Problem Set 7: CS103

Katherine Cheng, Richard Davis, Marty Keil

May 29, 2015

Problem 1

Problem 2

Problem 3

i

Proof. Let the turing machine M reject any input. It is then a decider, because M halts on all inputs by rejecting. Also for any string w, the statement if M accepts w, then x $\in \Sigma^*$ is always a true statement because M never accepts w. A truth table is always true when the antecedent of an implication is false. □

ii

Proof. Let the turing machine M accept any input. It is then a decider, because M halts on all inputs by accepting. Also for any string w, the statement if M rejects w, then x $\notin \Sigma^*$ is always a true statement because M accepts all w. A truth table is always true when the consequent of an implication is true. □

iii

Proof. Let the turing machine M infinitely loop on any input. Both Statements 2 and 3 are always true in this case because both antecedents in the implications are always false. M never accepts any string w and M never rejects any string w. As before, the truth table is always true when the antecedent of an implication is false. □

iv If L satisfies all 3 statements then we know the language is decidable. We then also know that L $\in$ R. And since R is a subset of RE, we can also state that L is recognizable.

Problem 4

Problem 5

Problem 6

Problem 7

Problem 8

i
```
function bool inL1uL2(string w) {
    inL1(w) OR inL2(w);
}
```
From this new method we can see that L1 ∪ L2 is also decidable. When either L1(w) or L2(w) is accepted, inL1uL2 returns true. If both L1(w) and L2(w) are rejected then inL1uL2 returns false. This covers all possibilities and always returns either an accepting or rejecting state, so the method is therefore decidable.

ii
```
function bool Concat (w) {
    for all L1 {
        for all L2 {
            for ( i = 0; i++; i < w.length) {
                'if (L1(w.split AHHH!! ) {
                    return true;
}
```

iii
```
function bool SymDiff (w) {
    For All L1, L2
        If (inL1(w) OR inL2(w)) {
            If( !( inL1(w) AND inL2(w)) {
                return true;
            }
        else return false;
        }
    else return false;
    }
```
This method returns true whenever inL1 or inL2 is accepts, but not when inL1 and inL2 both accept. In this

case and also when both reject, the method rejects. This covers all possibilities and always returns either an accepting or rejecting state, so the method is therefore decidable.