CSCI 3130: Formal Languages and Automata Theory — Homework 4
The Chinese University of Hong Kong, Fall 2017 — due 11:59pm Thursday November 9

Collaborating on homework is encouraged, but you must write your own solutions in your own words and list your collaborators. Copying someone else's solution will be considered plagiarism and may result in failing the whole course.

Please answer clearly and concisely. Explain your answers. Unexplained answers will get lower scores or even no credits.

(1) (30 points) Consider the following context-free grammar G:

$$S \to (S) \mid S() \mid ()$$

It generates expressions like `()`, `(())()`, and so on.

(a) Every partially completed rule of the form $A \to \alpha \bullet \beta$ is known as an *item*. Write all items in the grammar G and construct an NFA for the valid item updates.

(b) Convert the NFA to a DFA. Which of the states are shift states and which are reduce states? Are there any conflicts?

(c) Using the DFA, show an execution of the LR(0) parsing algorithm on the input

`(()())`

Show the stack of states, stack of processed input, and remaining input throughout the execution.

(d) Now consider the following extended context-free grammar G':

$$S \to ()S \mid ()$$

Show that G' is not an LR(0) grammar.

(2) (25 points) In this problem, you will design Turing machines for the following two languages. Briefly explain how your Turing machine works (insufficient explanation may get zero points).

(a) $L_1 = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i \leqslant j \leqslant k\}$. The input alphabet is $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. Give both a high-level description and a state diagram of your Turing machine.

(b) $L_2 = \{\mathtt{a}^n \mid n \text{ is a perfect square}\}$. The input alphabet is $\Sigma = \{\mathtt{a}\}$. Give only a high-level description of the Turing machine. A state diagram is not necessary.

Recall that an integer n is a perfect square if it is the square of some integer. The first perfect squares are $0, 1, 4, 9, 16, 25, 36$.

Hint: How is one perfect square related to the next perfect square?

(3) (25 points) A *queue automaton* is like a push-down automaton without the input tape and the stack is replaced by a queue. A queue is a tape allowing symbols to be read only at the left end and written only at the right end. At each time step, the queue automaton may perform either a read or write operation. Each read operation (called a *pop*) reads and removes a symbol from the left and of the tape and each write operation (called a

push) writes a symbol at the right end. For example, if the state of the tape is `abcaaab`, the operation **pop** `a` yields `bcaaab`. Now **push** `c` yields `bcaaabc`. The internal state of the queue automaton may change after each *pop* or *push* operation (and this transition may depend on the symbol pushed or popped). Initially, the queue contains the input followed by the special end-of-input symbol `$`. The automaton accepts (rejects) by going into a special state q_{accept} (q_{reject}). The transitions in a queue automaton are deterministic. You will argue that a queue automaton is equivalent to a Turing machine: Every queue automaton can be simulated on a Turing machine, and vice versa.

(a) Write a formal definition of a queue automaton. A formal definition of an automaton will look like page 17 of Lecture 14 or page 9/slide 7 of Lecture 10.

(b) Show how to simulate a queue automaton on a Turing machine. For this, you need to specify

- how the tape of the Turing machine will be used to represent the queue automaton;
- how the Turing machine tape should be set up initially;
- what the Turing machine should do when the automaton performs a **push** or a **pop** (you may specify in 1-2 sentences the general idea, omitting the tedious details);
- what the Turing machine should do when the queue automaton accepts/rejects.

(c) Show how to simulate a Turing machine on a queue automaton. For concreteness, you may assume the Turing machine has tape alphabet $\Gamma = \{\mathtt{a}, \mathtt{b}, \square\}$. Again you should specify simulation details similar to those in part (b).

(4) (20 points) The Church–Turing Thesis is often quoted as the claim that Turing machines are a universal model of computation: Any computation that can be performed on any computer we will ever build can also be done on a Turing machine. Here are some possible objections to the Church–Turing Thesis. For each of these objections, say if you think it is reasonable or not, and explain why.

(For some parts below, you won't be graded based on whether your answer is "right" or "wrong", but based on how well you explain your answer. We expect your answer to be about 5-10 sentences long, but feel free to elaborate more if necessary. For this question, you are encouraged to research on the Internet for supporting arguments, and cite appropriately.)

(a) Suppose I want to know what is the smallest country in the world. In real life, I would use Google, type in "smallest country", and I find out the answer after a few clicks. But I cannot do this on a Turing Machine. How do I even connect a Turing Machine to the Internet? Since there are computations we can do in real life but not on a Turing Machine, the Church–Turing thesis is false.

(b) Look at this computer program:

```
int F(int n) {
  if (n == 1) return 1;
  else return F(n - 1) + F(n - 2);
}
```

This program uses *recursion*: A procedure makes a subroutine call to itself. But Turing machines do not support recursion. Therefore Turing machines are not as powerful as ordinary programming languages, and the Church–Turing thesis is false.

(c) Humans can also be modeled as computers: We take inputs from the environment (by seeing, hearing, touching) and produce outputs (via speaking and gestures). If the Church–Turing thesis is true, then any task that humans can do can also be done on a Turing Machine, and so on any machine. But there are tasks that humans are better at than machines: Learning foreign languages, identifying objects in images, winning basketball games, and so on. Therefore the Church–Turing Thesis cannot be true.