Formal Languages

László Aszalós, Tamás Mihálydeák

Department of Computer Science

December 10, 2017

Problems

- Analysis and generation are typical tasks in informatics:

Problems

- Analysis and generation are typical tasks in informatics:
  - In communication different standards are used (HTML, XML, json, YAML, etc). WE have to generate on sending side, and analyze on receiving side.

Problems

- Analysis and generation are typical tasks in informatics:
  - In communication different standards are used (HTML, XML, json, YAML, etc). WE have to generate on sending side, and analyze on receiving side.
- In the cases of forms:

Problems

- Analysis and generation are typical tasks in informatics:
  - In communication different standards are used (HTML, XML, json, YAML, etc). WE have to generate on sending side, and analyze on receiving side.
- In the cases of forms:
  - Is the input an e-mail address?

Problems

- Analysis and generation are typical tasks in informatics:
  - In communication different standards are used (HTML, XML, json, YAML, etc). WE have to generate on sending side, and analyze on receiving side.
- In the cases of forms:
  - Is the input an e-mail address?
  - Does the password contain numbers?

Definition of a formal language

Definition

- *Let Σ be a finite non empty set! The memners of Σ are characters, the set Σ is an abc.*

Definition of a formal language

Definition

- *Let Σ be a finite non empty set! The memners of Σ are characters, the set Σ is an abc.*
- *The members of the set Σ^+ are finite non empty sequences from the characters of Σ ($w = a_1 \dots a_n, n \geq 1, a_i \in \Sigma$)*
  the set Σ^+ is the set of finite non empty words over Σ.

Definition of a formal language

Definition

- *Let Σ be a finite non empty set! The memners of Σ are characters, the set Σ is an abc.*
- *The members of the set Σ^+ are finite non empty sequences from the characters of Σ ($w = a_1 \dots a_n, n \geq 1, a_i \in \Sigma$)*
  the set Σ^+ is the set of finite non empty words over Σ.
- *If $w \in \Sigma^+$, then the length of the word w is the number of its characters($|a_1 \dots a_n| = n$).*

Definition of a formal language

Definition

- *Let Σ be a finite non empty set! The memners of Σ are characters, the set Σ is an abc.*
- *The members of the set Σ^+ are finite non empty sequences from the characters of Σ ($w = a_1 \dots a_n, n \geq 1, a_i \in \Sigma$)*
  the set Σ^+ is the set of finite non empty words over Σ.
- *If $w \in \Sigma^+$, then the length of the word w is the number of its characters($|a_1 \dots a_n| = n$).*
- *ε is the empty word (It does not contain any chatracter, and its length is 0.*

Definition of a formal language

Definition

- *Let Σ be a finite non empty set! The memners of Σ are characters, the set Σ is an abc.*
- *The members of the set Σ^+ are finite non empty sequences from the characters of Σ ($w = a_1 \dots a_n, n \geq 1, a_i \in \Sigma$)*
  the set Σ^+ is the set of finite non empty words over Σ.
- *If $w \in \Sigma^+$, then the length of the word w is the number of its characters($|a_1 \dots a_n| = n$).*
- *ε is the empty word (It does not contain any chatracter, and its length is 0.*
- *The set $\Sigma^\star = \Sigma^+ \cup \{\varepsilon\}$ is the set of words over Σ.*

Definition of a formal language

Definition

- *Let Σ be a finite non empty set! The memners of Σ are characters, the set Σ is an abc.*
- *The members of the set Σ^+ are finite non empty sequences from the characters of Σ ($w = a_1 \dots a_n, n \geq 1, a_i \in \Sigma$)*
  the set Σ^+ is the set of finite non empty words over Σ.
- *If $w \in \Sigma^+$, then the length of the word w is the number of its characters($|a_1 \dots a_n| = n$).*
- *ε is the empty word (It does not contain any chatracter, and its length is 0.*
- *The set $\Sigma^\star = \Sigma^+ \cup \{\varepsilon\}$ is the set of words over Σ.*
- *If $L \subseteq \Sigma^\star$ then the set L is a (formal) language over the abc Σ.*

Definition of a formal language

Definition

- *Let Σ be a finite non empty set! The memners of Σ are characters, the set Σ is an abc.*
- *The members of the set Σ^+ are finite non empty sequences from the characters of Σ ($w = a_1 \dots a_n, n \geq 1, a_i \in \Sigma$)*
  the set Σ^+ is the set of finite non empty words over Σ.
- *If $w \in \Sigma^+$, then the length of the word w is the number of its characters($|a_1 \dots a_n| = n$).*
- *ε is the empty word (It does not contain any chatracter, and its length is 0.*
- *The set $\Sigma^\star = \Sigma^+ \cup \{\varepsilon\}$ is the set of words over Σ.*
- *If $L \subseteq \Sigma^\star$ then the set L is a (formal) language over the abc Σ.*
- *$L_\emptyset$ is the empty language (it has no any word).*

Operations of formal languages

Definition

Let L, L_1 és L_2 be formal languages over Σ. Then

- $L_1 \cup L_2 = \{x \mid x \in L_1$ *vagy* $x \in L_2\}$ *(union);*

Operations of formal languages

Definition

Let L, L_1 és L_2 be formal languages over Σ. Then

- $L_1 \cup L_2 = \{x \mid x \in L_1$ *vagy* $x \in L_2\}$ *(union);*
- $L_1 \cap L_2 = \{x \mid x \in L_1$ *és* $x \in L_2\}$ *(intersection);*

Operations of formal languages

Definition

Let L, L_1 és L_2 be formal languages over Σ. Then

- $L_1 \cup L_2 = \{x \mid x \in L_1$ *vagy* $x \in L_2\}$ *(union);*
- $L_1 \cap L_2 = \{x \mid x \in L_1$ *és* $x \in L_2\}$ *(intersection);*
- $L_1 \circ L_2 = \{xy \mid x \in L_1$ *és* $y \in L_2\}$ *concatenation;*

Operations of formal languages

Definition

Let L, L_1 és L_2 be formal languages over Σ. Then

- $L_1 \cup L_2 = \{x \mid x \in L_1$ *vagy* $x \in L_2\}$ *(union);*
- $L_1 \cap L_2 = \{x \mid x \in L_1$ *és* $x \in L_2\}$ *(intersection);*
- $L_1 \circ L_2 = \{xy \mid x \in L_1$ *és* $y \in L_2\}$ *concatenation;*
- *Power of a formal language is concatenation with itself:*

Operations of formal languages

Definition

Let L, L_1 és L_2 be formal languages over Σ. Then

- $L_1 \cup L_2 = \{x \mid x \in L_1$ *vagy* $x \in L_2\}$ *(union);*
- $L_1 \cap L_2 = \{x \mid x \in L_1$ *és* $x \in L_2\}$ *(intersection);*
- $L_1 \circ L_2 = \{xy \mid x \in L_1$ *és* $y \in L_2\}$ *concatenation;*
- *Power of a formal language is concatenation with itself:*
  - $L^0 = \{\varepsilon\}$

Operations of formal languages

Definition

Let L, L_1 és L_2 be formal languages over Σ. Then

- $L_1 \cup L_2 = \{x \mid x \in L_1$ *vagy* $x \in L_2\}$ *(union);*
- $L_1 \cap L_2 = \{x \mid x \in L_1$ *és* $x \in L_2\}$ *(intersection);*
- $L_1 \circ L_2 = \{xy \mid x \in L_1$ *és* $y \in L_2\}$ *concatenation;*
- *Power of a formal language is concatenation with itself:*
  - $L^0 = \{\varepsilon\}$
  - $L^{n+1} = L^n \circ L$

Operations of formal languages

Definition

Let L, L_1 és L_2 be formal languages over Σ. Then

- $L_1 \cup L_2 = \{x \mid x \in L_1$ *vagy* $x \in L_2\}$ *(union);*
- $L_1 \cap L_2 = \{x \mid x \in L_1$ *és* $x \in L_2\}$ *(intersection);*
- $L_1 \circ L_2 = \{xy \mid x \in L_1$ *és* $y \in L_2\}$ *concatenation;*
- *Power of a formal language is concatenation with itself:*
  - $L^0 = \{\varepsilon\}$
  - $L^{n+1} = L^n \circ L$
- *Transitive closure (Kleene-star operation):* $L^\star = \cup_{i=0}^{\infty} L^i$, *i.e.*

$$L^\star = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ és } x_i \in L\}$$

Remark

- $L_\varepsilon^\star = \{\varepsilon\}$, *ahol* $L_\varepsilon = \{\varepsilon\}$.

Examples

Let $\Sigma = \{a, b, c\}$, $L_1 = \{a, bb, ccc\}$ and $L_2 = \{ac, bc\}$. Then

Remark

- $L_\varepsilon^\star = \{\varepsilon\}$, *ahol* $L_\varepsilon = \{\varepsilon\}$.
- $L_\emptyset^\star = \{\varepsilon\}$, *mert definíció szerint* $L_\emptyset^0 = \{\varepsilon\}$

Examples

Let $\Sigma = \{a, b, c\}$, $L_1 = \{a, bb, ccc\}$ and $L_2 = \{ac, bc\}$. Then

Remark

- $L_\varepsilon^\star = \{\varepsilon\}$, *ahol* $L_\varepsilon = \{\varepsilon\}$.
- $L_\emptyset^\star = \{\varepsilon\}$, *mert definíció szerint* $L_\emptyset^0 = \{\varepsilon\}$

Examples

Let $\Sigma = \{a, b, c\}$, $L_1 = \{a, bb, ccc\}$ and $L_2 = \{ac, bc\}$. Then

- $L_1 \cup L_2 = \{a, ac, bb, bc, ccc\}$;

Remark

- $L_\varepsilon^\star = \{\varepsilon\}$, *ahol* $L_\varepsilon = \{\varepsilon\}$.
- $L_\emptyset^\star = \{\varepsilon\}$, *mert definíció szerint* $L_\emptyset^0 = \{\varepsilon\}$

Examples

Let $\Sigma = \{a, b, c\}$, $L_1 = \{a, bb, ccc\}$ and $L_2 = \{ac, bc\}$. Then

- $L_1 \cup L_2 = \{a, ac, bb, bc, ccc\}$;
- $L_1 \circ L_2 = \{aac, abc, bbac, bbbc, cccac, cccbc\}$;

Remark

- $L_\varepsilon^\star = \{\varepsilon\}$, *ahol* $L_\varepsilon = \{\varepsilon\}$.
- $L_\emptyset^\star = \{\varepsilon\}$, *mert definíció szerint* $L_\emptyset^0 = \{\varepsilon\}$

Examples

Let $\Sigma = \{a, b, c\}$, $L_1 = \{a, bb, ccc\}$ and $L_2 = \{ac, bc\}$. Then

- $L_1 \cup L_2 = \{a, ac, bb, bc, ccc\}$;
- $L_1 \circ L_2 = \{aac, abc, bbac, bbbc, cccac, cccbc\}$;
- $L_1^\star = \{\varepsilon, a, bb, ccc, aa, abb, accc, bba, bbbb, bbccc, \dots\}$;

Remark

- $L_\varepsilon^\star = \{\varepsilon\}$, *ahol* $L_\varepsilon = \{\varepsilon\}$.
- $L_\emptyset^\star = \{\varepsilon\}$, *mert definíció szerint* $L_\emptyset^0 = \{\varepsilon\}$

Examples

Let $\Sigma = \{a, b, c\}$, $L_1 = \{a, bb, ccc\}$ and $L_2 = \{ac, bc\}$. Then

- $L_1 \cup L_2 = \{a, ac, bb, bc, ccc\}$;
- $L_1 \circ L_2 = \{aac, abc, bbac, bbbc, cccac, cccbc\}$;
- $L_1^\star = \{\varepsilon, a, bb, ccc, aa, abb, accc, bba, bbbb, bbccc, \dots\}$;
- Σ is a formal language over itself, and its transitive closure is $\Sigma^\star$ (i.e. the set of words over the abc Σ.

Defining formal languages

- By specification: see in the previous examples:

Defining formal languages

- By specification: see in the previous examples:
  - in this cases formal langueges may have only finite many words.

Defining formal languages

- By specification: see in the previous examples:
  - in this cases formal langueges may have only finite many words.
- By generalization (by inductive definition):

Defining formal languages

- By specification: see in the previous examples:
  - in this cases formal langueges may have only finite many words.
- By generalization (by inductive definition):
  - generative grammars (Chomsky type classes of formal languages).

Defining formal languages

- By specification: see in the previous examples:
  - in this cases formal langueges may have only finite many words.
- By generalization (by inductive definition):
  - generative grammars (Chomsky type classes of formal languages).
- By automaton:

Defining formal languages

- By specification: see in the previous examples:
  - in this cases formal langueges may have only finite many words.
- By generalization (by inductive definition):
  - generative grammars (Chomsky type classes of formal languages).
- By automaton:
  - an automation decides whether a given word is a member of our language.

Defining formal languages

- By specification: see in the previous examples:
  - in this cases formal langueges may have only finite many words.
- By generalization (by inductive definition):
  - generative grammars (Chomsky type classes of formal languages).
- By automaton:
  - an automation decides whether a given word is a member of our language.
- By expressions which characterize the word of a language:

Defining formal languages

- By specification: see in the previous examples:
  - in this cases formal langueges may have only finite many words.
- By generalization (by inductive definition):
  - generative grammars (Chomsky type classes of formal languages).
- By automaton:
  - an automation decides whether a given word is a member of our language.
- By expressions which characterize the word of a language:
  - for example the system of regular expressions.

Defining formal languages

- By specification: see in the previous examples:
  - in this cases formal langueges may have only finite many words.
- By generalization (by inductive definition):
  - generative grammars (Chomsky type classes of formal languages).
- By automaton:
  - an automation decides whether a given word is a member of our language.
- By expressions which characterize the word of a language:
  - for example the system of regular expressions.

Defining formal languages

- By specification: see in the previous examples:
  - in this cases formal langueges may have only finite many words.
- By generalization (by inductive definition):
  - generative grammars (Chomsky type classes of formal languages).
- By automaton:
  - an automation decides whether a given word is a member of our language.
- By expressions which characterize the word of a language:
  - for example the system of regular expressions.

Remark

The formal possibilities are useful in giving formal languages with different properties.

Regular expressions

Problem

- Let $\Sigma = \{0, 1\}$ be an abc. How to give the language L (how to express the language L) with the help of other (possible simpler) languages which contains the words with subwords 010!

Regular expressions

Problem

- Let $\Sigma = \{0, 1\}$ be an abc. How to give the language L (how to express the language L) with the help of other (possible simpler) languages which contains the words with subwords 010!
- 'Informal' solution': If $w_1, w_2 \in \Sigma^\star$, then $w_1 010 w_2 \in L$.

Regular expressions

Problem

- Let $\Sigma = \{0, 1\}$ be an abc. How to give the language L (how to express the language L) with the help of other (possible simpler) languages which contains the words with subwords 010!
- 'Informal' solution': If $w_1, w_2 \in \Sigma^\star$, then $w_1 010 w_2 \in L$.
  - How to define the words w_1 és w_2?

Regular expressions

Problem

- Let $\Sigma = \{0, 1\}$ be an abc. How to give the language L (how to express the language L) with the help of other (possible simpler) languages which contains the words with subwords 010!
- 'Informal' solution': If $w_1, w_2 \in \Sigma^\star$, then $w_1 010 w_2 \in L$.
  - How to define the words w_1 és w_2?
  - We know that w_1 és w_2 contains characters 0 and 1 in non empty cases.

Regular expressions

Problem

- Let $\Sigma = \{0, 1\}$ be an abc. How to give the language L (how to express the language L) with the help of other (possible simpler) languages which contains the words with subwords 010!
- 'Informal' solution': If $w_1, w_2 \in \Sigma^\star$, then $w_1 010 w_2 \in L$.
  - How to define the words w_1 és w_2?
  - We know that w_1 és w_2 contains characters 0 and 1 in non empty cases.
  - Let $L_0 = \{0\}$ and $L_1 = \{1\}$. Then $L_0 \cup L_1 = \{0, 1\}$, and $(L_0 \cup L_1)^\star$ contains the possible words w_1 és w_2 (with the empty word).

Regular expressions

Problem

- Let $\Sigma = \{0, 1\}$ be an abc. How to give the language L (how to express the language L) with the help of other (possible simpler) languages which contains the words with subwords 010!
- 'Informal' solution': If $w_1, w_2 \in \Sigma^\star$, then $w_1 010 w_2 \in L$.
  - How to define the words w_1 és w_2?
  - We know that w_1 és w_2 contains characters 0 and 1 in non empty cases.
  - Let $L_0 = \{0\}$ and $L_1 = \{1\}$. Then $L_0 \cup L_1 = \{0, 1\}$, and $(L_0 \cup L_1)^\star$ contains the possible words w_1 és w_2 (with the empty word).
  - If $L_{010} = \{010\}$, then $L = (L_0 \cup L_1)^\star \circ L_{010} \circ (L_0 \cup L_1)^\star$.

Regular expressions

Problem

- Let $\Sigma = \{0, 1\}$ be an abc. How to give the language L (how to express the language L) with the help of other (possible simpler) languages which contains the words with subwords 010!
- 'Informal' solution': If $w_1, w_2 \in \Sigma^\star$, then $w_1 010 w_2 \in L$.
  - How to define the words w_1 és w_2?
  - We know that w_1 és w_2 contains characters 0 and 1 in non empty cases.
  - Let $L_0 = \{0\}$ and $L_1 = \{1\}$. Then $L_0 \cup L_1 = \{0, 1\}$, and $(L_0 \cup L_1)^\star$ contains the possible words w_1 és w_2 (with the empty word).
  - If $L_{010} = \{010\}$, then $L = (L_0 \cup L_1)^\star \circ L_{010} \circ (L_0 \cup L_1)^\star$.
  - L_0, L_1, L_{010} are languages with only one word!

Regular expressions

Problem

- Let $\Sigma = \{0, 1\}$ be an abc. How to give the language L (how to express the language L) with the help of other (possible simpler) languages which contains the words with subwords 010!
- 'Informal' solution': If $w_1, w_2 \in \Sigma^\star$, then $w_1 010 w_2 \in L$.
  - How to define the words w_1 és w_2?
  - We know that w_1 és w_2 contains characters 0 and 1 in non empty cases.
  - Let $L_0 = \{0\}$ and $L_1 = \{1\}$. Then $L_0 \cup L_1 = \{0, 1\}$, and $(L_0 \cup L_1)^\star$ contains the possible words w_1 és w_2 (with the empty word).
  - If $L_{010} = \{010\}$, then $L = (L_0 \cup L_1)^\star \circ L_{010} \circ (L_0 \cup L_1)^\star$.
  - L_0, L_1, L_{010} are languages with only one word!
  - On an other way: $(0 + 1)^\star \cdot 010 \cdot (0 + 1)^\star$

Examples for regular expressions

Examples

- Define the language over abc $\{0, 1\}$ containing words with subwords 000 or 111!

Examples for regular expressions

Examples

- Define the language over abc $\{0, 1\}$ containing words with subwords 000 or 111!
- $(0 + 1)^\star \cdot (000 + 111) \cdot (0 + 1)^\star$

Examples for regular expressions

Examples

- Define the language over abc $\{0, 1\}$ containing words with subwords 000 or 111!
- $(0 + 1)^\star \cdot (000 + 111) \cdot (0 + 1)^\star$
- Define the language over abc $\{0, 1\}$ containing words in which the number of character 1 can be divided by 5!

Examples for regular expressions

Examples

- Define the language over abc $\{0, 1\}$ containing words with subwords 000 or 111!
- $(0 + 1)^* \cdot (000 + 111) \cdot (0 + 1)^*$
- Define the language over abc $\{0, 1\}$ containing words in which the number of character 1 can be divided by 5!
- $0^* + (0^* \cdot 1 \cdot 0^* \cdot 1 \cdot 0^* \cdot 1 \cdot 0^* \cdot 1 \cdot 0^* \cdot 1 \cdot 0^*)^*$

Regular expressions

Definition

Let Σ be an abc such that $\emptyset, \varepsilon, +, \cdot, ^\star, (,) \notin \Sigma$. The set $\mathcal{R}$ of regular expressions over the abc Σ can be defined by the following inductive expressions:

1. $\emptyset \in \mathcal{R}$

Regular expressions

Definition

Let Σ be an abc such that $\emptyset, \varepsilon, +, \cdot, ^\star, (,) \notin \Sigma$. The set $\mathcal{R}$ of regular expressions over the abc Σ can be defined by the following inductive expressions:

1. $\emptyset \in \mathcal{R}$
2. $\varepsilon \in \mathcal{R}$

Regular expressions

Definition

Let Σ be an abc such that $\emptyset, \varepsilon, +, \cdot, ^\star, (,) \notin \Sigma$. The set $\mathcal{R}$ of regular expressions over the abc Σ can be defined by the following inductive expressions:

1. $\emptyset \in \mathcal{R}$
2. $\varepsilon \in \mathcal{R}$
3. *If $a \in \Sigma$, then $a \in \mathcal{R}$.*

Regular expressions

Definition

Let Σ be an abc such that $\emptyset, \varepsilon, +, \cdot, ^\star, (,) \notin \Sigma$. The set $\mathcal{R}$ of regular expressions over the abc Σ can be defined by the following inductive expressions:

1. $\emptyset \in \mathcal{R}$
2. $\varepsilon \in \mathcal{R}$
3. *If $a \in \Sigma$, then $a \in \mathcal{R}$.*
4. *If $R_1 \in \mathcal{R}$ and $R_2 \in \mathcal{R}$, then $(R_1 + R_2) \in \mathcal{R}$.*

Regular expressions

Definition

Let Σ be an abc such that $\emptyset, \varepsilon, +, \cdot, ^\star, (,) \notin \Sigma$. The set $\mathcal{R}$ of regular expressions over the abc Σ can be defined by the following inductive expressions:

1. $\emptyset \in \mathcal{R}$
2. $\varepsilon \in \mathcal{R}$
3. *If $a \in \Sigma$, then $a \in \mathcal{R}$.*
4. *If $R_1 \in \mathcal{R}$ and $R_2 \in \mathcal{R}$, then $(R_1 + R_2) \in \mathcal{R}$.*
5. *If $R_1 \in \mathcal{R}$ and $R_2 \in \mathcal{R}$, then $(R_1 \cdot R_2) \in \mathcal{R}$.*

Regular expressions

Definition

Let Σ be an abc such that $\emptyset, \varepsilon, +, \cdot, ^\star, (,) \notin \Sigma$. The set $\mathcal{R}$ of regular expressions over the abc Σ can be defined by the following inductive expressions:

1. $\emptyset \in \mathcal{R}$
2. $\varepsilon \in \mathcal{R}$
3. *If $a \in \Sigma$, then $a \in \mathcal{R}$.*
4. *If $R_1 \in \mathcal{R}$ and $R_2 \in \mathcal{R}$, then $(R_1 + R_2) \in \mathcal{R}$.*
5. *If $R_1 \in \mathcal{R}$ and $R_2 \in \mathcal{R}$, then $(R_1 \cdot R_2) \in \mathcal{R}$.*
6. *If $R \in \mathcal{R}$, then $R^\star \in \mathcal{R}$.*

Formal languages described by regular expressions

Definition

Let $\mathcal{R}$ be the set of regular expressions over abc Σ and $R \in \mathcal{R}$. The language described by the regular expression R is given by the following inductive definition:

1. *If $R = \emptyset$, then $L_R = L_\emptyset$.*

Formal languages described by regular expressions

Definition

Let $\mathcal{R}$ be the set of regular expressions over abc Σ and $R \in \mathcal{R}$. The language described by the regular expression R is given by the following inductive definition:

1. *If $R = \emptyset$, then $L_R = L_\emptyset$.*
2. *If $R = \varepsilon$, then $L_R = L_\varepsilon = \{\varepsilon\}$.*

Formal languages described by regular expressions

Definition

Let $\mathcal{R}$ be the set of regular expressions over abc Σ and $R \in \mathcal{R}$. The language described by the regular expression R is given by the following inductive definition:

1. *If $R = \emptyset$, then $L_R = L_\emptyset$.*
2. *If $R = \varepsilon$, then $L_R = L_\varepsilon = \{\varepsilon\}$.*
3. *If $R = a, a \in \Sigma$, then $L_R = \{a\}$.*

Formal languages described by regular expressions

Definition

Let $\mathcal{R}$ be the set of regular expressions over abc Σ and $R \in \mathcal{R}$. The language described by the regular expression R is given by the following inductive definition:

1. *If $R = \emptyset$, then $L_R = L_\emptyset$.*
2. *If $R = \varepsilon$, then $L_R = L_\varepsilon = \{\varepsilon\}$.*
3. *If $R = a, a \in \Sigma$, then $L_R = \{a\}$.*
4. *If $R = (R_1 + R_2)$, then $L_R = L_{R_1} \cup L_{R_2}$.*

Formal languages described by regular expressions

Definition

Let $\mathcal{R}$ be the set of regular expressions over abc Σ and $R \in \mathcal{R}$. The language described by the regular expression R is given by the following inductive definition:

1. *If $R = \emptyset$, then $L_R = L_\emptyset$.*
2. *If $R = \varepsilon$, then $L_R = L_\varepsilon = \{\varepsilon\}$.*
3. *If $R = a, a \in \Sigma$, then $L_R = \{a\}$.*
4. *If $R = (R_1 + R_2)$, then $L_R = L_{R_1} \cup L_{R_2}$.*
5. *If $R = (R_1 \cdot R_2)$, then $L_R = L_{R_1} \circ L_{R_2}$.*

Formal languages described by regular expressions

Definition

Let $\mathcal{R}$ be the set of regular expressions over abc Σ and $R \in \mathcal{R}$. The language described by the regular expression R is given by the following inductive definition:

1. *If $R = \emptyset$, then $L_R = L_\emptyset$.*
2. *If $R = \varepsilon$, then $L_R = L_\varepsilon = \{\varepsilon\}$.*
3. *If $R = a, a \in \Sigma$, then $L_R = \{a\}$.*
4. *If $R = (R_1 + R_2)$, then $L_R = L_{R_1} \cup L_{R_2}$.*
5. *If $R = (R_1 \cdot R_2)$, then $L_R = L_{R_1} \circ L_{R_2}$.*
6. *If $R = R_1^\star$, then $L_R = L_{R_1}^\star$.*

Regular language

Definition

A language L over abc Σ is regular, if there is a regular expression R over abc Σ such that $L_R = L$

Regular language

Definition

A language L over abc Σ is regular, if there is a regular expression R over abc Σ such that $L_R = L$

Theorem

If languages L_1 és L_2 over abc Σ are regular, then languages $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 \backslash L_2$, $\overline{L_1} (= \Sigma^\star \backslash L_1)$, $L_1 \circ L_2$ és $L_1^\star$ are regular.

Finite automata

- The simplest automaton: it is only able to read and it has no memory.

Definition

A finite automaton is an ordered 5-tuple: $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, *where*

Finite automata

- The simplest automaton: it is only able to read and it has no memory.
- The set of states is finite.

Definition

A finite automaton is an ordered 5-tuple: $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, *where*

Finite automata

- The simplest automaton: it is only able to read and it has no memory.
- The set of states is finite.

Definition

A finite automaton is an ordered 5-tuple: $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, *where*

- Q: *the finite set (internal) states,*

Finite automata

- The simplest automaton: it is only able to read and it has no memory.
- The set of states is finite.

Definition

A finite automaton is an ordered 5-tuple: $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, *where*

- Q: *the finite set (internal) states,*
- Σ: *the finite (external) abc (the abc of the string to be analyzed),*

Finite automata

- The simplest automaton: it is only able to read and it has no memory.
- The set of states is finite.

Definition

A finite automaton is an ordered 5-tuple: $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, *where*

- Q: *the finite set (internal) states,*
- Σ: *the finite (external) abc (the abc of the string to be analyzed),*
- $\delta : Q \times \Sigma \to Q$: *transition function,*

Finite automata

- The simplest automaton: it is only able to read and it has no memory.
- The set of states is finite.

Definition

A finite automaton is an ordered 5-tuple: $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, *where*

- Q: *the finite set (internal) states,*
- Σ: *the finite (external) abc (the abc of the string to be analyzed),*
- $\delta : Q \times \Sigma \to Q$: *transition function,*
- $q_0 \in Q$: *initial state*

Finite automata

- The simplest automaton: it is only able to read and it has no memory.
- The set of states is finite.

Definition

A finite automaton is an ordered 5-tuple: $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, *where*

- Q: *the finite set (internal) states,*
- Σ: *the finite (external) abc (the abc of the string to be analyzed),*
- $\delta : Q \times \Sigma \to Q$: *transition function,*
- $q_0 \in Q$: *initial state*
- $F \subseteq Q$: *the set of accepting states.*

Remark.

- A transition function determines the new state of the automaton after reading a character.

Remark.

- A transition function determines the new state of the automaton after reading a character.
- An automaton accepts a sequence of characters (a string), if

Remark.

- A transition function determines the new state of the automaton after reading a character.
- An automaton accepts a sequence of characters (a string), if
  - it reads the whole string and

Remark.

- A transition function determines the new state of the automaton after reading a character.
- An automaton accepts a sequence of characters (a string), if
  - it reads the whole string and
  - after reading the last character it is in accepting state.

Let $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, where

- $Q = \{q_0, q_1, q_2, q_3, q_c\}$ is the finite set of (internal) states;

Let $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, where

- $Q = \{q_0, q_1, q_2, q_3, q_c\}$ is the finite set of (internal) states;
- $\Sigma = \{a, b, c\}$ is the finite (external) abc (the abc of the string to be analyzed);

Let $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, where

- $Q = \{q_0, q_1, q_2, q_3, q_c\}$ is the finite set of (internal) states;
- $\Sigma = \{a, b, c\}$ is the finite (external) abc (the abc of the string to be analyzed);
- $\delta(q_0, a) = q_1, \delta(q_1, a) = q_1, \delta(q_1, b) = q_2, \delta(q_2, b) = q_2, \delta(q_2, c) = q_3, \delta(q_3, c) = q_3$ and the value of δ is q_r otherwise (the transition function);

Let $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, where

- $Q = \{q_0, q_1, q_2, q_3, q_c\}$ is the finite set of (internal) states;
- $\Sigma = \{a, b, c\}$ is the finite (external) abc (the abc of the string to be analyzed);
- $\delta(q_0, a) = q_1, \delta(q_1, a) = q_1, \delta(q_1, b) = q_2, \delta(q_2, b) = q_2, \delta(q_2, c) = q_3, \delta(q_3, c) = q_3$ and the value of δ is q_r otherwise (the transition function);
- $q_0 \in Q$ is the initial state;

Let $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, where

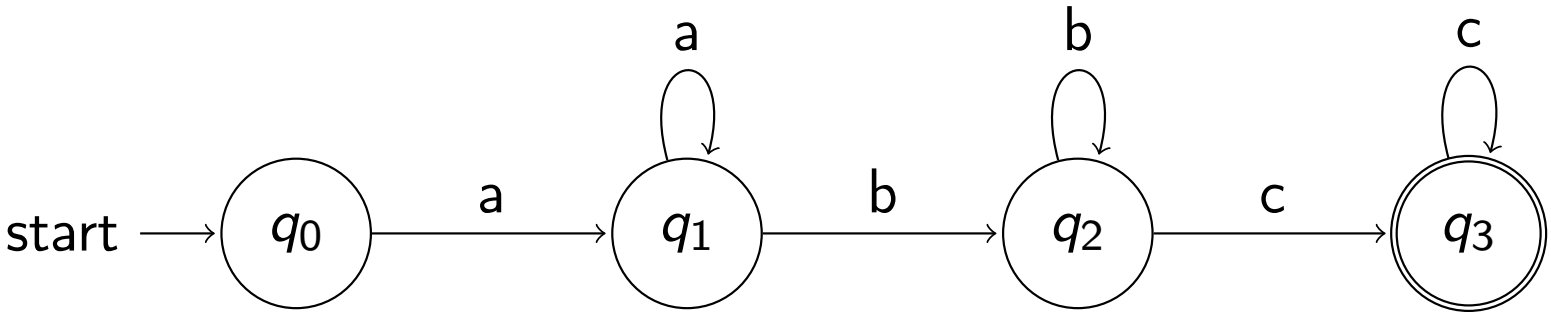
- $Q = \{q_0, q_1, q_2, q_3, q_c\}$ is the finite set of (internal) states;
- $\Sigma = \{a, b, c\}$ is the finite (external) abc (the abc of the string to be analyzed);
- $\delta(q_0, a) = q_1, \delta(q_1, a) = q_1, \delta(q_1, b) = q_2, \delta(q_2, b) = q_2, \delta(q_2, c) = q_3, \delta(q_3, c) = q_3$ and the value of δ is q_r otherwise (the transition function);
- $q_0 \in Q$ is the initial state;
- $F = \{q_3\}$ i.e. q_3 is the accepting state.

Let $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, where

- $Q = \{q_0, q_1, q_2, q_3, q_c\}$ is the finite set of (internal) states;
- $\Sigma = \{a, b, c\}$ is the finite (external) abc (the abc of the string to be analyzed);
- $\delta(q_0, a) = q_1, \delta(q_1, a) = q_1, \delta(q_1, b) = q_2, \delta(q_2, b) = q_2, \delta(q_2, c) = q_3, \delta(q_3, c) = q_3$ and the value of δ is q_r otherwise (the transition function);
- $q_0 \in Q$ is the initial state;
- $F = \{q_3\}$ i.e. q_3 is the accepting state.

Let $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, where

- $Q = \{q_0, q_1, q_2, q_3, q_c\}$ is the finite set of (internal) states;
- $\Sigma = \{a, b, c\}$ is the finite (external) abc (the abc of the string to be analyzed);
- $\delta(q_0, a) = q_1, \delta(q_1, a) = q_1, \delta(q_1, b) = q_2, \delta(q_2, b) = q_2, \delta(q_2, c) = q_3, \delta(q_3, c) = q_3$ and the value of δ is q_r otherwise (the transition function);
- $q_0 \in Q$ is the initial state;
- $F = \{q_3\}$ i.e. q_3 is the accepting state.

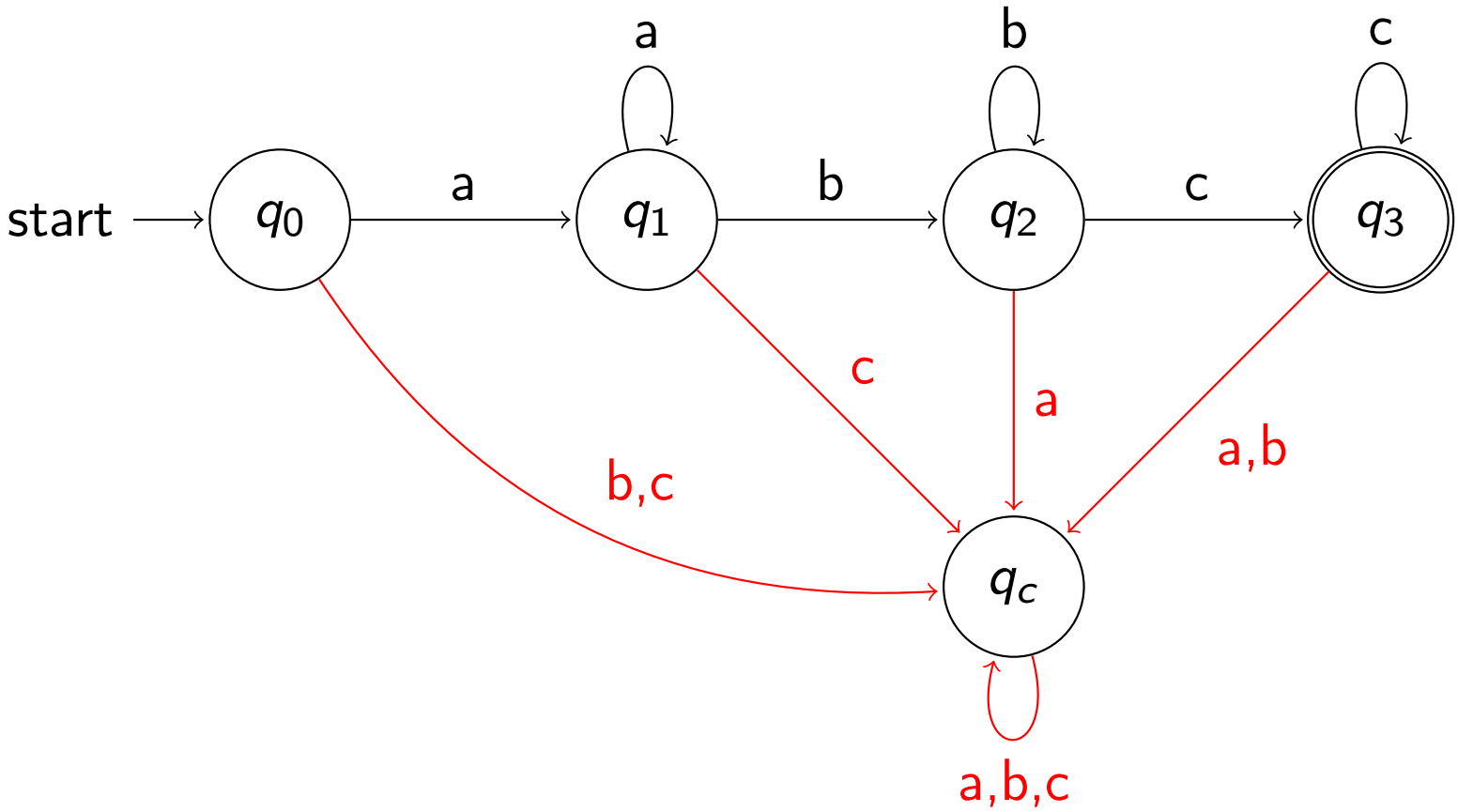


$a^i b^j c^k$, ahol $i, j, k > 0$

- According to our definition the function δ is total and not partial. In order to make the figure simpler the missing arrows show to a corrupted state.

- According to our definition the function δ is total and not partial. In order to make the figure simpler the missing arrows show to a corrupted state.

- According to our definition the function δ is total and not partial. In order to make the figure simpler the missing arrows show to a corrupted state.



- Internal states are represented by circles, the transition function is denoted by arrows and the characters of internal abc. The initial state is the state with an arrow without a character. The final states is the doubled circle.

Accepting automata

Definition

$A = \langle Q, \Sigma, \delta, q_0, F \rangle$ is an automaton, and $w = a_1 a_2 \dots a_n \in \Sigma^\star$ is a word. The automaton accepts the word w if

$$\delta(\dots(\delta(\delta(q_0, a_1), a_2)\dots), a_n) \in F.$$

Accepting automata

Definition

$A = \langle Q, \Sigma, \delta, q_0, F \rangle$ is an automaton, and $w = a_1 a_2 \dots a_n \in \Sigma^\star$ is a word. The automaton accepts the word w if

$$\delta(\dots(\delta(\delta(q_0, a_1), a_2)\dots), a_n) \in F.$$

Definition

Let $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ be an automaton If $L = \{w \in \Sigma^\star | A \text{ accepts } w\text{-t}\}$, then A accepts (recognizes) the language L. In denoting: $L = L_A$

Accepting automata

Definition

$A = \langle Q, \Sigma, \delta, q_0, F \rangle$ is an automaton, and $w = a_1 a_2 \dots a_n \in \Sigma^\star$ is a word. The automaton accepts the word w if

$$\delta(\dots(\delta(\delta(q_0, a_1), a_2)\dots), a_n) \in F.$$

Definition

Let $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ be an automaton If $L = \{w \in \Sigma^\star | A \text{ accepts } w\text{-t}\}$, then A accepts (recognizes) the language L. In denoting: $L = L_A$

Theorem

Finite automata accept regular languages, i.e. if L is a regular language, then there is an automaton A such that $L = L_A$