# ストリングパターンマッチングマシンにおける検索キー追加方法

6G−8

津田　和彦†　青江　順一††

†住友金属工業（株）　††徳島大学

## 1．まえがき

ストリングパターンマッチングマシンは，文献検索，テキスト編集など多くの分野に応用されている。その中でもAhoとCorasickのマシン（以後マシンACと呼ぶ）は，複数のキーワードをテキストストリングの一回の走査で発見する優れたアルゴリズムである．しかし，検索キーの追加方法が提案されていないため，検索キーの追加を実行するたびに、マシン自体を再構築する必要があった．そこで、本論文では，マシンACのキーワードの追加方法を提案する．

## 2．キーワードの追加

マシンACとは，キーワードより作成した状態遷移図を示すgoto関数，状態遷移図に遷移が定義されていない場合の遷移先を示すfail関数，および，状態に到達したことによってキーワードの検出ができたことを示すoutput関数の3つの関数によって構成される。

ここで，キーワード$y=a_1a_2\cdots a_n$($a_i$は記号)を追加する場合を考える．yのdep文字目まで状態遷移図に定義されている場合（すなわち，$g(0,a_1a_2\cdots a_{dep})=s_{dep}$）を仮定する．$s_{dep}$より$a_{dep+1}$から$a_n$までの文字列によって遷移する状態列$s_{dep+1},s_{dep+2},\cdots,s_n$の追加方法，および，これらの状態のfail，output関数の追加方法はAhoらの作成方法と同様なので省略する．よって，本論文では状態$s_{dep+1},s_{dep+2},\cdots,s_n$が追加されたことにより必要となる既存状態のfail，output関数の更新方法について論じる。

fail関数を更新するべき状態は，fail関数による遷移で状態$s_{dep}$に遷移できる状態から更にgoto関数によって，$a_{dep+1}a_{dep+2}\cdots a_n$のうち少なくとも1つ以上遷移した状態である．ここで次の2点を注意しなければならない．

1) fail関数で状態$s_{dep}$に遷移できる状態は複数存在する．

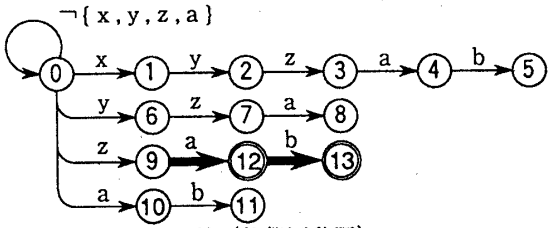
2) 複数回のfail関数で状態$s_{dep}$に到達する状態も存在する．

ここで，$p_0\in f^{-1}(s_{dep})$なる$p_0$を基点とするgotoグラフに$p_1=g(p_0,a_{dep+1})$，$p_2=g(p_1,a_{dep+2})$，$\cdots$，$p_i=g(p_{i-1},a_{dep+i})$のi個の遷移が存在し；$q_0\in f^{-1}(p_0)$なる$q_0$に対して，$q_1=g(q_0,a_{dep+1}),q_2=g(q_1,a_{dep+2}),\cdots,q_j=g(q_{j-1},a_j)$のj個の遷移が存在する仮定しよう．

このとき，$p_1,p_2,\cdots,p_i$のfail関数の値は，各$s_{dep+1},s_{dep+2},\cdots,s_{dep+i}$に変更する．次に$q_0$を基点とするgotoグラフは，i番目までの状態$q_1,q_2,\cdots,q_i$のfail関数値は変更せず$q_{i+1},q_{i+2},\cdots,q_j$のfail関数値を$s_{dep+i+1},s_{dep+i+2},\cdots,s_{dep+j}$と変更する．つまり，$q_0$を基点とするgotoグラフでは，$p_0$を基点とするgotoグラフで$p_i$まで遷移できた場合，基点を$q_i(\in f^{-1}(p_i))$としたgotoグラフのfail関数値を変更すればよい．以下，fail関数によって$q_0$に遷移してくる状態を基点としたgotoグラフについても同様の処理を行うことでfail関数の値を更新する．

次にoutput関数の更新は，fail関数の値の更新手順でgoto関数により$a_n$まで遷移した状態と複数回のfail関数による遷移で$s_n$に達する状態のoutput関数値とyの和集合を取ることで更新する．

以上の処理を図1に示す例を用いて説明する．図1はキーワード{xyzab,yzab,z,ab}より構成されたマシンACにキーワード{zab}を追加した例である．追加されたgotoグラフの状態12，13の基点である状態9にfail関数で遷移してくる状態を検索する．この例の場合は状態7のみである．次にg(7,a)＝8と定義されているのでf(8)＝12と更新する．次にg(12,b)＝failなのでfail関数で状態8へ遷移する状態4へ移り，g(4,b)＝5なのでfail(5)＝13と更新する．ここで，追加したキーワードの最終文字まで遷移したので，output(5)にキーワードzabを追加する．次にfail関数により状態5へ遷移してくる状態の検出を行うが，この例の場合は存在しないので，更新処理を終了する．




(a) Goto関数（状態遷移図）

| s | f(s) |
|---|---|
| 1 | 0 |
| 2 | 6 |
| 3 | 7 |
| 4 | 8 |
| 5 | 11 |
| 6 | 0 |
| 7 | 9 |
| 8 | 10 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |

(b) 追加前のFail関数

| s | f(s) |
|---|---|
| 1 | 0 |
| 2 | 6 |
| 3 | 7 |
| 4 | 8 |
| 5 | 13 |
| 6 | 0 |
| 7 | 9 |
| 8 | 12 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |
| 12 | 10 |
| 13 | 11 |

(c) 追加後のFail関数

| s | output(s) |
|---|---|
| 3 | {z} |
| 4 | {yza} |
| 5 | {xyzab,ab} |
| 7 | {z} |
| 8 | {yza} |
| 9 | {z} |
| 11 | {ab} |

(d) 追加前のOutput 関数

| s | output(s) |
|---|---|
| 3 | {z} |
| 4 | {yza} |
| 5 | {xyzab,ab,zab} |
| 7 | {z} |
| 8 | {yza} |
| 9 | {z} |
| 11 | {ab} |
| 13 | {zab,ab} |

(e) 追加後のOutput関数

図1　キーワード追加前後のマシンAC

---

An Insertion Method of a Keyword for String Pattern Matching Machines.

Kazuhiko TSUDA† Jun-ichi AOE††

†Sumitomo Metal Industries, LTD. ††The University of Tokushima.

図2にfail，output関数の更新アルゴリズムを示す．第1番目のfor文では，fail関数により$s_{dep}$へ遷移してくる全ての状態sと，その状態から次に遷移しなければならない文字が追加キーワードyの何文字目かを示す番号dep+1をセットでqueueへ格納する．第1番目のwhile文では，既存の状態のfail，output関数の変更作業を行う．このwhile文では，queueに格納された状態を1つづつ取り出し，while文でこの状態よりgoto関数によって定義されている$a_i$以降の文字による状態遷移を求め，この状態のfail関数値を更新する．次のfor文では，while文によって到達した状態にfail関数によって遷移してくる状態をqueueに追加し，再帰的処理を実現することで複数のfalire関数による状態遷移に対する問題を解決している．次のif文では，while文による処理で追加キーワードの最終文字まで到達した状態を検出し，この状態のoutput関数にキーワードyを追加する．

```
Algorithm: Updateing algorithm of failure
and output functions.
Input:Machine AC,dep and s_dep.
Output:Modified machine AC.
Method
  begin
    queue←empty;
    for each s such that f(s)=s_dep do
      queue←queue∪{(s,dep+1)};
    while queue ≠ empty do
      begin
        Let r be the next state in queue;
        Let i be the next number in queue;
        queue←queue−{(r,i)};
        while s←g(r,a_i)≠fail do
          begin
            f(s)←s_i;
            r←s;
            i←i+1;
          end
        for each s such that f(s)=r do
          queue←queue∪{(s,i)};
        if i>n then
          output(s)←output(s)∪{y};
      end
  end
```

図2　failure、output関数の更新アルゴリズム

## 3．評価

表1にマシンAC全体の再構築と本更新アルゴリズムに基づく再構築を行った場合の構築に要する計算コストを示す．実験方法は，追加処理の場合，一つのキーワードyを除いて構成したマシンACに対して，そのyを追加する処理を全てのキーワードに対して実行した．

本論文で提案した更新手法は，マシンAC全体の再構築に比べて，キーワードの逐次更新が，最悪でも約5～110倍，平均では約18～934倍高速になることが実証された．また，表1よりも判るように，キーワード数が少ない場合の改善率は低いが，数百を越えると改善率は非常に高くなる．実際の実行時間は，Sparc Station2で，最悪の場合でも1秒以内で実行できた．

## 4．むすび

以上，本論文ではAhoら[1]の提案したストリングパターンマッチングマシンの検索キー追加法を提案し、実験によりその有効性を確認した．

今後は，本手法を用いた検索キー追加可能なマシンをよりコンパクトに，より高速なものとするため，青江[2]の提案したダブル配列法を用いて実現する予定である．

## 文　献

(1) Aho A.V.and Corasick,M.J.:"Efficient string pattern matching An aid to bibliographic search" ,Commun.ACM, 18, 6. pp.333-340 (1975).

(2)Aoe J. : "An Efficient Digital Search Algorithm by Using a Double-Array Structure", IEEE Trans. Software Eng., 15,9, pp.1066-1077 (Sept.1989).

(3)津田，入口，青江："ストリンクパターンマッチングマシンにおける検索キー更新方法", 情処学AL研報, 32-6 (1993-03)

表1　キーワードの追加に伴うマシン構築の時間コスト表

| | C言語 | pascal | COBOL | UNIX | 世界の都市 |
|---|---|---|---|---|---|
| マシンACの概要 | | | | | |
| キーワードの数 | 32 | 35 | 310 | 685 | 1480 |
| キーワード長の平均 | 5.19 | 4.14 | 6.50 | 5.58 | 8.51 |
| キーワード長の総和 | 166 | 145 | 2016 | 3819 | 12599 |
| 状態数 | 141 | 125 | 1247 | 2123 | 8261 |
| 変数depの平均値 | 0.78 | 0.57 | 2.48 | 2.48 | 2.93 |
| マシン作成コスト | 1275 | 1134 | 12607 | 22306 | 77420 |
| 追加処理 | | | | | |
| 平均計算コスト | 71.34 | 55.00 | 53.76 | 46.43 | 82.91 |
| 最大計算コスト | 271 | 141 | 250 | 297 | 699 |
| 平均failure逆関数遷移数 | 11.59 | 8.86 | 7.15 | 6.33 | 9.12 |
| 最大failure逆関数遷移数 | 55 | 28 | 50 | 61 | 121 |