Free University of Bozen-Bolzano – Faculty of Computer Science
Master of Science in Computer Science
Theory of Computing – A.A. 2004/2005
Final exam – 7/6/2005 – Part 2
Solutions

Problem 2.1 [6 points] Decide which of the following statements is TRUE and which is FALSE. You must give a brief explanation of your answer to receive full credit.

(a) For all languages L, if L is in NP, then its complement $\overline{L}$ is in P.

(b) For all languages L_1 and L_2, if L_1 is in P and $L_1 <_{poly} L_2$, then L_2 is in NP.

(c) The class NP is closed under intersection.

(d) There exists a language L such that L is recursively enumerable and $\overline{L}$ is recursive.

Solution:

(a) If L is in NP, then its complement $\overline{L}$ is in coNP. In fact, we do not know whether coNP is equal to P or not, so we do not know whether the statement is TRUE or FALSE.

(b) FALSE. If we know that L_1 is in P and $L_1 <_{poly} L_2$, then we have in fact no upper bound on L_2, and in particular we do not know whether L_2 is in NP.

(c) TRUE. Given two NTM M_1 and M_2 accepting respectively L_1 and L_2 in polynomial time, we can construct an NTM M accepting $L_1 \cap L_2$ in polynomial time as follows. M is a 2-tape TM that first copies the content of the input tape (tape 1) to tape 2, then runs M_1 on tape 1, and from the accepting state of M_1 moves to a state that runs M_2 on tape 2. M accepts if M_2 accepts. Since M_1 and M_2 run in polynomial time, and the additional bookkeeping can be done in polynomial time, also M runs in polynomial time. Moreover, the 2-tape TM can be simulated in polynomial time by a 1-tape TM.

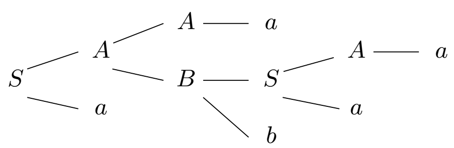
(d) TRUE. Every recursive language is such a language (and all such languages are recursive). Indeed, a recursive language is recursively enumerable and its complement $\overline{L}$ is recursive. (Note that L should be "recursively enumerable", and not "recursively enumerable and non-recursive".)

Problem 2.2 [6 points] Consider the context-free grammar $G = (\{S, A, B\}, \{a, b, c\}, P, S)$, where P consists of the following productions

$$
\begin{array}{rcl}
S & \longrightarrow & aA \\
A & \longrightarrow & BA \mid a \\
B & \longrightarrow & bS \mid cS
\end{array}
$$

Construct a PDA M that accepts $\mathcal{L}(G)$ by empty stack. Draw the parse tree of G for the string $abaaa$, and show the corresponding execution trace for M.

Solution: Parse tree for the string $abaaa$:



$M = (\{q_0\}, \{a, b, c\}, \{a, b, c, S, A, B\}, \delta, q_0, S, \emptyset)$, with δ defined as follows:

$$\begin{array}{rcl} \delta(q_0, \varepsilon, S) & = & \{(q_0, aA)\} \\ \delta(q_0, \varepsilon, A) & = & \{(q_0, BA), (q_0, a)\} \\ \delta(q_0, \varepsilon, B) & = & \{(q_0, bS), (q_0, cS)\} \end{array} \qquad \begin{array}{rcl} \delta(q_0, a, a) & = & \{(q_0, \varepsilon)\} \\ \delta(q_0, b, b) & = & \{(q_0, \varepsilon)\} \\ \delta(q_0, c, c) & = & \{(q_0, \varepsilon)\} \end{array}$$

Execution trace of M for the string $abaaa$: $(q_0, abaaa, S) \vdash (q_0, abaaa, aA) \vdash (q_0, baaa, A) \vdash (q_0, baaa, BA) \vdash (q_0, baaa, bSA) \vdash (q_0, aaa, SA) \vdash (q_0, aaa, aAA) \vdash (q_0, aa, AA) \vdash (q_0, aa, aA) \vdash (q_0, a, A) \vdash (q_0, a, a) \vdash (q_0, \varepsilon, \varepsilon)$

Problem 2.3 [6 points] Consider the context free grammar $G = (\{S, A, B, C\}, \{a, b\}, P, S)$ where P consists of the following productions:

$$\begin{array}{rcl} S & \longrightarrow & aAb \mid bBa \mid AA \\ A & \longrightarrow & S \mid B \end{array} \qquad \begin{array}{rcl} B & \longrightarrow & C \\ C & \longrightarrow & S \mid \varepsilon \end{array}$$

Convert G into Chomsky Normal Form. Illustrate the various steps of the algorithm.

Solution:

1. *Eliminate ε-productions.* The nullable non-terminal symbols are C, B, A, S, i.e., all non-terminals. By eliminating ε-productions (except for the start symbol S) we obtain the grammar G_1 accepting $\mathcal{L}(G) \setminus \{\varepsilon\}$:

$$\begin{array}{rcl} S & \longrightarrow & aAb \mid bBa \mid AA \mid ab \mid ba \mid A \\ A & \longrightarrow & S \mid B \end{array} \qquad \begin{array}{rcl} B & \longrightarrow & C \\ C & \longrightarrow & S \end{array}$$

2. *Eliminate unit productions.* We have that

$$S \Rightarrow A \Rightarrow B \Rightarrow C \Rightarrow S \Rightarrow aAb \mid bBa \mid AA \mid ab \mid ba$$

   Hence, by eliminating the unit productions we obtain the grammar G_2 equivalent to G_1:

$$\begin{array}{rcl} S & \longrightarrow & aAb \mid bBa \mid AA \mid ab \mid ba \\ A & \longrightarrow & aAb \mid bBa \mid AA \mid ab \mid ba \\ B & \longrightarrow & aAb \mid bBa \mid AA \mid ab \mid ba \\ C & \longrightarrow & aAb \mid bBa \mid AA \mid ab \mid ba \end{array}$$

3. *Eliminate non-generating symbols.* All symbols are generating.

4. *Eliminate unreachable symbols.* The symbol C is unreachable. By eliminating C and the productions involving C we obtain the grammar G_3 equivalent to G_2:

$$\begin{array}{rcl} S & \longrightarrow & aAb \mid bBa \mid AA \mid ab \mid ba \\ A & \longrightarrow & aAb \mid bBa \mid AA \mid ab \mid ba \\ B & \longrightarrow & aAb \mid bBa \mid AA \mid ab \mid ba \end{array}$$

5. *Convert into Chomsky Normal Form.* By arranging bodies of length 2 or more to consist only of variables, and then breaking bodies of length 3 or more into a cascade of productions, we obtain the grammar G_4 that is in Chomsky Normal Form and is equivalent to G_3, and hence to G_2, G_1, and G:

$$\begin{array}{rcl} S & \longrightarrow & A'D \mid B'E \mid AA \mid A'B' \mid B'A' \\ A & \longrightarrow & A'D \mid B'E \mid AA \mid A'B' \mid B'A' \\ B & \longrightarrow & A'D \mid B'E \mid AA \mid A'B' \mid B'A' \end{array} \qquad \begin{array}{rcl} A' & \longrightarrow & a \\ B' & \longrightarrow & b \\ D & \longrightarrow & AB' \\ E & \longrightarrow & BA' \end{array}$$

Problem 2.4 [6 points] Describe how to obtain, for any given TM M, a new TM M' such that $\mathcal{L}(M') = \mathcal{L}(M)$, but when M' halts its tape is exactly as in the initial configuration (i.e., it contains the input string w, and the head of M' is positioned on the leftmost symbol of w). You can make use of standard constructions presented in the course.

Solution: Let $M = (Q, \Sigma, \Gamma, \delta, q_0, \mathtt{B}, F)$. Without loss of generality, we can assume that the tape alphabet Γ does not contain the symbol $\sharp$, that there is a single final state q_f, i.e., $F = \{q_f\}$, and that δ does not contain transitions from q_f. We first construct a 2-tape TM M' that performs the same transitions as M, except for the following modifications:

1. It first copies the content of tape 1 (the input tape) to tape 2, positions the head on the first symbol of tape 1, and then moves to the initial state q_0 of M.

2. Whenever M would write a B on the tape, M' writes a $\sharp$ instead (we have assumed that $\sharp$ is not part of the tape alphabet of M). This will allow M' to erase the tape content at the end.

3. For each transition $\delta(q, \mathtt{B}) = (q', x, d)$ of M, M' contains also a transition $\delta(q, \sharp) = (q', y, d)$, where $y = x$ if $x \neq \mathtt{B}$, and $y = \sharp$ if $x = \mathtt{B}$. This takes into account that M' behaves on a $\sharp$ exactly in the same way as M does on a B.

4. The accepting state q_f of M is not accepting for M'. Instead, M' contains the following transitions:
$$\begin{array}{rcll} \delta(q_f, x) & = & (q_r^a, x, R), & \text{for each } x \in \Sigma \cup \{\sharp\} \\ \delta(q_r^a, x) & = & (q_r^a, x, R), & \text{for each } x \in \Sigma \cup \{\sharp\} \setminus \{\mathtt{B}\} \\ \delta(q_r^a, \mathtt{B}) & = & (q_e^a, \mathtt{B}, L) & \\ \delta(q_e^a, x) & = & (q_e^a, \mathtt{B}, L), & \text{for each } x \in \Sigma \cup \{\sharp\} \setminus \{\mathtt{B}\} \\ \delta(q_e^a, \mathtt{B}) & = & (q_c^a, \mathtt{B}, R) & \end{array}$$
Notice that we have written the above transitions as if M' were a single-tape TM, i.e., ignoring tape 2. Intuitively, when in q_f, M' switches to q_r^a in which it sweeps to the rightmost symbol on the tape. Then it switches to q_e^a from where it sweeps left till it finds a B, erasing the tape while doing so. Then it moves to state q_c^a.

   From state q_c^a, M' copies the content of tape 2 to tape 1, and then positions head 1 on the leftmost symbol on tape 1, and moves to an accepting state.

5. Similarly as for the moves above from the accepting state q_f, for each state q and tape symbol x for which $\delta(q, x)$ is undefined, M' contains a transition $\delta(q, x) = (q_r^n, x, R)$, and then transitions for states q_r^n, q_e^n, and q_c^n that are analogous to those for states q_r^a, q_e^a, and q_c^a, respectively, except that, in this case, at the end M' rejects instead of accepting.

Finally, we can convert M' to a single-tape TM using the standard construction.

Problem 2.5 [6 points] For a TM M, let $\mathcal{E}(M)$ denote the encoding of M. Consider the language $L = \{\mathcal{E}(M) \mid M$, when started on a blank tape eventually writes a 1 somewhere on the tape$\}$.

(a) Show that L is recursively enumerable. [*Hint*: Make use of a universal TM.]

(b) Show that L is not recursive. [*Hint*: Exploit a reduction from the halting problem.]

Solution:

(a) We can construct a TM M_L that accepts L as follows. We first let M_L transform its input $\mathcal{E}(M)$ so that it becomes the encoding of the pair $\langle \mathcal{E}(M), \varepsilon \rangle$. Then M_L behaves as a universal TM U, except that, when U would accept, M_L blocks in a non-final state, and when U would write a 1 on the tape, M_L moves instead to an accepting state.

(b) Let $H = \{\mathcal{E}(M) \mid M$, when started on a blank tape eventually halts$\}$ be the halting language, i.e., the language of codes of TMs that halt on the blank tape. Given a TM M, we show how to convert it to a TM M' such that $\mathcal{E}(M) \in H$ if and only if $\mathcal{E}(M') \in L$. The conversion we are going to describe is such that it can be performed by a TM that always terminates, i.e., an algorithm, that takes $\mathcal{E}(M)$ as input and leaves $\mathcal{E}(M')$ as output on the tape. Hence, it provides a reduction from the halting language H to the language L. Since H is not recursive, neither can be L.

To obtain M', the TM M is modified as follows:

1. All occurrences of symbol 1 in the transitions of M (either read or written) are replaced in M' with $\sharp$, where $\sharp$ is a new symbols not occurring in the tape alphabet of M.
2. The final states of M are no longer final in M'. Instead, for each final state q_f of M, we add to M' transitions $\delta(q_f, x) = (q_f, 1, R)$, for every tape symbol x.
3. Similarly, for each state q and tape symbol x for which $\delta(q, x)$ is undefined in M (and hence M would block), we add to M' a transition $\delta(q, x) = (q_f, 1, R)$.

It is easy to see that M will eventually halt when started on the blank tape if and only if M' will eventually write a 1 somewhere on the tape.