項到達可能性の判定における成長TRSに対する手法と正規化規則による手法の関係

村田 龍彦 (Tatsuhiko Murata)[†]
酒井 正彦 (Masahiko Sakai)[††]　西田 直樹 (Naoki Nishida)[††]
草刈 圭一朗 (Keiichirou Kusakari)[††]　坂部 俊樹 (Toshiki Sakabe)[††]

名古屋大学大学院情報科学研究科
(Graduate School of Information Science, Nagoya University)
[†]tatsuhiko@sakabe.i.is.nagoya-u.ac.jp
[††]{sakai,nishida,kusakari,sakabe}@is.nagoya-u.ac.jp

あらまし

項到達可能性問題とは，与えられた 2 つの項と項書換え系 (TRS) に対し，片方の項（初期項）からもう片方の項（標的項）へ到達できるかという問題である．この問題は一般には決定不能だが，決定可能な TRS のクラスも存在する．F.Jacquemard は，TRS が右線形逆成長の場合において，初期項から到達可能な全ての項を受理する木オートマトン（TA）を構成し，標的項が TA に受理されるかで到達可能性を判定する方法を提案した．さらに，TRS が右線形の場合においては初期項から到達可能な項を全て受理する TA を構築して到達不可能性を判定する手法を示した．本稿では，Jacquemard の手法を元にして，任意の TRS において初期項から到達可能な項を全て受理する TA を，TA 生成ツール Timbuk が必ず出力するような，Timbuk の入力生成アルゴリズムを提案する．また，提案手法と Jacquemard の手法を比較し，TRS が右線形の場合においては得られる TA の能力が等価であることを示す．さらに，提案したアルゴリズムを改良し近似精度の向上を図る． **キーワード**　木オートマトン, 近似, 項到達可能性，Timbuk

1　はじめに

項到達可能性問題とは，与えられた 2 つの項と項書換え系 (TRS) に対し，片方の項（初期項）を TRS の規則によって書き換えて，もう片方の項（標的項）に書き換わるかという問題である．到達可能性は，初期項から到達可能な全ての項を受理する木オートマトン（TA）を生成し，標的項が受理されるかで判定を行う [2] [4]．一般に，到達可能問題は決定不能のため，初期項から到達可能な項全てのみからなる集合を含む近似集合を受理する TA(近似 TA) を生成することにより，標的項が近似 TA で受理されないならば，到達しないことを保証する手法がいくつか提案されている [2] [5]．F.Jacquemard は，右線形逆成長 TRS に対しては初期項から到達可能な全ての項のみを受理する TA が生成できることを示し，右線形 TRS に対しては，右線形逆成長 TRS に近似して近似 TA を生成する手法を提案した [2]．T.Genet と V. Viet Triem Tong は，TRS と初期項のみを受理する TA と状態割当の指針を示す正規化規則の集合から近似 TA を自動生成するツール Timbuk を実装した [5]．Timbuk は近似 TA を出力するためには，適切な入力をユーザーが与える必要がある．そこで本稿では Timbuk の入力を自動生成するアルゴリズムを提案する．まず，Timbuk を用いた近似 TA 生成手法と Jacquemard の近似 TA 生成手法の比較のため Jacquemard の手法を Timbuk で行う方法を示した．そして，提案した Timbuk の入力生成アルゴリズムを改良し近似精度の向上を図る．

2　準備

本稿では，項書換え系と木オートマトンの一般的な記法に従う [3].

関数記号の集合 $\mathcal{F}$ と変数の可算無限集合 $\mathcal{X}$ から構成されるすべての**項**の集合を $T(\mathcal{F},\mathcal{X})$ とする．項 s 中のどの変数も 1 回しか現れないとき，s は**線形**であるという．

項 s が項 t の一部であるとき，s を t の**部分項**と呼び，$s \trianglelefteq t$ と記述する．特に，$s \trianglelefteq t$ かつ $s \neq t$ のとき，s を t の**真部分項**と呼び，$s \triangleleft t$ と書く．項 t の位置 p の部分項を $t|_p$ と書く．

項 t の**深さ** $|t|$ とは次の様に再起的に定義できる．(1) $t \in \mathcal{X}$ ならば，$|t| = 0$，(2) $t \equiv f(t_1,...,t_n)$ ならば，$|t| = 1 + max(|t_1|,...,|t_n|)$.

代入 σ は $\sigma(x) \neq x$ なる x が有限個であるような変数集合から，項の集合への写像である．σ の定義域と値域はそれぞれ $Dom(\sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$，$Ran(\sigma) = \{\sigma(x) \mid x \in Dom(x)\}$ と定義する．$Dom(\sigma) = x_1,...,x_n$，$\sigma(x_i) \equiv t_i$ のとき，σ を $\{x_1 \mapsto t_1,...,x_n \mapsto t_n\}$ と表す．代入は項上への写像に自然に拡張され，$\sigma(t)$ を $t\sigma$ と略記する．項 t に対して，$t\sigma$ を t の**具体項**と呼ぶ．

$\mathcal{F}$ 上の書換え規則 $l \to r$ はそれぞれ左辺と右辺と呼ばれる $\mathcal{F}$ 上の項 l と r の 2 項組 (l,r) である．書換え規則の集合を**項書換え系（**TRS**）**と呼び，$\mathcal{R}$ で表す．$\mathcal{R}$ のすべての規則の左辺（右辺）が線形であるとき，$\mathcal{R}$ は**左（右）線形**であるという．R から定まる項上の 2 項関係 $\to_{\mathcal{R}}$ は，$\to_{\mathcal{R}} = \{s,t \mid s|_p = l\sigma, t|_p = r\sigma, l \to r \in \mathcal{R}\}$ と定義され，**書換え関係**と呼ぶ．書換え関係 $\underset{R}{\to}$ の**反射推移閉包**を $\underset{\mathcal{R}}{\overset{*}{\to}}$ で表す．また，s を項，L を項の集合とすると，TRS にて書換えられた先の全ての項の集合 $\{u \mid s \underset{\mathcal{R}}{\overset{*}{\to}} u\}$，$\{u \mid t \underset{\mathcal{R}}{\overset{*}{\to}} u, t \in L\}$ を，それぞれ s の**到達可能集合**，L の**到達可能集合**と呼び，$\underset{\mathcal{R}}{\overset{*}{\to}}\{s\}$，$\underset{\mathcal{R}}{\overset{*}{\to}} L$ で表す．

項 s,t と TRS $\mathcal{R}$ 上の**項到達可能性問題**とは，与えられた項 s,t と TRS $\mathcal{R}$ に対して，s から t に書換え系列が存在するかという問題である．$s \underset{\mathcal{R}}{\overset{*}{\to}} t$ となる場合，s **は** $\mathcal{R}$ **によって** t **に項到達可能**という．

木オートマトン (TA) は関数記号の有限集合 $\mathcal{F}$，状態の有限集合 $\mathcal{Q}$，最終状態の集合 $\mathcal{Q}_f(\subseteq \mathcal{Q})$，遷移規則の集合 Δ からなる．**遷移規則**は，$f(q_1,...,q_m)$（$f \in \mathcal{F}, q_1,...q_n \in \mathcal{Q}$）の形式の項 (**状態項**) から状態への書換え規則 $f(q_1,...,q_n) \to q$ または，状態から状態への書換え規則 $q \to q'$ である．Δ によって書換えられる関係は $\underset{\Delta}{\to}$ で表し，その反射推移閉包を $\underset{\Delta}{\overset{*}{\to}}$ で表す．遷移規則の左辺が同じ規則を 2 つ以上持たない TA は**決定的**である．項 s が TA $\mathcal{A} = (\mathcal{F},\mathcal{Q},\mathcal{Q}_f,\Delta)$ によって $s \underset{\Delta}{\overset{*}{\to}} q_f(q_f \in \mathcal{Q})$ と書き換わるとき，s **は** $\mathcal{A}$ **で受理される**，もしくは $\mathcal{A}$ **は** s **を受理する**という．$\mathcal{A}$ が受理する項の集合を $L(\mathcal{A}) = \{s \in T(\mathcal{F}) \mid s \underset{\Delta}{\overset{*}{\to}} q_f, q_f \in \mathcal{Q}\}$ と表す．$L = L(\mathcal{A})$ の場合，$\mathcal{A}$ **は** L **を認識する**という．また，項の集合 $T \subseteq T(\mathcal{F})$ に対して，$T \subset L(\mathcal{A})$ のとき，$\mathcal{A}$ を T の**近似集合受理** TA(**単に近似** TA) と呼ぶ．2 つの TA $\mathcal{A} = (\mathcal{F},\mathcal{Q},\mathcal{Q}_f,\Delta)$，$\mathcal{A}' = (\mathcal{F},\mathcal{Q}',\mathcal{Q}_f,\Delta')$ に対し，$\mathcal{Q} \subseteq \mathcal{Q}'$ かつ $\Delta \subseteq \Delta'$ ならば，$\mathcal{A} \subseteq \mathcal{A}'$ と書く．

Q-**代入** $\sigma_{\mathcal{Q}}$ は代入の値域を $Ran(\sigma) = \{\sigma(x) \in \mathcal{Q} \mid x \in Dom(x)\}$ と変更したものである．項 $s \in T(\mathcal{F} \cup \mathcal{Q}, \mathcal{X})$，項 $t \in T(\mathcal{F} \cup \mathcal{Q})$ に対して，$t = s\sigma_{\mathcal{Q}}$ を満たす Q-代入 $\sigma_{\mathcal{Q}}$ が存在するならば $t \preceq s$ と表す．

TA $\mathcal{A}$ と TRS $\mathcal{R}$ が，任意の $t \in T(\mathcal{F})$ に対し，$t = l\sigma \underset{\Delta}{\overset{*}{\to}} q$ となる代入 σ，書換え規則 $l \to r \in \mathcal{R}$ が存在した場合，$t \underset{\Delta}{\overset{*}{\to}} l\sigma_{\mathcal{Q}} \underset{\Delta}{\overset{*}{\to}} q$ となる Q-代入 $\sigma_{\mathcal{Q}}$ も存在するならば，$\mathcal{A}$ と TRS $\mathcal{R}$ の組は**左線形性**を満たすという [1].

3 Jacquemard の近似木オートマトン生成手法

項 s,t と TRS $\mathcal{R}$ 上の到達可能性問題の判定は，s の到達可能集合を認識する TA によって行う．t が $\mathcal{A}$ に受理されれば到達可能性あり，そうでなければ，到達可能性なしと判定する．

一般に到達可能性問題は決定不能なため，s の到達可能性集合の近似 TA $\mathcal{A}'$ を生成して t が $\mathcal{A}'$ に受理されなければ到達可能性なしと判定する．

本節ではそれら TA，近似 TA の生成手法のうち Jacquemard の提案した手法について説明する．

Jacquemard は右線形逆成長 TRS に対して到達可能集合を認識する TA を生成するアルゴリズムを提案した [2].

定義 3.1 (逆成長項書換え系（逆成長 TRS）) TRS $\mathcal{R}$ のどの書換え規則 $l \to r \in \mathcal{R}$ についても，両辺に同時に現れる変数が右辺には深さ 1 以下にしか出現しないとき，$\mathcal{R}$ を**逆成長項書換え系（逆成長** TRS**）**と呼ぶ． □

Jacquemard の提案した右線形逆成長 TRS に対する TA 生成アルゴリズムを以下に記す.

入力 右線形逆成長 TRS $\mathcal{R}$,
TA $\mathcal{A} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta)$

出力 $\xrightarrow[\mathcal{R}]{*} L(\mathcal{A})$ を認識する TA $\mathcal{A}^{IG}$

手順 下記の処理を順番に行う.

1. $k = 0$.
   全ての書換え規則 $l \to r \in \mathcal{R}$ に対し，右辺の任意の具体項が状態項へ遷移するよう，TA $\mathcal{A}^{RS}$=($\mathcal{F}$, $\mathcal{Q}^{RS}$, $\mathcal{Q}_f^{RS}$, Δ^{RS}) を生成する.
   TA $\mathcal{A}_0^{IG} = (\mathcal{F}, \mathcal{Q}_0, \mathcal{Q}_0^f, \Delta_0) =$
   $(\mathcal{F}, \mathcal{Q} \uplus \mathcal{Q}^{RS}, \mathcal{Q}_f, \Delta \uplus \Delta^{RS})$.
2. $l\sigma_{\mathcal{Q}} \xrightarrow[\Delta_k]{*} q$ かつ $r\sigma_{\mathcal{Q}} \not\xrightarrow[\Delta_k]{*} q$ を満たす，書換え規則 $l \to r \in \mathcal{R}$，状態 $q \in \mathcal{Q}$，Q-代入 $\sigma_{\mathcal{Q}}$ が存在するならば，$r\sigma_{\mathcal{Q}}$ の形により場合分けして遷移規則を Δ_{k+1} に追加する.
   (a) $r\sigma_{\mathcal{Q}} = q$ の場合
   $\Delta_{k+1} = \Delta_k \cup$
   $\bigcup_{h(q_1'',...,q_m'') \to q} (h(q_1'', ..., q_m'') \to q)$
   (b) $r\sigma_{\mathcal{Q}} \notin \mathcal{Q}$ の場合
   $r = g(r_1, ..., r_m)$ として,
   $\Delta_{k+1} = \Delta_k \cup \{g(q_1', ..., q_m') \to q\}$
   ただし $q_j' = \begin{cases} q_{r_j}^f & (r_j \notin X) \\ r_j\sigma_{\mathcal{Q}} & (r_j \in X) \end{cases}$ とする.
3. $\mathcal{A}_{k-1}^{IG} \neq \mathcal{A}_k^{IG}$ ならば，k に 1 を加えて，5. へ. そうでなければ，$\mathcal{A}^{IG} = \mathcal{A}_k^{IG}$ として出力する.

定理 3.1 (Jacquemard 法の TA の受理集合 [2]) $\mathcal{R}$ を右線形逆成長 TRS，$s \in T(\mathcal{F})$ を基底項とする．このとき，s の到達可能集合を認識する TA が存在する.

略証 $\{s\}$ を認識する TA は容易に作成できるので，これを Jacquemard のアルゴリズムに入力することにより条件を満たす TA が生成できることが，k に関する帰納法で示せる. □

さらに Jacquemard は，右線形 TRS に対して到達可能集合の近似 TA が生成できることを示した.

定理 3.2 (Jacquemard 法の受理集合 [2]) $\mathcal{R}$ を右線形 TRS，$s \in T(\mathcal{F})$ を基底項とする．このとき，s の到達可能集合の近似 TA が必ず生成できる.

略証 $\mathcal{R}$ の全ての書き換え規則に対して，右辺の深さ 2 以上に出現する変数をそれぞれ新しい変数に置き換える．この置き換えによって得られる TRS $\mathcal{R}'$ は右線形逆成長である．従って定理 3.1 より，$\mathcal{R}'$ と $\{s\}$ を認識する TA を Jacquemard のアルゴリズムに入力すれば，集合 $\xrightarrow[\mathcal{R}']{*} \{s\}$ を認識する TA $\mathcal{A}$ が得られる．$\xrightarrow[\mathcal{R}]{*} \{s\} \subseteq \xrightarrow[\mathcal{R}']{*} \{s\}$ は明らかなので，$\mathcal{A}$ は s の到達可能集合の近似 TA である. □

4 Timbuk

ここでは，近似 TA 生成ツール Timbuk について説明する．Timbuk は，左線形性を満たす TRS $\mathcal{R}$ と TA $\mathcal{A}$，正規化規則の集合を入力とし，集合 $\xrightarrow[\mathcal{R}]{*} L(\mathcal{A})$ の近似 TA を出力する.

正規化規則は，状態の割り当て方を示すものであり，次の条件を満たす $((s,z),\alpha)$ の形式をしている.

- $s \in T(\mathcal{F} \cup \mathcal{Q}, \mathcal{X}) \backslash \mathcal{Q} \cup \mathcal{X}$
- $z \in \mathcal{Q} \cup \mathcal{X}$
- $\alpha = \{(l_1, r_1), ..., (l_n, r_n)\}$
  - $l_1, ..., l_n \in f(w_1, ..., w_m)$
    $(f \in \mathcal{F}, w_1, ..., w_m \in \mathcal{Q} \cup \mathcal{X})$
  - $r_1, ..., r_n \in \mathcal{Q} \cup \{z\}$

$((s,z),\alpha)$ を正規化規則，$t \in T(\mathcal{F} \cup \mathcal{Q})$ を $t \preceq s$ を満たす項，q を状態とする．このとき，t の部分項 t' に対する状態の割当は，部分関数 α_q で次のように再帰的に定義する.

$$\alpha_q(t') = \begin{cases} t' & (t' \in \mathcal{Q}) \\ q & (t' = f(t_1, ..., t_n) \text{ かつ } f(\alpha_q(t_1), ..., \alpha_q(t_n)) \preceq l \text{ を満たす } (l,z) \in \alpha \text{が存在}) \\ r & (t' = f(t_1, ..., t_n) \text{ かつ } f(\alpha_q(t_1), ..., \alpha_q(t_n)) \preceq l \text{ を満たす } (l,r) \in \alpha (r \neq z) \text{ が存在}) \\ \text{(未定義)} & (\text{上記以外}) \end{cases}$$

例 4.1 項 $f(g(f(a,q')),b)$ から状態 q へ，正規化規則 $((f(x,\ b),\ y),\ \{(f(x,\ b),\ y),\ (g(x),\ q_g),\ (f(q_a,q'),y),\ (a,\ q_a),\ (b,\ q_b)\})$ を用いて状態割り当てを行う．この場合，各部分項への状態割当は，それぞれ $\alpha_q(a)=q_a$,$\alpha_q(b)=q_b$,$\alpha_q(f(q_a,q'))=q$, $\alpha_q(g(q))=q_g$,$\alpha_q(f(q_g,q_b)))=q$, となる． □

正規化規則を ξ，正規化規則の集合を Ξ と表す．

Timbuk では，α_q が一意に決まる正規化規則のみを扱う．

定義 4.1 (完全な正規化規則) 正規化規則 $((s,z),\alpha)$ が**完全**であるとは，次を満たすことである．

任意の Q-代入 $\sigma_{\mathcal{Q}}$，状態 q，ならびに項 s の全ての部分項 s' に対して，$\alpha_q(s'\sigma_{\mathcal{Q}})$ が状態を返す． □

定義 4.2 (TRS に対して完全な正規化規則の集合) 正規化規則の集合 Ξ が TRS $\mathcal{R}$ **に対して完全**であるとは，次を満たすことである．

$\mathcal{R}$ の全ての書換え規則 $l\to r\in\mathcal{R}$ に対し，完全な正規化規則 $((r,z),\alpha)\in\Xi$ が存在する． □

Timbuk は正規化規則に基づいて**正規化**と呼ばれる処理を行う．正規化とは，与えられた項 $t\in T(\mathcal{F}\cup\mathcal{Q})$ と状態 q に対し，ある Q-代入 $\sigma_{\mathcal{Q}}$ に対して，$t=s\sigma_{\mathcal{Q}}$ が成り立つような正規化規則 $\xi=((s,z),\alpha)$ を用いて，複数ステップで t から q へ遷移するのに必要な，遷移規則の集合 $Norm_\xi(t\to q)$ を求めることである．$Norm_\xi(t\to q)$ は，部分関数 $Norm_\xi$ によって次のように再帰的に定義される．

$$Norm_\xi(t\to q)=\begin{cases}\emptyset & (t=q)\\ \{t\to q\} & (t\neq q \text{ かつ } t\in\mathcal{Q})\\ \{f(\alpha_q(t_1),...,\alpha_q(t_n)) & (t=f(t_1,...,t_n)\\ \ \to\alpha_q(t)\}\cup & \text{かつ全ての } t'\trianglelefteq t\\ \ \bigcup_{i=1}^n Norm_\xi(t_i\to\alpha_q(t_i)) & \text{に対し}\alpha_q(t')\text{が}\\ & \text{存在})\end{cases}$$

例 4.2 遷移規則 $f(q_1,g(q_2))\to q_3$ に対し正規化規則 $\xi=((f(x,g(y)),z),\{(f(x,q_4),z),(g(y),q_4)\})$ を用いて正規化を行う．この場合，遷移規則の集合 $Norm_\xi(f(q_1,g(q_2))\to q_3)=\{f(q_1,q_4)\to q_3, g(q_2)\to q_4\}$ が得られる． □

Timbuk の動作について説明する．Timbuk は，入力 $\mathcal{A}$ と $\mathcal{R}$，Ξ において，書換え規則 $l\to r\in\mathcal{R}$ に対して，$l\sigma_{\mathcal{Q}}\xrightarrow[\mathcal{A}]{*}q$ かつ $r\sigma_{\mathcal{Q}}\not\xrightarrow[\mathcal{A}]{*}q$ を満たす Q-代入 $\sigma_{\mathcal{Q}}$ を探す．そして，$r\sigma_{\mathcal{Q}}$，q を正規化し，遷移規則の集合 $Norm_\xi(r\sigma_{\mathcal{Q}}\to q)$ の要素を $\mathcal{A}$ の遷移規則に追加する．この処理を繰り返し，遷移規則が追加されなくなったとき，$\mathcal{A}$ を近似 TA として出力する．

Timbuk のアルゴリズムを以下に記す．

入力 TRS $\mathcal{R}$，TA $\mathcal{A}$，正規化規則の集合 Ξ

出力 $\xrightarrow[R]{*}L(\mathcal{A})$ の近似 TA $\mathcal{A}^{Tim}$

手順 1. から順に処理を行う．

1. $i=0$．$\mathcal{A}_0=(\mathcal{F},\mathcal{Q},\mathcal{Q}_f,\Delta_0)=\mathcal{A}$.
2. $l\sigma_{\mathcal{Q}}\xrightarrow[\Delta_i]{*}q$ かつ $r\sigma_{\mathcal{Q}}\not\xrightarrow[\Delta_i]{*}q$ を満たす書換え規則 $l\to r\in\mathcal{R}$，状態 $q\in\mathcal{Q}$，Q-代入 $\sigma_{\mathcal{Q}}$ が存在するならば，$r\sigma_{\mathcal{Q}}$ の形により場合分けして遷移規則を Δ_{i+1} に追加する．
   (a) $r\sigma_{\mathcal{Q}}\in\mathcal{Q}$ の場合
   $\Delta_{i+1}=\Delta_i\cup\bigcup_{s\to q\in\Delta_i}(s\to r\sigma_{\mathcal{Q}})$
   (b) $r\sigma_{\mathcal{Q}}=g(q_1,...,q_m)$ の場合
   $\Delta_{i+1}=\Delta_i\cup\{r\sigma_{\mathcal{Q}}\to q\}$
   (c) 上記以外の場合
   $\Delta_{i+1}=\Delta_i\cup Norm_\xi(r\sigma_{\mathcal{Q}}\to q)$
   ($Norm_\xi(r\sigma_{\mathcal{Q}}\to q)$ が生成できないならばユーザーに正規化規則の入力を求める．入力が無い場合は正規化は行わない．)
3. $\Delta_i=\Delta_{i+1}$ ならば $\mathcal{A}^{Tim}=\mathcal{A}_i$ として終了．そうでないとき，i に 1 を加えて，2. へ戻る．

定理 4.1 (Timbuk の出力 TA の受理集合 [1]) 入力した TRS $\mathcal{R}$，TA $\mathcal{A}$ が左線形性を満たし，Timbuk が近似 TA $\mathcal{A}^{Tim}$ を出力した場合，以下の関係が成立する．

$$L(\mathcal{A}^{Tim})\supseteq\xrightarrow[\mathcal{R}]{*}(L(\mathcal{A}))$$

5 Timbuk による Jacquemard 法の自動化

Timbuk は近似 TA を出力するためには，適切な入力をユーザーが与える必要がある．そこで本節では，Timbuk が近似 TA を出力する入力生成アルゴリズムを提案する．そのために，まず Timbuk が全

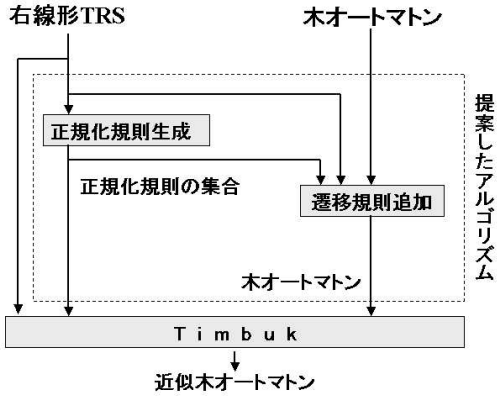

図 1: Timbuk を用いた近似 TA 自動生成図

ての TRS に対して出力を保証する入力の十分条件を定義する．次に，Jacquemard の近似 TA 生成アルゴリズムに基いて，Timbuk の入力生成アルゴリズムを提案する．最後に，提案したアルゴリズムの出力を Timbuk に入力した近似 TA は，Jacquemard の手法で生成した近似 TA と同じ近似精度をもつことを証明する．

補題 5.1 (Timbuk が出力する十分条件)
Timbuk への TRS $\mathcal{R}$，TA $\mathcal{A}$，正規化規則の集合 Ξ を入力する．Ξ が $\mathcal{R}$ に対して完全ならば，Timbuk は停止し TA を出力する． □

以下において，Jacquemard の提案手法で生成する近似 TA を，Timbuk で生成できるような，Timbuk の入力生成アルゴリズムを提案する．

このアルゴリズム（図 1）は，TRS $\mathcal{R}$ と TA $\mathcal{A}$ を入力とし，以下の 2 つの処理を行い，TA と正規化規則を出力する．

1. $\mathcal{R}$ に対して完全な正規化規則の集合を生成する．
2. $\mathcal{R}$ の全ての右辺の具体項が状態項に遷移する様に，$\mathcal{A}$ に状態と遷移規則を追加する．

以下では各処理ごとにわけて，提案したアルゴリズムを説明する．

入力 TRS に対して完全な正規化規則の生成処理

この処理では，Timbuk の入力を満たすように，入力された TRS $\mathcal{R}$ の全ての右辺に対して正規化規則を生成し，正規化規則の集合を出力する．生成される正規化規則 $((r,z),\alpha)$ は，$\mathcal{R}$ に対して完全かつ，全ての状態 q に対して部分関数 α_q が一意に決まる．

入力 TRS $\mathcal{R}$

出力 決定的かつ $\mathcal{R}$ に対して完全な正規化規則 Ξ

手順 以下の処理を順番に行う．

1. $\Xi_0 = \phi$．$k = 0$
2. 全ての書換え規則 $l \to g(r_1,...,r_n) \in \mathcal{R}$ に対し，正規化規則 ξ と割り当てる状態を以下のように定義し，Ξ_k に加える．
   $\xi = ((g(r_1,...,r_n),z),(g(w_1,...,w_n),z) \uplus \bigcup_{h(s_1,...,s_m)\lhd g(r_1,...,r_n)}(h(w'_1,...,w'_m), q_{h(s_1,...,s_m)}))$
   ただし，$w_i, w'_j = \begin{cases} s_i & (s_i \in \mathcal{X}) \\ q_{s_i} & (\text{その他}) \end{cases}$ とする．
3. $\Xi_{k-1} = \Xi_k$ ならば，$\Xi = \Xi_k$ として出力．
   そうでなければ，k に 1 を足して 2 へ戻る．

右辺項を状態項へ遷移させるための追加処理

この処理では，Jacquemard の手法に基いて，入力された TRS の書換え規則の全ての右辺の具体項が状態項に遷移できるように，入力された TA に状態と遷移規則を追加する．その際，追加する状態は正規化規則にもとづいて決定する．

入力 TRS $\mathcal{R}$，TA $\mathcal{A} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta)$，正規化規則の集合 Ξ

出力 TA $\mathcal{A}_0^{Tim}$

手順 以下の処理を順番に行う．

1. $\mathcal{A}^0 = \mathcal{A}$，$k = 0$.
2. 全ての書換え規則 $l \to r \in \mathcal{R}$ の，右辺項の全ての非変数な真部分項 $h(r'_1,...,h'_m)$ に対して，正規化規則 $((r,z),\alpha) \in \Xi$ にもとづき次のように遷移規則を生成し，$\mathcal{A}^k$ に加える．
   $h(q_1,...,q_m) \to q_{m+1}$

ただし，$q_i = \begin{cases} q & ((r_i', q) \in \alpha) \\ q_\perp & (r_i' \in \mathcal{X}) \end{cases}$ とする．
（r_i' が変数の場合，状態 $q_\perp$ を生成して $\mathcal{A}$ に加える．）

3. $\mathcal{A}^{k-1} = \mathcal{A}^k$ ならば，$\mathcal{A}^k = \mathcal{A}_0^{Tim}$ として出力．そうでなければ，k に 1 を足して，2 へ戻る．

定理 5.1 TRS $\mathcal{R}$，TA $\mathcal{A}$ を，提案手法に入力して得た近似 TA を $\mathcal{A}^{Tim}$，Jacquemard の手法に入力して得た近似 TA を $\mathcal{A}^{IG}$ とする．その場合，全ての右線形 TRS $\mathcal{R}$ と TA $\mathcal{A}$ に対して，$\mathcal{A}^{Tim} = \mathcal{A}^{IG}$ が成立する．

略証 Jacquemard のアルゴリズムのループ回数に関する帰納法と，Timbuk のアルゴリズムのループ回数に関する帰納法で証明される． □

6 Timbuk 補助ツールの設計

5 節で提案したアルゴリズムは，入力した TRS $\mathcal{R}$ が右線形の場合に限り，Timbuk が集合 $\xrightarrow[\mathcal{R}]{*} L(\mathcal{A})$ の近似 TA を出力する入力を生成する．

しかし，定理 4.1 によれば，Timbuk の入力 TRS $\mathcal{R}$，TA $\mathcal{A}$ が左線形性を満たせばその出力として集合 $\xrightarrow[\mathcal{R}]{*} L(\mathcal{A})$ の近似 TA が得られる．この定理に基づき，本節では提案したアルゴリズムの TA への遷移規則追加処理を変更することにより，アルゴリズムを提案する．そして，改良したアルゴリズムを用いて Timbuk が出力した近似 TA は，Jacquemard の手法と比較して，全ての TRS に対応し，かつ近似精度は低下せず一部の TRS において向上することを示す．

改良したアルゴリズム（図 2）内の処理のうち，正規化規則生成処理は前節と同じである． よって，変更した TA の生成処理のみを以下に記す．この処理では，まず入力された TRS $\mathcal{R}$ と TA $\mathcal{A}$ が左線形性の十分条件を満たすかを確認する．

定理 6.1 (左線形性の十分条件 [1]) 以下のいずれかの条件を満たすならば，TRS $\mathcal{R}$ と TA $\mathcal{A}$ は左線形性を持つ．
- $\mathcal{R}$ が左線形
- $\mathcal{A}$ が決定 TA □

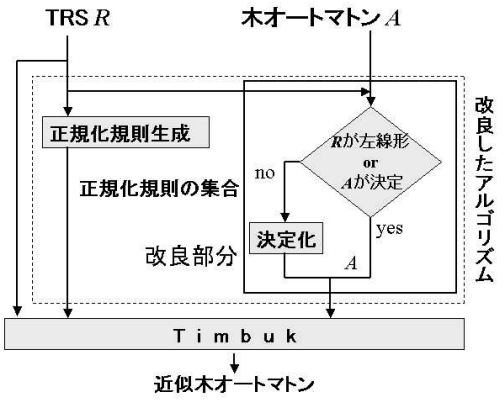

図 2: Timbuk を用いた近似 TA 自動生成図 (改良版)

左線形性を満たす木オートマトン生成処理

この処理では，入力された TRS $\mathcal{R}$ と TA $\mathcal{A}$ が左線形性の十分条件を満たす場合は $\mathcal{A}$ をそのまま出力し，持っていない場合は TA を決定化して出力する．

入力 TRS $\mathcal{R}$，TA $\mathcal{A}$

出力 TA $\mathcal{A}_0^{Tim'}$

手順 $(\mathcal{R}, \mathcal{A})$ が左線形性の十分条件を満たすかで場合分けする．

- $\mathcal{R}$ が左線形，または $\mathcal{A}$ が決定 TA である場合 $\mathcal{A}_0^{Tim'} = \mathcal{A}$ として出力する．
- それ以外の場合 $\mathcal{A}$ の決定 TA を $\mathcal{A}_0^{Tim'}$ として出力する．

改良したアルゴリズムを用いて Timbuk が出力した近似 TA が，右線形 TRS において Jacquemard 法と比較して近似精度が悪化せず，一部の TRS において向上することを示す．

その準備として，まず Timbuk において次の定理が成立することを証明する．

定理 6.2 与えられた 2 つの TA $\mathcal{A}, \mathcal{A}'$ が $\mathcal{A}' \subseteq \mathcal{A}$ を満たすならば，任意の TRS $\mathcal{R}$ と $\mathcal{R}$ に対して完全な正規化規則 Ξ に対して，$\mathcal{A}, \mathcal{R}, \Xi$ を Timbuk に入力して得た近似 TA $\mathcal{A}^{Tim}$ と，$\mathcal{A}', \mathcal{R}, \Xi$ を Timbuk に入力して得た近似 TA $\mathcal{A}^{Tim'}$ に対して，$L(\mathcal{A}^{Tim'}) \subseteq L(\mathcal{A}^{Tim})$ が成立する．

略証 Timbukが出力するTA A^{Tim} と $A^{Tim'}$ を生成する際に，アルゴリズムのループ回数 k にて生成されたTAをそれぞれ A_k^{Tim}，$A_k^{Tim'}$ とする．

Timbukのアルゴリズムのループ回数 k に関する帰納法で，全ての k に対して $L(\mathcal{A}_k^{Tim'}) \subseteq L(\mathcal{A}_k^{Tim})$ が成立することが証明される． □

以上より，次の定理が得られる．

定理 6.3 TRS $\mathcal{R}$，TA $\mathcal{A}$ に改良手法，改良前の提案手法，Jacquemard法を適用して得た近似TAを，それぞれ $\mathcal{A}^{Tim'}$，$\mathcal{A}^{Tim}$，$\mathcal{A}^{IG}$ とする．このとき，$\mathcal{R}$，$\mathcal{A}$ が左線形性を満たし，かつ $\mathcal{R}$ が右線形ならば，$\xrightarrow[\mathcal{R}]{*} L(\mathcal{A}) \subseteq L(\mathcal{A}^{Tim'}) \subseteq L(\mathcal{A}^{Tim}) = L(\mathcal{A}^{IG})$ が成立する．

証明 補題5.1より，$\mathcal{R}$ と $\mathcal{A}$ に対して，Timbukは $\mathcal{A}^{Tim'}$，$\mathcal{A}^{Tim}$ を出力する．$\mathcal{R}$，$\mathcal{A}$ が左線形性を満たすので，定理4.1より，$\xrightarrow[\mathcal{R}]{*} (L(\mathcal{A})) \subseteq L(\mathcal{A}^{Tim'})$ が成立する．

定理6.2より，$L(\mathcal{A}^{Tim'}) \subseteq L(\mathcal{A}^{Tim})$ が成立する．

定理5.1より，$\mathcal{A}^{Tim} = \mathcal{A}^{IG}$ なので，$L(\mathcal{A}^{Tim}) = L(\mathcal{A}^{IG})$ が成立する．

よって，$\xrightarrow[\mathcal{R}]{*} L(\mathcal{A}) \subseteq L(\mathcal{A}^{Tim'}) \subseteq L(\mathcal{A}^{Tim}) = L(\mathcal{A}^{IG})$ が成立する． □

改良手法の方が改良前の手法より近似精度が向上する例を示す．

例 6.1 TRS $\mathcal{R} = \{f(x,y) \to f(x,g(y))\}$，TA $\mathcal{A} = (\{a,b,f(,),g()\},\{q_a,q_b,q_f\},\{q_f\},\{a \to q_a, b \to q_b, f(q_a,q_b) \to q_f\})$ に対して，改良手法，改良前の提案手法，Jacquemard法を適用して得た近似TAを，それぞれ $\mathcal{A}^{Tim'}$，$\mathcal{A}^{Tim}$，$\mathcal{A}^{IG}$ とする．各近似TAの認識する集合は，$L(\mathcal{A}^{Tim'}) = \{f(a,g^*(b))\}$，$L(\mathcal{A}^{Tim}) = L(\mathcal{A}^{IG}) = \{f(a,b), f(a,g^+(s)) \mid s \in T(\mathcal{F})\}$ である． よって，関係 $\xrightarrow[\mathcal{R}]{*} (L(\mathcal{A})) = L(\mathcal{A}^{Tim'}) \subset L(\mathcal{A}^{Tim}) = L(\mathcal{A}^{IG})$ が成立する．

例 6.2 TRS $\mathcal{R} = \{g(x) \to g(g(g(g(g(x)))))\}$，TA $\mathcal{A} = (\{a,g()\},\{q_a,q_f\},\{q_f\},\{a \to q_a, g(q_a) \to q_f\})$ に対して，本節，前節，Jacquemardの提案手法を適用して得た近似TAを，それぞれ $\mathcal{A}^{Tim'}$，$\mathcal{A}^{Tim}$，$\mathcal{A}^{IG}$ とする．各近似TAの認識する集合は，$L(\mathcal{A}^{Tim'}) = \{g(a), g^m(a) \mid m \geq 6\}$ である． よって，$L(\mathcal{A}^{Tim}) = L(\mathcal{A}^{IG}) = \{g^m(a) \mid m \geq 3\}$，関係 $\xrightarrow[\mathcal{R}]{*} (L(\mathcal{A})) \subset L(\mathcal{A}^{Tim'}) \subset L(\mathcal{A}^{Tim}) = L(\mathcal{A}^{IG})$ が成立する．

7 まとめ

Jacquemardの手法が生成する近似木オートマトンをTimbukにて生成する手法を提案した．さらに，提案した手法を，全てのTRSに対して近似木オートマトンが生成できるように改良し，改良手法はJacquemardの手法と比較して近似精度は低下せず一部のTRSにおいては向上する事を示した．

今後の課題として，改良したアルゴリズムとTimbukによって到達可能集合を受理するTAを出力するTRSのクラスの発見，アルゴリズムのさらなる改良による近似精度の向上があげられる．また，Timbukは左変数TRSや余剰変数が出現する等の特殊なTRSは入力として扱わない．そのため，それらTRSに対応するようにTimbuk自身を改良することがあげられる．

参考文献

[1] Guillaume Feuillade, Thomas Genet and Valerie Viet Triem Tong. “Reachability Analysis of Term Rewriting Systems.” Technical Report RR-4970, INRIA, 2003. (Or the corrected version To be published in Journal of Automated Reasoning, 2004.).

[2] Florent Jacquemard, “Decidable Problems for Term Rewriting Systems ”, Lecture Notes in Computer Science 1103, pp. 362–376, 1996.

[3] Hubert Comon, Max Dauchet, Remi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi, “Tree Automata and Application” http://www.grappa.univ-lille3.fr/tata/

[4] Takashi Nagaya, “Reduction Strategies for Term Rewriting Sustems”,School of Information Science Japan Advanced Institute of Science and Technology (March 1999).

[5] Thomas Genet, and Valerie Viet Triem Tong, : Tree Automata Library. http://www.irisa.fr/lande/genet/Timbuk