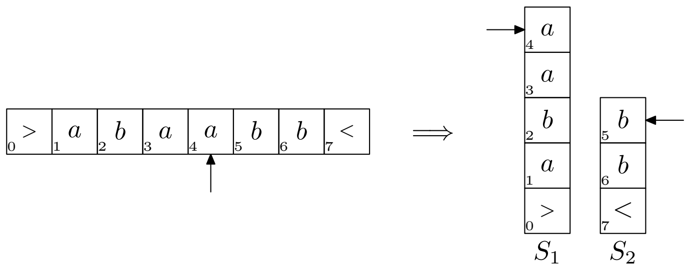
T-79.1001 **Syksy 2005**
Introduction to Theoretical Computer Science (T)
Session 9
Answers to demonstration exercises

4. **Problem**: Show that pushdown automata with two stacks (rather than just one as permitted by the standard definition) would be capable of recognizing exactly the same languages as Turing machines.

   Solution: We first show that a two-stack pushdown automaton can simulate a Turing machine. The only difficulty is to find a way to simulate the Turing machine tape using two stacks. This can be done using a construction that is similar to the one presented in the first problem: the first stack holds the part of tape that is left to the read/write head (in reversed order), and the second stack holds the symbols that are right to the head.



   The computation of the automaton can be divided into two parts:

   (a) Initialization, when the automaton copies the input to stack S_1 one symbol at a time, and then moves it, again one-by-one, to stack S_2. (With the exception of the first symbol).

   (b) Simulation, where the automaton decides its next transition by examining the top symbol of S_1. If the machine moves its head to left, the top element of S_1 is moved into S_2. If it moves to the other direction, the top element of S_2 is moved to S_1.

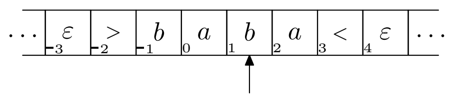
   A two-stack pushdown automaton that is formed using these principles simulates a given Turing machine. The formal details are presented in an appendix.

   Next we show that we can simulate a two-stack pushdown automaton using a Turing machine. This can be done trivially using a two tape nondeterministic Turing machine where both stacks are stored on their own tapes.

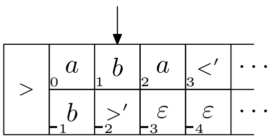
5. **Problem**: Extend the notion of a Turing machine by providing the possibility of a two-way infinite tape. Show that nevertheless such machines recognize exactly the same languages as the standard machines whose tape is only one-way infinite.

   Solution: A Turing machine with a two-way infinite tape works otherwise in a same way than a standard machine except that the position of the tape start symbol (>) is not fixed and it can move in a same way than the end symbol (<). The tape positions are indexed by the set $\mathbb{Z}$ of integers where 0 denotes the initial position of >.

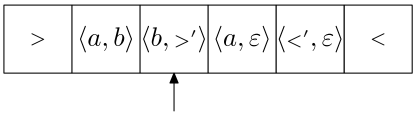
   We can simulate such a Turing machine by a two-track one-way Turing machine. Conceptually, we divide the tape into two parts: upper and lower. The upper part holds the two-way tape cells $i \geq 0$ and the lower part cells $i < 0$. For example, a two-way tape:



is expressed as a one-way tape:



In practice the construction of two tracks is done by replacing the alphabet Σ by a new alphabet $\Sigma' = (\Sigma \cup \{<', >'\}) \times (\Sigma \cup \{<', >'\})$. Each symbol of Σ' thus denotes two symbols of Σ. The symbols $\{<', >'\}$ are new symbols that denote the start and end symbols of the original tape. So, the above example is expressed as:



We still need a way to decide which tape-half is used. The easiest way to do this is to define a mirror image state q' for each state q. When the machine is in state q, it examines only the upper track when it decides what move to take next (tape head is on right side of the tape). Similarily, in state q' it examines only the lower symbol (tape head is on the left side). Since the lower tape is in a reversed order, all tape head moving instructions have to be also reversed.

The formal definition of this construction is presented in an appendix.

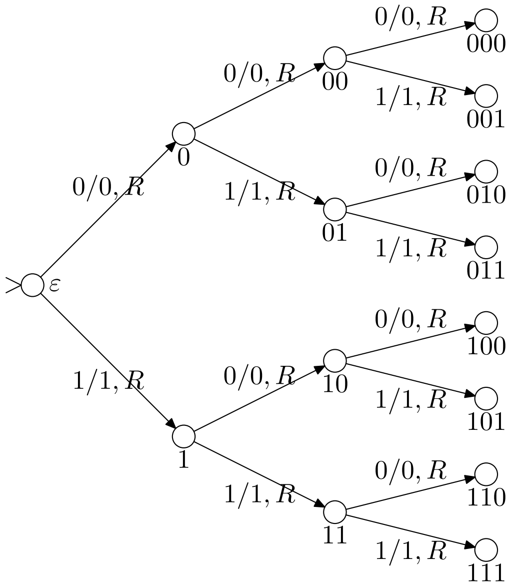
6. **Problem**: Show that Turing machines whose tape alphabet contains at most two symbols in addition to the input symbols are capable of recognising exactly the same languages as the standard machines.

   Solution:

   Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}})$ be a Turing machine such that $|\Gamma - \Sigma| > 2$. We want to construct a machine M' such that $\Gamma' = \{0, 1\}$. Let $\Gamma = \{a_1, \ldots, a_n\}$. The basic idea of the construction is to identify the elements of Γ with the integers $\{ 1, \ldots, n \}$ and represent them as k-bit integers, where $k = \lceil \log_2(|\Gamma|)$. In other words, each element of M's tape alphabet is replaced with k bits. For example, suppose that $N = 3$ and the tape has the input $a_1 a_2 a_3$. In this case the encoding is:

   $$\boxed{>}\boxed{\underline{a_1}}\boxed{a_2}\boxed{a_3}\boxed{<} \Longrightarrow \boxed{>}\boxed{\underline{0}}\boxed{1}\boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{<}$$

   The transition function of M' is defined so that for each step of M, M' does first k steps where it first decides what symbol of Γ is encoded in the tape cells to the right of the read/write head. This can be done using a Turing machine that reads k symbols from the tape while moving its head to right at each step and that remembers the input in its states. For example, if $k = 3$, then the following Turing machine may be used:



If the machine ends in the state 011, then the input symbol is a_3 since $011_2 = 3_{10}$. The symbol that is written to the tape is similarily done using k different transitions. Finally, the tape head is moved k steps to the correct direction.

Appendix: the formalisation of solution 5

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}})$ be a two-way tape Turing machine. Define a standard Turing machine M' as follows:

$$\begin{aligned} M' &= (Q', \Sigma', \Gamma', \delta', q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}}) \\ Q' &= Q \cup \{q' \mid q \in Q\} \\ \Sigma' &= (\Sigma \cup \{<', >'\}) \times (\Sigma \cup \{<', >'\}) \\ \Gamma' &= (\Gamma \cup \{<', >'\}) \times (\Gamma \cup \{<', >'\}) \end{aligned}$$

The transition function δ' is defined as follows:

$$\begin{aligned} \delta' = \; & \{(q_1, \langle a, \gamma \rangle, q_2, \langle b, \gamma \rangle, \Delta) \mid (q_1, a, q_2, b, \Delta) \in \delta, \gamma \in \Gamma'\} \\ & \cup \{(q_1, \langle \sigma', \gamma \rangle, q_2, \langle b, \gamma \rangle, \Delta) \mid (q_1, \sigma, q_2, b, \Delta) \in \delta, \gamma \in \Gamma', \sigma \in \{<, >\}\} \\ & \cup \{(q_1', \langle \gamma, a \rangle, q_2', \langle \gamma, b \rangle, \overline{\Delta}) \mid (q_1, a, q_2, b, \Delta) \in \delta, \gamma \in \Gamma'\} \\ & \cup \{(q', \langle \gamma, a \rangle, q_{\mathrm{end}}, \langle \gamma, b \rangle, \overline{\Delta}) \mid (q, a, q_{\mathrm{end}}, b, \Delta) \in \delta, q_{\mathrm{end}} \in \{q_{\mathrm{acc}}, q_{\mathrm{rej}}\}, \gamma \in \Gamma'\} \\ & \cup \{(q_1', \langle \gamma, \overline{\sigma}' \rangle, q_2', \langle \gamma, b \rangle, \overline{\Delta}) \mid (q_1, \sigma, q_2, b, \Delta) \in \delta, \gamma \in \Gamma', \sigma \in \{<, >\}\} \\ & \cup \{(q, >, q', >, R), (q', >, q, >, R) \mid q \in Q\}, \end{aligned}$$

where $\overline{L} = R$, $\overline{R} = L$, $\overline{<} = >$ and $\overline{>} = <$.