Daily Proof Questions (Section 5.1) Solutions

1) Show that EQ_{CFG} = {<G_1, G_2> | G_1 and G_2 are CFGs with $L(G_1) = L(G_2)$ } is undecidable.

Solution

We know that ALL_{CFG} is undecidable, so we can use this as a starting point.

Assume to the contrary that EQ_{CFG} is decidable. Let the Turing Machine R decide membership in EQ_{CFG}. Now, we will build a Turing Machine S that decides membership in ALL_{CFG} as follows (assume an alphabet of {0, 1} for simplicity):

S(Grammar G) {

1. Create a Grammar G' = $S \rightarrow 0 \mid 1 \mid \varepsilon \mid 0S \mid 1S$, (Note: $L(G) = \Sigma^*$.)
2. Let ans = R(G, G'). Namely, run R on G and G'.
3. Return ans. If R accepts, accept G, else reject.

}

Since R can tell us if two grammars are equivalent, we can use it to determine if G produces all strings by comparing it to the grammar G', which we know produces all strings.

Since we know that ALL_{CFG} is undecidable, we must have made a mistake in the proof. The only mistake we could have made was assuming that EQ_{CFG} was decidable. Thus, it follows that EQ_{CFG} is undecidable as desired.

2) Let L = { <M> | M is a Turing Machine such that L(M) only contains even-length strings}. Prove that L is undecidable.

Solution

To the contrary, assume L is decidable. Let TM R decide membership in L. Here is how we can build a Turing Machine S to decide membership in A_{TM}:

S(Machine M, String w) {

1. Create a machine M' that automatically accepts all even length strings. If its input is of odd length, it erases its input and writes w. Then it simulates M's directions on its w. L(M') is either only even length strings, or all strings.
2. Let ans = R(M')
3. Return !ans. If R accepts, then M doesn't accept w. Alternatively, if R rejects, M must accept w.

}

The key here is that M' accepts odd lengths strings iff M accepts w. Since we've decided A_{TM}, there must be a problem with our proof. Our initial assumption must be wrong and we must have that L is undecidable.

3) Let $SS_{TM} = \{ <M_1, M_2> \mid M_1$ and M_2 are Turing Machines with $L(M_1) \subseteq L(M_2)$. }. Show that SS_{TM} is not decidable by showing that if you had a decider for SS_{TM}, you could build a decider for A_{TM}.

<u>Solution</u>

To the contrary, assume L is decidable. Let TM R decide membership in SS_{TM}. Here is how we can build a Turing Machine S to decide membership in A_{TM}:

S(Machine M, String w) {

> 1. Create a machine M' that runs like M for all inputs except w. If the input is w, M' automatically accepts.
> 2. Let ans = R(M', M)
> 3. Return ans. If R accepts, then M must accept w for L(M') to be a subset of L(M). If R rejects then M' accepts one string (w) that M doesn't.

}

The key here is that $L(M') \subseteq L(M)$ if and only if M accepts w. We have created M' to run just like M in nearly all cases, except for when the input string is w. Thus, if we know whether or not $L(M') \subseteq L(M)$, we can decide membership in A_{TM}. Since A_{TM} is undecidable, it follows that our initial assumption that SS_{TM} was decidable is incorrect. It follows that SS_{TM} is undecidable, as desired.