Theory of Computer Science

C3. Regular Languages: Regular Expressions

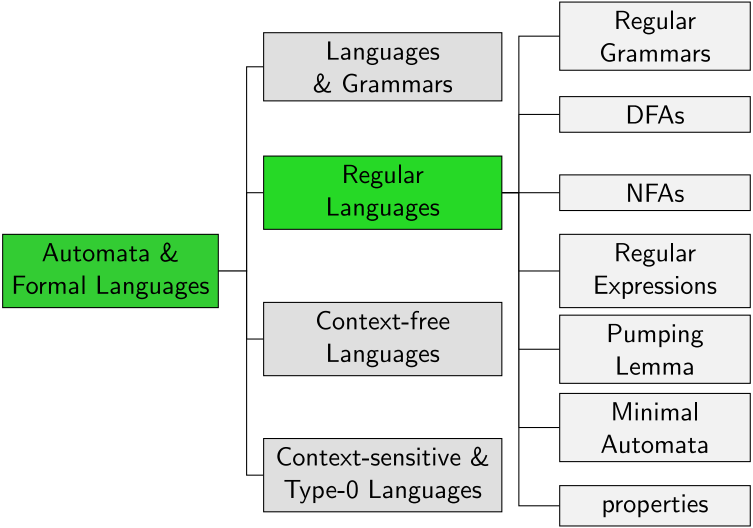
Gabriele Röger

University of Basel

March 25, 2020

Regular Expressions

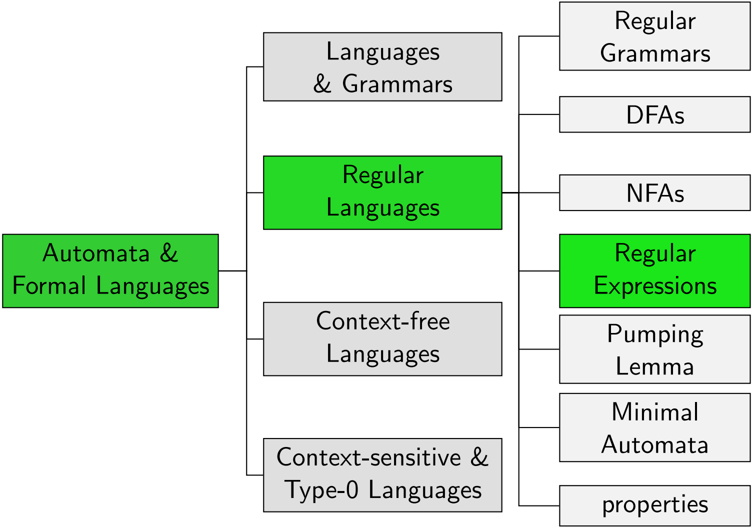
Overview


Automata &
Formal Languages
Languages
& Grammars
Regular
Languages
Context-free
Languages
Context-sensitive &
Type-0 Languages
Regular
Grammars
DFAs
NFAs
Regular
Expressions
Pumping
Lemma
Minimal
Automata
properties

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! $\rightsquigarrow$ regular expressions

Overview


Automata &
Formal Languages
Languages
& Grammars
Regular
Languages
Context-free
Languages
Context-sensitive &
Type-0 Languages
Regular
Grammars
DFAs
NFAs
Regular
Expressions
Pumping
Lemma
Minimal
Automata
properties

Concatenation of Languages and Kleene Star

Concatenation

- For two languages L_1 (over Σ_1) and L_2 (over Σ_2), the concatenation of L_1 and L_2 is the language $L_1 L_2 = \{w_1 w_2 \in (\Sigma_1 \cup \Sigma_2)^* \mid w_1 \in L_1, w_2 \in L_2\}$.

Concatenation of Languages and Kleene Star

Concatenation

- For two languages L_1 (over Σ_1) and L_2 (over Σ_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 Σ are defined inductively:

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

If α and β are regular expressions, then so are:

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

German: reguläre Ausdrücke, Verkettung, Alternative, kleenesche Hülle

Regular Expressions: Omitting Parentheses

omitted parentheses by convention:

- Kleene closure α^* 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: $\mathtt{ab}^*\mathtt{c}|\varepsilon|\mathtt{abab}^*$ abbreviates $((((\mathtt{a}(\mathtt{b}^*))\mathtt{c})|\varepsilon)|(((\mathtt{ab})\mathtt{a})(\mathtt{b}^*)))$.

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 γ, 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 α and β are regular expressions, then $\mathcal{L}(\gamma) = \mathcal{L}(\alpha)\mathcal{L}(\beta)$.
- If $\gamma = (\alpha|\beta)$, where α and β are regular expressions, then $\mathcal{L}(\gamma) = \mathcal{L}(\alpha) \cup \mathcal{L}(\beta)$.
- If $\gamma = (\alpha^*)$ where α is a regular expression, then $\mathcal{L}(\gamma) = \mathcal{L}(\alpha)^*$.

Examples: blackboard

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 ε if $w = \varepsilon$.)

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

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.

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 γ be a regular expression.
We show the statement by induction over the structure of regular expressions.

For $\gamma = \emptyset, \gamma = \varepsilon$ and $\gamma = a$,
NFAs that accept $\mathcal{L}(\gamma)$ are obvious. ...

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 (continued).

For $\gamma = (\alpha\beta)$, let M_α and M_β be NFAs that (by ind. hypothesis) accept $\mathcal{L}(\alpha)$ and $\mathcal{L}(\beta)$. W.l.o.g., their states are disjoint.

Construct NFA M for $\mathcal{L}(\gamma)$ by "daisy-chaining" M_α and M_β:

- states: union of states of M_α and M_β
- start states: those of M_α; if $\varepsilon \in \mathcal{L}(\alpha)$, also those of M_β
- end states: end states of M_β
- state transitions: all transitions of M_α and of M_β; additionally: for every transition to an end state of M_α, an equally labeled transition to all start states of M_β

...

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 (continued).

For $\gamma = (\alpha|\beta)$, by the induction hypothesis let $M_\alpha = \langle Q_\alpha, \Sigma, \delta_\alpha, S_\alpha, E_\alpha \rangle$ and $M_\beta = \langle Q_\beta, \Sigma, \delta_\beta, S_\beta, E_\beta \rangle$ be NFAs that accept $\mathcal{L}(\alpha)$ and $\mathcal{L}(\beta)$. W.l.o.g., $Q_\alpha \cap Q_\beta = \emptyset$.

Then the "union automaton"

$$M = \langle Q_\alpha \cup Q_\beta, \Sigma, \delta_\alpha \cup \delta_\beta, S_\alpha \cup S_\beta, E_\alpha \cup E_\beta \rangle$$

accepts the language $\mathcal{L}(\gamma)$. . . .

German: Vereinigungsautomat

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 (continued).

For $\gamma = (\alpha^*)$, by the induction hypothesis let $M_\alpha = \langle Q_\alpha, \Sigma, \delta_\alpha, S_\alpha, E_\alpha \rangle$ be an NFA that accepts $\mathcal{L}(\alpha)$.

If $\varepsilon \notin \mathcal{L}(\alpha)$, add an additional state to M_α that is a start and end state and not connected to other states. M_α now recognizes $\mathcal{L}(\alpha) \cup \{\varepsilon\}$.

M is constructed from M_α by adding the following new transitions: whenever M_α has a transition from s to end state s' with symbol a, add transitions from s to every start state with symbol a.

Then $\mathcal{L}(M) = \mathcal{L}(\gamma)$. □

DFAs Not More Powerful Than Regular Expressions

Theorem

Every language accepted by a DFA can be described by a regular expression.

Without proof.

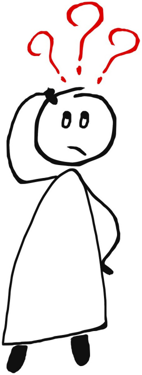
Regular Languages vs. Regular Expressions

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.

Questions



Questions?

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.