Lecture #11: Multi-Tape Turing Machines, Nondeterministic Turing Machines, and the Church-Turing Thesis

Lecture Presentation

Problem To Be Solved

Function To Be Computed

Let $\Sigma_1 = \Sigma_2 = \{0, 1\}$ and let $L_{\text{bin}} \subseteq \Sigma_1^\star$ be the set of strings in $\Sigma_1^\star$ that are unpadded binary representations of non-negative integers. Then L_{bin} is the union of the set (of size one) $\{0\}$ and the set of non-empty strings in $\Sigma_1^\star$ that begin with 1. Consider a total function $f_{+1} : \Sigma_1^\star \to \Sigma_2^\star$ such that, for all $\omega \in \Sigma_1^\star$,

- If $\omega \in L_{\text{bin}}$, so that ω is the unpadded binary representation of a non-negative integer n, then $f_{+1}(\omega)$ is the unpadded binary representation of the integer $n + 1$.
- If $\omega \notin L_{\text{bin}}$ then $f_{+1}(\omega) = \lambda$, the empty string.

High-Level Description

A ***high-level description*** of a Turing machine, that solves a problem, is an algorithm to solve the problem — which can be given as pseudocode or simple English.

An algorithm to compute the function f_{+1}, given above, is shown in Figure 1 on the next page.

Sketching a Proof of Correctness

What You Need To Do.

On input $\omega \in \Sigma_1^\star$:

1. `if` $(\omega == \lambda)$ `{`
2. `  return` λ
3. `}` `else if` (ω starts with 0) `{`
4. `  if` $(|\omega| == 1)$ `{`
5. `  return` 1
   `}` `else {`
6. `  return` λ
   `}`
7. `}` `else if` (ω includes at least one copy of 0) `{`
8. Return the string obtained from ω by replacing the ***rightmost*** copy of 0 with 1, and by replacing every copy of 1 to the right of that symbol with 0

   `}` `else {`
9. `  return` 10^h, where $h = |\omega|$

   `}`

Figure 1: A "High-Level Description"' of a Turing Machine to Compute f_{+1}

Details for Useful Cases.

A Proof of Correctness.

What Have We Gained, Here: How is This Progress?

Implementation-Level Description

An ***implementation-level description*** is consistent with the high-level description of the Turing machine being developed. It generally includes details about the following:

- the way the Turing machine uses its finite control to remember information,
- how information (mentioned in the algorithm) is represented using the Turing machine's tape, and
- the way the Turing machine updates its tape, and location of its tape head, to carry out the algorithm's steps.

Application To This Example: Use of the Finite Control

Representing Information on the Tape — and Keeping Track of Things

Implementing the Steps of the Algorithm

Implementing the Tests at Lines 1 and 3. Since the computation is just starting when these tests are reached these steps will be implemented — at least, in part — using transitions out of the Turing machine's start state. The leftmost cell of the tape should be marked, so that it can be recognized later, as part of this implementation.

Implementing the Step at Line 2. This is easy, once you think about what the input is if this step has been reached and what output is required.

Implementing the Test at line 4. The implementation of this step includes the *completion* of a transition that was considered when the test at line 3 was considered — which includes the identification of another non-halting state — and the beginning of several other transitions.

Implementing the Step at Line 5. A careful *completion* of one of the transitions considered above, is quite a bit of what is needed here:

Implementing the Step at Line 6. In this case we need to complete another of the transitions already being considered — and then do the following.

Erasure of the input can be carried out as follows.

Implementing the Test at Line 7. If you also look at the steps that must be implemented *after* this test you might see that it is easiest to consider these when the tape head is all the way to the right of the input string — so let's include a complete sweep to the right of the input string as part of the input string.

What you need to remember during the sweep to the right — and how this sweep ends:

Implementing the Step at line 8. If this step has been reached then the input string has the form $\mu 01^h$ for some (non-empty) string $\mu \in \Sigma_1^\star$ and for some non-negative integer h. The tape head is resting at the leftmost "$\sqcup$", to the right of the input, when implementing this step begins.

We want to return the string $\mu 10^h$ as output. Here's how we do it:

Implementing the Step at Line 9. If this step has been reached then the input string has the form 1^k for a positive integer k. The tape had is resting at the leftmost "$\sqcup$"to the right of the input, when implementing this step begins.

We want to return the string 10^k as output. Here's how we do it:

Formal Description

The ***formal description*** of the Turing machine is the description of the Turing machine

$$M = (Q, \Sigma_1, \Sigma_2, \Gamma, \delta, q_0, q_{\text{halt}})$$

that we have already been considering — with each of the components of M presented as described in the previous lecture.

It turns out that the Turing machine (computing f_{+1}), considered in the previous lecture presentation, is consistent with the "implementation-level' Turing machine that has now been described.

Details: