Fundamentele Informatica 3

najaar 2016

`http://www.liacs.leidenuniv.nl/~vlietrvan1/fi3/`

Rudy van Vliet
kamer 124 Snellius, tel. 071-527 5777
rvvliet(at)liacs(dot)nl

college 6, 11 oktober 2016

8. Recursively Enumerable Languages
8.3. More General Grammars
8.4. Context-Sensitive Languages and The Chomsky Hierarchy

Huiswerkopgave 1

Voor 0.4pt

Inleveren: donderdag 20 oktober 2016, 13:45 uur

A slide from lecture 5

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 $\chi_L : \Sigma^* \to \{0, 1\}$

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

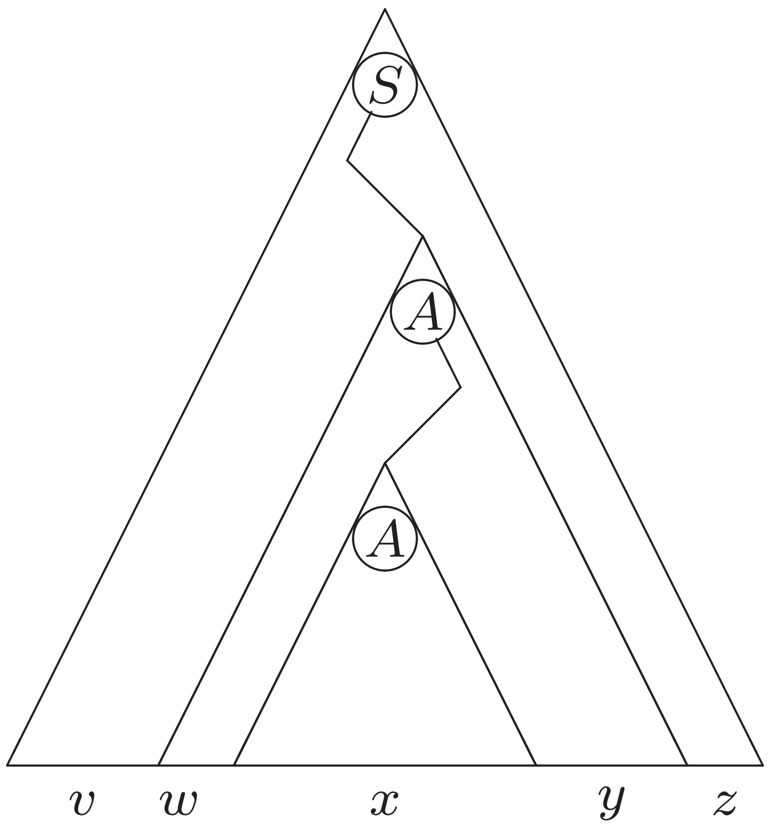
and L is *recursive*,
if there is a TM that decides L.

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	

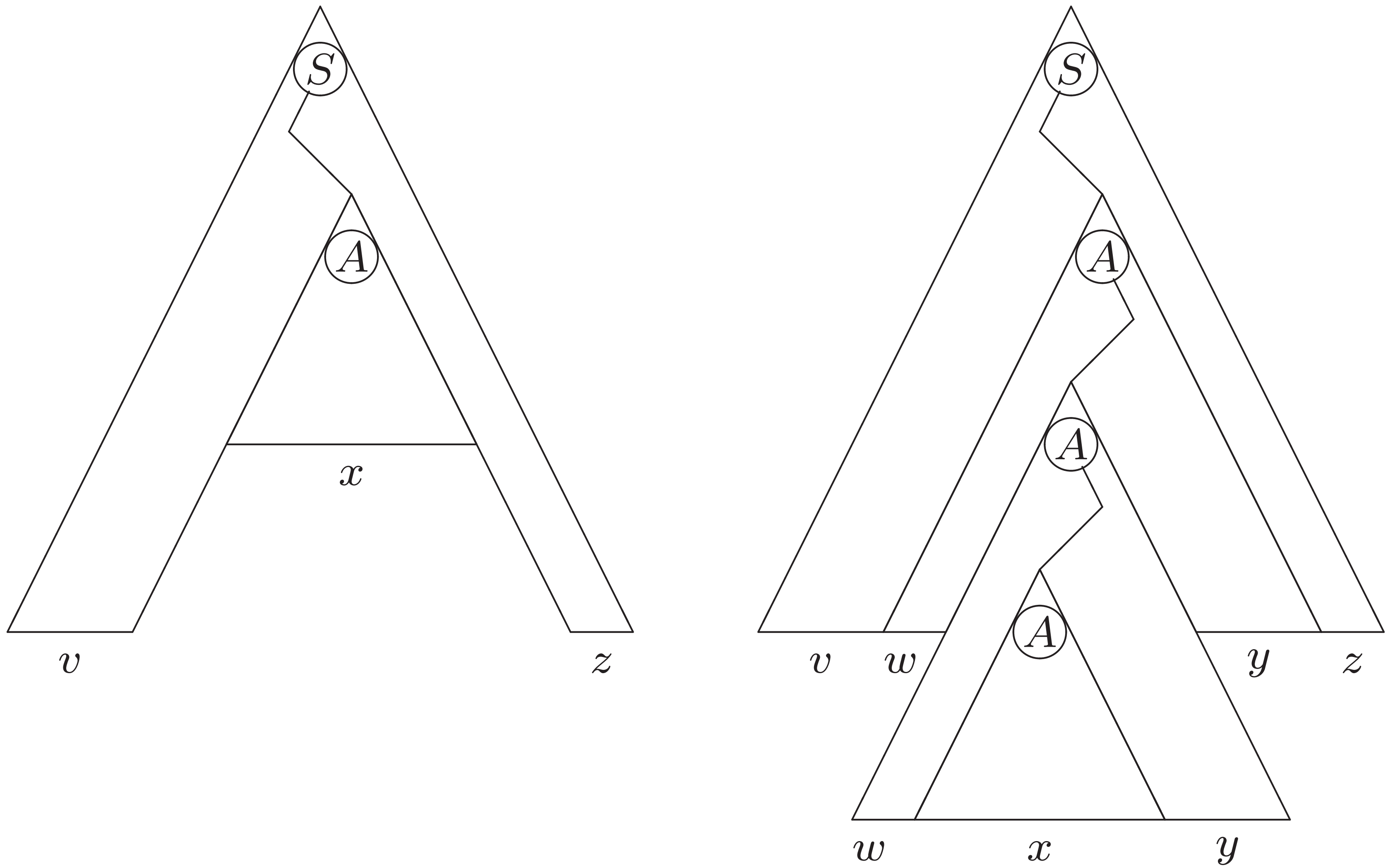
A slide from lecture 1

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 sets of variables and terminals, respectively, S is an element of V called the start symbol, and P is a set of productions of the form

$$\alpha \to \beta$$

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. . .

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

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

$$S \to SABC \mid LABC$$

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

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

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

$$\{a, a^2, a^4, a^8, a^{16}, \ldots\} = \{a, aa, aaaa, aaaaaaaa, aaaaaaaaaaaaaaaa, \ldots\}$$

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

$$\{a, a^2, a^4, a^8, a^{16}, \ldots\} = \{a, aa, aaaa, aaaaaaaa, aaaaaaaaaaaaaaaa, \ldots\}$$

$$S \to LaR$$

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

$$L \to \Lambda \quad R \to \Lambda$$

Example.

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

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

$$S \to aSa \quad | \quad bSb \quad | \quad a \quad | \quad b \quad | \quad \Lambda$$

Example.

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

$$S \to aAS \mid bBS \mid M$$

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

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

Theorem 8.13.
For every unrestricted grammar G, there is a Turing machine T with $L(T) = L(G)$.

Proof.
1. Move past input
2. Simulate derivation in G on the tape of a Turing machine
3. Equal

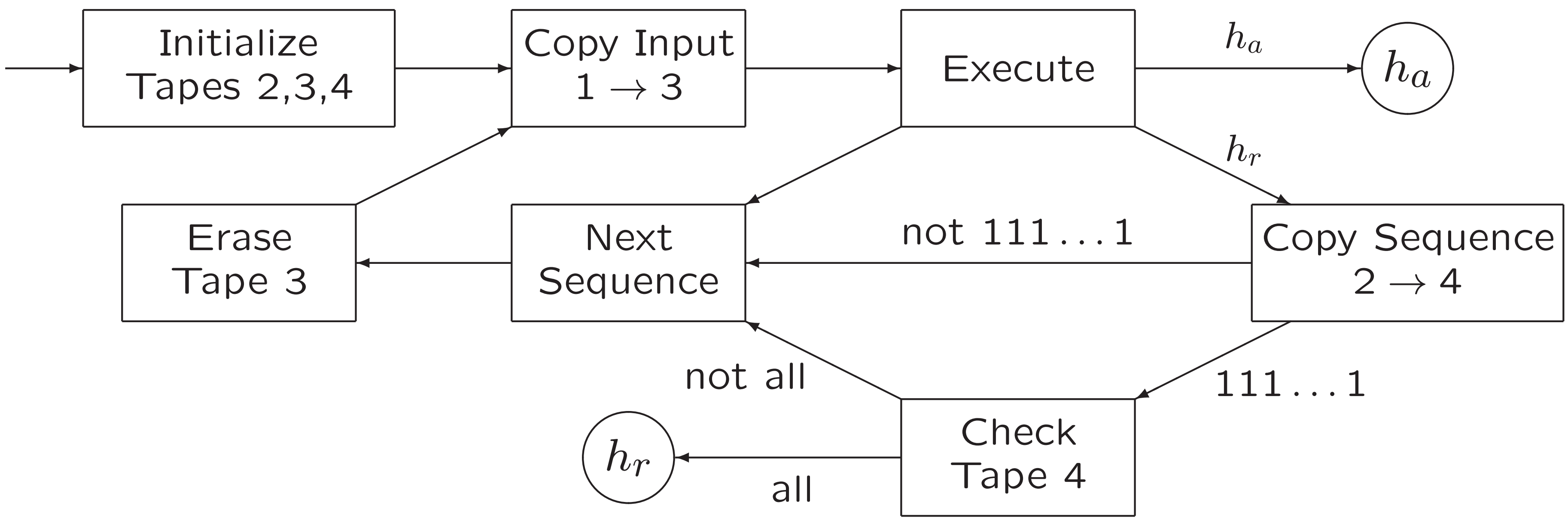
Theorem 8.13.
For every unrestricted grammar G, there is a Turing machine T with $L(T) = L(G)$.

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 $\alpha \to \beta$
   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 $L(T_1) = L(T)$.

Proof. . .



Example.

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

$$S \to aBS \mid \Lambda \qquad aB \to Ba \qquad Ba \to aB \qquad B \to b$$

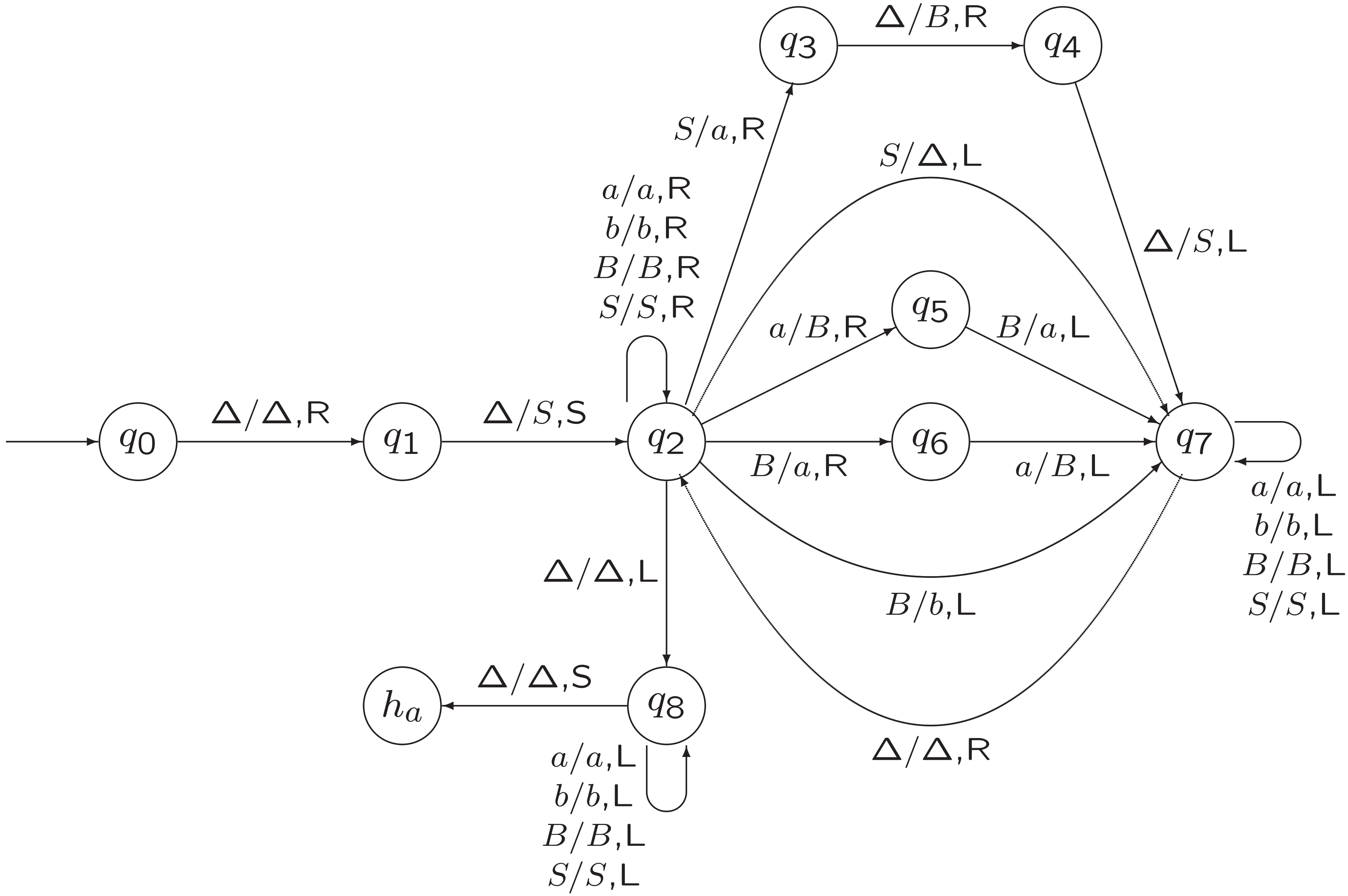
See next slide

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


q_0
Δ/Δ,R
q_1
Δ/S,S
q_2
a/a,R
b/b,R
B/B,R
S/S,R
S/a,R
q_3
Δ/B,R
q_4
Δ/S,L
S/Δ,L
a/B,R
q_5
B/a,L
B/a,R
q_6
a/B,L
q_7
a/a,L
b/b,L
B/B,L
S/S,L
B/b,L
Δ/Δ,R
Δ/Δ,L
q_8
Δ/Δ,S
h_a
a/a,L
b/b,L
B/B,L
S/S,L

8.4. Context-Sensitive Languages and the Chomsky Hierarchy

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	

Definition 8.16. Context-Sensitive Grammars
A *context-sensitive grammar* (CSG) is an unrestricted grammar in which no production is length-decreasing.
In other words, every production is of the form $\alpha \to \beta$, where $|\beta| \geq |\alpha|$.

A language is a context-sensitive language (CSL) if it can be generated by a context-sensitive grammar.

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

$$S \to SABC \mid LABC$$

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

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

Not context-sensitive.

Example 8.17. A CSG Generating $L = \{a^n b^n c^n \mid n \geq 1\}$

$$S \to SABC \mid \mathcal{A}BC$$

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

$$\mathcal{A} \to a \quad aA \to aa \quad aB \to ab \quad bB \to bb \quad bC \to bc \quad cC \to cc$$

Example.

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

$$S \to aAS \mid bBS \mid M$$

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

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

Not context-sensitive.

Exercise 8.24.

Find a context-sensitive grammar generating the language

$$XX - \{\Lambda\} = \{xx \mid \ x \in \{a,b\}^* \text{ and } x \neq \Lambda\}$$

Programming languages

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
3. Equal

Definition 8.18. Linear-Bounded Automata
A *linear-bounded automaton* (LBA) is a 5-tuple $M = (Q, \Sigma, \Gamma, q_0, \delta)$ that is identical to a nondeterministic Turing machine, with the following exception.

There are two extra tape symbols [and], assumed not to be elements of the tape alphabet Γ.
The initial configuration of M corresponding to input x is $q_0[x]$, with the symbol [in the leftmost square and the symbol] in the first square to the right of x.
During its computation, M is not permitted to replace either of these brackets or to move its tape head to the left of the [or to the right of the].

Theorem 8.19.
If $L \subseteq \Sigma^*$ is a context-sensitive language, then there is a linear-bounded automaton that accepts L.

Proof. . .

Theorem 8.19.
If $L \subseteq \Sigma^*$ is a context-sensitive language, then there is a linear-bounded automaton that accepts L.

Proof. Much like the proof of Theorem 8.13, except
- two tape tracks instead of move past input
- reject also if we (want to) write on]

Theorem 8.19.
If $L \subseteq \Sigma^*$ is a context-sensitive language, then there is a linear-bounded automaton that accepts L.

Proof.

1. Create second tape track
2. Simulate derivation in G on track 2
3. Equal

Theorem 8.19.
If $L \subseteq \Sigma^*$ is a context-sensitive language, then there is a linear-bounded automaton that accepts L.

Proof.

1. Create second tape track
2. Simulate derivation in G on track 2:

   Write S on track 2

   Repeat

   a. Select production $\alpha \to \beta$

   b. Select occurrence of α on track 2 (if there is one)

   c. Try to replace occurrence of α by β

   until b. fails (caused by . . .)

   or c. fails (caused by . . .); then reject
3. Equal

Theorem 8.19.
If $L \subseteq \Sigma^*$ is a context-sensitive language, then there is a linear-bounded automaton that accepts L.

Proof. Much like the proof of Theorem 8.13, except
- two tape tracks instead of move past input
- reject also if we (want to) write on]

Alternative proof.
Simulate derivation of string x from S in reverse order
c.f., bottom-up parsing
Then one tape track is sufficient

Just an observation

Every context-sensitive language is recursively enumerable

A slide from lecture 5

Theorem 8.2.
Every recursive language is recursively enumerable.

Proof. . .

Theorem 8.22. Every context-sensitive language L is recursive.

Proof. . .

Theorem 8.22. Every context-sensitive language L is recursive.

Proof.

Let CSG G generate L
Let LBA M accept strings generated by G (as in Theorem 8.19)

Simulate M by NTM T, which
- inserts markers [and]
- also has two tape tracks
- maintains list of (different) strings generated so far

a. Select production $\alpha \to \beta$
b. Select occurrence of α on track 2 (if there is one)
c. Try to replace occurrence of α by β
d. Compare new string to strings to the right of]
until b. fails (caused by ...); then Equal
or c. fails (caused by ...); then reject
or d. finds match; then reject

A slide from lecture 5

Corollary.

If L is accepted by a nondeterministic TM T, and if there is no input string on which T can possibly loop forever,
then L is recursive.

Proof. . .