

Section 5 Handout

CS 121

October 25, 2009

Today's Topics

- Context Free Grammars and Chomsky Normal Form
- Turing Machines
- General Grammars

1 Chomsky Normal Form.

Given a grammar G and a string w , we want to be able to tell efficiently (or relatively efficiently) whether $w \in L(G)$. A way to efficiently tell whether a given string is in a grammar is to transform the grammar into Chomsky Normal Form. Although this one-time transformation can take time exponential to the size of the grammar, once we perform it we can determine membership in the language in time $O(n^3)$.

Definition 1.1 (Chomsky Normal Form). *A context-free grammar G is in Chomsky Normal Form if every rule is either of the form $A \rightarrow BC$, $A \rightarrow a$, or $S \rightarrow \epsilon$ where $A, B, C \in V$, S is the start state, and $B, C \neq S$*

Exercise 1.1. *Consider the following Context-free Grammar:*

$$G = (\{S, T\}, \{a, b\}, \{S \rightarrow aSb, S \rightarrow T, T \rightarrow bTa, T \rightarrow e\})$$

(a) *Put G into Chomsky-Normal Form.*

(b) *Determine whether the string $abab$ is in $L(G)$.*

2 Turing Machines

Recall that a Turing Machine is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$.

Exercise 2.1. Consider the Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where

- $Q = \{q_0, q_1, q_2, q_{\text{accept}}, q_{\text{reject}}\}$,
- $\Sigma = \{a, b\}$ and $\Gamma = \{a, b, \sqcup\}$,
- The start, accept and reject states are q_0 , q_{accept} , and q_{reject} respectively.
- The function δ is given by:

q	a	b	\sqcup
q_0	(q_0, b, R)	(q_1, a, R)	$(q_{\text{reject}}, \sqcup, R)$
q_1	(q_2, b, R)	(q_1, a, R)	$(q_{\text{reject}}, \sqcup, R)$
q_2	(q_2, b, R)	(q_2, a, R)	$(q_{\text{accept}}, \sqcup, R)$

(a) Give the sequence of configurations describing M 's computation on the string $aabba$.

(b) Describe informally what M does when run on arbitrary input.

In class we talked about the Church-Turing thesis, which claims that any Turing Machines provide a universal model for computation. In particular, we showed that single-tape Turing machines are equivalent to multi-tape Turing machines. These models are also equivalent to many other computational models. To illustrate some of the flexibility and power of Turing machines and these models:

Exercise 2.2. Give an implementation-level description of a Turing machine that decides the language $\{a^{2^n} : n \geq 0\}$

Exercise 2.3. Given a PDA $M = \{Q, \Sigma, \Gamma, \delta, q_0, F\}$, construct a nondeterministic Turing machine N that simulates M .

3 General Grammars

One of the models equivalent to Turing machines that we talked about were the general grammars. Recall that general grammars are a more powerful version of context-free grammars. In any rule $u \rightarrow v$ of a general grammar, unlike context-free grammars, u may be any string of terminals and nonterminals, so long as it contains at least one nonterminal.

Exercise 3.1. Write a general grammar that generates the language $\{a^{n^2} : n \geq 0\}$.

Exercise 3.2. Write a general grammar that generates the language:

$$\{w\#1^n : n \geq 1 \text{ and } w \text{ is the binary representation of } n\}$$

4 More Exercises

Exercise 4.1. Show that the Turing-decidable languages are closed under concatenation.

Exercise 4.2. A Boring Turing Machine (BTM) can only write $\#$ on the tape (assume $\# \notin \Sigma$). Show that the BTMs are equivalent in power to the TMs.

Exercise 4.3. Consider the language

$$L = \{P : P \text{ is a polynomial such that there exists } x_1, \dots, x_n \in \mathbb{N} \text{ such that } P(x_1, \dots, x_n) = 0\}$$

- (a) Giving a high-level description, show that L is Turing-recognizable.
- (b) As food for thought, does L appear to be Turing-decidable? In general, what sorts of languages are Turing-decidable and what languages are Turing-recognizable?