Fundamentele Informatica 3

voorjaar 2012

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

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

college 12, maandag 23 april 2012

8. Recursively Enumerable Languages
8.4. Context-Sensitive Languages and the Chomsky Hierarchy
8.5. Not Every Language is Recursively Enumerable

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.

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.

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 8.14.
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 $\sigma_1 \in \Sigma$
$h_a(\Delta\sigma_2) \to \Lambda$

8.4. Context-Sensitive Languages and the Chomsky Hierarchy

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

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

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

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

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

Definition 8.10. 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. Much like the proof of Theorem 8.13. . .

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 8.19.
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 8.20. 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. . .

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. 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

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 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)

8.5. Not Every Language is Recursively Enumerable

From Fundamentele Informatica 1:

Definition 8.23.
A Set A of the Same Size as B or Larger Than B

Two sets A and B, either finite or infinite, are the same size if there is a bijection $f : A \to B$.

A is larger than B if some subset of A is the same size as B but A itself is not.

From Fundamentele Informatica 1:

Definition 8.24.
Countably Infinite and Countable Sets

A set A is *countably infinite* (the same size as $\mathbb{N}$) if there is a bijection $f : \mathbb{N} \to A$, or a list $a_0, a_1, \ldots$ of elements of A such that every element of A appears exactly once in the list.

A is *countable* if A is either finite or countably infinite.

Theorem 8.25.
Every infinite set has a countably infinite subset,
and every subset of a countable set is countable.

Proof...

(proof of second claim is Exercise 8.35)

Example 8.26. The Set $\mathbb{N} \times \mathbb{N}$ is Countable

$$\mathbb{N} \times \mathbb{N} = \{(i, j) \mid i, j \in \mathbb{N}\}$$

although $\mathbb{N} \times \mathbb{N}$ looks much bigger than $\mathbb{N}$

Example 8.28.
A Countable Union of Countable Sets Is Countable

$$S = \bigcup_{i=0}^{\infty} S_i$$

Same construction as in Example 8.26, but. . .

Example 8.29. Languages Are Countable Sets

$$L \subseteq \Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$$

Two ways to list Σ^*

Example 8.30. The Set of Turing Machines Is Countable

Let $\mathcal{T}$ be set of Turing machines
There is injective function $e : \mathcal{T} \to \{0,1\}^*$
(e is encoding function)

Hence, set of recursively enumerable languages is countable

Example 8.31. The Set $2^{\mathbb{N}}$ Is Uncountable

Hence, because $\mathbb{N}$ and $\{0,1\}^*$ are the same size,
there are uncountably many languages over $\{0,1\}$

Example 8.31. The Set $2^{\mathbb{N}}$ Is Uncountable (continued)

No list of subsets of $\mathbb{N}$ is complete,
i.e., every list $A_0, A_1, A_2, \ldots$ of subsets of $\mathbb{N}$ leaves out at least one.

Take

$$A = \{i \in \mathbb{N} \mid i \notin A_i\}$$

Example 8.31. The Set $2^{\mathbb{N}}$ Is Uncountable (continued)

$$A = \{i \in \mathbb{N} \mid i \notin A_i\}$$

$$\begin{aligned}
A_0 &= \{0,2,5,9\} \\
A_1 &= \{1,2,3,8,12,\ldots\} \\
A_2 &= \{0,3,6\} \\
A_3 &= \emptyset \\
A_4 &= \{4\} \\
A_5 &= \{2,3,5,7,11,\ldots\} \\
A_6 &= \{8,16,24,\ldots\} \\
A_7 &= \mathbb{N} \\
A_8 &= \{1,3,5,7,9,\ldots\} \\
A_9 &= \{n \in \mathbb{N} \mid n > 12\} \\
&\ldots
\end{aligned}$$

Theorem 8.32. Not all languages are recursively enumerable. In fact, the set of languages over $\{0,1\}$ that are not recursively enumerable is uncountable.

Proof. . .

(including Exercise 8.38)