Lecture #6: Computation on Strings — Intermediate Models

Exercises and Review

Questions for Review

Let Σ be some finite, nonempty alphabet

$$\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$$

with size $n \geq 1$. An ***encoding*** $\#(\sigma_i)$ for each ***symbol*** $\sigma_i \in \Sigma$ can be defined by setting $\#(\sigma_i) = i$ for $1 \leq i \leq n$. An encoding $\#(\omega)$ for every ***string*** $\omega \in \Sigma^\star$ can be defined by setting $\#(\lambda) = 0$, where λ is the empty string, and for a string

$$\omega = \gamma_m \gamma_{m-1} \cdots \gamma_1 \gamma_0 \in \Sigma^\star$$

with positive length $m + 1$, setting the encoding $\#(\omega)$ of ω to be

$$\#(\gamma_m) \cdot n^m + \#(\gamma_{m-1}) \cdot n^{m-1} + \cdots + \#(\gamma_1) \cdot n + \#(\gamma_0).$$

It can be shown that every natural number $k \in \mathbb{N}$ is the encoding $\#(\omega)$ for ***exactly one*** string $\omega \in \Sigma^\star$ if this encoding scheme is used.

Consider the programming language $\mathcal{S}_n$ that is defined and considered at the beginning of the lecture notes.

1. What are the possible values of ***variables*** in the programming language $\mathcal{S}_n$?
2. Briefly describe what happens when each of the following ***statements*** in $\mathcal{S}_n$ is executed.
   (a) $V \leftarrow \sigma V$, where V is a variable and $\sigma \in \Sigma$
   (b) $V \leftarrow V^-$, where V is a variable
   (c) IF V ENDS σ GOTO L, where V is a variable, $\sigma \in \Sigma$, and L is a label
3. What does it mean for a partial or total function $f : \mathbb{N}^k \to \mathbb{N}$ (for a positive integer k) to be ***$\mathcal{S}_n$-computable***, for a positive integer n?

4. Once again, consider a partial or total function $f : \mathbb{N}^k \to \mathbb{N}$ (for a positive integer k).
   (a) If f is $\mathcal{S}$-computable then is f $\mathcal{S}_n$-computable for some positive integer n? How can this be proved?
   (b) If f is $\mathcal{S}_n$-computable, for some positive integer n, then is f $\mathcal{S}$-computable? How can this be proved?

Post-Turing programs were introduced later in the notes for this lecture.

5. Consider ***Post-Turing programs***.
   (a) Give the formal definition of a ***Post-Turing program*** — describing all three components of this.
   (b) Describe the way that ***information*** is stored and accessed when a Post-Turing machine is being executed.
   (c) Describe each of the ***statements*** in the programming language $\mathcal{T}$ used by Post-Turing programs and their effects.
   (d) Describe the ***initial configuration*** of a Post-Turing program when it is executed on a given sequence of inputs.
6. Recall that ***two*** related notions of "computability" were introduced using Post-Turing programs.
   (a) What does it mean for a Post-Turing program to ***strongly compute*** a partial or total function from sequences of strings to strings?
   (b) What does it mean for a Post-Turing program to ***weakly compute*** a partial or total function from sequences of strings to strings?
7. Describe, as best as you can, how all the notions of "computability" of partial or total functions are related. Which, if any, of these relationships have not been proved yet?

Practice Problems

These problems — which continue activities in the lecture presentation — will not be discussed during the lecture, and solutions for these problems will not generally be made available. They can be used as "practice" problems that can help you to practice skill considered in the lecture presentation for Lecture #6.

1. Consider the simulation of the execution of an $\mathcal{S}_n$-program $\mathcal{P}$ using a Post-Turing machine that is given to prove Claim #3 in the lecture notes. As in the lecture you notes, you should assume that the only *input variables* mentioned in $\mathcal{P}$ are

$$X_1, X_2, \ldots, X_k$$

when $\mathcal{P}$ is being used to compute a function $f : (\Sigma^\star)^k \to \Sigma$ (for an alphabet Σ). You should also assume the only *local* variables mentioned in the program are (some of)

$$Z_1, Z_2, \ldots, Z_\ell$$

(for a non-negative integer ℓ). Setting $m = k + \ell + 1$ one can then set the variable V_i as follows, for $1 \leq i \leq m$:

$$V_i = \begin{cases} X_i & \text{if } 1 \leq i \leq k, \\ Z_{i-k} & \text{if } k+1 \leq i \leq k+\ell, \\ Y & \text{if } i = k+\ell+1 = m. \end{cases}$$

You should also assume that no label is used to label more than one statement in $\mathcal{P}$.

Since an $\mathcal{S}_n$-program is being simulated, the "input alphabet" for this has the form

$$\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$$

— and this should be the input alphabet for the Post-Turing program as well.

(a) Let $\omega_1, \omega_2, \ldots, \omega_\ell \in \Sigma^\star$ Reviewing the lecture notes as needed, describe how the Post-Turing program's tape is used to represent a state, during the execution of $\mathcal{P}$, such that V_i has value ω_i for $1 \leq i \leq \ell$. Make sure to say, as precisely as you can, where the tape head should be located and what the contents of the non-blank part of the tape should be.

(b) Suppose that the execution of $\mathcal{P}$ continues with an execution of an instruction

$$V_h \leftarrow \sigma_j V_h \tag{1}$$

for integers h and j such that $1 \leq h \leq m$ and $1 \leq j \leq n$. What are the values of the variables $V_1, V_2, \ldots, V_m$ after this statement has been executed?

(c) Describe how the Post-Turing program's tape should represent this: Where should the tape head be located and what should the contents of the tape be now?

(d) Reviewing the lecture notes once again, give the code (including the Post-Turing program's sequence of instructions $\mathcal{C}$) that is used to simulate the execution of the instruction at line (1). This will include several pseudo-instructions, including "RIGHT TO NEXT BLANK", that are described in the lecture notes. Sketch a proof of the correctness of this part of the Post-Turing program's code, assuming the correctness of the macros implementing the pseudo-instructions that are used.

(e) Locate the macro, in the lecture notes, that implements the pseudo-instruction "RIGHT TO NEXT BLANK" and sketch a proof of *its* correctness. As part of this you should say (in a bit more detail than the lecture notes) precisely what this pseudo-instruction should accomplish.

The macro implementing this includes a statement with a label, "A". Your proof should state the number of times that this statement is reached, and it should describe what the Post-Turing program's tape looks like, and where the tape head is located each time this statement is reached (a claim that can be proved by induction on the number of times the instruction has been visited).

Your answer should describe how this claim (and what it says about the *last* time the statement is visited) can be used to complete a proof of the correctness of the macro.

(f) List the other instructions (of an $\mathcal{S}_n$-program) whose simulations (using code in the simulating Post-Turing program) should be described and proved to be correct — along with other pseudo-instructions and macros for a Post-Turing program that should be examined.

Ideally, after doing this, you will understand that complete proof of Claim #3 would probably be quite long — but that, given sufficient time, such a proof could be developed.

2. Consider the simulation of an execution of a Post-Turing program using an $\mathcal{S}$-program that is given to prove Claim #4 in the lecture notes. Suppose, in particular, that you are simulating the execution of a Post-Turing program

$$\mathcal{P} = (\Sigma, \Gamma, \mathcal{C})$$

such that

$$\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$$

and

$$\Gamma = \{\sqcup, \sigma_1, \sigma_2, \ldots, \sigma_{m+n}\}$$

for integers $n \geq 1$ and $m \geq 0$. In the construction described to develop a simulation $\sqcup$ is encoded by $n + m + 1$ (so that, effectively, $\sqcup = \sigma_{n+m+1}$).

Let $\mathcal{Q}$ be the $\mathcal{S}$-program, discussed in the proof of Claim #5, that computes the same function as $\mathcal{P}$ (by simulating the Post-Turing program's execution on a given input string).

(a) Describe, as precisely as you can, how variables in $\mathcal{Q}$ are used to represent the non-blank part of $\mathcal{P}$'s tape and the location of its tape head.

(b) Describe what must be done for the "initialization" phase of the simulation used to prove Claim #4, and the part of the $\mathcal{S}$-program, $\mathcal{Q}$, that is used to carry this out.

(c) Consider an execution of instruction "RIGHT".

i. Describe how this changes the contents of the Post-Turing program's tape and location of its tape head.

ii. Describe how values of $\mathcal{Q}$'s variables must be changed in order to represent the update of the tape's contents and tape head location.

iii. Give the code that should be included in $\mathcal{S}$ to carry this out and sketch a proof of its correctness. The code can be found in the lecture notes; you will need to supply additional details to justify its correctness.

(d) List the other instructions of a Post-Turing program whose executions must also be simulated. As time permits, describe how values of a simulating $\mathcal{S}$-program's variables must be updated to represent an execution of each instruction, and check that the part of the simulating program, $\mathcal{Q}$, that implements this, is correct.

(e) Describe the "cleanup" phase that is needed to end the simulation. Describe how this is implemented (in code for the simulating $\mathcal{S}$-program $\mathcal{Q}$) and explain why this implementation is correct.

Ideally, after doing all this, you will understand that a detailed proof of Claim #4 might also be quite long — but that, given sufficient time, such a proof could be completed.