論　文

準拡張正規表現に対する新しい有限オートマトンモデルについて

山本　博章†

New Finite Automata Corresponding to Semi-Extended Regular Expressions

Hiroaki YAMAMOTO†

あらまし　準拡張正規表現とは，共通集合演算をもつ正規表現である．長さ m の正規表現は，たかだか $2m$ 状態の非決定性有限オートマトンに変換できることが知られているが，準拡張正規表現を NFA に変換しようとすると，共通集合演算のため，状態数が指数的に増加してしまう．本論文では，部分入力同期式交代有限オートマトンと呼ばれる新しいモデルを提案し，長さ m の準拡張正規表現はたかだか $2m$ 状態の部分入力同期式交代有限オートマトンに変換できることを示す．この結果の応用としては，Yamamoto [10] による準拡張正規表現の所属問題への適用がある．

キーワード　準拡張正規表現，交代有限オートマトン，同期

1. まえがき

正規表現（正則表現とも呼ばれている），有限オートマトンは，コンパイラ，パターンマッチングアルゴリズムなど計算機科学の分野で広くその理論が使われている．正規表現を非決定性有限オートマトン (NFA) に変換する方法はいくつか知られており（例えば，[1], [3], [4], [8] を参照），多くの応用でこのような変換が用いられている．例えば，パターンマッチング問題への応用として文献 [1], [6], [8] などがある．一方，準拡張正規表現は，これは共通集合演算をもつ正規表現であるが，単純に NFA に変換しようとすると，共通集合演算のため，状態数が指数的に増加してしまう．そのため，所属問題などの応用に対し，単純に有限オートマトンを使うことができない．我々は，正規表現と NFA のように，準拡張正規表現に対しても線形に対応する有限オートマトンを見つけたい．しかし，従来の有限オートマトンでは対応付けが難しい．

本論文は，Chandra [2] らによって導入された交代有限オートマトンに入力同期式と呼ばれる新しい同期機構を導入することによって，部分入力同期式交代有限オートマトンと呼ばれる新しいオートマトンを導入し，長さ m の準拡張正規表現は状態数がたかだか $2m$ の部分入力同期式交代有限オートマトンに変換できることを示す．正規表現を NFA に変換する方法の一つとして，Thompson の方法，これは長さ m の正規表現をたかだか $2m$ 状態の NFA に変換する方法，が知られているが，我々の結果はこの方法を拡張することによって得られる．なお，準拡張正規表現に更に補集合演算を加えたものを拡張正規表現というが，拡張正規表現へ部分入力同期式交代有限オートマトンを適用できるかどうかはわからない．

交代有限オートマトンへの同期機構の導入は，従来，並列計算の能力を調べるために行われた．例えば，Slobodova [7] は交代性オートマトンに同期機構を導入し，その計算能力に関して調べ，同期式交代有限オートマトンは正規言語より広いクラスの言語を受理すること示した．更に，Hromkovic ら [5] は，この結果を改良し，同期式交代有限オートマトンによって受理される言語のクラスは，文脈依存言語のクラスと同じになることを示した．しかし，これらの研究で扱っている同期は我々の入力同期とは違い，準拡張正規表現との対応付けが難しい．入力同期は並列に動作するプロセス間で入力の位置を同期させるものであるのに対し，彼らの同期はプロセス間での適当な記号の通信を可能にするものである．

入力同期式の計算能力については，Yamamoto [9] が調べている．彼は，今回導入するモデルとは少し定義が異なる入力同期式の交代有限オートマトンについ

† 信州大学工学部，長野市
Faculty of Engineering, Shinshu University, 4–17–1 Wakasato, Nagano-shi, 380–8553 Japan

てその能力を調べ，1方向オートマトンの場合，その受理する言語は正規言語であることを示した．この結果から考えて，今回導入する部分入力同期式交代有限オートマトンについても，正規言語だけを受理することが予想されるが，計算能力に関しては今後の課題である．

今回の変換の応用としては，Yamamoto [10] が準拡張正規表現の所属問題に適用しているが，パターンマッチング問題など他分野への応用も期待される．また，入力同期の考えを交代有限オートマトンに導入すること自体も自然な考えであると思われるから，このようなモデル自身の応用も今後期待される．

なお，本論文は，Yamamoto [10] における部分入力同期条件の定義の不十分な点を修正し，完全な証明を与えるものになっている．

2. 準拡張正規表現

以下で，準拡張正規表現の定義を与える．

[定義 1] Σ をアルファベットとする．そのとき，Σ 上の準拡張正規表現（Semi-Extended Regular Expression, SERE と略記する）は以下のように定義される．

（1） $\emptyset$, ϵ 及び $a\ (\in \Sigma)$ は SERE であり，それぞれ空集合，$\{\epsilon\}$, $\{a\}$ なる集合を表す．ここで，ϵ は空記号列を表す．

（2） r_1, r_2 を，それぞれ集合 R_1, R_2 を表す SERE とする．そのとき，$(r_1 \vee r_2)$, $(r_1 r_2)$, (r_1^*) 及び $(r_1 \wedge r_2)$ もまた SERE で，それぞれ集合 $R_1 \cup R_2$（和集合），$R_1 R_2$（連接），R_1^*（閉包）及び $R_1 \cap R_2$（共通集合）を表す．

共通集合のための演算子 $\wedge$ がないのが正規表現であり，表現の長さのたかだか2倍の状態数をもつ NFA に変換できることが知られている．SERE r の表す言語を $L(r)$ と書く．

3. 部分入力同期式交代有限オートマトン

さて次に，我々は新しいオートマトンモデルである部分入力同期式交代有限オートマトン（Partially Input-Synchronized Alternating Finite Automaton, PISAFA と略記する）を定義する．これは，交代有限オートマトンに入力同期という考えを導入して一般化したものである．

[定義 2] PISAFA M は次のような8項組 $M = (Q, S, \Sigma, \delta, q_0, \mu, \psi, F)$ で定義される：

- Q は状態の有限集合．
- $S\ (\subseteq Q)$ は同期状態の集合．
- Σ は入力アルファベット．
- $q_0\ (\in Q)$ は初期状態．
- μ は Q から $\{\vee, \wedge\}$ への関数．もし $\mu(q) = \wedge$ ならば，q は全称状態と呼ばれ，もし $\mu(q) = \vee$ ならば，存在状態と呼ばれる．
- ψ は S から $S \cup \{\perp\}$ への関数で，親関数と呼ばれる．
- $F\ (\subseteq Q)$ は最終状態の集合．
- δ は $Q \times (\Sigma \cup \{\epsilon\})$ から 2^Q への関数で，状態遷移関数と呼ばれる．

M の計算状況は (q, pos) のペアで定義される．ここで，q は状態，pos は入力の位置を表す．もし q が全称状態（それぞれ，存在状態）ならば，そのとき，(q, pos) は全称状況（それぞれ，存在状況）と呼ばれる．計算状況の中で，もし状態 q が最終状態ならば，その計算状況は受理状況と呼ばれる．計算状況 $(q_0, 1)$ は初期状況と呼ばれる．任意の同期状態 $p, q \in S$ に対し，もし $\psi(q) = p$ ならば，p は q の親と呼ばれる．もし $\psi(q) = \perp$ ならば，このような q は親をもたない．親の役割は，同期状態による同期の条件（後で定義）のリセットである．状態遷移関数 $\delta(q, a) = \{q_1, \ldots, q_l\}$ の解釈は，M は入力記号 a を読み込み，その状態を q から $q_1, \ldots, q_l$ へ変化させる．このとき，もし $a \neq \epsilon$ ならば，M は入力ヘッドを1記号右へ進め，もし $a = \epsilon$ ならば，M は入力ヘッドを進めない（これは ϵ-動作と呼ばれる）．

さて以下で，PISAFA が受理する言語について正確に定義する．なお，以下の各定義に現れる PISAFA M を，$M = (Q, S, \Sigma, \delta, q_0, \mu, \psi, F)$ とする．

[定義 3] M を PISAFA，$x = a_1 \cdots a_n$ を長さ n の入力記号列とする．x 上の M の完全な計算木とは次のようなラベル付きの木 T である：

- T の各ノードは M の計算状況でラベル付けされている．
- T の各枝は $\{a_1, \ldots, a_n, \epsilon\}$ の中の記号でラベル付けされている．
- T の根は $(q_0, 1)$ でラベル付けされている．
- v を葉でないノードとする．もし v が (q, pos) でラベル付けされておりかつある記号 $a \in \{\epsilon, a_{pos}\}$ に対して $\delta(q, a) = \{q_1, \ldots, q_k\}$ ならば，v は k 個の子供 $v_1, \ldots, v_k$ をもつ．ここで，各子 v_i は (q_i, pos') でラベル付けされており，すべての i に対し，v から

v_i への枝は記号 a でラベル付けされている．更に，もし $a = a_{pos}$ ならば，$pos' = pos + 1$ であり，もし $a = \epsilon$ ならば，$pos' = pos$ である．

[定義 4] M を PISAFA，x を長さ n の入力記号列，T を入力 x 上の M の完全な計算木とする．また，v_0 を T の根とする．このとき，T の任意のノード v に対し，v_0 から v までのラベルの列から状態が同期状態である計算状況すべてを順に選んで得られる部分列を $\alpha = (p_1, b_1) \cdots (p_u, b_u)$ とする．α が次を満足することは明らかだ：(1) $b_1 \leqq b_2 \leqq \cdots \leqq b_u$，(2) すべての i $(1 \leqq i \leqq u)$ に対し，p_i は同期状態である．

このような部分列 α を v-同期列と呼ぶ．また，単に M の同期状態をもつ計算状況だけを並べた列 $(p_1, b_1) \cdots (p_u, b_u)$ で $b_1 \leqq \cdots \leqq b_u$ を満足するものを M の同期列と呼ぶ．したがって，v-同期列は M の同期列である．

次に同期ペアを定義する．

[定義 5] M を PISAFA，p を M の任意の同期状態とする．$\alpha = (p, b_1) \cdots (p, b_u)$ 及び $\beta = (p, c_1) \cdots (p, c_t)$ を同期状態が p である長さ 0 以上の任意の M の同期列とする．そのとき，ペア (α, β) が同期ペアであるとは，α が β の接頭語になっているかまたはその逆が成り立つときである．なお，長さ 0 の列は空記号列を意味する．

同期ペアの例を挙げる．例えば，二つの M の同期列 $\alpha = (p_1, 2)$ と $\beta = (p_1, 2)(p_1, 9)$ は，α は β の接頭語になっているから同期ペアである．

[定義 6] M を PISAFA，$\alpha = (p_1, b_1) \cdots (p_u, b_u)$ を長さ 0 以上の任意の M の同期列とする．そのとき，任意の $q \in S \cup \{\perp\}$ に対し，α の q-部分列とは，α に $\psi(p) = q$ なる同期状態 p（複数あるかもしれない）をもつ計算状況が $e-1$ 回出現するとき，α をこのような計算状況で分割することによって得られる部分列 $\alpha_1, \ldots, \alpha_e$ のことである．すなわち，α_j $(1 \leqq j \leqq e)$ は，$j-1$ 番目に p が出現する計算状況から j 番目に p が出現する計算状況までのすべての計算状況をとって得られる α の部分列で，p が出現する計算状況を除いたものである．なお，0 番目は (p_1, b_1)，e 番目は (p_u, b_u) と定義する．したがって，もしこのような p が存在しなければ，$\alpha = \alpha_1$ である．

q-部分列の例を挙げる．$\alpha = (p_1, 2)(p_1, 4)(p_2, 6)(p_3, 8)(p_4, 9)(p_1, 10)(p_2, 12)$ であるとする．もし $\psi(p_2) = \perp$，$\psi(p_4) = \perp$ ならば，$\perp$-部分列は $(p_1, 2)(p_1, 4), (p_3, 8), (p_1, 10), \epsilon$ となる．

次に部分入力同期条件を定義する．これは PISAFA の受理計算が満足する重要な条件で，再帰的に定義される．このとき，q-部分列は，同期をとる範囲を制限するために使われる．

[定義 7] M を PISAFA とする．$\alpha = (p_1, b_1) \cdots (p_u, b_u)$ 及び $\beta = (q_1, c_1) \cdots (q_t, c_t)$ を任意の M の同期列とする．そのとき，ペア (α, β) が部分入力同期条件を満足するとは，次の再帰的な手続き *SyncTest* に対し，$SyncTest(\alpha, \beta, \perp)$ が TRUE を返すときである．

$SyncTest(\bar{\alpha}, \bar{\beta}, q)$

（1） もし $\psi(p) = q$ なる同期状態 p が存在しなければ *TRUE* を返し，終了する．

（2） $\psi(p) = q$ なるすべての同期状態 p に対して，以下の (a)，(b) を行う．

（a） $\bar{\alpha}$ 及び $\bar{\beta}$ から p をもつ計算状況をすべて取り出して順に並べたものをそれぞれ $\bar{\alpha}^p$，$\bar{\beta}^p$ とする．

（b） もし $(\bar{\alpha}^p, \bar{\beta}^p)$ が同期ペアならば次の同期状態のチェックへ行く．さもなければ *FALSE* を返し，終了する．

（3） $\bar{\alpha}$ 及び $\bar{\beta}$ の q-部分列をそれぞれ $\alpha_1, \ldots, \alpha_e$ と $\beta_1, \ldots, \beta_l$ とする．ここで，一般性を失うことなく $e \leqq l$ とする．そのとき，すべての $1 \leqq j \leqq e$ と $\psi(p) = q$ なるすべての p に対し，$SyncTest(\alpha_j, \beta_j, p)$ が *TRUE* を返すならば *TRUE* を返し，終了する．

さて以下で，計算木の概念を導入し，それから言語の受理にかかわる受理計算木を定義する．

[定義 8] M を PISAFA，x を長さ n の入力とする．x 上の M の計算木 T' は完全な計算木 T の部分木で次の三つの条件を満足するものである：

（1） もし T' のノード v が全称状況でラベル付けされているならば，v は T と同じ子をもつ．

（2） もし T' のノード v が存在状況でラベル付けされているならば，v は T の子のうちたかだか一つの子をもつ．

（3） v_1 と v_2 を T' の任意のノードとする．そのとき，v_1-同期列と v_2-同期列は部分入力同期条件を満足する．

3 番目の条件は，任意の同期状態 q_s に対し，並列に動いているプロセスは同期状態 q_s の親に出会うまで，それらは q_s で常に同じ位置の入力記号を読むことを保証している．

[定義 9] 定義 8 の計算木の中で，有限の計算木で，

根が初期状況でラベル付けされかつ葉がすべて入力位置 $n+1$ をもつ受理状況，すなわち $q \in F$ なる $(q, n+1)$，でラベル付けされたものを受理計算木と呼ぶ．

PISAFA M が入力 x を受理する必要十分条件は x 上の M の受理計算木が存在することである．我々は M によって受理される言語を $L(M)$ と書く．

簡単な PISAFA の例を与えよう．今，アルファベット $\{0,1\}$ 上の記号列を考える．R_2^1 を 0 の個数が 2 で割って 1 余る記号列の集合，R_3^2 を 0 の個数が 3 で割って 2 余る記号列の集合とする．そのとき，図 1 は $L = (R_2^1 \cap R_3^2)\{0\}^*\{1\}$ を受理する PISAFA である．ここで，q_0 だけが全称状態で，他はすべて存在状態である．また，同期状態は q_s だけで，$\psi(q_s) = \perp$ である．この PISAFA が L を受理することは簡単に検証できる．なぜなら，同期状態 q_s によって，初期状態から q_s に到達する記号列は必ず $R_2^1 \cap R_3^2$ に入っていなければならないからである．この例などは，通常の交代有限オートマトンで簡潔に記述することは難しい．これは，全称状態で分岐した後に同じ記号列を受理させる入力同期のような機構がないためである．例えばこの例では，もし q_s が同期状態でなければ，L にない 001 が受理されてしまう．

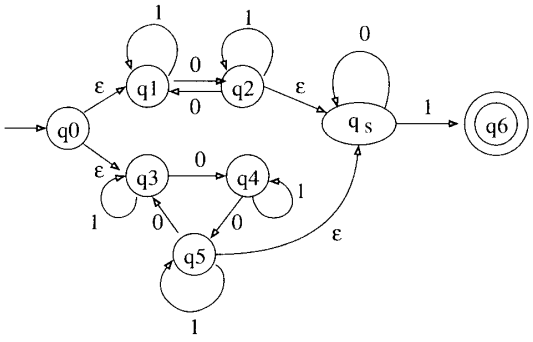

図 1　PISAFA の例
Fig. 1　An example of PISAFAs.

4. 準拡張正規表現から PISAFA への変換

我々は，長さ m の SERE はたかだか $2m$ 状態の PISAFA へ変換できることを示す．まず変換法を示し，それから変換例を与える．

4.1 結　　果

[定理 1]　r をアルファベット Σ 上の長さ m の SERE とする．そのとき，r を $L(r) = L(M)$ でかつたかだか $2m$ 状態の PISAFA M に $O(m)$ 時間で変換することができる．

(証明)　求める PISAFA を M としよう．そのとき，r を M に変換するアルゴリズムは正規表現を NFA に変換する Thompson の方法を拡張することによって作られる．その方法については文献 [4] を参照してほしい．違いは，SERE の共通集合演算の処理である．これに対しては，もとの表現の言語を変えないように同期を設定してやる必要がある．

M への変換は，図 2 に示すように r に出現する演算子に基づいて再帰的に行われる．図 2 は変換の概略を示しており，(a)，(b)，(c) 及び (d) はそれぞれ，$r = r_1 \vee r_2$（和集合），$r = r_1 r_2$（連接），$r = r_1^*$（閉包），$r = r_1 \wedge r_2$（共通集合）に対する変換を表している．M_1 と M_2 はそれぞれ r_1 と r_2 に対応する PISAFA を表している．まず変換法であるが，基礎と

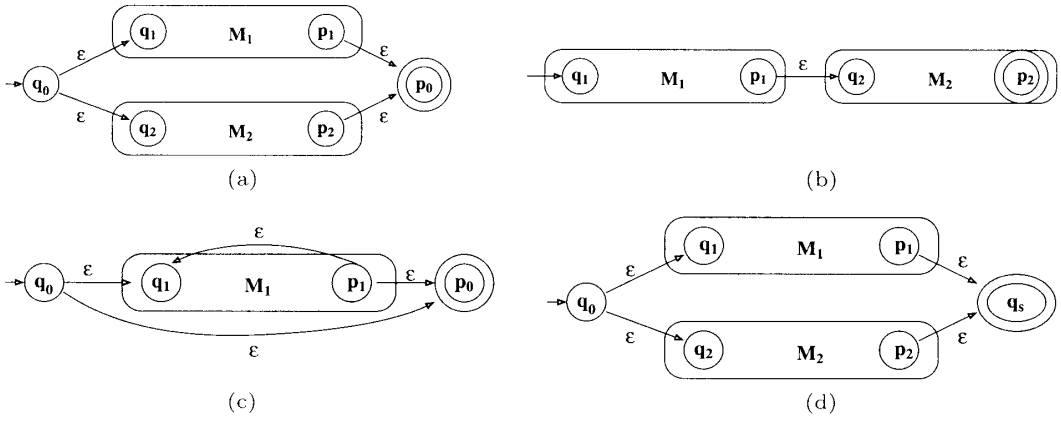

図 2　変換法．(a) 和集合 (b) 連接 (c) 閉包 (d) 共通集合
Fig. 2　Translation. (a) For union. (b) For concatenation. (c) For closure. (d) For intersection.

なる，$\emptyset$，ϵ，$a \in \Sigma$ については正規表現から NFA への変換と同じであるから省略する．次に，各演算について見てみる．図 2 で示されているように，和集合，連接，閉包についてもほとんど同じである．新しく追加される状態はすべて存在状態であり，同期状態，親関数はもとの PISAFA のものがそのまま継承される．したがって，詳細は略す．共通集合演算に関してのみ詳しく述べる．

今，$M_1 = (Q_1, S_1, \Sigma, \delta_1, q_1, \mu_1, \psi_1, \{p_1\})$ と $M_2 = (Q_2, S_2, \Sigma, \delta_2, q_2, \mu_2, \psi_2, \{p_2\})$ をそれぞれ r_1 と r_2 に対する PISAFA とする．求める PISAFA を $M = (Q, S, \Sigma, \delta, q_0, \mu, \psi, \{q_s\})$ とする．まず，$Q = Q_1 \cup Q_2 \cup \{q_0, q_s\}$ と設定する．ここで，q_0, q_s は新しい状態で，それぞれ M の初期状態と最終状態になる．なお，どの PISAFA も最終状態が一つしかないが，これは再帰的な変換のどの段階においても最終状態が一つしか出ないことより明らかである．同期状態の集合を $S = S_1 \cup S_2 \cup \{q_s\}$ と定義する．関数 δ，μ は次のように定義する．M_1 の状態に対しては δ_1，μ_1 を，M_2 の状態に対しては δ_2, μ_2 を適用する．また，$\delta(q_0, \epsilon) = \{q_1, q_2\}$，$\delta(p_1, \epsilon) = \{q_s\}$，$\delta(p_2, \epsilon) = \{q_s\}$，$\mu(q_0) = \wedge$（すなわち，全称状態），$\mu(q_s) = \vee$（存在状態）と定義する．親関数 ψ は次のように定義する．まず，$\psi(q_s) = \perp$ と定義する．任意の同期状態 $q \in S_1$ に対し，もし $\psi_1(q) = \perp$ ならば，$\psi(q) = q_s$; さもなければ $\psi(q) = \psi_1(q)$ と定義する．同様に，任意の同期状態 $q \in S_2$ に対し，もし $\psi_2(q) = \perp$ ならば，$\psi(q) = q_s$; さもなければ $\psi(q) = \psi_2(q)$ と定義する．この変換法により，全称状態と同期状態は 1 対 1 に対応することに注意する．

次に，上記の変換法の正しさ，すなわち，M によって受理される言語は $L(r)$ に等しいことを r に出現する演算子の個数 k 上の帰納法で証明しよう．$k = 0$ ならば，r は正規表現であるから明らかである．さて，$k \geq 0$ なるある k まで成り立っているとき，$k+1$ について成り立つことを証明しよう．我々は SERE の再帰的な構成に従ってこのことを証明する．今，r_1 及び r_2 を演算子の数がたかだか k 個からなる SERE とする．帰納法の仮定によって，これらはそれぞれ，$L(r_1)$，$L(r_2)$ を受理する PISAFA M_1，M_2 へ変換できる．図 2 に従って，演算の種類ごとに場合分けして証明する．変換法により，M_1 と M_2 は共通の同期状態をもたないことに注意する．

(a) $r = r_1 \vee r_2$ の場合：もし $x \in L(r)$ ならば，$x \in L(r_1)$ または $x \in L(r_2)$ である．これは x が M_1 または M_2 の少なくともどちらか一方によって受理されることを意味する．このとき，x を受理する方の M_i ($i = 1$ または 2) の受理計算木から簡単に M の受理計算木を構成できる．したがって，x は M によって受理される．逆も明らかである．

(b) $r = r_1 r_2$ の場合：まず最初に，もし $x \in L(r)$ ならば，x 上の M の受理計算木が存在することを示す．$x \in L(r)$ より，$x = x_1 x_2$ でかつ $x_1 \in L(r_1)$ 及び $x_2 \in L(r_2)$ となるように x を分割できる．仮定により，x_1 上の M_1 の受理計算木が存在するから，それを T_1 とする．また，x_2 上の M_2 の受理計算木も存在するから，それを T_2 とする．このとき，T を T_1 のすべての葉の下に T_2 をくっつけた計算木とすれば，この T は明らかに x 上の M の受理計算木となる．したがって，x は M によって受理される．

逆に，x が M によって受理されるならば，$x \in L(r)$ を証明する．x は M によって受理されるから，x 上の M の受理計算木 T が存在する．M は M_1 と M_2 の直列接続になっているから，x を受理するためには，必ず M_1 の最終状態を通って M_2 の初期状態へと行く．したがって，T は図 3 のように，M の初期状態を根とし，M_1 の最終状態を葉とする一つの木 T_1 と，M_2 の初期状態を根とし，M の最終状態を葉とする h (≥ 1) 本の木 $T_2^1, \ldots, T_2^h$ に分割することができる．今，T_1 上の根から葉への各パスについて考える．以下，単にパスといった場合は対象となっている木の根から葉へのパスを表すものとする．このとき，M_1 と M_2 は直列接続であるから，T_1 上のパスと $T_2^1, \ldots, T_2^h$ は 1 対 1 に対応している．さて，ここで問題なのは，T_1 上の各パスは異なる記号列に対応しているかも知れないということである．この場合，T_1 は，x の任意の接頭語に対して，M_1 の受理計算木

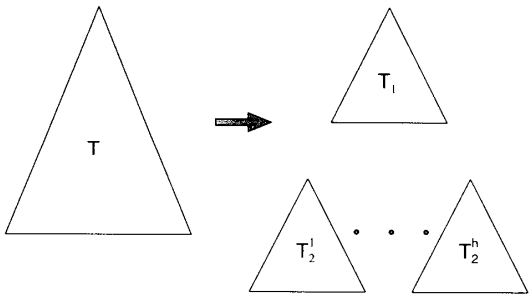

図 3　T の $T_1, T_2^1, \ldots, T_2^h$ への分割
Fig. 3　Decomposition of T into $T_1, T_2^1, \ldots, T_2^h$.

にならない．このため，このままでは，$x = x_1x_2$ で $x_1 \in L(r_1)$ かつ $x_2 \in L(r_2)$ となるような x_1 と x_2 に x を分割できない．我々は以下で，T_1 から x のある接頭語 x_1 上の M_1 の受理計算木を構成できること，及び $T_2^1, \ldots, T_2^h$ のうち少なくとも一つは $x = x_1x_2$ なる x_2 上の M_2 の受理計算木であることを示す．これによって，$x \in L(r)$ が成り立つ．

もし T_1 が1本のパスだけからなる木ならば，単純に x を T_1 に対応する部分と T_2^1 に対応する部分に分割すればよい．したがって，$x \in L(r)$ は明らかである．

次に，T_1 に2本以上のパスが存在する場合について考える．さて，T_1 から新しい計算木を生成する以下の再帰的な手続き *TreeConv* を考える．*TreeConv* は，キーとなる同期状態を選択しながら，T_1 上を根から葉へ向かってすべてのパスが同じ記号列に対応するように T_1 を変形していく．このとき，T_1' は変換途中の木を保存し，T_t は T_1' 内の変換すべき箇所を示す．我々は，*TreeConv*(T_1, T_1) 実行する．

TreeConv(T_1', T_t)

（1） T_t 上に出現する最初の全称状態に対応する同期状態を q_{s_t} とする．すなわち，q_{s_t} は，$\psi(q) = \perp$ なる同期状態の中で，T_t 上の各パス上で最初に出現する同期状態である．

（2） T_t 上の各パスで q_{s_t} が最初に現れる計算状況をそれぞれ $(q_{s_t}, b_1), \ldots, (q_{s_t}, b_k)$ とする．このとき，部分入力同期条件より，$b_1 = \cdots = b_k$ となる．

（3） T_t^j $(1 \leqq j \leqq k)$ を (q_{s_t}, b_j) を根とする T_t の部分木とする．そのとき，すべての j $(2 \leqq j \leqq k)$ に対し，T_1' 内の T_t^j を T_t^1 で置き換える（図4を参照）．

（4） もし T_t^1 が1本のパスだけからなる木ならば手続きを終了する．さもなければ，置き換えたすべての T_t^1 に対して，*TreeConv*(T_1', T_t^1) を行う．

TreeConv により最終的に得られる木 T_1' は，T_1' 上のすべてのパスが同じ記号列に対応する．更に，*TreeConv* は，常に最初に出会う $\psi(q) = \perp$ なる同期状態 q を置換えの起点としているため，$\psi(q) = \perp$ なる任意の同期状態 q に対し，T_1' 上のどのパスにおいても q の出現回数は等しくなる．したがって，T_1 が計算木であることを考えれば，以下の性質が成り立つ．

[補題1] *TreeConv* によって得られる T_1' は次の性質を満足する．

（1） T_1' は，x のある接頭語 x_1 上の M_1 の受理計算木である．

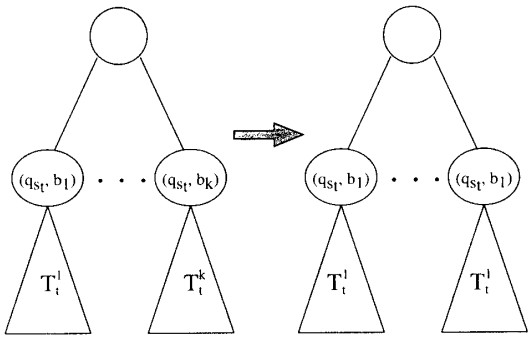

図4 木の変換．q_{s_t} は最初の全称状態に対応する同期状態

Fig. 4 Tree transformation. q_s is a synchronizing state corresponding to the first universal state occurring in T_t.

（2） $\psi(q) = \perp$ なる任意の同期状態 q に対し，T_1' 上のどのパスにおいても q の出現回数は等しい．

次に，x_1 を補題1の x_1 としたとき，$x = x_1x_2$ なる x_2 に対し，x_2 上の M_2 の受理計算木が存在することを示す．*TreeConv* では，T_1 のパスの中で少なくとも1本は必ず T_1' の中に同じものが残る．しかもこれは x_1 に対応するパスである．したがって，そのパスに対応する計算木を T_2^j とすると，明らかにこの木は，$x = x_1x_2$ なる x_2 上の M_2 の受理計算木になっている．このように，x が M によって受理されれば，$x_1 \in L(r_1)$ かつ $x_2 \in L(r_2)$ なる x_1 と x_2 に分割できる．以上より，$x \in L(r)$ となる．

(c) $r = r_1^*$ の場合：まずはじめに，任意の k $(\geqq 0)$ に対し，M の受理言語は $L(r_1^k)$ を含むことを k 上の帰納法で証明する．ここで，r_1^k は r_1 の k 個の連接を表す．また，$r_1^0 = \epsilon$ と定義する．

$k = 0$ のとき．$r_1^0 = \epsilon$ であるから，結果は明らかである．

$k = 1$ のとき．この場合 r_1 であるから，PISAFA への変換の帰納法の仮定より，M_1 は $L(r_1)$ を受理する．したがって，M も $L(r_1)$ を受理する．

$k \geqq 1$ なるある k まで成り立つとする．そのとき，r_1^{k+1} について証明する．もし $x \in L(r_1^{k+1})$ ならば，$r_1^{k+1} = r_1r_1^k$ であるから，x は $x_1 \in L(r_1)$ 及び $x_2 \in L(r_1^k)$ となる x_1 と x_2 に分割できる．帰納法の仮定により，x_1 上の M の受理計算木 T_1 と x_2 上の M の受理計算木 T_2 が存在する．連接の場合はこれらの木は共通の状態をもたないが，今回は両方とも M の計算木であるため，共通の状態が存在する可能性が

ある．したがって，連接のように単純に T_1 に T_2 を付けることができない．しかしこのことは，連接で用いた *TreeConv* によって解決される．今，T_1' を T_1 から *TreeConv* で得られた木とすると，これは T_1 と同じ x_1 上の M_1 の受理計算木になっている．我々は，T_1' の下に T_2 を付けた木 T を考える（実際には，T_1' の最後で最終状態へ行く代わりに M_1 の初期状態へ行くようにし，T_2 は M_1 の初期状態からスタートするようにする）．次の考察により，T_1' と T_2 では別々に部分入力同期条件がチェックされるから，T は部分入力同期条件を満足し，x 上の M の受理計算木となる．

（1） $\psi(q) = \perp$ なる同期状態 q の場合．補題 1 の（2）から，$\psi(q) = \perp$ なる任意の同期状態 q に対し，T_1' と T_2 間で同期の条件がチェックされることはない．

（2） $\psi(q) \neq \perp$ なる T_1' 内の同期状態 q の場合．部分入力同期条件の判定における $\perp$-部分列を考えたとき，補題 1 の（2）より，$\psi(p) = \perp$ なる同期状態の出現が等しいから，必ず T_1' 内だけで条件チェックが行われる．したがって，このような同期状態 q は，T_1' 内と T_2 内で同期する必要がない．

以上より，M は $L(r_1^{k+1})$ を受理する．よって，$x \in L(r_1^*)$ ならば M は x を受理する．

逆方向について証明する．すなわち，記号列 x が M によって受理されるならば $x \in L(r)$ を示す．今，T を x 上の M の受理計算木とする．T の各パス上で，M_1 の初期状態が出現する回数の最大値を k としたとき，M は M_1 を k 回通って x を受理するという．そのとき，M が x を M_1 を k 回通って受理するならば，$x \in \cup_{0 \leqq j \leqq k} L(r_1^j)$ であることを k $(\geqq 0)$ 上の帰納法で証明する．

$k = 0$ のとき．x は空記号列なので明らか．

$k = 1$ のとき．M_1 が受理する言語は $L(r_1)$ なので明らか．

$k \geqq 1$ なるある k まで成り立つとする．そのとき，M_1 を $k+1$ 回通る場合について証明する．x 上の M の受理計算木を T とすると，連接の場合と同じように，T を最初の 1 回目に対応する部分 T_1 と残りのたかだか k 回に対応する部分 $T_2^1, \ldots, T_2^h$ に分割することができる．

今，T_1 に *TreeConv* を施して T_1' を得る．T_1' の最後で，M_1 の最終状態から M の最終状態へ行く遷移を加えれば，明らかに，これは x のある接頭語 x_1 上の M の受理計算木で M_1 を 1 回通るものになっている．したがって，$x_1 \in \cup_{0 \leqq j \leqq 1} L(r_1^j)$ となる．更に，$x = x_1 x_2$ なる x_2 に対する受理計算木が $T_2^1, \ldots, T_2^h$ の中に少なくとも一つ存在する．この受理計算木のはじめに，M の初期状態から M_1 の初期状態へ行く遷移を加えれば，これは M_1 をたかだか k 回通る x_2 上の M の受理計算木になっている．したがって，$x_2 \in \cup_{0 \leqq j \leqq k} L(r_1^j)$ である．以上より，$x \in \cup_{0 \leqq j \leqq k+1} L(r_1^j)$ となる．

(d) $r = r_1 \wedge r_2$ の場合：まず，もし $x \in L(r)$ ならば，x は M によって受理されることを示す．$x \in L(r)$ であるから，$x \in L(r_1)$ かつ $x \in L(r_2)$ であり，仮定より，x は M_1 と M_2 の両方によって受理される．さて今，x が与えられたときの M の計算で次のようなものを考える．M は初期状態 q_0（これは全称状態）で分岐し，それぞれ x 上の M_1 の受理計算及び M_2 の受理計算を実行する．M_1 及び M_2 が x を読み終わって最終状態へ行ったならば，M は最終状態 q_s（これは同期状態でもある）へ行く．この計算が受理計算木を構成することを示す．この木を T とする．そのとき，T の任意のノード v_1 及び v_2 に対し，v_1-同期列と v_2-同期列が部分入力同期条件を満足することを示せばよい．まず，$\psi(q) = \perp$ なる同期状態は q_s だけである．これは x を読み終わった最後に出てくるだけであるから明らかに部分入力同期条件を満足する．

次に，*SyncTest* は，二つの同期列を $\perp$-部分列に分割してチェックする．このとき，$\perp$-部分列は，各同期列を q_s で分割することによって得られる．したがって，v_1-同期列の $\perp$-部分列は，M_1 の同期列かまたは M_2 の同期列であり，M_1 または M_2 の受理計算木上の同期列となっている．v_2-同期列も同様である．変換法により，$\psi(p) = q_s$ なる同期状態 p は，M_1 または M_2 において，$\psi(p) = \perp$ であった．したがって，もし両方とも M_1 または M_2 の同期列ならば，受理計算木の同期列であるから，*SyncTest* は *TRUE* を返す．もし一方が M_1 でもう一方が M_2 の同期列ならば，M_1 と M_2 は共通の同期状態をもたないから，同期ペアのチェックでは常に少なくとも一方は空記号列になる．よって，この場合も *SyncTest* をパスする．以上より，T は x 上の M の受理計算木となる．

逆の場合，x が M によって受理されれば，$x \in L(r)$ であることも同じように証明できる．すなわち，x 上の M の受理計算木 T は M_1 と M_2 の受理計算木から作られていることを同様に証明できる．

以上で，各演算によって作られる新たな SERE に対しても，PISAFA が正しく構築されることが証明された．再帰的な変換法の各ステップでたかだか 2 状態し

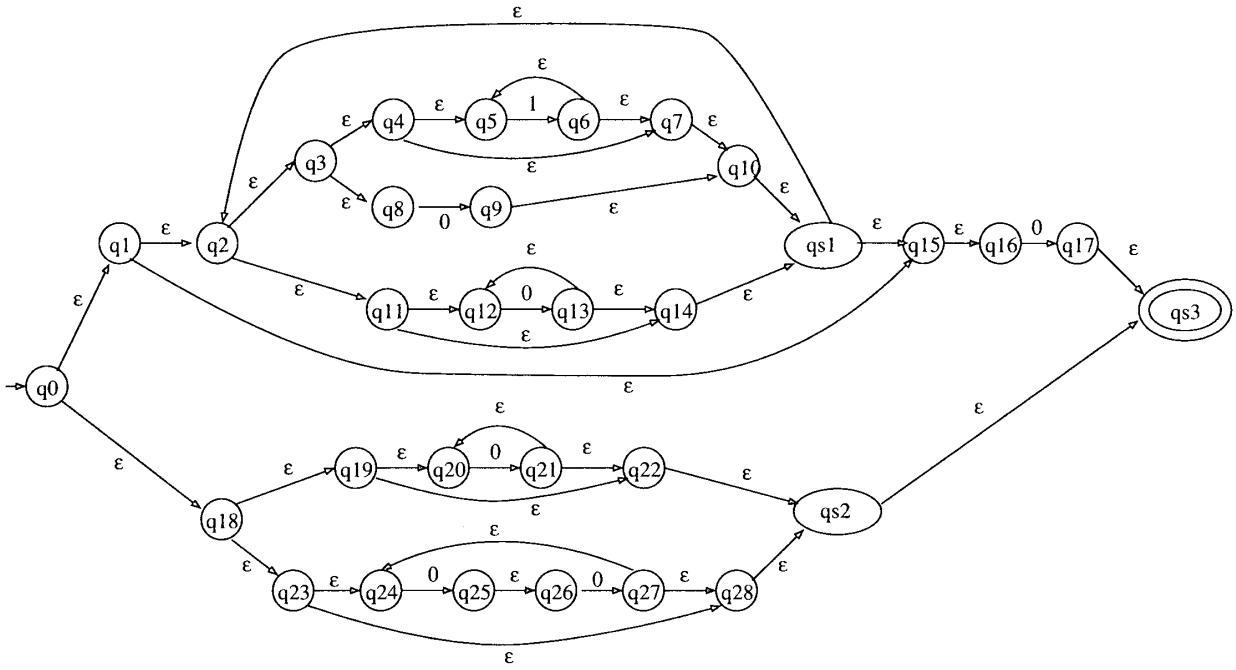

図 5 $r = ((0^* \wedge (0 \vee 1^*))^*0) \wedge (0^* \wedge (00)^*)$ に対する PISAFA
Fig. 5 PISAFA for $r = ((0^* \wedge (0 \vee 1^*))^*0) \wedge (0^* \wedge (00)^*)$.

か増えないので，SERE の長さが m ならば，生成される PISAFA の状態数はたかだか $2m$ で，変換時間も $O(m)$ である． □

4.2 変 換 例

ここでは，上記で与えた変換に対する例を与える．SERE $r = ((0^* \wedge (0 \vee 1^*))^*0) \wedge (0^* \wedge (00)^*)$ を考える．図 5 は r に対する PISAFA を示している．ここで，状態 q_0，q_2 及び q_{18} が全称状態で，他はすべて存在状態である．また，状態 q_{s_1}, q_{s_2} 及び q_{s_3} は同期状態で，$\psi(q_{s_1}) = \psi(q_{s_2}) = q_{s_3}$, $\psi(q_{s_3}) = \perp$ と定義される．

5. む す び

本論文では，SERE に対する新しいオートマトンモデルを提案し，線形変換できることを示した．しかし，拡張正規表現へと拡張できるかはわからない．今回は，Thompson の方法を利用したが，Thompson の方法より少ない状態数の NFA を生成することが知られている別の方法（例えば，文献 [1], [3], [8] を参照）を適用できるかは今後の課題である．また，提案したモデルの計算能力を明らかにすることも今後の課題である．

謝辞 有益な御助言を頂きました査読者の方々に深謝致します．

文 献

[1] A.V. Aho, "Algorithms for finding patterns in strings," in Handbook of theoretical computer science, ed. J.V. Leeuwen, Elsevier Science Pub., 1990.
[2] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer, "Alternation," J. Assoc. Comput. Mach., vol.28, no.1, pp.114–133, 1981.
[3] C.H. Chang and R. Paige, "From regular expressions to DFA's using compressed NFA's," Theor. Comput. Sci., vol.178, pp.1–36, 1997.
[4] J.E. Hopcroft and J.D. Ullman, Introduction to automata theory language and computation, Addison Wesley, Reading Mass, 1979.
[5] J. Hromkovic, K. Inoue, B. Rovan, A. Slobodova, I. Takanami, and K.W. Wagner, "On the power of one-way synchronized alternating machines with small space," Int. J. Found. Comput. Sci., vol.3, no.1, pp.65–79, 1992.
[6] G. Myers, "A four russians algorithm for regular expression pattern matching," J. Assoc. Comput. Mach., vol.39, no.4, pp.430–448, 1992.
[7] A. Slobodova, "On the power of communication in alternating machines," Proc. 13th MFCS'88, LNCS 324, pp.518–528, 1988.
[8] G. Navarro and M. Raffinot, "Compact DFA representation for fast regular expression search," Proc. WAE2001, LNCS 2141, pp.1–12, 2001.
[9] H. Yamamoto, "On the power of input-synchronized alternating finite automata," Proc. COCOON2000, LNCS 1858, pp.457–466, 2000.

[10] H. Yamamoto, "An automata-based recognition algorithm for semi-extended regular expressions," Proc. MFCS2000, LNCS 1893, pp.699–708, 2000.

（平成 14 年 10 月 15 日受付，15 年 1 月 6 日再受付）

山本　博章　（正員）

昭 55 信州大・工・情報卒．昭 60 東北大大学院博士課程了．同年東北大電気通信研究所助手．現在，信州大・工・助教授．工博．主として，オートマトン，計算量の理論，アルゴリズム理論などの研究に従事．情報処理学会，日本ソフトウェア科学会，EATCS 各会員．