Section 5 Handout

CS 121

October 18, 2010

Today's Topics

- Context Free Grammars and Chomsky Normal Form
- Turing Machines

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 $A \to BC$, $A \to a$, or $S \to \epsilon$ 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:*

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

(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 $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$.

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.*

Exercise 2.4. *Show that the Turing-decidable languages are closed under concatenation.*

Exercise 2.5. *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 2.6. *Consider the language*

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

(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?*