D-016

ダブル配列を用いた AC 法の照合アルゴリズム

Pattern Matching Algorithm of Aho-Corasick Machine Using a Double-Array Structure

蔵満 琢麻† 松浦 寛生† 尾崎 拓郎† 望月 久稔†

Takuma KURAMITSU Nobutaka MATSUURA Takuro OZAKI Hisatoshi MOCHIZUKI

1． はじめに

テキストから特定の文字列を抽出することは文書処理において基本的な処理のひとつであり，効率的な照合手法は重要である．高速な文字列照合アルゴリズムとして，AC 法 [1] や，suffix array[2] などがよく知られている．特に AC 法は，複数のキーに対して有効な手法で，データ構造は部分的に木構造を用いたものである．また，多分木に対するマルチウェイ基数探索法のデータ構造に，定数時間での遷移を可能とするダブル配列 [3] がある．本論文では，ダブル配列に配列 FAILURE と配列 OUTPUT を追加して構築した AC 法の照合アルゴリズムを提案し，その効率について評価する．

2． AC 法

AC 法 [1] は，与えられたキー集合からマシン AC と呼ばれる一種の有限オートマトンを作成し，テキストを入力として与えることで文字列照合する手法である．マシン AC は，以下に示す関数 Goto，関数 Failure，関数 Output から構成される．以下，関数 Goto による遷移を Goto 遷移，関数 Failure による遷移を Failure 遷移とする．

関数 Goto(s, l)：状態 s からラベル l による遷移先 t を返す．定義されていない遷移に対しては，遷移先未定義を表す *fail* を返す．
関数 Failure(s)：関数 Goto が *fail* を返したときに呼び出される関数で，遷移失敗時における状態 s からの遷移先を返す．
関数 Output(s)：状態 s において検出されるキーを出力する関数である．Goto 遷移により照合ポインタが状態 s に到達したときに呼び出される．また，ある状態において検出されるキーは，一つとは限らず，集合として管理されている．

3． ダブル配列を用いた AC 法

ダブル配列 [3] のデータ構造である配列 BASE，CHECK に加え，配列 FAILURE，OUTPUT を追加し，AC 法を実装する．以下，状態 x における BASE 値，CHECK 値，FAILURE 値，OUTPUT 値をそれぞれ $B[x]$，$C[x]$，$F[x]$，$O[x]$ とする．

3.1 データ構造

配列 BASE，CHECK を用いて，状態 s からラベル l による状態 t への Goto 遷移を式 (1) で定義する．配列 FAILURE を用いて，状態 s からラベル l による Goto 遷移先が存在しない場合の Failure 遷移を式 (2) で定義する．式 (2) において u は状態 s からの Failure 遷移により辿りつく状態を表す．

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

$$F[s] = u \tag{2}$$

マシン AC には，初期状態から初期状態自身への Goto 遷移が存在する [1]．しかしダブル配列の定義上，同じ状態に対して複数の Goto 遷移を構築することができない．そこで，ある状態 s において Goto(s, l) が *fail* を返した場合，s が初期状態かの判定を加えることで初期状態自身への Goto 遷移を定義する．

初期状態から状態 s までの遷移により構成する文字列を $str(s)$ とし，ある状態において出力するキー集合を Output 集合とする．状態 s における Output 集合は，マシン AC に登録するキーに一致する $str(s)$ の接尾辞を要素にもつ [1]．

Output 集合を配列 OUTPUT と配列 OUTNEXT を用いてリスト構造で管理する．キーの登録順にキー番号を付け，配列 OUTNEXT の添字に対応させることで，キー番号を用いて Output 集合を扱う．以下，キー番号 k における OUTNEXT 値を $N[k]$ とする．$O[s]$ には $str(s)$ がマシン AC に登録するキーと一致する場合はキー番号を正値で格納し，一致しない場合は状態 s における Output 集合のうち最長のキーのキー番号を負値で格納する．また，Output 集合が空集合の場合は 0 を格納することで状態 s におけるキーが存在しないことを表す．$N[k]$ には Output 集合のうち，k が指すキーの次に長いキーのキー番号を格納する．該当するキーが存在しない場合は 0 を格納して Output 集合の終端を表す．

例 1: キー集合 K {“abcd”, “abcde”, “bcdd”, “d”, “dec”} をそれぞれ k_1, k_2, k_3, k_4, k_5 とし，図 1 にキー集合 K を登録したマシン AC の状態遷移図と，配列に格納した値を示す．状態遷移図における実線は Goto 遷移を表し，破線は状態 1 以外への Failure 遷移を表す．ここで，遷移種 ‘\0’, ‘a’, ‘b’, ···, ‘e’ の内部表現値を 0, 1, 2, ···, 5 とする．状態 8 における Output 集合は，$O[8]$ が指すキー k_1 と，$N[O[8] = 1]$ が指すキー k_4 である．ここで，$N[N[1] = 4]$ は 0 であり，他のキーは存在しない．また，状態 11 においては，$O[11]$ が指すキー k_4 が Output 集合となる．

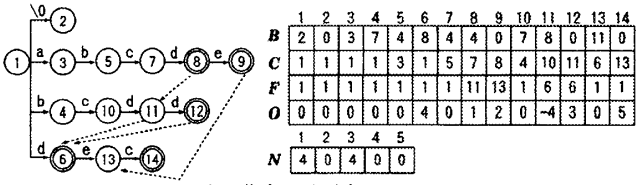

図 1 キー集合 K を登録したマシン AC

3.2 照合アルゴリズム

提案手法の照合アルゴリズムと，キーの出力に使用する関数 Output を以下に定義する．

(手順 1) 状態の初期化

配列 *buffer* にテキストを取り込む．照合ポインタ s を初期状態の状態番号 1 に，*buffer* の走査位置 *location* を 0 に設定する．

(手順 2) ダブル配列上の遷移

location が *buffer* に取り込んだサイズ未満の間，手順 2 を繰り返す．ラベル l に *buffer*[*location*] を設定する．式 (1) で定義した Goto 遷移の有無により条件分岐する．以下に条件 A，B およびそのときの処理を示す．

† 大阪教育大学, Osaka Kyoiku University

［条件 A］ Goto 遷移が存在する場合

s を Goto 遷移先に設定し，*location* をひとつ進める．$O[s]$ が正値なら関数 Output($O[s]$) を呼び出し，$O[s]$ が負値なら関数 Output($-O[s]$) を呼び出す．

［条件 B］ Goto 遷移が存在しない場合

s が初期状態であるならば *location* をひとつ進める．そうでなければ s を $F[s]$ に設定する．

関数 Output(*index*)：キー番号 *index* に対応するキー情報を出力し，*index* に $N[index]$ を設定する．この動作を *index* が 0 になるまで繰り返す．

例 2: 例 1 のマシン AC からテキスト “abcdcd”を照合する例を示す．手順 1 より *buffer* に “abcdcd”，s に 1，*location* に 0 を設定する．このとき *buffer* に取り込んだサイズは 6 である．

式 (1) より，状態 1 からラベル ‘a’ による状態 3 への Goto 遷移が存在するため，手順 2 の条件 A より，s に 3 を設定し．*location* をひとつ進める．同様にトライ上を状態 5，7，8 の順に Goto 遷移し，s は 8，*location* は 4 となる．ここで状態 8 は Output 集合をもつため関数 Output により，k_1，k_4 を得る（例 1）．

次に，状態 8 からラベル ‘c’ による Goto 遷移が存在しないため，手順 2 の条件 B より，s に 11 を設定する．同様にトライ上を状態 6，1 の順に Failure 遷移する．ここで，状態 1 からラベル ‘c’ による Goto 遷移は存在しないが，状態 1 は初期状態なので *location* をひとつ進め，*location* は 5 となる．

その後，状態 1 から状態 6 へ Goto 遷移し，関数 Output により k_4 を得る．このとき，*location* が 6 となり，*buffer* に取り込んだサイズ以上となるので照合を終了する．

4． 実験による評価

提案手法の有効性を示すため，リストを用いた二分木でトライ部分を構築した AC 法 (以下，比較手法 L) および二分探索を用いて文字列照合する suffix array[2](以下，比較手法 S) との比較実験を Intel Pentium4 2.8GHz，Fedora Core 4 上で行った．実験では，ランダムに抽出した英単語 30 万語（以下キー集合 U とする）を用いた．また，遷移のラベルとなるキーの各構成文字の内部表現値は ASCII コードとした．

キー集合 U を母集団として 1 万語から 30 万語まで 1 万語毎に単語数を増やしたキー集合と，U を連結したものをテキストとして作成し，各キー集合とテキストを照合した．各手法において，照合時間を図 2 に，照合に必要な空間を図 3 に示す．また，表 1 に提案手法の Failure 遷移回数と出力回数を示す．

図 2 に示すように，提案手法は比較手法 L よりも照合時間が短い．これはトライ上の遷移先決定において，比較手法 L が各遷移先をリストで辿る必要があるのに対し，提案手法は一意に遷移先が決まるからである．また，キー数に照合時間が比例する比較手法 S に比べ，提案手法はキー数の増加に伴う照合時間の増加を抑えられる．これは表 1 に示すように，キー数の増加に伴い，出力回数は増えるが，Failure 遷移回数が減るためである．

また，図 3 より，照合に必要な空間において提案手法は比較手法 L とほぼ同等なことが分かる．提案手法はダブル配列により状態数以上の配列サイズを要するが，一状態あたりの空間が比較手法 L よりも小さいからである．

表 1 提案手法の Failure 遷移回数と Output 回数

キー数	1 万	15 万	30 万
Failure 遷移回数	2,049,882	1,439,016	1,238,067
出力回数	162,050	3,530,015	7,151,581

AC 法と suffix array の性質の違いにより，対象のテキストが変わることで図 2，図 3 に示したグラフの位置関係は変化する．今回の実験においては，キー数が 27 万を越えたとき，比較手法 S と比べ，提案手法は使用空間を多く要するが，図 2 より，そのときの照合時間は短いことが分かる．

以上より，ダブル配列を拡張してマシン AC を実装することにより，AC 法の照合時間の短縮が図れ，複数キーの照合に有効な手法であることが示された．

5． おわりに

本論文では，ダブル配列を導入した AC 法の照合アルゴリズムを提案し，実験で複数キーにおける照合の有効性を示した．今後の課題として，構築を含めた評価が挙げられる．

参考文献

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

[2] Juha K., Peter S., Simple Linear Work Suffix Array Construction, ICALP 2003, LNCS 2719, pp.943-955, 2003.

[3] J. Aoe, An Efficient Digital Search Algorithm by Using a Double-Array Structure, IEEE Trans. on Software Engineering, Vol.15, No.9, pp.1066-1077, 1989.

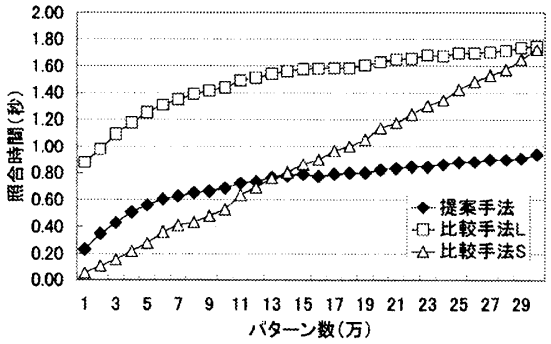

図 2 各手法における照合時間

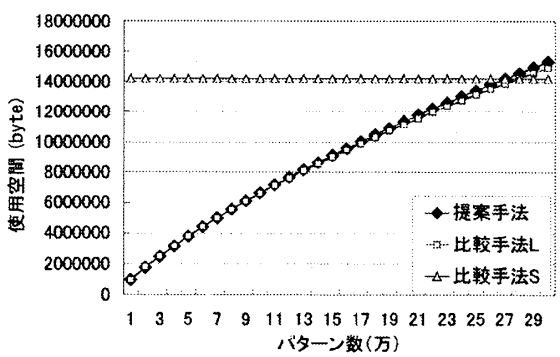

図 3 各手法において照合に必要な空間