人工知能学会研究会資料
SIG-FPAI-B402-10

有限オートマトンに対する一般化曖昧性解消演算の正当性

The correctness of a generalized disambiguation algorithm for finite automata

林 克彦 [1*]　　永田 昌明 [1]
Katsuhiko Hayashi[1]　　Masaaki Nagata[1]

[1] 日本電信電話株式会社 NTT コミュニケーション科学基礎研究所
[1] NTT Communication Science Laboratories, NTT Corp.

Abstract: We present a generalized disambiguation algorithm of finite state automata, and show a proof of its correctness. This algorithm can remove ambiguities of finite state and tree automata. Our proposed algorithm can make finite state and tree automata more efficient to use in many applications.

1 はじめに

有限状態オートマトン (Finite State Automaton: FSA) は記号列を入力にとり，記号の読み込みとそれに応じた動作を時系列的に行う計算モデルである [4]．一方，Thatcher と Wright は，入力にラベル付きの木をとる有限木オートマトン (Finite Tree Automaton: FTA) を提案し，FSA を代数的に一般化した [8].

FSA と FTA は形式言語，音声言語処理，バイオインフォマティクス，機械学習などの問題をモデル化するために幅広く応用されている．しかし，現実世界の問題を表したオートマトンは一般に複雑なものとなり，そのままでは非効率な動作をとることが多い．

オートマトンの最適化演算の 1 つである決定化は，各状態から同じ記号での遷移を排除し，その動作を効率化できるため，実用上では欠かせない演算となっている [1, 5]．ただし，決定化で作られるオートマトンの状態数は，最悪の場合，元のオートマトンの状態数 N の $\Omega(2^N)$ となるため記憶容量が問題となる [7, 2].

一方，決定性に類似した概念に**無曖昧性** (unambiguous) と呼ばれるものがある [7]．オートマトンが無曖昧とは，ある入力を受理できる経路が 1 つしか存在しない場合を言う．近年，Mohri は FSA を無曖昧化するための**曖昧性解消演算** (disambiguation) を提案し，記憶容量の点において，決定化よりも優れることを報告している [6]．実際，無曖昧な FSA の状態数は元の等価な FSA の状態数 N に対して，たかだか $\Omega(2^{\sqrt{N}})$ にしかならない [7].

しかし，Mohri の曖昧性解消演算は，その適用範囲が FSA に限定されているという問題がある．そこで，本稿では曖昧性解消演算を FTA へも適用可能な形に一般化することを試みる．また，提案するアルゴリズムの正当性についても細かく分析し，証明を与える．本稿の貢献によって，FSA や FTA を使った実問題のモデル化がさらに進むことが大きく期待される．

2 木の定義

ランク付きアルファベットは組 (Σ, rk) とし，記号の有限集合 Σ と写像 $rk : \Sigma \to \mathbb{N}$ から成る．写像 rk は Σ の全ての要素にランクを付与する．全ての $k \in \mathbb{N}$ に対し，$\Sigma^{(k)} \subseteq \Sigma$ を $rk(\sigma) = k$ となる記号 $\sigma \in \Sigma$ の集合とする．$\sigma \in \Sigma^{(k)}$ を $\sigma^{(k)}$ と書くが，自明な場合は (k) を省略する．T_Σ は次の条件を満たす要素の集合とする．

- 全ての $\sigma \in \Sigma^{(0)}$ に対して，σ から成る単一の頂点 $\sigma()$ は T_Σ に含まれる木である．
- 全ての $\sigma \in \Sigma^{(k)}$ $(k \geq 1)$ と $t_1, \ldots, t_k \in T_\Sigma$ に対して，σ でラベル付けされ，木 $t_1, \ldots, t_k$ を子に持つ $\sigma(t_1, \ldots, t_k)$ は T_Σ に含まれる木である．

木 $t = \sigma(t_1, \ldots, t_k)$ の位置集合 $Pos(t)$ を

$$Pos(t) = \{root\} \cup \{i.w | 1 \leq i \leq k, w \in Pos(t_i)\}$$

として定義する．$root$ は t のルート記号の位置とする．葉の位置集合 $leaves(t)$ は

$$leaves(t) = \{w | w \in Pos(t), \forall i \in \mathbb{N}, w.i \notin Pos(t)\}$$

として定義される．木 t のサイズは $|Pos(t)|$ とし，位置 pos にある記号は $t(pos)$ として書く．また，木 t の位置 pos の部分木を $t|_{pos}$ とする．木 t の高さは $height(t) = 1 + \max(height(t|_i) | 1 \leq i \leq rk(t(root)))$ として定義する．例えば，木 $t = \sigma(\gamma(\alpha, \beta(z)), \alpha, z(z))$ を考えると，$Pos(t) = \{root, 1, 1.1, 1.2, 1.2.1, 2, 3, 3.1\}$，$leaves(t) = \{1.1, 1.2.1, 2, 3.1\}$，$size(t) = 8$，$t(root) = \sigma$，$t(1.1) = \alpha$，$t|_{1.2} = \beta(z)$，$height(t) = 4$ となる．

*連絡先: NTT コミュニケーション科学基礎研究所
〒 619-0237 京都府相楽郡精華町光台 2-4
E-mail: hayashi.katsuhiko@lab.ntt.co.jp

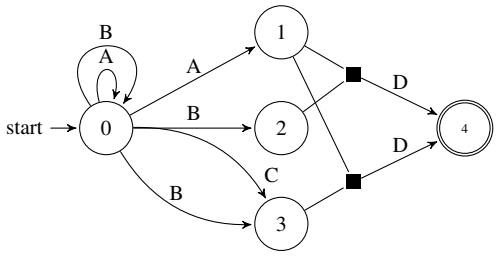

図 1: 有限木オートマトンの例.

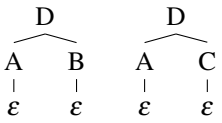

図 2: 図 1 のオートマトンで受理可能な木.

3 有限木オートマトンの定義

(上昇型) 有限木オートマトン (FTA) $A = (\Sigma, Q, i, F, E)$ を次のように定義する.

- Σ はランク付きアルファベット,
- Q は状態の有限集合,
- $i \in Q$ は初期状態,
- $F \subseteq Q$ は終了状態の集合,
- $E \subseteq \overbrace{Q \times \cdots \times Q}^{k} \times \Sigma^{(k)} \times Q$ は k 個の状態から成るベクトルを入力にして次の状態へと遷移する辺の集合である. 辺は $\sigma(q_1, \ldots, q_k) \to q$ $(q, q_1, \ldots, q_k \in Q,\ \sigma \in \Sigma^{(k)})$ として書く.

木 t $(\in T_\Sigma)$ に対する実行 r は写像 $r : Pos(t) \to Q$ として定義され, $w \in Pos(t)$ に対して, $t(w) = \sigma^{(k)}$ $(rk(t(w)) = k)$ であるとき, $\sigma^{(k)}(r(w.1), \ldots, r(w.k)) \to r(w) \in E$ が満たされる必要がある $(w.1, \ldots, w.k \in Pos(t))$. A において, 木 t に対する全ての実行の集合を $R_A(t)$ と書く. もし, $r(root) \in F \wedge (\forall pos \in leaves(t), r(pos) = i)$ となる r が $Run_A(t)$ に存在するならば, t は A によって**受理可能**と呼ぶ. また, そのような実行 r を**受理可能な実行**と呼ぶ. A の言語 $L(A) = \{t \mid \forall t \in T_\Sigma, t$ は A によって受理可能 $\}$ として定義する. $L(A) = L(A')$ のとき, A が木オートマトン A' と**等価**と呼ぶ. A が**無曖昧**であるとは, 全ての $t \in L(A)$ に対して, $Run_A(t)$ には高々1 つの受理可能な実行しか存在しないときをいう.

複数の状態から構成されるベクトルの集合 $V \subseteq Q^*$, 状態の集合 $U \subseteq Q$, 木 $t \in T_\Sigma$ に対して, V 中のベクトルから U 中の状態へ木 t で遷移する実行の集合を $Run(V, t, U)$ と書き, V が単集合 $\{\{p_1, \ldots, p_k\}\}$ のとき, 単純に $W(\{p_1, \ldots, p_k\}, t, U)$ と書き, U に対しても同様に書く. さらに, V 中のベクトルを始点とし, 木 $t \in T_\Sigma$ で到達できる状態の集合を $\delta(V, t)$ と書く. $Run(\{p_1, \ldots, p_j, \ldots, p_k\}, t, F) \cap Run(\{q_1, \ldots, q_j, \ldots, q_k\}, t, F) \neq \emptyset$ $(1 \leq j \leq k)$ となる木 $t \in T_\Sigma$ が存在するとき, 2 つの状態 p_j と q_j は**将来を共有する**と呼ぶ.

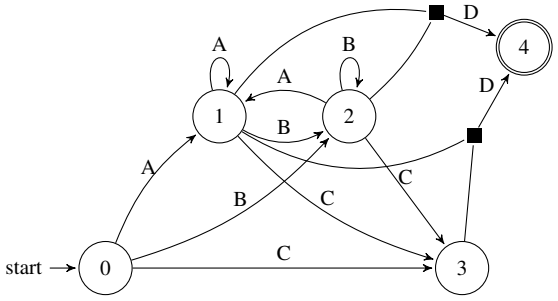

図 3: 図 1 のオートマトンに対する決定化の結果.

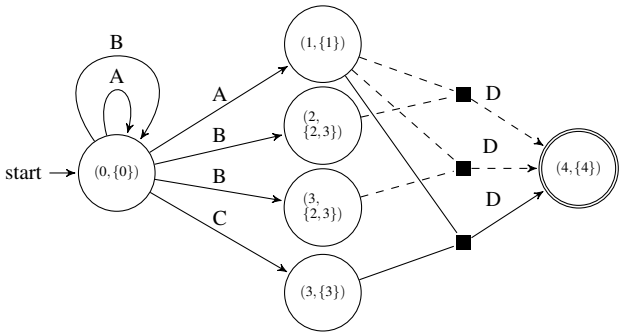

図 4: 図 1 のオートマトンに対する曖昧性解消の結果.

FTA の例を図 1 に示す. このオートマトンは次の要素

- $\Sigma = \{\mathrm{A}^{(1)}, \mathrm{B}^{(1)}, \mathrm{C}^{(1)}, \mathrm{D}^{(2)}\}$, $Q = \{0, 1, 2, 3, 4\}$,
- $i = 0$, $F = \{4\}$,
- $E = \{\varepsilon() \to 0, \mathrm{A}(0) \to 0, \mathrm{B}(0) \to 0, \mathrm{A}(0) \to 1, \mathrm{B}(0) \to 2, \mathrm{C}(0) \to 3, \mathrm{B}(0) \to 3, \mathrm{D}(1,2) \to 4, \mathrm{D}(1,3) \to 4\}$,

から構成される[1]. 図 2 には図 1 の FTA によって受理することができる木の一部を示す. 図や表記からは, 本文を見やすくするため, 適宜, 記号からランクを除いて書いた. 図 2 の左に示した木 D(A B) は次の二つの実行 r_1, r_2 によって受理される.

$$r_1 = [0, 1, 2, 4], \qquad r_2 = [0, 1, 3, 4]$$

よって, 図 1 の FTA は曖昧性を持つ. 図 3 には図 1 の FTA に決定化 [5] を適用した結果を示した. 図 3 からは決定化の結果が複雑な FTA となっていることがわかる.

4 一般化曖昧性解消演算

Algorithm 1 に有限木オートマトン (FTA) を適用可能な形に一般化した曖昧性解消演算アルゴリズムの疑似

[1] FTA では複数の状態を始点として次の状態へと遷移することが許され, 超グラフ [3] の構造で表される.

Algorithm 1 一般化曖昧性解消演算アルゴリズム

procedure DISAMBIGUATION($A = \{\Sigma, Q, i, F, E\}$)
  $B \leftarrow trim(A \cap A)$
  $i' \leftarrow (i, \{i\})$
  $Q' \leftarrow Q' \cup \{i'\}$
  $Push(\mathscr{Q}, i')$
  $R \leftarrow R \cup \{(i', i')\}$
  $Add(M(i), i')$
  while $\mathscr{Q} \neq \emptyset$ **do**
    $(p, s) \leftarrow Pop(\mathscr{Q})$
    if $(p \in F)$ and $((p', s') \notin F')$ with $(p', s')R(p, s)$ **then**
      $F' \leftarrow F' \cup \{(p, s)\}$
    end if
    for $\sigma^{(k)}(p_1, \ldots, p, \ldots, p_k) \to q \in E$ **do**
      $cand \leftarrow GetCandidate(M, \{p_1 \ldots p \ldots p_k\}, (p, s))$
      while $|cand| > 0$ **do**
        $\{\{(p_1, s_1) \ldots (p, s) \ldots (p_k, s_k)\}, \mathbf{v}\} \leftarrow Pop(cand)$ //$s_1 = \{p''_{1,1}, \ldots, p''_{1,\ell_1}\}, \ldots, s_k = \{p''_{k,1}, \ldots, p''_{k,\ell_k}\}$
        for $j_1, \ldots, j_k \in [1, \ell_1], \ldots, [1, \ell_k]$ **do** // 全ての組み合わせについて
          $t \leftarrow \{r \in \delta(\{p''_{1,j_1}, \ldots, p, \ldots, p''_{k,j_k}\}, \sigma^{(k)}) : (q, r) \in B\}$
        end for
        if $\sigma^{(k)}((p'_1, s'_1), \ldots, (p', s'), \ldots, (p'_k, s'_k)) \to (q, t) \notin E'$ with $(p'_1, s'_1)R(p_1, s_1) \ldots (p', s')R(p, s) \ldots (p'_k, s'_k)R(p_k, s_k)$ **then**
          if $(q, t) \notin Q'$ **then**
            $Q' \leftarrow Q' \cup \{(q, t)\}$
            $Push(\mathscr{Q}, (q, t))$
            $Add(M(q), (q, t))$
          end if
          $E' \leftarrow E' \cup \{\sigma^{(k)}((p_1, s_1), \ldots, (p, s), \ldots, (p_k, s_k)) \to (q, t)\}$
          for $\{(p'_1, s'_1) \ldots (p', s') \ldots (p'_k, s'_k)\}$ s.t. $((p'_1, s'_1)R(p_1, s_1) \ldots (p', s')R(p, s) \ldots (p'_k, s'_k)R(p_k, s_k))$
              and $\sigma^{(k)}((p'_1, s'_1), \ldots, (p', s'), \ldots, (p'_k, s'_k)) \to (q', t') \in E'$ **do**
            $R \leftarrow R \cup \{(q, t), (q', t')\}$
          end for
        end if
        $AppendNext(cand, M, \{\{(p_1, s_1) \ldots (p, s) \ldots (p_k, s_k)\}, \mathbf{v}\})$
      end while
    end for
  end while
  return $A' = \{\Sigma, Q', i', F', E'\}$
end procedure

procedure GETCANDIDATE($M, \{p_1 \ldots p \ldots p_k\}, (p, s)$)
  $cand \leftarrow \{\}$
  if $|M(p_1)| > 0$ and $\ldots$ and $|M(p_k)| > 0$ **then**
    $cand \leftarrow cand \cup \{\{Get(M(p_1), 1) \ldots (p, s) \ldots Get(M(p_k), 1)\}, \mathbf{1}\}$
  end if
  return $cand$
end procedure

procedure APPENDNEXT($cand, M, \{\{(p_1, s_1) \ldots (p, s) \ldots (p_k, s_k)\}, \mathbf{v}\}$)
  for $i \leftarrow 1 \ldots k$ **do**
    if $p_i \neq p$ **then**
      $\mathbf{v}' \leftarrow \mathbf{v} + \mathbf{b}^i$
      if $v'_i \leq |M(p_i)|$ **then**
        $(p_i, s'_i) \leftarrow Get(M(p_i), v'_i)$
        if $\{\{(p_1, s_1) \ldots (p_i, s'_i) \ldots (p_k, s_k)\}, \mathbf{v}'\} \notin cand$ **then**
          $cand \leftarrow cand \cup \{\{(p_1, s_1) \ldots (p_i, s'_i) \ldots (p_k, s_k)\}, \mathbf{v}'\}$
        end if
      end if
    end if
  end for
end procedure

コードを示す．このアルゴリズムはFTA $A=(\Sigma,Q,i,F,E)$ を入力として，FTA $A'=(\Sigma,Q',i',F',E')$ を出力する．

このアルゴリズムでは A' における 2 つの状態に対して，同値関係 R を定義する．R は 2 つの状態が同じ木で到達できる実行を持つときに関係を持つ．疑似コードの 2 行目では，A と A の積集合を計算し，その結果から終了状態へ到達できない状態を排除した FTA を B に代入する[2]．B に存在する状態は (q,r) $(q,r\in Q)$ の形をとり，q と r が将来を共有するかどうかを定数時間でチェックするために B を使う．

3 行目では，A' の初期状態 $(i,\{i\})$ として初期化する．A' の状態は (p,s) $(p\in Q, s\subseteq Q*)$ の形をとる．s は同じ木で到達でき，p と将来を共有する状態の集合である．i は ε 遷移によって，i に到達すると考えることができる．4 行目では新たな状態の集合 Q' に，5 行目ではキュー $\mathcal{Q}$ に i' を追加する．6 行目では関係 R に (i',i') を追加する．7 行目では，リスト $M(i)$ に i' を追加する．リスト $M(p)$ は $(p,\cdot)\in Q'$ を取り出すために構築する．

8 から 35 行目は新たな状態と辺を構築する繰り返し処理になっている．10 から 12 行目では 9 行目でキューから取り出し状態 (p,s) が終了状態であるかをチェックし，F' に登録する．ただし，$(p',s')R(p,s)$ が存在する，すなわち，同じ木で終了状態に至る (p',s') がすでに存在する場合は (p,s) を F' に登録しない．

13 から 34 行目では p から出る辺 $\sigma^{(k)}(p_1,\ldots,p,\ldots,p_k)\to q$ ごとに処理が進む．14 行目の GetCandidate 関数では，$p_1,\ldots,p_k$ に対してすでに構築された A' の状態をリスト $M(\cdot)$ の先頭から取り出す (41 から 43 行目)．ただし，p に対しては (p,s) を使う．42 行目の $\mathbf{1}$ は k 次元の数値ベクトルで，各次元の値は全て 1 をとる．

15 行目から 33 行目の処理は *cand* に登録された A' における k 個の状態集合と k 次元の数値ベクトルのペアがなくなるまで繰り返される．17 から 19 行目では，状態集合 s_1 から状態集合 s_k までに含まれている状態の全組み合わせを考慮して，その各組み合わせ $\{p''_{1,j_1},\ldots,p,\ldots,p''_{k,j_k}\}$ ごとに記号 $\sigma^{(k)}$ で遷移できる状態 r を取り出す．また，r と q が将来を共有するなら，新たな状態集合 t の要素とする．

20 から 31 行目では新たな状態 (q,t) と辺 $\sigma^{(k)}((p_1,s_1),\ldots,(p,s),\ldots,(p_k,s_k))\to(q,t)$ の登録を行う．ただし，20 行目で $(p'_1,s'_1)R(p_1,s_1)\ldots(p',s')R(p,s)\ldots(p'_k,s'_k)R(p_k,s_k)$ を持つ辺 $\sigma^{(k)}((p'_1,s'_1),\ldots,(p',s'),\ldots,(p'_k,s'_k))\to(q,t)$ が E' にすでに存在しないかをチェックし，存在しない場合のみ 26 行目で E' に登録を行う．21 から 25 行目では (q,t) が Q' に登録されていないとき，Q'，$\mathcal{Q}$，$M(q)$ への登録を行う．

27 から 30 行目では (q,t) について関係 R を構築できるか調べる．$((p'_1,s'_1)R(p_1,s_1)\ldots(p',s')R(p,s)\ldots(p'_k,s'_k)R(p_k,s_k))$ を持つ辺 $\sigma^{(k)}((p'_1,s'_1),\ldots,(p',s'),\ldots,(p'_k,s'_k))\to(q',t')$ があるとき，(q,t) と (q',t') は同じ木で到達できることを意味する．よって，29 行目でそれらを関係 R に登録する．

32 行目の AppendNext 関数では，状態 $p_1,\ldots,p,\ldots,p_k$ の p 以外の状態に対して，リスト M に登録された次の状態へ更新することで，A' における k 個の状態集合と k 次元の数値ベクトルの新たなペアを作り出す．AppendNext 関数の 50 行目の $\mathbf{b}^i$ は i 次元目だけが 1，その他の次元は全て 0 を持つ k 次元の数値ベクトルを表す．51 から 56 行目では，i 番目の状態を $M(p_i)$ に登録されている次の状態に置き換えることで，新たなペアを *cand* に追加している．

5 分析

5.1 アルゴリズムの停止性

Algorithm 1 では作られる状態と辺の数が有限であるため，停止することが保証される．なぜなら，FTA A の状態に対して，可能な部分集合 s の数は有限なので，作られる状態 (p,s) の数も有限となり，また，(p,s) から作られる辺の数は，多くても，A における p から出る辺の数に一致するからである．最悪の場合，部分集合 s を指数関数的に構築することがあるので，Algortihm 1 の動作には指数関数時間かかる．

5.2 アルゴリズムの正当性

補題 5.1 有限木オートマトン *(FTA)* A に対して，*Algorithm 1* を動かしたとき，状態 (q,t) と (q',t') が作られるとする．そのとき，(q,t) と (q',t') が初期状態から同じ木で到達できるなら，関係 $(q',t')R(q,t)$ が作られる．

証明 木の高さ h に対して，数学的帰納法により証明する．(q,t) と (q',t') が初期状態の場合，*Algorithm 1* の *6* 行目から関係 $(q',t')R(q,t)$ が作られ，共に ε で到達できる．木の高さ $h\le n$ のとき，補題 *5.1* が成り立つと仮定する．ここで $x=\sigma(x_1,\ldots,x_k)$ $(\sigma\in\Sigma^{(k)}, x_1,\ldots,x_k\in T_\Sigma)$ を高さ $n+1$ の木とし，(q,t) と (q',t') は共に x で到達できるとする．このとき，x_1 によって到達する (p_1,s_1)，…，x_k によって到達する (p_k,s_k) が存在し，σ を読み込むことで (q,t) へと到達する．同様に，x_1 によって到達する (p'_1,s'_1)，…，x_k によって到達する (p'_k,s'_k) が存在し，σ を読み込むことで (q',t') へと到達する．ここで仮定から $(p_1,s_1)R(p'_1,s'_1)$，…，$(p_k,s_k)R(p'_k,s'_k)$ が存在し，*Algorithm 1* の *27* から *30* 行目の動作によって，$(q,t)R(q',t')$ が作られることが保証される．逆は *Algorithm 1* の *27* から *30* 行目で自明である． □

命題 5.1 *FTA* A に対して，*Algorithm 1* を動作させたとき，構築される *FTA* A' は曖昧性を持たない．

[2] FTA の積集合演算は $O(|Q|^2)$ で計算でき，$trim(\cdot)$ は状態数に対して線形時間で計算できる．

証明 A' における初期状態 i' から F' への 2 つの実行 d_1 と d_2 は同じ木 x $(x \in T_\Sigma)$ を構築すると考える．
もし $x = \varepsilon$ なら，d_1 と d_2 は初期状態 $(i,\{i\})$ から $(i,\{i\})$ への実行である．よって，$d_1 = d_2$ は明らかである．ここで $d_1(root) = (q_1,t_1)$，$d_2(root) = (q_2,t_2)$ とする．(q_1,t_1) と (q_2,t_2) は共に木 x によって到達できるので，補題 *5.1* から $(q_1,t_1)R(q_2,t_2)$ が存在する．*Algorithm 1* の *10* から *12* 行目の動作で，関係 R を持つ 2 つの状態が共に最終状態になることはないので，$(q_1,t_1) = (q_2,t_2)$ となる．
もし $x = \varepsilon$ なら，d_1 と d_2 は一致する．一方，$x \neq \varepsilon$ なら，$x = \sigma(x_1,\ldots,x_k)$ $(x_1,\ldots,x_k \in T_\Sigma,\ \sigma \in \Sigma^{(k)})$ と書ける．ここで，実行 d_1 と d_2 は状態 (q_1,t_1) へ記号 σ で遷移する辺 $e_1 = \sigma((p'_1,s'_1),\ldots,(p'_k,s'_k)) \to (q_1,t_1)$ と $e_2 = \sigma((p''_1,s''_1),\ldots,(p''_k,s''_k)) \to (q_1,t_1)$ によって，状態 $(p'_1,s'_1),\ldots,(p'_k,s'_k)$ への各実行 $d'_1,\ldots,d'_k$，状態 $(p''_1,s''_1),\ldots,(p''_k,s''_k)$ への各実行 $d''_1,\ldots,d''_k$ へと分解できる．d'_1 と d''_1 は共に木 x_1 に対する実行であり，補題 *5.1* から，関係 $(p'_1,s'_1)R(p''_1,s''_1)$ が存在する．同様に，$(p'_2,s'_2)R(p''_2,s''_2)$，$\ldots$，$(p'_k,s'_k)R(p''_k,s''_k)$ も存在する．ここで *Algorithm 1* の *20* 行目のチェックにより，遷移 e_1 と e_2 の両方が認められることはない．よって，$(p'_1,s'_1) = (p''_1,s''_1)$，$\ldots$，$(p'_k,s'_k) = (p''_k,s''_k)$ が成り立たなければならない．この過程を d'_1 と d''_1，$\ldots$，d'_k と d''_k へ各々再起的に繰り返し適用することで，実行 $d_1 = d_2$ が証明できる． □

補題 5.2 *FTA A* に対して，*Algorithm 1* を動作させたとき，状態 (p,s) が構築されるとする．そのとき，初期状態から木 u と v で (p,s) へ到達できるとすると，A 上で木 u で到達でき，かつ，p と将来を共有する状態の集合は，A 上で木 v で到達でき，かつ，p と将来を共有する状態の集合に一致する．

証明 木 u の高さに対する数学的帰納法によって，次のことを示す．*FTA* A' 上において状態 (p,s) が木 u で到達できるとき，s は A 上において u で到達でき，かつ，p と将来を共有する状態の集合である．
$u = \varepsilon$ のとき，*Algorithm 1* の *3* 行目から上のことは明らかである．$height(u) \leq n$ のとき，上のことが成り立つと仮定する．ここで木 $u = \sigma(u_1,\ldots,u_k)$ とする $(u_1,\ldots,u_k \in T_\Sigma,\ \sigma \in \Sigma^{(k)})$．状態 (p,s) は木 u によって到達できるとき，木 u_1 で到達できる (p_1,s_1)，$\ldots$，木 u_k で到達できる (p_k,s_k) が存在し，記号 $\sigma^{(k)}$ を読み込んで状態 (p,s) へ到達できる必要がある．仮定から s_1 は木 u_1 で到達でき，かつ，p_1 と将来を共有する状態の集合である．$s_2,\ldots,s_k$ についても，仮定からそれぞれ同様のことが言える．*Algorithm 1* の *18* 行目で $s_1,\ldots,s_k$ に対して，その要素の組み合わせを考えて，s が構築される *(*疑似コードでは *t)*．よって，s に含まれる全ての状態は木 u で到達でき，p と将来を共有している．
逆に，木 u で到達でき，かつ，p と将来を共有している状態 q があるとする．そのとき，木 u_1 で到達できる状態 q_1，$\ldots$，木 u_k で到達できる状態 q_k が存在し，記号 σ による遷移が存在する．状態 q_1，$\ldots$，q_k から記号 σ で状態 q へと遷移可能であり，かつ，状態 p_1，$\ldots$，p_k から記号 σ で状態 p へと遷移可能である．また，状態 p と q は将来を共有しているので，状態 p_1 と q_1，$\ldots$，状態 p_k と q_k も各々将来を共有している．仮定から，s_1 は木 u_1 で到達でき，かつ，p_1 と将来を共有している状態の集合である．それゆえ，s_1 は q_1 を含む．同様のことが，s_2，$\ldots$，s_k についても言える．ここで，$q \in \delta(\{q_1,\ldots,q_k\}\sigma^{(k)})$，かつ，$q$ は p と将来を共有しているので，s は q を含む．これより，s は木 u によって到達でき，かつ，p と将来を共有する状態の集合である． □

補題 5.3 *FTA A* に対して，*Algorithm 1* を動作させたとき，*FTA* A' ができるとする．A において，木 x で到達できる状態 q があるとする．そのとき，ある集合 t に対して，木 x で到達できる状態 (q,t) が A' に存在する．

証明 木 x の高さに対する数学的帰納法によって証明する．$x = \varepsilon$ のとき，*Algorithm 1* の *3*，*4* 行目から上の性質は明らかである．$height(x) \leq n$ のとき，補題 *5.3* が成り立つと仮定する．ここで $x = \sigma(x_1,\ldots,x_k)$ とし，$height(x) = n+1$ で $\sigma \in \Sigma^{(k)}$ とする．もし，*FTA A* 上で状態 q が木 x で到達できるならば，木 x_1 で到達できる状態 q_0，$\ldots$，木 x_k で到達できる状態 q_k が A 上には存在し，記号 σ を読み込んで，状態 q へと遷移できる．仮定から，*FTA* A' には，木 x_1 で到達できる状態 (q_1,s_1)，$\ldots$，木 x_k で到達できる状態 (q_k,s_k) が存在する．ここで，*Algorithm 1* の *13* から *26* 行目の作業によって，ある t_0 *(18* 行目で定義される*)* に対して，状態 (q_1,s_1)，$\ldots$，(q_k,s_k) から記号 σ を読み込んで状態 (q,t_0) への遷移が構築されるときは補題 *5.3* の性質が成り立つことは明らかである．一方で，*20* 行目の検証からこの遷移の構築が行われない場合，A' には関係 $(q'_1,s'_1)R(q_1,s_1)$ を持つ状態 (q'_1,s'_1)，$\ldots$，関係 $(q'_k,s'_k)R(q_k,s_k)$ を持つ状態 (q'_k,s'_k) が存在し，記号 σ を読み込んで状態 (q,t_0) への遷移が認められている．ここで，状態 (q'_1,s'_1)，$\ldots$，(q'_k,s'_k) から記号 σ で (q,t_0) への遷移があることから，状態 q'_1，$\ldots$，q'_k から記号 σ で状態 q への遷移が存在することになる．それゆえ，A において，状態 q_1 と q'_1，$\ldots$，状態 q_k と q'_k は各々，将来を共有している．関係 $(q'_1,s'_1)R(q_1,s_1)$，$\ldots$，関係 $(q'_k,s'_k)R(q_k,s_k)$ より，補題 *5.1* から，状態 (q'_1,s'_1) と (q_1,s_1) は共通の木 x'_1，$\ldots$，状態 (q'_k,s'_k) と (q_k,s_k) は共通の木 x'_k で各々到達できる．それゆえ，(q_1,s_1) は木 x_1 と x'_1 の両方，$\ldots$，(q_k,s_k) は木 x_k と x'_k の両方で各々到達可能である．補題 *5.2* から A において，木 x_1 によって到達可能で，かつ，状態 q_1 と将来を共有している状態の集合は，木 x'_1 によって到達可能で，かつ，状態 q_1 と将来を共有している状態の集合と同一である．同様のことは，2，$\ldots$，k でも言える．状態 q_1 と q'_1 は将来を共有し，共に，木 x'_1 で到達できるので，q'_1 には木 x_1 でも到達できなければならない．同様のことは，2，$\ldots$，k でも言える．
ここで木 x_1 で (q'_1,s'_1)，$\ldots$，木 x_k で (q'_k,s'_k) に到達するならば，木 x によって状態 (q,t_0) に到達できることは明らかである．もしそうでなければ，仮定から，A' に

は木 x_1 で到達できる状態 (q''_1, s''_1)，…，木 x_k で到達できる状態 (q''_k, s''_k) が存在し，状態 q''_1，…，q''_k からは記号 σ によって状態 q へと遷移できる．ここで，(q_1, s_1)，…，(q_k, s_k) に対して行った上の説明を，(q''_1, s''_1)，…，(q''_k, s''_k) に対して行う．そうすると，木 x で (q, t_0) へ到達できる実行が見つかるか，または，A' において，状態 (q'''_1, s'''_1)，…，(q'''_k, s'''_k) が見つかり，これらは各々 (q_1, s_1)，…，(q_k, s_k) と同じ性質を持つことがわかる．A' の状態は有限なので，この手続きを繰り返せば，A' において，木 x で到達できる状態が見つかる． □

補題 5.4 *FTA A* に対して，*Algorithm 1* を動作させたとき，*FTA* A' ができるとする．このとき，$L(A') \subseteq L(A)$ である．

証明 A において，木 x で最終状態 $q \in F$ に到達できるとする．補題 *5.3* から，A' において，木 x で到達できる状態 (q_0, t_0) が存在する．もし，*Algorithm 1* の *10* から *12* 行目の操作によって終了状態となれば，木 x は A' によって受理される．一方で，もし終了状態とならなければ，関係 $(q_1, t'_0)R(q_0, t_0)$ を持つ終了状態 (q_1, t'_0) が存在しなければならない．ここで $(q_1, t'_0)R(q_0, t_0)$ が存在することから，それら *2* つの状態は共に木 x' で到達できる．(q_0, t_0) は木 x と x' の両方で到達できるので，補題 *5.2* から，木 x で到達でき，かつ，q_0 と将来を共有する状態の集合と，木 x' で到達でき，かつ，q_0 と将来を共有する状態の集合は同一である．q_0 と q_1 は終了状態であり，共に木 x' で到達できるので，q_1 は q_0 と将来を共有している．よって，q_1 は x で到達可能である．ここで，木 x で状態 (q_1, t'_0) に到達するならば，A' において，木 x で到達できる状態が存在する．もし，そうでないならば，補題 *5.3* から，A' において，木 x で到達できる状態 (q_1, t_1) が存在する．ここで，(q_1, t_1) に対して，(q_0, t_0) と同じように，上の説明を行う．そうすると，A' において，木 x で到達する終了状態が見つかるか，また，(q_0, t_0) と同一の性質を持つ別の状態 (q_2, t_2) $(q_0 \neq q_1 \neq q_2)$ が見つかる．A' の状態は有限なので，この過程を繰り返すことで木 x で到達できる終了状態が見つかる． □

命題 5.2 *FTA A* に対して，*Algorithm 1* を動作させて *FTA* A' ができるとする．このとき，A' は A と等価である．

証明 *Algorithm 1* の動作では，A' に実行 $r = [(p_1, s_1), \ldots, (p_k, s_k)]$ が存在するのは，A に実行 $r' = [p_1, \ldots, p_k]$ が存在するときだけである．また，A' で状態 (p, s) が終了状態となるのは，A に終了状態 p が存在するときだけである．よって，実行 r によって，木 x が受理されるなら，A でも木 x は受理される．これは $L(A) \subseteq L(A')$ を示している．逆は，補題 *5.4* で示した． □

以上，命題 5.1 と命題 5.2 から定理 5.1 が導ける．

定理 5.1 *FTA A* に対して，*Algorithm 1* を動作させて得られる *FTA* A' は曖昧性がなく，A と等価である．

6 むすび

本稿では，有限状態オートマトン及び有限木オートマトンから曖昧性を解消するための一般的な演算を考案し，そのアルゴリズムの正当性に証明を与えた．本稿ではアルゴリズムの原理に焦点を当てたが，実用化ではオートマトンに重みを考えることが重要となる．現在，ここで提案した一般化曖昧性解消演算を重み付きオートマトンへ拡張する研究を進めているが，それについては別の文献で詳細を触れたいと考えている [9].

参考文献

[1] Cyril Allauzen and Mehryar Mohri. On the determinizability of weighted automata and transducers. In *In Proceedings of the WATA*. Citeseer, 2002.

[2] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2007. release October, 12th 2007.

[3] Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. Directed hypergraphs and applications. *Discrete applied mathematics*, 42(2):177–201, 1993.

[4] John E Hopcroft. *Introduction to automata theory, languages, and computation*. Pearson Education India, 1979.

[5] Jonathan May and Kevin Knight. A better n-best list: Practical determinization of weighted finite tree automata. In *In Proceedings of the HLT-NAACL*, pages 351–358. Association for Computational Linguistics, 2006.

[6] Mehryar Mohri. A disambiguation algorithm for finite automata and functional transducers. In *Implementation and Application of Automata*, pages 265–277. Springer, 2012.

[7] Erik Meineche Schmidt. Succinctness of descriptions of context-free, regular, and finite languages. *DAIMI Report Series*, 7(84), 1978.

[8] James W. Thatcher and Jesse B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical systems theory*, 2(1):57–81, 1968.

[9] 林克彦 永田昌明. 重み付き有限オートマトンに対する曖昧性解消演算の一般化とその応用. In **第 *95* 回人工知能学会基本問題研究会**, 2014.