CSE 311: Foundations of Computing I

Homework Final Assignment (due Wednesday, March 18 at 11:00 PM)

1. Set and Match (20 points)

(a) [7 Points] Prove that for all sets X, Y, and Z, if $X \cup Z = Y \cup Z$ then $X \setminus Z = Y \setminus Z$.

(b) [7 Points] Prove that for all sets X and Z, $X = (X \setminus Z) \cup (X \cap Z)$.

(c) [6 Points] Combine the results of parts (a) and (b) to show that for all sets X, Y, and Z, if $X \cup Z = Y \cup Z$ and $X \cap Z = Y \cap Z$ then $X = Y$.

2. Break me off a piece! (25 points)

Assume that a chocolate bar consists of n squares arranged in a rectangular pattern as in the example shown below. (n is not necessarily equal to 24.) In any one step, you can break the whole bar, or any one piece you have formed, along a single horizontal or vertical line that divides the squares.

(a) [5 Points] Determine how many breaks (as a function of n) that you must make in order to break the bar into n separate squares.

(b) [20 Points] Use strong induction to prove your answer correct.

3. Exclusively Not (25 points)

Let $\mathcal{A} = \{\ldots, p, q, r, \ldots\}$ be a fixed set of atomic propositions. We then define the set $\mathbf{Prop}$ as follows:

Basis Elements For any $p \in \mathcal{A}$, $\texttt{Atomic}(p) \in \mathbf{Prop}$.

Recursive Step If $A, B \in \mathbf{Prop}$, then $\mathsf{NOT}(A) \in \mathbf{Prop}$ and $\mathsf{XOR}(A, B) \in \mathbf{Prop}$.

The set $\mathbf{Prop}$ represents parse trees of propositions that use only the operations of negation (represented by NOT) and exclusive or (represented by XOR).

Next, we define a function $\mathcal{T}$ that takes a parse tree (an element of $\mathbf{Prop}$) as input and returns the proposition that it represents. Formally, we define

$$\begin{aligned}
\mathcal{T}(\mathsf{Atomic}(p)) &= p && \text{for any } p \in \mathcal{A} \\
\mathcal{T}(\mathsf{NOT}(A)) &= \neg\mathcal{T}(A) && \text{for any } A \in \mathbf{Prop} \\
\mathcal{T}(\mathsf{XOR}(A, B)) &= (\mathcal{T}(A)) \oplus (\mathcal{T}(B)) && \text{for any } A, B \in \mathbf{Prop}
\end{aligned}$$

Now, let $p \in \mathcal{A}$ be any atomic proposition. The function neg_p takes a parse tree as input and returns another parse tree with all nodes representing p replaced by nodes representing $\neg p$ instead:

$$\begin{aligned}
\mathsf{neg}_p(\mathsf{Atomic}(q)) &= \mathsf{NOT}(\mathsf{Atomic}(q)) && \text{if } q = p \\
\mathsf{neg}_p(\mathsf{Atomic}(q)) &= \mathsf{Atomic}(q) && \text{for any } q \in \mathcal{A} \text{ with } q \neq p \\
\mathsf{neg}_p(\mathsf{NOT}(A)) &= \mathsf{NOT}(\mathsf{neg}_p(A)) && \text{for any } A \in \mathbf{Prop} \\
\mathsf{neg}_p(\mathsf{XOR}(A, B)) &= \mathsf{XOR}(\mathsf{neg}_p(A), \mathsf{neg}_p(B)) && \text{for any } A, B \in \mathbf{Prop}
\end{aligned}$$

(a) [15 Points] Prove the following: for any $A \in$ **Prop** and any $p \in \mathcal{A}$, we have either $\mathcal{T}(\mathsf{neg}_p(A)) \equiv \mathcal{T}(A)$ or $\mathcal{T}(\mathsf{neg}_p(A)) \equiv \neg\mathcal{T}(A)$.

You may find it useful to use, without proof, the following easy-to-verify fact: $(\neg p) \oplus q \equiv \neg(p \oplus q) \equiv p \oplus (\neg q)$.

(b) [5 Points] Let $A \in$ **Prop** be an expression using only the atomic variables $p, q \in \mathcal{A}$. Describe what the truth table of $\mathcal{T}(A)$ must look like in the case that $\mathcal{T}(\mathsf{neg}_p(A)) \equiv \mathcal{T}(A)$? How about if $\mathcal{T}(\mathsf{neg}_p(A)) \equiv \neg\mathcal{T}(A)$?

(c) [5 Points] We saw in lecture that $\neg$, $\vee$, and $\wedge$ together can express any proposition (e.g., by writing it in sum-of-products or product-of-sums form). De Morgan's Law tells us that $\wedge$ can be written instead with $\vee$, so any proposition can be expressed with only $\neg$ and $\vee$. Prove that the same is not true of $\neg$ and $\oplus$.

Specifically, use the results of the earlier parts to show that $p \wedge q$ cannot be represented by any expression using only $\neg$ and $\oplus$.

4. Relatively Natural (20 points)

Define relation D on $\mathbb{N}$ by $(a, b) \in D$ iff a divides b and relations E_m on $\mathbb{N}$ for each $m \in \mathbb{N} \setminus \{0\}$ by $(a, b) \in E_m$ iff $a \equiv b \pmod{m}$. Prove or disprove each of the following for all the values where the relations are defined.

(a) [5 Points] If $(a, b) \in E_m$ and $(n, m) \in D$ then $(a, b) \in E_n$.

(b) [5 Points] If $(a, b) \in E_m$ and $(m, n) \in D$ then $(a, b) \in E_n$.

(c) [5 Points] $D \circ D \subseteq D$.

(d) [5 Points] If $(a, c) \in D$ and $(b, d) \in D$ then $(a, b) \in E_m$ implies $(c, d) \in E_m$.

5. Stringing it together (15 points)

(a) [7 Points] Give a regular expression for the set of all binary strings that end in 101 or have an odd number of 0s.

(b) [8 Points] Give a context-free grammar that generates the set of all binary strings of the form $0^n 1^m$ where $m \neq n$.

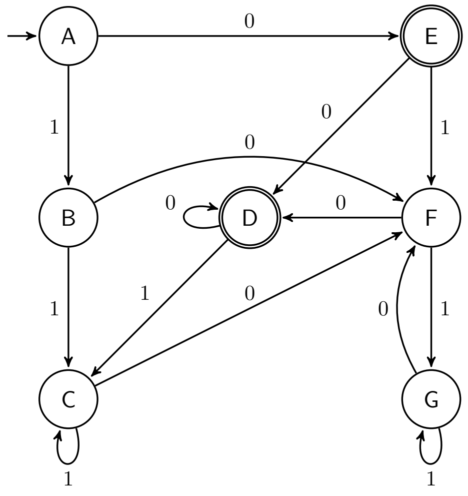
6. State of the Art (10 points)

Draw a DFA that recognizes the set of all binary strings that contain 010 and have an even number of 1s. Document your DFA by describing, for each state, the set of input strings that end at each state.

7. Stateless (15 points)

Use the algorithm for minimization that we discussed in class to minimize the following automaton.

For each step of the algorithm write down the groups of states, which group was split in the step and the reason for splitting that group. At the end, write down the minimized DFA, with each state named by the set of states of the original machine that it represents (e.g., "B, C" if it represents B and C).



8. To Infinity and Beyond (20 points)

Use the method described in lecture to prove that the following language is **not regular**.

The set of binary strings of the form $\{0^x 1^m 0^y \mid x, m, y > 1 \text{ and } x \equiv y \pmod{m}\}$.