# Theory of Computer Science
## B5. Regular Languages: Regular Expressions

Gabriele Röger

University of Basel

March 20, 2023

# B5.1 Regular Expressions

# Formalisms for Regular Languages

▶ DFAs, NFAs and regular grammars can all describe exactly the regular languages.

▶ Are there other concepts with the same expressiveness?

▶ Yes! ⤳ regular expressions

⤳ see it in the RealWorld™

# Reminder: Concatenation of Languages and Kleene Star

**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
$L_1 L_2 = \{w_1 w_2 \in (\Sigma_1 \cup \Sigma_2)^* \mid w_1 \in L_1, w_2 \in L_2\}$.

**Kleene star**

▶ For language $L$ define
  ▶ $L^0 = \{\varepsilon\}$
  ▶ $L^1 = L$
  ▶ $L^{i+1} = L^i L$ for $i \in \mathbb{N}_{>0}$
▶ The definition of Kleene star on $L$ is $L^* = \bigcup_{i \geq 0} L^i$.

# Regular Expressions: Definition

---

**Definition (Regular Expressions)**

Regular expressions over an alphabet $\Sigma$ are defined inductively:

- $\emptyset$ is a regular expression
- $\varepsilon$ is a regular expression
- If $a \in \Sigma$, then $a$ is a regular expression

If $\alpha$ and $\beta$ are regular expressions, then so are:

- $(\alpha\beta)$ (concatenation)
- $(\alpha|\beta)$ (alternative)
- $(\alpha^*)$ (Kleene closure)

---

# Regular Expressions: Omitting Parentheses

omitted parentheses by convention:

▶ Kleene closure $\alpha^*$ binds more strongly than concatenation $\alpha\beta$.

▶ Concatenation binds more strongly than alternative $\alpha|\beta$.

▶ Parentheses for nested concatenations/alternatives are omitted (we can treat them as left-associative; it does not matter).

Example: $ab^*c|\varepsilon|abab^*$ abbreviates $((((a(b^*))c)|\varepsilon)|(((ab)a)(b^*)))$.

## Regular Expressions: Examples

some regular expressions for $\Sigma = \{0, 1\}$:

- $0^*10^*$
- $(0|1)^*1(0|1)^*$
- $((0|1)(0|1))^*$
- $01|10$
- $0(0|1)^*0|1(0|1)^*1|0|1$

# Regular Expressions: Language

---

**Definition (Language Described by a Regular Expression)**

The language described by a regular expression $\gamma$, written $\mathcal{L}(\gamma)$, is inductively defined as follows:

- If $\gamma = \emptyset$, then $\mathcal{L}(\gamma) = \emptyset$.
- If $\gamma = \varepsilon$, then $\mathcal{L}(\gamma) = \{\varepsilon\}$.
- If $\gamma = a$ with $a \in \Sigma$, then $\mathcal{L}(\gamma) = \{a\}$.
- If $\gamma = (\alpha\beta)$, where $\alpha$ and $\beta$ are regular expressions, then $\mathcal{L}(\gamma) = \mathcal{L}(\alpha)\mathcal{L}(\beta)$.
- If $\gamma = (\alpha|\beta)$, where $\alpha$ and $\beta$ are regular expressions, then $\mathcal{L}(\gamma) = \mathcal{L}(\alpha) \cup \mathcal{L}(\beta)$.
- If $\gamma = (\alpha^*)$ where $\alpha$ is a regular expression, then $\mathcal{L}(\gamma) = \mathcal{L}(\alpha)^*$.

---

Examples: blackboard

## Regular Expressions: Exercise

Specify a regular expression that describes
$L = \{w \in \{0, 1\}^* \mid$ every 0 in $w$ is followed by at least one $1\}$.

# B5.2 Regular Expressions vs. Regular Languages

# Finite Languages Can Be Described By Regular Expressions

### Theorem
*Every finite language can be described by a regular expression.*

### Proof.
For every word $w \in \Sigma^*$, a regular expression describing
the language $\{w\}$ can be built from regular expressions $a \in \Sigma$
by using concatenations.
(Use $\varepsilon$ if $w = \varepsilon$.)

For every finite language $L = \{w_1, w_2, \ldots, w_n\}$,
a regular expression describing $L$ can be built from the regular
expressions for $\{w_i\}$ by using alternatives.
(Use $\emptyset$ if $L = \emptyset$.) □

We will see that this implies that all finite languages are regular.

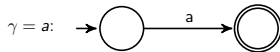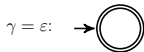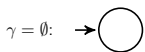# Regular Expressions Not More Powerful Than NFAs

**Theorem**

*For every language that can be described by a regular expression, there is an NFA that accepts it.*

**Proof.**

Let $\gamma$ be a regular expression.

We show the statement by induction over the structure of regular expressions.

For $\gamma = \emptyset$, $\gamma = \varepsilon$ and $\gamma = a$, the following three NFAs accept $\mathcal{L}(\gamma)$:



For $\gamma = (\alpha\beta)$, $\gamma = (\alpha|\beta)$ and $\gamma = (\alpha^*)$ we use the constructions that we used to show that the regular languages are closed under concatenation, union, and star, respectively. $\qquad\square$

# Regular Expression to NFA: Exercise

Construct an NFA that recognizes the language
that is described by the regular expression $(ab|a)^*$.

## DFAs Not More Powerful Than Regular Expressions

### Theorem
*Every language recognized by a DFA can be described*
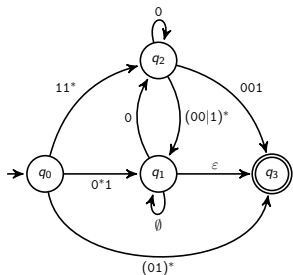*by a regular expression.*

We can prove this using a generalization of NFAs.
We specify the corresponding algorithm.

# Generalized Nondeterministic Finite Automata (GNFAs)

GNFAs are like NFAs but the transition labels can be arbitrary regular expressions over the input alphabet.



For convenience, we require a special form:

▶ The start state has a transition to every other state but no incoming one.

▶ One accept state ($\neq$ start state)

▶ The accept state has an incoming transition from every other state but no outgoing one.

▶ For all other states, one transition goes from every state to every other state and also to itself.

# Generalized Nondeterministic Finite Automaton: Definition

> ### Definition (Generalized Nondeterministic Finite Automata)
>
> A generalized nondeterministic finite automaton (GNFA) is a
> 5-tuple $M = \langle Q, \Sigma, \delta, q_s, q_a \rangle$ where
>
> - $Q$ is the finite set of states
> - $\Sigma$ is the input alphabet
> - $\delta : (Q \setminus \{q_a\}) \times (Q \setminus \{q_s\}) \to \mathcal{R}_\Sigma$ is the transition function
>   (with $\mathcal{R}_\Sigma$ the set of all regular expressions over $\Sigma$)
> - $q_s \in Q$ is the start state
> - $q_a \in Q$ is the accept state

# GNFA: Accepted Words

**Definition (Words Accepted by a GNFA)**

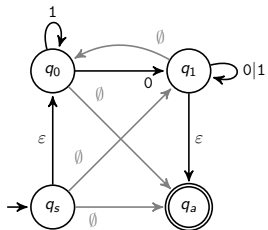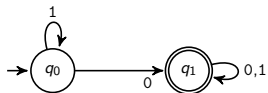GNFA $M = \langle Q, \Sigma, \delta, q_s, q_a \rangle$ accepts the word $w$

if $w = w_1 \ldots w_k$, where each $w_i$ is in $\Sigma^*$

and a sequence of states $q_0, q_1, \ldots, q_k \in Q$ exists with

1. $q_0 = q_s$,
2. for each $i$, we have $w_i \in \mathcal{L}(R_i)$, where $R_i = \delta(q_{i-1}, q_i)$, and
3. $q_k = q_a$.

## DFA to GNFA

We can transform every DFA into a GNFA of the special form:



▶ Add a new start state with an $\epsilon$-transition to the original start state.

▶ Add a new accept state with $\epsilon$-transitions from the original accept states.

▶ Combine parallel transitions into one, labelled with the alternative of the original labels.

▶ If required transitions are missing, add transitions labelled with $\emptyset$.

## Conversion of GNFA to a Regular Expressions

Convert($M = \langle Q, \Sigma, \delta, q_s, q_a \rangle$)

1. If $|Q| = 2$ return $\delta(q_s, q_a)$.

2. Select any state $q \in Q \setminus \{q_s, q_a\}$ and let
   $M' = \langle Q \setminus \{q\}, \Sigma, \delta', q_s, q_a \rangle$,
   where for any $q_i \neq q_a$ and $q_j \neq q_s$
   we define
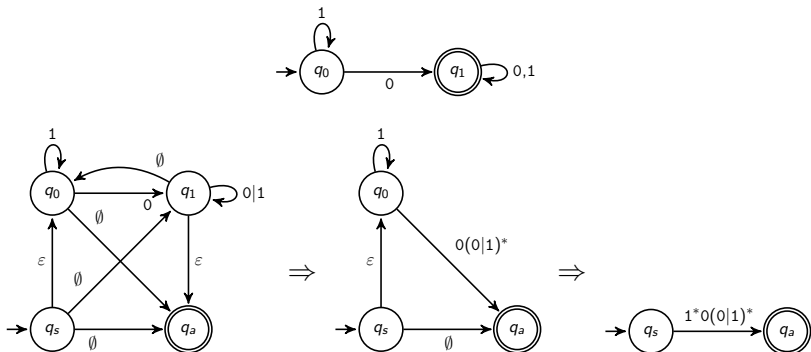   $$\delta'(q_i, q_j) = (\gamma_1)(\gamma_2)^*(\gamma_3) | (\gamma_4)$$
   with
   $\gamma_1 = \delta(q_i, q)$, $\gamma_2 = \delta(q, q)$, $\gamma_3 = \delta(q, q_j)$, $\gamma_4 = \delta(q_i, q_j)$.

3. Return Convert($M'$)

## Example

For DFA:



Regular expression: $1^*0(0|1)^*$

# Regular Languages vs. Regular Expressions

> ### Theorem (Kleene)
> *The set of languages that can be described by regular expressions is exactly the set of regular languages.*

This follows directly from the previous two theorems.

# B5.3 Summary

# Summary

▶ Regular expressions are another way to describe languages.
▶ All regular languages can be described by regular expressions, and all regular expressions describe regular languages.
▶ Hence, they are equivalent to finite automata.