Automata & Complexity

Jörg Endrullis

Vrije Universiteit Amsterdam

2018

Looking Back

Previous subjects relevant for this lecture:

- Context-free languages via context-free grammars
- Context-free languages via pushdown automata

Looking Back

Previous subjects relevant for this lecture:

- Context-free languages via context-free grammars
- Context-free languages via pushdown automata

Question

Are all languages context-free?

Pumping Lemma for Context-Free Languages (1961)

Theorem

Let L be a context-free language.

There exists $m > 0$ such that for every word $w \in L$ with $|w| \geq m$:

$$w = uvxyz$$

with $|vxy| \leq m$ and $|vy| \geq 1$, and $uv^ixy^iz \in L$ for every $i \geq 0$.

Pumping Lemma for Context-Free Languages (1961)

Theorem

Let L be a context-free language.

There exists $m > 0$ such that for every word $w \in L$ with $|w| \geq m$:

$$w = uvxyz$$

with $|vxy| \leq m$ and $|vy| \geq 1$, and $uv^ixy^iz \in L$ for every $i \geq 0$.

Proof

Let G be a context-free grammar with $L(G) = L \setminus \{\lambda\}$

- with k variables, and
- without λ and unit productions.

Pumping Lemma for Context-Free Languages (1961)

Theorem

Let L be a context-free language.

There exists $m > 0$ such that for every word $w \in L$ with $|w| \geq m$:

$$w = uvxyz$$

with $|vxy| \leq m$ and $|vy| \geq 1$, and $uv^ixy^iz \in L$ for every $i \geq 0$.

Proof

Let G be a context-free grammar with $L(G) = L \setminus \{\lambda\}$

- with k variables, and
- without λ and unit productions.

Let m = 1 + maximum number of leaves of derivation trees of depth $\leq k + 1$.

Pumping Lemma for Context-Free Languages (1961)

Theorem

Let L be a context-free language.

There exists $m > 0$ such that for every word $w \in L$ with $|w| \geq m$:

$$w = uvxyz$$

with $|vxy| \leq m$ and $|vy| \geq 1$, and $uv^ixy^iz \in L$ for every $i \geq 0$.

Proof

Let G be a context-free grammar with $L(G) = L \setminus \{\lambda\}$

- with k variables, and
- without λ and unit productions.

Let m = 1 + maximum number of leaves of derivation trees of depth $\leq k + 1$.

continued on next slide...

Pumping Lemma for Context-Free Languages (1961)

Let $w \in L$ with $|w| \geq m$. Consider a derivation tree for w.

Pumping Lemma for Context-Free Languages (1961)

Let $w \in L$ with $|w| \geq m$. Consider a derivation tree for w.

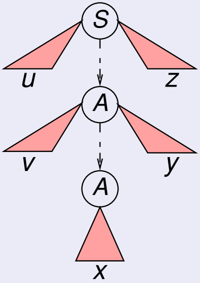
There must be a path of length $> k$. Consider the longest path.

Pumping Lemma for Context-Free Languages (1961)

Let $w \in L$ with $|w| \geq m$. Consider a derivation tree for w.

There must be a path of length $> k$. Consider the longest path.

As there are only k variables, there must be a variable A that occurs twice among the last $k + 1$ variable nodes of the path.

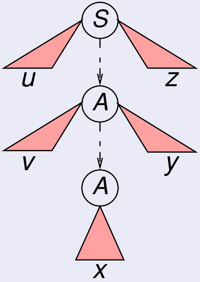


Pumping Lemma for Context-Free Languages (1961)

Let $w \in L$ with $|w| \geq m$. Consider a derivation tree for w.

There must be a path of length $> k$. Consider the longest path.

As there are only k variables, there must be a variable A that occurs twice among the last $k + 1$ variable nodes of the path.



We have $w = uvxyz$ with

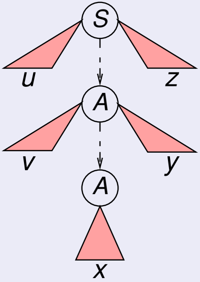
- $S \Rightarrow^* uAz$
- $A \Rightarrow^+ vAy$
- $A \Rightarrow^+ x$

Pumping Lemma for Context-Free Languages (1961)

Let $w \in L$ with $|w| \geq m$. Consider a derivation tree for w.

There must be a path of length $> k$. Consider the longest path.

As there are only k variables, there must be a variable A that occurs twice among the last $k+1$ variable nodes of the path.



We have $w = uvxyz$ with

- $S \Rightarrow^* uAz$
- $A \Rightarrow^+ vAy$
- $A \Rightarrow^+ x$

Hence

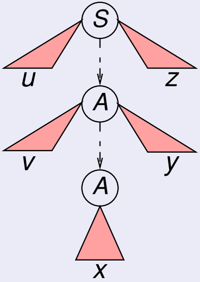
- $S \Rightarrow^+ uv^ixy^iz$ for every $i \geq 0$.

Pumping Lemma for Context-Free Languages (1961)

Let $w \in L$ with $|w| \geq m$. Consider a derivation tree for w.

There must be a path of length $> k$. Consider the longest path.

As there are only k variables, there must be a variable A that occurs twice among the last $k+1$ variable nodes of the path.



We have $w = uvxyz$ with

- $S \Rightarrow^* uAz$
- $A \Rightarrow^+ vAy$
- $A \Rightarrow^+ x$

Hence

- $S \Rightarrow^+ uv^ixy^iz$ for every $i \geq 0$.

Then

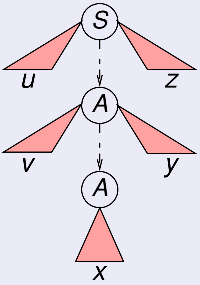
- $|vxy| \leq m$ as the subtree generating vxy has depth $\leq k+1$,

Pumping Lemma for Context-Free Languages (1961)

Let $w \in L$ with $|w| \geq m$. Consider a derivation tree for w.

There must be a path of length $> k$. Consider the longest path.

As there are only k variables, there must be a variable A that occurs twice among the last $k+1$ variable nodes of the path.



We have $w = uvxyz$ with

- $S \Rightarrow^* uAz$
- $A \Rightarrow^+ vAy$
- $A \Rightarrow^+ x$

Hence

- $S \Rightarrow^+ uv^ixy^iz$ for every $i \geq 0$.

Then

- $|vxy| \leq m$ as the subtree generating vxy has depth $\leq k+1$,
- $|vy| \geq 1$, since there are no λ and unit productions. □

Using the Pumping Lemma

Attention

A contradiction of the pumping lemma for specific values of m, or of u, v, x, y, z, is not sufficient!

Using the Pumping Lemma

Attention

A contradiction of the pumping lemma for specific values of m, or of u, v, x, y, z, is not sufficient!

Pumping lemma as formula (note the quantifiers):

$$\exists m > 0.$$
$$\quad \forall w \in L \text{ with } |w| \geq m.$$
$$\quad\quad \exists u, v, x, y, z \text{ with } w = uvxyz,\ |vxy| \leq m,\ |vy| \geq 1.$$
$$\quad\quad\quad \forall i \geq 0.\, uv^ixy^iz \in L$$

Using the Pumping Lemma

Attention

A contradiction of the pumping lemma for specific values of m, or of u, v, x, y, z, is not sufficient!

Pumping lemma as formula (note the quantifiers):

$$
\begin{aligned}
&\exists m > 0.\\
&\quad \forall w \in L \text{ with } |w| \geq m.\\
&\quad\quad \exists u, v, x, y, z \text{ with } w = uvxyz,\ |vxy| \leq m,\ |vy| \geq 1.\\
&\quad\quad\quad \forall i \geq 0.\, uv^i xy^i z \in L
\end{aligned}
$$

To **contradict the pumping lemma**, we prove the negation:

$$
\begin{aligned}
&\forall m > 0.\\
&\quad \exists w \in L \text{ with } |w| \geq m.\\
&\quad\quad \forall u, v, x, y, z \text{ with } w = uvxyz,\ |vxy| \leq m,\ |vy| \geq 1.\\
&\quad\quad\quad \exists i \geq 0.\, uv^i xy^i z \notin L
\end{aligned}
$$

Pumping Lemma as a Game

To **contradict the pumping lemma**, we prove the negation:

$$\forall m > 0.$$
$$\exists w \in L \text{ with } |w| \geq m.$$
$$\forall u, v, x, y, z \text{ with } w = uvxyz,\ |vxy| \leq m,\ |vy| \geq 1.$$
$$\exists i \geq 0.\, uv^i xy^i z \notin L$$

Pumping Lemma as a Game

Given is a language L.

Pumping Lemma as a Game

To **contradict the pumping lemma**, we prove the negation:

$\forall m > 0.$
$\quad \exists w \in L$ with $|w| \geq m.$
$\quad\quad \forall u, v, x, y, z$ with $w = uvxyz,\ |vxy| \leq m,\ |vy| \geq 1.$
$\quad\quad\quad \exists i \geq 0.\, uv^i xy^i z \notin L$

Pumping Lemma as a Game

Given is a language L. We want to prove that L is not context-free.

Pumping Lemma as a Game

To **contradict the pumping lemma**, we prove the negation:

$\forall m > 0.$
$\exists w \in L$ with $|w| \geq m.$
$\forall u, v, x, y, z$ with $w = uvxyz,\ |vxy| \leq m,\ |vy| \geq 1.$
$\exists i \geq 0.\, uv^i xy^i z \notin L$

Pumping Lemma as a Game

Given is a language L. We want to prove that L is not context-free.

1. Opponent picks m.

Pumping Lemma as a Game

To **contradict the pumping lemma**, we prove the negation:

$$\forall m > 0.$$
$$\quad \exists w \in L \text{ with } |w| \geq m.$$
$$\quad\quad \forall u, v, x, y, z \text{ with } w = uvxyz,\ |vxy| \leq m,\ |vy| \geq 1.$$
$$\quad\quad\quad \exists i \geq 0.\, uv^i xy^i z \notin L$$

Pumping Lemma as a Game

Given is a language L. We want to prove that L is not context-free.

1. Opponent picks m.
2. We choose a word $w \in L$ with $|w| \geq m$.

Pumping Lemma as a Game

To **contradict the pumping lemma**, we prove the negation:

$$\forall m > 0.$$
$$\exists w \in L \text{ with } |w| \geq m.$$
$$\forall u, v, x, y, z \text{ with } w = uvxyz,\ |vxy| \leq m,\ |vy| \geq 1.$$
$$\exists i \geq 0.\, uv^i xy^i z \notin L$$

Pumping Lemma as a Game

Given is a language L. We want to prove that L is not context-free.

1. Opponent picks m.
2. We choose a word $w \in L$ with $|w| \geq m$.
3. Opponent picks u, v, x, y, z
   with $w = uvxyz$, $|vxy| \leq m$ and $|vy| \geq 1$.

Pumping Lemma as a Game

To **contradict the pumping lemma**, we prove the negation:

$$\forall m > 0.$$
$$\exists w \in L \text{ with } |w| \geq m.$$
$$\forall u, v, x, y, z \text{ with } w = uvxyz,\ |vxy| \leq m,\ |vy| \geq 1.$$
$$\exists i \geq 0.\, uv^i xy^i z \notin L$$

Pumping Lemma as a Game

Given is a language L. We want to prove that L is not context-free.

1. Opponent picks m.
2. We choose a word $w \in L$ with $|w| \geq m$.
3. Opponent picks u, v, x, y, z with $w = uvxyz$, $|vxy| \leq m$ and $|vy| \geq 1$.
4. If we can find $i \geq 0$ such that $uv^i xy^i z \notin L$, then **we win.**

Pumping Lemma as a Game

To **contradict the pumping lemma**, we prove the negation:

$$\forall m > 0.$$
$$\exists w \in L \text{ with } |w| \geq m.$$
$$\forall u, v, x, y, z \text{ with } w = uvxyz,\ |vxy| \leq m,\ |vy| \geq 1.$$
$$\exists i \geq 0.\, uv^i xy^i z \notin L$$

Pumping Lemma as a Game

Given is a language L. We want to prove that L is not context-free.

1. Opponent picks m.
2. We choose a word $w \in L$ with $|w| \geq m$.
3. Opponent picks u, v, x, y, z with $w = uvxyz$, $|vxy| \leq m$ and $|vy| \geq 1$.
4. If we can find $i \geq 0$ such that $uv^i xy^i z \notin L$, then **we win.**

If we can always win, then L does not fulfil the pumping lemma!

Example

Assume that $L = \{ a^n b^n c^n \mid n \geq 0 \}$ was context-free.

Example

Assume that $L = \{ a^n b^n c^n \mid n \geq 0 \}$ was context-free.

According to the pumping lemma there is $m > 0$ such that

$$a^m b^m c^m = uvxyz$$

with $|vxy| \leq m$, $|vy| \geq 1$, and $uv^i xy^i z \in L$ for every $i \geq 0$.

Example

Assume that $L = \{ a^n b^n c^n \mid n \geq 0 \}$ was context-free.

According to the pumping lemma there is $m > 0$ such that

$$a^m b^m c^m = uvxyz$$

with $|vxy| \leq m$, $|vy| \geq 1$, and $uv^i xy^i z \in L$ for every $i \geq 0$.

Note:

- opponent picks m,
- we pick $w = a^m b^m c^m$,
- opponent u, v, x, y, z.

Example

Assume that $L = \{ a^n b^n c^n \mid n \geq 0 \}$ was context-free.

According to the pumping lemma there is $m > 0$ such that

$$a^m b^m c^m = uvxyz$$

with $|vxy| \leq m$, $|vy| \geq 1$, and $uv^i xy^i z \in L$ for every $i \geq 0$.

Note:

- opponent picks m,
- we pick $w = a^m b^m c^m$,
- opponent u, v, x, y, z.

We do not know u, v, x, y, z, but we can reason about them:

Example

Assume that $L = \{ a^n b^n c^n \mid n \geq 0 \}$ was context-free.

According to the pumping lemma there is $m > 0$ such that

$$a^m b^m c^m = uvxyz$$

with $|vxy| \leq m$, $|vy| \geq 1$, and $uv^i xy^i z \in L$ for every $i \geq 0$.

Note:

- opponent picks m,
- we pick $w = a^m b^m c^m$,
- opponent u, v, x, y, z.

We do not know u, v, x, y, z, but we can reason about them:

- Since $|vxy| \leq m$, it follows that $vy = a^j b^k$ or $vy = b^j c^k$.

Example

Assume that $L = \{ a^n b^n c^n \mid n \geq 0 \}$ was context-free.

According to the pumping lemma there is $m > 0$ such that

$$a^m b^m c^m = uvxyz$$

with $|vxy| \leq m$, $|vy| \geq 1$, and $uv^i xy^i z \in L$ for every $i \geq 0$.

Note:

- opponent picks m,
- we pick $w = a^m b^m c^m$,
- opponent u, v, x, y, z.

We do not know u, v, x, y, z, but we can reason about them:

- Since $|vxy| \leq m$, it follows that $vy = a^j b^k$ or $vy = b^j c^k$.
- Since $|vy| \geq 1$ we have $j + k \geq 1$.

Example

Assume that $L = \{ a^n b^n c^n \mid n \geq 0 \}$ was context-free.

According to the pumping lemma there is $m > 0$ such that

$$a^m b^m c^m = uvxyz$$

with $|vxy| \leq m$, $|vy| \geq 1$, and $uv^i xy^i z \in L$ for every $i \geq 0$.

Note:

- opponent picks m,
- we pick $w = a^m b^m c^m$,
- opponent u, v, x, y, z.

We do not know u, v, x, y, z, but we can reason about them:

- Since $|vxy| \leq m$, it follows that $vy = a^j b^k$ or $vy = b^j c^k$.
- Since $|vy| \geq 1$ we have $j + k \geq 1$.

Then $uv^2 xy^2 z$ does not contain equally many a's, b's and c's.

Example

Assume that $L = \{ a^n b^n c^n \mid n \geq 0 \}$ was context-free.

According to the pumping lemma there is $m > 0$ such that

$$a^m b^m c^m = uvxyz$$

with $|vxy| \leq m$, $|vy| \geq 1$, and $uv^i xy^i z \in L$ for every $i \geq 0$.

Note:

- opponent picks m,
- we pick $w = a^m b^m c^m$,
- opponent u, v, x, y, z.

We do not know u, v, x, y, z, but we can reason about them:

- Since $|vxy| \leq m$, it follows that $vy = a^j b^k$ or $vy = b^j c^k$.
- Since $|vy| \geq 1$ we have $j + k \geq 1$.

Then $uv^2 xy^2 z$ does not contain equally many a's, b's and c's.

Contradiction, thus L is not context-free.

Exercises

Show that the pumping lemma holds for:

- $\{a^n b^n \mid n \geq 0\}$
- $\{ww^R \mid w \in \{a, b\}^*\}$
- $\{w \in \{a, b\}^* \mid w = w^R\}$

Exercises

Show that the pumping lemma holds for:

- $\{a^n b^n \mid n \geq 0\}$
- $\{ww^R \mid w \in \{a, b\}^*\}$
- $\{w \in \{a, b\}^* \mid w = w^R\}$

Argue that the pumping lemma does **not** hold for:

- $\{ww \mid w \in \{a, b\}^*\}$
- $\{a^{2^k} \mid k \geq 0\}$

Basic Properties of Context-Free Languages

Theorem

If L_1 and L_2 are context-free, then also

$$L_1 \cup L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^*$$

Basic Properties of Context-Free Languages

Theorem

If L_1 and L_2 are context-free, then also

$$L_1 \cup L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^*$$

Proof.

Let G_i be a context-free grammar with start variable S_i s.t.

$$L_i = L(G_i)$$

for $i = 1, 2$.

Basic Properties of Context-Free Languages

Theorem

If L_1 and L_2 are context-free, then also

$$L_1 \cup L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^*$$

Proof.

Let G_i be a context-free grammar with start variable S_i s.t.

$$L_i = L(G_i)$$

for $i = 1, 2$. Let G_1 and G_2 have no variables in common.

Basic Properties of Context-Free Languages

Theorem

If L_1 and L_2 are context-free, then also

$$L_1 \cup L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^*$$

Proof.

Let G_i be a context-free grammar with start variable S_i s.t.

$$L_i = L(G_i)$$

for $i = 1, 2$. Let G_1 and G_2 have no variables in common.

- $L_1 \cup L_2$:

Basic Properties of Context-Free Languages

Theorem

If L_1 and L_2 are context-free, then also

$$L_1 \cup L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^*$$

Proof.

Let G_i be a context-free grammar with start variable S_i s.t.

$$L_i = L(G_i)$$

for $i = 1, 2$. Let G_1 and G_2 have no variables in common.

- $L_1 \cup L_2$: Add rules $S \to S_1 \mid S_2$, and pick S as start variable.

Basic Properties of Context-Free Languages

Theorem

If L_1 and L_2 are context-free, then also

$$L_1 \cup L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^*$$

Proof.

Let G_i be a context-free grammar with start variable S_i s.t.

$$L_i = L(G_i)$$

for $i = 1, 2$. Let G_1 and G_2 have no variables in common.

- $L_1 \cup L_2$: Add rules $S \to S_1 \mid S_2$, and pick S as start variable.
- L_1^R:

Basic Properties of Context-Free Languages

Theorem

If L_1 and L_2 are context-free, then also

$$L_1 \cup L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^*$$

Proof.

Let G_i be a context-free grammar with start variable S_i s.t.

$$L_i = L(G_i)$$

for $i = 1, 2$. Let G_1 and G_2 have no variables in common.

- $L_1 \cup L_2$: Add rules $S \to S_1 \mid S_2$, and pick S as start variable.
- L_1^R: Reverse all right-hand sides ($x \to y$ becomes $x \to y^R$).

Basic Properties of Context-Free Languages

Theorem

If L_1 and L_2 are context-free, then also

$$L_1 \cup L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^*$$

Proof.

Let G_i be a context-free grammar with start variable S_i s.t.

$$L_i = L(G_i)$$

for $i = 1, 2$. Let G_1 and G_2 have no variables in common.

- $L_1 \cup L_2$: Add rules $S \to S_1 \mid S_2$, and pick S as start variable.
- L_1^R: Reverse all right-hand sides ($x \to y$ becomes $x \to y^R$).
- $L_1 L_2$:

Basic Properties of Context-Free Languages

Theorem

If L_1 and L_2 are context-free, then also

$$L_1 \cup L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^*$$

Proof.

Let G_i be a context-free grammar with start variable S_i s.t.

$$L_i = L(G_i)$$

for $i = 1, 2$. Let G_1 and G_2 have no variables in common.

- $L_1 \cup L_2$: Add rules $S \to S_1 \mid S_2$, and pick S as start variable.
- L_1^R: Reverse all right-hand sides ($x \to y$ becomes $x \to y^R$).
- $L_1 L_2$: Add $S \to S_1 S_2$, and pick S as start variable.

Basic Properties of Context-Free Languages

Theorem

If L_1 and L_2 are context-free, then also

$$L_1 \cup L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^*$$

Proof.

Let G_i be a context-free grammar with start variable S_i s.t.

$$L_i = L(G_i)$$

for $i = 1, 2$. Let G_1 and G_2 have no variables in common.

- $L_1 \cup L_2$: Add rules $S \to S_1 \mid S_2$, and pick S as start variable.
- L_1^R: Reverse all right-hand sides ($x \to y$ becomes $x \to y^R$).
- $L_1 L_2$: Add $S \to S_1 S_2$, and pick S as start variable.
- L_1^*:

Basic Properties of Context-Free Languages

Theorem

If L_1 and L_2 are context-free, then also

$$L_1 \cup L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^*$$

Proof.

Let G_i be a context-free grammar with start variable S_i s.t.

$$L_i = L(G_i)$$

for $i = 1, 2$. Let G_1 and G_2 have no variables in common.

- $L_1 \cup L_2$: Add rules $S \to S_1 \mid S_2$, and pick S as start variable.
- L_1^R: Reverse all right-hand sides ($x \to y$ becomes $x \to y^R$).
- $L_1 L_2$: Add $S \to S_1 S_2$, and pick S as start variable.
- L_1^*: Add $S \to S_1 S \mid \lambda$, and pick S as start variable.

□

Basic Properties of Context-Free Languages

The intersection $L_1 \cap L_2$ is not always context-free.
(for context free languages L_1 and L_2)

Basic Properties of Context-Free Languages

The intersection $L_1 \cap L_2$ is not always context-free.
(for context free languages L_1 and L_2)

The languages L_1 and L_2 are context-free:

$$L_1 = \{ a^n b^n c^m \mid n \geq 0 \land m \geq 0 \}$$
$$L_2 = \{ a^n b^m c^m \mid n \geq 0 \land m \geq 0 \}$$

Basic Properties of Context-Free Languages

The intersection $L_1 \cap L_2$ is not always context-free.
(for context free languages L_1 and L_2)

The languages L_1 and L_2 are context-free:

$$L_1 = \{ a^n b^n c^m \mid n \geq 0 \wedge m \geq 0 \}$$
$$L_2 = \{ a^n b^m c^m \mid n \geq 0 \wedge m \geq 0 \}$$

However $L_1 \cap L_2 = \{ a^n b^n c^n \mid n \geq 0 \}$ is **not** context-free.

Basic Properties of Context-Free Languages

The intersection $L_1 \cap L_2$ is not always context-free.
(for context free languages L_1 and L_2)

The languages L_1 and L_2 are context-free:

$$L_1 = \{ a^n b^n c^m \mid n \geq 0 \wedge m \geq 0 \}$$
$$L_2 = \{ a^n b^m c^m \mid n \geq 0 \wedge m \geq 0 \}$$

However $L_1 \cap L_2 = \{ a^n b^n c^n \mid n \geq 0 \}$ is **not** context-free.

Also $\overline{L_1}$ and $L_1 \setminus L_2$ are not always context-free.
(for context free languages L_1 and L_2)

Basic Properties of Context-Free Languages

The intersection $L_1 \cap L_2$ is not always context-free.
(for context free languages L_1 and L_2)

The languages L_1 and L_2 are context-free:

$$L_1 = \{ a^n b^n c^m \mid n \geq 0 \wedge m \geq 0 \}$$
$$L_2 = \{ a^n b^m c^m \mid n \geq 0 \wedge m \geq 0 \}$$

However $L_1 \cap L_2 = \{ a^n b^n c^n \mid n \geq 0 \}$ is **not** context-free.

Also $\overline{L_1}$ and $L_1 \setminus L_2$ are not always context-free.
(for context free languages L_1 and L_2)

Namely, we have:

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}} \qquad \overline{L_1} = \Sigma^* \setminus L_1$$

Basic Properties of Context-Free Languages

Theorem

If L_1 is context-free and L_2 regular, then $L_1 \cap L_2$ is context-free.

Basic Properties of Context-Free Languages

Theorem

If L_1 is context-free and L_2 regular, then $L_1 \cap L_2$ is context-free.

Construction

Let

- $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ be an NPDA accepting L_1, and
- $N = (R, \Sigma, \epsilon, r_0, G)$ a DFA accepting L_2.

Basic Properties of Context-Free Languages

Theorem

If L_1 is context-free and L_2 regular, then $L_1 \cap L_2$ is context-free.

Construction

Let

- $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ be an NPDA accepting L_1, and
- $N = (R, \Sigma, \epsilon, r_0, G)$ a DFA accepting L_2.

We construct an NPDA $\widehat{M} = (\widehat{Q}, \Sigma, \Gamma, \widehat{\delta}, \widehat{q}_0, z, \widehat{F})$ where

$$\widehat{Q} = Q \times R \qquad \widehat{q}_0 = (q_0, r_0) \qquad \widehat{F} = F \times G$$

Basic Properties of Context-Free Languages

Theorem

If L_1 is context-free and L_2 regular, then $L_1 \cap L_2$ is context-free.

Construction

Let

- $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ be an NPDA accepting L_1, and
- $N = (R, \Sigma, \epsilon, r_0, G)$ a DFA accepting L_2.

We construct an NPDA $\widehat{M} = (\widehat{Q}, \Sigma, \Gamma, \widehat{\delta}, \widehat{q}_0, z, \widehat{F})$ where

$$\widehat{Q} = Q \times R \qquad \widehat{q}_0 = (q_0, r_0) \qquad \widehat{F} = F \times G$$

The transition function $\widehat{\delta}$ is defined by:

Basic Properties of Context-Free Languages

Theorem

If L_1 is context-free and L_2 regular, then $L_1 \cap L_2$ is context-free.

Construction

Let

- $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ be an NPDA accepting L_1, and
- $N = (R, \Sigma, \epsilon, r_0, G)$ a DFA accepting L_2.

We construct an NPDA $\widehat{M} = (\widehat{Q}, \Sigma, \Gamma, \widehat{\delta}, \widehat{q}_0, z, \widehat{F})$ where

$$\widehat{Q} = Q \times R \qquad \widehat{q}_0 = (q_0, r_0) \qquad \widehat{F} = F \times G$$

The transition function $\widehat{\delta}$ is defined by:

- $\widehat{M}$: $(q, r) \xrightarrow{a[b/v]} (q', r')$ if M: $q \xrightarrow{a[b/v]} q'$ and N: $r \xrightarrow{a} r'$

Basic Properties of Context-Free Languages

Theorem

If L_1 is context-free and L_2 regular, then $L_1 \cap L_2$ is context-free.

Construction

Let

- $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ be an NPDA accepting L_1, and
- $N = (R, \Sigma, \epsilon, r_0, G)$ a DFA accepting L_2.

We construct an NPDA $\widehat{M} = (\widehat{Q}, \Sigma, \Gamma, \widehat{\delta}, \widehat{q}_0, z, \widehat{F})$ where

$$\widehat{Q} = Q \times R \qquad \widehat{q}_0 = (q_0, r_0) \qquad \widehat{F} = F \times G$$

The transition function $\widehat{\delta}$ is defined by:

- $\widehat{M}$: $(q, r) \xrightarrow{a[b/v]} (q', r')$ if M: $q \xrightarrow{a[b/v]} q'$ and N: $r \xrightarrow{a} r'$
- $\widehat{M}$: $(q, r) \xrightarrow{\lambda[b/v]} (q', r)$ if M: $q \xrightarrow{\lambda[b/v]} q'$

Basic Properties of Context-Free Languages

Theorem

If L_1 is context-free and L_2 regular, then $L_1 \cap L_2$ is context-free.

Construction

Let

- $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ be an NPDA accepting L_1, and
- $N = (R, \Sigma, \epsilon, r_0, G)$ a DFA accepting L_2.

We construct an NPDA $\widehat{M} = (\widehat{Q}, \Sigma, \Gamma, \widehat{\delta}, \widehat{q}_0, z, \widehat{F})$ where

$$\widehat{Q} = Q \times R \qquad \widehat{q}_0 = (q_0, r_0) \qquad \widehat{F} = F \times G$$

The transition function $\widehat{\delta}$ is defined by:

- $\widehat{M}$: $(q, r) \xrightarrow{a[b/v]} (q', r')$ if M: $q \xrightarrow{a[b/v]} q'$ and N: $r \xrightarrow{a} r'$
- $\widehat{M}$: $(q, r) \xrightarrow{\lambda[b/v]} (q', r)$ if M: $q \xrightarrow{\lambda[b/v]} q'$

Then $L(\widehat{M}) = L(M) \cap L(N)$.

Question

Question

Why does the construction not work for two NPDA's?
(instead of an NPDA and a DFA)

Basic Properties of Context-Free Languages

Theorem

If L_1 is context-free and L_2 regular, then $L_1 \backslash L_2$ is context-free.

Basic Properties of Context-Free Languages

Theorem

If L_1 is context-free and L_2 regular, then $L_1 \backslash L_2$ is context-free.

Proof.

$\overline{L_2}$ is regular, thus $L_1 \backslash L_2 = L_1 \cap \overline{L_2}$ is context-free. □

Basic Properties of Context-Free Languages

Theorem

If L_1 is context-free and L_2 regular, then $L_1 \backslash L_2$ is context-free.

Proof.

$\overline{L_2}$ is regular, thus $L_1 \backslash L_2 = L_1 \cap \overline{L_2}$ is context-free. □

$L_2 \backslash L_1$ is **not** always context-free. Namely

$$\overline{L_1} = \Sigma^* \backslash L_1$$

Applications

$L \setminus \{\lambda\}$ is context-free for every context-free language L.

Applications

$L \setminus \{\lambda\}$ is context-free for every context-free language L.

$\{ a^n b^n \mid n \geq 1000 \}$ is context-free.

Applications

$L \setminus \{\lambda\}$ is context-free for every context-free language L.

$\{ a^n b^n \mid n \geq 1000 \}$ is context-free.

Show that the language

$$L = \{ w \in \{a, b, c\}^* \mid n_a(w) = n_b(w) = n_c(w) \}$$

is **not** context-free.

Applications

$L \setminus \{\lambda\}$ is context-free for every context-free language L.

$\{ a^n b^n \mid n \geq 1000 \}$ is context-free.

Show that the language

$$L = \{ w \in \{a, b, c\}^* \mid n_a(w) = n_b(w) = n_c(w) \}$$

is **not** context-free.

For a contradiction, assume L was context-free.

Applications

$L \setminus \{\lambda\}$ is context-free for every context-free language L.

$\{ a^n b^n \mid n \geq 1000 \}$ is context-free.

Show that the language

$$L = \{ w \in \{a, b, c\}^* \mid n_a(w) = n_b(w) = n_c(w) \}$$

is **not** context-free.

For a contradiction, assume L was context-free.

The language $a^* b^* c^*$ is regular, thus

$$L \cap a^* b^* c^* = \{ a^n b^n c^n \mid n \geq 0 \}$$

would be context-free.

Applications

$L \setminus \{ \lambda \}$ is context-free for every context-free language L.

$\{ a^n b^n \mid n \geq 1000 \}$ is context-free.

Show that the language

$$L = \{ w \in \{a, b, c\}^* \mid n_a(w) = n_b(w) = n_c(w) \}$$

is **not** context-free.

For a contradiction, assume L was context-free.

The language $a^* b^* c^*$ is regular, thus

$$L \cap a^* b^* c^* = \{ a^n b^n c^n \mid n \geq 0 \}$$

would be context-free. However, we know that it is not.

Applications

$L \setminus \{ \lambda \}$ is context-free for every context-free language L.

$\{ a^n b^n \mid n \geq 1000 \}$ is context-free.

Show that the language

$$L = \{ w \in \{a, b, c\}^* \mid n_a(w) = n_b(w) = n_c(w) \}$$

is **not** context-free.

For a contradiction, assume L was context-free.

The language $a^* b^* c^*$ is regular, thus

$$L \cap a^* b^* c^* = \{ a^n b^n c^n \mid n \geq 0 \}$$

would be context-free. However, we know that it is not.

Contradiction. Thus L is not context-free.

Basic Questions about Context-Free Grammars

Given context-free grammar G and H.

Which of the following questions are **decidable**?

1. Given $w \in \Sigma^*$, do we have $w \in L(G)$?
2. Is $L(G)$ empty ?
3. Does $L(G) = \Sigma^*$ hold ?
4. Does $L(G)$ contain a palindrome ($w = w^R$) ?
5. Does $L(G) = L(H)$ hold ?
6. Is $L(G) \cap L(H)$ empty ?

Basic Questions about Context-Free Grammars

Given context-free grammar G and H.

Which of the following questions are **decidable**?

1. Given $w \in \Sigma^*$, do we have $w \in L(G)$?
2. Is $L(G)$ empty ?
3. Does $L(G) = \Sigma^*$ hold ?
4. Does $L(G)$ contain a palindrome ($w = w^R$) ?
5. Does $L(G) = L(H)$ hold ?
6. Is $L(G) \cap L(H)$ empty ?

Only the first two questions are decidable.

Basic Questions about Context-Free Grammars

Given context-free grammar G and H.

Which of the following questions are **decidable**?

1. Given $w \in \Sigma^*$, do we have $w \in L(G)$?
2. Is $L(G)$ empty ?
3. Does $L(G) = \Sigma^*$ hold ?
4. Does $L(G)$ contain a palindrome ($w = w^R$) ?
5. Does $L(G) = L(H)$ hold ?
6. Is $L(G) \cap L(H)$ empty ?

Only the first two questions are decidable.

Remove all λ and unit productions.

1. Words $S \Rightarrow^* v$ with $|v| \leq |w|$ can be computed in finite time.
2. $L(G)$ is empty $\iff$ starting variable is useless.

Basic Questions about Context-Free Grammars

Given context-free grammar G and H.

Which of the following questions are **decidable**?

1. Given $w \in \Sigma^*$, do we have $w \in L(G)$?
2. Is $L(G)$ empty ?
3. Does $L(G) = \Sigma^*$ hold ?
4. Does $L(G)$ contain a palindrome ($w = w^R$) ?
5. Does $L(G) = L(H)$ hold ?
6. Is $L(G) \cap L(H)$ empty ?

Only the first two questions are decidable.

Remove all λ and unit productions.

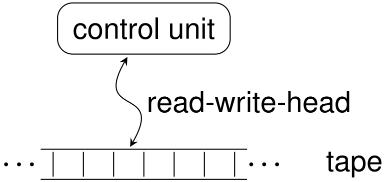
1. Words $S \Rightarrow^* v$ with $|v| \leq |w|$ can be computed in finite time.
2. $L(G)$ is empty $\iff$ starting variable is useless.

Surprisingly all other questions are undecidable.

Turing Machines

Turing machines can read and write the input word.

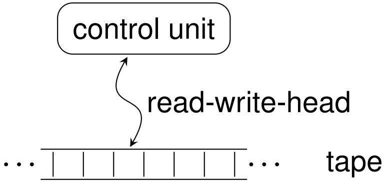
Input is written on a tape on which a read-write-head works.



Turing Machines

Turing machines can read and write the input word.

Input is written on a tape on which a read-write-head works.



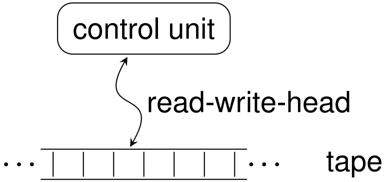
In each step:

- the read-write-head reads a symbol from the tape,
- overwrites the symbol, and
- moves one place to the left or right.

Turing Machines

Turing machines can read and write the input word.

Input is written on a tape on which a read-write-head works.



In each step:

- the read-write-head reads a symbol from the tape,
- overwrites the symbol, and
- moves one place to the left or right.

The tape is two-sided infinite: unlimited memory!

Turing Machines

Turing Machines

We introduce a blank symbol $\Box$. The initial tape content is

$$\cdots \Box\Box\Box\Box \text{ input word } \Box\Box\Box\Box \cdots$$

There is a finite set of states Q and a finite tape alphabet Γ.

Turing Machines

We introduce a blank symbol $\Box$. The initial tape content is

$$\cdots \Box\Box\Box\Box \text{ input word } \Box\Box\Box\Box \cdots$$

There is a finite set of states Q and a finite tape alphabet Γ.

The transition function δ has the form

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{ L, R \}$$

Here δ is a **partial function**: $\delta(q, a)$ may be undefined.

Turing Machines

We introduce a blank symbol $\Box$. The initial tape content is

$$\cdots \Box\Box\Box\Box \ \text{input word} \ \Box\Box\Box\Box \cdots$$

There is a finite set of states Q and a finite tape alphabet Γ.

The transition function δ has the form

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{ L, R \}$$

Here δ is a **partial function**: $\delta(q, a)$ may be undefined.

$\delta(q, a) = (q', b, X)$ means: if

- the machine is in state q, and
- the head reads a from the tape

then

- then a is overwritten by b,
- the head moves 1 position left if $X = L$, right if $X = R$, and
- the machine switches to state q'.

Turing Machines

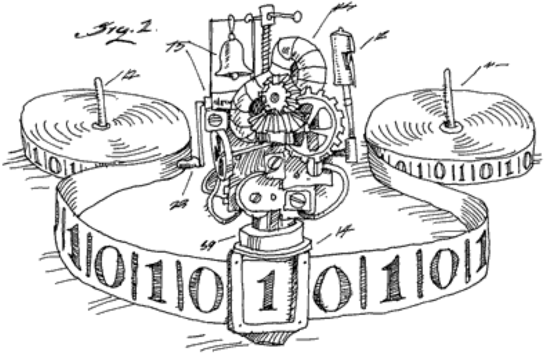
A deterministic Turing machine, short TM, is a 7-tuple

$$(Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$$

where

- Q is a finite set of states,
- $\Sigma \subseteq \Gamma \setminus \{ \Box \}$ a finite input alphabet,
- Γ a finite tape alphabet,
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{ L, R \}$ a partial transition function,
- q_0 the starting state,
- $\Box \in \Gamma$ the blank symbol,
- $F \subseteq Q$ a set of final (accepting) states.

Turing Machines



Turing Machine Configuration

A **configuration** of a TM is a word vqw ($q \in Q$, $v, w \in \Gamma^*$).

- the machine is in state q
- the tape content is vw (extended with $\Box$'s left and right)
- the head stands on the first symbol of w

Turing Machine Configuration

A **configuration** of a TM is a word vqw ($q \in Q$, $v, w \in \Gamma^*$).

- the machine is in state q
- the tape content is vw (extended with $\Box$'s left and right)
- the head stands on the first symbol of w

The computation steps are as follows:

$$vqaw \vdash vbq'w \qquad \text{if } \delta(q, a) = (q', b, R)$$

$$vq\lambda \vdash vbq'\lambda \qquad \text{if } \delta(q, \Box) = (q', b, R)$$

Note that λ is the empty word, not a symbol!

Turing Machine Configuration

A **configuration** of a TM is a word vqw ($q \in Q$, $v, w \in \Gamma^*$).

- the machine is in state q
- the tape content is vw (extended with $\Box$'s left and right)
- the head stands on the first symbol of w

The computation steps are as follows:

$$
\begin{array}{rcll}
vqaw & \vdash & vbq'w & \text{if } \delta(q,a) = (q',b,R) \\
vq\lambda & \vdash & vbq'\lambda & \text{if } \delta(q,\Box) = (q',b,R) \\
vcqaw & \vdash & vq'cbw & \text{if } \delta(q,a) = (q',b,L) \\
\lambda qaw & \vdash & \lambda q'\Box bw & \text{if } \delta(q,a) = (q',b,L) \\
vcq\lambda & \vdash & vq'cb\lambda & \text{if } \delta(q,\Box) = (q',b,L) \\
\lambda q\lambda & \vdash & \lambda q'\Box b\lambda & \text{if } \delta(q,\Box) = (q',b,L)
\end{array}
$$

Note that λ is the empty word, not a symbol!

Turing Machines and Languages

The **language $L(M)$ accepted by** TM $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ is

$$\{\, w \in \Sigma^* \mid q_0 w \vdash^* uqv \text{ for some } q \in F,\ u, v \in \Gamma^* \,\}$$

Turing Machines and Languages

The language $L(M)$ accepted by TM $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ is

$$\{\, w \in \Sigma^* \mid q_0 w \vdash^* uqv \text{ for some } q \in F,\ u, v \in \Gamma^* \,\}$$

A **halting state** is a configuration $vqaw$ (or $vq\lambda$) such that

$\delta(q, a)$ (or $\delta(q, \Box)$)

is **undefined**.

Turing Machines and Languages

The language $L(M)$ accepted by TM $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ is

$$\{\, w \in \Sigma^* \mid q_0 w \vdash^* uqv \text{ for some } q \in F,\ u, v \in \Gamma^* \,\}$$

A **halting state** is a configuration $vqaw$ (or $vq\lambda$) such that

$$\delta(q, a) \qquad\qquad \text{(or } \delta(q, \Box)\text{)}$$

is **undefined**.

Note: an execution can be infinite (never reach a halting state).

Turing Machines and Languages

The language $L(M)$ accepted by TM $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ is

$$\{\, w \in \Sigma^* \mid q_0 w \vdash^* uqv \;\text{ for some } q \in F,\; u, v \in \Gamma^* \,\}$$

A **halting state** is a configuration $vqaw$ (or $vq\lambda$) such that

$$\delta(q, a) \qquad\qquad \text{(or } \delta(q, \Box)\text{)}$$

is **undefined**.

Note: an execution can be infinite (never reach a halting state).

Assumption: $\delta(q, a)$ is undefined for every $q \in F$ and $a \in \Gamma$.

Turing Machines and Languages

The language $L(M)$ accepted by TM $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ is

$$\{\, w \in \Sigma^* \mid q_0 w \vdash^* uqv \text{ for some } q \in F,\ u, v \in \Gamma^* \,\}$$

A **halting state** is a configuration $vqaw$ (or $vq\lambda$) such that

$$\delta(q, a) \qquad\qquad (\text{or } \delta(q, \Box))$$

is **undefined**.

Note: an execution can be infinite (never reach a halting state).

Assumption: $\delta(q, a)$ is undefined for every $q \in F$ and $a \in \Gamma$.

So a configuration vqw with $q \in F$ is a halting state.

Turing Machines and Languages

The language $L(M)$ accepted by TM $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ is

$$\{ w \in \Sigma^* \mid q_0 w \vdash^* uqv \text{ for some } q \in F,\ u, v \in \Gamma^* \}$$

A **halting state** is a configuration $vqaw$ (or $vq\lambda$) such that

$$\delta(q, a) \qquad\qquad (\text{or } \delta(q, \Box))$$

is **undefined**.

Note: an execution can be infinite (never reach a halting state).

Assumption: $\delta(q, a)$ is undefined for every $q \in F$ and $a \in \Gamma$.

So a configuration vqw with $q \in F$ is a halting state.

Note that $w \notin L(M)$ can have two causes:

- the execution halts in a state $q \notin F$, or
- the execution is infinite (never halts).

Example

We construct a TM M with $L(M) = \{ a^n b^n c^n \mid n \geq 1 \}$.

Example

We construct a TM M with $L(M) = \{ a^n b^n c^n \mid n \geq 1 \}$.

Idea: stepwise replace one a by 0, one b by 1 and one c by 2.

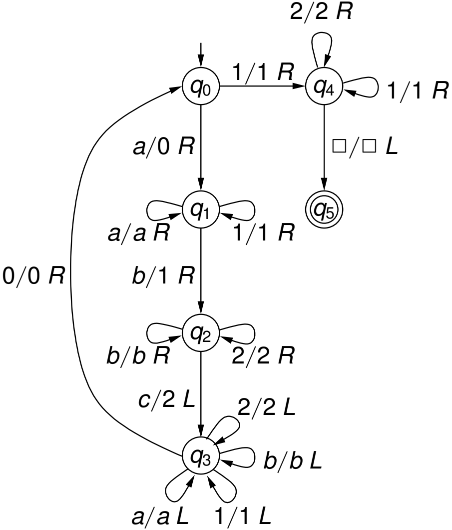
Example

We construct a TM M with $L(M) = \{ a^n b^n c^n \mid n \geq 1 \}$.

Idea: stepwise replace one a by 0, one b by 1 and one c by 2.

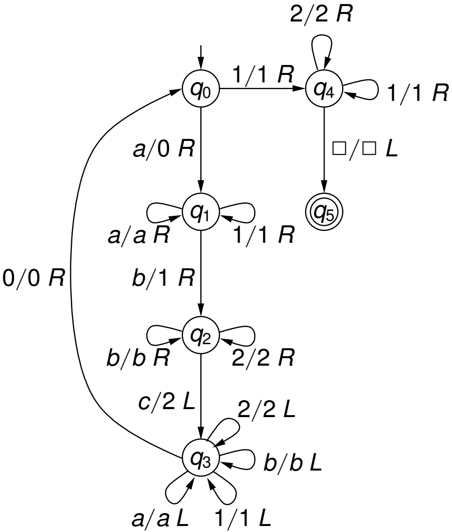
- $\Sigma = \{ a, b, c \}$ and $\Gamma = \{ a, b, c, 0, 1, 2, \Box \}$
- q_0: Read a, replace by 0, move right and switch to q_1.
- q_1: Keep moving right until we read b.
  Replace b by 1, move right and switch to q_2.
- q_2: Keep moving right until we read c.
  Replace c by 2, move left and switch to q_3.
- q_3: Keep moving left until we read 0.
  Move right and switch back to q_0.
- If we read 1 in q_0, switch to q_4.
- q_4: Keep moving right to check whether there are a's, b's or c's left. If not, then go to final state q_5.

Example



2/2 R
1/1 R
q0
q4
1/1 R
a/0 R
□/□ L
q1
q5
a/a R
1/1 R
0/0 R
b/1 R
q2
b/b R
2/2 R
c/2 L
2/2 L
q3
b/b L
a/a L
1/1 L

Example



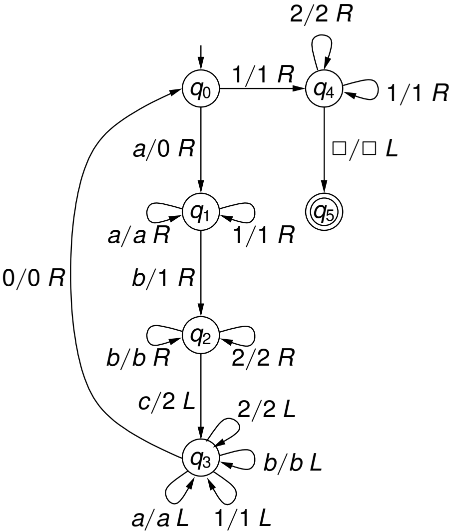
$q_0 aabbcc$

$\vdash^+ 0 q_0 a1b2c$

$\vdash^+ 00 q_0 1122$

$\vdash^+ 00112 q_5 2$

Example



$q_0 aabbcc$

$\vdash^+ 0 q_0 a1b2c$

$\vdash^+ 00 q_0 1122$

$\vdash^+ 00112 q_5 2$

$q_0 aabbbcc$

$\vdash^+ 0 q_0 a1bb2c$

$\vdash^+ 00 q_0 11b22$

$\vdash^+ 0011 q_4 b22$

Exercise

(Groups of two, 1 minutes)

Construct a Turing machine accepting all words of **odd** length over the alphabet $\Sigma = \{a, b\}$.

Recursively Enumerable Languages

A language L is **recursively enumerable** if L is accepted by a (deterministic) Turing machine.

Recursively Enumerable Languages

A language L is **recursively enumerable** if L is accepted by a (deterministic) Turing machine.

Extensions of Turing machines, such as

- multiple tapes
- nondeterminism
- ...

do **not** give extra expressive power.

Recursively Enumerable Languages

A language L is **recursively enumerable** if L is accepted by a (deterministic) Turing machine.

Extensions of Turing machines, such as

- multiple tapes
- nondeterminism
- ...

do **not** give extra expressive power.

Church-Turing thesis: Every computation of a computer can be simulated by a deterministic Turing machine.

Alonzo Church & Alan Turing

Alonzo Church (1903-1995) is inventor of the λ-calculus.

Alan Turing (1912-1954)

- introduced the Turing machine,
- invented the Turing test,
- played a key role in cracking the German Enigma machine.

In 1938, both proved undecidability of validity in predicate logic.

And there is of couse the famous Church-Turing thesis.

Looking Back

- Proving that a language is not context-free
  (Pumping lemma for context-free languages)
- If L_1, L_2 context-free, then
  - $L_1 \cup L_2$, L_1^R, $L_1 L_2$, L_1^* context-free
  - $L_1 \cap L_2$, $\overline{L_1}$, $L_1 \setminus L_2$ not always context-free
- Turing machines: automata with unlimited memory (tape)

Looking Forward

Read:

- Linz 8.1–8.2, 9.1, 9.3

Do the following exercises:

- Linz 8.1: 2, 3, 5, 8c
- Linz 8.2: 1, 11, 22, 23
- Linz 9.1: 4, 5, 7d,e, 8

Following lecture:

- Variations of Turing machines
- Recursive and recursively enumerable languages
- Unrestricted, context-sensitive grammars
- Linear bounded automata