3. 正則表現:
(テキスト3.1,3.2)

3.1. 正則表現

– 正則表現(または正規表現)とは、文字列の集合(=言語)を有限個の記号列で表現する方法の1つ

例: **(01)*** … 「01を繰り返す文字列」

0(0+1)* … 「0の後に0か1が繰り返す文字列」

つまり

(01)* = { ε ,01,0101,010101,01010101,…}

0(0+1)*={0,00,01,000,001,010,011,0000,…}

– UNIX系の人にはおなじみ…grep, emacs, awk, perl, …

– Windows系の人にも…ファイル名のワイルドカードなど

1/46

3. Regular Expression:
(Text 3.1,3.2)

3.1. Regular Expression

– Regular Expression describes a set of words (=language) by finite characters

Ex: **(01)*** … words repeating '01'

0(0+1)* … repeating '0' or '1' after 0

I.e.

(01)* = { ε ,01,0101,010101,01010101,…}

0(0+1)*={0,00,01,000,001,010,011,0000,…}

– Familiar if you are UNIX-user…grep, emacs, awk, perl, …

– So-called 'Wild-card' in filenames on Windows

2/46

3. 正則表現:
(テキスト3.1,3.2)

3.1. 正則表現の直感的な定義と意味

– 文字や文字列はそのまま解釈:
  - **a** → {*a*}
  - **ab** → {*ab*}
– 「+」は「または」の意味:
  - **ab**+**a** → {*ab*,*a*}
– 「()」はグループ化
– 「*」は「0回以上の繰り返し」の意味
  - (**ab**)* → { ε , *ab*, *abab*, *ababab*,…}

ちょっと複雑な例:

((**ab**)*c)+(**a***)

→ { ε , *a*, *c*, *aa*, *aaa*, *abc*, *aaaa*, *aaaaa*, *ababc*,… }

3/46

3. Regular Expression:
(Text 3.1,3.2)

3.1. Brief explanation

– A character and string is as is:
  - **a** → {*a*}
  - **ab** → {*ab*}
– '+' = OR
  - **ab**+**a** → {*ab*,*a*}
– '()' makes a group
– '*' means repetition 0 times or more
  - (**ab**)* → { ε , *ab*, *abab*, *ababab*,…}

An complicated example:

((**ab**)*c)+(**a***)

→ { ε , *a*, *c*, *aa*, *aaa*, *abc*, *aaaa*, *aaaaa*, *ababc*,… }

4/46

3. 正則表現:
(テキスト3.1,3.2)

3.1.1. 正則表現の演算

1. 和集合(union): 二つの言語 L, M の和集合$L \cup M$は、LかMのどちらかに含まれる要素の集合.
   - 例: $\{AB,C\} \cup \{a,b,c\} = \{a,b,c,AB,C\}$
2. 連接(concatenation): 二つの言語L,Mの連接LM(または$L \cdot M$)は、それぞれの要素を一つづつとってつなげたものの集合
   – 例: $\{AB,C\}\{a,b,c\} = \{ABa, ABb, ABc, Ca, Cb, Cc\}$
3. 閉包(closure): ある言語Lの閉包L^*は、Lの要素を0個以上連接したものの集合
   – 例: $\{a,b,c\}^*=\{ \varepsilon ,a,b,c,aa,ab,ac,ba,bb,bc,ca,cb,cc,aaa,\ldots\}$

5/46

3. Regular Expression:
(Text 3.1,3.2)

3.1.1. Operations over regular expressions

1. The union $L \cup M$ contains elements in L or M.
   - Ex: $\{AB, C\} \cup \{a, b, c\} = \{a, b, c, AB, C\}$
2. The concatenation of two languages L and M is denoted by LM (or $L \cdot M$) is a set of all combinations of two elements from L and M, respectively.
   – Ex: $\{AB, C\}\{a, b, c\} = \{ABa, ABb, ABc, Ca, Cb, Cc\}$
3. The closure L^* of a language L is a set of all concatenations of any number of elements in L.
   – Ex: $\{a,b,c\}^*=\{ \varepsilon ,a,b,c,aa,ab,ac,ba,bb,bc,ca,cb,cc,aaa,\ldots\}$

6/46

3. 正則表現:
(テキスト3.1,3.2)

3.1.1. 正則表現の演算

2.5. 言語の連接の補足: LLはL^2, LLLはL^3と書くことがある。

- 例: $\{a,ab\}^2 = \{a,ab\}\{a,ab\} = \{aa,aab,aba,abab\}$
- 定義: $L^0 := \{\varepsilon\}$, $L^1 := L$, $L^k := L^{k-1}L$ $(k>1)$

3.5. 言語の閉包の補足: 2.5 より、L^* は以下の定義と同値。

$$L^* := \bigcup_{i=0}^{\infty} L^i$$

3. Regular Expression:
(Text 3.1,3.2)

3.1.1. Operations over regular expressions

2.5. Comment on Concatenation: Sometimes, LL is denoted by L^2, and LLL *is denoted by* L^3, and so on.

- Ex: $\{a,ab\}^2 = \{a,ab\}\{a,ab\} = \{aa,aab,aba,abab\}$
- Def: $L^0 := \{\varepsilon\}$, $L^1 := L$, $L^k := L^{k-1}L$ $(k>1)$

3.5. Comment on Closure: By 2.5, L^* can be defined as follows.

$$L^* := \bigcup_{i=0}^{\infty} L^i$$

3. 正則表現:
(テキスト3.1,3.2)

Φ={}

3.1.2. 正則表現の構成

正則表現 E とそれが表現する言語 $L(E)$ の定義

1. 定数εとΦは正則表現で、$L(\varepsilon)=\{\varepsilon\}$, $L(\Phi)=\Phi$.
2. 記号 a に対して、**a** は正則表現で、$L(\mathbf{a}) = \{a\}$.
3. E と F が正則表現のとき、
   1. $E+F$ は正則表現。定義される言語; $L(E+F) = L(E) \cup L(F)$
   2. EF(または$E \cdot F$)は正則表現。定義される言語; $L(EF) = L(E)L(F)$
   3. E^*は正則表現。定義される言語; $L(E^*) = (L(E))^*$
   4. (E)は正則表現。定義される言語; $L((E)) = L(E)$

3. Regular Expression:
(Text 3.1,3.2)

Φ={}

3.1.2. Construction of a regular expression

Definition of a regular expression E and corresponding language $L(E)$

1. Two constants ε and Φ are regular expressions that represent $L(\varepsilon)=\{\varepsilon\}$, $L(\Phi)=\Phi$.
2. For a symbol a, **a** is regular expression that represents $L(\mathbf{a}) = \{a\}$.
3. For two regular expressions E and F,
   1. $E+F$ is R.E. which represents $L(E+F) = L(E) \cup L(F)$
   2. EF(or $E \cdot F$) is R.E. which represents $L(EF) = L(E)L(F)$
   3. E^* is R.E. which represents $L(E^*) = (L(E))^*$
   4. (E) is R.E. which represents $L((E)) = L(E)$

3. 正則表現:
(テキスト3.1,3.2)

3.1.2. 正則表現の構成

例: 「0と1が交互に現れる文字列」という言語

1. 発想(1): (a)01の繰り返し か (b)10の繰り返し か (c) 1のあとに(a) か (d) 0のあとに(b)
   - $(\mathbf{01})^* + (\mathbf{10})^* + \mathbf{1}(\mathbf{01})^* + \mathbf{0}(\mathbf{10})^*$
2. 発想(2): 01の繰り返しの前に1かεを追加、後に0かεを追加
   - $(\mathbf{1}+\varepsilon)(\mathbf{01})^*(\mathbf{0}+\varepsilon)$

同じ言語に違った表現があること

3. Regular Expression:
(Text 3.1,3.2)

3.1.2. Construction of a regular expression

Ex: A language that is the set of words such that '0 and 1 appear alternately'

1. Idea (1): (a) <u>repetition of 01</u>, (b) <u>repetition of 10</u>, (c) <u>(a) follows after 1</u>, or (d) <u>(b) follows after 0</u>.
   - $(\mathbf{01})^* + (\mathbf{10})^* + \mathbf{1}(\mathbf{01})^* + \mathbf{0}(\mathbf{10})^*$
2. Idea (2): add 1 or ε before repetition of 01, and add 0 or ε after it.
   - $(\mathbf{1}+\varepsilon)(\mathbf{01})^*(\mathbf{0}+\varepsilon)$

Different representations for the same language

3. 正則表現:
(テキスト3.1,3.2)

3.1.2. 正則表現の演算順序

すべて()で明記してもよいが、<u>優先順位</u>を定義すれば、()は適宜省略できる。

1. 同じ演算は左から右: **abc** = (**ab**)**c**, **a**+**b**+**c**=(**a**+**b**)+**c**
2. *は最優先: **ab***=**a**(**b**)*≠(**ab**)*
3. ・は2番目: **a**+**bc** = **a**+(**bc**)≠(**a**+**b**)**c**
4. +は最後: **a**+**bc***+**d** = (**a**+(**b**(**c***)))+**d**

3. Regular Expressions:
(Text 3.1,3.2)

3.1.2. Priority of the operations

We can omit some ()s if we define the priority of the operations below:

1. From left to right for the same operations: **abc** = (**ab**)**c**, **a**+**b**+**c**=(**a**+**b**)+**c**
2. * has top priority: **ab***=**a**(**b**)*≠(**ab**)*
3. ・ has the second: **a**+**bc** = **a**+(**bc**)≠(**a**+**b**)**c**
4. + is the last: **a**+**bc***+**d** = (**a**+(**b**(**c***)))+**d**

3. 正則表現:
(テキスト3.1,3.2)

3. 2. 有限オートマトンと正則表現

– ゴール: 正則表現で表現できる言語＝オートマトンで受理できる言語
  1. 与えられた正則表現から、ε-NFAが構成できること
  2. 与えられたDFAから正則表現が構成できること

  2'. 与えられたε-NFAから正則表現が構成できること

- ε-NFAは(見かけ上)表現力が高い
- DFAは構成要素が(見かけ上)少ない

3. Regular Expression:
(Text 3.1,3.2)

3. 2. Finite automata and regular expressions

Goal: Class of languages represented by regular expressions = Class of languages accepted by automata

1. For any given regular expression, we can construct an ε-NFA that accepts the same language
2. For any given DFA, we can construct a regular expression that represents the same language

2'. For any give NFA, …(this is easier)

- ε-NFA seems more descriptive
- DFA has simpler structure than NFA

3. 正則表現:
(テキスト3.1,3.2)

3. 2. 3. 正則表現→ε-NFA

正則表現とそれが表現する言語の定義

1. ε, Φ, 記号 *a* は正則表現; *L*(ε)={ε}, *L*(Φ)=Φ, *L*(**a**) = {*a*}.
2. 正則表現 *E* と *F* に対し、
   1. *E*+*F* は正則表現; *L*(*E*+*F*) = *L*(*E*) ∪ *L*(*F*)
   2. *EF*(または*E*・*F*)は正則表現; *L*(*EF*) = *L*(*E*)*L*(*F*)
   3. *E**は正則表現; *L*(*E**) = (*L*(*E*))*
   4. (*E*)は正則表現; *L*((*E*)) = *L*(*E*)

から直接ε-NFAを構成する。

受理状態が1つしかない

3. Regular Expression:
(Text 3.1,3.2)

3. 2. 3. Regular Expression→ε-NFA

From the definition of R.E. and corresponding set

1. Two constants ε and Φ are regular expressions that represent *L*(ε)={ε}, *L*(Φ)=Φ.
2. For a symbol *a*, **a** is regular expression that represents *L*(**a**) = {*a*}.
3. For two regular expressions *E* and *F*,
   1. *E*+*F* is R.E. which represents *L*(*E*+*F*) = *L*(*E*) ∪ *L*(*F*)
   2. *EF*(or *E*・*F*) is R.E. which represents *L*(*EF*) = *L*(*E*)*L*(*F*)
   3. *E** is R.E. which represents *L*(*E**) = (*L*(*E*))*
   4. (*E*) is R.E. which represents *L*((*E*)) = *L*(*E*)

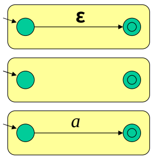
we construct an ε-NFA accepting the same language.

with one accepting state

3. 正則表現:
(テキスト3.1,3.2)

3. 2. 3. 正則表現→ ε -NFA

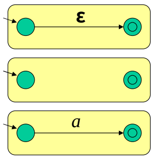
1. ε , Φ, 記号 a は正則表現; $L($ ε $)=\{$ ε $\}$, $L(Φ)=Φ$, $L(\mathbf{a}) = \{a\}$.



3. Regular Expression
(Text 3.1,3.2)

3. 2. 3. Regular Expression → ε -NFA

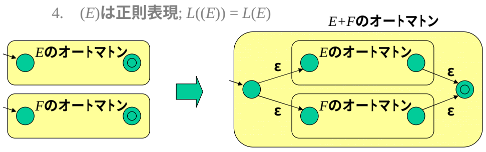
1. ε , Φ, a symbol a is R.E.; $L($ ε $)=\{$ ε $\}$, $L(Φ)=Φ$, $L(\mathbf{a}) = \{a\}$.



3. 正則表現:
(テキスト3.1,3.2)

3. 2. 3. 正則表現→ ε -NFA

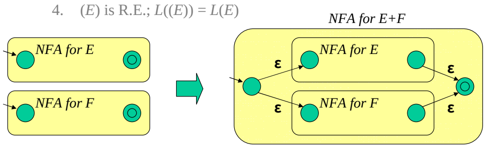
2. 正則表現 E と F に対し、
   1. $E+F$ は正則表現; $L(E+F) = L(E) \cup L(F)$
   2. EF(または$E \cdot F$)は正則表現; $L(EF) = L(E)L(F)$
   3. E^*は正則表現; $L(E^*) = (L(E))^*$
   4. (E)は正則表現; $L((E)) = L(E)$



3. Regular Expression:
(Text 3.1,3.2)

3. 2. 3. Regular Expression→ ε -NFA

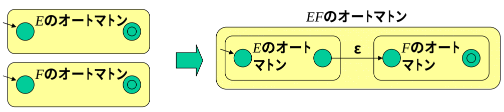
2. For R.E. E and F,
   1. $E+F$ is R.E.; $L(E+F) = L(E) \cup L(F)$
   2. EF (or $E \cdot F$) is R.E.; $L(EF) = L(E)L(F)$
   3. E^* is R.E.; $L(E^*) = (L(E))^*$
   4. (E) is R.E.; $L((E)) = L(E)$



3. 正則表現:
(テキスト3.1,3.2)

3. 2. 3. 正則表現→ ε -NFA

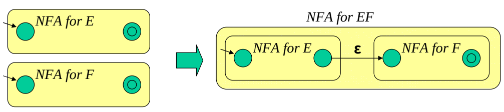
2. 正則表現 E と F に対し、
   1. $E+F$ は正則表現; $L(E+F) = L(E) \cup L(F)$
   2. EF(または$E \cdot F$)は正則表現; $L(EF) = L(E)L(F)$
   3. E^*は正則表現; $L(E^*) = (L(E))^*$
   4. (E)は正則表現; $L((E)) = L(E)$



3. Regular Expression:
(Text 3.1,3.2)

3. 2. 3. Regular Expression → ε -NFA

2. For R.E. E and F ,
   1. $E+F$ is R.E.; $L(E+F) = L(E) \cup L(F)$
   2. EF (or $E \cdot F$) is R.E.; $L(EF) = L(E)L(F)$
   3. E^* is R.E.; $L(E^*) = (L(E))^*$
   4. (E) is R.E.; $L((E)) = L(E)$



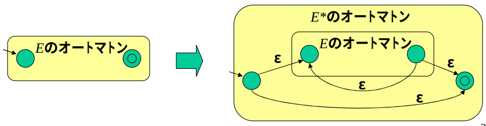
3. 正則表現:
(テキスト3.1,3.2)

3. 2. 3. 正則表現→ ε -NFA

どの規則も受理状態が1つのオートマトンしか作らない

2. 正則表現 E と F に対し、
   1. $E+F$ は正則表現; $L(E+F) = L(E) \cup L(F)$
   2. EF(または$E \cdot F$)は正則表現; $L(EF) = L(E)L(F)$
   3. E^*は正則表現; $L(E^*) = (L(E))^*$
   4. (E)は正則表現; $L((E)) = L(E)$

特に何もしない



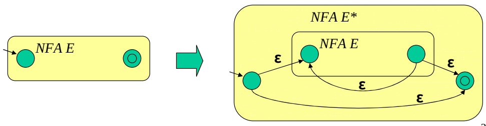
3. Regular Expression:
(Text 3.1,3.2)

3. 2. 3. Regular Expression→ ε -NFA

Each automaton has exactly one accepting state

2. For R.E. E and F,
   1. $E+F$ is R.E.; $L(E+F) = L(E) \cup L(F)$
   2. EF (or $E \cdot F$) is R.E.; $L(EF) = L(E)L(F)$
   3. E^* is R.E.; $L(E^*) = (L(E))^*$
   4. (E) is R.E.; $L((E)) = L(E)$

Nothing to do

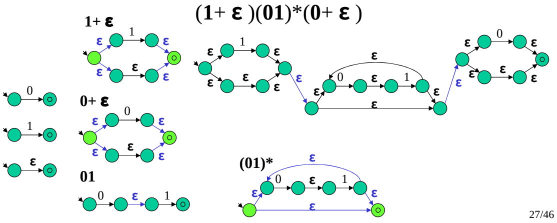


3. 正則表現:
(テキスト3.1,3.2)

3. 2. 3. 正則表現→ ε -NFA

例:「0と1が交互に出てくる文字列」の正則表現

$(1+\varepsilon)(01)^*(0+\varepsilon)$

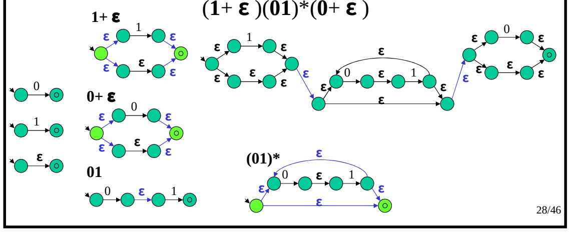


3. Regular Expression:
(Text 3.1,3.2)

3. 2. 3. Regular Expression → ε -NFA

Ex：R.E. of strings s.t. 0 and 1 appear alternately;

$(1+\varepsilon)(01)^*(0+\varepsilon)$



3. 正則表現:
(テキスト3.1,3.2)

3. 2. *. ε -NFA→正則表現

補題: 任意の ε -NFA Aに対し、$L(A)=L(A')$で、以下の条件を満たす ε -NFA A'が存在する。

1. 受理状態は1つで、受理状態からの遷移はない
2. 任意の状態 q に対し、初期状態から q への遷移と、q から受理状態への遷移が存在する

3. Regular Expression:
(Text 3.1,3.2)

3. 2. *. ε -NFA→ Regular Expression

Lemma: For any ε -NFA A, there exists an ε -NFA A' with $L(A)=L(A')$ such that

1. A' contains exactly one accepting state, and there are no transitions from the accepting state, and
2. for any state q, there is a path from the initial state, and there is a path to the accepting state.

3. 正則表現:
(テキスト3.1,3.2)

3. 2. *. ε -NFA→正則表現

補題: 任意の ε -NFA Aに対し、$L(A)=L(A')$で、以下の条件を満たす ε -NFA A'が存在する。

1. 受理状態は1つで、受理状態からの遷移はない
2. 任意の状態 q に対し、初期状態から q への遷移と、q から受理状態への遷移が存在する

証明:

2. 初期状態から到達できない状態と、受理状態に到達できない状態は受理する言語とは無関係なので、取り除いてよい。

3. Regular Expression:
(Text 3.1,3.2)

3. 2. *. ε -NFA→ Regular Expression

Lemma: For any ε -NFA A, there exists an ε -NFA A' with $L(A)=L(A')$ such that

1. A' contains exactly one accepting state, and there are no transitions from the accepting state, and
2. for any state q, there is a path from the initial state, and there is a path to the accepting state.

Proof:

2. The other states are redundant, or they have nothing to accept the language. Hence we can remove them.

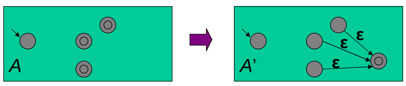
3. 正則表現:
(テキスト3.1,3.2)

3. 2. *. ε -NFA→正則表現

補題: 任意の ε -NFA Aに対し、$L(A)=L(A')$で、以下の条件を満たす ε -NFA A'が存在する。

1. 受理状態は1つで、受理状態からの遷移はない
2. 任意の状態 q に対し、初期状態から q への遷移と、q から受理状態への遷移が存在する

説明: [アイデア] <u>複数の受理状態</u>から、<u>単一の受理状態</u>へとε-動作で遷移するように与えられたオートマトンを「改造」する。



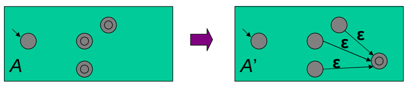
3. Regular Expression:
(Text 3.1,3.2)

3. 2. *. ε -NFA→ Regular Expression

Lemma: For any ε -NFA A, there exists an ε -NFA A' with $L(A)=L(A')$ such that

1. A' contains exactly one accepting state, and there are no transitions from the accepting state, and
2. for any state q, there is a path from the initial state, and there is a path to the accepting state.

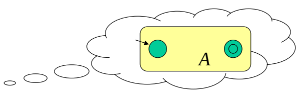
[Idea for Proof] We *modify* the automaton to translate from <u>several accepting states</u> to the <u>unique accepting state</u> by an ε -move.



3. 正則表現:
(テキスト3.1,3.2)

3. 2. *. ε -NFA→正則表現

定理: 任意の ε -NFA A に対し、$L(A)=L(E)$ となる正則表現 E が存在する。



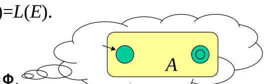
証明:

- $L(A)=\Phi$なら、$E=\Phi$
- 以下では $L(A)\neq\Phi$と仮定する。Aは以下の補題の条件を満たすとする。
  1. 受理状態は1つで、受理状態からの遷移はない
  2. 任意の状態 q に対し、初期状態から q への遷移と、q から受理状態への遷移が存在する

3. Regular Expression:
(Text 3.1,3.2)

3. 2. *. ε -NFA→ Regular Expression

Theorem: For any given ε -NFA A, there is a regular expression E such that $L(A)=L(E)$.



Proof:

- When $L(A)=\Phi$, we have $E=\Phi$.
- Thus we assume that $L(A)\neq\Phi$. We moreover suppose that A satisfies the following conditions by Lemma.

1. A contains exactly one accepting state, and there are no transitions from the accepting state, and
2. for any state q, there is a path from the initial state, and there is a path to the accepting state.

3. 正則表現:
(テキスト3.1,3.2)

3. 2. *. ε-NFA→正則表現

定理: 任意の ε-NFA A に対し、$L(A)=L(E)$ となる正則表現 E が存在する。

証明:

証明のアイデア:

– 辺のラベルに正規表現を構築していく
– 頂点を順番に削除していく

[注意] 構築途中で現れる``NFA''は正確にはNFAではない
(∵NFAでは辺のラベルはアルファベット1文字しか許されていない)

37/46

3. 正則表現:
(テキスト3.1,3.2)

3. 2. *. ε-NFA→ Regular Expression

Theorem: For any given ε-NFA A, there is a regular expression E such that $L(A)=L(E)$.

Proof:

Idea of the proof:

– We construct the regular expression as the label on an edge of A, and
– We remove states step by step.

[Note] The ``NFA'' during the process is not real NFA.
(∵Only one alphabet is allowed as a label on an edge.)

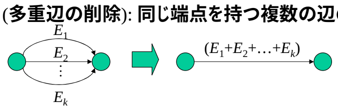
38/46

3. 2. *. ε-NFA→正則表現

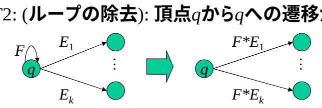
定理: 任意の ε-NFA A に対し、$L(A)=L(E)$ となる正則表現 E が存在する。

証明:

T1 (多重辺の削除): 同じ端点を持つ複数の辺の一本化



T2: (ループの除去): 頂点qからqへの遷移が1本のとき



T3: (頂点 q の削除):

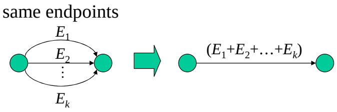
39/46

3. 2. *. ε-NFA→Regular Expression

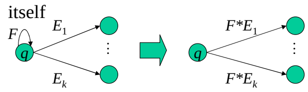
Theorem: For any given ε-NFA A, there is a regular expression E such that $L(A)=L(E)$.

Proof:

T1 (Remove multi-edges): Unify several edges with the same endpoints



T2: (Remove self-loops): When one loop from the node q to itself



T3: (Remove the node q):

40/46

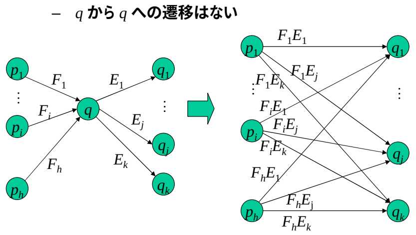
3. 2. *. ε-NFA→正則表現

定理: 任意の ε-NFA A に対し、$L(A)=L(E)$ となる正則表現 E が存在する。

証明:

T3: (頂点 q の削除):

– q は初期状態、受理状態でない
– q から q への遷移はない



41/46

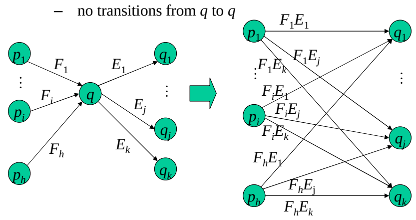
3. 2. *. ε-NFA→Regular Expression

Theorem: For any given ε-NFA A, there is a regular expression E such that $L(A)=L(E)$.

Proof:

T3: (Remove the node q):

– q is neither the initial state nor the accepting state
– no transitions from q to q



42/46

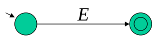
3. 正則表現:
(テキスト3.1,3.2)

3. 2. *. ε -NFA→正則表現

定理: 任意の ε -NFA A に対し、$L(A)=L(E)$ となる正則表現 E が存在する。

証明: 与えられた ε -NFA Aに対し、
1. T1(多重辺の除去)を可能な限り適用
2. T2(ループの除去)を可能な限り適用
3. T3(頂点の削除)を適用

すると、Aの初期状態と(唯一の)受理状態以外の状態が一つ減る。これを繰り返すと、初期状態と受理状態だけのNFA A'



ができる。このときの辺のラベル E が求める正規表現となる。

43/46

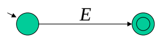
3. Regular Expression:
(Text 3.1,3.2)

3. 2. *. ε -NFA→Regular Expression

Theorem: For any given ε -NFA A, there is a regular expression E such that $L(A)=L(E)$.

Proof: For any given ε -NFA A,
1. apply T1(Remove multi-edges) as possible as you can,
2. apply T2(Remove self-loops) as possible as you can, and
3. apply T3(Remove a node).

Then a state in A (except initial and accepting states) is removed. Repeating this process, we have an NFA A' consisting of two states:



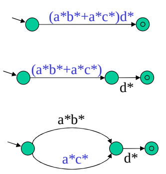
Then the label E of the unique edge gives us the regular expression.

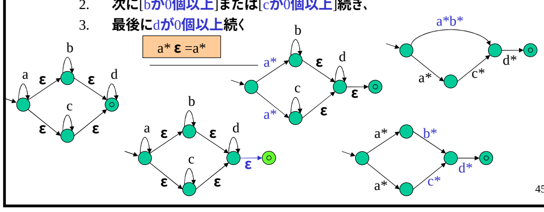
44/46

3. 正則表現:
(テキスト3.1,3.2)

3. 2. *. ε -NFA→正則表現

例:
1. まずaが0個以上続き、
2. 次に[bが0個以上]または[cが0個以上]続き、
3. 最後にdが0個以上続く



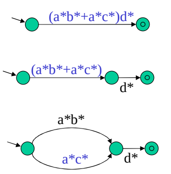


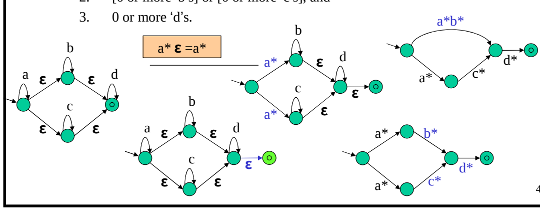
45/46

3. Regular Expression:
(Text 3.1,3.2)

3. 2. *. ε -NFA→Regular Expression

Ex.:
1. 0 or more 'a's,
2. [0 or more 'b's] or [0 or more 'c's], and
3. 0 or more 'd's.





46/46

お知らせ

- 今日のオフィスアワーは講義
  (5) 正則集合(1)

Information

- Today's Office Hour: Lecture
  (5) Regular set (1)

47/46