CSE 103 Computational Models

Fall 2021 Prof. Fremont

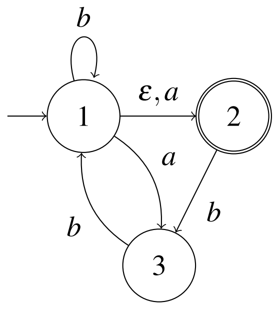
HW 3

Due October 20 at 11:59pm

(6 questions, 230 points total)

1. (40 pts.) Conversion of NFAs to DFAs

Consider the following NFA N over the alphabet $\Sigma = \{a, b\}$:



(a) <u>(30 pts.)</u> Use the construction we described in class to build a DFA D equivalent to N. Draw D as a graph, omitting any states which are not reachable from the start state.

(b) <u>(10 pts.)</u> Even dropping unreachable states, D is not the smallest DFA equivalent to N. Draw another DFA D' which is equivalent to N and has as few states as possible (you don't need to prove this).

2. (45 pts.) Intersecting automata

We showed in class that if L_1 and L_2 are regular languages, then so is $L_1 \cap L_2$, since $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ and regular languages are closed under union and complement.

(a) (10 pts.) Suppose we have DFAs D_1 and D_2 for L_1 and L_2 with n and m states respectively. Describe the sequence of transformations you would use to turn these into a DFA for $\overline{\overline{L_1} \cup \overline{L_2}}$ (and so for $L_1 \cap L_2$), stating after each step how many states the resulting automaton has.

(b) (35 pts.) A more direct way to build a DFA for $L_1 \cap L_2$ is the product construction. The idea is to run the DFAs for L_1 and L_2 in parallel: each time we read an input symbol, we update the states of both automata, and we accept only if both automata end in an accepting state. To simulate running both automata in parallel, we use a DFA whose states are labeled by pairs of states, one from each automaton; then the transitions are defined by updating each component of the pair independently. Formally, starting with DFAs $D_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $D_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$, the product automaton D is defined as follows:

- $Q = Q_1 \times Q_2$ (all pairs with a state from Q_1 and a state from Q_2)
- $\delta((q_1, q_2), s) = (\delta_1(q_1, s), \delta_2(q_2, s))$
- $q_0 = (q_{01}, q_{02})$
- $F = \{(q_1, q_2) \in Q \mid q_1 \in F_1 \text{ and } q_2 \in F_2\}$.

Prove that the product construction works: $L(D) = L(D_1) \cap L(D_2)$.
(Hint: Compare the extended transition functions of D, D_1, and D_2.)

3. (35 pts.) Reading regular expressions

For each of the following pairs of regular expressions over $\Sigma = \{a, b, c\}$, state whether their languages are equal (=), one is a proper subset of the other ($\subset$), or if they are incomparable. If $L(R_1) \subset L(R_2)$, give an example of a string in $L(R_2)$ that is not in $L(R_1)$, or vice versa if $L(R_2) \subset L(R_1)$; if the languages are incomparable, give an example string from each language that is not in the other.

(a) $R_1 = a^*b^*$
$R_2 = (a^*b^*)^*$

(b) $R_1 = (a|b|c)^*$
$R_2 = (a^*b^*c^*)^*$

(c) $R_1 = c^*c$
$R_2 = (cc)^* \mid c(cc)^*$

(d) $R_1 = c^*(a|b|\varepsilon)c^*$
$R_2 = c^*ac^* \mid c^*bc^*$

(e) $R_1 = \emptyset b \mid aa^*$
$R_2 = a^*$

4. (45 pts.) Writing regular expressions

Write regular expressions for the following languages over $\Sigma = \{a, b, c, \ldots, z\}$:

(a) Strings containing the substring *dog* followed eventually (not necessarily immediately) by either the substring *cat* or the substring *rat*.

(b) Strings containing both x and y, where the first x is followed by a y within 3 symbols (e.g. *xy* and *xzzy* are fine but not *xzzzy*).

(c) Strings containing an even number of zs.

5. (20 pts.) Converting a regular expression to an NFA

Convert the regular expression $(a \mid c\Sigma)^*a$ to an equivalent NFA.

6. (45 pts.) Properties of regular languages

Prove each of the following facts about regular languages. Give a short argument using results from class and/or simple constructions of automata or regular expressions (no need for proofs by induction).

(a) Every finite language (i.e., a language consisting of finitely many strings) is regular.

(b) If I, T, and E are regular languages over Σ, then so is the language

$$\textsc{If-Then-Else}(I,T,E) = \{xy \in \Sigma^* \mid x \in I \to y \in T \text{ and } x \notin I \to y \in E\}.$$

(c) If $s \in \Sigma$, and L and R are regular languages over Σ, then so is the language $\textsc{Replace}(L,s,R)$ obtained by taking every string in L and replacing each occurrence of the symbol s by any string in R.
(For example, if $L = \{a, aa, aba\}$, $s = a$, and $R = \{c, dd\}$, then $\textsc{Replace}(L,s,R)$ is $\{c, dd, cc, cdd, ddc, dddd, cbc, cbdd, ddbc, ddbdd\}$.)