2023 年 10 月 20 日

「オートマトンと形式言語」補足資料 (3)

本資料では，発展編として DFA に関する 2 つの話題を取り上げる．1 つは計算論 (教科書の続巻 [2] の主な内容) の基本的なトピックの 1 つである決定可能性についての話であり，1 節と 2 節で取り上げる．3 節と 4 節では，DFA の状態最小化について取り上げる．有限オートマトンを構成する演習問題で，自分の答えが解答例と異なっていて，正しいかどうか困ったことはないだろうか．DFA の場合，同じ役割をする状態を 1 つにまとめる (状態最小化) ことによって，ある意味 1 つの DFA が得られる．状態最小化のやり方がわかれば，解答例と違う答えであっても，状態最小化によって解答例と同じ DFA が得られるかどうかで，正しい解答だったかを確かめることが自分でできるだろう．

1 決定可能性

入力に対して YES または NO で答える (つまり，1 ビットで答える) 問題を，**判定問題**あるいは**決定問題** (decision problem) という．例えば，「命題論理式 A の入力に対して，A がトートロジーか?」や「あるアルファベット上の文字列 w と DFA M の入力に対して，$w \in L(M)$ か?」などは，判定問題の例である．(当然ながら，「人類は永遠に存続するのか?」のような漠然とした問いではなく，これらの例のように，理論的に形式化されている問題のみを対象としている．)

ある判定問題が，**決定可能** (decidable) とは，YES/NO を答えるプログラムが存在するときをいう．十分に強力なプログラミング言語を使えば，決定可能性は，用いるプログラミング言語に依らない．そこで，より一般的に，YES/NO を答える計算手続きがあるとき，と言い換えることもできる．判定問題に YES/NO の解答を与える計算手続きを，その問題の**決定手続き**という．

決定可能な問題 (**決定可能問題**とよばれる) は直観が働きやすいと思うが，逆に，決定可能でないような判定問題 (**決定不能問題**とよばれる) というのはどういうものだろうか．判定問題が決定可能でないということは，YES/NO を答えるプログラムが原理的に存在しないということになる．そんなことがわかるのだろうか．実は，決定可能でない判定問題は，**決定手続きが原理的に存在しえない，ということを証明する**ことによって得られる．難しそうだと感じるかもしれないが，典型的な証明方法がいくつか知られており，よく知られている決定不能問題はほとんどそれらの方法で証明されている．

実は，決定可能でないような判定問題は沢山ある．例えば，プログラムを入力として，そのプログラムが止まるかを答えるという問題は，決定可能でないことが知られている．(直観的な説明としては，プログラムを実行して止まるときは YES で答えればよいが，止まらないときはいつまで待てばよいのかわからない，ということを考えると，NO を答えるのは難しいことがわかるだろう．)

決定可能性については，ここでは，これ以上立ち入らない．決定可能性，および，もっと一般的に計算可能性 (プログラムで計算できることとはどういうことか)，というのは，教科書の続巻 (第 2 巻) で扱われる話題．

ただ，正規言語に関するどのような問題が決定可能かという話は，オートマトンの理論において重要な話題の 1 つなので，講義では，ここで取り挙げる．なお，教科書の続巻 (第 2 巻) では，4.1 節で扱われている話題になる．

2　正規言語に関する判定問題

文字列の集合 A が有限集合の場合には，要素を全部記述すればよいので，(メモリやディスクが十分沢山あるとして)A を記述するのに困難はないだろう．一方，無限集合の場合は，まず一般的には，A を記述するのは困難そうに思える．しかし，A が正規言語の場合には，DFA や NFA を使って記述できる．集合を記述できるなら，さらに，これらの集合に関する計算を，有限オートマトンを使って出来れば無限集合を扱うようなさまざま応用の可能性が広がる．では，正規言語に関するどのような問題なら，有限オートマトンを使って，計算可能なのだろうか．

なお，以下では，DFA と NFA は互いに等価なものに変換できるので，DFA のみを主に対象として考える．

次のような幾つかの重要で有用な判定問題が，決定可能．

1. **受理問題**

   この問題は，有限オートマトン M と文字列 w が与えられたときに，$w \in L(M)$ かを判定する．受理問題は決定可能である．つまり，以下のようなプログラムが実現可能:

プログラムの入力:	DFA M	
	M のアルファベット上の文字列 w	
プログラムの出力:	`YES`	($w \in L(M)$ のとき)
	`NO`	($w \notin L(M)$ のとき)

   手続きのスケッチ: 開始状態から，入力 w に従って M の状態遷移図を辿っていき，入力が終わったときに受理状態にいるかをチェックすればよい．

2. **空性判定問題**

   この問題では，有限オートマトン M が与えられたときに，$L(M)$ が空集合かを判定する．つまり，以下のようなプログラムが実現可能:

プログラムの入力:	DFA M	
プログラムの出力:	`YES`	($L(M) = \emptyset$ のとき)
	`NO`	($L(M) = \emptyset$ でないとき)

   手続きのスケッチ: M の状態遷移図を使って，開始状態から辿りつける状態の集合を構成し，辿りつける状態の集合のなかに受理状態があるかをチェックすればよい．

3. **包含性判定問題**

この問題では，共通のアルファベットをもつ有限オートマトン M_1 と M_2 が与えられたとき， $L(M_1) \subseteq L(M_2)$ かを判定する．つまり，以下のようなプログラムが実現可能:

プログラムの入力:	2つの共通のアルファベットをもつ DFA M_1,M_2
プログラムの出力:	`YES` $(L(M_1) \subseteq L(M_2)$ のとき$)$
	`NO` $(L(M_1) \subseteq L(M_2)$ でないとき$)$

手続き: $A \subseteq B \iff A \cap \overline{B} = \emptyset$，という性質を使う．

Step 1. $L(M_2)$ の補集合を認識する DFAM_2' を構成する．

Step 2. $L(M_2') \cap L(M_1)$ を認識する DFAM を構成する．

Step 3. 空性判定問題の決定手続きを使って，$L(M) = \emptyset$ かを判定する．
$L(M) = \emptyset$ なら `YES`. $L(M) \neq \emptyset$ なら `NO`.

4. **等価性判定問題**

この問題では，有限オートマトン M_1 と M_2 が与えられたとき， $L(M_1) = L(M_2)$ かを判定する．つまり，以下のようなプログラムが実現可能:

プログラムの入力:	2つの共通のアルファベットをもつ DFA $M1_1$,M_2
プログラムの出力:	`YES` $(L(M_1) = L(M_2)$ のとき$)$
	`NO` $(L(M_1) = L(M_2)$ でないとき$)$

手続きのスケッチ: 集合についての，$A = B \iff A \subseteq B \wedge B \subseteq A$，という性質を使えば，包含性判定問題に帰着できる．

以上，有限オートマトンに関する決定可能問題を 4 つ紹介した．

上の手続きからわかるように，包含性検査や等価性判定問題の手続きの鍵となるのは，**補集合や積集合を認識する有限オートマトンを構成する手続き (アルゴリズム) があること**である．実際，正規言語のクラスが補集合演算や積集合演算に関して閉じていることの証明に使った DFA の作り方は，“構成的” であり，プログラムで実現できる．

3 DFA の状態最小化

これまでの有限オートマトンの説明や演習問題を解いてきて気がついたかもしれないが，初期状態から到達できないような状態を削除しても認識する言語は変わらないし，また，同じ “役割” をしている 2 つの状態を 1 つにまとめてしまっても認識する言語は変わらない．演習問題の解答で別解を作った経験のある人もいるかと思うが，それも冗長な状態を 1 つにまとめると解答例が得られるはずである．DFA が与えられたときに，冗長な状態を消して状態数を最小にすることが出来る．このような操作を状態最小化とよぶ．驚くことに，正規言語それぞれにおい

て，状態数が最小の DFA の状態遷移図は，グラフ同型のもとで一意に定まる (つまり，同じグラフの形になる).

教科書では状態最小化の話は取りあげていないが，DFA についての理解を深めるために非常に良い話題と思うので，Kozen の教科書 [1]13,14 章から，DFA の状態最小化について紹介する.

状態を最小化する手順は以下の通り：

Step 1. 初期状態から到達できない状態を削除する.

Step 2. “等価な” 状態を 1 つにまとめる.

Step 1 については自明なので省略する．以下では，Step 2 についての詳細を与える．“等価な” 状態とは何なのか，それらを 1 つにまとめるとはどうことなのか，についてのきちんとした定義を与える.

定義 1 $M = (Q, \Sigma, \delta, q_0, F)$ を DFA とし，$p, q \in Q$ とする．このとき p, q が等価 $(p \approx q)$ であるとは，以下の性質が成立することをいう：

$$\forall w \in \Sigma^*.\ \hat{\delta}(p, w) \in F \iff \hat{\delta}(q, w) \in F$$

容易に確かめられるように，$\approx$ は Q 上の等価関係である．DFA $M/\approx$ (これを商オートマトンとよぶ) を以下のように定めることができる[1]．なお，q の等価クラス $\{p \in Q \mid p \approx q\}$ を $[q]$ と記す.

$M/\approx = (Q', \Sigma, \delta', q_0', F')$，ただし，

$$\begin{array}{lcl} Q' & = & \{[q] \mid q \in Q\} \\ \delta'([p], x) & = & [\delta(p, x)] \\ q_0' & = & [q_0] \\ F' & = & \{[q] \mid q \in F\} \end{array}$$

ここで，関数 δ' の定義が well-defined であることは，次の補題から成立する.

補題 2 $p \approx q$ ならば，任意の $x \in \Sigma$ について，$\delta(p, x) \approx \delta(q, x)$.

証明. $p \approx q$，$x \in \Sigma$ とする．このとき，任意の $w \in \Sigma^*$ について，

$$\begin{array}{lcll} \hat{\delta}(\delta(p, x), w) \in F & \iff & \hat{\delta}(p, xw) \in F & (\hat{\delta} \text{ の定義より}) \\ & \iff & \hat{\delta}(q, xw) \in F & (\text{仮定 } p \approx q \text{ より}) \\ & \iff & \hat{\delta}(\delta(q, x), w) \in F & (\hat{\delta} \text{ の定義より}) \end{array}$$

□

次に，M と $M/\approx$ の等価性を証明するための補題を 2 つ示す.

補題 3 $q \in F$ であるとき，そのときに限り，$[q] \in F'$.

証明. $(\Longrightarrow)$ は F' の定義より自明．$(\Longleftarrow)$ $\{q_1, \ldots, q_n\} \in F'$ とする．このとき，任意の $1 \leq i \leq n$ について，$q_i \in F$ を示す．F' の定義より，ある j について，$q_j \in F$ かつ $\{q_1, \ldots, q_n\} = [q_j]$．このとき，$q_i \approx q_j$ であることから，$\forall w \in \Sigma^*.\ \hat{\delta}(q_i, w) \in F \iff \hat{\delta}(q_j, w) \in F$．ここで，$w = \varepsilon$ ととると，$q_i = \delta(q_i, \varepsilon) \in F \iff q_j = \delta(q_j, \varepsilon) \in F$ が得られる． □

[1] このあたりは，離散数学の範疇であり，すでに学習済みのはずなので，簡潔に記す.

補題 4 任意の $w \in \Sigma^*$ について，任意の $p \in Q$ に対して，$\widehat{\delta'}([p], w) = [\hat{\delta}(p, w)]$.

証明. w の長さに関する帰納法で示す.

- 基本ステップ. $\widehat{\delta'}([p], \varepsilon) = [p] = [\hat{\delta}(p, \varepsilon)]$ より成立.
- 帰納ステップ. $w = xw'$ とおく.

$$\begin{aligned}
\widehat{\delta'}([p], xw') &= \widehat{\delta'}(\delta'([p], x), w') && (\widehat{\delta'} \text{ の定義より}) \\
&= \widehat{\delta'}([\delta'(p, x)], w') && (\delta' \text{ の定義より}) \\
&= [\hat{\delta}(\delta(p, x), w')] && (\text{帰納法の仮定より}) \\
&= [\hat{\delta}(p, xw')] && (\hat{\delta} \text{ の定義より})
\end{aligned}$$

よって，成立. □

定理 5 M を DFA とするとき，$L(M) = L(M/\approx)$.

証明.

$$\begin{aligned}
w \in L(M) &\iff \hat{\delta}(q_0, w) \in F && (L(M) \text{ の定義より}) \\
&\iff [\hat{\delta}(q_0, w)] \in F' && (\text{補題 3 より}) \\
&\iff \widehat{\delta'}([q_0], w)] \in F' && (\text{補題 4 より}) \\
&\iff \widehat{\delta'}(q'_0, w)] \in F' && (q'_0 \text{ の定義より}) \\
&\iff w \in L(M/\approx) && (L(M/\approx) \text{ の定義より})
\end{aligned}$$

□

4 DFA の状態最小化手続き

これまで見てきた他の構成方法と違って，状態最小化は，そのままでは，どのように $\approx$ や $M/\approx$ を計算できるのか自明ではない．ここでは，$M/\approx$ の構成手続きを紹介する.

定義 6 $M = (Q, \Sigma, \delta, q_0, F)$ を DFA とする．このとき，相異なる 2 個の状態の集合を要素とする D を，以下の条件を満たす最小の集合と定義する：

(1) $p \in F$ かつ $q \notin F$ ならば，$\{p, q\} \in D$

(2) 任意の $x \in \Sigma$ と相異なる任意の状態 $p, q \in Q$ について，$\{\delta(p, x), \delta(q, x))\} \in D$ ならば $\{p, q\} \in D$.

補題 7 $\{p, q\} \in D \iff p \not\approx q$.

証明. まず，$p \approx q \overset{\text{def}}{\iff} \forall w \in \Sigma^*.\ \hat{\delta}(p, w) \in F \iff \hat{\delta}(q, w) \in F$ であることに注意すると，$p \not\approx q$ となることの必要十分条件は，$\hat{\delta}(p, w) \in F$ かつ $\hat{\delta}(q, w) \notin F$，もしくは，$\hat{\delta}(p, w) \notin F$ かつ $\hat{\delta}(q, w) \in F$，となる $w \in \Sigma^*$ が存在することなのがわかる．
($\Longrightarrow$) D の構成に関する帰納法で示す.

1. 条件 (1) により，$\{p, q\} \in D$ となった場合．このとき，$p \in F$ かつ $q \notin F$ が成立する．いま，$p = \hat{\delta}(p, \varepsilon)$，$q = \hat{\delta}(q, \varepsilon)$ であるから，$\hat{\delta}(p, \varepsilon) \in F$ かつ $\hat{\delta}(q, \varepsilon) \notin F$ が成立する．よって，$p \not\approx q$.

2. 条件 (2) により，$\{p,q\} \in D$ となった場合．このとき，ある相異なる状態 $p',q' \in Q$ $\{p',q'\} \in D$ および $x \in \Sigma$ が存在して，$p = \delta(p',x)$，$q = \delta(q',x)$．帰納法の仮定より，$p' \not\approx q'$．よって，ある $w' \in \Sigma^*$ が存在して，$\hat{\delta}(p',w') \in F$ かつ $\hat{\delta}(q',w') \notin F$，もしくは，$\hat{\delta}(p',w') \notin F$ かつ $\hat{\delta}(q',w') \in F$ が成立する．一般性を失うことなく，$\hat{\delta}(p',w') \in F$ かつ $\hat{\delta}(q',w') \notin F$ とする．よって，$w = xw'$ を考えると，$\hat{\delta}(p,w) = \hat{\delta}(\delta(p,x),w') = \hat{\delta}(p',w') \in F$，かつ，$\hat{\delta}(q,w) = \hat{\delta}(\delta(q,x),w') = \hat{\delta}(q',w') \notin F$ が成立する．よって，$\{p,q\} \in D$.

($\Longleftarrow$) まず,

性質 (A)： 任意の $w \in \Sigma^*$ について，任意の相異なる状態 p,q について，
$\hat{\delta}(p,w) \in F$ かつ $\hat{\delta}(q,w) \notin F$ または $\hat{\delta}(p,w) \notin F$ かつ $\hat{\delta}(q,w) \in F$ ならば，
$\{p,q\} \in D$

が成立することを，w に関する帰納法で示す.

- 基本ステップ．$w = \varepsilon$ の場合，一般性を失うことなく，$\hat{\delta}(p,w) \in F$ かつ $\hat{\delta}(q,w) \notin F$ の場合を考える．このとき，$p = \hat{\delta}(p,\varepsilon)$，$q = \hat{\delta}(q,\varepsilon)$ であるから，$p \in F$ かつ $q \notin F$ となる．よって，条件 (1) より，$\{p,q\} \in D$

- 帰納ステップ．一般性を失うことなく，$\hat{\delta}(p,w) \in F$ かつ $\hat{\delta}(q,w) \notin F$ の場合を考える．このとき，$w = xw'$ $(x \in \Sigma, w' \in \Sigma^*)$ とおくことが出来る．$\hat{\delta}(p,xw) = \hat{\delta}(\delta(p,x),w)$，$\hat{\delta}(q,xw) = \hat{\delta}(\delta(q,x),w)$ であるから，$\hat{\delta}(\delta(p,x),w) \in F$ かつ $\hat{\delta}(\delta(q,x),w) \notin F$ となる．ここで，$\hat{\delta}$ は関数なので，$\hat{\delta}(\delta(p,x),w) \in F$ かつ $\hat{\delta}(\delta(q,x),w) \notin F$ であることから，$\delta(p,x) \neq \delta(q,x)$．よって，帰納法の仮定より，$\{\delta(p,x),\delta(q,x) \in D$．したがって，条件 (2) より，$\{p,q\} \in D$.

(性質 (A) の証明の終わり)

いま，$p \not\approx q$ と仮定する．このとき，ある $w \in \Sigma^*$ が存在して，$\hat{\delta}(p,w) \in F$ かつ $\hat{\delta}(q,w) \notin F$，もしくは，$\hat{\delta}(p,w) \notin F$ かつ $\hat{\delta}(q,w) \in F$．よって，性質 (A) より，$\{p,q\} \in D$ となる． □

補題 7 より，DFA $M = (Q,\Sigma,\delta,q_0,F)$ が与えられたとき，以下の手続きによって，集合 $\{\{p,q\} \mid p \not\approx q\}$ が計算できる :

Step 1. $D := \{\{p,q\} \mid p \in F, q \in Q\backslash F\}$ とおく．

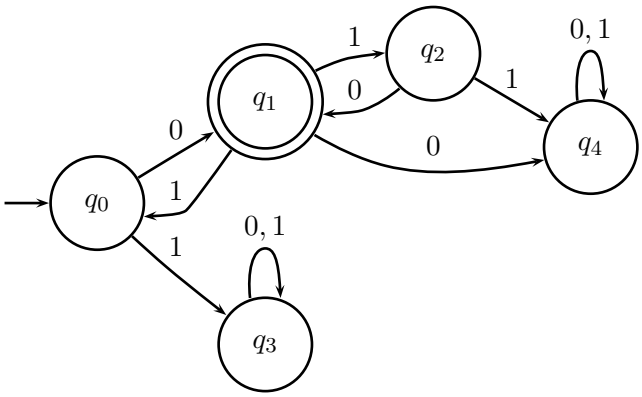
Step 2. 以下の D_{add} を計算する．

$$D_{add} = \bigcup_{x\in\Sigma,p,q\in Q} \{\{p,q\} \mid p \neq q, \{\delta(p,x),\delta(q,x)\} \in D\}$$

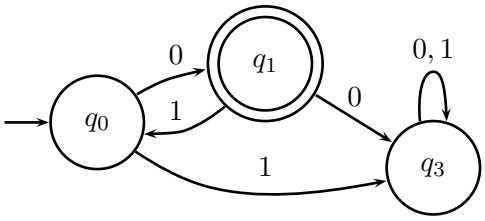
Step 3. $D_{add} \subseteq D$ なら終了．そうでなければ，$D = D \cup D_{add}$ と更新して，Step 1 に戻る．

集合 $\{\{p,q\} \mid p \not\approx q\}$ を使って，$p \approx q$ は判定できるので，あとは，$M/\approx$ の定義にしたがって，$M/\approx$ を構成すればよい．

例 8 以下の 2 つの DFA M_1 と M_2 を考えてみよう．
M_1:



M_2:



どちらも

$$\{w \in \{0,1\}^* \mid w = 0101\cdots 0\}$$

を認識する DFA となっている．ここで，M_1 の q_0 と q_2 を一緒にし，q_3 と q_4 を一緒にしたものが，M_2 になることが見てとれるだろうか? 実は，M_2 は，M_1 から状態最小化の手続きを使って得ることが出来る．

M_1 に状態最小化の手続きを適用するには，以下のような表を考えるとよい．q_i と q_j の交差している場所が q_i と q_j が同じ状態にはならないか (区別されるか) の印をつける場所になっている．初期状態は $-$ となっていて，まだ同じになるかどうか不明であることを表わす．区別されることがわかったら ✓ を記すこととする．

$$\begin{array}{cccccc} q_0 & & & & \\ - & q_1 & & & \\ - & - & q_2 & & \\ - & - & - & q_3 & \\ - & - & - & - & q_4 \end{array}$$

まず，状態 q_1 とその他の状態 q_i $(i = 0, 2, 3, 4)$ については，q_1 が受理状態でその他が受理状態でないので区別される (定義 6 の (1))．そこで，表を以下のように更新することが出来る．

$$\begin{array}{cccccc} q_0 & & & & \\ \checkmark & q_1 & & & \\ - & \checkmark & q_2 & & \\ - & \checkmark & - & q_3 & \\ - & \checkmark & - & - & q_4 \end{array}$$

次に，定義 6 の (2) を使って，✓ を増やせないか考えよう．定義 6 の (2) は，ある入力について，それぞれ，すでに区別されている状態に遷移するような 2 つの状態は，区別できるということをいっている．この表ではすでに幾つかの状態に

ついては区別されているので，その情報を利用することで，✓を増やすことが出来る．例えば，q_0 と q_3 については，入力 0 について，それぞれ，すでに区別されている q_1 と q_3 に遷移している．したがって，q_0 と q_3 は区別される．同様にして，q_2 と q_3，q_0 と q_4，q_2 と q_4 について，区別されることがわかる．このようにして表を更新すると，以下のようになる．

$$
\begin{array}{cccccc}
q_0 & & & & & \\
\checkmark & q_1 & & & & \\
- & \checkmark & q_2 & & & \\
\checkmark & \checkmark & \checkmark & q_3 & & \\
\checkmark & \checkmark & \checkmark & - & q_4 &
\end{array}
$$

最後に，q_0 と q_2，それから，q_3 と q_4 が区別されるかわからないまま残っている．ここで，q_0 と q_2 について見てみると，0 の場合は，両者とも q_1 に遷移している．1 の場合は，それぞれ q_3 と q_4 に遷移しているが，q_3 は q_4 は区別印がまだついていない．また，q_3 と q_4 については，0 でも 1 でも，それぞれ q_3 と q_4 に遷移しているが，q_3 は q_4 は区別印がまだついていない．ということで，✓はもう増やせないことがわかる．

このように，✓が増やせなくなったら，表は完成である．この表から，q_0 と q_2，それから，q_3 と q_4 は，同じ状態にまとめてよいことがわかり，M_2 の DFA が得られる．

参考文献

[1] D.C. Kozen. *Automata and Complexity.* Springer-Verlag, 1997.