CS_275 Automata and Formal Language Theory

Course Notes
Part II: The Recognition Problem (II)
Chapter II.4.: Properties of Regular Languages (13)

Anton Setzer
(Based on a book draft by J. V. Tucker and K. Stephenson)
Dept. of Computer Science, Swansea University

http://www.cs.swan.ac.uk/~csetzer/lectures/automataFormalLanguage/current/index.html

April 19, 2018

Theorem II.4.1.1

We will show that regular expressions coincide with regular languages and with languages recognised by a DFA or NFA.
Here we prove one part of this result:

Theorem (II.4.1.1)

For every right linear grammar G there exists an NFA A s.t.

$$L(G) = L(A)$$

A can be computed from G.

Proof Idea

- A derivation of a word in G has the form

$$\begin{aligned} S = A_0 \Rightarrow a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \cdots \Rightarrow & a_1 a_2 \cdots a_{n-1} A_{n-1} \\ \Rightarrow & a_1 a_2 \cdots a_{n-1} a_n \end{aligned}$$

where we have productions

$$A_i \longrightarrow a_{i+1} A_{i+1} \quad A_{n-1} \longrightarrow a_n$$

or

$$\begin{aligned} S = A_0 \Rightarrow a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \cdots \Rightarrow & a_1 a_2 \cdots a_{n-1} A_{n-1} \\ \Rightarrow & a_1 a_2 \cdots a_{n-1} \end{aligned}$$

where we have productions

$$A_i \longrightarrow a_{i+1} A_{i+1} \quad A_{n-1} \longrightarrow \epsilon$$

Proof Idea

Define A with states $N \cup \{q_F\}$ for a special new accepting state q_F s.t. the derivation

$$\begin{aligned} S = A_0 \Rightarrow a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \cdots \Rightarrow & a_1 a_2 \cdots a_{n-1} A_{n-1} \\ \Rightarrow & a_1 a_2 \cdots a_{n-1} a_n \end{aligned}$$

corresponds to a sequence of transitions

$$S = A_0 \xrightarrow{a_1} A_1 \xrightarrow{a_2} A_2 \xrightarrow{a_3} \cdots \xrightarrow{a_{n-1}} A_{n-1} \xrightarrow{a_n} q_F$$

and a derivation

$$\begin{aligned} S = A_0 \Rightarrow a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \cdots \Rightarrow & a_1 a_2 \cdots a_{n-1} A_{n-1} \\ \Rightarrow & a_1 a_2 \cdots a_{n-1} \end{aligned}$$

corresponds to a sequence of transitions

$$S = A_0 \xrightarrow{a_1} A_1 \xrightarrow{a_2} A_2 \xrightarrow{a_3} \cdots \xrightarrow{a_{n-1}} A_{n-1} \in F$$

Proof Idea

So we have:

- If $B \longrightarrow aB'$, then $B \xrightarrow{a} B'$.
- If $B \longrightarrow a$ then $B \xrightarrow{a} q_F$.
- $q_F \in F$.
- If $B \longrightarrow \epsilon$, then $B \in F$.

Constructed NFA

We obtain from $G = (N, T, S, P)$ the following NFA:

automaton	A
states	$N \cup \{q_F\}$
terminals	T
start	S
final	$B \in N$ s.t. $B \longrightarrow \epsilon$. q_F
transitions	$B \xrightarrow{a} B'$ if $B \longrightarrow aB'$. $B \xrightarrow{a} q_F$ if $B \longrightarrow a$.

Proof of Theorem II.4.1.1

Can be found in the additional material.

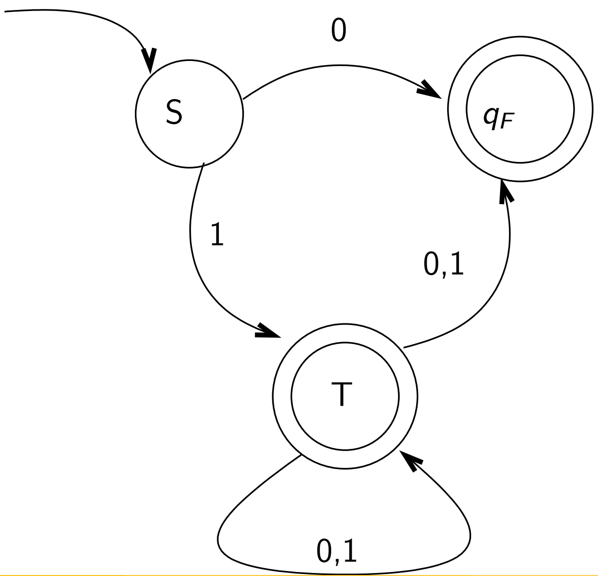
Example

Consider the Grammar:

grammar	G
terminals	$0, 1$
nonterminals	$S,\ T$
start symbol	S
productions	$S \longrightarrow 0,\ S \longrightarrow 1T,$ $T \longrightarrow 0T,\ T \longrightarrow 1T,$ $T \longrightarrow \epsilon,\ T \longrightarrow 0,\ T \longrightarrow 1$

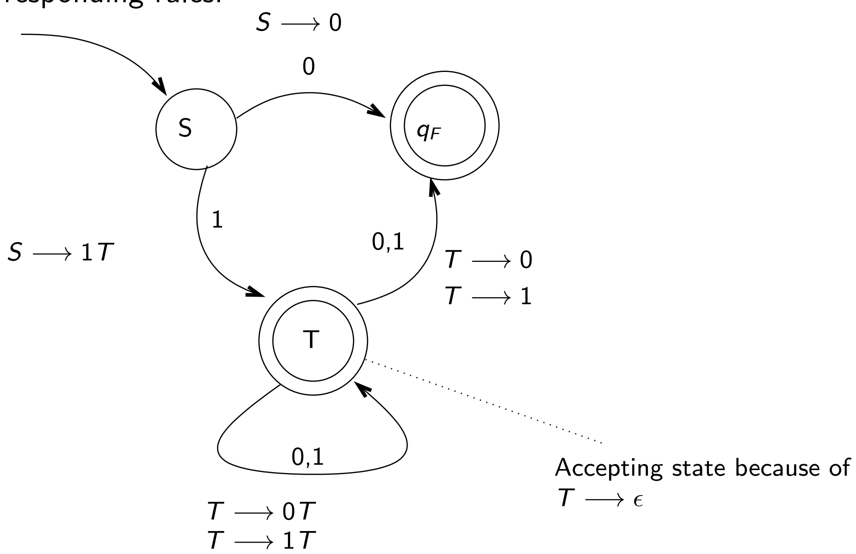
Corresponding Automaton

(Note that it is non-deterministic).



Corresponding Automaton

With corresponding rules:



Computing from NFA a Right Linear Grammar

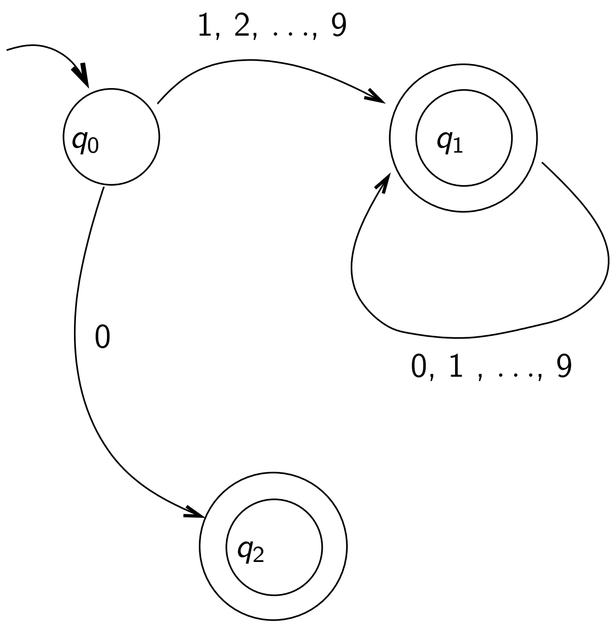
- One can as well easily compute from an NFA an equivalent right linear grammar by inverting the above procedure:
  - Non-terminals are the set of states of the NFA.
  - Productions are
    $$q \longrightarrow aq' \text{ if } q \stackrel{a}{\longrightarrow} q'$$
    and
    $$q \longrightarrow \epsilon \text{ if } q \text{ final state}$$
  - Start symbol = start state of the NFA.

Right Linear Grammar obtained from NFA

Assume an NFA A with states Q, terminals T, start state q_0, final states F, transitions $\longrightarrow$.
The following is an equivalent NFA:

grammar	A'		
terminals	T		
nonterminals	Q		
start symbol	q_0		
productions	$q \longrightarrow aq'$	if	$q \stackrel{a}{\longrightarrow} q'$
	$q \longrightarrow \epsilon$	if	$q \in F$

Example: Consider the following Automaton



Right Linear Grammar obtained from the NFA

grammar	$G^{\text{NumbersNoLeadingZeros}}$		
terminals	$0, 1, \ldots, 9$		
nonterminals	q_0, q_1, q_2		
start symbol	q_0		
productions	$q_0 \longrightarrow xq_1$	for	$x \in \{1, \ldots, 9\}$
	$q_1 \longrightarrow xq_1$	for	$x \in \{0, \ldots, 9\}$
	$q_0 \longrightarrow 0q_2$		
	$q_1 \longrightarrow \epsilon$		
	$q_2 \longrightarrow \epsilon$		

Theorem II.4.1.2

Theorem (II.4.1.2)

Let $A = (Q, q_0, F, T, \longrightarrow)$ be an NFA.
Then there exist a regular expression E s.t. $L(E) = L(A)$.
E can be computed from A.

Proof Idea of Theorem II.4.1.2

- Let for $q, q' \in Q$ and $Q' \subseteq Q'$

  $\mathrm{L}^{Q'}_{q,q'}$:= the words which allow you to get from q to q' while having as intermediate states states in Q' only

- Now you define regular expressions for $\mathrm{L}^{Q'}_{q,q'}$ by starting with $Q' = \emptyset$ and than systematically adding states to Q' until you have obtained $Q' = Q$.
- The case $Q' = \emptyset$ is as follows:

$$\mathrm{L}^{\emptyset}_{q,q'} = \begin{cases} \{t \mid t \in T, q \xrightarrow{t} q'\} & \text{if } q \neq q' \\ \{\epsilon\} \cup \{t \mid t \in T, q \xrightarrow{t} q'\} & \text{if } q = q' \end{cases}$$

  which is a finite set which can therefore be expressed as a regular expression.
  Note that you cannot use q, q' as intermediate states in this case, we have to go in at most one step from q to q'.

Proof Idea of Theorem II.4.1.2

- If you have define $\mathrm{L}^{Q'}_{q,q'}$ and Q'' is obtained by adding to Q' one more state q'', then $\mathrm{L}^{Q''}_{q,q'}$ is the language obtained by
  - either going from q to q' by using states in Q'' only
  - of by going from q to q'', then arbitrary many times from q'' to itself, and then from q'' to q', always using states in Q'' only.

So

$$\mathrm{L}^{Q''}_{q,q'} = \mathrm{L}^{Q'}_{q,q'} \cup (\mathrm{L}^{Q'}_{q,q''}.(\mathrm{L}^{Q'}_{q'',q''})^*.\mathrm{L}^{Q'}_{q'',q'})$$

So from regular expressions for $\mathrm{L}^{Q'}_{q,q'}$ for all q, q' we obtain regular expressions for $\mathrm{L}^{Q''}_{q,q'}$ for all q, q'.

Proof Idea of Theorem II.4.1.2

- Now the language of the automaton with initial state q_0 and final state F is
$$\bigcup_{q' \in \mathrm{F}} \mathrm{L}^{Q}_{q_0,q'}$$
which again can be expressed by a regular expression.
- More details can be found together with an example in the additional material.
- A nice exposition can be found in Sect. 3.2.1. of [HMU07] John Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman: Introduction to automata theory, languages, and computation, Addison Wesley, 3rd Ed, 2006.

Notations w^{R}, L^{R}

- If w is a word, w^{R} is the result of reversing the word. For instance, if $w = abc$, then $w^{\mathrm{R}} = cba$.
- If L is a language,
$$L^{\mathrm{R}} := \{w^{\mathrm{R}} \mid w \in L\}$$
For instance if
$$L = \{abc, def, ghi\}$$
then
$$L^{\mathrm{R}} = \{cba, fed, ihg\}$$

Theorem II.4.1.3

Theorem (II.4.1.3)

Let L be a language over an alphabet T. The following are equivalent:

1. *L is definable by a regular expression.*
2. *L is definable by a right-linear grammar.*
3. *L is definable by a left-linear grammar.*
4. *L is definable by an NFA with empty moves*
5. *L is definable by an NFA.*
6. *L is definable by a DFA.*

Furthermore, the corresponding regular expressions, right linear grammars, left-linear grammars, NFAs with empty moves, NFAs, DFAs can be computed from each other.

Proof of Theorem II.4.1.3

- The following directions have been introduced or at least sketched above:
  - Translations between DFA, NFA, NFA with empty moves.
  - Translation between NFA and right linear grammars
  - Translation of regular expressions into left/right linear grammars.
  - Translation of NFA and therefore as well right linear grammars into regular expressions.
- What is more complicated is
  Translation between left linear and right linear grammars.
  - A regular expression for L can easily be translated into a regular expression for L^{R}.
  - If we reverse the right hand sides of productions we obtain from a right linear grammar for L a left linear grammar for L^{R} and from a left linear grammar for L a right linear grammar for L^{R}.

Proof of Theorem II.4.1.3

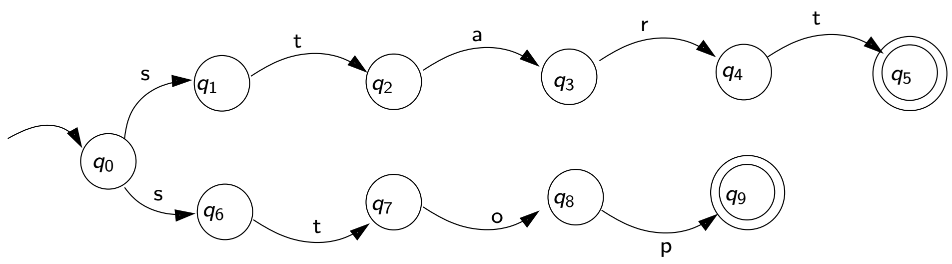
- Now
  - from a left linear grammar for L we obtain
  - a right linear grammar for L^{R};
  - then a regular expression for L^{R};
  - then a regular expression for L;
  - then a right linear grammar for L.
- In the other direction
  - from a right linear grammar for L we obtain
  - a regular expression for L;
  - then a regular expression for L^{R};
  - then a right linear grammar for L^{R};
  - then a left linear grammar for L.
- Together the above translations provide a proof of Theorem II.4.1.3.
- Full details can be found in the additional material.

Alternative of getting from L to L^{R}

- Alternatively, one can easily obtain from an NFA for a language L an NFA with empty moves for the language L^{R}:
  - Reverse all the transitions in the automaton.
  - Add a new start state q_New and ϵ transitions to each of the previous accepting states.
  - Replace the accepting states by making the old start state the only new accepting state.
- This way we can now translate a
  - a right linear grammar for L into an NFA for L
  - into an NFA with empty moves for L^{R}
  - into an NFA for L^{R}
  - into a right linear grammar for L^{R}
  - into a left linear grammar for L.
- Similarly we get from a left linear grammar for L to a right linear grammar for L.

Example Converting NFA for L into one for L^{R}

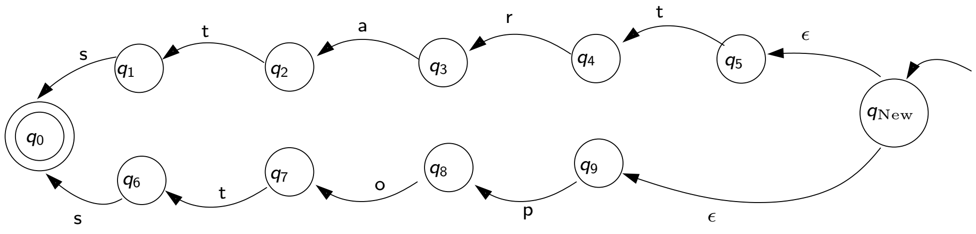
Consider example we had before:



Language accepted was $L = \{\mathrm{start}, \mathrm{stop}\}$.

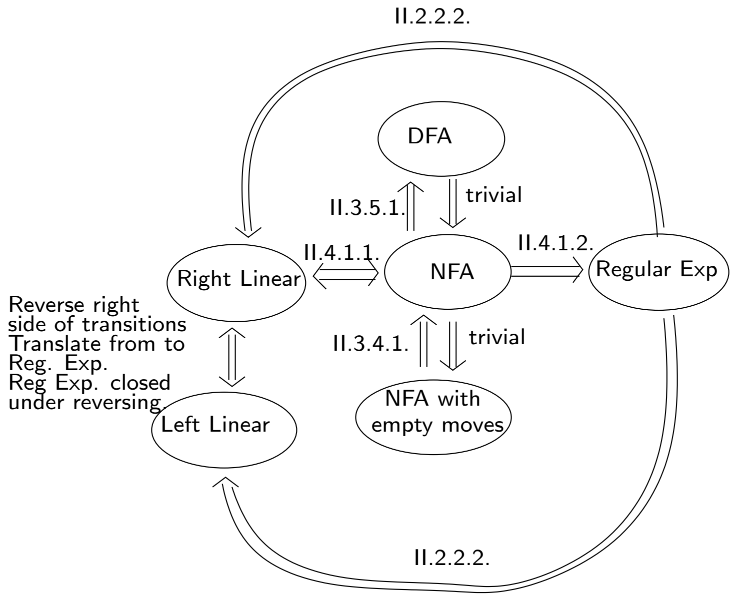
Example Converting NFA for L into one for L^{R}

Result of reverting the arrows:



Language accepted is $L^{\mathrm{R}} = \{\mathrm{trats}, \mathrm{pots}\}$.

Directions in Equivalence Theorem



Closure Properties

Theorem (II.4.4.1.)

If L, L' are regular languages over alphabet T, so are

1. *the complement L^c of L,*
   - *(here $L^c := \{w \in T^* \mid w \notin L\}$),*
2. *the intersection $L \cap L'$ of L and L'*
3. *the union $L \cup L'$ of L and L'*
4. *the relative complement $L \setminus L'$ of L and L'*
   - *(here $L \setminus L' := \{w \in L \mid w \notin L\}$),*
5. *the reverse L^R of L*
   - *(here $L^R := \{w^R \mid w \in L\}$, where w^R is the result of reverting w).*

Furthermore, regular expressions, regular grammars, NFAs and DFAs for L^c, $L \cap L'$, $L \cup L'$, $L \setminus L'$, L^R can be computed from those for L and L'.

Proof Idea for Theorem II.4.4.1.

- We will use that languages defined by regular expressions, DFAs, NFAs, and regular grammars are equivalent, and that corresponding automata, regular expressions and grammars can be computed from each other.
- From a DFA for L one can easily define a DFA for L^c.
- One see that that from NFAs for L and L' one can obtain a NFA for $L \cap L'$ which essentially executes both NFAs in parallel.
- One can see that from an NFA for L and L' one can obtain an NFA with empty moves for $L \cup L'$.
- $L \setminus L' = L \cap (L')^c$.
- From a regular expression for L one can easily obtain a regular expression for L^R. (See additional material, Lemma II.4.3.4).
- Therefore the assertion follows.
- Full details can be found in the additional material.

Decision Problems

Theorem (II.4.4.3.)

- *We can decide for regular languages whether* $L = \emptyset$.
- *We can decide for regular languages* L *and* L' *whether* $L \subseteq L'$.
- *We can decide for regular languages* L *and* L' *whether* $L = L'$.

Proof of Theorem II.4.4.3.

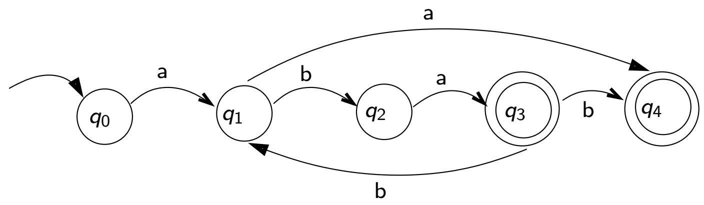
- Again we use the equivalence of languages defined by regular expressions, DFAs, NFAs, and regular grammars, and that corresponding automata, regular expressions and grammars can be computed from each other.
- $L = \emptyset$ can be decided easily for languages defined by regular expressions.
  - For NFA it can be decided as well directly by checking whether any accepting state can be reached from the start state.
- $L \subseteq L' \Leftrightarrow L \setminus L' = \emptyset$.
- $L = L' \Leftrightarrow (L \subseteq L' \wedge L' \subseteq L)$.

Motivation

- We want to show that there are languages which are context-free but not regular.
- In order to do this we prove the pumping lemma, which uses the fact that an NFA has only finitely many states.
  (We could use as well the fact that a regular grammar has only finitely many nonterminals).
- **Note** The following slides contain some coloured parts. The colours are indistinguishable in the black and white handouts. It is recommended to look at them using the online version.

Using the Finiteness of an NFA

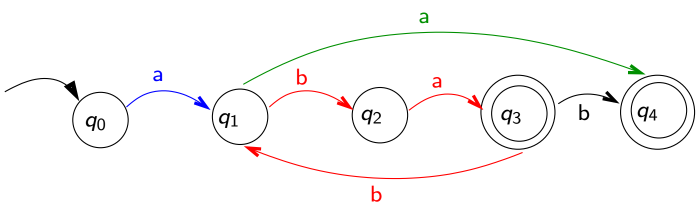
Consider an NFA



This NFA has 5 states.
Any accepting run of the NFA for a word of length ≥ 5 uses at least 6 states.
Therefore it must visit one state at least twice.
So there must be a loop within the first 5 letters of such a word.

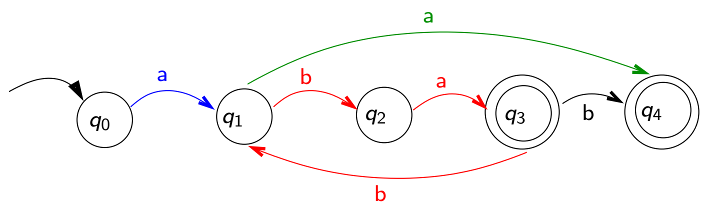
Using the Finiteness of an NFA

Here is the accepting run for the word z = **a**bab**a** = **u**v**w** using colours **blue**, red and **green**.



- The **blue part** is the part before we reached a state visited twice, corresponding to the word $u = a$.
- The red part is the part from the state visited twice until we reach it again, corresponding to the word $v = bab$.
- The **green part** is the remaining part, corresponding to the word $w = a$.
- The loop must occur within the first 5 letters, so $|uv| \leq 5$. Because v is along a loop, $v \neq \epsilon$.

Using the Finiteness of an NFA



- If we repeat the loop several times, we obtain as well an accepting run of the automaton.
  - If we start with **u =** a, then repeat the loop following the word v $= bab$ i times, then the follow the word **w =** a, we obtain an accepting run.
  - It accepts the word $a(bab)^i a$.
    - E.g. in case $i = 2$ the word is $ababbaba$.
    - In case $i = 0$ the word is aa.
  - In general we get that the word uv^iw is an element of the language as well.

Generalisation

Assume an NFA A having k states.
Then for every word $x \in L(A)$ s.t. $|x| \geq k$ there exist words u, v, w s.t.

$$x = uvw,\ |uv| \leq k,\ v \neq \epsilon$$

and s.t.

$$uv^iw \in L(A) \text{ for all } i \in \mathbb{N}$$

This follows by the above considerations.

So we have proved the following theorem:

Pumping Lemma for Regular Languages

Theorem (Pumping Lemma for Regular Languages)

Let L be a regular language.
Then there exist a fixed number k depending on L only s.t. we have the following:

- *If $x \in L$ is a word, $|x| \geq k$, then there exist words u, v, w s.t.*

  $$x = uvw\ ,\ |uv| \leq k\ ,\ v \neq \epsilon$$

  and s.t.

  $$uv^iw \in L(A) \text{ for all } i \in \mathbb{N}$$

Remark

- In most proofs one uses the pumping lemma for the following values of i:
  - $i = 2$, i.e. that $uvvw \in L(A)$.
  - $i = 0$, i.e. that $uw \in L(A)$.
- Usually the pumping lemma is used in order to prove that a language L is **not** regular:
  - One assumes it **were** regular
  - Then there exists some k as in the pumping lemma.
  - One **chooses** a specific word $x \in L$ s.t. $|x| \geq k$.
  - One knows that $x = uvw$ for some u, v, w with the conditions as in the pumping lemma.
  - One shows that for some value of i it is not the case that $uv^iw \in L$.
  - Therefore one gets a contradiction to the pumping lemma.

Example 1

Lemma

The language $L := \{a^ib^i \mid i \geq 1\}$ is context-free but not regular.

Proof (Example 1)

- We have already seen that L is context-free.
  - Start symbol S
  - Productions $S \longrightarrow aSb$, $S \longrightarrow ab$.
- Assume L is regular.
- Let k be as in the pumping lemma.
- Consider $x := a^k b^k \in L$.
- $|x| \geq k$, so there exist u, v, w s.t.
  $x = uvw$, $|uv| \leq k$, $v \neq \epsilon$,
  and s.t.
  $uv^i w \in L$ for all $i \in \mathbb{N}$.
- Since $|uv| \leq k$, u and v are substrings of a^k.

Proof (Example 1, Cont.)

- $a^k b^k = uvw$, so we have
  $a^k b^k = \underbrace{a \cdots a}_{i}\underbrace{a \cdots a}_{l}\underbrace{a \cdots a}_{k-(i+l)}\underbrace{b \cdots b}_{k}$.
- $u = a^i$, $w = a^l$, $w = a^{k-(i+l)} b^k$, where $l = |v| > 0$.
- Therefore $uv^2 w = a^i a^l a^l a^{k-(i+l)} b^k = a^{k+l} b^k$.
- But $a^{k+l} b^k \notin L$, a **contradiction**.

Example 2

Lemma

The language $L := \{xx^R \mid x \in \{a, b\}^*\}$ is context-free but not regular.

Proof (Example 2)

- We have already seen that L is context-free.
  - Start symbol S
  - Productions $S \longrightarrow aSa$, $S \longrightarrow bSb$, $S \longrightarrow \epsilon$.
- Assume L is regular.
- Let k be as in the pumping lemma.
- Consider $x := a^k bba^k \in L$.
- $|x| \geq k$, so there exist u, v, w s.t.
  $x = uvw$, $|uv| \leq k$, $v \neq \epsilon$,
  and s.t.
  $uv^i w \in L$ for all $i \in \mathbb{N}$.
- Since $|uv| \leq k$, u and v are substrings of a^k.
- Therefore $uv^2 w = a^{k+l} bba^k$ where $l = |v| \geq 1$.
- But $a^{k+l} bba^k \notin L$, a **contradiction**.