

Harvard University
Computer Science 121

Problem Set 5

Due Friday, October 30, 2009 at 1:20 PM.

Late problem sets may be turned in until Monday, November 2, 2009 at 1:20 PM with a 20% penalty.

Please hand in Parts A and B separately; each part must be stapled.

All problem sets should be dropped off in the CS 121 box in the basement of Maxwell Dworkin.

See syllabus for collaboration policy.

There are several possible levels of formalism for describing Turing machines:

Formal description: You can write out a formal 7-tuple representation and use either a state diagram or a table to describe the transition function, as done in Sipser 3.9.

Implementation description: You can describe *clearly* how the tape and head of the TM work without specifying the states or the transition function, as done in Sipser 3.11 and 3.12.

High-level description: You can give a still higher level description, as done in Sipser 3.23. In this Problem Set, please give implementation description unless otherwise specified.

PART A (Graded by Olga)

PROBLEM 1 (7+6+3+2 points)

Let $G = (V, \Sigma, R, S)$ where $V = \{S, V\}$, $\Sigma = \{a, b\}$, and R is the set of rules:

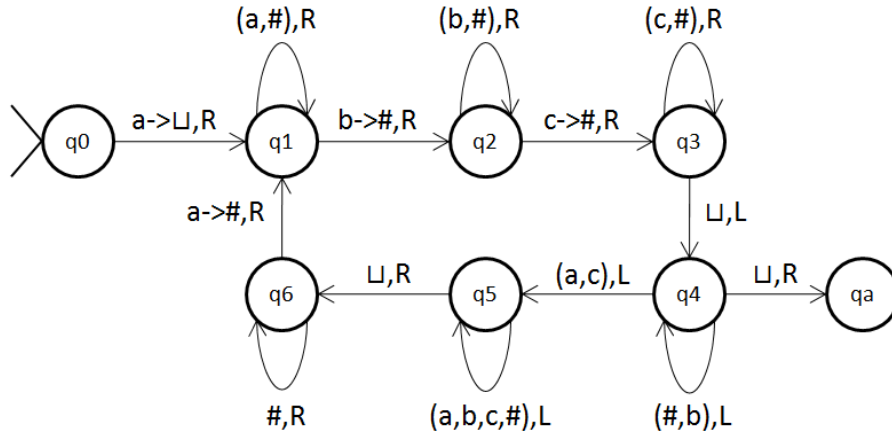
$$S \rightarrow bSS \mid aS \mid aV$$

$$V \rightarrow aVb \mid bVa \mid VV \mid \varepsilon$$

- (A) Transform G into an equivalent grammar G' in Chomsky normal form.
- (B) Verify that the string $abaab$ is generated by G' , using the recognition algorithm for grammars in Chomsky normal form given in class. Show the complete filled-in matrix.
- (C) Draw a parse tree for the derivation of $abaab$ from the transformed grammar G' .
- (D) In one sentence, what language does G generate?

PROBLEM 2 (5+5 points)

Let M be the Turing machine with states $Q = \{q_0, \dots, q_6, q_a, q_r\}$, input alphabet $\Sigma = \{a, b, c\}$, tape alphabet $\Gamma = \{a, b, c, \#, \sqcup\}$, start state q_0 , accept state q_a , and reject state q_r , and transition function given by the following state diagram:



Note: $(x, y, z), L$ means “when you read an x, y , or z , move left and do not change the tape”.
 Note: All transitions not drawn lead to the reject state.

- (A) Show each step in the computation of this machine on the string $aabbbcc$.
- (B) What language does M decide or recognize? Explain briefly.

PROBLEM 3 (12 points)

Provide as tight a bound as you can for the pumping length of a Chomsky normal form (CNF) grammar, as a function of the number of rules in the grammar.

That is, find functions $f_L(n)$ and $f_U(n)$ that are as close to each other as possible such that:

1. Any CNF grammar (V, Σ, R, S) satisfies the context-free pumping lemma with pumping length $p = f_U(|R|)$.
2. For every $n \geq 1$, there exists a CNF grammar (V, Σ, R, S) with $|R| = n$ whose pumping length is no less than $f_L(|R|)$.

PART B (Graded by David)

PROBLEM 4 (4+6+6+4 points)

True or false? Justify your answers, giving a proof or a counterexample as appropriate.

(A) For any two languages L_1 and L_2 , $L_1 = (L_1 - L_2) \cup L_2$.

(B) The language of valid regular expressions over the alphabet $\{a, b\}$ is regular.
(Here, we consider any regular expression itself to be a string over the alphabet $\{a, b, \varepsilon, (,), *, \cup, \emptyset\}$.
For instance, “ $(ab)^*a \cup \varepsilon \cup aab^*$ ”.)

(C) If A is a countable set, then $A \times \mathbb{N}$ is countable, where \mathbb{N} is the set of natural numbers.

(D) The following language is Turing-decidable:

$$L = \{a^n : n \text{ is the smallest counterexample to the generalized Twin Primes conjecture}\}$$

(The generalized Twin Primes conjecture states that for every positive even integer n , there are infinitely many prime numbers p such that $p + n$ is also prime. It is not known if the generalized Twin Primes conjecture is true or false.)

PROBLEM 5 (15+5 points)

A queue is similar to a stack, except that pushing and popping happen at opposite ends. That is, symbols are pushed onto the right of the queue, and symbols are popped off the left of the queue. A queue automaton (QA) is a deterministic automaton that, in addition to having a finite set of states and transitions, has a queue for data storage. At each step, a QA pops off the next symbol from its queue, and based on that symbol and its current state, transitions to another state and pushes any number of symbols on to the queue. When run on an input string w , a QA begins with the string $w\$$ in the queue, where $\$$ is a symbol not in the input alphabet that marks the end of the string. A QA accepts by transitioning to a special accept state. A QA can fail to accept by looping forever or if its queue becomes empty (so that it cannot pop a symbol and make any more transitions).

(A) Show by simulation that QAs and Turing machines are equivalent in power. You need not prove your constructions correct, but they should be described at the implementation-level and correctly handle any corner cases.

(B) Estimate the number of states in your construction of a QA as a function of the number of states in the simulated TM and the size of its alphabet.