Staff Solutions to Mini-Quiz 4

Problem 1 (4 points).
Start with 110 coins on a table, 10 showing heads and 100 showing tails.

There are two ways to change the coins:

(i) Remove 20 coins from the table, 10 of which must be heads and the other 10 must be tails, or

(ii) Let \( n \) be the number of heads showing. If there are more tails than heads on the table, place \( n \) additional coins, all showing heads, on the table.

(a) Model this situation as a state machine, carefully defining the set of states, the start state and the possible state transitions. *Hint:* Be sure to state the conditions of the state transitions.

**Solution.** States are tuples of the form \((H, T)\) where \( H \geq 0 \) and \( T \geq 0 \). The start state is \((10, 100)\). The transitions are of the form \((H, T) \rightarrow (2H, T)\) with the restriction of \( T > H \), and \((H, T) \rightarrow (H - 10, T - 10)\) with the restriction of \( H \geq 10 \) and \( T \geq 10 \).

(b) Let \( H \) := the number of heads and \( T \) := the number of tails. For each of the derived variables below, indicate the strongest of the following properties that it satisfies: constant \( C \), strictly increasing \( Sinc \), strictly decreasing \( Sdec \), weakly increasing \( Winc \), weakly decreasing \( Wdec \), none of these \( N \).

1. \( T \)
2. \( H + T \)
3. \( T - H \)
4. \( 2T - H \)

**Solution.**
1. \( T \): weakly decreasing
2. \( H + T \): none
3. \( T - H \): weakly decreasing
4. \( 2T - H \): strictly decreasing

Problem 2 (6 points).
The set, \( M \), of strings of brackets is recursively defined as follows:

**Base case:** \( \lambda \in M \).

**Constructor cases:** If \( s, t \in M \), then

- \([s] \in M\), and
- \( s \cdot t \in M\).
The set, RecMatch, of strings of matched brackets was defined recursively in class. Recall the definition:

**Base case:** \( \lambda \in \text{RecMatch} \).

**Constructor case:** If \( s, t \in \text{RecMatch} \), then \([s]t \in \text{RecMatch} \).

Fill in the following parts of a proof by structural induction that

\[
\text{RecMatch} \subseteq M. \tag{1}
\]

(a) State an **induction hypothesis** suitable for proving (1) by structural induction.

**Solution.**

\[
P(x) ::= x \in M
\]

(b) State and prove the **base case(s).**

**Solution.** **Base case** \((x = \lambda)\): By definition of \(M\), the empty string is in \(M\).

(c) Prove the **inductive step.**

**Solution.** **Proof.** **Constructor case** \((x = [s]t \text{ for } s, t \in \text{RecMatch})\): By structural induction hypothesis, we may assume that \(s, t \in M\). By the first constructor case of \(M\), it follows that \([s] \in M\). Then, by the second constructor case of \(M\), it follows that \([s]t \in M\).

As a matter of fact, \(M = \text{RecMatch}\), though we won’t prove this. An advantage of the RecMatch definition is that it is **unambiguous**, while the definition of \(M\) is ambiguous.

(d) Give an example demonstrating that \(M\) is ambiguously defined.

**Solution.** Consider derivations of the empty string. This could be derived directly from the base case \(\lambda\), or by starting with \(\lambda\) and then constructing \(\lambda\lambda\) through the second constructor case.