有限オートマトン学習支援システムの開発とその応用

鎌田 晋伍[1]　石坂 裕毅[2]

概要：本論文では，有限オートマトン学習支援システムとその応用である AC 実験ツールの開発について述べる．有限オートマトン学習支援システムは，オートマトンの作成，合成，最小化，NFA から DFA への変換を GUI ベースで行えるツールであり，これを用いることで手軽にオートマトンに関する検証を行うことができる．AC 実験ツールは，Aho-Corasick 法と呼ばれる文字列照合法に関する実験演習を行うためのツールであり，本大学の授業で実際に用いられている．
また，b 進数による k の剰余が r となるような文字列を受理する剰余オートマトンの状態最小化について，本システムを用いた実験と理論的考察から得られたいくつかの結果を与える．

Development of a Learning Assistant System for Finite Automata Theory and Its Applications

Shingo KAMATA[1]　Hiroki ISHIZAKA[2]

Abstract: This paper describes the results of developments of a learning assistant system for Finite Automata theory and AC simulator. The system can be used for defining automata, constructing automata that recognize languages that are obtained by applying language operations, DFA minimization, and converting NFA to DFA. AC simulator is a tool which simulates Aho-Corasick string matching algorithm. AC simulator also was developed as an application of the system. Moreover, we discuss Residue Automata minimization. Residue Automata is a DFA which can accepting the strings corresponding to b-base numbers of a residue r by a divisor k. This paper describes some results obtained from experiments of the system and a theoretical studies of Residue Automata.

1. はじめに

オートマトン理論は，情報科学・工学において極めて重要な理論である．情報学科のある多くの大学で，オートマトン理論に関する講義が開講されている．しかし，実際にオートマトンを作成し，状態最小化やオートマトンの合成等を検証するための十分なツールはまだ普及していない．いくつかの検証ツールはあるが，それらは，学習を行うという観点からいくつかの問題がある．例えば，グラフベースで DFA や NFA を入力し，文字列を走査できるものがあるが，これは入力したオートマトンに対して合成・最小化などを行うことができない．また，この入力方法では，大きく複雑なオートマトンに対する検証が困難である．その他にも，ユーザがコンパイルして実行する形式のものや，プログラムの API として提供されているものもあるが，それらは手軽に検証を行えるという観点から不十分である．

そこで，専用のエディタから編集を行う形で，オートマトンの入力・保存ができ，合成，最小化，NFA から DFA への変換にも対応した学習支援システムの開発を行った．このシステムは，Java 言語で開発を行ったため，OS によらず実行可能であり，現在は Web システム化を行っている．

また，本稿では，Aho-Corasick 法とよばれる文字列照合法に関する実験ツール開発についても述べる．Aho-Corasick 法は，pmm と呼ばれる一種のムーアマシンを用いることで文字列照合を行うアルゴリズムである．そのため，このツールは有限オートマトン学習支援システムをベースとして開発することができた．開発したツールは，筆者らが所

[1] 九州工業大学大学院情報工学府
Graduate School of Computer Science and Systems Engineering , Kyushu Institute of Technology
[2] 九州工業大学情報工学研究院
Department of Artificial Intelligence , Kyushu Institute of Technology

属する大学の講義「情報工学基礎実験 IIA オートマトン」にて実際に運用されている．

さらに，本システムを用いることで検証を行った，剰余オートマトンとよばれる DFA についての考察も述べる．剰余オートマトンとは，b 進数による k の剰余が r となるような文字列を受理する DFA であり，有限オートマトン学習支援システムによる多くの実験データと，それらに対する理論的考察から，いくつかの性質を数学的に証明することができた．

2. 学習の対象とする範囲

この章では，剰余オートマトンや有限オートマトン学習支援システムを理解する上で必要となる有限オートマトンの定義や性質について，参考文献 [1] に基づいて解説を行う．

決定性有限オートマトン（deterministic finite automaton 以下，**DFA** と略す）は，5 項組 $(Q, \Sigma, \delta, q_0, F)$ によって定義される．Q は空でない有限集合であり，その要素は状態とよばれる． Σ はアルファベットである．δ は $Q \times \Sigma$ から Q への関数で遷移関数とよばれる．q_0 は Q のある要素で初期状態といい，F は Q の部分集合で，F の要素を最終状態という．遷移関数 δ の拡張 δ^* を

$$\begin{aligned} &\delta^*(q, \varepsilon) = q && (q \in Q) \\ &\delta^*(q, ax) = \delta^*(\delta(q, a), x) && (q \in Q,\ a \in \Sigma,\ x \in \Sigma^*) \end{aligned}$$

と定義する．語 $w \in \Sigma^*$ に対し，$\delta^*(q_0, w) \in F$ であるとき，M は w を受理するという．M によって受理されるすべての語の集合を $L(M)$ で表す．$L = L(M)$ が成り立つとき，M は言語 L を受理するという．

DFA が異なっていても，言語の受理能力が等価な場合がある．DFA の場合，より状態数が少ない，つまり状態数最小な DFA を求めるアルゴリズムが存在する．図 1 は，ある DFA とその状態最小後の DFA を比較したものである．

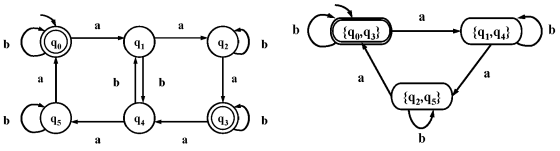


図 1: 最小化前の DFA と最小化後の DFA

DFA $M = (Q, \Sigma, \delta, q_0, F)$ の 2 つの状態 $q_1, q_2 \in Q$ が，任意の $x \in \Sigma^*$ に対して

$$\delta^*(q_1, x) \in F \Leftrightarrow \delta^*(q_2, x) \in F$$

であるとき同値であるといい，$q_1 \underset{Q}{\equiv} q_2$ と書く．関係 $\underset{Q}{\equiv}$ は明らかに同値関係である．この $\underset{Q}{\equiv}$ 関係を用いて，DFA $M = (Q, \Sigma, \delta, q_0, F)$ の最小状態 DFA は以下のように求めることが出来る．まず，M の状態に到達不能な状態が存在すれば，その状態は言語の受理能力に影響を及ばさないため，削除を行う．到達不能な状態は，有限時間で検出可能であることが保証されている．$q \in Q$ を代表元とする $\underset{Q}{\equiv}$ に関する状態の同値類を $[q]$ で表し，M をもとに 5 項組 $M' = (Q', \Sigma, \delta', q_0', F')$ を以下のように定義する．

$$\begin{aligned} &Q' = \{[q] \mid q \in Q\} \\ &\delta'([q], a) = [\delta(q, a)] && (q \in Q, a \in \Sigma) \\ &q_0' = [q_0] \\ &F' = \{[q] \mid q \in F\} \end{aligned}$$

この構成法で作られた M' は $L(M)$ を受理する最小状態 DFA になることが保証されている．

非決定性有限オートマトン（nondeterministic finite automaton 以下．**NFA** と略す）は，5 項組 $(Q, \Sigma, \delta, Q_0, F)$ によって定義される．δ は $Q \times \Sigma$ から Q の部分集合全体 2^Q への関数である．NFA が受理する言語を定めるため，δ を拡張する．まず，$Q \times \Sigma^*$ から 2^Q への関数 δ^* を

$$\begin{aligned} &\delta^*(q, \varepsilon) = \{q\} \quad (q \in Q) \\ &\delta^*(q, ax) = \bigcup_{p \in \delta(q,a)} \delta^*(p, x) \quad (q \in Q,\ a \in \Sigma,\ x \in \Sigma^*) \end{aligned}$$

（ただし，$\delta(q, a) = \phi$ の場合には $\bigcup_{p \in \delta(q,a)} \delta^*(p, x) = \phi$）と定義し，さらに，$2^Q \times \Sigma^*$ から 2^Q への関数 $\widehat{\delta^*}$ を

$$\widehat{\delta^*}(S, x) = \bigcup_{q \in S} \delta^*(q, x) \quad (S \subseteq Q, x \in \Sigma^*)$$

と定義する．語 $w \in \Sigma^*$ に対し，$\widehat{\delta^*}(Q_0, w) \cap F \neq \phi$ であるとき，M は w を受理するという．NFA と DFA は等価であるため，NFA は DFA に変換することが出来る．この，有限オートマトンが受理できる言語の族を正則言語とよぶ．NFA $M = (Q, \Sigma, \delta, Q_0, F)$ に対応する DFA $M' = (Q', \Sigma, \delta', q_0', F')$ は以下のように構成される．

$$\begin{aligned} &Q' = 2^Q \\ &\delta'(S, a) = \bigcup_{q \in S} \delta(q, a) \quad (S \in Q' = 2^Q, a \in \Sigma) \\ &q_0' = Q_0 \\ &F' = \{R \mid R \in Q' \text{かつ } R \cap F \neq \phi\} \end{aligned}$$

この DFA M' の受理能力は NFA M と等価であることが保証されている．

正則言語は，和集合，積集合，差集合，補集合に関する演算に対して閉じている．これらの演算を行った正則言語を受理するためには，演算に対応した有限オートマトンの合成を行う．ここでは簡単のため，Σ 上の 2 つの DFA

$$\text{DFA } M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1) \quad \text{s.t.} \quad L(M_1) = L_1$$
$$\text{DFA } M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2) \quad \text{s.t.} \quad L(M_2) = L_2$$

をもとに，正則言語 L_1, L_2 を定義し，L_1, L_2 間の言語演算を受理する DFA M を M_1，M_2 を用いることで構築する．

$L_1 \cap L_2$ を受理する DFA $M = (Q, \Sigma, \delta, q_0, F)$ は以下のように構成すれば良い．

$$Q = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1,\ q_2 \in Q_2\}$$
$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)) \quad (q_1 \in Q_1,\ q_2 \in Q_2,\ a \in \Sigma)$$
$$q_0 = (q_0^1, q_0^2)$$
$$F = F_1 \times F_2$$

また，$L_1 \cup L_2$ を受理する DFA は，$L_1 \cap L_2$ を受理する DFA M の F を

$$F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$$

と変更すれば良い．

L_2 の補集合 L_2^C を受理するためには，DFA $M = (Q_2, \Sigma, \delta_2, q_0, Q_2 - F_2)$ とすればよい．

$L_1 - L_2$ を受理する DFA を構成するためには，$L_1 - L_2 = L_1 \cap L_2^C$ より，既に構成を示した補集合と積集合を受理する DFA を組み合わせればよい．

L_1 と L_2 の連節，$L_1 \cdot L_2$ を受理する NFA $M = (Q, \Sigma, \delta, Q_0, F)$ は以下のように構成できる．

$$Q = Q_1 \cup Q_2$$
$$\delta(q, a) = \begin{cases} \{\delta_1(q, a)\} & (q \in Q_1 \text{ かつ } \delta_1(q, a) \notin F_1) \\ \{\delta_1(q, a), q_0^2\} & (q \in Q_1 \text{ かつ } \delta_1(q, a) \in F_1) \\ \{\delta_2(q, a)\} & (q \in Q_2) \end{cases}$$
$$Q_0 = \begin{cases} \{q_0^1\} & (q_0^1 \notin F_1) \\ \{q_0^1, q_0^2\} & (q_0^1 \in F_1) \end{cases}$$
$$F = F_2$$

L_1 の閉包 L_1^* を受理する NFA $M = (Q, \Sigma, \delta, Q_0, F)$ は以下のように構成できる．

$$Q = Q_1 \cup \{q_0'\} \qquad (q_0' \notin Q_1)$$
$$\begin{cases} \delta(q_0', a) & = \phi \\ \delta(q, a) & = \begin{cases} \{\delta_1(q, a), q_0^1\} & \delta_1(q, a) \in F_1 \\ \{\delta_1(q, a)\} & \delta_1(q, a) \notin F_1 \end{cases} \end{cases}$$
$$Q_0 = \{q_0^1, q_0'\}$$
$$F = F \cup \{q_0'\}$$

Aho-Corasick 文字列照合法（以下，AC 法と略す）は，キーワードの集合であるパターン集合から，パターン照合機械（Pattern Maching Machine 以下，pmm と略す）と呼ばれる ε 遷移を持ったムーアマシンを作成し，対象テキストの走査を行うことで各パターンの出現位置を検出するアルゴリズムである．よってこのアルゴリズムは，与えられたパターン集合から pmm を作成する部分と，pmm 上でテキストの走査を行う部分とに分かれる．pmm 構成部分についてはパターンの長さの和に関して，テキストの走査を行う部分については入力テキストの長さに関して，どちらも線形であるため，効率のよいアルゴリズムとして知られている [2]．pmm は以下の 3 つの手順から作ることができる．1. パターン集合から，トライ木と呼ばれる有効グラフの作成を行う．2. トライ木に失敗遷移と呼ばれる ε 遷移を付け加える．3. 出力となるキーワードの統合を行う．例えば，パターン集合 $\{ac, ba, bb, acb, bac\}$ に対する pmm は，図 2 のようになる．

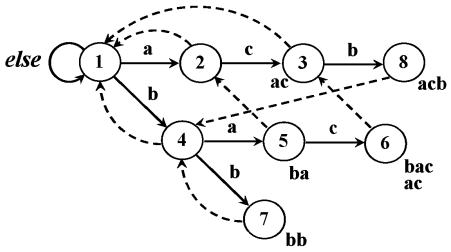


図 2: トライに failure 関数を加えたもの

pmm は failure 関数を除去することで，直感的には決定性の有限オートマトンと同じものとすることができる．失敗遷移を除去したものが図 3 の pmm である．

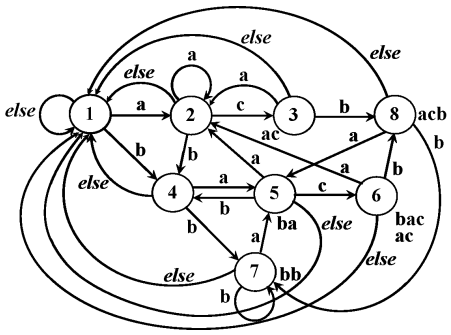


図 3: failure 関数を除去した pmm

図 3 の pmm 上でテキスト cbbacb の走査を行った結果が図 4 である．

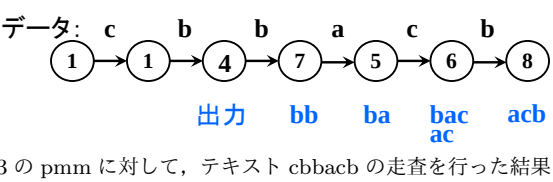


図 4: 図 3 の pmm に対して，テキスト cbbacb の走査を行った結果

3. 有限オートマトン学習支援システム

ここでは，作成したツールとその改善について述べる．一連のツールは Java で開発を行い，特にインタフェース部では Java Swing [3], [4]，ファイル保存・読み込み時の構文解析では JavaCC [5]，グラフの描画ではオープンソースのグラフライブラリ jung2.0 [6] を用いた．作成した一連のツールにはバリデーション（値検査）が施してあり，不正なオートマトンが誤って入力される可能性を極力排除している．

3.1 JavaSwing による学習支援システム

学習支援システムには大きく分けて 3 つの機能がある．

a) 有限オートマトンを入力する機能
b) 入力したオートマトンを合成，最小化，非決定性から決定性へ変換する機能
c) 入力したオートマトンを描画する機能

また，a)，b) を行った後で，再利用を可能とするためにテキストファイルとしての保存・読み込みの機能もある．

図 5 のツールが a) にあたる入力ツール，図 6 のツールが b) にあたる合成ツールである．このシステムはタブでツールの切り替えを行う．これらは JavaSwing で作成したアプリケーションであるため，デスクトップ・アプリケーションとして実行される．

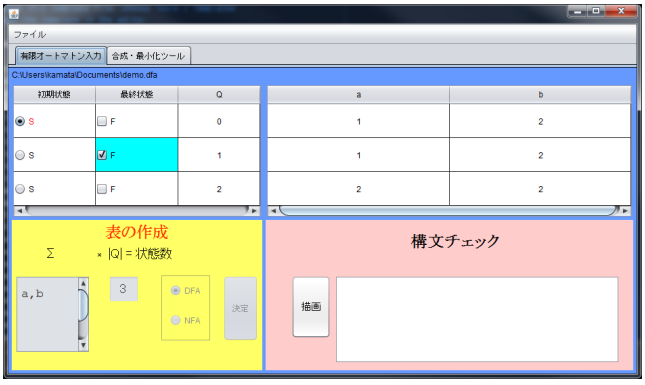


図 5: 入力ツール

a) 入力ツール

入力したいオートマトンの状態数 $|Q|$ と文字集合 Σ，DFA か NFA の選択を入力し，作成ボタンを押すと上部に 2 つの表が作成される．状態名は状態数に応じて自然数が 0 から割り当てられる．左側の表は初期状態と最終状態の選択，右側の $|Q| \times |\Sigma|$ の表は，遷移関数の入力を行う．左表のセルはトグルボタンとなっているため，ワンクリックで初期状態，最終状態の選択ができる．DFA の場合には初期状態はちょうど 1 個であるため，初期状態選択ボタンはラジオボタンとして実装されている．最終状態は，$0 \sim |Q| - 1$ を任意個選択することができる．右側の表では，行 $q \in Q$ と列 $a \in \Sigma$ に対する遷移先 $\delta(q, a)$ を入力する．状態名は自然数しか許されていないので，各セルには $0 \sim |Q| - 1$ の自然数が 1 つ入力される事になる．入力時に不正な値（$0 \sim |Q| - 1$ 外の値）を検出した場合には，図 7 の様な警告を表示させ，セル値をリセットする．

図 6: 合成ツール

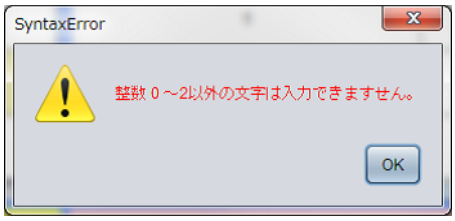


図 7: バリデーション時の警告

NFA を指定して表を生成した場合には，”,”（カンマ）によって遷移先の状態を区切ることで，複数の遷移先を指定できる．初期状態の選択は DFA と異なり，最終状態と同じように複数の状態が選択できる（図 8）．

図 8: NFA 時の入力ツール

画面左上には，図 9 のようなファイル保存読み込みボタンがあり，DFA は.dfa，NFA は.nfa の拡張子で保存される．ファイルは，図 10 にあるような形式の中間ファイルとして保存されている．

b) 合成ツール

合成ツールでは，有限オートマトンの合成，DFA の最小

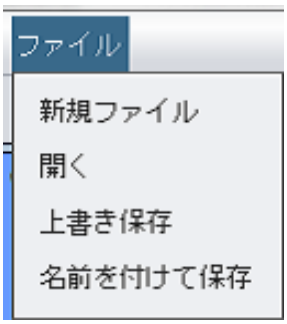


図 9: ファイル保存ボタン

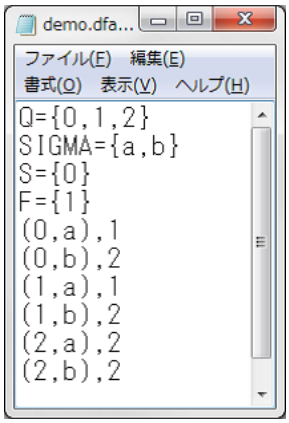


図 10: 中間ファイル

化，NFA から DFA の変換を行う．保存した中間ファイルを 1 つないし 2 つ指定し，行う変換と変換の結果に対して最小化を行うか選択し，実行する．変換がうまくいくと，図 11 のように，もとのオートマトンと変換後のオートマトンが比較して出力される．

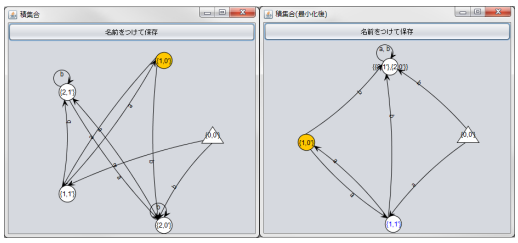


図 11: 合成ツールの実行

入力ファイルが 1 つの場合には，DFA の最小化，NFA から DFA の変換，補集合に対する合成，閉包に対する合成が選択できる．入力ファイルが 2 つの場合には，和，積，差，連接 についての合成が可能である．選択できるファイルは，入力ツールで出力された.dfa，.nfa に対応したファイルのみである．また，最小化が行われた際には，右側のテキストボックスに同値類の分割を行っていく過程が表示される．

c) 描画

入力ツールの描画ボタンや合成ツールの作成ボタンで，入力したオートマトンのグラフを描画することができる (図 11)．三角のノードは初期状態，黄色のノードは最終状態をそれぞれ表している．グラフにはマウスイベントが実装されているため，ドラッグをして動かすことや，スクロールで拡大することができる．

3.2 Java Applet への移行

作成したシステムを実際に教育の場で応用 (「オートマトンと言語理論」の学習教材として運用) するために，Web システム化を行った．Web システム化を行うにあたって，以下の要件を満たさなければならなかった．

(1) レポート作成期間やテスト時にアクセス遮断可能である
(2) 同時接続台数がある程度の数確保されている
(3) OS やブラウザに依存せず使用可能である

また，既存のシステムが JavaSwing で開発されている以上，なるべくプログラムに手を加えずに実現可能であることが求められた．Web システムも Java であれば，3 の要件を満たすことができる．

例えば，jar ファイル (実行ファイル) の直接配布を行ったり，Java Web Start などを用いると，1 の要件を満たすことができない．1 の要件をみたすためには，サーバー・クライアントモデルでのアプリケーション実行を行い，必要となればサーバーを切ることでアクセス不可とするような実装が必要である．そのため，Swing アプリケーションの Java Applet 化を採用した．ブラウザから Java Applet によるアプリケーションにアクセスした様子が図 12 である．Swing のデスクトップ・アプリケーションをそのままブラウザで実行したような形となる．現在はこの Java Applet によるシステムで運用がされている．

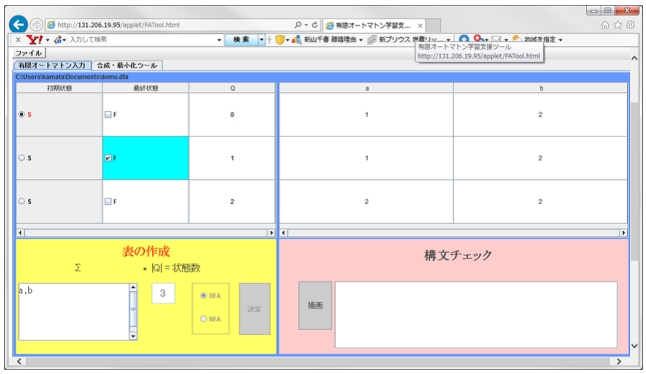


図 12: Applet を用いた Web システム

しかし，Java Applet は脆弱性が発覚し，現在はあまり推奨されない形態となってしまった．そのため，新システムの開発も行っている．新システムでは，インタフェースを JavaScript によるプログラムで一新する代わりに，合成・最小化などのアルゴリズム・ロジック部分は既存の Java プログラムを流用している．図 13，14 はブラウザから新システムを実行した様子である．

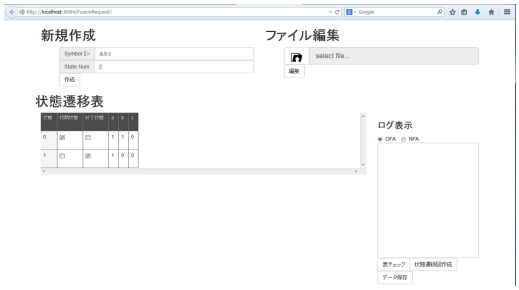


図 13: 新しいシステムの入力

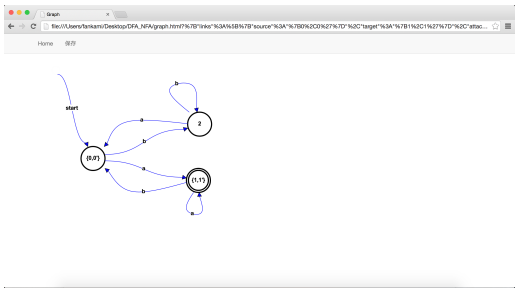


図 14: 新しいシステムのグラフ画面

3.3 Aho-Corasick 文字列照合法の学習支援ツール

AC 法を実装した検索ツールはいくつかあるが，本ツールは所属大学の講義，情報工学基礎実験 IIA「オートマトン」における学習支援が目的であるため，AC 機械の入力とシミュレートを行うツールとなっている．実験の最終週では，レポート課題として，各自異なったパターン集合に対して pmm 作成を行う．このパターン集合はゲノムデータをベースとしたものとなっており，検索対象テキストはタンパク質ゲノムデータ（1M バイト）となっている．つまり，この実験はバイオインフォマティクスとも関わりがある実験となる．

実験は 3 週にわたって行われる．1 週目では，学生に手計算で pmm を作成させ，作成した pmm をツール上で入力，シミュレートを行い，TA がシミュレート結果（検索結果）を確認する．2 週目では，作成した pmm を手計算で決定性オートマトンに変換し，同じように確認を行う．そのため，1 週目と 2 週目に合わせて，失敗遷移付き，失敗遷移なし，2 種類のツール開発を行った．また，ツールが検索結果をテキストとして出力し，それがエビデンスとなるため，pmm を用いない検索を行ってもレポート課題をクリアすることはできないようになっている．図 15 に示すのが失敗遷移付き，図 16 に示すのが失敗遷移なしのツールである．

AC 法の学習支援ツールは，有限オートマトン学習ツールを元に作られているので，同じようなインタフェース・操作性となっている．最終状態と初期状態を指定するかわりに状態の出力を定義する部分，検索対象テキストを指定する部分などで異なっている．図 17 は，実際にシミュレートを行った実行結果である．シミュレートに成功すると，マッチングの結果をテキストファイルとして保存できる．図 18 は，マッチング結果のテキストファイル，図 19 は，入力した AC マシンの中間ファイルである．

図 15: 失敗遷移付きの AC ツール

図 16: 失敗遷移なしの AC ツール

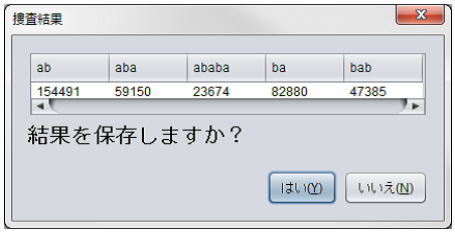
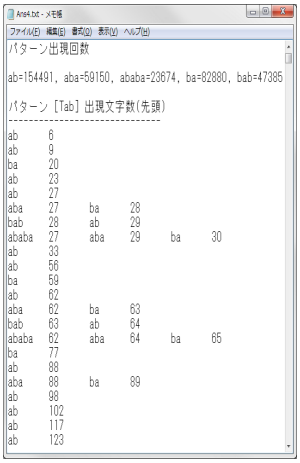


図 17: AC ツールの実行結果



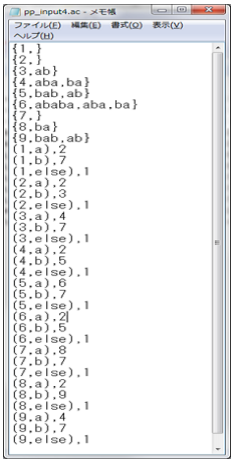


図 18: マッチングを行った結果 図 19: AC ツールの中間ファイル

4. 剰余オートマトンとその最小化について

剰余オートマトンとは，自然数 k による剰余が r となる b 進法表記数を語とする言語を受理する DFA のことであり，$M_b^k(r)$ と表記する．$M_b^k(r)$ は，$b \geqq 2,\ k \geqq 2,\ k-1 \geqq r \geqq 0$ を満たす任意の自然数 b, k, r に対して構成可能である [7]. 例えば，図 20 に示す DFA は $M_2^3(1)$ であり，図 21 に示す DFA は $M_2^4(1)$ である．

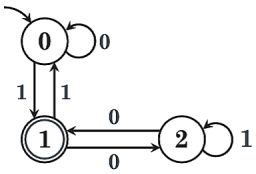


図 20: $M_2^3(1)$

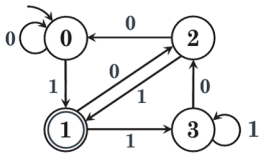


図 21: $M_2^4(1)$

剰余オートマトンの構成を求めるのは比較的簡単である．しかし，どのような場合に状態数が減るのか，状態数が減る場合はどこまで減らすことができるのか，予想をすることは簡単ではない．図 20，21 の剰余オートマトンを最小化したものが，それぞれ図 22，23 である．

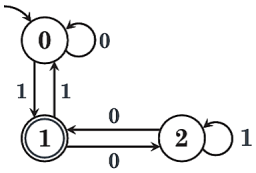


図 22: 最小化後の $M_2^3(1)$

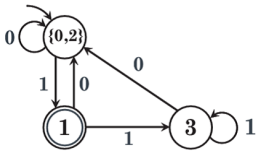


図 23: 最小化後の $M_2^4(1)$

$M_2^3(1)$ は状態数に変化がないのに対し，$M_2^4(1)$ は 1 状態小さくなることがわかる．

類似の研究としては，Alexeev による $M_b^k(0)$ における状態数の考察がある [8] が，本稿では，$r \neq 0$ である場合も考慮し，Alexeev とは異なる証明を与える．

まず進法表記数とその意味について定義を行う．

定義 1　b 進法表記において**数字**とは, $\Sigma = \{0, 1, 2, ..., b-1\}$ に対して $a \in \Sigma$ となる a のことである．また b 進法表記において**数**とは，$x \in \Sigma^+$ となる x のことであり，b **進法表記数**という．

定義 2　b 進法表記数の**意味 (値)** は，Σ^+から $\mathbb{N}$（$\mathbb{N}$ は 0 を含む自然数全体）への関数 $\| \cdot \|_b$ を用いて以下のよう定義する．$a \in \Sigma,\ x \in \Sigma^+$ に対し，

$$\| a \|_b = \text{a に対応する自然数 a}$$
$$\| ax \|_b = \| a \|_b \times b^{|x|} + \| x \|_b$$

文字列 w が b 進法表記数であるとき，w は b' $(b' > b)$ 進法表記数でもある．しかし，$\| w \|_b = \| w \|_{b'}$ は一般には成り立たない．そのため，文字列が何進法で表記されているのかという情報は重要である．

上記の定義を用いて，剰余オートマトンを以下のように定義する．

定義 3　$b \geqq 2,\ k \geqq 2,\ k-1 \geqq r \geqq 0$ となる自然数 b, k, r によって一意に定義される以下の決定性有限オートマトン $M_b^k(r) = (Q, \Sigma, \delta, 0, \{r\})$ を剰余オートマトンとよぶ．ただし，Q は自然数からなる集合とみなし．% は剰余演算である．

$$Q = \{0, 1, 2, \cdot, k-1\}$$
$$\delta(q, a) = (q \times b + \| a \|_b) \% k \qquad (\forall i \in Q,\ \forall a \in \Sigma)$$

この構成法で与えられる DFA は，$r = 0$ の時に限って，空語 ε も受理してしまう．そのため，厳密には $r = 0$ の場合には状態 0 と全く同じように遷移する非最終である初期状態 q_0 を別に用意する必要がある．しかし，議論が複雑となるため, 以下では $L(M_b^k(r)) \subseteq \Sigma^+$ と考える．

以下で述べる命題は，任意の剰余オートマトン $M_b^k(r) = (Q, \Sigma, \delta, 0, \{r\})$ に対して成り立つ命題である．従って，命題中の b, k, r の記号は $M_b^k(r)$ の b, k, r を意味し，Q は $M_b^k(r)$ の状態集合を意味する．以下の 2 つは，剰余オートマトン $M_b^k(r)$ に関する命題のなかで，興味深い性質が分かるものである．

補題 1　任意の $i \in Q$，任意の $w \in \Sigma^+$ に対して，$\delta^*(i, w) = (i \times b^{|w|} + \| w \|_b) \ \%k$ が成り立つ．

証明　w の長さに関する帰納法で証明できる　□

補題 2　$\forall i, \forall j \in Q$ に対して，$\exists w \in \Sigma^*$ s.t. $\delta^*(i, w) = j$ かつ $|w| = \lceil \log_b(k) \rceil$

証明　$m = \lceil \log_b(k) \rceil$ とする．このとき，$|w| = m$ を満たす $w \in \Sigma^*$ に対して，$\delta^*(i, w) = (i \times b^m + \| w \|_b) \% k$ が補題 1 より成り立つ．b 進法表記数において，$0 \sim k-1$ までの値は明らかに m 桁で表現できるので，w は $|w| = m$ のまま，$0 \sim k-1$ までの値を取ることができる．よって，$\delta^*(i, w)$ は明らかに Q への全射となる．つまり，i から Q の任意の状態へ遷移可能な長さ m の文字列 w が存在する．□

上記の補題 2 は，任意の状態 i から任意の状態 j へ $\lceil \log_b(k) \rceil$ 文字で遷移できることを意味する．例として，$M_2^8(1)$ を考えてみる．$\lceil \log_2(8) \rceil = 3$ より，3 文字で任意の状態から任意の状態へ遷移ができる．例えば，$i = 1 \in Q$ とした場合に $j \in Q$ に対する $w \in \Sigma^*$ は，それぞれ表 1 のようになる．

表 1: $M_2^s(1)$ において，状態 1 から状態 j へ遷移を行うための $w \in \Sigma^*$ ($|w| = 3$)

j	w
0	000
1	001
2	010
3	011
4	100
5	101
5	110
7	111

補題 3 $\forall i, \forall j \in Q$ について，$\exists w \in \Sigma^*$ s.t. $(\delta^*(i,w) = \delta^*(j,w) \Rightarrow \forall w' \in \Sigma^* (|w'| \geqq |w| \Rightarrow \delta^*(i,w') = \delta^*(j,w')))$

証明 前提より，$i \times b^{|w|} + \parallel w \parallel_b \equiv j \times b^{|w|} + \parallel w \parallel_b \pmod{k}$ が成り立つ．よって，合同式の性質 [9] を用いれば以下が成り立つ．

$$
\begin{aligned}
& i \times b^{|w|} + \parallel w \parallel_b \equiv j \times b^{|w|} + \parallel w \parallel_b \pmod{k} \\
& \Rightarrow i \times b^{|w|} \equiv j \times b^{|w|} \pmod{k} \\
& \Rightarrow (i \times b^{|w|}) \times b^{|w'|-|w|} \equiv (j \times b^{|w|}) \times b^{|w'|-|w|} \pmod{k} \\
& \Rightarrow i \times b^{|w'|} + \parallel w' \parallel_b \equiv j \times b^{|w'|} + \parallel w' \parallel_b \pmod{k} \\
& \Rightarrow (i \times b^{|w'|} + \parallel w' \parallel_b) \% k = (j \times b^{|w'|} + \parallel w' \parallel_b) \% k \\
& \Rightarrow \delta^*(i,w') = \delta^*(j,w') \quad \square
\end{aligned}
$$

定理 4 $\forall i, \forall j \in Q$ について，$i \underset{Q}{\equiv} j \Leftrightarrow (w$ が $\delta^*(i,w) = r$ を満たす最短の文字列 $\leftrightarrow w$ が $\delta^*(j,w) = r$ を満たす最短の文字列)

証明 $\Rightarrow$ は，$\underset{Q}{\equiv}$ の定義より明らか．よって，$\Leftarrow$ のみ示す．
$\Leftarrow$) 補題 2 より，任意の状態から最終状態まで遷移する文字列は必ず存在するので，その最短長を m とする．$w \in \Sigma^*$ が $|w| < m$ のとき，$\delta^*(i,w) \notin F$ かつ $\delta^*(j,w) \notin F$ であるから，$\delta^*(i,w) \in F \Leftrightarrow \delta^*(j,w) \in F$ が成り立つ．$|w| \geqq m$ のとき，補題 3 より $\delta^*(i,w) = \delta^*(j,w)$ も成り立つ．よって，$\delta^*(i,w) \in F \Leftrightarrow \delta^*(j,w) \in F$ も明らかに成りたつ． $\square$

最小状態の $M_b^k(r)$ を $minM_b^k(r)$ と表し，$|minM_b^k(r)|$ でその状態数を表すことにする．$k = 2$ の場合は，明らかに $|minM_b^2(r)| = 2$ となる．以下の命題は，$|minM_b^k(r)|$ の上限や $|minM_b^k(r)|$ の値が求まる場合について述べた命題である．補題 5，定理 6 の証明は文献 [7] を参照されたい．定理 7 の証明については，補題 5 の証明で用いた $\sim$ 関係が，$b \geqq k$ ならば，$\forall i, \forall j \in Q - \{r\}$ に対して，$i \sim j \Leftrightarrow i \underset{Q}{\equiv} j$ となることから簡単に示せる．

補題 5 $|minM_b^k(r)| \leqq (k/\gcd(b,k)) + 1$.

定理 6 $k \geqq 3$ とする．このとき，$\gcd(b,k) = 1 \Leftrightarrow |minM_b^k(r)| = k$

定理 7 $b \geqq k$ ならば，$|minM_b^k(r)| = (k/\gcd(b,k)) + 1$. ただし，$\gcd(b,k) = 1$ のときは，$|minM_b^k(r)| = k$

以上より，$|minM_b^k(r)|$ の値は，$b \geqq k$ の場合，$\gcd(b,k) = 1$ の場合については求めることができた．$b < k$ かつ $\gcd(b,k) \neq 1$ の場合については，今後の課題としたい．

5. おわりに

有限オートマトン学習支援システムを開発したことによって，剰余オートマトンのような興味深い対象についての考察を行うことができた．また，有限オートマトン学習支援システムは，学習補助教材として，Aho-Corasick 文字列照合法の学習支援ツールは，演習のためのシミュレータとして，実際に教育の場で用いられている．以上のような実績から，開発した一連のシステムは，情報科学の分野に貢献することができたと考えている．

剰余オートマトンに関する考察においては，剰余オートマトンの特徴的な性質を示し，特定の場合においては最小状態数を求めることができた．また，今回は紙面の都合上割愛したが，剰余オートマトンに対して，通常の状態最小化アルゴリズムよりも効率的に状態最小化を行うアルゴリズムの提案も行った．よって今後は，$b < k$ かつ $\gcd(b,k) > 1$ の場合における状態数を考察することや，提案したアルゴリズムの正当性の証明，通常の状態最小化アルゴリズムとの計算量の比較などを行っていきたい．

参考文献

[1] 有川節夫（監修），西野哲朗，石坂裕毅，形式言語の理論，丸善，1999.
[2] 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(1987)，pp.8-23
[3] きしだなおき，創る Java, 改訂第 3 版第 1 刷, 毎日コミュニケーションズ，2009.
[4] 大村忠史, Java GUI プログラミング〈Vol.1〉さらにパワーアップした Swing, 初版第 1 刷, カットシステム，2002.
[5] 早乙女健治, JavaCC–コンパイラ・コンパイラ for Java, 初版第 1 刷, テクノプレス，2003.
[6] 「JUNG2.0 チュートリアル」 http://www.cs.tsukuba.ac.jp/ misue/open/tutorial/jung2/
[7] 鎌田晋伍，石坂裕毅，平田耕一：進法表記数の剰余類を受理する DFA とその最小化について，人工知能基本問題研究会 第 91 回，2013.
[8] Boris Alexeev: Minimal DFA for testing divisibility, Journal of COMPUTER and SYSTEM SCIENCE, 2004.
[9] 楫 元: 工科系のための初等整数論入門—公開鍵暗号をめざして, 培風館，2000.