Lecture #12: Multi-Tape Turing Machines, Nondeterministic Turing Machines, and the Church-Turing Thesis

What Will Happen During the Lecture

The goals of this lecture presentation include a discussion of ***Turing machines that compute functions*** — which will be needed in this course, when "computability and decidability" is being considered. It is also intended to help students to be more familiar with multi-tape Turing machines, to see how Turing machine design can get a little bit *easier* when multi-tape Turing machines are available, and, finally, to give a bit more evidence in support of the ***Church-Turing thesis***.

Review

The lecture presentation will begin with a ***brief*** review of the material in the preparatory video and documents for this lecture — and students will have the chance to ask questions about this.

Designing a Multi-Tape Turing Machine for Additional of Binary Numbers

During the previous lecture (and in a supplemental document for it) ***Turing machines that compute functions*** were introduced. Now that ***multi-tape Turing machines*** (that recognize languages) have been introduced, ***multi-tape Turing machines that compute functions*** can be introduced too — and a supplemental document with information about them is available.

Let $\Sigma_{\text{binary}} = \{0, 1\}$, and let $f_{\text{b\_inc}} : \Sigma_{\text{binary}}^{\star} \to \Sigma_{\text{binary}}^{\star}$ such that the following properties are satisfied for every string $\omega \in \Sigma_{\text{binary}}^{\star}$:

- If ω is the unpadded binary representation of a non-negative integer n then $f_{\text{b\_inc}}(\omega)$ is the unpadded binary representation of $n + 1$.

- On the other hand, if ω is *not* the unpadded binary representation of any non-negative integer, then $f_{\text{b\_inc}}(\omega) = \lambda$, the empty string.

Let $f_{\text{b\_dec}} : \Sigma_{\text{binary}}^{\star} \to \Sigma_{\text{binary}}^{\star}$ such that the following properties are satisfied for every string $\omega \in \Sigma_{\text{binary}}^{\star}$:

- If ω is the unpadded binary representation of a *positive* integer n, then $f_{\text{b\_dec}}(\omega)$ is the unpadded binary representation of $n-1$.

- On the other hand, if ω is *not* the unpadded binary representation of any positive integer, then $f_{\text{b\_dec}}(\omega) = \lambda$, the empty string.

In the supplemental document for the previous lecture about Turing machines that compute functions, a Turing machine that computes the above function $f_{\text{b\_inc}}$ was presented, along with a proof of its correctness. The document ended with an exercise; if the exercise was completed then a Turing machine that computes the function $f_{\text{b\_dec}}$ was obtained, and proved to be correct, as well.

Now let $\Sigma_{\text{pair}} = \{0, 1, \#\}$ and consider a function $f_{\text{add}} : \Sigma_{\text{pair}}^{\star} \to \Sigma_{\text{binary}}^{\star}$ such that the following properties are satisfied for every string $\omega \in \Sigma_{\text{pair}}^{\star}$:

- If $\omega = \mu \# \nu$, where $\mu, \nu \in \Sigma_{\text{binary}}^{\star}$ are the unpadded binary representations of a pair of non-negative integers n and m, respectively, then $f_{\text{add}}(\omega)$ is the unpadded binary representation of their sum, $n + m$.

- Otherwise $f_{\text{add}}(\omega) = \lambda$, the empty string.

During the lecture presentation, a ***multi-tape Turing machine*** (that computes functions, and that uses Turing machines computing $f_{\text{b\_inc}}$ and $f_{\text{b\_dec}}$ as components) will be designed, and a proof that it corrects the function f_{add} will be sketched.