Fundamentele Informatica 1 (I&E)

najaar 2015

http://www.liacs.leidenuniv.nl/~vlietrvan1/fi1ie/

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

college 13, dinsdag 8 december 2015

7.3. Turing Machines That Compute Partial Functions

8.3. More General Grammars

A slide from lecture 1:

Computer receives input, performs ‘computation’, gives output

- Given instance of Nim. Who wins?

- Given sequence of numbers. Sort

- Given edge-weighted graph.
  Give shortest route from A to B

A slide from lecture 12:

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

7.3. Turing Machines That Compute Partial Functions

Example 7.10. The Reverse of a String

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

Simple version of:

Definition 7.9. A Turing Machine Computing a Function

Let $T = (Q, \Sigma, \Gamma, q_0, \delta)$ be a Turing machine, and f a partial function on Σ^* with values in Γ^*. We say that T computes f if for every x in the domain of f,

$$q_0 \Delta x \vdash_T^* h_a \Delta f(x)$$

and no other input string is accepted by T.

Definition 7.9. A Turing Machine Computing a Function

Let $T = (Q, \Sigma, \Gamma, q_0, \delta)$ be a Turing machine, k a natural number, and f a partial function on $(\Sigma^*)^k$ with values in Γ^*. We say that T computes f if for every $(x_1, x_2, \ldots, x_k)$ in the domain of f,

$$q_0 \Delta x_1 \Delta x_2 \Delta \ldots \Delta x_k \vdash_T^* h_a \Delta f(x_1, x_2, \ldots, x_k)$$

and no other input that is a k-tuple of strings is accepted by T.

A partial function $f : (\Sigma^*)^k \to \Gamma^*$ is Turing-computable, or simply computable, if there is a TM that computes f.

Functions on natural numbers. . .

Example 7.12. The Quotient and Remainder Mod 2

Exercise.

Draw a TM that computes the function

$$f(x, y) = x + y$$

where x, y are integers ≥ 0.

Assume that the TM uses unary notation, both for its input and for its output.

Exercise.

Draw a TM that computes the function $f(x, y) = x \bmod y$

Hint: implement the following algorithm:

```
while (x >= y)
  x = x - y;
```

Een Intermezzo

http://www.youtube.com/watch?v=E3keLeMwfHY

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

Definition 8.1. Accepting a Language (. . .)

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

(. . .)

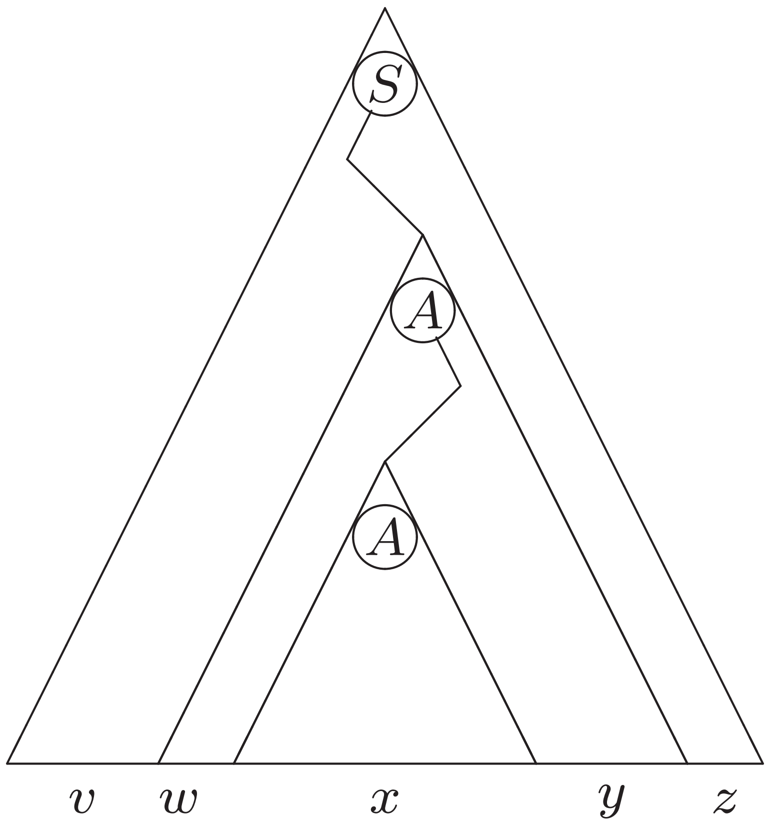
A language L is *recursively enumerable*,
if there is a TM that accepts L,

(. . .)

8.3. More General Grammars

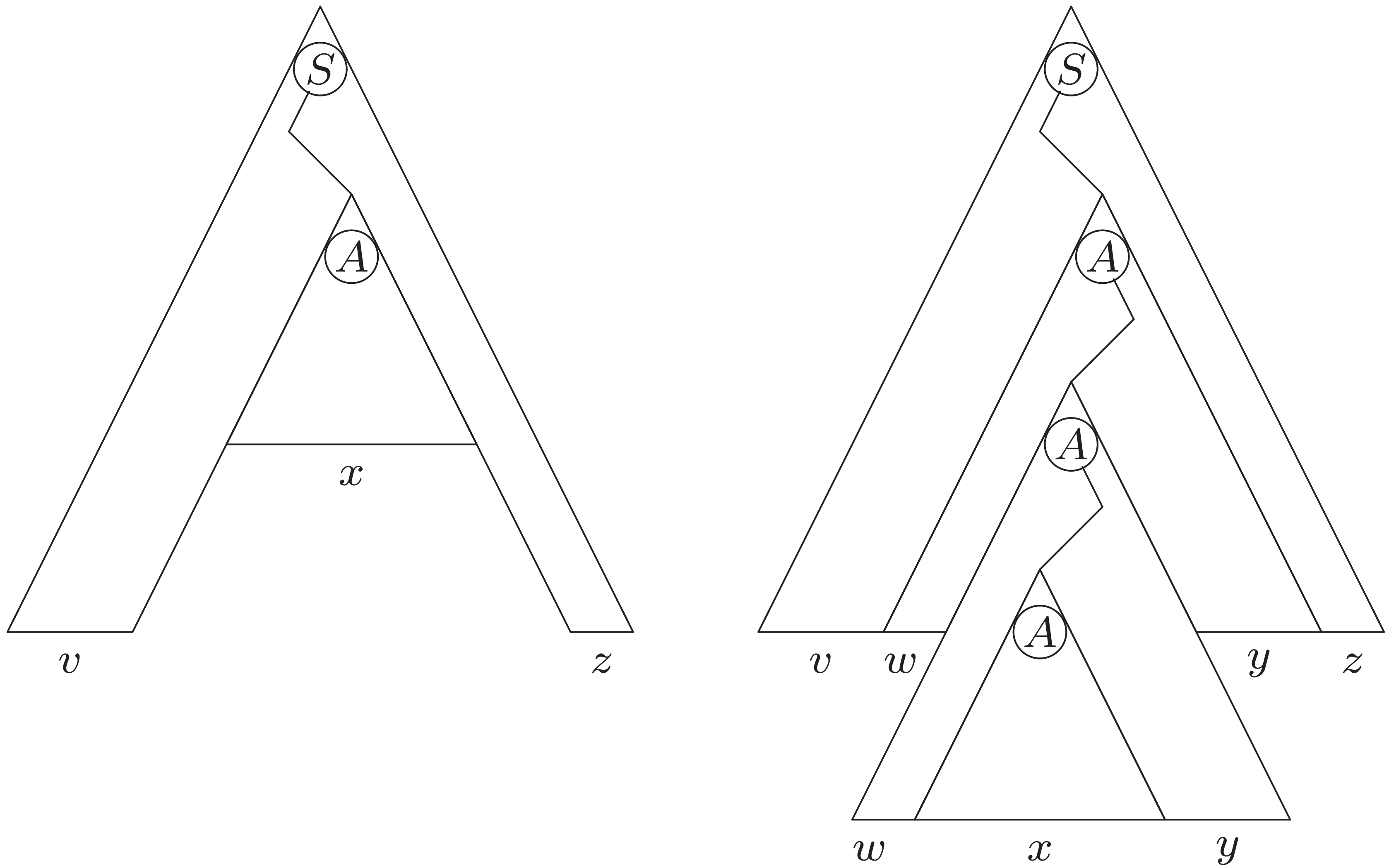
A slide from lecture 11:

Pumping Lemma for CFLs



A slide from lecture 11:

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 $\{a^n b^n c^n \mid n \geq 1\}$

$$S \to SABC \mid LABC$$

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

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

Correct and incorrect derivation for $aabbcc\ldots$

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

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

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

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

$$S \to LaR \quad L \to LD \quad Da \to aaD \quad DR \to R \quad L \to \Lambda \quad R \to \Lambda$$

Correct and incorrect derivation for $aaaa\ldots$

Example.

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

Example.

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

$$S \to LM \qquad M \to AMa \mid BMb \mid \Lambda$$

$$LA \to LA_1 \qquad LB \to LB_1$$

$$A_1A \to AA_1 \qquad A_1B \to BA_1 \qquad A_1a \to aa \qquad A_1b \to ab$$

$$B_1A \to AB_1 \qquad B_1B \to BB_1 \qquad B_1a \to ba \qquad B_1b \to bb$$

$$L \to \Lambda$$

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

Theorem 8.14.
For every Turing machine T with input alphabet Σ, there is an unrestricted grammar G generating the language $L(T) \subseteq \Sigma^*$.

In other words: the languages generated by unrestricted grammars are exactly the recursively enumerable languages.

The proofs of these results do not have to be known for the exam.

En verder. . .

Vrijdag 11 december 2015, 13:45:
Inleveren huiswerkopgave

Donderdag 7 januari 2016, 10:00–13:00:
Tentamen (in Leiden)

Vragenuur?