

# Theory of Computer Science

## B4. Regular Languages: Closure Properties and Decidability

Gabriele Röger

University of Basel

March 17, 2021

# Theory of Computer Science

March 17, 2021 — B4. Regular Languages: Closure Properties and Decidability

## B4.1 Introduction

## B4.2 Closure Properties

## B4.3 Decidability

# B4.1 Introduction

## Further Analysis

We can convert freely between regular grammars, DFAs and NFAs. So don't let's analyse them individually but instead focus on the corresponding class of regular languages:

- ▶ With what operations can we “combine” regular languages and the result is again a regular language?  
E.g. is the intersection of two regular languages regular?
- ▶ What general questions can we resolve algorithmically for any regular language?  
E.g. is there an algorithm that takes a regular grammars and a word as input and returns whether the word is in the generated language?

## B4.2 Closure Properties

# Closure Properties

How can we combine  
regular languages  
so that the result is guaranteed  
to be regular as well?



Picture courtesy of stockimages / FreeDigitalPhotos.net

# Concatenation of Languages

## Concatenation

- ▶ For two languages  $L_1$  (over  $\Sigma_1$ ) and  $L_2$  (over  $\Sigma_2$ ), the **concatenation** of  $L_1$  and  $L_2$  is the language  $L_1 L_2 = \{w_1 w_2 \in (\Sigma_1 \cup \Sigma_2)^* \mid w_1 \in L_1, w_2 \in L_2\}$ .
- ▶  $L_1 = \{\text{Pancake, Waffle}\}$   
 $L_2 = \{\text{withIceCream, withMushrooms, withCheese}\}$   
 $L_1 L_2 =$

German: Produkt

# Kleene Star

## Kleene star

- ▶ For language  $L$  define
  - ▶  $L^0 = \{\varepsilon\}$
  - ▶  $L^1 = L$
  - ▶  $L^{i+1} = L^i L$  for  $i \in \mathbb{N}_{>0}$
- ▶ Definition of (Kleene) **star** on  $L$ :  $L^* = \bigcup_{i \geq 0} L^i$ .
- ▶  $L = \{\text{ding, dong}\}$   
 $L^* =$

German: (Kleen)-Stern



# Set Operations

Let  $L$  and  $L'$  be regular languages over  $\Sigma$  and  $\Sigma'$ , respectively.

Languages are just sets of words, so we can also consider the standard set operations:

- ▶ **union**  $L \cup L' = \{w \mid w \in L \text{ or } w \in L'\}$  over  $\Sigma \cup \Sigma'$
- ▶ **intersection**  $L \cap L' = \{w \mid w \in L \text{ and } w \in L'\}$  over  $\Sigma \cap \Sigma'$
- ▶ **complement**  $\bar{L} = \{w \in \Sigma^* \mid w \notin L\}$  over  $\Sigma$

# Closure Properties

General terminology: What do we mean with closure?

## Definition (Closure)

Let  $\mathcal{K}$  be a class of languages.

Then  $\mathcal{K}$  is **closed**...

- ▶ ... under union if  $L, L' \in \mathcal{K}$  implies  $L \cup L' \in \mathcal{K}$
- ▶ ... under intersection if  $L, L' \in \mathcal{K}$  implies  $L \cap L' \in \mathcal{K}$
- ▶ ... under complement if  $L \in \mathcal{K}$  implies  $\bar{L} \in \mathcal{K}$
- ▶ ... under concatenation if  $L, L' \in \mathcal{K}$  implies  $LL' \in \mathcal{K}$
- ▶ ... under star if  $L \in \mathcal{K}$  implies  $L^* \in \mathcal{K}$

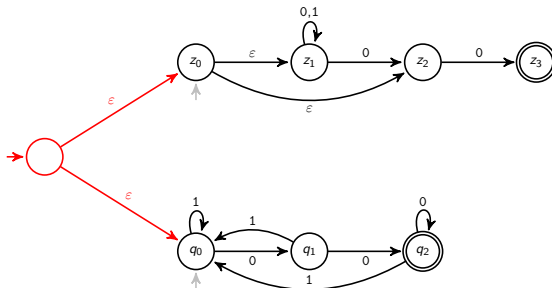
**German:** Abgeschlossenheit,  $\mathcal{K}$  ist abgeschlossen unter Vereinigung (Schnitt, Komplement, Produkt, Stern)

# Closure Properties of Regular Languages: Union

## Theorem

*The regular languages are closed under union.*

Proof idea:



# Closure Properties of Regular Languages: Union

## Proof.

Let  $L_1, L_2$  be regular languages.

Let  $M_1 = \langle Q_1, \Sigma_1, \delta_1, q_1, F_1 \rangle$  and  $M_2 = \langle Q_2, \Sigma_2, \delta_2, q_2, F_2 \rangle$  be NFAs with  $\mathcal{L}(M_1) = L_1$  and  $\mathcal{L}(M_2) = L_2$ . W.l.o.g.  $Q_1 \cap Q_2 = \emptyset$ .

Then NFA  $M = \langle Q, \Sigma_1 \cup \Sigma_2, \delta, q_0, F_1 \cup F_2 \rangle$  with

- ▶  $q_0 \notin Q_1 \cup Q_2$  and
- ▶  $Q = \{q_0\} \cup Q_1 \cup Q_2$ ,
- ▶ for all  $q \in Q, a \in \Sigma_1 \cup \Sigma_2 \cup \{\varepsilon\}$

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & \text{if } q \in Q_1 \text{ and } a \in \Sigma_1 \cup \{\varepsilon\} \\ \delta_2(q, a) & \text{if } q \in Q_2 \text{ and } a \in \Sigma_2 \cup \{\varepsilon\} \\ \{q_1, q_2\} & \text{if } q = q_0 \text{ and } a = \varepsilon \\ \emptyset & \text{otherwise} \end{cases}$$

recognizes  $L_1 \cup L_2$ . □

# Closure Properties of Regular Languages: Concatenation

The proof idea for the closure under concatenation is very similar to the one for union.  
Can you figure it out yourself?

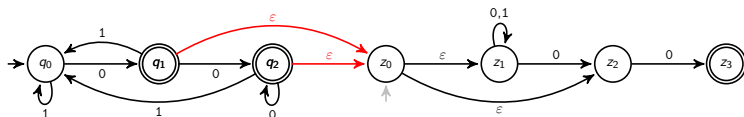


# Closure Properties of Regular Languages: Concatenation

## Theorem

*The regular languages are closed under concatenation.*

Proof idea:



# Closure Properties of Regular Languages: Concatenation

## Proof.

Let  $L_1, L_2$  be regular languages.

Let  $M_1 = \langle Q_1, \Sigma_1, \delta_1, q_1, F_1 \rangle$  and  $M_2 = \langle Q_2, \Sigma_2, \delta_2, q_2, F_2 \rangle$  be NFAs with  $\mathcal{L}(M_1) = L_1$  and  $\mathcal{L}(M_2) = L_2$ . W.l.o.g.  $Q_1 \cap Q_2 = \emptyset$ .

Then NFA  $M = \langle Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, \delta, q_1, F_2 \rangle$  with

► for all  $q \in Q, a \in \Sigma_1 \cup \Sigma_2 \cup \{\varepsilon\}$

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & \text{if } q \in Q_1 \setminus F_1 \text{ and } a \in \Sigma_1 \cup \{\varepsilon\} \\ \delta_1(q, a) & \text{if } q \in F_1 \text{ and } a \in \Sigma_1 \\ \delta_1(q, a) \cup \{q_2\} & \text{if } q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q, a) & \text{if } q \in Q_2 \text{ and } a \in \Sigma_2 \cup \{\varepsilon\} \\ \emptyset & \text{otherwise} \end{cases}$$

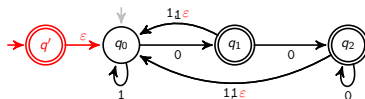
recognizes  $L_1 L_2$ . □

# Closure Properties of Regular Languages: Star

## Theorem

*The regular languages are closed under star.*

Proof idea:





# Closure Properties of Regular Languages: Star

## Proof.

Let  $L$  be a regular language.

Let  $M = \langle Q, \Sigma, \delta, q_0, F \rangle$  be an NFA with  $\mathcal{L}(M) = L$ .

Then NFA  $M' = \langle Q', \Sigma, \delta', q'_0, F \cup \{q'\} \rangle$  with

- ▶  $q'_0 \notin Q$ ,
- ▶  $Q' = Q \cup \{q'_0\}$ , and
- ▶ for all  $q \in Q', a \in \Sigma \cup \{\varepsilon\}$

$$\delta'(q, a) = \begin{cases} \delta(q, a) & \text{if } q \in Q \setminus F \\ \delta(q, a) & \text{if } q \in F \text{ and } a \in \Sigma \\ \delta(q, a) \cup \{q_0\} & \text{if } q \in F \text{ and } a = \varepsilon \\ \{q_0\} & \text{if } q = q'_0 \text{ and } a = \varepsilon \\ \emptyset & \text{otherwise} \end{cases}$$

recognizes  $L^*$ .



# Closure Properties of Regular Languages: Complement

## Theorem

*The regular languages are closed under complement.*

## Proof.

Let  $L$  be a regular language.

Let  $M = \langle Q, \Sigma, \delta, q_0, F \rangle$  be a DFA with  $\mathcal{L}(M) = L$ .

Then  $M' = \langle Q, \Sigma, \delta, q_0, Q \setminus F \rangle$  is a DFA with  $\mathcal{L}(M') = \bar{L}$ . □

# Closure Properties of Regular Languages: Intersection

## Theorem

*The regular languages are closed under intersection.*

## Proof.

Let  $L_1, L_2$  be regular languages.

Let  $M_1 = \langle Q_1, \Sigma_1, \delta_1, q_{01}, F_1 \rangle$  and  $M_2 = \langle Q_2, \Sigma_2, \delta_2, q_{02}, F_2 \rangle$  be DFAs with  $\mathcal{L}(M_1) = L_1$  and  $\mathcal{L}(M_2) = L_2$ .

The **product automaton**

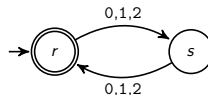
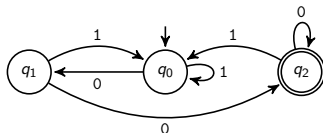
$$M = \langle Q_1 \times Q_2, \Sigma_1 \cap \Sigma_2, \delta, \langle q_{01}, q_{02} \rangle, F_1 \times F_2 \rangle$$

$$\text{with } \delta(\langle q_1, q_2 \rangle, a) = \langle \delta_1(q_1, a), \delta_2(q_2, a) \rangle$$

accepts  $\mathcal{L}(M) = \mathcal{L}(M_1) \cap \mathcal{L}(M_2)$ . □

**German:** Kreuzproduktautomat

# Product Automaton: Example



# Closure Properties of Regular Languages

In summary...

## Theorem

*The regular languages are closed under:*

- ▶ *union*
- ▶ *intersection*
- ▶ *complement*
- ▶ *concatenation*
- ▶ *star*

## B4.3 Decidability

# Decision Problems and Decidability (1)

## “Intuitive Definition:” Decision Problem, Decidability

A **decision problem** is an algorithmic problem where

- ▶ for a given **input**
- ▶ an **algorithm** determines if the input has a given **property**
- ▶ and then produces the **output** “yes” or “no” accordingly.

A decision problem is **decidable** if an algorithm for it (that always terminates and gives the correct answer) exists.

**German:** Entscheidungsproblem, Eingabe, Eigenschaft, Ausgabe, entscheidbar

**Note:** “exists”  $\neq$  “is known”

## Decision Problems and Decidability (2)

### Notes:

- ▶ not a formal definition: we did not formally define “algorithm”, “input”, “output” etc. (which is not trivial)
- ▶ lack of a formal definition makes it difficult to prove that something is **not** decidable
- ↪ studied thoroughly in the next part of the course



## Decision Problems: Example

For now we describe decision problems in a semi-formal  
“given” / “question” way:

### Example (Emptiness Problem for Regular Languages)

The **emptiness problem**  $P_{\emptyset}$  for regular languages  
is the following problem:

**Given:** regular grammar  $G$

**Question:** Is  $\mathcal{L}(G) = \emptyset$ ?

**German:** Leerheitsproblem

# Word Problem

## Definition (Word Problem for Regular Languages)

The **word problem**  $P_{\in}$  for regular languages is:

**Given:** regular grammar  $G$  with alphabet  $\Sigma$   
and word  $w \in \Sigma^*$

**Question:** Is  $w \in \mathcal{L}(G)$ ?

**German:** Wortproblem (für reguläre Sprachen)

# Decidability: Word Problem

## Theorem

*The word problem for regular languages is **decidable**.*

## Proof.

Construct a DFA  $M$  with  $\mathcal{L}(M) = \mathcal{L}(G)$ .

(The proofs in Chapter B3 describe a possible method.)

Simulate  $M$  on input  $w$ . The simulation ends after  $|w|$  steps.

The DFA  $M$  is in an accept state after this iff  $w \in \mathcal{L}(G)$ .

Print “yes” or “no” accordingly.



# Emptiness Problem

## Definition (Emptiness Problem for Regular Languages)

The **emptiness problem**  $P_\emptyset$  for regular languages is:

**Given:** regular grammar  $G$

**Question:** Is  $\mathcal{L}(G) = \emptyset$ ?

**German:** Leerheitsproblem

## Decidability: Emptiness Problem

### Theorem

*The emptiness problem for regular languages is **decidable**.*

### Proof.

Construct a DFA  $M$  with  $\mathcal{L}(M) = \mathcal{L}(G)$ .

We have  $\mathcal{L}(G) = \emptyset$  iff in the transition diagram of  $M$  there is no path from the start state to any accept state.

This can be checked with standard graph algorithms (e.g., breadth-first search).



# Finiteness Problem

## Definition (Finiteness Problem for Regular Languages)

The **finiteness problem**  $P_\infty$  for regular languages is:

**Given:** regular grammar  $G$

**Question:** Is  $|\mathcal{L}(G)| < \infty$ ?

**German:** Endlichkeitsproblem

# Decidability: Finiteness Problem

## Theorem

*The finiteness problem for regular languages is **decidable**.*

## Proof.

Construct a DFA  $M$  with  $\mathcal{L}(M) = \mathcal{L}(G)$ .

We have  $|\mathcal{L}(G)| = \infty$  iff in the transition diagram of  $M$  there is a cycle that is reachable from the start state and from which an accept state can be reached.

This can be checked with standard graph algorithms. □

# Intersection Problem

## Definition (Intersection Problem for Regular Languages)

The **intersection problem**  $P_{\cap}$  for regular languages is:

**Given:** regular grammars  $G$  and  $G'$

**Question:** Is  $\mathcal{L}(G) \cap \mathcal{L}(G') = \emptyset$ ?

**German:** Schnittproblem



# Decidability: Intersection Problem

## Theorem

*The intersection problem for regular languages is **decidable**.*

## Proof.

Using the closure of regular languages under intersection, we can construct (e.g., by converting to DFAs, constructing the product automaton, then converting back to a grammar) a grammar  $G''$  with  $\mathcal{L}(G'') = \mathcal{L}(G) \cap \mathcal{L}(G')$  and use the algorithm for the emptiness problem  $P_\emptyset$ . □

# Equivalence Problem

## Definition (Equivalence Problem for Regular Languages)

The **equivalence problem**  $P_{=}$  for regular languages is:

**Given:** regular grammars  $G$  and  $G'$

**Question:** Is  $\mathcal{L}(G) = \mathcal{L}(G')$ ?

**German:** Äquivalenzproblem

# Decidability: Equivalence Problem

## Theorem

*The equivalence problem for regular languages is **decidable**.*

## Proof.

In general for languages  $L$  and  $L'$ , we have

$$L = L' \text{ iff } (L \cap \bar{L}') \cup (\bar{L} \cap L') = \emptyset.$$

The regular languages are closed under intersection, union and complement, and we know algorithms for these operations.

We can therefore construct a grammar for  $(L \cap \bar{L}') \cup (\bar{L} \cap L')$  and use the algorithm for the emptiness problem  $P_{\emptyset}$ . □

# Summary

- ▶ The regular languages are **closed** under all usual operations (union, intersection, complement, concatenation, star).
- ▶ All usual decision problems (word problem, emptiness, finiteness, intersection, equivalence) are **decidable** for regular languages.