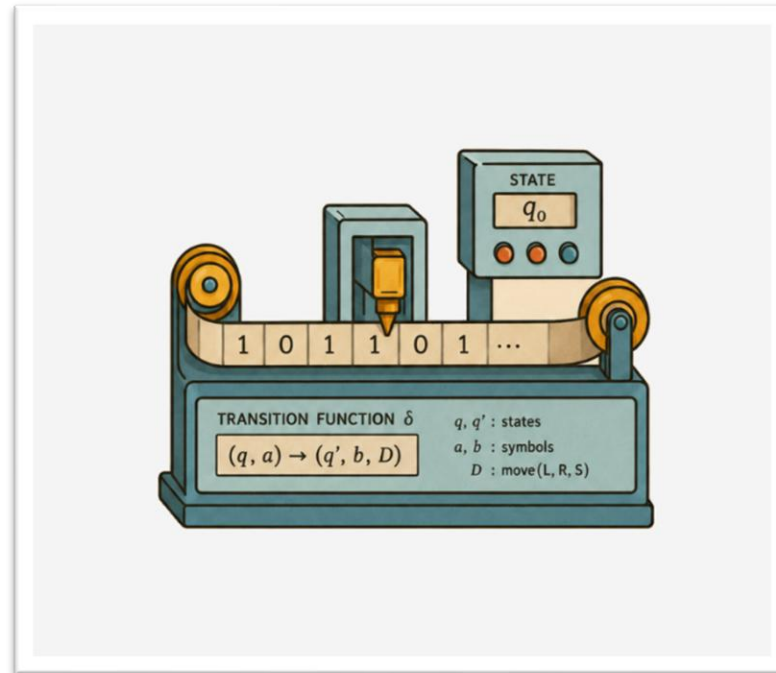


BU CS 332 – Theory of Computation

Lecture 5:

- Closure Properties
- Regular Expressions

Homework #1 is due
tonight at 11.59pm

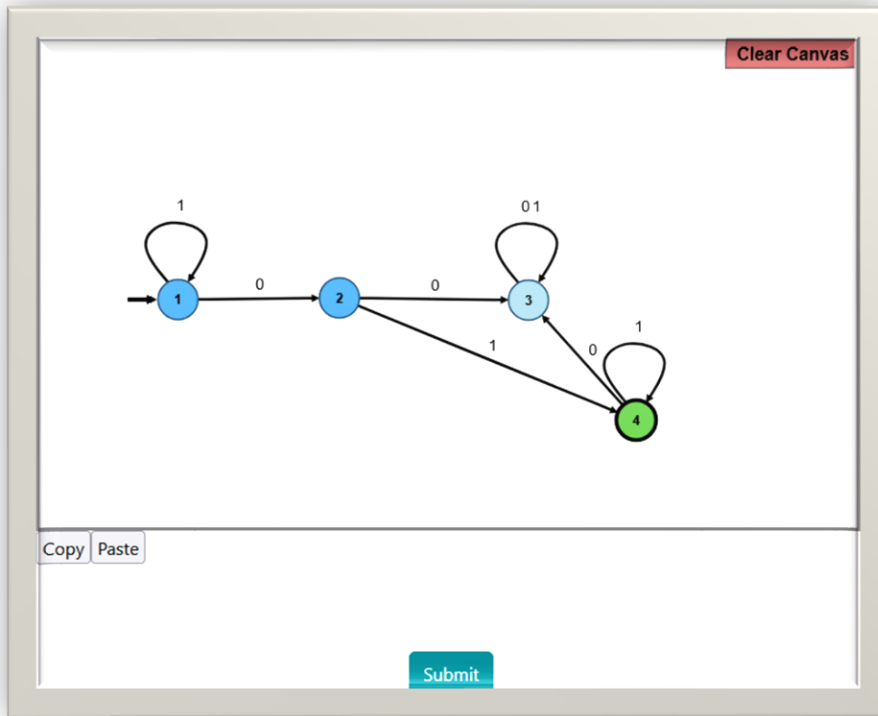


Alexander Poremba

September 16, 2026

Automata Tutor (Homework #2)

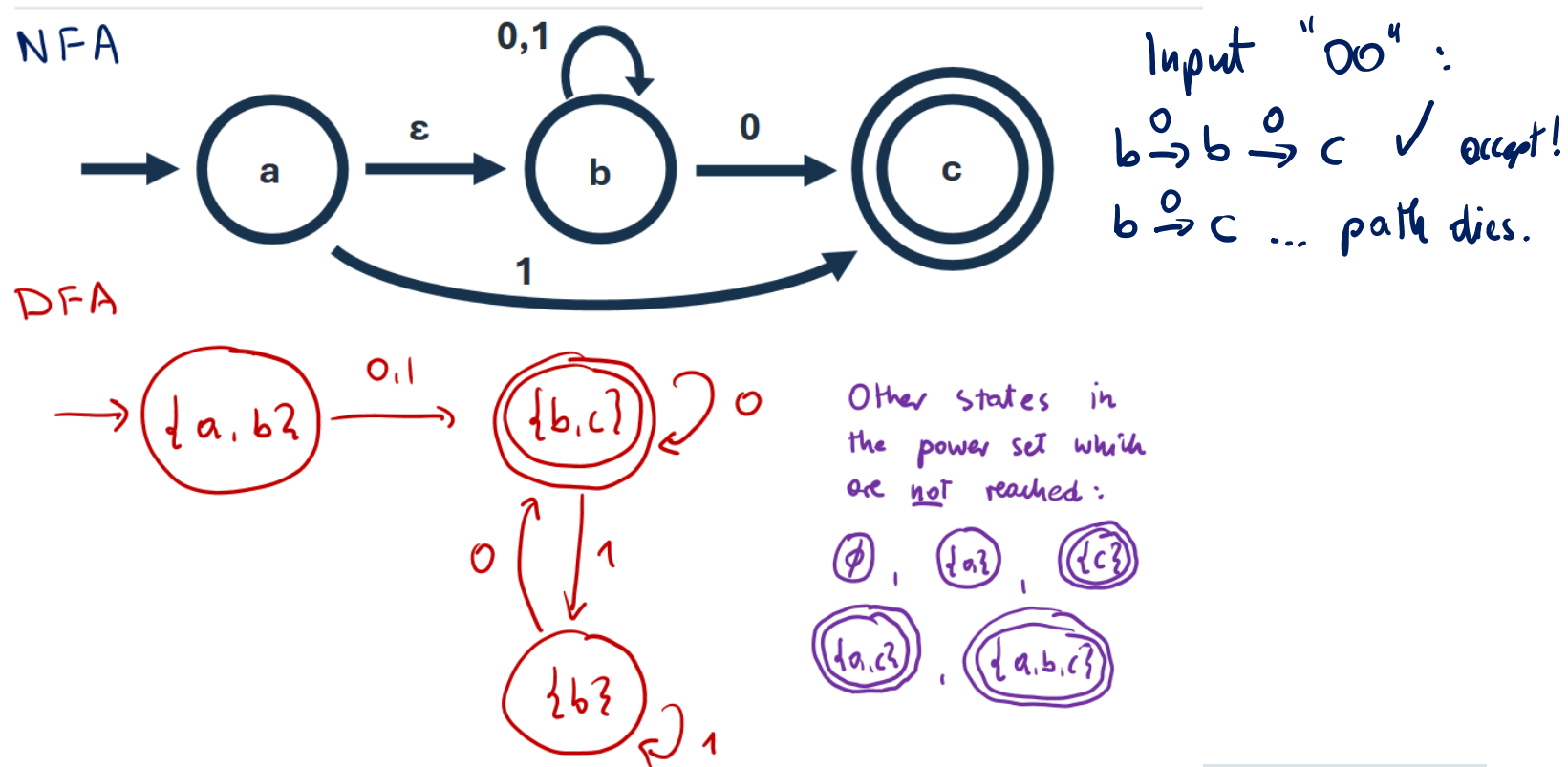
Website: <https://automata-tutor.live-lab.fi.muni.cz/>



- **Instructions** on Piazza:
- Sign up on Automata Tutor
- Make account using your name and **BU e-mail**
- Access problems via link
- Automata Tutor problems will be autograded (no need to submit to Gradescope)

Last Time

- **Nondeterministic** Finite Automata
- Equivalence of NFAs vs. DFAs (**Subset construction**: NFA $\rightarrow$ DFA)



Last Time

- **Nondeterministic** Finite Automata
- Equivalence of NFAs vs. DFAs (**Subset construction**: $\text{NFA} \rightarrow \text{DFA}$)

A language is **regular** if and only if there exists a DFA (equivalently, an NFA) that **recognizes** it.



Closure Properties

An Analogy

In algebra, we try to identify operations which are common to many different mathematical structures

Example: The integers $\mathbb{Z} = \{\dots - 2, -1, 0, 1, 2, \dots\}$ are **closed** under

- Addition: $x + y$
- Multiplication: $x \times y$
- Negation: $-x$
- ...but **NOT** Division: x / y

closure under these operations !

We'd like to investigate similar closure properties of the **class of regular languages**

Regular operations on languages

Let $A, B \subseteq \Sigma^*$ be languages. Define

Union: $A \cup B = \{w \mid w \in A \text{ or } w \in B\}$

Concatenation: $A \circ B = \{xy \mid x \in A, y \in B\}$

Star: $A^* = \{a_1 a_2 \dots a_n \mid n \geq 0, \text{ each } a_i \in A\}$

↑ all finite strings over A , including ϵ .

$$= \underbrace{\{\epsilon\}}_{n=0} \cup \underbrace{A}_{n=1} \cup \underbrace{(A \circ A)}_{n=2} \cup \underbrace{(A \circ A \circ A)}_{n=3} \dots$$

Recall :

Σ^* = all finite strings
formed over Σ ,
including ϵ .

Other operations

Let $A, B \subseteq \Sigma^*$ be languages. Define

Complement: $\bar{A} = \{w \mid w \notin A\}$

Intersection: $A \cap B = \{w \mid w \in A \textbf{ and } w \in B\}$

Reverse: $A^R = \{w \mid w^R \in A\}$

Operations on languages

Let $A, B \subseteq \Sigma^*$ be languages. Define

Regular Operations

Union: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

Concatenation: $A \circ B = \{xy \mid x \in A, y \in B\}$

Star: $A^* = \{w_1 w_2 \dots w_n \mid n \geq 0 \text{ and } w_i \in A\}$

Complement: $\bar{A} = \{x \mid x \notin A\}$

Intersection: $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$

Reverse: $A^R = \{a_1 a_2 \dots a_n \mid a_n \dots a_1 \in A\}$

Theorem: The class of regular languages is **closed under all six of these operations**, i.e., if A and B are **regular**, applying any of these operations yields a **regular** language



Proving Closure Properties

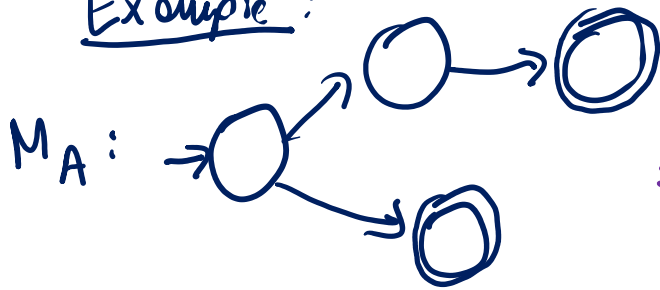
Complement

Complement: $\bar{A} = \{w \mid w \notin A\}$

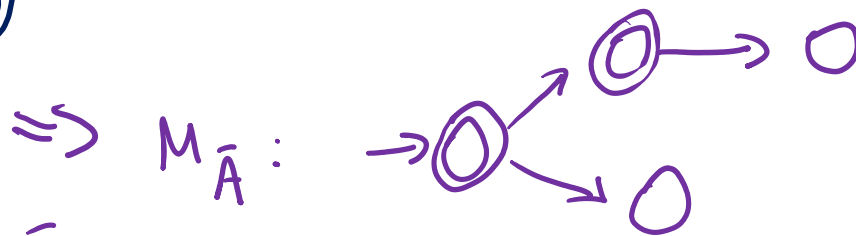
Theorem: If A is regular, then $\bar{A}$ is also regular

Proof sketch: let A be any regular language and suppose a DFA M_A recognizes it. Need to show: There exists a DFA/NFA that recognizes $\bar{A}$.

Example:



Idea: Interchange the accept & reject states



Claim: $M_{\bar{A}}$ recognizes $\bar{A}$.

- $x \in \bar{A} \Rightarrow x \notin A \Rightarrow M_A$ rejects $\Rightarrow M_{\bar{A}}$ accepts ✓
- $x \notin \bar{A} \Rightarrow x \in A \Rightarrow M_A$ accepts $\Rightarrow M_{\bar{A}}$ rejects ✓

Question: Complement, Formally?

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing a language A . Which of the following represents a DFA recognizing $\bar{A}$?



- a) $(F, \Sigma, \delta, q_0, Q)$
- ☒ b) $(Q, \Sigma, \delta, q_0, Q \setminus F)$, where $Q \setminus F$ is the set of states in Q that are not in F
- c) $(Q, \Sigma, \delta', q_0, F)$ where $\delta'(q, s) = p$ such that $\delta(p, s) = q$
- d) None of the above

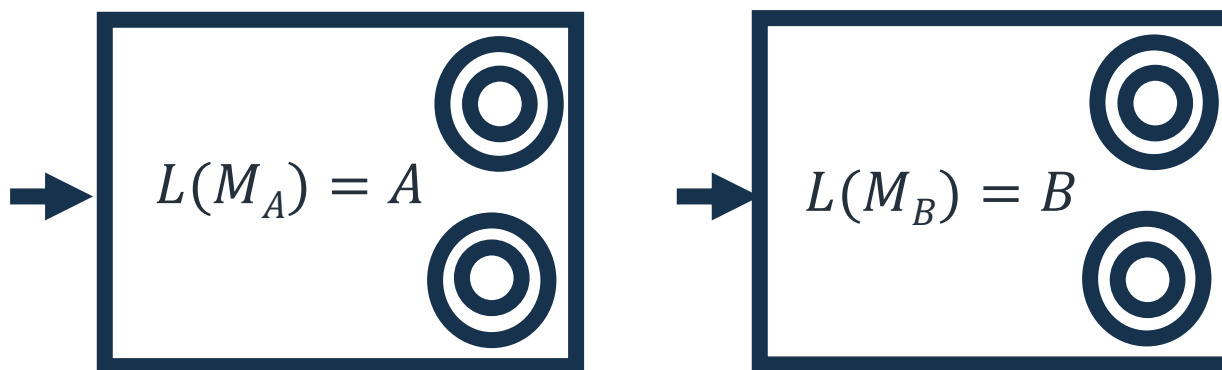
Closure under Concatenation

Concatenation: $A \circ B = \{ xy \mid x \in A, y \in B \}$

Theorem. If A and B are regular, then $A \circ B$ is also regular.

Proof idea: Given DFAs M_A and M_B , construct an NFA for $A \circ B$ by

- Connecting all accept states in M_A to the start state in M_B .
- Making all states in M_A non-accepting.



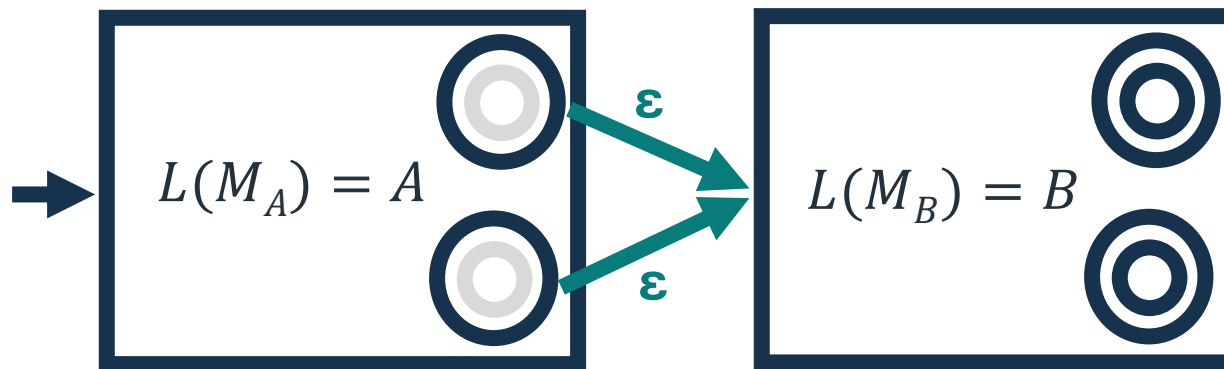
Closure under Concatenation

Concatenation: $A \circ B = \{ xy \mid x \in A, y \in B \}$

Theorem. If A and B are regular, then $A \circ B$ is also regular.

Proof idea: Given DFAs M_A and M_B , construct an NFA for $A \circ B$ by

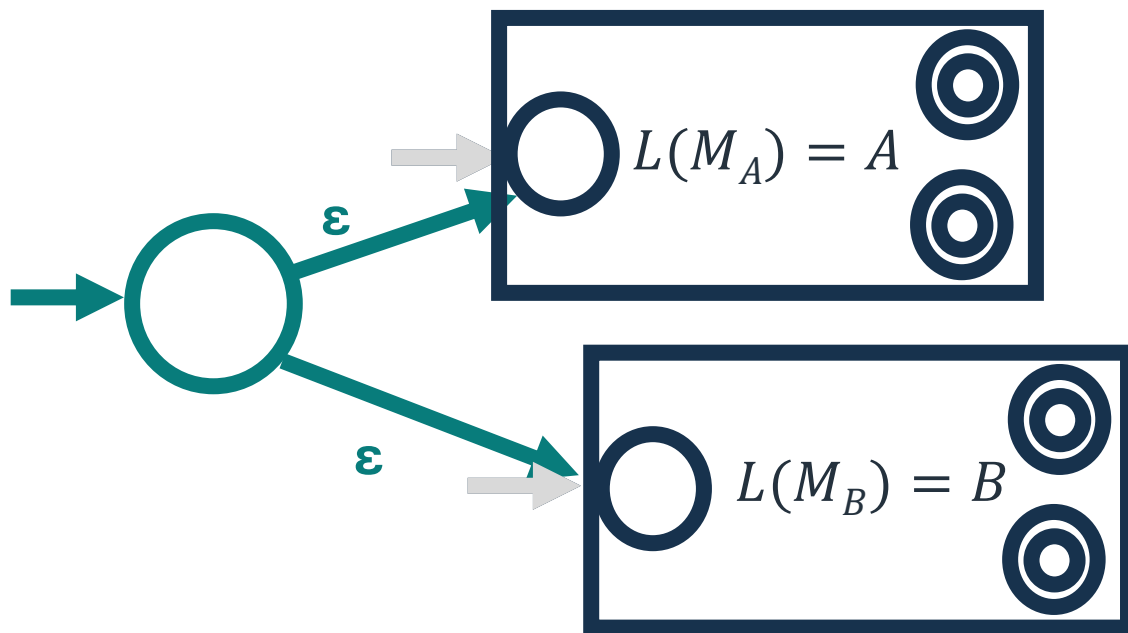
- Connecting all accept states in M_A to the start state in M_B .
- Making all states in M_A non-accepting.



- If $z \in A \circ B$,
the machine accepts ✓
- If $z \notin A \circ B$,
the machine rejects. ✓

Question: Mystery Construction?

Given DFAs M_A recognizing A and M_B recognizing B , what does the following NFA recognize?

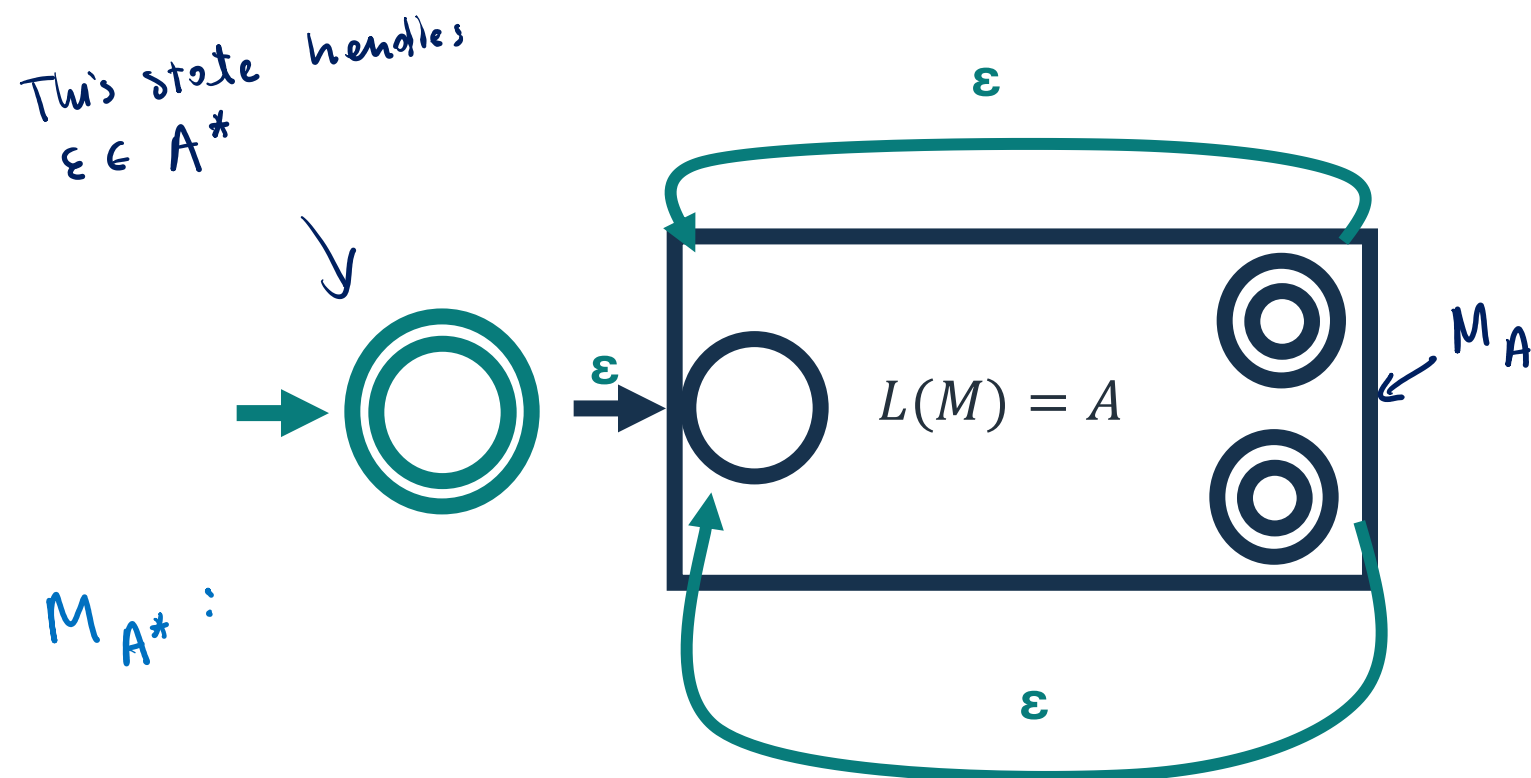


- a) $A \cup B$
- b) $A \circ B$
- c) $A \cap B$
- d) $\{\epsilon\} \cup A \cup B$

Closure under Star

Star: $A^* = \{ a_1 a_2 \dots a_n \mid n \geq 0 \text{ and each } a_i \in A \}$

Theorem. If A is regular, then A^* is also regular.



On proving your own closure properties

You'll have homework/test problems of the form “show that the regular languages are closed under some operation”

What would Sipser do?

- Give the “**proof idea**”: Explain how to take machine(s) recognizing regular language(s) and create a new machine
- Explain in a few sentences why the construction works
- Give a formal description of the construction
- **No need to formally prove** that the construction works



Regular Expressions



succinct way to describe
a regular language.

Regular Expressions

- A different way of describing regular languages
- A regular expression expresses a (possibly complex) language by combining simple languages using the regular operations

“Simple” languages: $\emptyset, \{\varepsilon\}, \{a\}$ for some $a \in \Sigma$ $L \subseteq \Sigma^*$

Regular operations:

Union: $A \cup B$

Concatenation: $A \circ B = \{ab \mid a \in A, b \in B\}$

Star: $A^* = \{a_1 a_2 \dots a_n \mid n \geq 0 \text{ and } a_i \in A\}$

Regular Expressions – Syntax

A **regular expression** R is defined recursively using the rules:

1. ε , $\emptyset$, and a are regular expressions for every $a \in \Sigma$
2. If R_1 and R_2 are regular expressions, then so are
 $(R_1 \cup R_2)$, $(R_1 \circ R_2)$, and (R_1^*)

Examples: (over $\Sigma = \{a, b, c\}$)

$(a \circ b)$ $((((a \circ (b^*)) \circ c) \cup (((a^*) \circ b))^*))$ $(\emptyset^*)$

Regular Expressions – Semantics

$L(R)$ = the language a regular expression describes

1. $L(\emptyset) = \emptyset$
2. $L(\varepsilon) = \{\varepsilon\}$
3. $L(a) = \{a\}$ for every $a \in \Sigma$
4. $L((R_1 \cup R_2)) = L(R_1) \cup L(R_2)$
5. $L((R_1 \circ R_2)) = L(R_1) \circ L(R_2)$
6. $L((R_1^*)) = (L(R_1))^*$

Question: Regular Expression

$$L(\underbrace{((a^*) \circ (b^*))}_{\text{regular expression}}) = L(a^*) \circ L(b^*) \quad (\text{rule 5})$$

$$= \underbrace{L(a)}^* \circ \underbrace{L(b)}^* \quad (\text{rule 6})$$

$$\begin{aligned} \text{a) } & \{a^n b^n \mid n \geq 0\} \\ \text{b) } & \{a^m b^n \mid m, n \geq 0\} \\ \text{c) } & \{(ab)^n \mid n \geq 0\} \\ \text{d) } & \{a, b\}^* \end{aligned}$$

$$= \{a\}^* \circ \{b\}^* \quad (\text{rule 3})$$

$$= \{a^m b^n \mid m, n \geq 0\}$$



Simplifying Notation

- Omit $\circ$ symbol: $(ab) = (a \circ b)$

- Omit many parentheses, since union and concatenation are associative:

$$(a \cup b \cup c) = (a \cup (b \cup c)) = ((a \cup b) \cup c)$$

- Order of operations: Evaluate star, then concatenation, then union

$$ab^* \cup c = (a(b^*)) \cup c$$


Examples

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

Regular expressions?

Symbols : $0, 1, \emptyset, \varepsilon, \cup, \cdot, *, (,)$

$$L(\underline{(0 \cup 1)}) = L(0) \cup L(1) = \{0, 1\}$$

1. $\{w \mid w \text{ contains exactly one } 1\}$

$$0^* 1 0^* \Rightarrow L(0^* 1 0^*) = \{x 1 y \mid x \in L(0^*), y \in L(0^*)\}$$

2. $\{w \mid w \text{ has length at least 3 and its third symbol is } 0\}$

$$= \{x y 0 z \mid x \in L(0 \cup 1), y \in L(0 \cup 1), z \in L((0 \cup 1)^*)\}$$
$$(0 \cup 1)(0 \cup 1) 0 (0 \cup 1)^*$$

3. $\{w \mid \text{every odd position of } w \text{ is } 1\}$

$$(1(0 \cup 1))^* (\underline{\varepsilon \cup 1})$$

$$L(\varepsilon \cup 1) = L(\varepsilon) \cup L(1) = \{\varepsilon, 1\}$$

Regexes in the Real World

grep = globally search for a regular expression and print matching lines

```
$ grep '^xy*z' myfile
```

```
xyz
```

```
xyzde
```

```
xxz
```

```
xz
```

```
xyyz
```

```
xyyyz
```

```
xyyyyz
```

```
$ grep '^x.*z' myfile
```

```
xyz
```

```
xyzde
```

```
xxz
```

```
xzz
```

```
x\z
```

```
x*z
```

```
xz
```

```
x z
```

```
xYz
```

```
xyyz
```

```
xyyyz
```

```
xyyyyz
```

```
$ grep '^x\*z' myfile
```

```
x*z
```

```
$ grep '\\\'' myfile
```

```
x\z
```

```
$
```