

Solutions for Homework 2  
Course: CS 509  
Foundations of Computer Science

Lecturer: Joe Kilian  
Department of Computer Science  
Rutgers, The State University of New Jersey

October 3, 2006

**Problem 1 (Sipser 4.22).** *A useless state in a pushdown automaton (PDA) is never entered on any input string. Give a decision procedure for whether a given pushdown automaton has a useless state.*

*Answer.* Given a pushdown automaton  $P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_{\text{accept}}\})$  and one of its states  $q \in Q$ , we could test whether  $q$  is a *useless state* by creating a PDA  $P' = (Q', \Sigma', \Gamma', \delta', q'_0, \{q'_{\text{accept}}\})$  such that  $q$  is a *useless state* if and only if  $P'$  accepts empty language. the idea is that whenever some string  $w$  reaches  $q$ , we force  $P'$  to accept  $w$ . By the following two facts (see the textbook), we can decide whether  $q$  is useless.

**Fact 1.** There is a procedure which converts a PDA to an equivalent CFG.

**Fact 2.** Whether a CFG accepts empty language or not is decidable.

Therefore we may test every state to draw our conclusion.

$P'$  looks like the following:

$$\begin{aligned}
 q' & \text{ is a new state} \\
 Q' & = Q \cup q' \\
 \Sigma' & = \Sigma \\
 \Gamma' & = \Gamma \\
 \delta' & = \delta \cup \{(q, \epsilon, \epsilon) \rightarrow (q', \epsilon)\} \\
 & \quad \cup \{(q', a, \epsilon) \rightarrow (q', \epsilon) \mid a \in \Sigma'\} \\
 & \quad \cup \{(q', \epsilon, t) \rightarrow (q', \epsilon) \mid t \in \Gamma'\} \\
 q'_0 & = q_0 \\
 q'_{\text{accept}} & = q'
 \end{aligned}$$

□

**Problem 2 (Sipser 4.24).** *Given a DFA,  $M$ , show how to decide whether there exists a palindrome  $x$  (i.e.,  $x = x^R$ ) that is accepted by  $M$ .*

*Answer.* Consider the language  $L_p$  which consists of all of palindromes. Define  $L' = L_M \cap L_p$  and  $L'$  is also context free since Theorem 3.1 in lecture 3 tells us that the intersection between context free languages and regular languages is also context free. Therefore, such  $x$  exists if and only if  $L' \neq \phi$ . Testing whether  $L'$  is empty is decidable, so is deciding the existence of  $x$ . □

**Problem 3.** *Given a language  $L$ , let  $\text{substring}(L)$  be defined by,*

$$\text{substring}(L) = \{y \mid \exists x, z \text{ such that } xyz \in L\}$$

**A** *Show that if  $L$  is recursively enumerable, then  $\text{substring}(L)$  is recursively enumerable.*

**B** *Prove or disprove: if  $L$  is decidable,  $\text{substring}(L)$  is decidable.*

*Proof.* Two parts.

Part A. If  $L$  is recursively enumerable, then there exists a Turing machine  $M_L$  which accepts  $L$ . Run all strings in  $\Sigma^*y\Sigma^*$  on  $M_L$  in the dovetailing manner. If any of these computations accepts, accept. It is easy to see that our algorithm accepts  $\text{substring}(L)$ .

Part B. False. Consider the language  $L = \{ \langle M, x, t \rangle \mid M \text{ halts on } x \text{ within } t \text{ steps} \}$  (we use different alphabets

for  $M, x$  and  $t$ ). According to the existence of universal Turing machine, it is easy to see that  $L$  is decidable. But we could not decide whether  $\langle M, x \rangle$  is in  $\text{substring}(L)$  since the Halting problem is not decidable.  $\square$

**Problem 4 (Sipser 4.18).** *Let  $A$  and  $B$  be two disjoint languages. Say that language  $C$  separates  $A$  and  $B$  is  $A \subseteq C$  and  $B \subseteq \overline{C}$ . Show that if  $A$  and  $B$  are disjoint co-recursively enumerable sets (i.e.,  $\overline{A}$  and  $\overline{B}$  are recursively enumerable and  $A \cap B = \emptyset$ ), then there exists a decidable set  $C$  that separates  $A$  and  $B$ .*

*Proof.* According to the assumption, there are Turing machines  $M_{\overline{A}}$  and  $M_{\overline{B}}$  accepting  $\overline{A}$  and  $\overline{B}$  correspondingly. Then  $C$  decides the membership of any given string  $w$  based on the following algorithm: Run  $M_{\overline{A}}$  and  $M_{\overline{B}}$  on  $w$  simultaneously, if  $M_{\overline{A}}$  accepts first, rejects; if  $M_{\overline{B}}$  accepts first, accept.

Since  $A$  and  $B$  are disjoint,  $\overline{A} \cup \overline{B} = \Sigma^*$ , which means that our algorithm will halt on every string definitely. The correctness of separating  $A$  and  $B$  is obvious since  $A \subseteq \overline{B}$  and  $B \subseteq \overline{A}$ .  $\square$

**Problem 5.** *Not all context-free grammars (CFGs) accept regular languages, but some do. How much does the size of the representation increase? Given a context-free grammar  $G$ , let  $L(G)$  be the language accepted by this grammar and let  $|G|$  be the "size" of  $G$ , given as the number of characters needed to write  $G$  down (the details are not important). For a language  $L$ , let  $\text{min-reg}(L)$  be the number of states of the smallest deterministic finite automata accepting  $L$ , if such an automata exists, and define  $\text{min-reg}(L)$  to be 0 if no such automata exists ( $L$  is not regular). Finally, define  $\text{blowup}(n)$  by*

$$\text{blowup}(n) = \max_{G, |G| \leq n} \text{min-reg}(L(G)).$$

*Show that  $\text{blowup}(n)$  grows faster than any computable function.*

*Proof.* In lecture 4, Joe proved that  $\text{ALLCFG} = \{\text{CFG} \mid \text{CFG accepts } \Sigma^*\}$  is undecidable. He showed that there is a reduction  $f$  from Halting problem  $H$  to this language. The idea basically says that if  $\langle M, x \rangle \notin H$ , then  $f(\langle M, x \rangle) \in \text{ALLCFG}$  otherwise,  $f(\langle M, x \rangle)$  would fail to accept only a single string that is the halting configuration of  $M$  on  $x$ , denoted as  $h_{\langle M, x \rangle}$ . In both cases,  $f(\langle M, x \rangle)$  is regular since it is either  $\Sigma^*$  or  $\Sigma^* - \{h_{\langle M, x \rangle}\}$ , and in the latter case, to describe this language we need a DFA of at least

$|h(\langle M, x \rangle)|$  number of states, which is also an upper-bound of the number of steps within which  $M$  halts on  $x$ . So in other words, if  $\text{blowup}(n)$  is asymptotically smaller than some computable function  $f'(n)$ , then we can use  $f'(n)$  as a threshold to decide the Halting problem, which is impossible. Therefore, the statement in the problem holds.  $\square$