

Theory of Computer Science

C4. Regular Languages: Closure Properties and Decidability

Malte Helmert

University of Basel

March 29, 2017

Theory of Computer Science

March 29, 2017 — C4. Regular Languages: Closure Properties and Decidability

C4.1 Closure Properties

C4.2 Decidability

C4.3 Summary

C4.1 Closure Properties

Closure Properties

How can you combine regular languages in a way to get another regular language as a result?



Picture courtesy of stockimages / FreeDigitalPhotos.net

Closure Properties: Operations

Let L and L' be regular languages over Σ and Σ' , respectively.

We consider the following 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 Σ
- ▶ **product** $LL' = \{uv \mid u \in L \text{ and } v \in L'\}$ over $\Sigma \cup \Sigma'$
 - ▶ special case: $L^n = L^{n-1}L$, where $L^0 = \{\varepsilon\}$
- ▶ **star** $L^* = \bigcup_{k \geq 0} L^k$ over Σ

German: Abschlusseigenschaften, Vereinigung, Schnitt, Komplement, Produkt, Stern

Closure Properties

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 product 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

Theorem

The regular languages are closed under:

- ▶ union
- ▶ intersection
- ▶ complement
- ▶ product
- ▶ star

Closure Properties

Proof.

Closure under **union**, **product**, and **star** follows because for regular expressions α and β , the expressions $(\alpha|\beta)$, $(\alpha\beta)$ and (α^*) describe the corresponding languages.

Complement: Let $M = \langle Q, \Sigma, \delta, q_0, E \rangle$ be a DFA with $\mathcal{L}(M) = L$. Then $M' = \langle Q, \Sigma, \delta, q_0, Q \setminus E \rangle$ is a DFA with $\mathcal{L}(M') = \bar{L}$.

Intersection: Let $M_1 = \langle Q_1, \Sigma_1, \delta_1, q_{01}, E_1 \rangle$ and $M_2 = \langle Q_2, \Sigma_2, \delta_2, q_{02}, E_2 \rangle$ be DFAs. The **product automaton**

$$M = \langle Q_1 \times Q_2, \Sigma_1 \cap \Sigma_2, \delta, \langle q_{01}, q_{02} \rangle, E_1 \times E_2 \rangle$$

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

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

German: Kreuzproduktautomat

C4.2 Decidability

Decision Problems and Decidability (1)

“Intuitive Definition:” Decision Problem, Decidability

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

- ▶ for a given an **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 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 Σ
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 C2 describe a possible method.)

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

The DFA M is an end 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 end 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_∞ 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 end 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} . \square

C4.3 Summary

Summary

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