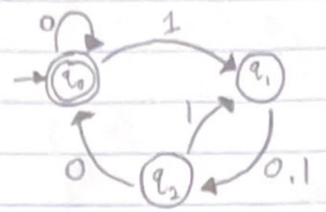
HW1

Theory of Computation

Brian DeFlaminio

DFA
M



1A. 0100 is in M via the following route: $q_0 \to q_1 \to q_2 \to q_0$ Accept

1B. 011 is not in M, $q_0 \to q_1 \to q_2$ reject

1C. $\langle M \rangle$ is not in M because there is no input string, the format is wrong so M can't accept $\langle M \rangle$

1E. Consider TM A that decides E_{DFA}

A = "On input $\langle M \rangle$ where M is a DFA
1) mark the start state of M
2) repeat until no new states are marked
3) mark any states stemming from the marked start state
4) If any marked states are accept states, reject, any marked accept states means it's not the empty language

DFA M from above rejects immediately because q_0 of M is the accept state and we mark that first. So $\langle M \rangle \notin E_{DFA}$

1F. $\langle M, M \rangle$ is EQ_{DFA} because $L(M) = L(M)$, also the derived language $L(M) \cap L(M) = \emptyset$

2A. $\Sigma = \{0, X, /\}$ and the Regular expression representing the possibilities is Σ^*. Let's pretend we have an exhaustive list of possible games, with diagonalization we can always devise a game different than every other one on the list. This means that these games have an uncountably infinite amount of unique results.

$\boxed{0}$ X / X 0 / / 0 X ...
X $\boxed{/}$ / / 0 X X / 0 ...
/ / $\boxed{X}$ X X 0 0 X X ...
⋮

New game starts with X, 0, 0 and is already different but we continue to change the i^{th} digit of the i^{th} game to create a unique one.

2B. If the games can only create one (even infinite) pattern before leveling out to just X's or O's then we have countably infinite games because the unique sequence before all X's or O's is mappable to the rational numbers. Which are countably infinite.

Σ^*'s complement is the $\emptyset$ empty set

$\uparrow$ lang of ALL_{DFA}

3. $ALL_{DFA} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \Sigma^*\}$

We prove that ALL_{DFA} is decidable by constructing DFA B that decides $\overline{L(A)}$. We then run a new DFA B' that decides E_{DFA} on $\langle B'\rangle$. If B' accepts, then accept, otherwise reject. This works by leveraging E_{DFA}'s decidability with the complement of L(A), which should be the empty set.

4. $A = \{\langle R, S\rangle \mid R \text{ and } S \text{ are DFAs and } L(R) \subseteq L(S)\}$

Show that A is decidable. We know that EQ_{DFA} is decidable, and that if S can be reduced to R, then we have it in terms we know to be decidable. We also know that $L(R) \subseteq L(S)$ iff $L(R) \cap \overline{L(S)} = \emptyset$

We construct TM Q that decides A

Q = "On input $\langle R, S\rangle$ where R & S are DFAs

1) Construct DFA D that accepts the language of $L(R) \cap \overline{L(S)}$

2) construct DFA D\` that accepts the empty language

3) Run EQ_{DFA} with input $\langle D, D'\rangle$ and if it accepts, accept, otherwise, reject.

Because $L(R) \cap \overline{L(S)} = \emptyset$ if $L(R) \subseteq L(S)$ we compare the language of $L(R) \cap \overline{L(S)}$ against the empty language and if they're the same then $L(R) \subseteq L(S)$

5. Prove EQ_{DFA} is decidable.

Let $EQ_{DFA} = \{\langle A, B\rangle \mid A, B \text{ are DFAS and } L(A) = L(B)\}$

$L(A) = L(B)$ iff A and B accept strings up to length mn where m and n are the # of states in A and B. If $L(A) \neq L(B)$ then there must be a string t that is the shortest string A and B differ on. Let L be the length of t. If $L \leq nm$ then A and B aren't equal. So nm or smaller is the sufficient size string to test if $A = B$

HW1
Theory of Computation

Brian DeFlaminio

6. $S = \{\langle M \rangle \mid M$ is a DFA that accepts w^R whenever it accepts $w\}$
Show that S is decidable.
We start by creating DFA M' where all arrows of transition functions in M are reversed, we also swap the start and final states so that $L(M') = w^R$. We then feed $\langle M, M' \rangle$ to the TM that decides EQ_{DFA}. If it accepts, accept, else reject.

7. Prove $AMBIG_{CFG}$ is undecidable.
If the PCP instance has a solution, then the CFG is ambiguous, as there exists multiple parse trees for the string $t_1[i_1]t_2[i_2]\ldots t_n[i_n] = b_1[i_1]b_2[i_2]\ldots b_n[i_n]$. In other words, the string can be generated both from the T and B production rules, hence the CFG is ambiguous.
Construction 1) $S \rightarrow T \rightarrow t_{i_1} T a_{i_1} \longrightarrow t_{i_1} t_{i_2} T a_{i_2} a_{i_1} \longrightarrow \ldots$
Construction 2) $S \rightarrow B \rightarrow b_{i_1} T a_{i_1} \longrightarrow b_{i_1} b_{i_2} T a_{i_2} a_{i_1} \longrightarrow \ldots$
If these constructions are ambiguous, then the PCP instance given in the question has a match.
As PCP is known to be undecidable, it follows that $AMBIG_{CFG}$ is undecidable as well.

8. A) Prove $OVERLAP_{CFG} = \{\langle G, H \rangle \mid G$ and H are CFGs where $L(G) \cap L(H) \neq \emptyset\}$ is undecidable
We first define the CFGs G and H and if they've got a string in common then we've reduced PCP to $Overlap_{CFG}$. As PCP is undecidable, it follows that $Overlap_{CFG}$ is undecidable if the PCP problem P has a match $t_{i_1} t_{i_2} \ldots t_{i_l} = b_{i_1} b_{i_2} \ldots b_{i_l}$ with $t_{i_1} t_{i_2} \ldots t_{i_l} a_{i_l} \ldots a_{i_2} a_{i_1} = b_{i_1} b_{i_2} \ldots b_{i_l} a_{i_l} \ldots a_{i_2} a_{i_1}$. Which is in $L(G)$ and $L(H)$ via the grammars:

G: $T \rightarrow t_1 T a_1 \mid \ldots \mid t_k T a_k \mid t_1 a_1 \mid \ldots \mid t_k a_k$
H: $B \rightarrow b_1 B a_1 \mid \ldots \mid b_k B a_k \mid b_1 a_1 \mid \ldots \mid b_k a_k$

HW1
Theory of Computation

Brian DeFlaminio

8B) $PREFIX\text{-}FREE_{CFG} = \{\langle G \rangle \mid G$ is a CFG where $L(G)$ is prefix free$\}$
We show that this is undecidable by reducing it to OVERLAP using the reduction f. Let $f(\langle G, H \rangle)$ be a CFG A which generates the lang $L(G)\# \cup L(H)\#\#$. If $x \in L(G) \cap L(H)$ then $x\#$ and $x\#\#$ are also in $L(D)$ so D is not prefix free. If that's true then y and $z \in L(D)$ where y is a proper prefix of z, which can only happen if $y = x\#$ and $z = x\#\#$ for some $x \in L(D) \cap L(H)$.

9. $T = \{\langle M \rangle \mid M$ is a TM that accepts w^R whenever it accepts $w\}$ Show that T is undecidable.
We do this by reducing A_{TM} to T. We construct the TM M' for this
M' = "On input x
1) $x \neq 01$ and $x \neq 10$ then reject
2) if $x = 01$ then accept
3) if $x = 10$ then simulate M on w. If it accepts, accept, else reject"
If $\langle M, w \rangle \in A_{TM}$ then $L(M') = \{01, 10\}$ so $\langle M' \rangle \in T$. The inverse is if $\langle M, w \rangle \notin A_{TM}$ then $L(M') = \{01\}$ so $\langle M' \rangle \notin T$. Therefore $\langle M, w \rangle \in A_{TM}$ $\leftrightarrow \langle M' \rangle \in T$

10. $MOVELEFT_{TM}$ implements the following algorithm. Simulate M on w until M moves left, M halts, or M repeats a state without moving left. If M moves left then $\langle M, w \rangle \in MOVELEFT_{TM}$ if M halts without moving left then $\langle M, w \rangle \notin MOVELEFT_{TM}$ and if M repeats a state without having moved left then $\langle M, w \rangle \notin MOVELEFT_{TM}$ because M's computation will just continue as $uvp\sqcup$, $uvvp\sqcup$, $uvvvp\sqcup$, ...
A TM can implement the aforementioned rules so MoveLeft is decidable.