D-032

複数パターンを高速に照合する遷移関数の実現

Implementation of Transition Function for a Fast Multiple Pattern Matching Machine

中川　知之[†]　　蔵満　琢麻[†]　　望月　久稔[†]

Tomoyuki NAKAGAWA　Takuma KURAMITSU　Hisatoshi MOCHIZUKI

1．はじめに

パターン照合は基本的な処理であり高速性が重要である．複数パターンの照合に用いられる手法として AC 法 [1] が知られている．高速なパターン照合を行うために，全遷移種分の要素を持つ配列を各節点に用いて，AC マシンを決定性有限オートマトンとして実現することが考えられるが，登録パターン数の増加に伴い節点数が増加し，照合時の記憶領域が膨大になる．

有川らが提案した遷移種を分割する AC マシン [4] は，照合時の記憶領域を抑制するが，照合時の遷移回数が増加するため記憶領域と照合時間がトレードオフの関係になる．また，信種らはダブル配列を用いて高速性とコンパクト性をあわせもつ AC マシン [3] を提案した．この手法は Failure 遷移を含む非決定性有限オートマトンである．

本論文では，ダブル配列を用いた AC マシンに遷移先を定義する条件を設けて，Failure 遷移を除去し，Goto 遷移先が未定義の際に遷移すべき節点を一意に決定する遷移関数を定義する．

2．照合を高速化する遷移関数

AC マシンにおける Goto 遷移の一部は登録パターン集合によるトライ構造を含む．以下，トライ構造を効率的に実現するダブル配列について説明し，ダブル配列を用いて高速な AC マシンを実現する方法を述べる．

ダブル配列は一次元配列 Base，Check を用いてトライを表現し，節点 s から遷移種 a による節点 t への遷移を式 (1) で定義する．節点 s の Base 値は遷移先を求めるための基底位置を表し，節点 t の Check 値は節点 s からの遷移種を表す．以下，節点 s における配列 Base，Check の要素それぞれを $\mathrm{B}[s]$，$\mathrm{C}[s]$ とする．

$$\begin{cases} t = \mathrm{B}[s] + a \\ \mathrm{C}[t] = a \end{cases} \tag{1}$$

AC マシンの照合速度を高速化するにはあらかじめ未定義である Goto 遷移を定義すればよい [1]．そのためには，同一節点へ複数の遷移を定義する必要があるが，式 (1) を用いて遷移を定義するダブル配列は異なる Base 値をもつ節点から同一節点への遷移を定義できない．そこで，既存節点 t へ新たに節点 s からの遷移を追加する場合，等しい Base 値をもつ節点が遷移情報を共有する性質 [2] [3] を利用し，節点 t と同じ Base 値をもつ節点 t_p を節点 s からの遷移先として作成する．これにより節点 t_p が節点 t と等価な遷移情報をもつため，擬似的に節点 s から節点 t への遷移を定義できる [2]．以下，節点 t_p を擬似節点と呼ぶ．

全ての節点に全遷移種による遷移先を擬似節点を用いて定義すると，登録パターン数の増加により節点数が膨大になるため，擬似節点の作成条件を設ける．式 (1) より，節点 s の Check 値は節点 s へ遷移するための遷移種を表す．照合中の節点 s における Check 値と対象データにおける遷移種を用いることで 2 遷移分の遷移種が分かるため，遷移すべき節点のトライ構造上の深さが 2 以下であれば初期節点から 2 回遷移することで遷移すべき節点を特定できる．以上より，擬似節点の作成条件を以下に定める．

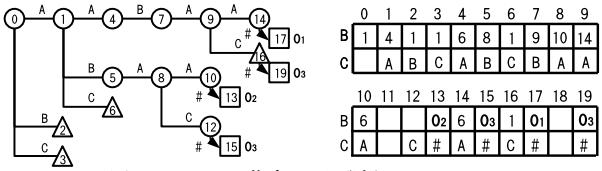

図 1　パターン集合 K を登録した AC マシン

[**条件 1**] 遷移すべき節点の深さが 3 以上である．

[**条件 2**] 遷移元となる節点の深さが 2 未満である．

条件 1 で作成する擬似節点により，Goto 遷移が未定義であった場合，遷移すべき節点をトライ構造上の深さが 2 以下の節点に限定できる．また，条件 2 で作成する擬似節点により初期節点からの 2 遷移をすべて定義できる．そこで，節点 s から遷移種 a による Goto 遷移が未定義である時，式 (2) を用いて遷移先を決定する．式 (2) を用いることで，Goto 遷移が未定義であるときの遷移先が一意に決まるため，照合速度の高速化が図れる．

$$t = \mathrm{B}[\mathrm{B}[\text{初期節点}] + \mathrm{C}[s]] + a \tag{2}$$

次に，作成する擬似節点数をさらに抑制する方法について述べる．Goto 遷移を持たない節点は，全ての Goto 遷移が未定義であるため，Failure 遷移先の遷移情報と等価にすればよい．そこで，Goto 遷移先を持たない節点においては，擬似節点を作成せずに，Filure 遷移先の Base 値を写像することで遷移先を定義する．これにより外部節点となる擬似節点を削減できるため，照合時の記憶領域を抑制できる．

最後に，提案マシンにおける Output 集合の管理方法について述べる．Output 集合の存在しない節点においては，Output 集合の管理領域は不要である．そこで，1 節点あたりの記憶領域を抑制するために，Output 集合のインデクスをパターンに使用しない遷移種による遷移先の Base 値で管理する．‘#’ を遷移種の最大値 +1 と定義し，節点 s における Output 集合のインデクスを管理する節点 s_o への遷移定義式を式 (3) に示す．

$$\begin{cases} s_o = s + \text{‘\#’} \\ \mathrm{C}[s_o] = \text{‘\#’} \end{cases} \tag{3}$$

[**例**] パターン集合 K={“AABAA”，“ABAA”，“ABAC”} を登録した提案マシンを図 1 に示す．図 1 において，遷移種 ‘A’，‘B’，‘C’，‘#’ の内部表現値をそれぞれ 0, 1, 2, 3 とし，丸枠でトライ構造を構成する節点 (以下, 一般節点), 三角枠で擬似節点, 四角枠で Output 集合を管理する節点を表す．各枠内の値は節点番号を表す．節点 16 が条件 1 により，節点 2, 3, 6 が条件 2 により作成した擬似節点に該当する．また，Output 集合 {“AABAA”，“ABAA”}，{“ABAA”}，{“ABAC”} のインデクスをそれぞれ O_1，O_2，O_3 とする．ダブル配列上の Base 値，Check 値がともに空白である要素は未使用であることを表し，Check 値にはいずれの遷移種とも一致しない値を格納する．

† 大阪教育大学, Osaka Kyoiku University

図 1 で対象データ “$A_1C_2A_3B_4A_5A_6A_7$” を照合する例を示す．対象データの添字は対象データにおける位置を表す．初期節点 0 において，遷移種 ‘A_1’ より式 (1) を用いて B[0]+‘A_1’= 1，C[1] = ‘A’ となり節点 1 へ Goto 遷移する．次に遷移種 ‘C_2’ より B[1]+‘C_2’= 6，C[6] =‘C’ となり擬似節点 6 へ Goto 遷移する．次に遷移種 ‘A_3’ より節点 1 へ遷移する．このように擬似節点 6 は節点 1 から遷移種 ‘C’ により遷移すべき節点 0 と等しい Base 値をもつため，節点 0 と等価な遷移情報を持つ．同様に遷移種 ‘B_4’，‘A_5’，‘A_6’ より節点 5，8，10 へ順に Goto 遷移する．節点 10 は，式 (3) より C[10 + ‘#’]=‘#’ であるから O_2 を出力し，式 (1) より B[10]+‘A_7’=6，C[6]≠‘A’ となり，Goto 遷移が未定義であるため，式 (2) を用いて遷移先を決定する．B[B[0] + C[10]]+‘A_7’= 4 となり節点 4 へ遷移する．ここで，対象データの末尾まで走査したため照合を終える． **(例終)**

このように Goto 遷移失敗時に遷移すべき節点を返す関数を定義することで，照合時に Failure 遷移が発生しないため，照合速度の高速化を図れる．

3． 実験による評価

提案手法の有効性を示すため，有川らの AC マシン [4](以下，比較手法 A) および信種らの AC マシン [3](以下，比較手法 B) との比較実験を Intel Pentium Dual 1.60GHz，Fedora7 上で行った．

実験ではランダムに作成した 10 万個のバイナリパターン集合 (平均パターン長 10) を母集団として用いた．母集団からパターン数を 1 万個から 10 万個まで 1 万個毎に増やしたパターン集合を作成し，それぞれ 20MB の対象データと照合した．実験におけるパターンの遷移種は 1byte で構成され，比較手法 A では遷移種を 4bit に分割して登録した．各手法における照合時間を図 2 に，照合時の記憶領域を図 3 に示す．また，各手法において，10 万個のパターン登録時における照合時間，照合時の記憶領域，節点数，照合時の遷移回数を表 1 に示す．また，1 万個，10 万個の集合を登録した提案手法の一般節点における Failure 遷移先の深さ平均と 1 節点あたりに付加する平均擬似節点数を表 2 に示す．

図 2 に示すように，提案手法は最も高速に対象データを照合した．10 万個のパターン登録時，提案手法の照合時間は比較手法 A の 0.35 倍，比較手法 B の 0.76 倍であった．これは表 1 に示すように，提案手法の遷移回数が最も少ないためである．

次に照合時の記憶領域について評価する．図 3 に示すように，提案手法は登録パターン数の増加に伴い記憶領域の増加率が増す．これは，表 2 に示すように，登録パターン数の増加に伴い Failure 遷移先のトライ構造上の深さが増し，1 節点あたりに付加する擬似節点数が増加するためである．これにより，10 万個のパターン登録時，提案手法は他手法に比べ節点数が多くなり，同じダブル配列を用いる比較手法 B に対しては記憶領域が 1.55 倍となった．しかし，提案手法は比較手法 A と比較すると 0.13 倍の記憶領域で対象データを照合した．これは比較手法 A が 1 節点あたりに 4bit で表現できる全遷移種による遷移先を保持する配列を必要とするのに対し，提案手法の 1 節点あたりに必要な記憶領域が Base 値，Check 値を保持する領域のみとなるためである．

以上より，提案手法が膨大な記憶領域を消費することなく照合時間を短縮できることを示した．

表 1　10 万個のパターン登録時の実験結果

	提案手法	比較手法 A	比較手法 B
照合時間 (s)	1.86	5.31	2.43
記憶領域 (byte)	14,347,460	109,449,672	9,271,248
節点数	3,586,865	1,609,556	2,317,812
遷移回数	20,000,000	40,000,000	39,895,726

表 2　1 万個，10 万個のパターン登録時の Failure 遷移先の深さ平均と平均擬似節点数

登録パターン数	1 万個	10 万個
Failure 遷移先の深さ平均	1.13	1.74
平均擬似節点数	0.54	2.16

4． おわりに

本論文では，Goto 遷移先が未定義の際に遷移すべき節点を一意に決定する遷移関数を提案し，比較実験によりその有効性を示した．今後の課題として，登録パターン数の増加に伴う擬似節点の増加を抑制することが挙げられる．

参考文献

[1] Aho A.V., M.J., Efficient String Matching An Aid to Bibliographic Search, Comm. ACM, vol.18, No.6, pp.333-340, 1975.

[2] 蔵満琢麻，松浦寛生，望月久稔，遷移関数の拡張により高速化したマシン AC とアンチウイルスへの応用，第 19 回データ工学ワークショップ論文集 (DEWS2008)，E7-3，2008.

[3] 信種真人，森田和宏，泓田正雄，青江順一，ダブル配列を用いたマシン AC の効率的格納手法，言語処理学会第 13 回年次大会，pp.831-834，2007.

[4] 有川節夫，篠原武，文字列パターン照合アルゴリズム，コンピュータソフトウェア，Vol.4，No.2，pp.98-119, 1987.

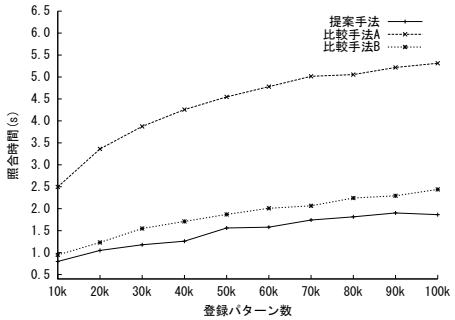

図 2　照合時間

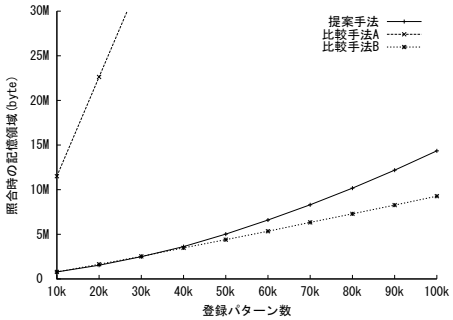

図 3　照合時の記憶領域