Programming Languages and Computation

Week 11: The Halting Problem & Reductions

* 1. (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.

** 2. Is the predicate

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

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

Solution

It is decidable. It is decided by a program which, on input $\ulcorner S \urcorner$,

- 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. Prove that the set

$$\mathsf{Zero} = \{ \ulcorner S \urcorner \mid [\![\mathtt{S}]\!]_{\mathtt{x}}(0) \downarrow \}$$

is semi-decidable. [Hint: As above, think simply, write informally, and do not let the syntactic poverty of While confine you.]

Solution

The set Zero is semi-decided by a program which performs the following actions: on input m,

- Decode $m = \ulcorner S \urcorner$.
- Simulate the running of S on input 0.
- If and when that terminates, check if the memory is the form $[\mathtt{x} \mapsto m]$. If it is, return 1. Otherwise, go into an infinite loop.

If $[\![\mathtt{S}]\!]_{\mathtt{x}}(0)\downarrow$ then the above simulation will terminate at some point, and our program will correctly return 1.

But if $[\![\mathtt{S}]\!]_{\mathtt{x}}(0)\uparrow$ then the simulation will either run forever or terminate in a 'rubbish' state (i.e. one with variables other than x set to a non-zero value). In the first case our program runs forever. In the second case, instruction 3 above forces our program to also run forever. So in either case our program runs forever.

Thus Zero is semi-decidable.

*** 4. Prove that if the predicates U and V are semi-decidable, then so is $U \cup V$. [Hint: use simulations.]

Solution

This was a trick question from previous week's sheets, which you now have the tools to solve.

Suppose we have a program A that computes the semi-characteristic function of U, and a program B that computes the semi-characteristic function of V. We want to build a program that computes the semi-characteristic function of $U \cup V$.

On input m,

- Set up a simulation of A on m, and of B on m.
- Alternate between running the first simulation for a finite number of computational steps (say, 42), and then running the second simulation for a finite number of steps.
- If either of the simulations ever halts and outputs 1, do the same.
- If both simulations halt in a 'rubbish' state, go into an infinite loop.

This program semi-decides $U \cup V$. If $m \in U$, then at some point the simulation of A on m will halt and output 1, and so will our program. Otherwise it will either halt in a 'rubbish' state, or run forever. Similarly if $m \in V$. But if m is in neither, then both simulations will either halt in a 'rubbish' state, or run forever. In the first case we loop forever, and in the second we are forced to keep simulating forever.

*** 5. 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? [Hint: if it is, describe a program that decides it. Think simply, write informally, and do not let the syntactic poverty of While confine you.]

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.

** 6. 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.

**** 7. Prove that the set

$$\mathsf{Zero} = \{ \ulcorner S \urcorner \mid [\![\mathrm{S}]\!]_{\mathrm{x}} (0) \downarrow \}$$

is undecidable by reduction from HALT.

Solution

We prove this 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 the source code $\ulcorner S \urcorner$ and n computes the source code $\ulcorner G_{S,n} \urcorner$. Furthermore, we have that

$$\langle \ulcorner S \urcorner, n \rangle \in \mathsf{HALT} \iff [\![G_{S,n}]\!]_{\mathrm{x}} (0) \downarrow \iff \ulcorner G_{S,n} \urcorner \in \mathsf{Zero}$$

Hence, if we could ecide Zero then we could decide HALT. So we cannot decide Zero.

**** 8. [Trick question.] Is the predicate

$$V = \{ \ulcorner S \urcorner \mid \forall n \in \mathbb{N}.\ [\![\mathrm{S}]\!]_{\mathrm{x}} (n) \downarrow \}$$

from the last lecture semi-decidable? Why or why not? Discuss only, do not prove.

Solution

It is indeed *not* semi-decidable.

However, you may intuitively notice that semi-decidability usually amounts to running simulations, which try to decide whether $\exists n.\phi(n)$ for some decidable property $\phi(n)$ of natural numbers by trying all possible $n \in \mathbb{N}$ in parallel, hoping that one might succeed.[1]

In this case, the property $\phi(n) = [\![\mathrm{S}]\!]_{\mathrm{x}} (n) \downarrow$ is asked to hold of *all* natural numbers. Thus, a positive answer seems to somehow require trying all numbers in parallel at the same time, and noting that the property succeeds for all numbers. Algorithmically, this seems to require an infinite number of simulations! While this is indeed true, you do not currently have the tools to show this.

[1] See also https://risingentropy.com/the-arithmetic-hierarchy-and-computability/.