Tutorial 01 – Turing Machines and other equivalent models of computation

Exercise 1. *Warming up*

Let a, b be integers in their binary representation.

1. Write the full description of a Turing machine that performs *addition* on input *a#b*.

2. Write the full description of a Turing machine that performs *multiplication* on input *a#b*.

Exercise 2. *Stimulating Simulation*

We will prove the details of the following crucial result mentioned in class:

Theorem. *There exists a TM $\mathcal{U}$ such that for every $x, \alpha \in \{0,1\}^*$, $\mathcal{U}(x, \alpha) = M_\alpha(x)$ where M_α denotes the TM represented by α.*
Moreover, if M_α halts on input x within T steps, then $\mathcal{U}(x, \alpha)$ halts within $C_\alpha T \log T$ steps, where $C_\alpha > 0$ depends on α but not on x.

In this exercise, we suppose that we can read and write on all tapes of a Turing machine.

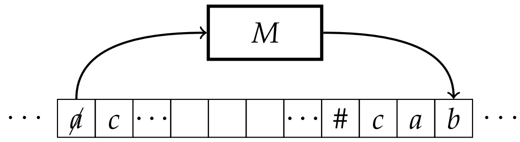
1. Show that a machine on alphabet Γ can be simulated with a machine on alphabet $\{0, 1, \square, \triangleright\}$. What is the cost of your simulation?

2. Show that a 2-tape machine can be simulated with a 1-tape machine. What is the cost of your simulation?
   Hint: You may suppose that the tape of the 1-tape machine is divided into four tracks, i.e., that every cell of the tape contains 4 symbols. Also, you may add new symbols to the alphabet Γ if you find it useful (and use the previous exercise to reduce the size of the alphabet).

3. We now want to simulate a k-tape machine using a 2-tape machine ($k \geq 3$). Show that this can be done in time $C_k T \log T$, where $C_k > 0$ depends only on k.

4. Describe the universal Turing machine $\mathcal{U}$ corresponding to the theorem.
   Hint: You may admit the following result (which can be proven using the previous exercises): There exists a TM that given $\alpha \in \{0,1\}^*$ produces a vector $\beta \in \{0,1\}^*$ such that M_β simulates M_α, has exactly 2 tapes, is on alphabet $\{0, 1, \square, \triangleright\}$, and halts within $C_\alpha T \log T$ steps.

Exercise 3. *Change the Model*



Definition. A *Post Machine* (PM, or *queue automaton*) M is a finite state machine with a single tape of unbounded length with FIFO access: in a single transition, M can choose to read and delete the symbol at the head of the FIFO queue and may append symbols to the tail of the queue.

1. Write the description of a PM M that translates a text in $\Sigma_1 = \{a, \ldots, z\}$ into a Morse text in $\Sigma_2 = \{-, \bullet, \_\}$, where _ denotes the long wait separating two characters. (If you don't have enough space on your sheet, you can restrict to the subset $\{a \to \bullet-; d \to - \bullet \bullet; e \to \bullet\}$).

2. Show that a Post Machine can be simulated by a Turing Machine.

3. Show that a Turing Machine can be simulated by a Post Machine.

   What is (somewhat) surprising with this result?

Exercise 4. *Change the Model* [2]

Definition. A *Markov algorithm* is an ordered string rewriting system P over the alphabet Σ, written $P = [\alpha_1 \to \beta_1; \ldots; \alpha_k \to \beta_k]$ along with a subset $F \subseteq P$ of *terminal rules*.
On input $u \in \Sigma^*$, a potentially infinite unique string sequence $u \xrightarrow{0} u_1 \xrightarrow{1} u_2 \xrightarrow{2} \ldots \xrightarrow{n} u_n \xrightarrow{n+1} \ldots$ is defined for $u_j \xrightarrow{j} u_{j+1}$ as the application on the leftmost instance of an α_i on the current string u_j of the first rule that can apply in P.
The algorithm stops on step $n-1$ and returns u_n if the last step $\xrightarrow{n-1} \in F$ and no rules can apply on u_n. Otherwise the algorithm does not terminate.

1. What does the following Markov algorithm do?

$$P = [\dot{1}0 \to 0\dot{1}\dot{1}; \quad 1 \to 0\dot{1}; \quad 0 \to \varepsilon] \qquad F = [0 \to \varepsilon]$$

2. Write a Markov algorithm computing the successor of a binary integer. You can suppose that the end of the integer is marked by the symbol #.

3. Show that a Markov algorithm can be simulated by a Turing machine.

4. Show that a Turing Machine can be simulated by a Markov Algorithm.