Fundamentele Informatica 3

voorjaar 2014

http://www.liacs.nl/home/rvvliet/fi3/

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

college 7, 24 maart 2014

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

A slide from lecture 6

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.

A slide from lecture 6

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

A slide from lecture 6

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.

A slide from lecture 6

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].

A slide from lecture 6

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

Proof. . .

A slide from lecture 6

8.4. Context-Sensitive Languages and the Chomsky Hierarchy

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

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.

A slide from lecture 3

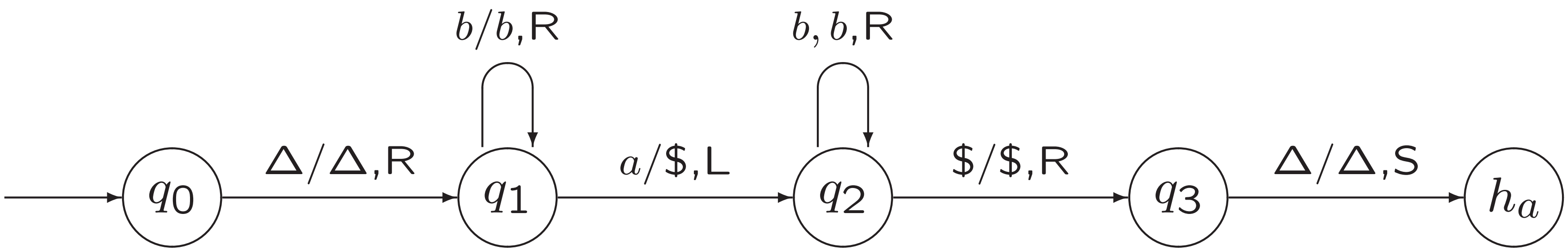
Notation:

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: $q_0\Delta x$

In the third edition of the book, a configuration is denoted as $(q, x\underline{y})$ or $(q, x\underline{\sigma}y)$ instead of xqy or $xq\sigma y$.
This old notation is also allowed for Fundamentele Informatica 3.


b/b,R
b,b,R
Δ/Δ,R
$a/\$$,L
$\$/\$$,R
Δ/Δ,S
q_0
q_1
q_2
q_3
h_a

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.

Ad 2. Move $\delta(p, a) = (q, b, R)$ of T
yields production $p(\sigma_1 a) \to (\sigma_1 b)q$

Ad 3. Propagate h_a all over the string
$h_a(\sigma_1\sigma_2) \to \sigma_1$, for $\sigma_1 \in \Sigma$
$h_a(\Delta\sigma_2) \to \Lambda$

Theorem 8.20. If $L \subseteq \Sigma^*$ is accepted by a linear-bounded automaton $M = (Q, \Sigma, \Gamma, q_0, \delta)$, then there is a context-sensitive grammar G generating $L - \{\Lambda\}$.

Proof...

Theorem 8.20. If $L \subseteq \Sigma^*$ is accepted by a linear-bounded automaton $M = (Q, \Sigma, \Gamma, q_0, \delta)$, then there is a context-sensitive grammar G generating $L - \{\Lambda\}$.

Proof. Much like proof of Theorem 8.14, except

- consider $h_a(\sigma_1\sigma_2)$ as a single symbol
- no additional $(\Delta\Delta)$'s needed
- incorporate [and] in leftmost/rightmost symbols of string

A slide from lecture 6

8.4. Context-Sensitive Languages and the Chomsky Hierarchy

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

Chomsky hierarchy

3	reg. languages	reg. grammar	FA	reg. expression
2	cf. languages	cf. grammar	PDA	
1	cs. languages	cs. grammar	LBA	
0	re. languages	unrestr. grammar	TM	

What about recursive languages?

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

Proof. . .

A slide from lecture 6

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 8.22. 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. . .

Chomsky hierarchy

3	reg. languages	reg. grammar	FA	reg. expression
2	cf. languages	cf. grammar	PDA	
1	cs. languages	cs. grammar	LBA	
0	re. languages	unrestr. grammar	TM	

$$\mathcal{S}_3 \subseteq \mathcal{S}_2 \subseteq \mathcal{S}_1 \subseteq \mathcal{R} \subseteq \mathcal{S}_0$$

(modulo Λ)