T-79.148 **Spring 2002**
Introduction to Theoretical Computer Science
Tutorial 9
Solutions to the demonstration problems

4. **Problem**: Prove that the class of context-free languages is not closed under intersections and complements. (*Hint:* Represent the language $\{a^k b^k c^k \mid k \geq 0\}$ as the intersection of two context-free languages.)

   Solution: Let $L = \{a^k b^k c^k \mid k \geq 0\}$. This language has been proven context-free (see compendium, p. 72). We can prove that context-free languages are not closed under intersection by finding two context-free languages L_1 and L_2 such that $L = L_1 \cap L_2$. Languages $L_1 = \{a^* b^k c^k \mid k \geq 0\}$ and $L_2 = \{a^k b^k c^* \mid k \geq 0\}$ fulfill this condition.

   A direct corollary is that the class of context-free languages cannot be closed under complementation, either, since they are closed under union and $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

   Finally, we prove that L_1 and L_2 are context-free by presenting context-free grammars that generate them. The language L_1 is generated by $G_1 = (\{S, A, B, a, b, c\}, \{a, b, c\}, P_1, S)$, where $P_1 = \{S \to AB, A \to aA \mid \varepsilon, B \to bBc \mid \varepsilon\}$. Similarly, L_2 is generated by $G_2 = (\{S, A, B, a, b, c\}, \{a, b, c\}, P_2, S)$, $P_2 = \{S \to AB, A \to aAb \mid \varepsilon, B \to cB \mid \varepsilon\}$.

5. **Problem**: Design Turing machines NEXT and DUP that perform the following tasks:

   (a) NEXT replaces a string given on the machine's tape by its immediate lexicographic successor;

   (b) DUP duplicates the string given on the tape, thus e.g. replacing the string *abb* by the string *abbabb*.

   Solution:

   (a) The lexicographic order $<_L$ over a set Σ^* is formed in terms of a total order $< \subset \Sigma \times \Sigma$. Most commmonly, $<$ is either numerical or alphabetic order. For example,

   $$\text{If } \Sigma = \{a, \ldots, z\}, \text{ then } a < b < c < \cdots < z$$
   $$\text{If } \Sigma = \{0, 1\}, \text{ then } 0 < 1$$

   The lexicographic order $<_L$ is defined as follows: Let $x, y \in \Sigma^*$, $x = x_1 \cdots x_n$ and $y = y_1 \cdots y_m$. Now, $x <_L y$ if one of the following conditions hold:

   (i) $n < m$, or

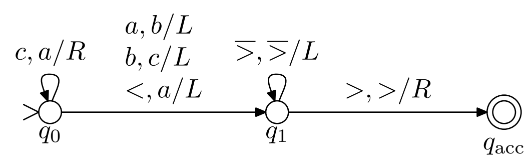
   (ii) $n = m$ and there exists $i \leq n$ such that $x_i < y_i$ and for all $j < i$, $x_i = y_i$.

   For example, the lexicographic order over the words of the alphabet $\{0, 1\}$ is defined as follows:

   $$\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, \ldots$$

   The solution constructs a Turing machine that works with the alphabet $\{a, b\}$. However, it is easy to generalize the construction to work with any alphabet (details in appendix).

   NEXT:



   In state q_1 we have used a short-hand notation $\overline{>}$ to denote any symbol other than the tape start symbol $>$. The machine works by changing all least significant c-letters to a-letters while it stays in the state q_0. As soon as the first letter $\sigma < c$ is found, it is replaced by the alphabetically next letter. Finally, the tape read/write head is returned to the start of the tape. (The reason for this is that it makes combining Turing machines easier).

Consider how NEXT finds the successor for $bacc$:

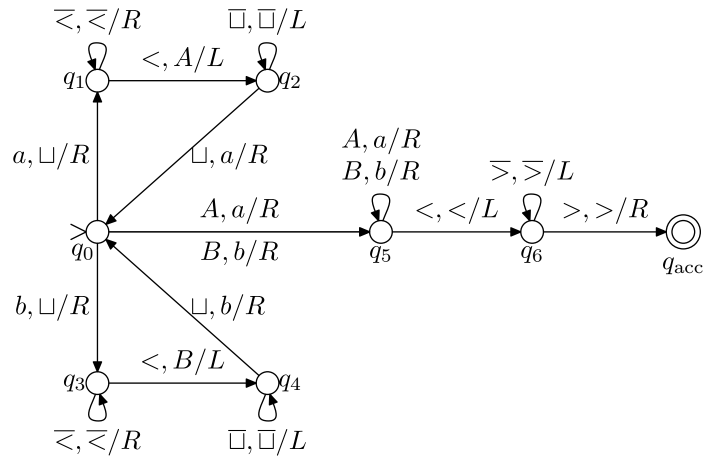
$$\begin{aligned}(q_0, \underline{c}cab) &\vdash (q_0, a\underline{c}ab) \vdash (q_0, aa\underline{a}b) \vdash (q_1, a\underline{a}bb) \\ &\vdash (q_1, \underline{a}abb) \vdash (q_1, \underline{>}aabb) \vdash (q_{\text{acc}}, \underline{a}abb)\end{aligned}$$

The result is $bbaa$.

(b) This solution supposes that $\Sigma = \{a, b\}$. However, it is trivial to extend this to allow also other alphabets.

The basic idea is that we copy one symbol to the end of the tape at a time. We keep track of the position of the currently worked-upon symbol by replacing it by $\sqcup$. We write the copy of the word initially using capital letters $\{A, B\}$ since otherwise we could not notice when the original word ends and the copy starts. Finally, all upper case letters are replaced by their lower case equivalents and the read/write head returned to the start of the tape.

DUP:



Consider how DUP works with the input abb:

$$\begin{aligned}(q_0, \underline{a}bb) &\vdash (q_1, \sqcup\underline{b}b) \vdash^* (q_1, \sqcup bb\underline{<}) \vdash (q_2, \sqcup b\underline{b}A) \vdash^* (q_2, \underline{\sqcup}bbA) \vdash (q_0, a\underline{b}bA) \\ &\vdash (q_3, a\sqcup\underline{b}A) \vdash^* (q_3, a\sqcup bA\underline{<}) \vdash (q_4, a\sqcup b\underline{A}B) \vdash^* (q_0, ab\underline{b}AB) \\ &\vdash^* (q_0, abb\underline{A}BB) \vdash (q_5, abba\underline{B}B) \vdash^* (q_5, abbabb\underline{<}) \vdash^* (q_{\text{acc}}, \underline{a}bbabb)\end{aligned}$$

Appendix: generalizing solution 5a

Let Σ be a finite alphabet and $<\subset \Sigma^* \times \Sigma^*$ be a full order. Since Σ is finite, $<$ is well-founded and it has both minimum a_{min} and maximum a_{max}. Let us define a successor function $f : (\Sigma - \{a_{\text{max}}\}) \to \Sigma$ as follows:

$$f(a) = b \Leftrightarrow a < b \land \neg\exists c : a < c \land c < b$$

Since $<$ is a full order, $f(a)$ is unambiguous.

A Turing machine M that computes the lexicographic successor of the input x is defined as follows: $M = (Q, \Sigma, \Gamma, \delta,\ q_0,\ q_{\text{acc}},\ q_{\text{rej}})$,

$$\begin{aligned}
Q &= \{q_0, q_1, q_{\text{acc}}, q_{\text{rej}}\} \\
\Gamma &= \Sigma \\
\delta &= \{(q_0, a_{\max}, q_0, a_{\min}, R), (q_0, <, q_1, a_{\min}, L)\} \\
&\quad \cup \{(q_0, a, q_1, f(a), L) \mid a \in (\Sigma - \{a_{\max}\})\} \\
&\quad \cup \{(q_1, a, q_1, a, L) \mid a \in (\Sigma \cup \{<\})\} \\
&\quad \cup \{(q_1, >, q_{\text{acc}}, >, R)\}
\end{aligned}$$

We see that we can obtain the Turing machine that was presented in the solution 5a directly from the above definition by setting $a_{\min} = a$, $a_{\max} = c$, $f(a) = b$, and $f(b) = c$.