A-033

正規表現を NFA に変換するためのアルゴリズムについて
An Algorithm for Translating Regular Expressions into NFAs

山本 博章 † 宮崎 敬 ‡ 岡本 正行 †
Hiroaki Yamamoto Takashi Miyazaki Masayuki Okamoto

1. まえがき

正規表現は計算機科学の分野において重要な役割を演じており，多くの応用分野で利用されている．一般に，正規表現を利用する場合，有限オートマトンに変換してから処理される．したがって，より小さなサイズの有限オートマトンを生成することは，効率的に正規表現を処理するために重要な要素になってくる．正規表現に対応する有限オートマトンとして，二つの非決定性有限オートマトン（NFA）が広く知られている．一つは Thompson オートマトン（TNFA と略す）呼ばれるものであり，もう一つは Glushkov オートマトン（GNFA と略す）と呼ばれるものである．

本論文では，正規表現を TNFA を使って GNFA に効率的に変換するビット並列アルゴリズムを与える．このようなビット並列化は文献 [4, 5] などで使われている．このアルゴリズムは $O(m^2/W)$ 時間かつ領域で正規表現を GNFA に変換することができる．ここで，W はコンピュータのワード長である．従来の最も良いアルゴリズムは $O(m^2)$ で動作するから（例えば，文献 [1, 2]），我々のアルゴリズムは従来のものよりも W 倍速い．

2. Thompson オートマトンと Glushkov オートマトン

正規表現の説明はここでは省略する．例えば，文献 [3] などのテキストを参照してほしい．

さて，r をアルファベット Σ 上の正規表現としたとき，TNFA と GNFA について説明する．なお以下で，n と m はそれぞれ r の長さおよび r に出現するアルファベット記号の数を表すものとする．

2.1 Thompson オートマトン

TNFA は最も良く知られたオートマトンであり，正規表現の再帰的定義に基づいて構成される．その構成法はここでは省略するが，例えば文献 [3] で見ることができる．この構成法を使うと $O(n)$ 時間と領域で TNFA を構成することができる．TNFA は高々$2n$ 個の状態と高々$4n$ 個の状態遷移を持つ．

M を TNFA とする．M の任意の状態 q に対し，q に入る遷移の数は高々2 である．このとき，遷移数が 2 の状態をジャンクション状態 (junction state) と呼ぶ．また，入る遷移がアルファベット記号による場合，その状態を sym-state と呼び，その記号でラベル付けする．同様に，ϵ 記号の場合，ϵ-state と呼び，ϵ でラベル付けする．また，M の状態を，初期状態から順番にトポロジカルオーダで順序を付けることができる．

さらに，M の逆オートマトン M^R を，すべての遷移を反転させ，かつ初期状態と最終状態を入れ替えることによってできるオートマトンと定義する．

例題 1 正規表現 $r = 1(00 \vee 11)^*1$ を考える．*Fig. 1* は r に対する *TNFA* である．状態 p_3, p_{12}, p_{13} は *junction state* である．このオートマトンの状態は初期状態からトポロジカルオーダにしたがって番号づけされている．ϵ でラベル付けされた状態が *ϵ-state*，0 または 1 でラベル付けされた状態が *sym-state* である．

2.2 Glushkov オートマトン

もう一つの有名なオートマトンが GNFA である．これは ϵ-free NFA であり，正規表現に出現するアルファベット記号が状態に対応している．GNFA はちょうど $m+1$ 個の状態と高々m^2 個の状態遷移を持つ．GNFA を構成する従来のアルゴリズムで最も良いものは $O(m^2)$ 時間および領域で動作する．

例題 2 前と同じ正規表現 $r = 1(00 \vee 11)^*1$ を考える．*Fig. 2* は *GNFA* を示している．各状態は r に出現するアルファベット記号に対応しており，例えば，状態 q_1 は 1 の最初の出現，状態 q_2 は 0 の最初の出現，状態 q_3 は 0 の *2* 番目の出現に対応している．状態 q_1 に入るすべての遷移が 1 によるものであるから，q_1 を 1 でラベル付けしている．他も同様である．

3. ビット並列アルゴリズム

さて，正規表現を TNFA を使って GNFA に変換するビット並列アルゴリズムを示す．Fig. 1 と Fig. 2 を比較すると，TNFA の *sym*-state と GNFA の状態が対応していることに気づく．すなわち，Fig. 1 の状態 $p_0, p_1, p_5, p_7, p_9, p_{11}, p_{15}$ はそれぞれ Fig. 2 の状態 $q_0, q_1, q_2, q_3, q_4, q_5, q_6$ に対応している．従って，*sym*-state 間の到達可能性を計算しかつ ϵ-state を削除することによって，TNFA を GNFA に変換することができる．我々のビット並列アルゴリズムは，TNFA およびその逆オートマトンの性質を利用し，この計算を効率よく行なっている．

アルゴリズム中，演算子 | および & はそれぞれビット単位の論理和および論理積を表す．さらに，*BitSet*(v, i) はビットベクトル v の i 番目のビットを 1 に設定し，*BitCheck*(v, i) はビットベクトル v の i 番目のビットが 1 かどうかチェックする．アルゴリズムは以下で与えられている *REtoGlu* である．まず，TNFA を求め，そこからビット並列化を使って GNFA を求めている．ここで，状態 q に対し，*sym(q)* および *num(q)* はそれぞれ状態 q に割り当てられた記号のラベルおよび順番の番号を表す．

Algorithm REtoGlu(r)
Input: regular expression r.
Output: Glushkov automaton $G = (Q', \Sigma, \delta', p_0, F')$.

†信州大学工学部, Shinshu University
‡長野工業高等専門学校, Nagano Natinal College of Thechnology

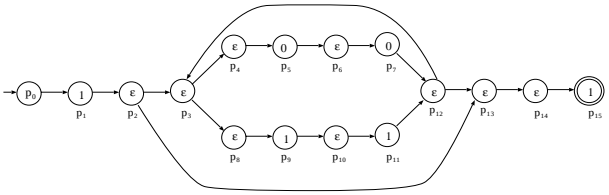

図 1: Thompson automaton for $r = 1(00 \vee 11)^*1$

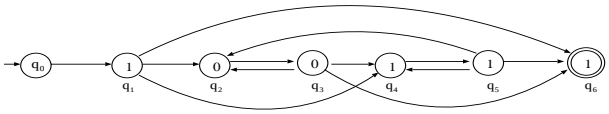

図 2: Glushkov automaton for $r = 1(00 \vee 11)^*1$

Step 1. Translate r into the Thompson automaton M.

Step 2. *ReachState*(M^R,E).

Step 3. Generate G as follows:

1. define Q' to be the set $\{q \mid q$ is a *sym*-state or the initial state of $M\}$,
2. define the initial state p_0 to be the initial state of M,
3. define δ' as follows:
   (a) for all $a \in \Sigma$, $T[a] := 0$,
   (b) for all *sym*-states $q \in Q'$, $BitSet(T[sym(q)], num(q))$,
   (c) for all symbols $a \in \Sigma$ and all states $q \in Q'$, $\delta'(q, a) := E[q]$ & $T[a]$.
4. for all states $q \in Q'$, if $BitCheck(E[q], num(q_f)) = 1$, then add q to F', where q_f is the final state of M.

上のアルゴリズムで使われている手続き *ReachState* を以下で与える．これは状態間の到達可能性を求める働きをする．

Procedure ReachState(N,E)

Step 1 For all states q of N, $E[q] := 0$.

Step 2 Repeat the following twice:
for all states q of N do the following in a topological order:

1. if q is the initial state, then $BitSet(E[q], num(q))$,
2. if q is a state such that there is a transition from a *sym*-state p_1 to q, then $BitSet(E[q], num(p_1))$,
3. if q is a junction state, then $E[q] := E[p_1] \mid E[p_2]$, where p_1 and p_2 are two predecessors of q,
4. otherwise $E[q] := E[p_1]$, where p_1 is a predecessor of q.

最終的次の定理を得る．

定理 1 アルゴリズム REtoGlu は正規表現 r を *GNFA* に $O(m^2/W)$ 時間および領域で変換する．

参考文献

[1] A. Brüggemann-Klein, Regular expressions into finite automata, Theoret. Comput. Sci., 120, 197-213, 1993.

[2] C.H. Chang and R. Paige, From regular expressions to DFA's using compressed NFA's, Theoret. Comput. Sci., 178, 1-36, 1997.

[3] J.E. Hopcroft and J.D. Ullman, Introduction to automata theory language and computation, Addison Wesley, Reading Mass, 1979.

[4] S. Wu, U. Manber and E. Myers, A Sub-Quadratic Algorithm for Approximate Regular Expression Matching, J. of Algorithm, 19, 346-360, 1995.

[5] H. Yamamoto and T. Miyazaki, *A Fast Bit-Parallel Algorithm for Matching Extended Regular Expressions*, Proc. of COCOON2003, LNCS 2697, 222-231, 2003.