Theory of Computer Science

C3. Regular Languages: Regular Expressions, Pumping Lemma

Malte Helmert

University of Basel

March 27, 2017

Theory of Computer Science

March 27, 2017 — C3. Regular Languages: Regular Expressions, Pumping Lemma

C3.1 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! $\rightsquigarrow$ regular expressions

Live demo

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).

For example, $\texttt{ab}^*\texttt{c}|\varepsilon|\texttt{abab}^*$ abbreviates $((((\texttt{a}(\texttt{b}^*))\texttt{c})|\varepsilon)|(((\texttt{ab})\texttt{a})(\texttt{b}^*)))$.

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 $L = \emptyset$.) □

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.

C3.2 Pumping Lemma

Pumping Lemma: Motivation

You can show that a language is regular by specifying an appropriate grammar, finite automaton, or regular expression. How can you you show that a language is not regular?

- Direct proof that no regular grammar exists that generates the language $\rightsquigarrow$ difficult in general
- Pumping lemma: use a necessary property that holds for all regular languages.

Picture courtesy of imagerymajestic / FreeDigitalPhotos.net

Pumping Lemma

Theorem (Pumping Lemma)

Let L be a regular language. Then there is an $n \in \mathbb{N}$ (a pumping number for L) such that all words $x \in L$ with $|x| \geq n$ can be split into $x = uvw$ with the following properties:

1. $|v| \geq 1$,
2. $|uv| \leq n$, *and*
3. $uv^iw \in L$ *for all* $i = 0, 1, 2, \dots$.

Question: what if L is finite?

Pumping Lemma: Proof

Theorem (Pumping Lemma)

Let L be a regular language. Then there is an $n \in \mathbb{N}$ (a pumping number for L) such that all words $x \in L$ with $|x| \geq n$ can be split into $x = uvw$ with the following properties:

1. $|v| \geq 1$,
2. $|uv| \leq n$, *and*
3. $uv^iw \in L$ *for all* $i = 0, 1, 2, \dots$.

Proof.

For regular L there exists a DFA $M = \langle Q, \Sigma, \delta, q_0, E\rangle$ with $\mathcal{L}(M) = L$. We show that $n = |Q|$ has the desired properties.

Consider an arbitrary $x \in \mathcal{L}(M)$ with length $|x| \geq |Q|$. Including the start state, M visits $|x| + 1$ states while reading x. Because of $|x| \geq |Q|$ at least one state has to be visited twice. ...

Pumping Lemma: Proof

Theorem (Pumping Lemma)

Let L be a regular language. Then there is an $n \in \mathbb{N}$ (a pumping number for L) such that all words $x \in L$ with $|x| \geq n$ can be split into $x = uvw$ with the following properties:

1. $|v| \geq 1$,
2. $|uv| \leq n$, *and*
3. $uv^iw \in L$ *for all* $i = 0, 1, 2, \dots$.

Proof (continued).

Choose a split $x = uvw$ so M is in the same state after reading u and after reading uv. Obviously, we can choose the split in a way that $|v| \geq 1$ and $|uv| \leq |Q|$ are satisfied. ...

Pumping Lemma: Proof

Theorem (Pumping Lemma)

Let L be a regular language. Then there is an $n \in \mathbb{N}$ (a pumping number for L) such that all words $x \in L$ with $|x| \geq n$ can be split into $x = uvw$ with the following properties:

1. $|v| \geq 1$,
2. $|uv| \leq n$, *and*
3. $uv^iw \in L$ *for all* $i = 0, 1, 2, \ldots$.

Proof (continued).

The word v corresponds to a loop in the DFA after reading u and can thus be repeated arbitrarily often. Every subsequent continuation with w ends in the same end state as reading x.
Therefore $uv^iw \in \mathcal{L}(M) = L$ is satisfied for all $i = 0, 1, 2, \ldots$. □

Pumping Lemma: Application

Using the pumping lemma (PL):

Proof of Nonregularity

- If L is regular, then the pumping lemma holds for L.
- By contraposition: if the PL does not hold for L, then L cannot be regular.
- That is: if there is no $n \in \mathbb{N}$ with the properties of the PL, then L cannot be regular.

Pumping Lemma: Caveat

Caveat:

The pumping lemma is a necessary condition for a language to be regular, but not a sufficient one

- $\rightsquigarrow$ there are languages that satisfy the pumping lemma conditions but are not regular
- $\rightsquigarrow$ for such languages, other methods are needed to show that they are not regular (e.g., the Myhill-Nerode theorem)

Pumping Lemma: Example

Example

The language $L = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$ is not regular.

Proof.

Assume L is regular. Then let p be a pumping number for L.

The word $x = \mathtt{a}^p\mathtt{b}^p$ is in L and has length $\geq p$.
Let $x = uvw$ be a split with the properties of the PL.

Then the word $x' = uv^2w$ is also in L. Since $|uv| \leq p$, uv consists only of symbols $\mathtt{a}$ and $x' = \mathtt{a}^{|u|}\mathtt{a}^{2|v|}\mathtt{a}^{p-|uv|}\mathtt{b}^p = \mathtt{a}^{p+|v|}\mathtt{b}^p$.

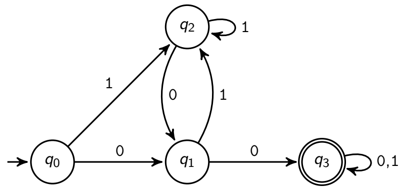
Since $|v| \geq 1$ it follows that $p + |v| \neq p$ and thus $x' \notin L$.

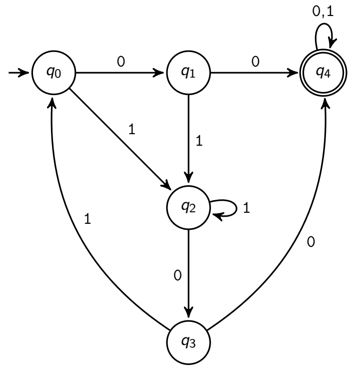
This is a contradiction to the PL. $\rightsquigarrow$ L is not regular. □

C3.3 Minimal Automata (skimmed)

Example

The following DFAs accept the same language:





Question: What is the smallest DFA that accepts this language?

Minimal Automaton: Definition

Definition

A minimal automaton for a regular language L is a DFA $M = \langle Q, \Sigma, \delta, q_0, E\rangle$ with $\mathcal{L}(M) = L$ and a minimal number of states.

This means there is no DFA $M' = \langle Q', \Sigma, \delta', q'_0, E'\rangle$ with $\mathcal{L}(M) = \mathcal{L}(M')$ and $|Q'| < |Q|$.

How to find a minimal automaton?

Idea:

- Start with any DFA that accepts the language.
- Merge states from which exactly the same words lead to an end state.

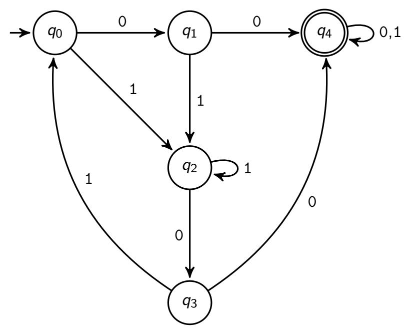
Minimal Automaton: Algorithm

Input: DFA M
(without states that are unreachable from the start state)

Output: list of states that have to be merged
to obtain an equivalent minimal automaton

1. Create table of all pairs of states $\{q, q'\}$ with $q \neq q'$.
2. Mark all pairs $\{q, q'\}$ with $q \in E$ and $q' \notin E$.
3. If there is an unmarked pair $\{q, q'\}$ where $\{\delta(q, a), \delta(q', a)\}$ for some $a \in \Sigma$ is already marked, then also mark $\{q, q'\}$.
4. Repeat the last step until there are no more changes.
5. All states in pairs that are still unmarked can be merged into one state.

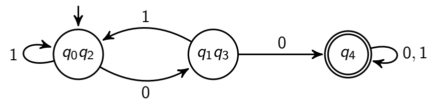
Minimal Automaton: Example



q_1	×			
q_2		×		
q_3	×		×	
q_4	×	×	×	×
	q_0	q_1	q_2	q_3

States q_0, q_2 and q_1, q_3 can be merged into one state each.

Result:



Computation and Uniqueness of Minimal Automata

Theorem

The algorithm described on the previous slides produces a minimal automaton for the language accepted by the given input DFA.

Theorem

All minimal automata for a language L are unique up to isomorphism (i.e., renaming of states).

Without proof.

C3.4 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.
- The pumping lemma can be used to show that a language is not regular.
- skimmed: minimal automata are the smallest possible DFAs for a given language and are unique for each language.