CS 373: Theory of Computation
Sariel Har-Peled and Madhusudan Parthasarathy

Discussion 15: Reductions

6 May 2009

Questions on homework ?

Any questions? Complaints, etc?

1 Problems

(Q1) If B is regular (or CFL), and $A \subseteq B$, can we deduce that B is regular (or CFL)?

Solution:

No, every language is a subset of Σ^* which is regular. More interesting sample is:

$$L_1 = \{a^n b^n c^n : n \geq 0\} \subseteq L_2 = \{a^n b^n c^k : n, k \geq 0\} \subseteq L_3 = L(a^* b^* c^*)$$

L_3 is regular, L_2 is not regular but is a CFL and L_1 is not a CFL.

(Q2) Give a direct argument why $L = \{ww : w \in \Sigma^*\}$ is not regular.

Solution:

Consider the infinite set of strings $\{a^n b : n \geq 0\}$. No two of these strings, if fed to a DFA that recognizes L, will land in the same state, because otherwise assume for $i \neq j$, $\delta^*(q_0, a^i b) = \delta^*(q_0, a^j b) = q$. Then we will have:

$$\delta^*(q_0, a^i b a^j b) = \delta^*(\delta^*(q_0, a^i b), a^j b) = \delta^*(q, a^j b) \notin F$$

$$\delta^*(q_0, a^j b a^j b) = \delta^*(\delta^*(q_0, a^j b), a^j b) = \delta^*(q, a^j b) \in F$$

Which is a contradiction.

But this is impossible since a DFA has just a finite number of states.

(Q3) Show that $L = \{\langle G_1, G_2, k\rangle : \exists w, |w| \leq k, w \in L(G_1) \cap L(G_2), G_1 \text{ and } G_2 \text{ are grammars.}\}$.

Solution:

For a given G_1, G_2 and k, the TM produces all strings w of length at most k and checks whether each string is in both $L(G_1)$ and $L(G_2)$ (first we convert the grammar into CNF and then we use CYK algorithm).

(Q4) Show that $L = \{\langle G_1, G_2\rangle : L(G_1) - L(G_2) = \varnothing\}$ (G_1 and G_2 are grammars) is undecidable.

Solution:

We can reduce $EQ_{CFG} = \{\langle G_1, G_2\rangle : L(G_1) = L(G_2)\}$ to L. If we have a decider for L, we can build a decider for EQ_{CFG} by querying it once for $\langle G_1, G_2\rangle$ and once for $\langle G_2, G_1\rangle$. (note that for any two sets A and B: $A = B \iff A - B = B - A = \varnothing$.)

(Q5) Show that $L = \{\langle R, w\rangle : \ R$ is an RA and $w \in \Sigma^*\}$ is decidable.

Solution:

We can design a decider like this: the decider converts R into a CFG G (remember we have seen the algorithm for this) and then converts G into CNF (remember the algorithm), and finally using CYK it detects if $w \in L(G)$ or not.

(Q6) Assume we have a language L and a TM M with this property: for every string w with length at least 2, $M(w)$ halts and outputs k strings $w_1, \cdots, w_k$ where $|w_i| < |w|$ for all i (k is not a constant and depends on string w). We know that $w \in L$ iff for all i, $w_i \in L$.

Assuming that $0 \in L, 1 \notin L, \epsilon \notin L$, design a decider for set L (you can use M as a subroutine) and prove that your decider works.

Solution:

Algorithm $Decider_L(w)$
1. **if** $|w| \leq 1$
2. **then if** $w = 0$ **return** Yes
3. **else return** No
4. **else** $w_1, \cdots, w_k \leftarrow M(w)$
5. **for** all i
6. **do if** $Decider_L(w_i) = No$ **return** No
7. **return** Yes

We prove by induction on the length of w that $Decider_L(w) = Yes \iff w \in L$.

For the base case, when $|w| \leq 1$, it is easy to see that lines 1,2,3 of the algorithm, return the correct result immediately. If $|w| > 1$ (inductive step), the algorithm returns true iff $Decider_L(w_i) = Yes$ for all i. But by induction hypothesis this happens exactly when $M(w_i) = Yes$ for all i. By the property of machine M, this happens iff $w \in L$, which is the desired result.