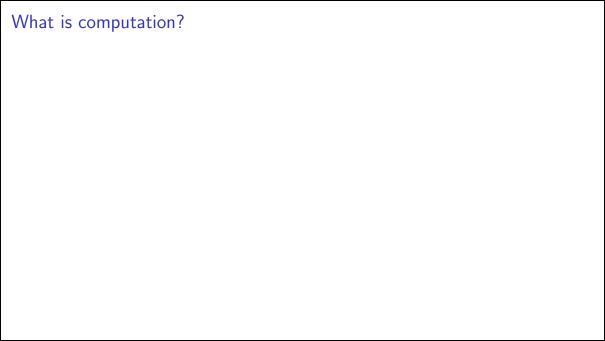
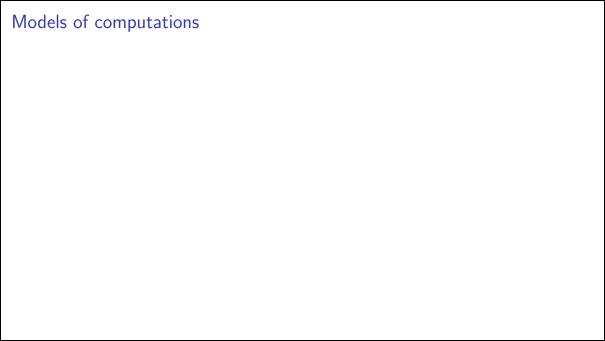
01204211 Discrete Mathematics Lecture 7a: Languages and regular expressions¹

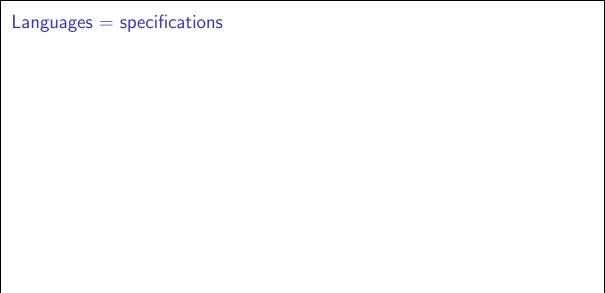
Jittat Fakcharoenphol

August 20, 2024

¹Based on lecture notes of *Models of Computation* course by Jeff Erickson.







Formal definition: strings

Intuitively, a string is a *finite* sequence of symbols. However, to be able to formally prove properties of strings we need a precise definition.

Let a finite set Σ be the **alphabet**. (E.g., for bit strings, $\Sigma = \{0, 1\}$; for digits, $\Sigma = \{0, 1, \dots, 9\}$; for English string $\Sigma = \{a, b, \dots, z\}$.) The following is a recursive definition of strings.

Recursive definition of strings

A **string** w over alphabet Σ is either

- \blacktriangleright the empty string ε , or
- $ightharpoonup a \cdot x$ where $a \in \Sigma$ and x is a string.

The set of all strings over alphabet Σ is denoted by Σ^* .

Review: more recursive definitions

Lengths

For a string w, let |w| be the length of w defined as

$$|w| = \left\{ \begin{array}{ll} 0 & \text{when } w = \varepsilon \\ 1 + |x| & \text{when } w = a \cdot x \end{array} \right.$$

Concatenation

For strings w and z, the concatenation $w \cdot z$ is defiend recursively as

$$w \cdot z = \left\{ \begin{array}{ll} z & \text{when } w = \varepsilon \\ a \cdot (x \cdot z) & \text{when } w = a \cdot x \end{array} \right.$$

Review: proving facts about strings

Lemma 1

For strings w and x, $|w \cdot x| = |w| + |x|$.

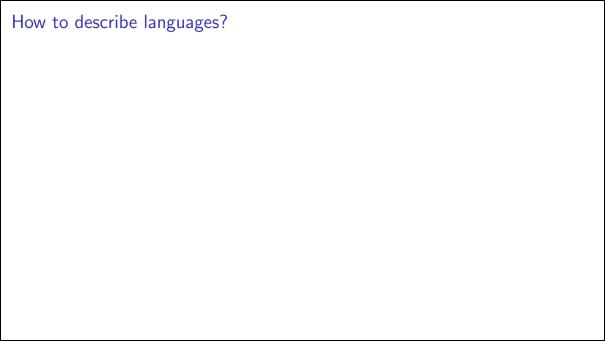
Proof.

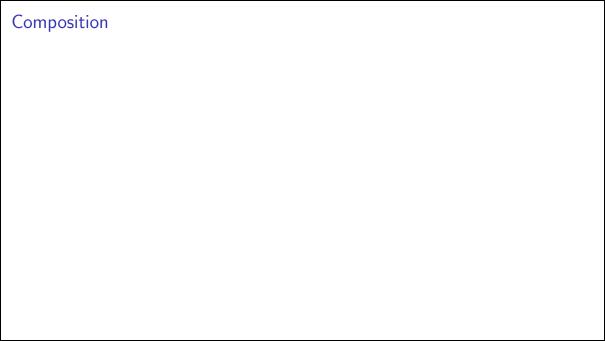
Formal languages

A **formal language** is a set of strings over some finite alphabet Σ . Examples:

Careful...

These are different languages: $\emptyset, \{\varepsilon\}$ And ε is not a language.





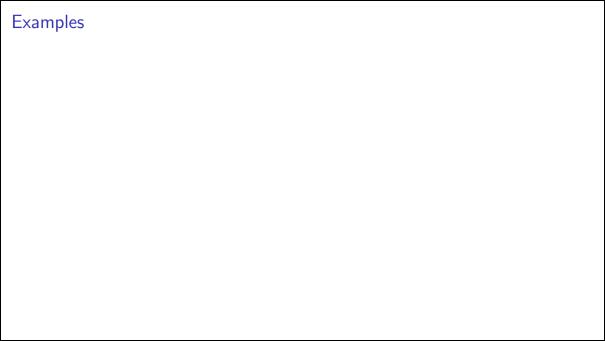
Combining languages

If A and B are languages over alphabet Σ .

- ▶ Basic set operations: $A \cup B$, $A \cap B$, $\bar{A} = \Sigma^* \setminus A$.
- ightharpoonup Concatenation: $A \cdot B$.

▶ Kleene closure or Kleene star: A^* .

Also $A^+ = A \cdot A^*$

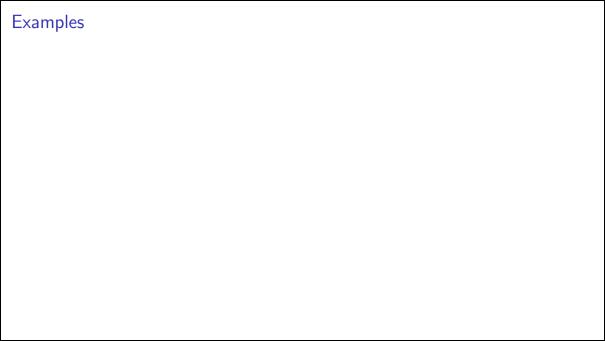


Regular languages

Definition: regular languages

A language L is regular if and only if it satisfies one of the following conditions:

- ► *L* is empty;
- ▶ L contains one string (can be the empty string ε);
- L is a union of two regular languages;
- ightharpoonup L is the concatenation of two regular languages; or
- ightharpoonup L is the Kleene closure of a regular language.



Regular expressions

Let
$$\Sigma = \{0, 1\}$$
. Consider

$$((\{01\} \cup (\{1\} \cdot (\{0\} \cup \{10\}))) \cup (\{00\} \cdot (\{1\})^*)) \cdot ((\{0\} \cdot \{0\}) \cdot \{1\}))$$

Regular expressions

Regular language

$$((\{01\} \cup (\{1\} \boldsymbol{\cdot} (\{0\} \cup \{10\}))) \cup (\{00\} \boldsymbol{\cdot} (\{1\})^*)) \boldsymbol{\cdot} ((\{0\} \boldsymbol{\cdot} \{0\}) \boldsymbol{\cdot} \{1\})$$

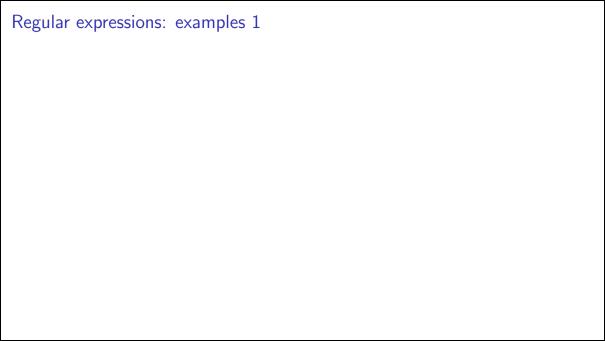
is represented as

$$(01 + 1(0 + 10) + 00(1)^*)001$$

Regular expressions

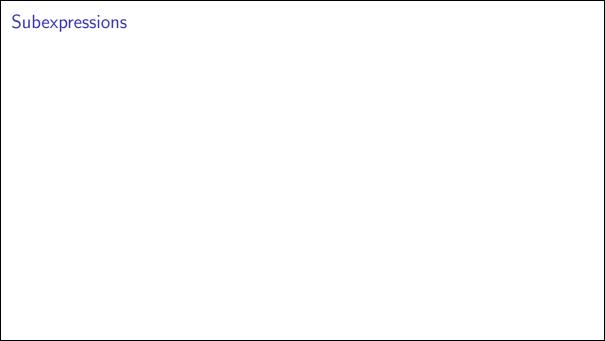
- omit braces around one-string sets
- ightharpoonup use + instead of \cup
- ▶ omit •
- ightharpoonup follow the precedence: Kleene star operator *, (implicitly), and +.

Remark: + and \cdot are associative, i.e., (A+B)+C=A+(B+C) and $(A \cdot B) \cdot C = A \cdot (B \cdot C)$.

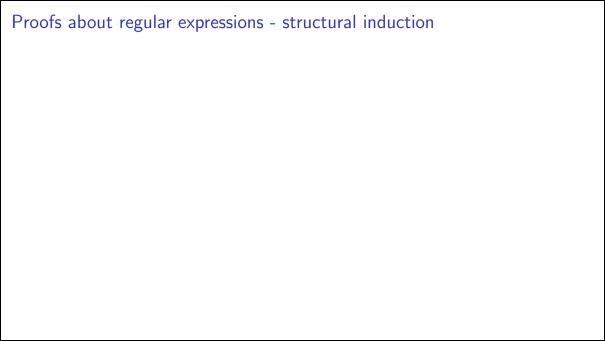


Regular expressions: examples 2

All strings over $\{0,1\}$ except 010.







Lemma 2

Every regular expression that does not use the symbol \emptyset represents a non-empty language.

Proof.

Let R be a regular expression that does not use the symbol \emptyset . We prove by (structural) induction that R represents a non-empty language.

Induction hypothesis: Every subexpression of R that does not use the symbol \emptyset represents a non-empty language.

Case 1: $R = \emptyset$.

Case 2: R is a single string.

Proof. (cont.2/4) Case 3: R = S + T for some regular expressions S and T.

Proof. (cont.3/4) Case 4: $R = S \cdot T$ for some regular expressions S and T.

Proof. (cont.4/4) Case 5: $R = S^*$ for some regular expression S.

In every case, the language ${\cal L}({\cal R})$ is non-empty.

Lemma 3

Every non-empty regular language is represented by a regular expression that does not use the symbol \emptyset .

Let R be a regular expression. We prove that if $L(R) \neq \emptyset$, then there exists a regular expression R' such that L(R) = L(R') and R' does not contain \emptyset . We prove by induction. What should the induction hypothesis be?

I.H.: For every subexpression S of R, if $L(S) \neq \emptyset$, there exists an \emptyset -free regular expression S' such that L(S) = L(S').

What are the cases that we have to consider?

- $ightharpoonup R = \emptyset$
- ightharpoonup R is a single string.
- ightharpoonup R = S + T for some regular expressions S and T.
- $ightharpoonup R = S \cdot T$ for some regular expressions S and T.
- $ightharpoonup R = S^*$ for some regular expression S.

(E-ex1-6) For string w, the reversal w^R is defined recursively as follows:

$$w^R = \left\{ \begin{array}{ll} \varepsilon & \text{if } w = \varepsilon \\ x^R \cdot a & \text{if } w = ax \text{ for some symbol } a \text{ and some string } x \end{array} \right.$$

For a language L, the reversal of L is defined as

$$L^R = \{ w^R \mid w \in L \}.$$

You may assume the following facts.

- $ightharpoonup L^* \cdot L^* = L^*$ for every language L.
 - $(w^R)^R = w$ for every string w.
 - $(x \cdot y)^R = y^R \cdot x^R \text{ for all strings } x \text{ and } y.$

Prove that $(L^R)^* \subseteq (L^*)^R$.