Programming Languages and Computation

Week 11: While eats itself

** 1. 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 $\langle \mathtt{skip}, \sigma \rangle$ before 127 steps, it returns true. Otherwise it returns false.

** 2. 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.

*** 3. 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.

*** 4. 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 represents the states of your DFA 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.