4. 正則言語の性質(2):
(テキスト4.3,4.4)

4.3. 正則言語に関する決定問題

言語に関する基本的な問題

1. 与えられた言語 L が L=Φか?または $L=\Sigma^*$ か?

   例) L_1={ w | w に含まれる0の数は偶数} $L_1 \cap L_2=\Phi$?

   L_2={ w | w に含まれる0の数は奇数} $L_1 \cup L_2=\Phi$?

2. 与えられた語 w が言語 L に属するか。

   例) 0000111101011000 ∈ L_1?

3. 二つの言語 L_1, L_2 は同じか。

   例) (01)* + (10)* + 1(01)* + 0(10)* = (1+ε)(01)*(0+ε)?

0と1が交互に現れる文字列

4. Regular languages (2):
(Text 4.3,4.4)

4.3. Decision problem for regular languages

Basic problems for a language

1. For given language L, determine if L=Φ? determine if $L=\Sigma^*$?

   Ex) L_1={ w | w contains even 0s} $L_1 \cap L_2=\Phi$?

   L_2={ w | w contains odd 0s} $L_1 \cup L_2=\Phi$?

2. Given word w, determine if $w \in L$ or not.

   Ex) 0000111101011000 ∈ L_1?

3. Determine two languages are the same?

   Ex) (01)* + (10)* + 1(01)* + 0(10)* = (1+ε)(01)*(0+ε)?

Words consists of the repetition of 0 and 1

4. 正則言語の性質(2):
(テキスト4.3,4.4)

[余談]
現実的には
NFA→DFAで
指数関数的に
状態数が増える
ことはあまりない。
ただし人工的に
そうした例を構成
することはできる。

4.3. 正則言語に関する決定問題

4.3.1. 異なる表現の間の変換

1. NFA→DFAのコスト(時間): O($n^3 2^n$)
2. DFA→NFAのコスト: O(n)
3. オートマトン→正則表現: O($n^3 4^n$)
4. 正則表現→ε-NFA: O(n)

多項式/指数関数かどう
かはシビアな問題

最悪の場合は
指数関数的(=爆発的)
に増加

[Note] Practically, that exponential increasing is rare.
But it can be theoretically.

4. Regular languages (2):
(Text 4.3,4.4)

4.3. Decision problem for regular languages

4.3.1. Exchange the representations

1. NFA→DFA takes O($n^3 2^n$) time
2. DFA→NFA takes O(n) time
3. Automaton → Regular expression takes O($n^3 4^n$) time
4. Regular expression → ε-NFA takes O(n) time

It is crucial that
polynomial or exponential

In the worst case,
it explosively increases!

4. 正則言語の性質(2):
(テキスト4.3,4.4)

4.3. 正則言語に関する決定問題

4.3.2. 正則言語の空言語判定(L=Φ?)

1. 正則言語ならオートマトンが構成できる。
2. 構成したオートマトン上で、初期状態から受理状態にたどりつければ L≠Φ である。
3. 「グラフの到達性」を解けばよい。

   ⇒ O(n^2)程度の時間で解ける。

4. Regular languages (2):
(Text 4.3,4.4)

4.3. Decision problem for regular languages

4.3.2. Check the emptiness (L=Φ?)

1. Construct an automaton for the regular language.
2. We can determine L≠Φ if the automaton can transfer to the accepting state from the initial state.
3. Solve the 'reachabile problem on a graph'

   ⇒ it can be solved in O(n^2) time.

4. 正則言語の性質(2):
(テキスト4.3,4.4)

4.3. 正則言語に関する決定問題

4.3.3. 正則言語の所属性判定($w \in L$?)

1. 正則言語ならオートマトンが構成できる。
2. 構成したオートマトン上で、w についての遷移を実際に模倣してみればよい。

   ⇒ DFAなら O(n) の時間で解ける。

4. Regular languages (2):
(Text 4.3,4.4)

4.3. Decision problem for regular languages

4.3.3. Membership problem ($w \in L$?)

1. Construct an automaton for the regular language.
2. Simulate the transfer for w on the automaton.

   ⇒ It can be solved in O(n) time for the DFA.

4. 正則言語の性質(2):
(テキスト4.3,4.4)

4.4. オートマトンの等価性と最小性

3. 二つの言語 L_1, L_2 は同じか。

例) $(\mathbf{01})^* + (\mathbf{10})^* + \mathbf{1}(\mathbf{01})^* + \mathbf{0}(\mathbf{10})^*$ と
$(\mathbf{1}+\varepsilon)(\mathbf{01})^*(\mathbf{0}+\varepsilon)$は同じ言語か?

[目標]

- DFA には「最小」のものがある
- 最小のDFAは本質的に1つしかない
- 最小のDFAは計算によって求めることができる
- <u>二つの正則言語の同値性を効率よく判定できる。</u>

4. Regular languages (2):
(Text 4.3,4.4)

4.4. Equivalence and Minimum of an automaton

3. Equivalence of two languages L_1 and L_2

Ex) Do $(\mathbf{01})^* + (\mathbf{10})^* + \mathbf{1}(\mathbf{01})^* + \mathbf{0}(\mathbf{10})^*$ and
$(\mathbf{1}+\varepsilon)(\mathbf{01})^*(\mathbf{0}+\varepsilon)$ represent the same language?

[Outline]

- There is the minimum DFA.
- The minimum DFA is uniquely determined.
- The minimum DFA can be computed efficiently.
- <u>We can determine if two regular languages are equivalent or not.</u>

4. 正則言語の性質(2):
(テキスト4.3,4.4)

4.4. オートマトンの等価性と最小性

4.4.1. 状態の同値性の判定

DFA における状態 p, q が同値(equivalent)

⇕

すべての文字列 w に対して、

$\hat{\delta}(p,w)$が受理状態⇔ $\hat{\delta}(q,w)$が受理状態

が成立する

必ずしも同じ状態でなくてもOK

4. Regular languages (2):
(Text 4.3,4.4)

4.4. Equivalence and Minimum of an automaton

4.4.1. Equivalence of two states

Two states p and q in DFA are equivalent

⇕

For every words w,

$\hat{\delta}(p,w)$ is accepting state ⇔ $\hat{\delta}(q,w)$ is accepting state

They are not necessarily the same state.

4. 正則言語の性質(2):
(テキスト4.3,4.4)

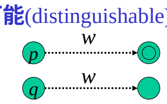
4.4. オートマトンの等価性と最小性

4.4.1. 状態の同値性の判定

DFA における状態 p, q が**区別可能(distinguishable)**

⇕

状態 p,q が<u>同値ではない</u>



⇕

ある文字列 w が存在して、以下が成立:

$\hat{\delta}(p,w)$, $\hat{\delta}(q,w)$ の一方は受理状態で、他方はそうでない

4. Regular languages (2):
(Text 4.3,4.4)

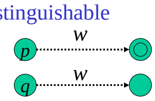
4.4. Equivalence and Minimum of an automaton

4.4.1. Equivalence of two states

Two states p and q in a DFA are distinguishable

⇕

The states p,q are not <u>equivalent</u>



⇕

There <u>*exists*</u> a word w such that one of $\hat{\delta}(p,w)$ and $\hat{\delta}(q,w)$ is accepting state, and the other is not.

4. 正則言語の性質(2):
(テキスト4.3,4.4)

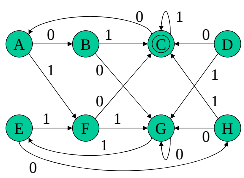
4.4. オートマトンの等価性と最小性

4.4.1. 状態の同値性の判定

例) 受理状態の集合をX={C}と書く。

$\hat{\delta}(C,\varepsilon) \in X$

$\hat{\delta}(G,\varepsilon) \notin X$



⇩

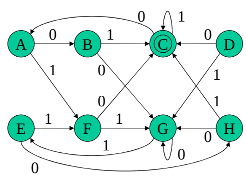
*C*と*G*は区別可能

4. Regular languages (2):
(Text 4.3,4.4)

4.4. Equivalence and Minimum of an automaton

4.4.1. Equivalence of two states

Ex) X={C} denotes the accepting state set



$\hat{\delta}(C,\varepsilon) \in X$

$\hat{\delta}(G,\varepsilon) \notin X$

⇩

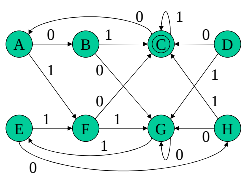
C and *G* are distinguishable

4. 正則言語の性質(2):
(テキスト4.3,4.4)

4.4. オートマトンの等価性と最小性

4.4.1. 状態の同値性の判定

例) 受理状態の集合をX={C}と書く。



$\hat{\delta}(A,\varepsilon) \notin X, \hat{\delta}(G,\varepsilon) \notin X$

$\hat{\delta}(A,0) \notin X, \hat{\delta}(G,0) \notin X$

$\hat{\delta}(A,1) \notin X, \hat{\delta}(G,1) \notin X$

$\hat{\delta}(A,01) \in X, \hat{\delta}(G,01) \notin X$

⇩

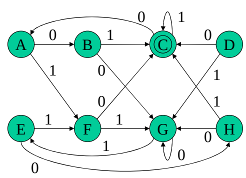
*A*と*G*は区別可能

4. Regular languages (2):
(Text 4.3,4.4)

4.4. Equivalence and Minimum of an automaton

4.4.1. Equivalence of two states

Ex) X={C} denotes the accepting state set



$\hat{\delta}(A,\varepsilon) \notin X, \hat{\delta}(G,\varepsilon) \notin X$

$\hat{\delta}(A,0) \notin X, \hat{\delta}(G,0) \notin X$

$\hat{\delta}(A,1) \notin X, \hat{\delta}(G,1) \notin X$

$\hat{\delta}(A,01) \in X, \hat{\delta}(G,01) \notin X$

⇩

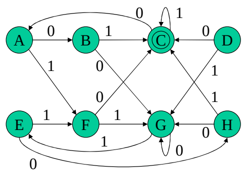
A and *G* are distinguishable

4. 正則言語の性質(2):
(テキスト4.3,4.4)

4.4. オートマトンの等価性と最小性

4.4.1. 状態の同値性の判定

例） 受理状態の集合を$X=\{C\}$と書く。



$\hat{\delta}(A,\varepsilon) \notin X, \hat{\delta}(E,\varepsilon) \notin X$

$\hat{\delta}(A,1) = \hat{\delta}(E,1) = F$

$\hat{\delta}(A,0) \notin X, \hat{\delta}(E,0) \notin X$

$\hat{\delta}(A,00) = \hat{\delta}(E,00) = G$

$\hat{\delta}(A,01) = \hat{\delta}(E,01) = C$

*A*と*E*は同値

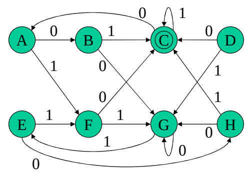
19/50

4. Regular languages (2):
(Text 4.3,4.4)

4.4. Equivalence and Minimum of an automaton

4.4.1. Equivalence of two states

Ex) $X=\{C\}$ denotes the accepting state set



$\hat{\delta}(A,\varepsilon) \notin X, \hat{\delta}(E,\varepsilon) \notin X$

$\hat{\delta}(A,1) = \hat{\delta}(E,1) = F$

$\hat{\delta}(A,0) \notin X, \hat{\delta}(E,0) \notin X$

$\hat{\delta}(A,00) = \hat{\delta}(E,00) = G$

$\hat{\delta}(A,01) = \hat{\delta}(E,01) = C$

A and *E* are equivalent

20/50

4. 正則言語の性質(2):
(テキスト4.3,4.4)

実装上の工夫:
区別可能なペアから逆に構成

4.4. オートマトンの等価性と最小性

4.4.1. 状態の同値性の判定

同値な状態のペアを求める穴埋めアルゴリズム (Table-filling algorithm)

1. 状態 *p* が受理状態で、*q* が受理状態ではないとき、{*p*,*q*} は区別可能
2. 状態 *p*, *q* と、ある入力文字 *a* に対して、*r*= δ(*p*,*a*), *s*= δ(*q*,*a*) としたとき、{*r*,*s*} が区別可能なら {*p*,*q*} も区別可能
3. ステップ2を繰り返し適用し、それ以上変化しなくなったら終了

21/50

4. Regular languages (2):
(Text 4.3,4.4)

4.4. Equivalence and Minimum of an automaton

4.4.1. Equivalence of two states

Table-filling algorithm that finds all equivalence pairs

1. If *p* is accepting state and *q* is not, they are distinguishable.
2. For two states *p*, *q* and an input alphabet *a*, let *r*= δ(*p*,*a*) and *s*= δ(*q*,*a*). Then, *p* and *q* are distinguishable if *r* and *s* are distinguishable.
3. Repeat step 2 while the table is updated.

Note for implementation: Mark the table in reverse from the distinguishable pairs.

22/50

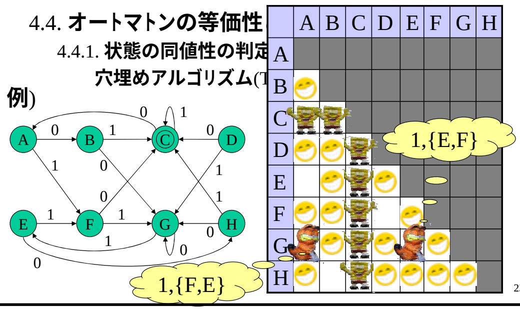
4. 正則言語の性質(2):
(テキスト4.3,4.4)

4.4. オートマトンの等価性

4.4.1. 状態の同値性の判定

穴埋めアルゴリズム(T

例)



23/50

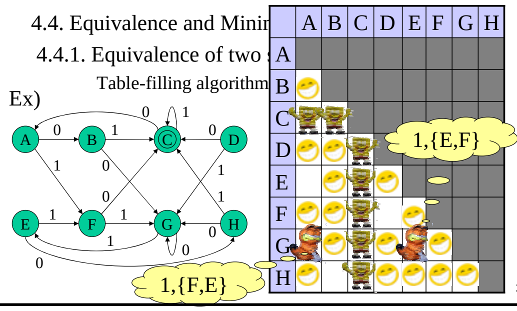
4. Regular languages (2):
(Text 4.3,4.4)

4.4. Equivalence and Minin

4.4.1. Equivalence of two s

Table-filling algorithm

Ex)



24/50

4. 正則言語の性質(2):
(テキスト4.3,4.4)

4.4. オートマトンの等価性と最小性

4.4.1. 状態の同値性の判定

穴埋めアルゴリズム(Table-filling algorithm)

2. 状態 p, q と、ある入力文字 a に対して、$r=\delta(p,a)$, $s=\delta(q,a)$ としたとき、$\{r,s\}$ が区別可能なら $\{p,q\}$ も区別可能

•$\{r,s\}$が区別可能 ⇒ ある文字列 w があって、$\delta(r,w)$ と $\delta(s,w)$ が一方は受理状態で、他方はそうではない
•文字列 aw が状態 p と q を区別可能にする。

⇒「区別可能」と判断されたものは、区別可能。 25/50

4. Regular languages (2):
(Text 4.3,4.4)

4.4. Equivalence and Minimum of an automaton

4.4.1. Equivalence of two states

Table-filling algorithm

1. For two states p, q and an input alphabet a, let $r=\delta(p,a)$ and $s=\delta(q,a)$. Then, p and q are distinguishable if r and s are distinguishable.

• When r and s are distinguishable ⇒ There exists a string w such that one of $\delta(r,w)$ and $\delta(s,w)$ is accepting state and the other one is not.
• So the string aw makes p and q distinguishable.

⇒A marked pair of two states should be distinguishable. 26/50

4. 正則言語の性質(2):
(テキスト4.3,4.4)

4.4.オートマトンの等価性と最小性

4.4.1. 状態の同値性の判定

穴埋めアルゴリズム(Table-filling algorithm)の正当性

- 区別可能なものは必ず区別可能と判断される
- 同値なペアは最後まで何も判断されず、空白となる

[定理] 穴埋めアルゴリズムによって区別されない二つの状態 p, q は同値である。

27/50

4. Regular languages (2):
(Text 4.3,4.4)

4.4. Equivalence and Minimum of an automaton

4.4.1. Equivalence of two states

Correctness of the Table-filling algorithm stands for

- distinguishable pair will be surely determined, and
- equivalent pair will be never marked.

[Theorem] Two states p and q are equivalent if the entry $\{p,q\}$ is not marked by the algorithm.

28/50

4. 正則言語の性質(2):
(テキスト4.3,4.4)

4.4. オートマトンの等価性と最小性

[定理] 穴埋めアルゴリズムによって区別されない二つの状態 p, q は同値である。

[証明] 背理法による。定理が間違っていたとする。

1. p, q が同値なのに区別されたとする。アルゴリズムの構成により、これは起こりえない。
2. p, q は区別可能で、かつ、アルゴリズムは区別しない。

2の条件を満たすペアを悪いペアと呼ぶ。

各悪いペア p,q に対して、それらを区別する最短の文字列 $w_{p,q}$ が存在する。

最短の文字列 $w_{p,q}$ が最短であるような悪いペア p,q を考える。

$\hat{\delta}(p,w_{p,q}),\hat{\delta}(q,w_{p,q})$の一方だけ受理状態 29/50

4. Regular languages (2):
(Text 4.3,4.4)

4.4. Equivalence and Minimum of an automaton

[Theorem] Two states p and q are equivalent if the entry $\{p,q\}$ is not marked by the algorithm.

[Proof] To derive contradictions, suppose theorem is not true.

1. By the construction of the algorithm, $\{p, q\}$ will be never marked if they are equivalent.
2. So, theorem is not true only if the algorithm fails to distinguish a distinguishable pair $\{p,q\}$.

We call the pair in condition 2 bad pair.

For each bad pair $\{p,q\}$, there is the shortest word $w_{p,q}$ that distinguish p and q.

We take a bad pair $\{p,q\}$ such that $w_{p,q}$ is the shortest among such words.

exactly one of $\hat{\delta}(p,w_{p,q})$ and $\hat{\delta}(q,w_{p,q})$ is accepting state 30/50

4. 正則言語の性質(2): (テキスト4.3,4.4)

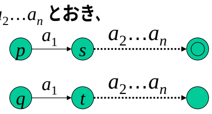
4.4. オートマトンの等価性と最小性

[定理] 穴埋めアルゴリズムによって区別されない二つの状態 p, q は同値である。

[証明] 背理法による。定理が間違っていたとする。

最短の文字列 w_{pq} が最短である悪いペア p,q を考える。

1. $w_{pq} = \varepsilon$ のとき; 穴埋めアルゴリズムは最初に p,q を区別する。したがって矛盾。
2. $w_{pq} \neq \varepsilon$ のとき; $w_{pq}=a_1a_2\ldots a_n$ とおき、
   $s := \delta(p,a_1)$
   $t := \delta(q,a_1)$
   とおく。



4. Regular languages (2): (Text 4.3,4.4)

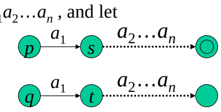
4.4. Equivalence and Minimum of an automaton

[Theorem] Two states p and q are equivalent if the entry $\{p,q\}$ is not marked by the algorithm.

[Proof] To derive contradictions, suppose theorem is not true.

We took the bad pair $\{p,q\}$ that gives the shortest $w_{p,q}$.

1. $w_{pq} = \varepsilon$; the algorithm should distinguish p and q at step 1. Hence we have a contradiction.
2. $w_{pq} \neq \varepsilon$; we let $w_{pq}=a_1a_2\ldots a_n$, and let
   $s := \delta(p,a_1)$
   $t := \delta(q,a_1)$



4. 正則言語の性質(2): (テキスト4.3,4.4)

4.4. オートマトンの等価性と最小性

[定理] 穴埋めアルゴリズムによって区別されない二つの状態p, q は同値である。

[証明] 背理法による。定理が間違っていたとする。

最短の文字列 w が最短である悪いペア p,q を考える。

2. $w_{pq} \neq \varepsilon$ のとき; $w_{pq}=a_1a_2\ldots a_n$, $s := \delta(p,a_1)$, $t := \delta(q,a_1)$ とおく。

すると s, t は文字列 $a_2a_3\ldots a_n$ によって区別される。

w_{pq}の最短性から、s, t は穴埋めアルゴリズムによって区別される。

したがって s, t が区別された次のステップで p, q も区別される。

これは p, q が悪いペアであることに矛盾。

したがって``最短の悪いペア''(つまり悪いペア)は存在しない。

4. Regular languages (2): (Text 4.3,4.4)

4.4. Equivalence and Minimum of an automaton

[Theorem] Two states p and q are equivalent if the entry $\{p,q\}$ is not marked by the algorithm.

[Proof] To derive contradictions, suppose theorem is not true.

We took the bad pair $\{p,q\}$ that gives the shortest $w_{p,q}$.

2. $w_{pq} \neq \varepsilon$; $w_{pq}=a_1a_2\ldots a_n$, $s := \delta(p,a_1)$, and $t := \delta(q,a_1)$.

Then, s, t is distinguishable by the word $a_2a_3\ldots a_n$.

Since w_{pq} is the shortest word, s and t should be distinguished by the algorithm.

Then, after s and t are distinguished, p and q should be distinguished.

This contradicts that $\{p,q\}$ is a bad pair.

Therefore, 'the shortest bad pair', or any bad pair, does not exist.

4. 正則言語の性質(2): (テキスト4.3,4.4)

4.4. オートマトンの等価性と最小性

4.4.2 正則言語の等価性の判定

与えられた正則言語L_1, L_2 の等価性は次の手順で判定できる。

1. L_1, L_2に対する DFA A_1, A_2 を構成する
2. 二つの DFA A_1, A_2 を全体として一つの DFA A とみなす。
3. A について穴埋めアルゴリズムを実行
4. A_1の初期状態とA_2の初期状態が同値なら $L_1=L_2$。そうでないなら $L_1 \neq L_2$。

素直に実装すると$O(n^4)$、工夫すると$O(n^2)$

4. Regular languages (2): (Text 4.3,4.4)

4.4. Equivalence and Minimum of an automaton

4.4.2 Decision of the equivalence of two regular languages

For any two regular languages L_1 and L_2, we can determine if they are the same or not as follows;

1. *Construct DFA A1 and A2 with L(A1)=L_1*, L(A2)=L_2
2. Regard that they are a DFA A without unique initial state
3. Perform Table-filling algorithm for A
4. *If the initial state of* A_1 and the initial state of A_2 are equivalent, we have $L_1=L_2$. Otherwise, $L_1 \neq L_2$.

Naïve implementation takes $O(n^4)$, but can be improved to $O(n^2)$

4. 正則言語の性質(2):
(テキスト4.3,4.4)

4.4. オートマトンの等価性と最小性

4.4.3. DFA の最小化

与えられた DFA *A* に対して穴埋めアルゴリズムを実行する→任意の状態ペア *p,q* は同値か区別可能

同値関係は推移律を満たす

[定理] *p* と *q* が同値で *q* と *r* が同値なら *p* と *r* も同値。

[略証] *p,q* が同値、*q,r* が同値で、かつ *p,r* が同値でないとする。このとき、*p,r* を区別する文字列 *w* が存在する。すると *w* によって *p* と *q* または *q* と *r* が同値ではなくなってしまい、矛盾。

4. Regular languages (2):
(Text 4.3,4.4)

4.4. Equivalence and Minimum of an automaton

4.4.3. Minimization of a DFA

Table-filling algorithm for a DFA *A*→ Each pair of states *p,q* is equivalent or distinguishable.

[Theorem] Suppose *p* and *q* are equivalent, and so are *q* and *r*. Then *p* and *r* are equivalent.

[Proof (Sketch)] Assume {*p,q*} and {*q,r*} are equivalent and {*p,r*} is not. Then there is a word *w* that distinguish *p* and *r*. But *w* also distinguish either *p,q* or *q,r*. This is a contradiction.

Equivalent relation is transitive

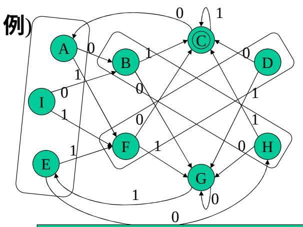
4.4. オートマトンの等価性と最小性

4.4.3. DFA の最小化

[定理] *p* と *q* が同値で *q* と *r* が同値なら *p* と *r* も同値。

[系] 同値関係は状態集合を同値なブロックに分割する。

例)



同値なペア;
{A,E},{D,F},{B,H},
{A,I},{E,I}

同値なブロック;
{A,E,I},{D,F},{B,H}

``同値なブロック''は一つの状態と見なせる。

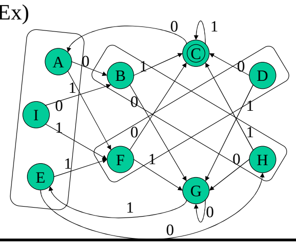
4.4. Equivalence and Minimum of an automaton

4.4.3. Minimization of a DFA

[Theorem] {*p,q*} and {*q,r*} are equivalent → {*p,r*} is equivalent

[Corollary] Equivalent relation partitions states into 'equivalent blocks.'

Ex)



Equivalent pairs;
{A,E},{D,F},{B,H},
{A,I},{E,I}

Equivalent blocks;
{A,E,I},{D,F},{B,H}

An ``equivalent block'' seems one state.

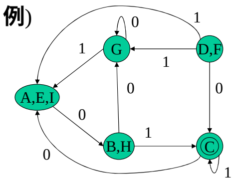
4.4. オートマトンの等価性と最小性

4.4.3. DFA の最小化

[定理] *p* と *q* が同値で *q* と *r* が同値なら *p* と *r* も同値。

[系] 同値関係は状態集合を同値なブロックに分割する。

例)



同値なペア;
{A,E},{D,F},{B,H},
{A,I},{E,I}

同値なブロック;
{A,E,I},{D,F},{B,H}

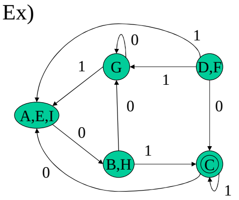
4.4. Equivalence and Minimum of an automaton

4.4.3. Minimization of a DFA

[Theorem] {*p,q*} and {*q,r*} are equivalent → {*p,r*} is equivalent

[Corollary] Equivalent relation partitions states into 'equivalent blocks.'

Ex)



Equivalent pairs;
{A,E},{D,F},{B,H},
{A,I},{E,I}

Equivalent blocks;
{A,E,I},{D,F},{B,H}

4.4. オートマトンの等価性と最小性

4.4.3. DFA の最小化

[定理] DFA $A=(Q, \Sigma, \delta, q, F)$ に対し、同値なブロックを ``新しい状態''とみなした DFA $B=(Q, \Sigma', \delta', q', F')$を構成する。詳しくは

1. 初期状態 q を含むブロックを新しい初期状態 q' とする。
2. 受理状態を含むブロックを新しい受理状態とする。
3. $\delta(p_1,a)=p_2$ で、p_1 がブロック P_1, p_2 がブロック P_2 に属すなら、$\delta'(P_1,a)=P_2$ とする。

このとき、$L(A)=L(B)$となる。

[略証]

2. 受理状態と同じブロックのものはすべて受理状態。
3. ブロック P_1 のどの状態 p_1' でも、入力 a で遷移すれば、必ず ブロック P_2 のどれかの状態に遷移することが示せる。

4.4. Equivalence and Minimum of an automaton

4.4.3. Minimization of a DFA

[Theorem] For a DFA $A=(Q, \Sigma, \delta, q, F)$, we construct a DFA $B=(Q, \Sigma', \delta', q', F')$ by regarding 'equivalent block' as 'a new state'. Precisely,

1. q' is the block containing q,
2. F' is the set of blocks containing a state in F,
3. $\delta'(P_1,a)=P_2$ if $\delta(p_1,a)=p_2$, p_1 is in block P_1, and p_2 is in block P_2.

Then, $L(A)=L(B)$.

[Proof (Outline)]

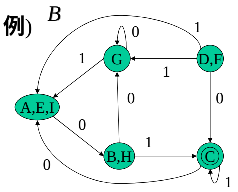
2. Any equivalent state of an accepting state is an accepting state.
3. For any state p_1' in P_1, $\delta(p_1,a) \in P_2$.

4.4. オートマトンの等価性と最小性

4.4.3. DFA の最小化

[定理] 前記の B はその言語を受理する DFA の中で、状態数が最小で、かつ一意的に構成される。

例) B



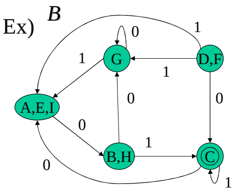
➢ B よりも少ない状態数ではこの言語は受理できない。
➢ どんな冗長なDFA から出発しても同じ B になる。

4.4. Equivalence and Minimum of an automaton

4.4.3. Minimization of a DFA

[Theorem] The automaton B constructed from A has the minimum number of states, and it is determined uniquely.

Ex) B



➢ No automaton with less states than B can accept the same language.
➢ From any redundant DFA, we have the same B if the language is the same.

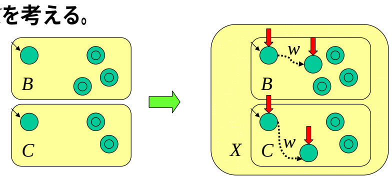
4.4. オートマトンの等価性と最小性

4.4.3. DFA の最小化

[定理] 前記の B はその言語を受理する DFA の中で、状態数が最小で、かつ一意的に構成される。

直積構成法

[略証] $L(B)=L(C)$ で、Cの状態数がBよりも少なかったと仮定する。B と C を内包して、同時に模倣するDFA Xを考える。



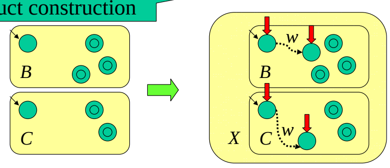
4.4. Equivalence and Minimum of an automaton

4.4.3. Minimization of a DFA

[Theorem] The automaton B constructed from A has the minimum number of states, and it is determined uniquely.

[Proof (Sketch)] Let C be a DFA with states less than B and $L(B)=L(C)$. Consider a DFA X that contains B and C, and simulates them simultaneously.

Product construction

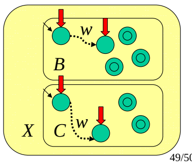


[定理] 前記の B はその言語を受理する DFA の中で、状態数が最小で、かつ一意的に構成される。

[略証] L(B)=L(C) で、Cの状態数がBよりも少なかったと仮定する。B と C を内包して、同時に模倣するDFA Xを考える。

一意性も B,C の最小性と同値性から示せる。

B の二つの状態 p_1, p_2 が C の中では同じ状態 p に対応することになる。
p_1, p_2 は区別可能なので、
ある文字列 w が存在して、
$\hat{\delta}(p_1,w)$と $\hat{\delta}(p_2,w)$は一方が
受理状態で、他方は違う。
よって $L(B) = L(C)$ に矛盾。

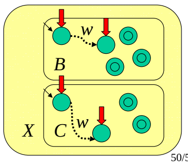


49/50

[Theorem] The automaton B constructed from A has the minimum number of states, and it is determined uniquely.

[Proof (Sketch)] Let C be a DFA with states less than B and L(B)=L(C). Consider a DFA X that contains B and C, and simulates them simultaneously.

Two distinct states p_1 and p_2 in B
corresponds to the same state p in C.
However, since p_1, p_2 are distinguishable,
There is a word w that brings
$\hat{\delta}(p_1,w)$ and $\hat{\delta}(p_2,w)$ to
the different states.
This contradicts that $L(B) = L(C)$.



Uniqueness is also derived from the fact that B is minimum and $L(C)$=$L(B)$.

50/50