Homework 6

CSC 445-01: Theory of Computation

Matthew Mabrey, Luke Kurlandski

April 27, 2021

4.3

A DFA will recognize Σ^* iff every reachable state is a final state. In a similar fashion to Theorem 4.4, we design the TM T to test whether or not this is the case and decide ALL_{DFA}.

T = "On input A, where A is a DFA:

1. Mark A's start state
2. Do until no new state is marked:
   (a) Mark any state that can be reached via the transition function from a marked state
3. If every marked state is a final state, then *accept*; Else any marked state is not a final state, then *reject*."

4.4

Construct a TM S that will decide $A\varepsilon_{CFG} = \{\langle G \rangle \mid G \text{ is a CFG that generates } \varepsilon\}$. Inside S the machine will:

1. Convert input CFG $\langle G \rangle$ to a CNF
2. If the new CNF grammar's start state has a transition $S_0 \rightarrow \epsilon$, then accept
3. Else, reject

4.11

We construct a TM I to decide INFINTE$_{PDA}$.

I = "On input M, where $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is a PDA that recognizes $L(P)$:

1. Construct a Context Free Grammar for $L(P)$, $G' = (V', \Sigma', R', S')$
2. Convert G' into Chomsky Normal Form grammar, $G = (V, \Sigma, R, S)$
3. Conduct a search for recursion in the rules, R
   - Use a BFS to prevent "getting stuck", as opposed to DFS
   - For some arbitrary non terminal $A \in V$, if the derivation $A \rightarrow uAv$ exists, for $u, v \in \{V \cup \Sigma\}^*$, then recursion exists in the rules
4. If recursion is found *accept*; Else *reject*

Since we can create a TM that decides this language, it is obviously a decideable language.

5.1

First we define

$$EQ_{CFG} = \{< G_1, G_2 > | G_1 \text{ and } G_2 \text{ are equivalent context free grammars}\}$$
$$ALL_{CFG} = \{< G > | G \text{ is a CFG and } L(G) = \Sigma^*\}$$

We will use a proof by contradiction to show that EQ_{CFG} is undecidable.

Suppose that EQ_{CFG} were decidable by some TM, R. Then we could use R to construct a TM, S, that decides ALL_{CFG}. We describe S in the following paragraph.

On input G, where G is a CFG:

1. Run $< G, G_{\Sigma^*} >$ on R, where $L(G_{\Sigma^*}) = \Sigma^*$
2. *accept* if R accepts; Else *reject*

In summary, machine S uses machine R to compare an input grammar, G, to a grammar whose language is Σ^*. The result of R's computation then determines if $L(G) = \Sigma^*$.

However, we know from Theorem 5.13 that ALL_{CFG} is undecidable. Therefore, we have a contradiction and EQ_{CFG} cannot be decidable.

5.4

No.

We revisit the definition of mapping reducibility. If $A \leq_m B$, then there is a computable function f where

$$w \in A \text{ iff } f(w) \in B$$

f is defined such that some Turing Machine, on input w, halts with the output $f(w)$ on its tape.

However, just because the function $f(w)$ produces strings that belong to the regular language B does not nessecitate that the input strings $w \in A$ form a regular language themselves. So A does not need to be regular for B to be regular.

For example, consider the following:

- $A = \{0^n 1^n \mid n \geq 0\}$
- $B = \{0^n \mid n \geq 0\}$
- $f(x) =$ is computed by a TM M that when it reads a 1, it deletes it. M continues until the end of the input string is reached, then halts.

A is not regular, B is regular and $A \leq_m B$

5.9

First we define

$$T = \{< M > | M \text{ is a TM that accepts } w^R \text{ whenever it accepts } w\}$$
$$A_{TM} = \{< M, w > | M \text{ is TM and accepts } w\}$$

1. Construct a TM T that decides $A = \{\langle M \rangle \mid w^R \text{ is accepted if w is accepted}\}$ inside a TM S used to decide A_{TM}.
2. Inside S, construct a new TM M_1 from input $\langle\langle M \rangle, w\rangle$. Using input w_1:
   - If $w_1 = w^R$, Accept
   - Else if $w_1 = w$, Run M on w_1
   - Else, Reject
3. Feed M_1 into TM T, if it accepts then M must accept w since T only accepts M if both w^R and w are accepted and we can guarantee w^R is accepted because we constructed M_1 to accept it.

Thus we can use TM T to decide A_{TM}, which is undecidable, so T must also be undecidable.

5.22

We will prove

$$\text{A is Turing-recognizable} \leftrightarrow A \leq_m A_{TM}$$

forward

We will prove

$$\text{A is Turing-recognizable} \rightarrow A \leq_m A_{TM}$$

If A is Turing recognizable, then some TM T_A recognizes it. We design a TM T that writes the concatenation of a word, w, and T_A on its tape. We describe the TM T:

On input w:

1. Write $< T_A, w >$ on the tape and halt

The Turing machine T is a computable function because on every input w, T will halt with just $f(w)$ on its tape. The language A is then mapping reducible to A_{TM} because there is a computable function where for every w, $w \in A \implies f(w) \in A_{TM}$. This fulfills the mapping reduction from A to A_{TM}.

backward

We will prove

$$A \leq_m A_{TM} \rightarrow \text{A is Turing-recognizable}$$

We know from theorem 5.28 that if $A \leq_m B$ and B is Turing-recognizable, then A is Turing recognizable. A_{TM} is a Turing-recognizable language, thus A must be as well.