CS 301: Languages and Automata **Fall 2015**

Problem Set #4

Due: Friday, December 4, 2015

Problem 1 Given an alphabet Σ, define an operation f on languages over Σ by

$$f(A, B) = \{x_1 y_1 x_2 y_2 \cdots x_n y_n \mid x_i, y_i \in \Sigma,\ x_1 x_2 \cdots x_n \in A, \text{ and } y_1 y_2 \cdots y_n \in B\}.$$

For example, given $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, consider

$$P = \{w \in \Sigma^* \mid w = w^{\mathcal{R}}\},$$
$$L = \{\mathtt{a}^n \mathtt{b}^n \mid n \geqslant 0\}.$$

Then $w_1 = \mathtt{aabaaaabbbab} \in f(P, L)$ because $\mathtt{abaaba} \in P$ and $\mathtt{aaabbb} \in L$.[1]

Prove that regular languages are closed under f. That is, given arbitrary regular languages A and B, $f(A, B)$ is regular. [*Hint: Let* $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ *and* $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ *be DFAs such that* $L(M_1) = A$ *and* $L(M_2) = B$ *and construct a new DFA* $M = (Q, \Sigma, \delta, q_0, F)$ *such that* $L(M) = f(A, B)$. *You will probably want* $Q = Q_1 \times Q_2 \times \{1, 2\}$.]

Problem 2 Similar to problem 1, given an alphabet Σ, define an operation g on languages over Σ by

$$g(A, B) = \{x_1 y_1 x_2 y_2 \cdots x_n y_n \mid x_i, y_i \in \Sigma^*,\ x_1 x_2 \cdots x_n \in A, \text{ and } y_1 y_2 \cdots y_n \in B\}.$$

Note carefully the difference between f and g. Namely each x_i and y_i are in Σ^* rather than Σ. For example, given P and L as in problem 1, we have $w_2 = \mathtt{ababbbba}\varepsilon \in g(P, L)$ because $\mathtt{abbbba} \in P$ and $\mathtt{ab} \in L$ but $w_2 \notin f(P, L)$.

Prove that regular languages are closed under g. [*Hint: Adapt the hint for problem 1 except the constructed machine should be an NFA* $N = (Q, \Sigma, \delta, q_0, F)$ *rather than a DFA. To keep things simple, for every state* $q \in Q$ *and* $t \in \Sigma$, $\delta(q, t)$ *should be a set containing exactly one state and* $\delta(q, \varepsilon)$ *should be a set also containing exactly one state.*]

Problem 3 Similar to problems 1 and 2, given an alphabet Σ, define an operation h on languages over Σ by

$$h(A, B) = \big\{x_1 y_1 x_2 y_2 \cdots x_n y_n \;\big|\; x_i, y_i \in \Sigma^*,\ x_1 x_2 \cdots x_n \in A,\ y_1 y_2 \cdots y_n \in B, \text{ and } |x_1 x_2 \cdots x_n| = |y_1 y_2 \cdots y_n|\big\}.$$

[1] The colors are just to show the correspondence with the definition of f. The red are the x_i and the blue are the y_i.

Here, notice the difference between h and g, namely that the constituents strings in A and B have the same length. Thus continuing the example from problems 1 and 2, $w_2 \in g(A, B)$ but $w_2 \notin h(A, B)$.[2] In contrast, $w_1 \in h(A, B)$ as is $w_3 = \varepsilon\texttt{aababbabbbbab}\varepsilon$ since $\texttt{babbab} \in P$, $\texttt{aaabbb} \in L$, and both have length 6. However, $w_3 \notin f(A, B)$ because $\texttt{abbaba} \notin P$. (Note that $f(A, B) \subset h(A, B) \subset g(A, B)$ and that for most languages both inclusions are strict, meaning there are strings in $g(A, B)$ that are not in $h(A, B)$ and strings in $h(A, B)$ that are not in $f(A, B)$. This fact is not important to this problem, but it's interesting.)

Prove that if A and B are regular languages, then $h(A, B)$ is context-free. [*Hint: Modify your construction from problem 2 such that, given two DFAs M_1 and M_2, you construct a PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$. Use the stack to keep track of the difference in lengths. It may help to review the construction of the PDA for the language $\{w \in \{\texttt{a}, \texttt{b}\}^* \mid w$ has the same number of* `a`*s as* `b`*s$\}$.*]

Problem 4 If A is context-free and B is regular, we know that $A \setminus B$ is context-free but that $B \setminus A$ may not be context-free. Nevertheless, $B \setminus A$ is decidable. Prove that fact. [*Build a decider or apply closure properties.*]

Problem 5 You've just graduated from UIC and landed a job at Google. On your first day of work, your new boss complains that Google has thousands of applications that it relies on but that from time to time, the applications crash which users don't like. Your boss tells you that your first assignment is to build a program that can analyze the source code of an application and output whether or not the application may crash. You think back to CS 301 and realize that such a task is impossible. Prove it by formulating the problem as a language and using a mapping reduction.

Problem 6 Define $ALL_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \Sigma^*\}$. Prove that ALL_{TM} is *not* co-Turing recognizable (coRE).

Problem 7 It is a fact (that we did not prove) that for every regular language, there is a unique, minimal[3] DFA. The uniqueness is up to renaming of states. I.e., the state diagrams are the same, but the states may have different names. This gives rise to a decider for the language EQ_{DFA}: convert both input DFAs to their unique minimal DFA and check if they are the same by trying all permutations of state names.

There is no analogous statement for context-free languages. For example, consider Chomsky-normal form (CNF). We know that CNF is not unique for a given CFL. Even converting a single CFG to CNF using two different orders of rules can give rise to very different CFGs. In fact, there are many different "normal forms" a CFG can be in.[4] However, none are unique. Prove this fact. [*Hint: Assume for contradiction*

[2] This takes a little work to see. But since $|w| = 8$, the string in L must be $\texttt{aabb}$ which means the string in P must be $\texttt{bbba} \notin P$, a contradiction.

[3] Minimal in the sense that it has the minimal number of states.

[4] See `http://ddi.cs.uni-potsdam.de/InformaticaDidactica/LangeLeiss2009.pdf` for some, if you're really curious.

that there is a procedure P that on input $\langle G \rangle$ outputs $\langle G_{normal} \rangle$ such that if G and G' are CFGs such that $L(G) = L(G')$, then $G_{normal} = G'_{normal}$ and if $L(G) \neq L(G')$, then $G_{normal} \neq G'_{normal}$. Derive a contradiction.]