

Computability

voorjaar 2025

<https://liacs.leidenuniv.nl/~vlietrvan1/computability/>

college 5, 3 maart 2025

- 8. Recursively Enumerable Languages
 - 8.3. More General Grammars

Praktische Informatie

- dit vak dit jaar voor het laatst

Huiswerkopgave

voor 0.4pt

individueel

inleveren: uiterlijk morgenavond, 23.59 uur

Staking?

10 maart 2025...

A slide from lecture 4

Definition 8.1. Accepting a Language and Deciding a Language

A Turing machine T with input alphabet Σ accepts a language $L \subseteq \Sigma^*$,
if $L(T) = L$.

T decides L ,

if T computes the characteristic function $\chi_L : \Sigma^* \rightarrow \{0, 1\}$

A language L is *recursively enumerable*,
if there is a TM that accepts L ,

and L is *recursive*,
if there is a TM that decides L .

A slide from lecture 4

Theorem 8.2.

Every recursive language is recursively enumerable.

Proof...

A slide from lecture 4

Theorem 8.4. If L_1 and L_2 are both recursively enumerable languages over Σ , then $L_1 \cup L_2$ and $L_1 \cap L_2$ are also recursively enumerable.

Proof...

Theorem 8.5. If L_1 and L_2 are both recursive languages over Σ , then $L_1 \cup L_2$ and $L_1 \cap L_2$ are also recursive.

Proof. Exercise 8.1.

Exercise 8.1.

Show that if L_1 and L_2 are recursive languages, then $L_1 \cup L_2$ and $L_1 \cap L_2$ are also.

Theorem 8.6. If L is a recursive language over Σ , then its complement L' is also recursive.

Proof...

Theorem 8.7. If L is a recursively enumerable language, and its complement L' is also recursively enumerable, then L is recursive (and therefore, by Theorem 8.6, L' is recursive).

Proof...

Corollary.

Let L be a recursively enumerable language.

Then

L' is recursively enumerable,

if and only

if L is recursive.

Corollary.

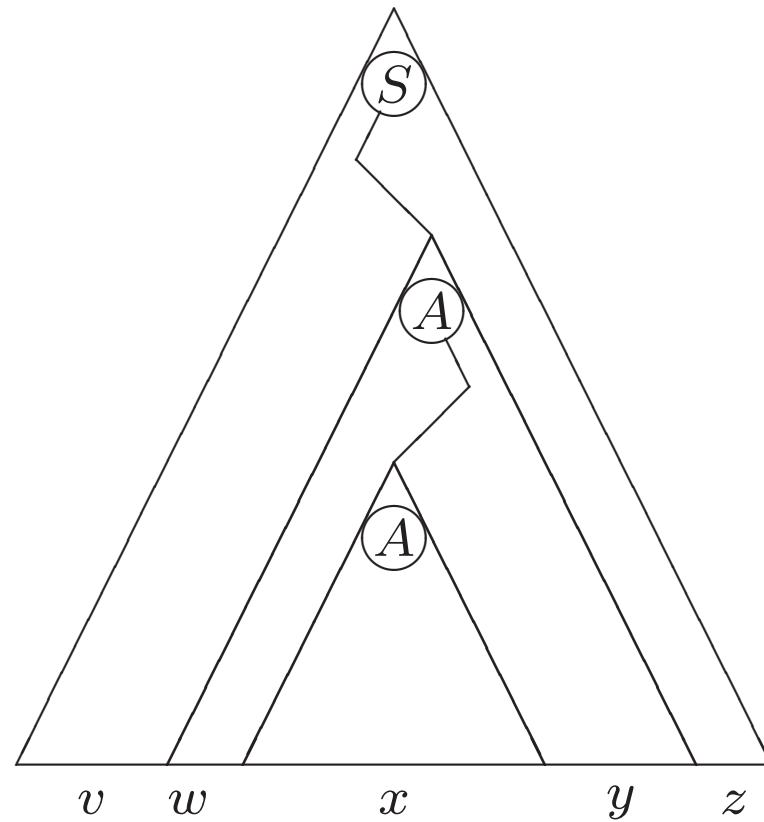
There exist languages that are not recursively enumerable,
if and only if
there exist languages that are not recursive.

8.3. More General Grammars

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	

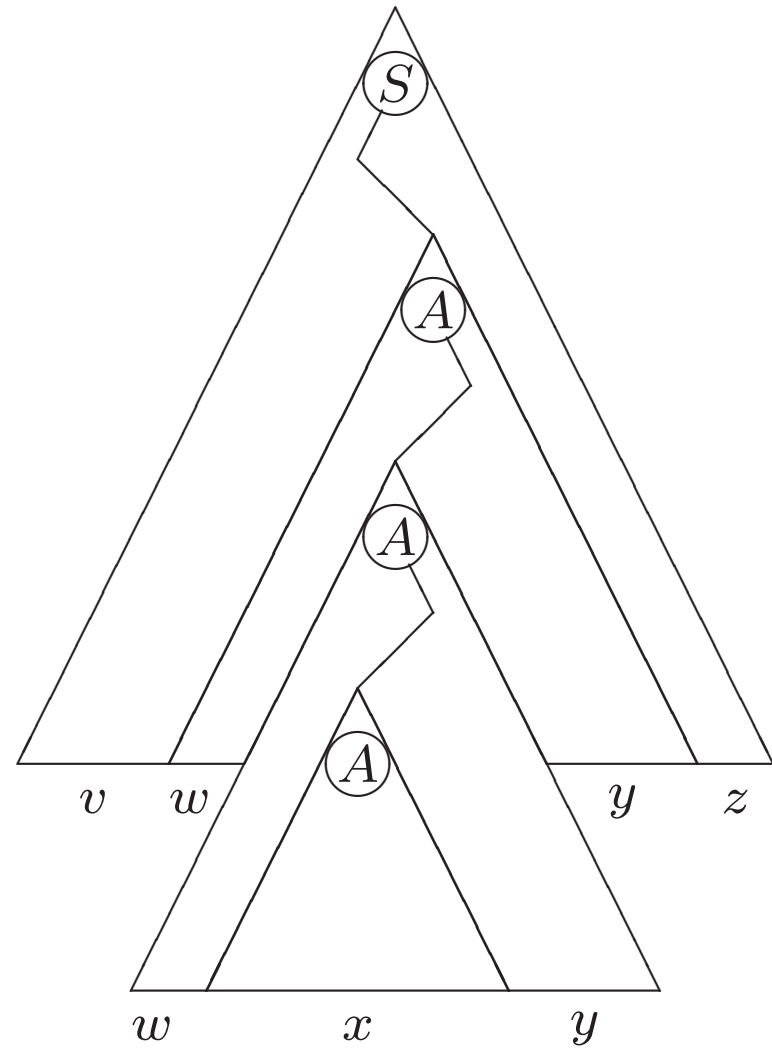
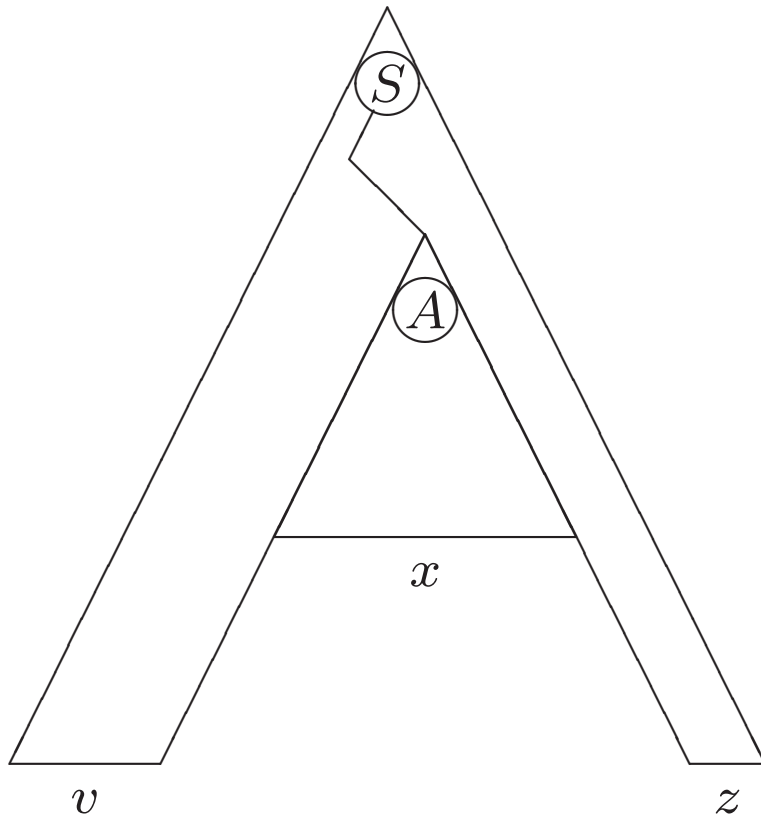
A slide from lecture 1

FI2: Pumping Lemma for CFLs



A slide from lecture 1

FI2: Pumping Lemma for CFLs



Definition 8.10. Unrestricted grammars

An unrestricted grammar is a 4-tuple $G = (V, \Sigma, S, P)$, where V and Σ are disjoint **finite** sets of variables and terminals, respectively, S is an element of V called the start symbol, and P is a **finite** set of productions of the form

$$\alpha \rightarrow \beta$$

where $\alpha, \beta \in (V \cup \Sigma)^*$ and α contains at least one variable.

Notation as for CFGs:

$$\alpha \Rightarrow_G^* \beta$$

$$L(G) = \{x \in \Sigma^* \mid S \Rightarrow_G^* x\}$$

but when $S \Rightarrow^* xAy \Rightarrow^* z \dots$

Notation \rightarrow vs \Rightarrow vs $\vdash \dots$

Example 8.12. A Grammar Generating $\{a^n b^n c^n \mid n \geq 1\}$

Example 8.12. A Grammar Generating $\{a^n b^n c^n \mid n \geq 1\}$

$$S \rightarrow SABC \mid LABC$$

$$BA \rightarrow AB \quad CB \rightarrow BC \quad CA \rightarrow AC$$

$$LA \rightarrow a \quad aA \rightarrow aa \quad aB \rightarrow ab \quad bB \rightarrow bb \quad bC \rightarrow bc \quad cC \rightarrow cc$$

Example 8.11. A Grammar Generating $\{a^{2^k} \mid k \in \mathbb{N}\}$

$$\{a, a^2, a^4, a^8, a^{16}, \dots\} = \{a, aa, aaaa, aaaaaaaaa, aaaaaaaaaaaaaaaaaaaaaa, \dots\}$$

Example 8.11. A Grammar Generating $\{a^{2^k} \mid k \in \mathbb{N}\}$

$$\{a, a^2, a^4, a^8, a^{16}, \dots\} = \{a, aa, aaaa, aaaaaaaaaa, aaaaaaaaaaaaaaaaaa, \dots\}$$

$$S \rightarrow LaR$$

$$L \rightarrow LD \quad Da \rightarrow aaD \quad DR \rightarrow R$$

$$L \rightarrow \Lambda \quad R \rightarrow \Lambda$$

Example.

An Unrestricted Grammar Generating $XX = \{xx \mid x \in \{a, b\}^*\}$

First a CFG for $Pal = \{x \in \{a, b\}^* \mid x = x^r\}$:

$$S \rightarrow aSa \mid bSb \mid a \mid b \mid \Lambda$$

Example.

An Unrestricted Grammar Generating $XX = \{xx \mid x \in \{a,b\}^*\}$

$$S \rightarrow aAS \mid bBS \mid M$$

$$Aa \rightarrow aA \quad Ab \rightarrow bA \quad Ba \rightarrow aB \quad Bb \rightarrow bB$$

$$AM \rightarrow Ma \quad BM \rightarrow Mb \quad M \rightarrow \Lambda$$

Theorem 8.13.

For every unrestricted grammar G , there is a Turing machine T with $L(T) = L(G)$.

Proof.

1. Move past input (= NB)
2. Simulate derivation in G on the tape of a Turing machine
3. (PB , followed by) Equal

Theorem 8.13.

For every unrestricted grammar G , there is a Turing machine T with $L(T) = L(G)$.

Proof.

1. Move past input
2. Simulate derivation in G on the tape of a Turing machine:
 - Write S on tape
 - Repeat
 - a. Select production $\alpha \rightarrow \beta$
 - b. Select occurrence of α (if there is one)
 - c. Replace occurrence of α by β
 - until b. fails (caused by ...)
3. Equal

A slide from lecture 4

Theorem 7.31.

For every nondeterministic TM $T = (Q, \Sigma, \Gamma, q_0, \delta)$,
there is an ordinary (deterministic) TM $T_1 = (Q_1, \Sigma, \Gamma_1, q_1, \delta_1)$
with $L(T_1) = L(T)$.

Moreover, if there is no input on which T can loop forever,
then T_1 also halts on every input.

The proof of this result does not have to be known for the exam.

Example.

(The second part of) the construction from Theorem 8.13 to obtain a TM simulating a derivation in the unrestricted grammar with productions

$$S \rightarrow aBS \mid \Lambda \quad aB \rightarrow Ba \quad Ba \rightarrow aB \quad B \rightarrow b$$

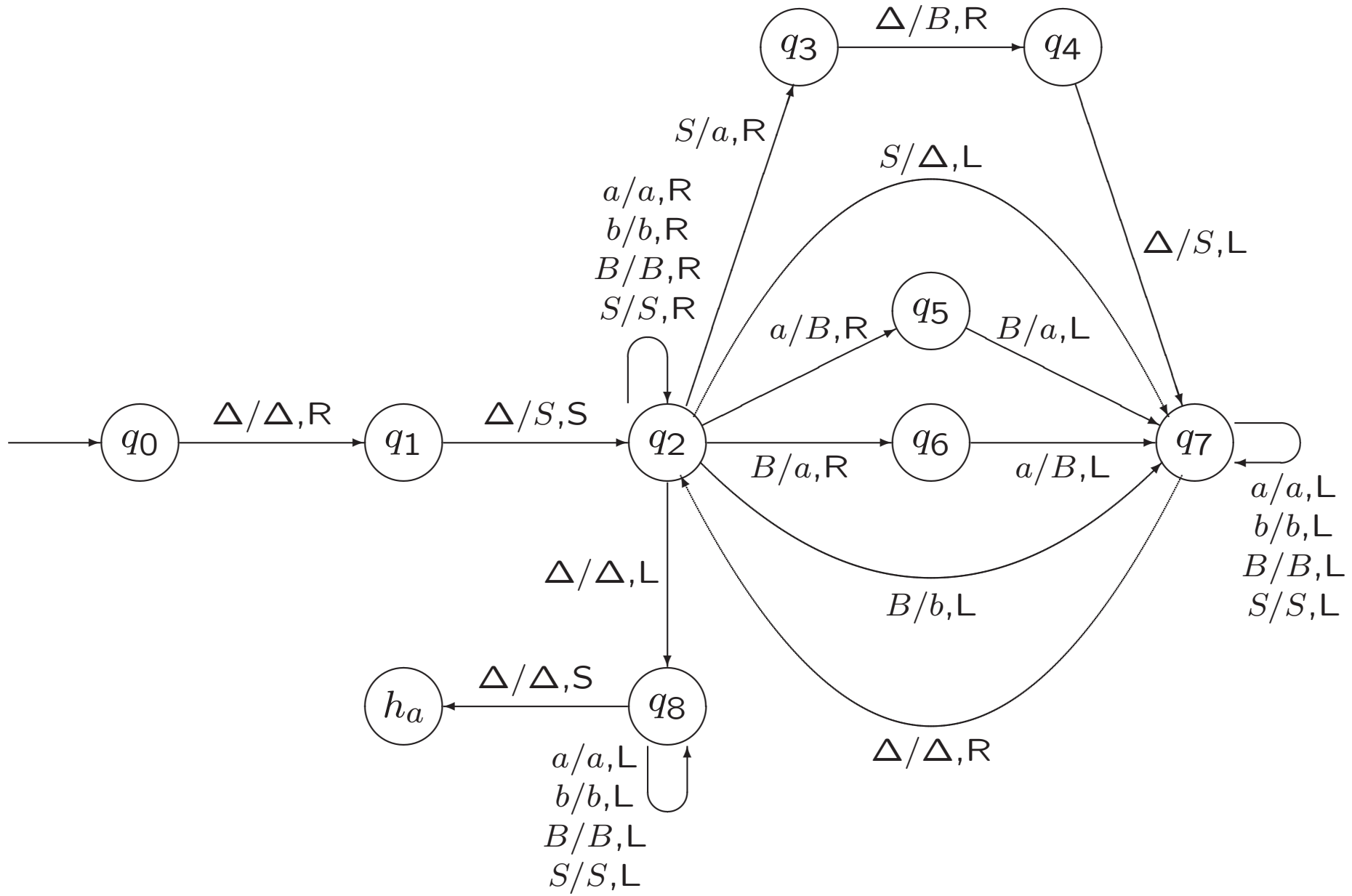
See next slide

N.B.:

In next slide, we simulate application of arbitrary production by

- first moving to arbitrary position in current string (at q_2)
- only then selecting (and applying) a possible production

This implementation of the construction must be known for the exam



Theorem 8.14.

For every Turing machine T with input alphabet Σ , there is an unrestricted grammar G generating the language $L(T) \subseteq \Sigma^*$.

Proof.

1. Generate (every possible) input string for T .
2. Simulate computation of T for this input string as derivation in grammar.
3. If T reaches accept state, reconstruct original input string.

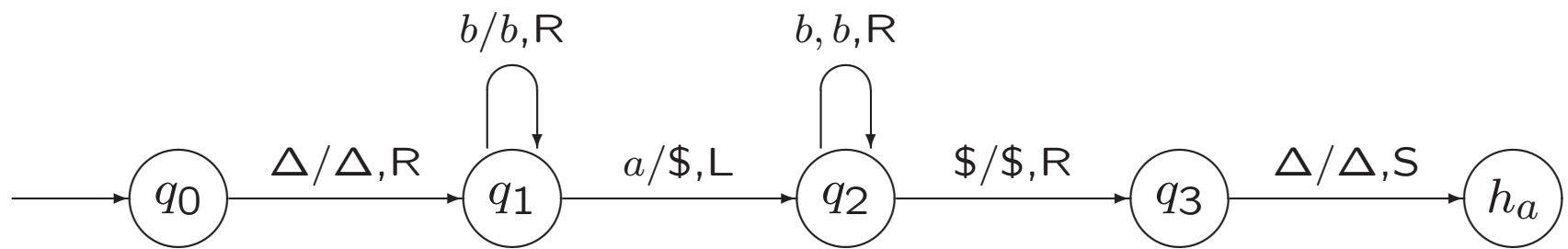
(Part of) a slide from lecture 2

Notation:

description of tape contents: $x\underline{\sigma}y$ or $x\underline{y}$

configuration $xqy = xqy\Delta = xqy\Delta\Delta$

initial configuration corresponding to input x : $q_0\Delta x$



Computation for $x = ba$.

Theorem 8.14.

For every Turing machine T with input alphabet Σ , there is an unrestricted grammar G generating the language $L(T) \subseteq \Sigma^*$.

Proof.

1. Generate (every possible) input string for T (two copies), with additional $(\Delta\Delta)$'s and state.
2. Simulate computation of T for this input string as derivation in grammar (on second copy).
3. If T reaches accept state, reconstruct original input string.