Huiswerk

- Huiswerkopgave 2: deadline dinsdag 3 november 2026

Theorem (Pumping Lemma for context-free languages)

$\forall$ *for every context-free language* L

$\exists$ *there exists a constant* $n \geq 2$

such that

$\forall$ *for every* $u \in L$

with $|u| \geq n$

$\exists$ *there exists a decomposition* $u = vwxyz$

such that

(1) $|wy| \geq 1$

(2) $|wxy| \leq n$,

$\forall$ *(3) for all* $m \geq 0$, $vw^m xy^m z \in L$

[M] Thm. 6.1

Example

AnBnCn is not context-free.

[M] E 6.3

$u = a^n b^n c^n$

Argument works also for $L = \{ x \in \{a, b, c\}^* \mid n_a(x) = n_b(x) = n_c(x) \}$

Application of pumping lemma:
mainly to prove that a language L cannot be generated by a context-free grammar.

How?
Find a string $u \in L$ with $|u| \geq n$ that cannot be pumped up!

What is n?
What should u be?
What can v, w, x, y and z be?
What should m be?

Suppose that there exists context-free grammar G with $L(G) = L$.
Let $n \geq 2$ be the integer from the pumping lemma.
We prove:
There exists $u \in L$ with $|u| \geq n$, such that
for every five strings v, w, x, y and z such that $u = vwxyz$
if
1. $|wy| \geq 1$
2. $|wxy| \leq n$

then
3. there exists $m \geq 0$, such that $vw^m xy^m z$ does not belong to L

Example

AnBnCn is not context-free.

[M] E 6.3

$u = a^n b^n c^n$

Argument works also for $L = \{ x \in \{a, b, c\}^* \mid n_a(x) = n_b(x) = n_c(x) \}$

Example

XX is not context-free.

[M] E 6.4

Applying the Pumping Lemma

Example

AnBnCn is not context-free.

[M] E 6.3

$u = a^n b^n c^n$

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

Example

XX is not context-free.

[M] E 6.4

$u = a^n b^n a^n b^n$

$\{ a^i b^j a^i b^j \mid i, j \geq 0 \}$

Example

$\{ x \in \{a, b, c\}^* \mid n_a(x) < n_b(x) \text{ and } n_a(x) < n_c(x) \}$ is not context-free.

[M] E 6.5

ABOVE

$L = \{ x \in \{a, b, c\}^* \mid n_a(x) < n_b(x) \text{ and } n_a(x) < n_c(x) \}$ is not context-free.

Proof by contradiction.

Suppose L is context-free, then there exists a pumping constant n for L.

Choose $u = a^n b^{n+1} c^{n+1}$. Then $u \in L$, and $|u| \geq n$.

This means that we can pump u within the language L.

Consider a decomposition $u = vwxyz$ that satisfies the pumping lemma, in particular $|wxy| \leq n$.

Case 1: wy contains a letter a. Then wy cannot contain letter c (otherwise $|wxy| > n$). Now $u_2 = vw^2xy^2z$ contains more a's than u, so at least $n + 1$, while u_2 still contains $n + 1$ c's. Hence $u_2 \notin L$.

Case 2: wy contains no a. Then wy contains at least one b or one c (or both). Then $u_0 = vw^0xy^0z = vxz$ has still n a's, but less than $n + 1$ b's or less than $n + 1$ c's (depending on which letter is in wy). Hence $u_0 \notin L$.

These are the only two possibilities for the decomposition $vwxyz$, in both cases we see that pumping leads out of the language L.

Hence u cannot be pumped.

Contradiction; so L is not context-free.

Example

The Set of Legal C Programs is Not a CFL

[M] E 6.6

Choose $u =$

```
main(){int aaa...a;aaa...a=aaa...a;}
```

where `aaa...a` contains $n + 1$ a's

From Automaten:

FA $M_i = (Q_i, \Sigma, q_i, A_i, \delta_i) \quad i = 1, 2$

Product construction

construct FA $M = (Q, \Sigma, q_0, A, \delta)$ such that

- $Q = Q_1 \times Q_2$
- $q_0 = (q_1, q_2)$
- $\delta(\ (p, q), \sigma) = (\ \delta_1(p, \sigma), \delta_2(q, \sigma)\ )$
- A as needed

Theorem (2.15 Parallel simulation)

- $A = \{(p, q) \mid p \in A_1$ *or* $q \in A_2\}$, *then* $L(M) = L(M_1) \cup L(M_2)$
- $A = \{(p, q) \mid p \in A_1$ *and* $q \in A_2\}$, *then* $L(M) = L(M_1) \cap L(M_2)$
- $A = \{(p, q) \mid p \in A_1$ *and* $q \notin A_2\}$, *then* $L(M) = L(M_1) - L(M_2)$

Proof...

[M] Sect 2.2

From Automaten:

Regular languages are closed under

- Boolean operations (complement, union, intersection, minus)
- Regular operations (union, concatenation, star)
- Reverse (mirror)
- (inverse) Homomorphism

From lecture 2:

Using building blocks

Theorem

If L_1, L_2 are CFL, then so are $L_1 \cup L_2$, $L_1 L_2$ and L_1^*.

$G_i = (V_i, \Sigma, S_i, P_i)$, having no variables in common.

Construction

$G = (V_1 \cup V_2 \cup \{S\}, \Sigma, S, P)$, new axiom S

– $P = P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}$ $L(G) = L(G_1) \cup L(G_2)$

– $P = P_1 \cup P_2 \cup \{S \to S_1 S_2\}$ $L(G) = L(G_1)L(G_2)$

$G = (V_1 \cup \{S\}, \Sigma, S, P)$, new axiom S

– $P = P_1 \cup \{S \to SS_1, S \to \Lambda\}$ $L(G) = L(G_1)^*$

[M] Thm 4.9

How about

- $L_1 \cap L_2$
- $L_1 - L_2$
- L_1'

for CFLs L_1 and L_2 ?

From lecture 2:

Example

AnBnCn is intersection of two context-free languages.

$L_1 = \{a^i b^i c^k \mid i, k \geq 0\}$
$L_2 = \{a^i b^k c^k \mid i, k \geq 0\}$

[M] E 6.10

Hence, CFL is not closed under intersection

Example

AnBnCn is intersection of two context-free languages.

[M] E 6.10

Hence, CFL is not closed under intersection

$L_1 \cap L_2 = (L_1' \cup L_2')'$

Hence, CFL is not closed under complement

$L_1' = \Sigma^* - L_1$

Hence, CFL is not closed under setminus

Example

Complement of XX
$= \{ x \in \{a, b\}^* \mid |x| \text{ is odd } \} \cup \{ xy \mid x, y \in \{a, b\}^*, |x| = |y|, x \neq y \}$
is context-free

[M] E 6.11

Indeed, CFL is not closed under complement

Example

Complement of $AnBnCn$ is context-free.

[M] E 6.12

Example

Complement of *AnBnCn* is context-free.

$AnBnCn = L_1 \cap L_2 \cap L_3$, with
$L_1 = \{a^i b^j c^k \mid i \leq j\}$
$L_2 = \{a^i b^j c^k \mid j \leq k\}$
$L_3 = \{a^i b^j c^k \mid k \leq i\}$
[M] E 6.12

Example

Complement of $\{x \in \{a, b, c\}^* \mid n_a(x) = n_b(x) = n_c(x)\}$ is context-free.

$\{x \in \{a, b, c\}^* \mid n_a(x) = n_b(x) = n_c(x)\} = A_1 \cap A_2 \cap A_3$, with
$A_1 = \{x \in \{a, b, c\}^* \mid n_a(x) \leq n_b(x)\}$
$A_2 = \{x \in \{a, b, c\}^* \mid n_b(x) \leq n_c(x)\}$
$A_3 = \{x \in \{a, b, c\}^* \mid n_c(x) \leq n_a(x)\}$

[M] E 6.12

Example

$L_1 = \{\ a^{2n} b^n \mid n \geq 1\ \}^*$

$a^{16} b^8 a^8 b^4 a^4 b^2 a^2 b^1$

$L_2 = a^* \{\ b^n a^n \mid n \geq 1\ \}^* \{\ b\ \}$

From Automaten:

FA $M_i = (Q_i, \Sigma, q_i, A_i, \delta_i) \quad i = 1, 2$

Product construction

construct FA $M = (Q, \Sigma, q_0, A, \delta)$ such that

- $Q = Q_1 \times Q_2$
- $q_0 = (q_1, q_2)$
- $\delta(\ (p, q), \sigma) = (\ \delta_1(p, \sigma), \delta_2(q, \sigma)\ )$
- A as needed

Theorem (2.15 Parallel simulation)

- $A = \{(p, q) \mid p \in A_1$ *or* $q \in A_2\}$, *then* $L(M) = L(M_1) \cup L(M_2)$
- $A = \{(p, q) \mid p \in A_1$ *and* $q \in A_2\}$, *then* $L(M) = L(M_1) \cap L(M_2)$
- $A = \{(p, q) \mid p \in A_1$ *and* $q \notin A_2\}$, *then* $L(M) = L(M_1) - L(M_2)$

Proof...

[M] Sect 2.2

Theorem

If L_1 is a CFL, and L_2 in REG, then $L_1 \cap L_2$ is CFL.

[M] Thm 6.13

product construction

PDA $M_1 = (Q_1, \Sigma, \Gamma, q_1, Z_1, A_1, \delta_1)$

FA $M_2 = (Q_2, \Sigma, q_2, A_2, \delta_2)$

$Q = Q_1 \times Q_2 \quad q_0 = \langle q_1, q_2 \rangle \quad A = A_1 \times A_2$

$\delta(\langle p_1, q_1 \rangle, \sigma, X) \ni (\langle p_2, q_2 \rangle, \alpha)$
whenever $\delta_1(p_1, \sigma, X) \ni (p_2, \alpha)$ and $\delta_2(q_1, \sigma) = q_2$

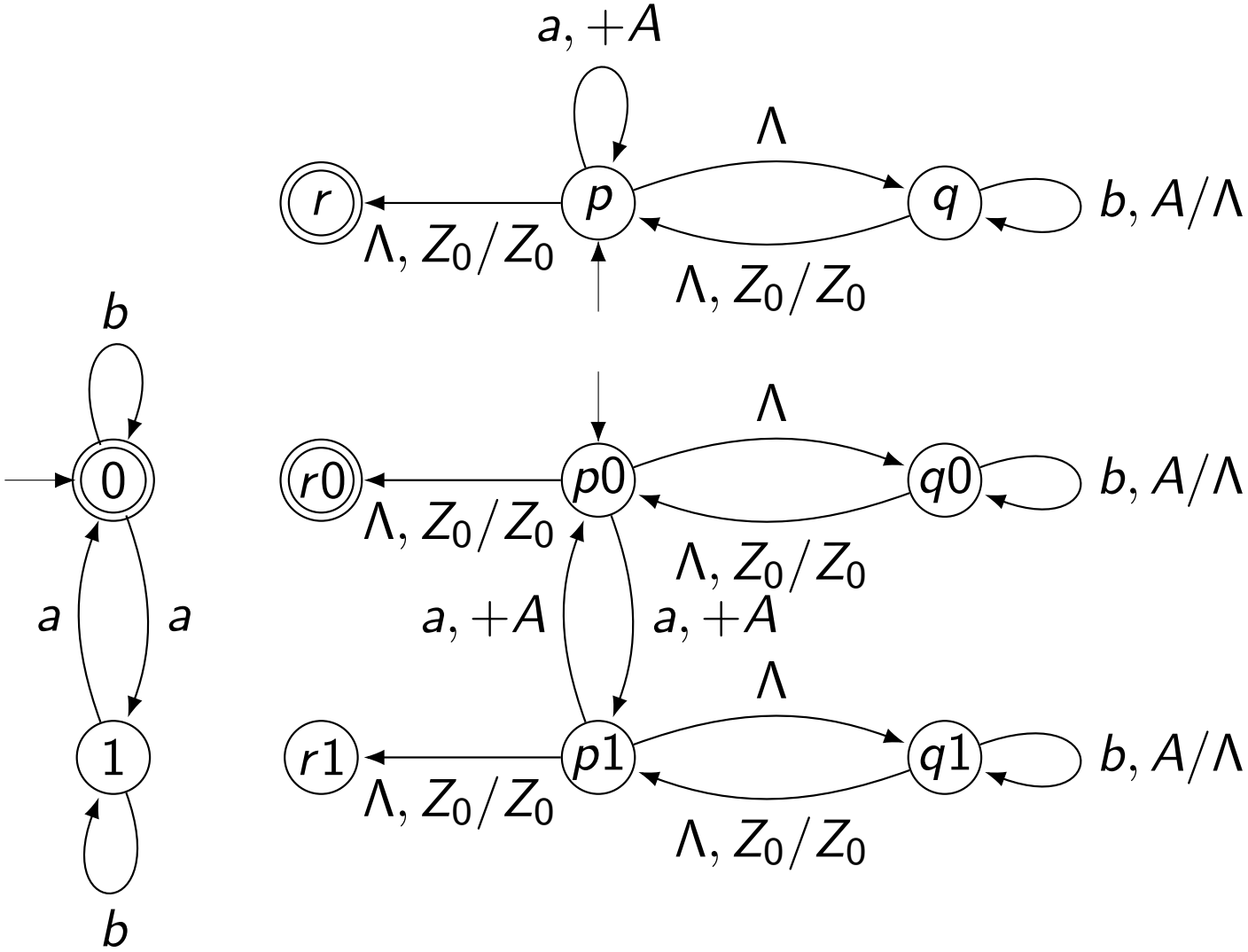
$\delta(\langle p_1, q \rangle, \Lambda, X) \ni (\langle p_2, q \rangle, \alpha)$
whenever $\delta_1(p_1, \Lambda, X) \ni (p_2, \alpha)$ and $q \in Q_2$

The inductive proof that this construction works does not have to be known for the exam.

Also CFG proof

Example: product construction

$\{\ a^n b^n \mid n \geq 1\ \}^* \cap \{\ w \in \{a, b\}^* \mid n_a(x) \text{ even}\ \}$



Non-determinism of PDA

- enables $L(M_1) \cup L(M_2)$
- 'prevents' $L(M_1)'$ (also Λ-transitions)

If L is accepted by DPDA without Λ-transitions, then so is L'

Even: if L is accepted by DPDA, then so is L'

Hence, if L is CFL and L' is not, then there is no DPDA for L
Not reversed (see *Pal*)

Decision problems for CFL

"given a CFL L, does it have property ... ?" yes/no
input CFG G

Given CFG G [G_1 and G_2]
– and given a string x, is $x \in L(G)$? membership problem
convert G to ChNF, and try all derivations of length $2|x| - 1$
(special case if $x = \Lambda$)
try all derivations up to...
Cocke, Younger, and Kasami (1967): n^3 (with DP)
Earley (1970): n^3 (and n^2 if G is unambiguous)

Decision problems for CFL

– is $L(G) \neq \varnothing$? non-emptiness
  is S useful?
  pumping lemma
– is $L(G)$ infinite?
  pumping lemma

Decision problems for CFL

– is $L(G_1) \cap L(G_2)$ nonempty? [M] Thm 9.20
– is $L(G) = \Sigma^*$? [M] Thm 9.23
– is $L(G_1) \subseteq L(G_2)$?
$L(G) = \Sigma^*$, if and only if $\Sigma^* \subseteq L(G)$

All undecidable

Given context-free L and regular R

– is $R \subseteq L$?

– is $L \subseteq R$?

ABOVE

$R \subseteq L$?

Special case $R = \Sigma^*$

$\Sigma^* \subseteq L$ iff $L = \Sigma^*$ undecidable

$L \subseteq R$?

iff $L \cap R' = \varnothing$

regular languages are closed under complement

CFL closed under intersection with regular languages

emptiness context-free language decidable

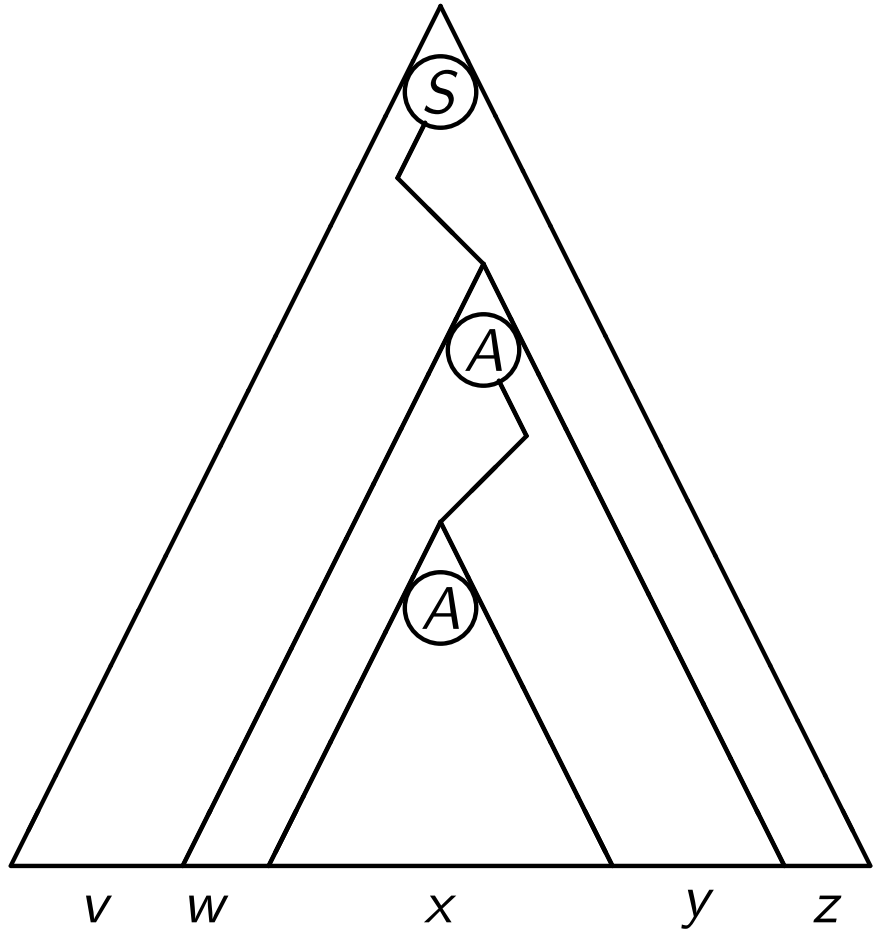
Thanks to HJH for the slides so far

Section 7

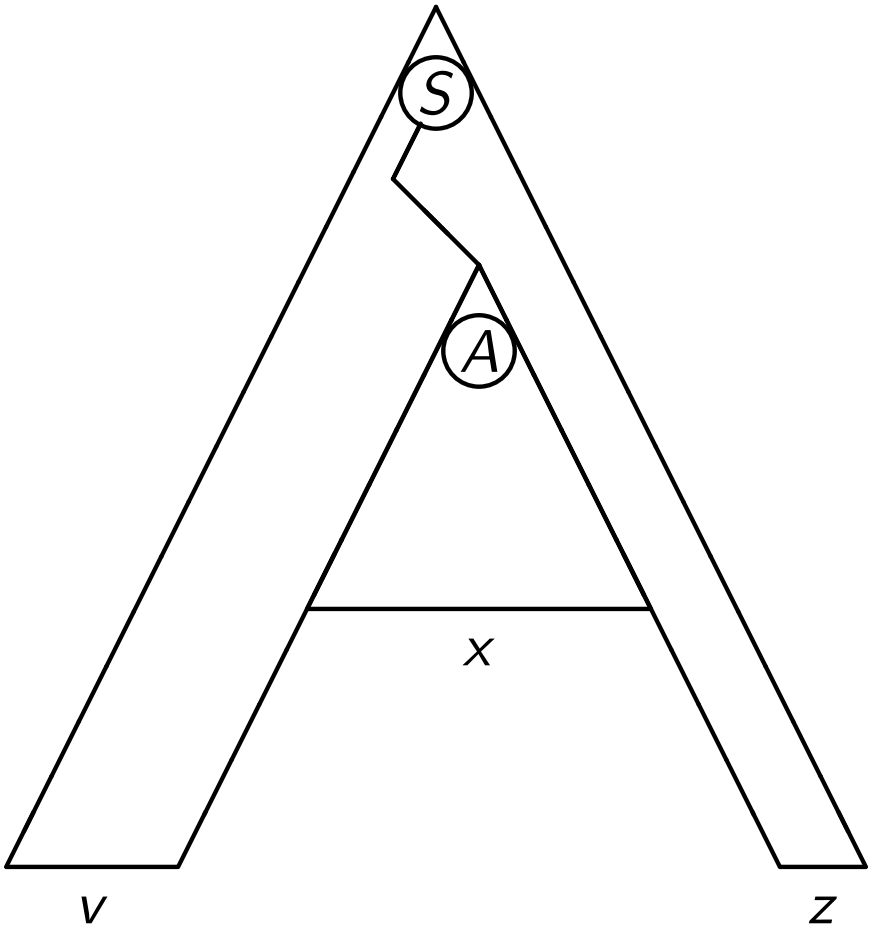
Turing Machines

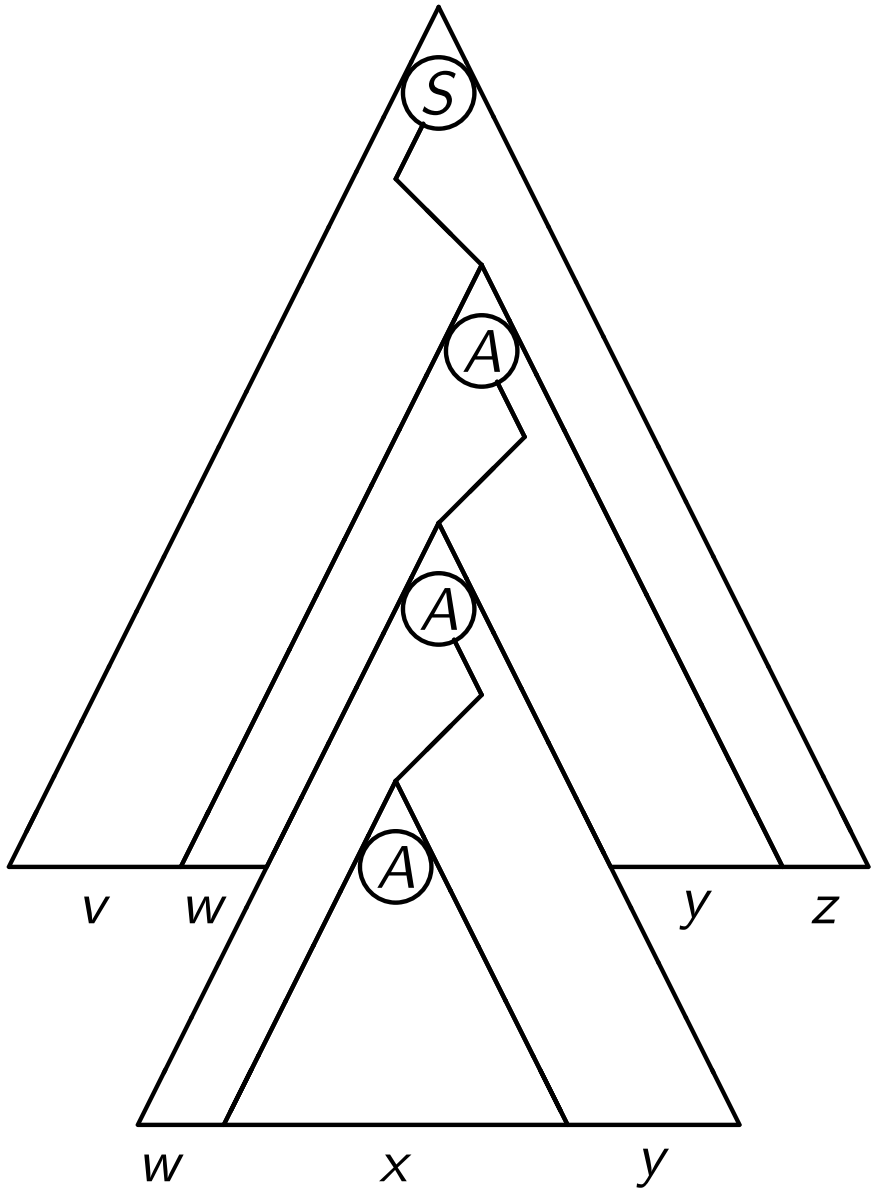
4 Turing Machines
- A General Model of Computation
- Turing Machines as Language Acceptors

Pumping Lemma for CFLs



Pumping Lemma for CFLs





7. Turing Machines

reg. languages	FA	reg. grammar	reg. expression
determ. cf. languages	DPDA		
cf. languages	PDA	cf. grammar	
cs. languages	LBA	cs. grammar	
re. languages	TM	unrestr. grammar	

7.1. A General Model of Computation

$AnBnCn = \{a^i b^i c^i \mid i \geq 0\}$

$XX = \{xx \mid x \in \{a, b\}^*\}$

Assumptions about a human computer working with a pencil and paper:

1. The only things written on the paper are symbols from some fixed finite alphabet;
2. Each step taken by the computer depends only on the symbol he is currently examining and on his "state of mind" at the time;
3. Although his state of mind may change as a result of his examining different symbols, only a finite number of distinct states of mind are possible.

Actions of a human computer on a sheet of paper:

1. Examining an individual symbol on the paper;
2. Erasing a symbol or replacing it by another;
3. Transferring attention from one symbol to another nearby symbol.

Turing machine

Turing machine has a finite alphabet of symbols.
    (actually two alphabets. . .)
Turing machine has a finite number of states.
Turing machine has a *tape*
    for reading input,
    as workspace,
    and for writing output (if applicable).
Tape is linear, instead of 2-dimensional.
Tape has a left end and is potentially infinite to the right.
Tape is marked off into squares, each of which can hold one symbol.
Tape head is centered on one square of the tape for reading and writing.

A move of a Turing machine consists of:

1. Changing from the current state to another, possibly different state;
2. Replacing the symbol in the current square by another, possibly different symbol;
3. Leaving the tape head on the current square, or moving it one square to the right, or moving it one square to the left if it is not already on the leftmost square.

Just like FA and PDA, Turing machine

- may be used to accept a language
- has a finite number of states

Just like FA, but unlike PDA

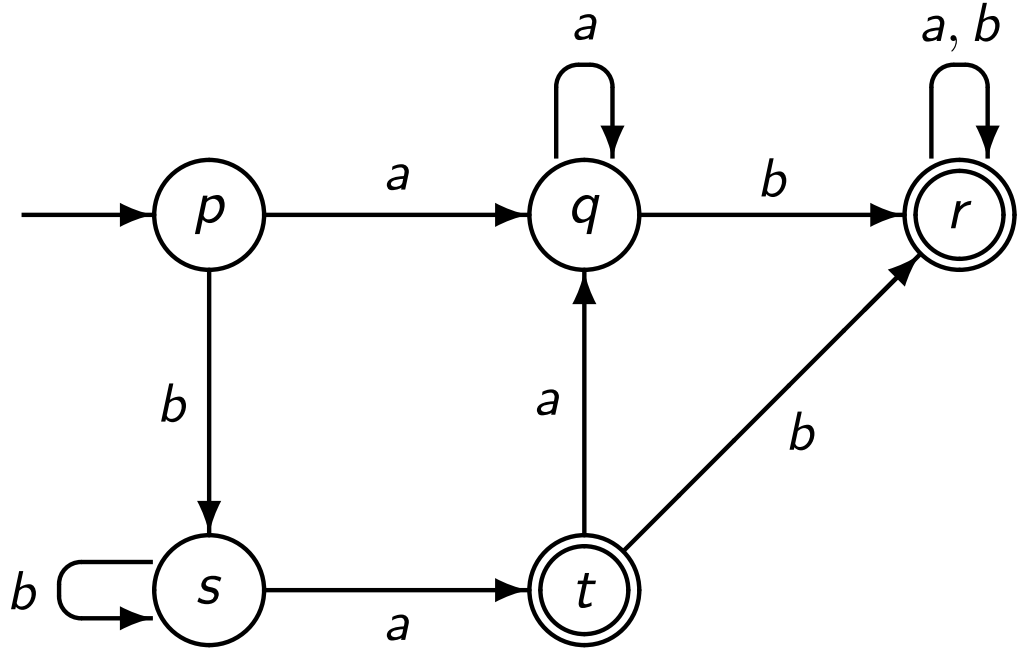
- by default TM is deterministic

Unlike FA and PDA, Turing machine

- may also be used to compute a function *
- is not restricted to reading input left-to-right *
- does not have to read all input *
- does not have a set of accepting states, but has two *halt* states: one for acceptance and one for rejection (in case of computing a function, . . .)
- might not decide to halt

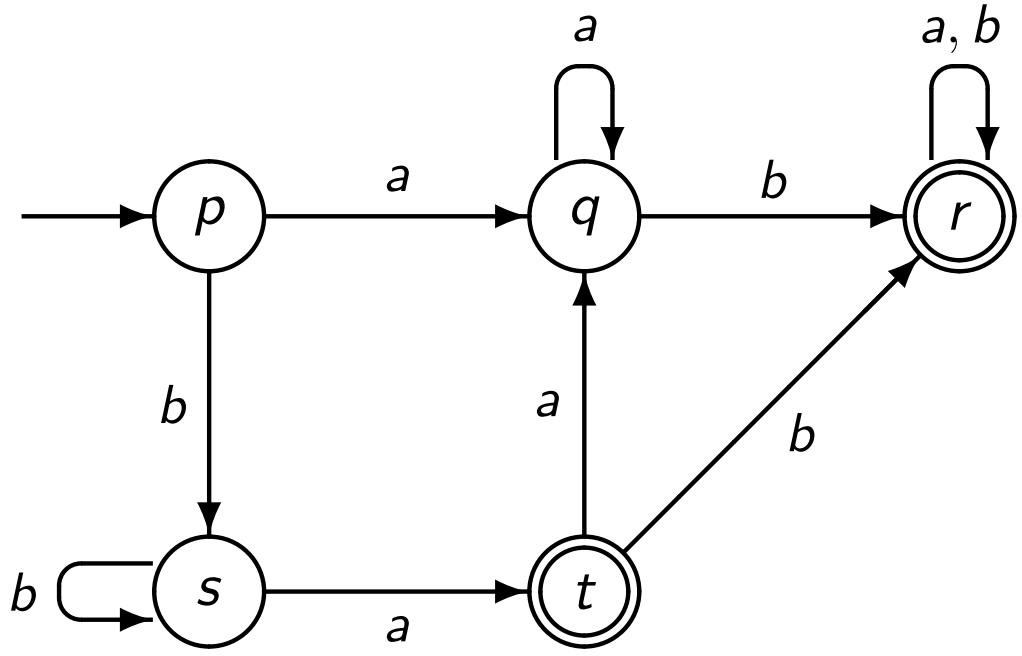
* = just like human computer

Example.
An FA Accepting $L = \ldots$



Example.

An FA Accepting $L = \{a, b\}^*\{ab\}\{a, b\}^* \cup \{a, b\}^*\{ba\}$



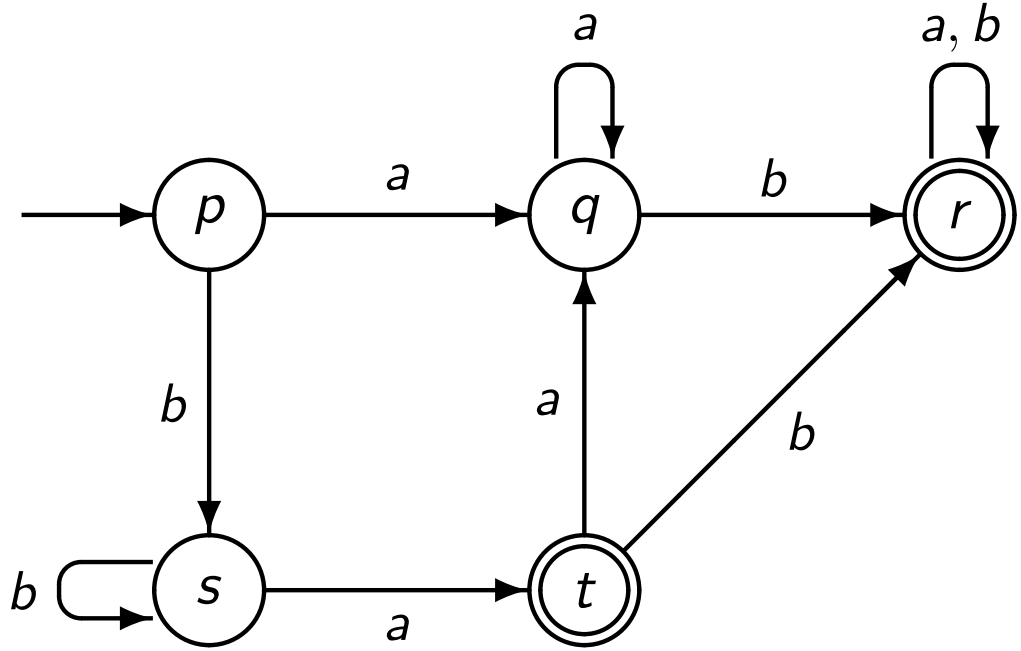
Why two accepting states?

7.2. Turing Machines as Language Acceptors

Example 7.3. A TM Accepting a Regular Language

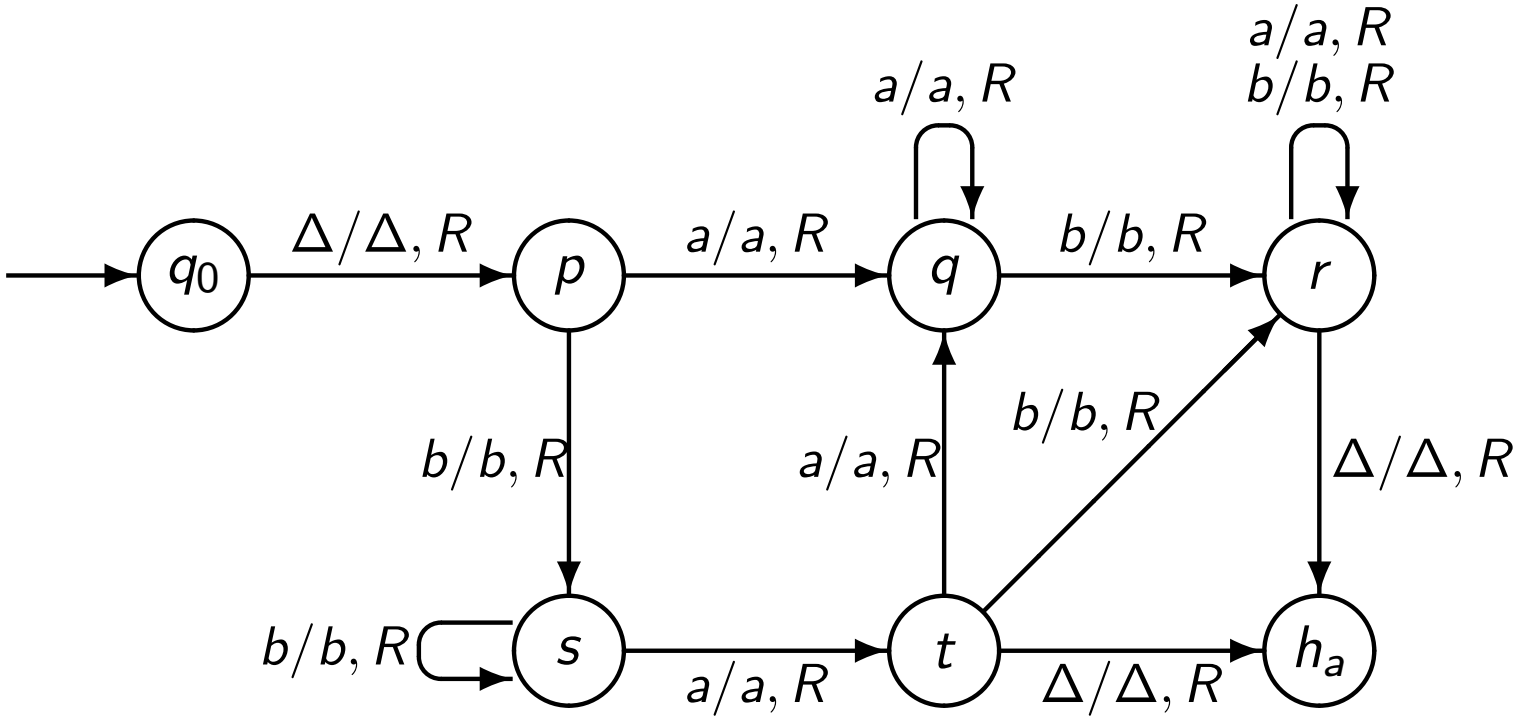
$L = \{a, b\}^*\{ab\}\{a, b\}^* \cup \{a, b\}^*\{ba\}$

First a finite automaton:



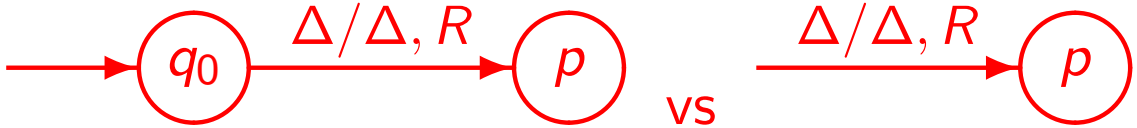
Example 7.3. A TM Accepting a Regular Language

$L = \{a, b\}^*\{ab\}\{a, b\}^* \cup \{a, b\}^*\{ba\}$



Δ vs Λ

Λ vs $\{\Lambda\}$ vs $\emptyset$



Example 7.3. A TM Accepting a Regular Language
$L = \{a, b\}^*\{ab\}\{a, b\}^* \cup \{a, b\}^*\{ba\}$
First a finite automaton, then a Turing machine
This conversion works in general for FAs.
As a result,
- only moves to the right,
- no modifications of symbols on tape,
- no moves to the reject state, but ...

In this case,
- we could modify TM, so that it does not always read entire input.

Example 7.5. A TM Accepting $XX = \{xx \mid x \in \{a, b\}^*\}$

$$\underline{\triangle} aabaab$$

$$\ldots$$

Example 7.5. A TM Accepting $XX = \{xx \mid x \in \{a, b\}^*\}$

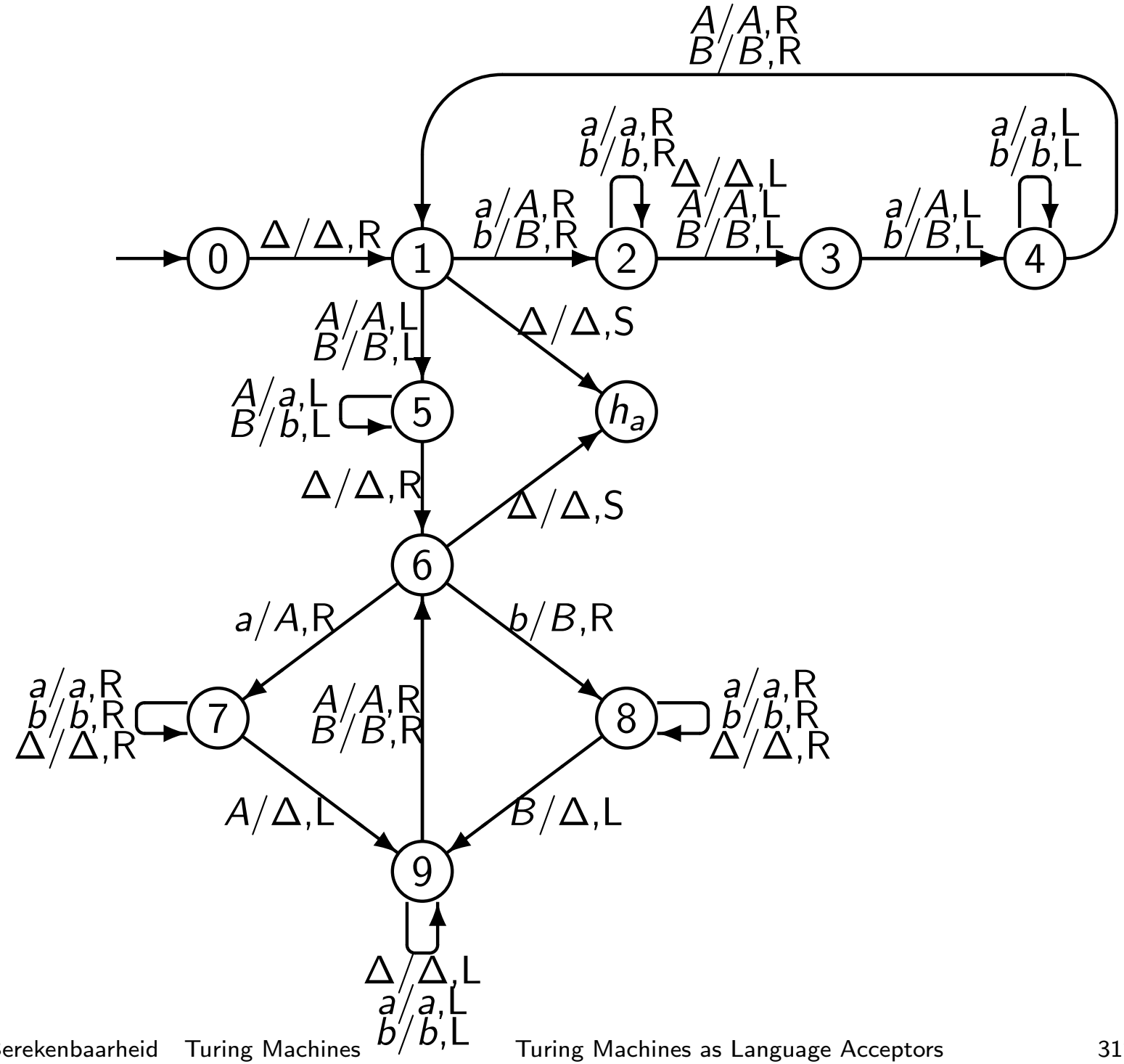
$$\underline{\Delta}\, a\, a\, b\, a\, a\, b$$
$$\Delta A\, a\, b\, a\, a\, b$$
$$\Delta A\, a\, b\, a\, a B$$
$$\Delta A A\, b\, a\, a B$$
$$\Delta A A\, b\, a A B$$
$$\Delta A A B\, a A B$$
$$\Delta A A B A A B$$
$$\ldots$$

Example 7.5. A TM Accepting $XX = \{xx \mid x \in \{a, b\}^*\}$

$$\underline{\Delta}\, a\, a\, b\, a\, a\, b$$
$$\Delta A\, a\, b\, a\, a\, b$$
$$\Delta A\, a\, b\, a\, a B$$
$$\Delta A A\, b\, a\, a B$$
$$\Delta A A\, b\, a A B$$
$$\Delta A A B\, a A B$$
$$\Delta A A B A A B$$
$$\Delta\, a\, a\, b A A B$$
$$\Delta A\, a\, b A A B$$
$$\ldots$$

Example 7.5. A TM Accepting $XX = \{xx \mid x \in \{a, b\}^*\}$

$$\underline{\Delta}\, a\, a\, b\, a\, a\, b$$
$$\Delta A\, a\, b\, a\, a\, b$$
$$\Delta A\, a\, b\, a\, a\, B$$
$$\Delta AA\, b\, a\, a\, B$$
$$\Delta AA\, b\, a\, A\, B$$
$$\Delta AAB\, a\, A\, B$$
$$\Delta AAB\, A\, A\, B$$
$$\Delta\, a\, a\, b\, A\, A\, B$$
$$\Delta A\, a\, b\, A\, A\, B$$
$$\Delta A\, a\, b\, \Delta\, A\, B$$
$$\Delta AA\, b\, \Delta\, A\, B$$
$$\Delta AA\, b\, \Delta\Delta\, B$$
$$\Delta AAB\, \Delta\Delta\, B$$
$$\Delta AAB\, \Delta\Delta\Delta$$


A/A,R
B/B,R
a/a,R
b/b,R
Δ/Δ,L
a/a,L
b/b,L
a/A,R
b/B,R
A/A,L
B/B,L
a/A,L
b/B,L
0
Δ/Δ,R
1
2
3
4
A/A,L
B/B,L
Δ/Δ,S
A/a,L
B/b,L
5
h_a
Δ/Δ,R
Δ/Δ,S
6
a/A,R
b/B,R
a/a,R
b/b,R
Δ/Δ,R
7
A/A,R
B/B,R
8
a/a,R
b/b,R
Δ/Δ,R
A/Δ,L
B/Δ,L
9
Δ/Δ,L
a/a,L
b/b,L

Example 7.5. A TM Accepting $XX = \{xx \mid x \in \{a, b\}^*\}$

$$
\begin{array}{l}
\underline{\Delta}\, a\, a\, b\, a\, a\, b \\
\Delta A\, a\, b\, a\, a\, b \\
\Delta A\, a\, b\, a\, a\, B \\
\Delta A A\, b\, a\, a\, B \\
\Delta A A\, b\, a\, A\, B \\
\Delta A A B\, a\, A\, B \\
\Delta A A B\, A\, A\, B \\
\Delta\, a\, a\, b\, A\, A\, B \\
\Delta A\, a\, b\, A\, A\, B \\
\Delta A\, a\, b\, \Delta\, A\, B \\
\Delta A A\, b\, \Delta\, A\, B \\
\Delta A A\, b\, \Delta \Delta\, B \\
\Delta A A B\, \Delta \Delta\, B \\
\Delta A A B\, \Delta \Delta \Delta
\end{array}
$$

What if input $w \notin XX$?

T decides on membership of XX