正規言語 L を受理する最小の DFA を構成する

1 決定性オートマトン (Deterministic Automata)

決定性オートマトン DA は 5 つ組で与えられる: $M = (Q, \Sigma, \delta, q_{\text{init}}, F)$.

- Q は状態の集合であり、$q_{\text{init}} \in Q$ は初期状態で、$F \subseteq Q$ は受理状態
  - **注意: 一般化のために Q は無限集合でも良いとする**
  - Q が有限のとき、M を **DFA**（決定性有限オートマトン、Deterministic Finite Automata）と呼ぶ。
- Σ は入力アルファベット（有限集合）
- $\delta : Q \times \Sigma \to \Sigma$ は遷移関数
  - 状態 q で入力 $a \in \Sigma$ を読んだ時に移る先の状態が $\delta(q, a)$ である。

$q \xrightarrow{a} q'$ と書いた時には、状態 q で入力記号 $a \in \Sigma$ を読むと q' に移ることを意味する。ここで、$q' = \delta(q, a)$ である。DA の遷移関数を拡張し、ある状態から「文字列」を読んだ時の行き先を与える関数 $\hat{\delta}$ を、以下のように帰納的に定義する:

$$\begin{array}{lcll} \hat{\delta}(q, \epsilon) & = & q, & (\text{一文字も読まないので状態は変わらない}) \\ \hat{\delta}(q, a) & = & \delta(q, a), & \\ \hat{\delta}(q, aw) & = & \hat{\delta}(\delta(q, a), w). & (\text{ただし } a \in \Sigma, w \in \Sigma^*) \end{array}$$

$q \xrightarrow{a} q'$ は、一文字 $a \in \Sigma$ による遷移であったが、これを拡張して状態 q から「文字列」$w \in \Sigma^*$ を読んだ時に q' に移るなら $q \overset{w}{\Rightarrow} q'$ と書く。このとき $q' = \hat{\delta}(q, w)$ である。ここから先では、$\hat{\delta}$ のことも単に δ で書く。

命題 1. 任意の状態 q と文字列 $w_1, w_2 \in \Sigma^*$ について、$\delta(q, w_1 w_2) = \delta(\delta(q, w_1), w_2)$ が成立する。

Proof. w_1 の長さに関する帰納法で証明する。

まず $w_1 = \epsilon$ の場合であるが、$\delta(\delta(q, \epsilon), w_2) = \delta(q, w_2)$ なので、明らか。

次に $w_1 = \sigma w$ の場合であるが、$\delta(\delta(q, \sigma w), w_2) = \delta(\delta(\delta(q, \sigma), w), w_2)$ まで展開でき、帰納法の仮定から、$\delta(\delta(\delta(q, \sigma), w), w_2) = \delta(\delta(q, \sigma), w w_2)$ となり、再び帰納法の仮定を用いると、$\delta(\delta(q, \sigma), w w_2) = \delta(q, \sigma w w_2) = \delta(q, w_1 w_2)$ である。 □

（代数に詳しい人向けの説明になってしまいますが、この命題によって $\delta : Q \times \Sigma^* \to Q$ は free monoid Σ^* を monoid とする right monoid action になっていることが保証されます。）

DA M の受理する言語を定義する前に、$\tau : Q \times \Sigma^* \to \mathbb{B} = \{0, 1\}$ という、状態と文字列を受け取って、真偽値を返す関数を定義する:

$$\tau(q, w) = \begin{cases} 1 & \text{if } \delta(q, w) \in F, \\ 0 & \text{if } \delta(q, w) \notin F. \end{cases}$$

DA M の受理する言語 $L(M)$ は次で定義される:

$$L(M) = \{w \in \Sigma^* : \tau(q_{\text{init}}, w) = 1\} = \{w \in \Sigma^* : q_{\text{init}} \xrightarrow{w} q_f, q_f \in F\}.$$

1.1 決定性オートマトンの性質 1

決定性オートマトンの定める言語における簡単な性質を確認する。

DA $A = (Q_A, \Sigma, \delta_A, q^A_{\text{init}}, F_A)$ と $B = (Q_B, \Sigma, \delta_B, q^B_{\text{init}}, F_B)$ について、$L(A) \cap L(B) = L(C)$ とする DA C が存在する。俗に言う「直積構成」と呼ばれる方法を使えば良い:

$$C = (Q_A \times Q_B,\ \Sigma,\ \delta_{A\times B},\ (q^A_{\text{init}}, q^B_{\text{init}}),\ F_A \times F_B).$$

ここで、$\delta_{A\times B} : (Q_A \times Q_B) \times \Sigma \to (Q_A \times Q_B)$ は次で定義される遷移関数である:

$$\delta_{A\times B}((q_A, q_B), \sigma) = (\delta_A(q_A, \sigma), \delta_B(q_B, \sigma)).$$

単に二つのオートマトンを同時に実行するだけであるから、明らかに次が成立する:

命題 2. $L(A) \cap L(B) = L(C)$ である。

1.2 決定性オートマトンの性質 2

DA $A = (Q, \Sigma, \delta, q_{\text{init}}, F)$ が与えられた時に、A では受理できなかった語からなる言語（A の補言語 $\Sigma^* \setminus L(A)$) を受理する DA $\overline{A}$ が存在する。これは次のように定義すると良い:

$$\overline{A} = (Q, \Sigma, \delta, q_{\text{init}}, Q \setminus F).$$

このとき、任意の文字列について $\tau_A(q, w) \neq \tau_{\overline{A}}(q, w)$ であるから、次が成立する。

命題 3. $L(\overline{A}) = \Sigma^* \setminus L(A)$.

2 決定性有限オートマトン (Deterministic Finite Automata)

Σ 上の言語 $L \subseteq \Sigma^*$ は、それを受理する **DFA** が存在するときに、「正規言語」と呼ばれる。

2.1 DFA の空性判定問題

DFA $A = (Q, \Sigma, \delta, q_{\text{init}}, F)$ が与えられたときに、$L(A) = \emptyset$ となるかどうか、つまり A が文字列を何でも良いから受理できるかどうか、を判定する問題を、「空性判定問題」と呼ぶ。

素朴なアプローチとして、短い方から文字列を順に作り ($\Sigma = \{a, b\}$ なら $\epsilon, a, b, aa, ab, ba, bb, aaa, \ldots$)、$\tau_A(q_{\text{init}}, w) = 1$ となるものが現れたら、$L(A) \neq \emptyset$ とすることを思いつく。ただ、この方法では $L(A) = \emptyset$ であった場合に、どこで計算を打ち切って良いのか分からない。

より良い方法は、DFA を有限な「有向グラフ」としてみて、グラフの二点間の到達可能性を調べる問題に帰着することが考えられる。具体的にはノード q_{init} から、F に属するノードに到達できるかどうかという問題を考え、これは Dijkstra 法や Warshall-Floyd 法といったものを使えば良い（もっと素朴な方法でもできる）。

2.2 DFA の言語等価性判定問題

DFA 上で解ける大事な問題として、言語の等価性判定問題をみる。これは二つの DFA A, B が与えられた時に、$L(A) = L(B)$ かどうかを判定するような問題である。

$$L(A) = L(B) \quad \Longleftrightarrow \quad L(A) \setminus L(B) = \emptyset \ \wedge \ L(B) \setminus L(A) = \emptyset$$

が成立するから、二つの DFA の言語の差 $L(A) \setminus L(B)$ が空集合 $\emptyset$ かどうかが分かれば良い。

$L(A) \setminus L(B) = L(A) \cap L(\overline{B})$ に等しく、$L(C) = L(A) \cap L(\overline{B})$ とする DFA C が構成できるので、C に対して空性判定を行えば、言語の等価性判定問題が解ける。

空性判定問題 $L(A) =_? \emptyset$ は、等価性判定問題の特殊な場合として考えることができる。上の議論は、一般の等価性判定問題を、特殊な場合に帰着できることを述べている。

2.3 Trim DA

DA M が以下の条件を満たす時に、M を **trim** DA と呼ぶ:

$$\forall q \in Q.\ \exists w \in \Sigma^*.\ \delta(q_{\text{init}}, w) = q.$$

これは、どんな状態 q に対しても、初期状態 q_{init} から到達できる $q_{\text{init}} \xrightarrow{w} q$ ということを述べている。

初期状態から到達できない状態というのは、言語の受理に影響を与えないので、これを取り除いてしまっても受理する言語に影響しないことが直感的に分かる。このことは、次の命題でまとめられる。

命題 4. DA $M = (Q, \Sigma, \delta, q_{\text{init}}, F)$ を DA とし、q_{init} から到達できる状態全体の集合を $P \subseteq Q$ とする。この時、P でない状態（すなわち到達できない状態）を削ぎ落として得られる次の DA

$$N = (P,\ \Sigma,\ \delta_{\cap P},\ q_{\text{init}},\ F \cap P)$$

は $L(M) = L(N)$ を満たす。また、N は trim DA である。ただし、$\delta_{\cap P} : P \times \Sigma \to P$ は $\delta_{\cap P}(\sigma) = \delta(\sigma)$ として δ の定義域を $P \times \Sigma$ に制限して得られる遷移関数である。

Proof. 証明は練習問題として、以下の二つを示してください:

- $L(M) = L(N)$
- N は trim DA

□

DFA M については、q_{init} から到達できる状態全体の集合を求めることは難しくないので（Warshall-Floyd 法なども使える)、到達可能な集合全体 P を計算できる。この時、上の構成で、$L(M) = L(N)$ とする trim DFA N を作ることができる。この考察に基づき、DFA は全て trim DFA で与えられると思うことにする。

3 同値関係と同値類

少しだけオートマトンの話を離れて、二項関係の話をする。

集合 X 上の二項関係 $\approx \subseteq X \times X$ は以下三つの条件を満たすとき、「同値関係」と呼ばれる:

反射律 任意の $x \in X$ について、$x \approx x$ が成立する。

対称律 任意の $x, y \in X$ について、$x \approx y$ ならば $y \approx x$ でもある。

推移律 任意の $x, y, z \in X$ について、$x \approx y$ かつ $y \approx z$ ならば $x \approx z$ でもある。

集合 X とその上の同値関係 $\approx \subseteq X \times X$ が与えられた時に、$x \in X$ の同値類 $[x]_{\approx}$ を次で定義する:

$$[x]_{\approx} = \{y : x \approx y\}.$$

文脈から $\approx$ が明らかな場合は、$[x]_{\approx}$ のことを単に $[x]$ で済ませます。

命題 5. 同値関係 $\approx$ については次が成立する:

$$x \approx y \iff [x] = [y].$$

Proof. まず $x \approx y$ を仮定し、$[x] \subseteq [y]$ を証明する。$z \in [x]$ ということはすなわち、$x \approx z$ ということになる。$\approx$ は同値関係であり対称律を満たすので $y \approx x$ でもあり、推移律も満たすから $y \approx x \wedge x \approx z$ が $y \approx z$ を導く。よって $[x] \subseteq [y]$ が分かった。$[y] \subseteq [x]$ も全く同様に証明できる。

次に $[x] = [y]$ を仮定し、$x \approx y$ を証明する。$\approx$ は反射律を満たすため $y \in [y]$ である。従って、$[y] \subseteq [x]$ により $y \in [x]$ が成立し、$x \approx y$ が得られた。 □

X を $\approx$ で割って得られる集合を「商集合」と呼び、これは $X/_{\approx}$ で表され次で定義される:

$$X/_{\approx} \triangleq \{[x] : x \in X\}.$$

4 決定性オートマトンにおける重要な同値関係

- DA $M = (Q, \Sigma, \delta, q_{\mathrm{init}}, F)$ を一つ固定する。
- 二つの状態 $p, q \in Q$ が M で「区別できない」ことを次で定義する:

$$\forall w \in \Sigma^*.\ \tau(p, w) = \tau(q, w).$$

- 二つの状態 $p, q \in Q$ が区別できないならば、$p \sim_M q$ と書くことにする。
  - M が文脈から明らかな場合は、$p \sim_M q$ ではなく単に $p \sim q$ と書きます。
- $p \not\sim q$ であるとき、ある文字列 $w \in \Sigma^*$ で、$\tau(p, w) \neq \tau(q, w)$ となるが、このとき w によって p と q が区別される、という。

命題 6. $\sim$ は Q 上の同値関係である。

Proof. 練習問題 □

命題 7. $p \sim q$ であるとき、任意の文字列 v で動いた先も区別できない: $\delta(p,v) \sim \delta(q,v)$.

Proof. どんな $w \in \Sigma^*$ についても、$\tau(\delta(p,v),w) = \tau(\delta(q,v),w)$ を示す。$\delta(\delta(p,v),w) = \delta(p,vw)$ かつ $\delta(\delta(q,v),w) = \delta(q,vw)$ であり、p と q は区別できない、すなわち $\tau(p,vw) = \tau(q,vw)$ である。従って、$\tau(\delta(p,v),w) = \tau(\delta(q,v),w)$ である。 □

5 DA を割って DA を作る

M を $\sim_M$ で割って得られる DA $M/\sim$ を次で定義する:

$$M/\sim = (Q/\sim,\ \Sigma,\ \delta/\sim,\ [q_{\text{init}}]\sim,\ F/\sim).$$

ただし、$X \subseteq Q$ に対する商集合 $X/\sim$ は次で定義されていたことを思い出されたい:

$$X/\sim = \{[x]\sim : x \in X\}.$$

また、遷移関数 $\delta : Q \times \Sigma \to Q$ に対する $\delta/\sim$ は次で定義される:

$$\delta/\sim([q],a) = [\delta(q,a)]\sim.$$

代表元を用いる構成で良くある注意点として、関数が well-defined かどうか、ということがある。この場合は、$p \sim q$ ならば、$\delta/\sim([p],a) = \delta/\sim([q],a)$ を確認する必要がある。すなわち、$\delta(p,a) \sim \delta(q,a)$ であって欲しいが、これは先に示した命題によって保証されている。

補題 1. DA M を $\sim_M$ で割って得られる DA は、同じ言語を受理する $L(M) = L(M/\sim)$.

Proof. 次のことだけが分かっていれば良い:

$$\delta/\sim([q],w) = [\delta(q,w)].$$

これを帰納法で証明する:

- $w = \epsilon$ の場合は、$\delta(q,\epsilon) = q$ で (これはそう定義したことを思い出して欲しい)、同様に $\delta/\sim([q],\epsilon) = [q]$ であるから、明らか。
- $w = av$ $(a \in \Sigma, v \in \Sigma^*)$ の場合は

$$\delta/\sim([q],av) = \delta/\sim([\delta(q,a)],v) =_{\text{帰納法の仮定}} [\delta(\delta(q,a),v)] = [\delta(q,av)] = [\delta(q,w)].$$

□

特に trim DA を割った場合に得られる DA は、trim DA になっている。

補題 2. trim DA M を割って得られる $M/\sim$ は trim DA である。

命題 8. L を正規言語とし、L を受理する DFA M を固定する。このとき、$|\Sigma^*/\sim| \leq |M|$ が成立する。

Proof. 背理法で導く。$|M| < |\Sigma^*/\sim|$ だとすると、鳩ノ巣原理により、以下を満たす文字列 $\alpha, \beta \in \Sigma^*$ が存在する:

$$\delta_M(q_{\text{init}}, \alpha) = \delta_M(q_{\text{init}}, \beta), \qquad \alpha \not\sim \beta.$$

$\alpha \not\sim \beta$ ということは、何らかの文字列 $w \in \Sigma^*$ によって $\alpha w \in L \iff \beta w \notin L$ として α と β が区別されなければならない。一方で $\delta_M(q_{\text{init}}, \alpha w) = \delta_M(q_{\text{init}}, \beta w)$ であり、M は L を受理するので $\alpha w \in L \iff \beta w \in L$ となる。よって矛盾する。 □

6 言語上の右合同関係

有限アルファベット Σ 上の言語 $L \subseteq \Sigma^*$ が与えられた時に、これを受理する DA M_L を以下のようにして作ることができる:

$$M_L = (\Sigma^*,\ \Sigma,\ \delta,\ \epsilon,\ L)$$

ただし、$\delta : \Sigma^* \times \Sigma \to \Sigma^*$ は $\delta(w, a) = wa$ として文字列と文字の結合で定義される。この無限状態オートマトンは、状態として文字列を考えていて、受理状態として L そのものを取っている。$\delta(v, w) = vw$ が成立することから、$L = L(M)$ は明らかである。

さて、M_L は DA であるから、$\sim_{M_L}$ を考えることができ、以下が成立する:

$$w \sim_{M_L} w' \iff \forall v \in \Sigma^*.\ wv \in L \iff w'v \in L.$$

$\sim_{M_L}$ のことを、$\cong_L$ や $\cong$ と書くことにする。上の命題により、$w \cong w'$ であれば、任意の文字列 v について $wv \cong w'v$ であることが分かる。このようにして、右に新しく文字列を持ってきても $\cong$ が破れないことから、$\cong$ を「右合同関係」と呼ぶ。

$w \cong_L w' \iff \forall v \in \Sigma^*.\ wv \in L \iff w'v \in L$ は、結果的には DA を考えずに、言語 L 単体で定義できることに注意されたい。従って、$\cong_L$ は、L の右合同関係と呼べる。この時、次の定理が成立する。

定理 1. 言語 L について $\Sigma^*/\cong_L$ が有限集合になるならば、L は正規言語である。

Proof. 上で見たように、L から DA $M_L = (\Sigma^*, \Sigma, \delta, \epsilon, L)$ を定めることができる。M_L を $\sim_{M_L}$ で割ると DA $M_L/\sim = (\Sigma^*/\sim, \Sigma, \delta/\sim, [\epsilon], L/\sim)$ が得られる。さて、$\Sigma^*/\cong_L = \Sigma^*/\sim$ であるから ($\cong_L$ は $\sim_{M_L}$ そのものである)、$M_L/\sim$ の状態数は有限となる。$L(M_L/\sim) = L$ であり、L は DFA で受理されるから正規言語である。 □

定理 1 の逆も成立するが、それを証明する前に、次の強力な補題を証明する。

補題 3. L を正規言語とし、L を受理する DFA N をとる。このとき、$L = L(M)$ とする **trim** DA M について、$M/\sim_M$ は DFA となり、しかも $|M/\sim_M| \leq |N|$ が成立する。

Proof. 背理法で導く。$M/\sim_M$ の状態数が無限であったり、仮に有限でも $|N| < |M/\sim_M|$ だとすると、鳩ノ巣原理により、以下を満たす文字列 $\alpha, \beta \in \Sigma^*$ が存在する:

$$\delta_N(q_{\text{init}}, \alpha) = \delta_N(q_{\text{init}}, \beta), \qquad \delta_{M/\sim}(q_{\text{init}}, \alpha) \neq \delta_{M/\sim}(q_{\text{init}}, \beta).$$

このことは

- M が trim DA なので $M/\sim_M$ も trim DA であることと

- $M/_{\sim_M}$ が trim DA であるから、$M/_{\sim_M}$ の各状態 q_i に到達させる文字列 w_i が存在する。
- N の状態数は $M/_{\sim_M}$ の状態数より少ないので、何らかの w_i と w_j で $\delta_N(q_{\text{init}}, w_i) = \delta_N(q_{\text{init}}, w_j)$ でなければならない。

$\delta_{M/\sim}(q_{\text{init}}, \alpha) \neq \delta_{M/\sim}(q_{\text{init}}, \beta)$ ということは、$\delta_M(q_{\text{init}}, \alpha) \not\sim_M \delta_M(q_{\text{init}}, \beta)$ であるので、何らかの文字列 $w \in \Sigma^*$ によって $\alpha w \in L \iff \beta w \notin L$ となってしまう。一方で $\delta_N(q_{\text{init}}, \alpha w) = \delta_N(q_{\text{init}}, \beta w)$ であり、$\alpha w \in L \iff \beta w \in L$ となるから矛盾する。 □

定理 2. 言語 L が正規言語であるならば、$\Sigma^*/_{\cong_L}$ は有限集合になる。

Proof. L は正規言語であるから、$L(N) = L$ とする DFA N が存在する。一方で、L から定まる DA M_L は trim DA である。$M_L/_{\sim_{M_L}}$ の状態集合は $\Sigma^*/_{\sim_{M_L}} = \Sigma^*/_{\cong_L}$ である。さて、上の補題により、$\Sigma^*/_{\cong_L}$ は有限集合でなければならない。 □

7 正規言語 L を受理する状態数最小の DFA を構成する

補題 3 のステートメントは次の通りであった:

> L を正規言語とし、L を受理する DFA N をとる。このとき、$L = L(M)$ とする **trim** DA M について、$M/_{\sim_M}$ は DFA となり、しかも $|M/_{\sim_M}| \leq |N|$ が成立する。

これは、L を受理する「どんな」DFA N を取ってきたとしても、割って得られる DFA の方が「状態数が少ない」ということを言っている。

補題 3 は、正規言語 L を受理する状態数最小の DFA を作る方法を述べている。つまり、正規言語 L を受理する DFA N が一つ与えられたならば、それを割って DFA $N/_{\sim_N}$ を作ると、これが L を受理する DFA のうち状態数最小のもの（の一つ）となる。もし $N/_{\sim_N}$ の状態数が最小でないとすると、DFA M で、$L(M) = L$ かつ $|M| < |N/_{\sim_N}|$ とするものがある。しかし、上の補題により $|N/_{\sim_N}| \leq M$ でなければならないから、矛盾してしまう。

DFA N から $N/_{\sim_N}$ をつくる際に必要な情報は、N の二つの状態 $p, q \in Q_N$ について、$p \sim_N q$ か $p \not\sim_N q$ を判断できることである。すなわち、次の関数 $f : Q \times Q \to \mathbb{B}$ を計算できていれば良い:

$$f(p,q) = \begin{cases} 1 & \text{if } p \sim_N q, \\ 0 & \text{if } p \not\sim_N q. \end{cases}$$

7.1 方法 1 (言語の等価性判定を用いる)

DFA $N = (Q, \Sigma, \delta, q_{\text{init}}, F)$ を固定する。N の状態 p について、初期状態を p で置き換えて得られる DFA を N_p と書く:

$$N_p = (Q, \Sigma, \delta, p, F).$$

この時、N の二つの状態が区別できないこと $p \sim_N q$ と、$L(N_p) = L(N_q)$ が等価になる。

命題 9. N の二状態 $p, q \in Q$ について、$p \sim_N q \iff L(N_p) = L(N_q)$.

Proof. 練習問題 □

二つの DFA A と B の言語が等しいかどうかが解けることは既に述べた通りなので、二つの状態が区別できるかどうかを判定できる。

7.2 方法 2 (もう少し直接的な方法)

このような関数 f を、以下の iteration algorithm で構成する。

まず、$g_0 : Q \times Q \to \{\circ, \bullet\}$ を作る:

$$\begin{aligned} g_0 \quad &: \quad Q \times Q \to \{\circ, \bullet\} \\ g_0(p,q) \quad &= \quad \begin{cases} \circ & \text{if } \tau(p,\epsilon) \neq \tau(q,\epsilon), \\ \bullet & \text{otherwise.} \end{cases} \end{aligned}$$

$g_0(p,q) = \circ$ であるならば、長さ 0 の文字列（空文字列 ϵ) で区別できることを意味する。一方で $g_0(p,q) = \bullet$ は、まだ証拠が見つかっていないので、p と q を区別できないことを意味している。

このあと、$g_0 \mapsto g_1 \mapsto g_2 \mapsto \ldots$ として、少しずつ区別できる状態を増やしていく。直感的には、$g_i(p,q) = \circ$ であれば、長さが i 以下の文字列で p と q が区別できて、$g_i(p,q) = \bullet$ であれば、長さが i 以下の文字列では区別できていない（まだ区別できていないだけで、最終的にどうなるかは不明）ということを表している。

$g_i \mapsto g_{i+1}$ として拡大するには次のようにすれば良い:

$$\begin{aligned} g_{i+1} \quad &: \quad Q \times Q \to \{\circ, \bullet\} \\ g_{i+1}(p,q) \quad &= \quad \begin{cases} \circ & \text{if } g_i(p,q) = \circ, \\ \circ & \text{if } \exists \sigma \in \Sigma.\ g_i(\delta(p,\sigma), \delta(q,\sigma)) = \circ, \\ \bullet & \text{otherwise.} \end{cases} \end{aligned}$$

補題 4. 任意の i で、$g_i(p,q) = \circ$ ならば、文字列 w で $|w| \leq i$ かつ $\tau(p,w) \neq \tau(q,w)$ とするものがある。

Proof. 帰納法で証明する。$g_0(p,q) = \circ$ ならば、$\tau(p,\epsilon) \neq \tau(q,\epsilon)$ なので、$p \not\sim q$ である。

$g_{i+1}(p,q) = \circ$ かつ $g_i(p,q) = \circ$ ならば、帰納法の仮定から $\tau(p,w) \neq \tau(q,w)$ とする長さ i 以下の文字列 w がある。当然 $|w| \leq i+1$ なのでこの場合は示せた。

$g_{i+1}(p,q) = \circ$ かつ $g_i(p,q) = \bullet$ ならば、ある $\sigma \in \Sigma$ で、$g_i(\delta(p,\sigma), \delta(q,\sigma)) = \circ$ である。帰納法の仮定から、ある w で $|w| \leq i$ かつ $\tau(\delta(p,\sigma), w) \neq \tau(\delta(q,\sigma), w)$ とするものがある。$\tau(p, \sigma w) = \tau(\delta(p,\sigma), w) \neq \tau(\delta(q,\sigma), w) = \tau(q, \sigma w)$ なので、$|\sigma w| \leq i+1$ をとれば題意を満たす。 □

補題 5. 文字列 w で $\tau(p,w) \neq \tau(q,w)$ ならば、$g_{|w|}(p,q) = \circ$ である。

Proof. 帰納法で証明する。

$w = \epsilon$ の場合は、定義から $g_0(p,q) = \circ$ なので明らか。

$w = \sigma v$ かつ $\tau(p, \sigma v) \neq \tau(q, \sigma v)$ とする。定義から $\tau(\delta(p,\sigma), v) \neq \tau(\delta(q,\sigma), v)$ でもある。帰納法の仮定から、$g_{|v|}(\delta(p,\sigma), \delta(q,\sigma)) = \circ$ である。従って $g_{|w|}(p,q) = \circ$ となる。 □

$g : Q \times Q \to \{\circ, \bullet\}$ 上に次のような関係 $\prec$ を入れる:

$$g \prec g' \iff g \neq g' \wedge \forall p, q \in Q.\ g(p,q) = \circ \implies g'(p,q) = \circ.$$

$g_i \prec g_{i+1}$ の直感的な意味は、g_i では区別が付いていなかった状態の組み合わせ (p,q) のうち $(g_i(p,q) = \bullet)$ g_{i+1} で区別がついた $(g_{i+1}(p,q) = \circ)$ ものが一つ以上あることを意味する。

さて、Q が有限集合なのでいつまでも拡大することはできず、次のように飽和してしまう:

$$g_0 \prec g_1 \prec g_2 \prec \cdots \prec g_k = g_{k+1} = g_{k+2} = \cdots$$

この飽和点 g_k について、次のことが成立する。

補題 6. $g_k(p,q) = \circ$ と $p \not\sim q$ が等価である。

Proof. まず $g_k(p,q) = \circ$ ならば $p \not\sim q$ を示す。これは補題 4 から明らかである。

逆に $p \not\sim q$ ならば $g_k(p,q) = \circ$ を示す。$p \not\sim q$ ということは、ある文字列 w で $\tau(p,w) \neq \tau(q,w)$ である。補題 5 により $g_{|w|}(p,q) = \circ$ である。さて、$|w| \leq k$ ならば、$g_{|w|} \prec g_k$ から $g_k(p,q) = \circ$ である。一方で、$k < |w|$ であったとすると $g_k = g_{|w|}$ なので、やはり $g_k(p,q) = \circ$ である。 □

この補題により、f は飽和点 g_k を用いて定義することができる:

$$f(p,q) = \begin{cases} 1 & \text{if } g_k(p,q) = \bullet, \\ 0 & \text{if } g_k(p,q) = \circ. \end{cases}$$