Homework 6: Turing Machines

CSE 30151 Spring 2017

Due 2017/03/30 at 11:55pm

Instructions

- Create a PDF file (or files) containing your solutions.
- Please name your PDF file(s) as follows:
  - If you're making a complete submission, please name your PDF file `netid-hw6.pdf`, where `netid` is replaced with your NetID.
  - If you're submitting some problems now and want to submit other problems later, name your PDF file `netid-hw6-123.pdf`, where `123` is replaced with the problems you are submitting at this time.
- Submit your PDF file in Sakai. Don't forget to click Submit!

Problems

1. **Designing TMs**

   (a) Write a **formal description** of a Turing machine that decides the language
   $$\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0\}.$$

   (b) Write an **implementation-level description** as defined in Section 3.3 of a Turing machine that decides the language
   $$\{\mathtt{a}^n \mid n \text{ is a Fibonacci number}\}.$$

2. **Doubly infinite tapes.** [Problem 3.11] A Turing machine with a doubly infinite tape is like a TM as defined in the book, but with a tape that extends infinitely in both directions (not just to the right). Initially, the head is at the first symbol of the input string, as usual, but there are infinitely many blanks to the left. Show how, given a TM with doubly infinite tape, to construct an equivalent standard TM. An **implementation-level description** in the style of Proof 3.13 is fine, and it's also fine to use any results proved in the book or in class.

3. **Brain fun.** This problem is about a programming language known as $\mathcal{P}''$ in polite company.[1] It was invented in 1964, in one of the foundational papers about structured programming, to show that we don't need `goto`.

   Let $\Gamma = \{a_0, \ldots, a_{n-1}\}$ and $\Sigma \subseteq \Gamma \setminus \{a_0\}$. A $\mathcal{P}''$ program works on a singly-infinite tape like a Turing machine. Each cell contains a symbol from Γ. The tape is initialized to an input string over Σ, followed by infinitely many a_0's. The head starts at the leftmost cell. Then a sequence of commands is executed sequentially. The possible commands are as follows:

`<`	Move the head to the left if possible; do nothing otherwise.
`>`	Move the head to the right.
`+`	Increment the symbol under the head: a_0 becomes a_1, a_1 becomes a_2, and so on; a_{n-1} becomes a_0.
`-`	Decrement the symbol under the head: a_{n-1} becomes a_{n-2}, a_{n-2} becomes a_{n-3}, and so on; a_0 becomes a_{n-1}.
`[` *cmds* `]`	Like a `while` loop: `while` (the symbol under the head is not a_0) `do` *cmds*.

   When the program finishes, if the symbol under the head is not a_0, the program accepts the input string; otherwise it rejects.

   For example, the following program (with $\Sigma = \{a_1, \ldots, a_{n-1}\}$) recognizes the language $\{a_i a_j \mid i + j \neq n\}$:

   ```
   [->+<]>
   ```

   Compare with the following pseudocode:

   ```
   while $tape[head] \neq 0$ do
       $tape[head]$ -= 1
       $head$ += 1
       $tape[head] = (tape[head] + 1) \% n$
       $head$ -= 1
   $head$ += 1
   return $tape[head] \neq 0$
   ```

   (a) Describe in words how to translate a $\mathcal{P}''$ program P into the **formal description** of an equivalent Turing machine M. The input to M would be a string $w \in \Sigma^*$, and it should accept w iff P accepts w. You don't need to prove the correctness of your construction. Recall that we added an N ("no move") action to the definition of TM.

   (b) Optional, not for credit: How might a TM be translated into an equivalent $\mathcal{P}''$ program?

[1] `https://bit.ly/pprimeprime`