Linear bounded automata and context-sensitive grammars, Intro to computability theory

NTIN071 Automata and Grammars

Jakub Bulín (KTIML MFF UK)

Spring 2024

** Adapted from the Czech-lecture slides by Marta Vomlelová with gratitude. The translation, some modifications, and all errors are mine.*

Recap of Lecture 10

- Turing machine: two-way infinite tape, read, write, move head
- Accept iff in a final state; configurations
- TMs with output, computing a function
- Recursively enumerable vs. recursive languages (always halt).
- Construction tricks:
  - storage in state
  - multiple tracks (on a single tape)
- Variants of TMs:
  - multi-tape (independent heads),
  - nondeterministic (accept iff some choices lead to final state)

3.3 Turing Machines and grammars

Chomsky hierarchy: type 0

Turing machines	↔	Formal grammars
Linear bounded automata	↔	Context-sensitive grammars
Pushdown automata	↔	Context-free grammars
Finite automata	↔	Regular grammars

Theorem

A language is recursively enumerable, if and only if it is generated by a Type 0 grammar.

Turing machine to grammar

- First generate the relevant portion of the tape and a copy of the input word (nonterminal $\underline{X}$ for each $x \in \Gamma$, in reverse)
- Why? TM can rewrite w, G must generate it, cannot modify
- We have $wB^n\underline{W}^RQ_0B^m$, where B^n, B^m is sufficient free space
- Then simulate moves (essentially reverse configs+free space)
- In a final state erase the simulated tape, keep only w

$G = (\{S, C, D, E\} \cup \{\underline{X}\}_{x\in\Gamma} \cup \{Q_i\}_{q_i\in Q}, \Sigma, \mathcal{P}, S)$ where $\mathcal{P}$ is:

(1)	$S \to DQ_0E$	simulation starts in initial state
	$D \to xD\underline{X} \mid E$	generate input word, reverse copy for simulation
	$E \to BE \mid \epsilon$	generate sufficient free space for simulation
(2)	$\underline{X}P \to Q\underline{X'}$	for all $\delta(p, x) = (q, x', R)$ [direction reversed!]
	$\underline{X}P\underline{Y} \to \underline{X'}\,\underline{Y}Q$	for all $\delta(p, x) = (q, x', L)$
(3)	$P \to C$	for all $p \in F$
	$C\underline{X} \to C$,$\underline{X}C \to C$	clean the tape
	$C \to \epsilon$	finish, generated w

Example: $L = \{a^{2n} | n \geq 0\}$

$M = (\{q_0, q_1, q_2, q_F\}, \{a\}, \{a\}, \delta, q_0, B, \{q_F\})$ where

$$\delta(q_0, a) = (q_1, a, R),$$
$$\delta(q_1, a) = (q_0, a, R),$$
$$\delta(q_0, B) = (q_F, B, L)$$

$G = (\{S, C, D, E, Q_0, Q_1, Q_F, \underline{a}\}, \{a\}, S, \mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{P}_3)$

Initialize: $\mathcal{P}_1$

$S \to DQ_0E$

$D \to aD\underline{a} \mid E$

$E \to BE \mid \epsilon$

Simulate: $\mathcal{P}_2$

$\underline{a}Q_0 \to Q_1\underline{a}$

$\underline{a}Q_1 \to Q_0\underline{a}$

$BQ_0\underline{a} \to B\underline{a}Q_F$

Cleanup: $\mathcal{P}_3$

$Q_F \to C$

$C\underline{a} \to C$

$\underline{a}C \to C$

$BC \to C$

$C \to \epsilon$

For $w = aa$: initialize $aaB\underline{a}\underline{a}Q_0$, simulate $aaB\underline{a}Q_F\underline{a}$, cleanup: aa

Proof

Grammar to Turing machine

3.4 Linear bounded automata and context-sensitive grammars

Chapter 4: Intro to computability theory