Programming Languages and Computation

Problem Sheet 11: The Halting Problem & Reductions

** 1. Show that if $f : U \lesssim V$ and $g : V \lesssim W$ then $g \circ f : U \lesssim W$.

Solution

By the definitions of $f : U \lesssim V$ and $g : V \lesssim W$ we have for any $n \in \mathbb{N}$ that

$$n \in U \iff f(n) \in V \iff g(f(n)) \in W$$

Hence $g \circ f$ is a reduction from U to W.

** 2. Is the set

$$\mathsf{LUCKY}_{127} = \{\ \gamma(S) \mid \text{ running } S \text{ on input 1 runs for at least 127 computational steps }\}$$

decidable? (Hint: if it is, describe a program that decides it. Think simply, write informally, and do not let the expressive poverty of While confine you.)

Solution

It is decidable. It is decided by a program which, on input $\gamma(S)$,

- Simulates a run of S on input 1.
- Counts the first 127 steps of that simulation.

If the simulation doesn't halt in a state of the form $\langle \texttt{skip}, \sigma \rangle$ before 127 steps, it returns true. Otherwise it returns false.

*** 3. Suppose we have a way of encoding every DFA M as a natural number $\delta(M) \in \mathbb{N}$.
Is the predicate

$$\mathsf{EMPTY} = \{\delta(M) \mid L(M) = \emptyset\}$$

decidable? Why, or why not? (Hint: the advice from the previous question applies here too.)

Solution

Augment whichever data structure holds your DFA states with a boolean flag that denotes whether a state is 'marked' or not.

Then:

1. 'Mark' the start state.

2. For every marked state, mark all the states to which one can take a transition.

3. Repeat step 2 as long as new states are being marked.

At the end of this process, look at whether any final state is 'marked.' If it is, there is a path to it, which spells some word $w \in \Sigma^*$; thus $w \in L(M)$, so return false. Otherwise, no path from the start state can reach a final state; thus $L(M) = \emptyset$, so return true.

** 4. (Trick question.) Is it decidable whether God exists?

Solution

God either exists, or does not. In the first case, the program that always returns Yes/True/1 correctly decides the existence of God. In the second case, the program that returns No/False/0 correctly decides the problem.

** 5. Prove that if $f : A \xrightarrow{\cong} B$ is a bijection, then so is its inverse $f^{-1} : B \to A$.

Solution

There are two ways to prove this fact. The quick one is use the characterisation of a bijection as a function that has an inverse. It suffices to notice that if the inverse of f is f^{-1}, then f^{-1} is also the inverse of f: the definition of inverses is *self-dual*.

The longer way is to prove in detail that f^{-1} is an injection and a surjection. It is an injection because $f^{-1}(b_1) = f^{-1}(b_2)$ implies that $f(f^{-1}(b_1)) = f(f^{-1}(b_2))$, which by one of the defining equation of inverses implies that $b_1 = b_2$. It is a surjection because the equation $f^{-1}(f(a)) = a$ for all $a \in A$ implies that each $a \in A$ has a preimage along f^{-1}, namely $f(a)$.

*** 6. Prove that the set

$$\mathbb{N} \to \mathbb{N} = \{f \mid f : \mathbb{N} \to \mathbb{N}\}$$

is uncountable.

Solution

Suppose that there is a bijection $k : (\mathbb{N} \to \mathbb{N}) \xrightarrow{\cong} \mathbb{N}$. Letting $f_i = k^{-1}(i)$ we have that

$$f_0, f_1, f_2, \ldots$$

is a list of every single function $\mathbb{N} \to \mathbb{N}$. We now build a function $g : \mathbb{N} \to \mathbb{N}$ that is not in this list. We define $g(i) = f_i(i) + 1$. For any i we have that $g \neq f_i$, because $g(i) = f_i(i) + 1 \neq f_i(i)$. Hence k cannot be a bijection, because $k^{-1}(j)$ is never equal to g for any j, so k^{-1} is not surjective.

**** 7. Prove that the set

$$\mathsf{ONE} = \{\, \gamma(S) \mid [\![\mathtt{S}]\!]_{\mathtt{x}}(0)\downarrow \}$$

is undecidable by reduction from HALT.

Solution

First we prove it by reduction from the halting problem. Suppose we want to decide the halting problem for a program S and input n. We construct the following program $G_{S,n}$. On input m,

1. Disregard the input m.
2. Simulate the program S on input n.
3. If that simulation terminates, output 0 (or any other number).

This code transformation is computable: we can write a program that given $\gamma(S)$ and n computes $\gamma(G_{S,n})$. Furthermore, we have that

$$\phi(\gamma(S), n) \in \mathsf{HALT} \iff [\![G_{S,n}]\!]_{\mathtt{X}}(0)\downarrow \iff \gamma(G_{S,n}) \in \mathsf{ONE}$$

Hence, if we could solve ONE we could solve HALT. So we cannot solve ONE.
