れるどの束縛変数より大きな自然数に定める。このときパターンpの出現$u \in Occ(p)$に対して遷移規則の集合 δ を以下で生成される遷移規則からなる集合とする。

(1) 任意の $u \in Occ(p)$ について，

(1-1) $p|_u \equiv @(@(\cdots @(n,t_1),\ldots,t_{m-1}),t_m)$ $(0 \le m)$ で，nが p の自由変数のとき[1]，$q_\perp \rightarrow q_u$

(1-2) それ以外のとき，

(1-2-1) $p|_u \equiv a \in \mathcal{F}$ のとき，$a \rightarrow q_u$

(1-2-2) $p|_u \equiv n \in \mathcal{N}$ のとき，$q_{vn} \rightarrow q_u$

(1-2-3) $p|_u \equiv @(s,t)$ のとき，$@(q_{u1},q_{u2}) \rightarrow q_u$

(1-2-4) $p|_u \equiv \lambda(s)$ のとき，$\lambda(q_{u1}) \rightarrow q_u$

(2) 任意の $a \in \mathcal{F}$ について，$a \rightarrow q_\perp$

(3) 任意の $n \in \{1,\ldots,k\}$ について，$q_{vn} \rightarrow q_\perp$

(4) 任意の $n \in \{1,\ldots,k\}$ について，$s(q_{vn-1}) \rightarrow q_{vn}$

(5) $s(q_\perp) \rightarrow q_\perp$

(6) $\lambda(q_\perp) \rightarrow q_\perp$

(7) $@(q_\perp, q_\perp) \to q_\perp$

(8) $0 \to q_{v0}$

ここで，s, 0 は$\mathcal{F}$に現れない関数記号とし，アリティはそれぞれ1と0である。

補題2 db 項p を線形な*fully-extended* パターンとする。このとき，pから生成された木オートマトン$A(p)$について，$L(A(p)) = \{p\,\theta\downarrow \mid \theta$は代入$\}$ が成り立つ。すなわち，高階パターンの具体項は木オートマトンで受理可能である。

証明　まず，$L(A) \supseteq \{p\,\theta\downarrow \mid \theta$は代入$\}$ となることを示す。具体的にはpの出現$u \in Occ(p)$として，$p|_u$の構造に関する帰納法で$(p|_u)\theta\downarrow \to_A^* q_u$ となることを証明する。最初に$\forall u \in Occ(p)$に対して，$(p|_u)\theta\downarrow \to_A^* q_\perp$となることは定義より明らかである。

- $p|_u \equiv @(@(\cdots @(n, t_1), \ldots, t_{m-1}), t_m)$ かつ n が自由変数の場合，$q_\perp \to q_u \in \delta$ が存在する。よって $(p|_u)\theta\downarrow \to_A^* q_\perp \to_A q_u$。
- それ以外の場合，
  - $p|_u \equiv a \in \mathcal{F} \cup \mathcal{N}$ の場合，遷移規則 $a \to q_u \in \delta$ が存在する。このとき，$(p|_u)\theta\downarrow \equiv a \to_A q_u$。
  - $p|_u \equiv \lambda(p_{u1})$ の場合，$\lambda(q_{u1}) \to q_u \in \delta$ が存在する。帰納法の仮定 $(p|_{u1})\tilde{\theta}\downarrow \to_A^* q_{u1}$ より，$(p|_u)\theta\downarrow \equiv \lambda((p|_{u1})\tilde{\theta}\downarrow) \to_A^* \lambda(q_{u1}) \to_A q_u$。
  - $p|_u \equiv @(s, t)$ の場合，$@(q_{u1}, q_{u2}) \to q_u \in \delta$ が存在する。$(p|_u)\theta\downarrow \equiv @(s\theta\downarrow, t\theta\downarrow)$ と表すことができる。ここで帰納法の仮定 $(p|_{u1})\theta\downarrow \equiv s\theta\downarrow \to_A^* q_{u1}$ および $(p|_{u2})\theta\downarrow \equiv t\theta\downarrow \to_A^* q_{u2}$ より，$(p|_u)\theta\downarrow \equiv @(s\theta\downarrow, t\theta\downarrow) \to_A^* @(q_{u1}, q_{u2}) \to_A q_u$。

次に $L(A) \subseteq \{p\theta\downarrow \mid \theta$ は代入 $\}$ であることを示すために，任意の db 項 t について $t \to_A^n q_u$ ならば $\exists\theta, t \equiv (p|_u)\theta\downarrow$ かつ θ は u における束縛変数に代入しないことを n に関する帰納法により証明する。

- $t \to_A^{n-1} q_\perp \to_A q_u$ の場合，$p|_u \equiv @(@(\cdots @(l, t_1), \ldots, t_{m-1}), t_m)$ かつ n が自由変数。このとき，p が fully-extended なので，各 t_i は変数であり，$l > max\{t_1, \ldots, t_m\}$。よって補題 1 より成立する。
- $t \to_A^{n-1} @(q_{u1}, q_{u2}) \to q_u$ の場合，$p|_u \equiv @(p|_{u1}, p|_{u2})$ かつ $t = @(t_1, t_2)$ である。帰納法の仮定より，$\exists\theta_1, \theta_2, t_1 \equiv (p|_{u1})\theta_1\downarrow$ かつ $t_2 \equiv (p|_{u2})\theta_2\downarrow$。$p$ が線形なので，$\theta = \theta_1 \cup \theta_2$ とすると，$t \equiv @(t_1, t_2) \equiv @(p|_{u1}, p|_{u2})\theta\downarrow \equiv (p|_u)\theta\downarrow$。
- $t \equiv a \to_A q_u$ の場合，$p|_u \equiv a \in F \cup N$ であり，$\theta = \{\!\!\{\ \}\!\!\}$ ととればよい。
- $t \to_A^{n-1} \lambda(q_{u1}) \to q_u$ の場合，$p|_u \equiv \lambda(p|_{u1})$ かつ $t \equiv \lambda(s)$ である。帰納法の仮定より，$\exists\theta, s \equiv (p|_{u1})\theta\downarrow$ かつ θ は 1 に代入しない。また，$\exists\theta', \tilde{\theta'} = \theta$ である。よって $t \equiv \lambda(s) \equiv \lambda((p|_{u1})\theta)\downarrow \equiv (p|_u)\theta'\downarrow$。 □

補題3 *db* 項pを線形な*fully-extended* パターンとする。このとき，pの具体項を部分項に持つ*db* 項の集合を受理する木オートマトンが存在する。

証明 $A(p)$ に次の規則を加えた木オートマトン$A(p)'$ は$C[p\theta\downarrow]$ を受理する。これは，$A(p)'$ の生成規則および補題2 よりあきらか。

$$\begin{aligned} \lambda(q_\varepsilon) &\to q_\varepsilon \\ @(q_\varepsilon, q_\perp) &\to q_\varepsilon \\ @(q_\perp, q_\varepsilon) &\to q_\varepsilon \end{aligned}$$

□

定理1 R を高階書換え系とする。Rで書換え可能な項の集合を受理する木オートマトンを作ることができる。

証明補題3 および木オートマトンが和集合について閉じていることより成り立つ。 □

[1]このとき，p がパターンなので $t_1, \ldots, t_m$ は自然数である．また特に $m = 0$ のときは $p|_u = n$ である。

5　書換えの近似と決定問題

高階書換え系の書換え関係$\rightarrow_R^*$は項書換え系と同様，一般的には決定不能なので，その書換えが必須書換えであるかは決定不可能である。そこで，文献[6] と同様な手法により補題2 と定理2 から，近似による書換えを用いた決定可能性の十分条件を得る。

5.1 GTT

GTT(Ground Tree Transducer)[4] は木の上の関係を表す一方法である。書換え計算における書換え関係をGTT で定義できればGTT の性質(以下の定理2) から，書換え関係の反射推移閉包を認識することができる。

定義4 (GTT) *GTT*は，同じアルファベット上の2 つの木オートマトンの組で表される。それぞれの状態の集合は同じ要素を含む場合がある。また，(t, t')という組が$GTT(A_1, A_2)$によって認識されるのは，文脈C が存在して$t = C[t_1, ..., t_n]$, $t' = C[t_1', ..., t_n']$ となり，状態$q_1, ..., q_n$ が両方のオートマトンに存在してすべてのiで$t_i \rightarrow_{A_1}^* q_i$ かつt_i' iで$t_i \rightarrow_{A_2}^* q_i$ となるときである。$GTT(A_1,A_2)$に受理される言語を$L(A_1,A_2)$と書く。 □

補題4 両辺に共通変数を持たない両辺がパターンである高階書換え系に対して，書換え規則の両辺の具体項をそれぞれ受理する2 つの木オートマトンを与えると，それによって構成される*GTT*はその規則による書換え関係を認識する。

証明 具体的に*GTT* を構成する方法を示す。高階書換え系$R = \{l_1 \triangleright r_1, ..., l_n \triangleright r_n\}$ とする。またl_i の具体項を受理する木オートマトンをA_i，r_i の具体項を受理する木オートマトンをB_i とし，各A_i 間と各B_i間に共通状態がないものとする。このとき求めるGTT は，$GTT(\cup_i A_i; \cup_i B_i)$ である。 □

また，次の定理[3] によりGTT は書換え関係の反射推移閉包を認識することができる。

定理2 関係$R \subseteq \mathcal{T}(\mathcal{F})^2$が認識可能であるとき，その逆$R^{-1}$，推移閉包$R^*$も認識可能である。また，$L$が$\mathcal{T}(\mathcal{F})$の認識可能な部分集合であるとき，$R[L] = \{s \mid sRt\ \exists t \in L\}$ も認識可能である。 □

この定理により，近似による書換えの必須書換えを決定することができ，高階書換え系の決定可能問題の十分条件を与えることができた。

6　おわりに

本論文では高階書換え系における頭必須書換えの決定性について議論することを目的として，高階パターンをde Bruijn 項で表し，その具体項を受理する木オートマトンを構築する方法を示し，構築した木オートマトンにより高階書換え系のリデックスを受理することを示した。さらにGTT が高階書換え系の書換え関係を認識することを示し，GTT を利用して高階書換え系における必須書換えの決定可能性を議論して，近似を用いて高階書換え系のリデックスが必須リデックスであるかどうか決定可能となる十分条件を示した。木オートマトンの実行の観点から頭必須書換えの効率的実行の実現を行い，さらに本手法の適用可能な範囲を明確にすることが今後の課題である。

参考文献

[1] Eduardo Bonelli, Delia Kesner, and Alejandro Rios. A de Bruijn notation for higher-order rewriting. In *Proceedings of RTA 2000*, volume 1833 of *LNCS*, pages 62-79, 2000.

[2] Hubert Comon and Yan Jurski. Higher-order matching and tree automata. In *Proceedings of CSL'97*, volume 1414 of *LNCS*, pages 157-176, 1997.

[3] Jean-Luc Conquidé and Rémi Gilleron. Proofs and reachability problem for ground rewrite systems. In *Proccedings of 6th International Meeting of Young Computer Scientists*, volume 464 of *LNCS*, pages 120-129, 1990.

[4] Max Dauchet and Sophie Tison. Decidability of confluence for ground term rewriting systems. In *Proccedings of 1st Fundamentals of Computation Theory*, volume 199 of *LNCS*, pages 80-89, 1985.

[5] N. G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation. *Indagations Mathematicae*, 34:381-392, 1972.

[6] Irène Durand and Aart Middeldorp. Decidable call by need computations in term rewriting (extended abstract). In *Proceedings of the 14th International Conference on Automated Deduction*, volume 1249 of *LNAI*, pages 4-18, 1997.

[7] Gérard Huet and Jean-Jacques Lévy. *Computations in Orthogonal Rewriting Systems, I and II*, pages 396-443. The MIT Press, 1991.

[8] Hideto Kasuya, Masahiko Sakai, and Kiyoshi Agusa. Descendants and head normalizaition of higher-order rewrite systems. In *Proceedings of FLOPS 2002*, volume 2441 of *LNCS*, pages 198-211, 2002.

[9] Aart Middeldorp. Call by need computations to root-stable form. In P*roceedings of the 24th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 94-105, 1997.

[10] Tobias Nipkow. Higher-order crtical pairs. In *Proceedings of 6th IEEE Symposium, Logic in Computer Science*, pages 342-349. IEEE Press, 1991.