CS30053 Foundations of Computing, Autumn 2005

End-semester examination : Solutions

---

1. Prove that the following languages are not regular.

(a) $L_{11} = \{a^m b^n \mid m, n > 0 \text{ and } \gcd(m, n) = 1\} \subseteq \{a, b\}^*$.

Solution Let L_{11} be regular. Take any prime p such that p is at least two more than a pumping lemma constant for L_{11}. Consider the string $\alpha = a^p b^{(p-1)!} \in L_{11}$. The pumping lemma gives a decomposition of α of the form $\alpha_1\alpha_2\alpha_3$ with $|\alpha_1\alpha_2| \leqslant p-2$, $1 \leqslant |\alpha_2| \leqslant p-2$ and $\alpha_1\alpha_2^k\alpha_3 \in L_{11}$ for all $k \in \mathbb{N}_0$. Let $r = |\alpha_2|$. Since the length of $\alpha_1\alpha_2$ is no more than the pumping lemma constant, it follows that α_2 consists of a's only. Take $k = 0$, i.e., $\alpha_1\alpha_3 = a^{p-r}b^{(p-1)!} \in L_{11}$. By the choice of p we have $2 \leqslant p-r \leqslant p-1$ and so $\gcd((p-r), (p-1)!) = p-r \geqslant 2$, a contradiction to the definition of L_{11}.

(b) $L_{12} = \{a^m b^n \mid m, n > 0 \text{ and } \gcd(m, n) > 1\} \subseteq \{a, b\}^*$.

Solution One can invoke the pumping lemma independently to prove that L_{12} is not regular. Here is an easier proof. Consider the language

$$L_1 = \{a^m b^n \mid m, n > 0\}.$$

L_1 is the language of the regular expression aa^*bb^* and so is regular. Also L_1 is the disjoint union of L_{11} and L_{12}, that is, $L_{11} = L_1 \setminus L_{12} = L_1 \cap \overline{L_{12}}$. If L_{12} were regular, so also would be L_{11}, since regular languages are closed under complementation and intersection. But we have already proved in Part (a) that L_{11} is not regular. So L_{12} cannot be regular.

2. Let $\Sigma = \{a, b, c, d\}$. Consider the language

$$L_2 = \{a^i b^j c^k d^l \mid i, j, k, l \geqslant 0 \text{ and } i + j = k + l\} \subseteq \Sigma^*.$$

(a) Design a context-free grammar G with $\mathcal{L}(G) = L_2$.

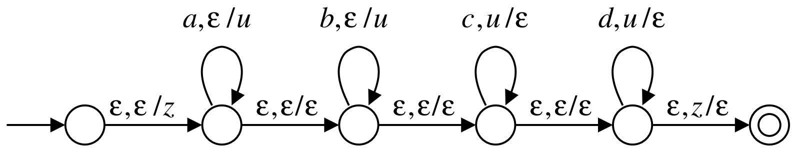
Solution Let S be the start variable. We first generate matching a's and d's. This step is repeated $\min(i, l)$ times. Next we branch based on the condition whether $i \leqslant l$ or $i \geqslant l$. If $i \leqslant l$, then a's are exhausted earlier than d's. So the remaining d's are to be matched against b's. After that we have to match the leftover b's with the c's. If $i \geqslant l$, the d's are exhausted first and the leftover a's are to be matched against c's. After all a's are taken care of, the remaining c's are matched against the b's. Thus the grammar $G = (\{S, U, V, W\}, \{a, b\}, R, S)$ generates L_2, where R consists of the following productions:

$$\begin{array}{rcl} S & \rightarrow & aSd \mid U \mid V \\ U & \rightarrow & bUd \mid W \\ V & \rightarrow & aVc \mid W \\ W & \rightarrow & bWc \mid \epsilon \end{array}$$

Here the variable U corresponds to the case $i \leqslant l$, and V to the case $i \geqslant l$. In both these cases, the final matching of b's against c's is handled by the variable W.

(b) Design a PDA to recognize L_2.

Solution We can use the CFG-to-PDA conversion procedure for constructing a PDA to recognize L_2. Let me instead design a PDA from the first principles. The PDA first reads a's and then the b's and keeps track of the total count of a's and b's read. Then it reads runs of c's and d's and matches the stored count against the total count of c's and d's. If the input string is not in the correct format (for example, when a b follows an a), the machine goes to the stuck position. Moreover, the jumps between runs of a's and b's, of b's and c's and of c's and d's are effected by ϵ-transitions. The following figure describes the PDA. Here the symbol $x, y/z$ means read x from the input and replace y at the top of the stack by z. Each of x, y, z is allowed to be empty (ϵ).



3. Prove that the following languages are decidable. Provide only high-level descriptions of deciders.

(a) $\mathrm{A}_{\text{DFA},n} = \{\langle D, n\rangle \mid D \text{ is a DFA that accepts some string of length } n\}$.

Solution Simulate D one-by-one on all strings of length n. Since A_{DFA} is decidable and since there are only finitely many ($|\Sigma|^n$) strings of length n, the simulation ends after a finite amount of time. If the DFA accepts any of these strings, *accept*. If the DFA rejects all strings of length n, *reject*.

(b) $\mathrm{SUBSET}_{\text{DFA}} = \{\langle D_1, D_2\rangle \mid D_1, D_2 \text{ are DFA with } \mathcal{L}(D_1) \subseteq \mathcal{L}(D_2)\}$. (Hint: Look at $\mathcal{L}(D_1) \setminus \mathcal{L}(D_2)$.)

Solution We have $\mathcal{L}(D_1) \subseteq \mathcal{L}(D_2)$ if and only if $\mathcal{L}(D_1) \setminus \mathcal{L}(D_2) = \mathcal{L}(D_1) \cap \overline{\mathcal{L}(D_2)} = \emptyset$. Given D_1, D_2, a DFA D can be constructed to recognize $\mathcal{L}(D_1) \cap \overline{\mathcal{L}(D_2)}$ (recall that regular languages are closed under complementation and intersection). Then feed the description of the DFA D to a decider for E_{DFA}.

(c) $\mathrm{FINITE}_{\text{PDA}} = \{\langle P\rangle \mid P \text{ is a PDA with } \mathcal{L}(P) \text{ finite}\}$. (Hint: Let n be a pumping lemma constant for $\mathcal{L}(P)$. First prove that $\mathcal{L}(P)$ is infinite if and only if $\mathcal{L}(P)$ contains a string of length between n and $2n-1$.)

Solution Let P be a PDA, $L = \mathcal{L}(P)$, and n a pumping lemma constant for L. If L contains a string α of length $\geqslant n$, then the pumping lemma on α gives an infinite collection of strings each of which belongs to L. In order that L is finite we then require L to consist of no strings of length $\geqslant n$. However, we cannot check that this condition is satisfied by simulating the PDA P on all strings of length $\geqslant n$, since there are infinitely many such strings and the sequence of simulation does not halt. Assume that L is infinite and l is the minimum length of a string in L of length $\geqslant n$. We claim that $n \leqslant l \leqslant 2n-1$. Assume not, i.e., $l \geqslant 2n$. Let α be a string of length l in L. The pumping lemma gives a decomposition $\alpha = \alpha_1\alpha_2\alpha_3\alpha_4\alpha_5$ so that $\beta = \alpha_1\alpha_3\alpha_5$ is in L too. We have $1 \leqslant |\alpha_2\alpha_4| \leqslant n$ by the pumping lemma. So β is again a string in L of length $\geqslant n$. This contradicts the choice of l (and α).

So it suffices to check only the strings of length between n and $2n-1$. There are finitely many of them. Since A_{PDA} is decidable, a TM can check in finite time whether each of these strings belongs to $\mathcal{L}(P)$. Finally, note that the pumping lemma constant n can be computed from the description of P. For example, we may take $n = b^{|V|+2}$, where V is the set of non-terminals and b is the maximum number of symbols on the right side of a rule in a CFG equivalent to P.

(d) $\mathrm{MOVE}_{\text{TM},\alpha} = \{\langle M, \alpha\rangle \mid M \text{ is a TM that makes at least ten moves on input } \alpha\}$.

Solution Simulate M on α for at most ten moves. If M halts before ten moves, *reject*, else *accept*.

(e) $\mathrm{MOVE}_{\text{TM},n} = \{\langle M, n\rangle \mid M \text{ is a TM that makes at least } n \text{ moves on some input}\}$. (Hint: First argue that it suffices to restrict attention only to input strings of length $\leqslant n$.)

Solution In n moves a TM M can scan at most n cells starting from the left end. So irrespective of what the length of the input string is, M makes at least n moves if and only if it does so on a string of length $\leqslant n$. So simulate M for at most n steps on each input string α of length $\leqslant n$. If any string of length $\leqslant n$ is found on which M does not halt before making n moves, then *accept*, else *reject*.

4. Consider the language

$$L_4 = \{\langle M \rangle \mid M \text{ is a TM which halts on the input } 01011\}.$$

Prove the following assertions:

(a) L_4 is Turing-recognizable.

Solution Simulate M on 01011. If M halts (after accepting or rejecting), then *accept*. If M does not halt on 01011, then the simulation does not stop and so $\langle M \rangle$ is anyway not accepted.

(b) L_4 is not Turing-decidable.

Solution Let us reduce A_{TM} to L_4, i.e., we convert $\langle M, \alpha \rangle$ to $\langle M' \rangle$ such that M' halts on 01011 if and only if M accepts α. Here is a description of M'.

Input: β.

Steps

if $\beta \neq 01011$, then *halt* (after accepting β).
if $\beta = 01011$,
    simulate M on 01011.
    if M accepts 01011 (and hence halts), then *halt* (after accepting β).
    if M rejects 01011 after halting, then go to an infinite loop.

It follows that M' halts on every input other than 01011. If the input is 01011, then there are three possibilities: M accepts α (after halting), M rejects α after halting, M goes to an infinite loop on α (and hence implicitly rejects α). Only in the first case, M' halts on 01011. In the second case, M' enters a forced infinite loop. In the third case, the simulation of M on α by M' never terminates.

(c) $\overline{L_4}$ is not Turing-recognizable.

Solution If L_4 were Turing-recognizable, then Part (a) would imply that L_4 is Turing-decidable, a contradiction to Part (b).

5. Consider the language

$$L_5 = \{\langle M \rangle \mid M \text{ is a TM which halts on every input}\}.$$

(a) Use a reduction from $\overline{L_4}$ to L_5 to prove that L_5 is not Turing-recognizable. (Hint: Suppose that $\langle M \rangle$ maps to $\langle M' \rangle$ under the reduction. Let M' simulate M on input 01011 for n steps, where n is the length of the input string for M'.)

Solution I propose a reduction from $\overline{L_4}$ to L_5 that maps $\langle M \rangle$ to $\langle M' \rangle$ such that M' halts on every input if and only if M does not halt on 01011. Here is a description of M'.

Input: β.

Steps

determine the length n of the input β.
simulate M on 01011 for exactly n steps.
if the simulation halts (after accepting or rejecting 01011) within n steps, enter an infinite loop,
else stop the simulation and *halt* (after accepting or rejecting β).

If M does not halt on 01011, then irrespective of the length n of β, the simulation of M on 01011 for n steps does not reach a halting configuration. In this case, M' simply halts after aborting the simulation. On the other hand, if M halts on 01011 after the m-th step (for $m < \infty$), then for any input β of length $n \geqslant m$, M' enters an infinite loop and fails to halt.

Since $\overline{L_4}$ is not Turing-recognizable (Exercise 4(c)), it follows that L_5 is also not Turing-recognizable.

(b) Use a reduction from $\overline{L_4}$ to $\overline{L_5}$ to prove that $\overline{L_5}$ is also not Turing-recognizable.

Solution Let me now describe a reduction from $\overline{L_4}$ to $\overline{L_5}$ that maps $\langle M \rangle$ to $\langle M' \rangle$ such that M' does not halt on some input string β if and only if M does not halt on 01011. It is natural to take $\beta = 01011$, so that M' can simply simulate M on input 01011. A description of M' now follows:

Input: β.

Steps

if $\beta \neq 01011$, *halt* (after accepting or rejecting β).
if $\beta = 01011$, then
    simulate M on 01011.
    if the simulation halts, *halt* (after accepting or rejecting β).

Evidently, M' halts on every input other than 01011. On the other hand, M' halts on the input 01011 if and only if M does so on the same input. Thus the reduction is as desired.

Finally, since $\overline{L_4}$ is not Turing-recognizable, it follows that $\overline{L_5}$ too is not Turing-recognizable.