CSCI3390-Lecture 5: The Universal Turing Machine

September 18, 2018

1 Summary

- The Turing machine as introduced is a special-purpose computer–*i.e.,* a programmed computer–that runs just a single program: adding two numbers, checking if two strings are equal, sorting its input, adding two numbers, *etc.*
- It is also possible to build a Turing machine $\mathcal{U}$ that functions as a *general-purpose* computer–*i.e.,* a programmable computer. Such a machine takes as input a pair consisting of the specification of a Turing machine $\mathcal{M}$, and a string w, and accepts this input if and only if $\mathcal{M}$ accepts w.
- $\mathcal{U}$ is called a *universal Turing machine.*

2 Encoding Turing machines

We can encode a Turing machine as a string. All we need to do is list all the transitions, with each transition being a quintuple

$$(q, a, p, b, D),$$

where $D \in \{L, R\}$, to denote

$$\delta(q, a) = (p, b, d).$$

For example, we can encode each state, tape symbol and direction as an integer, and simply write each quintuple as a sequence of five integers separated by some

special separator symbol. So the encoding of a Turing machine might begin

$$0\#5\#1\#12\#1\#\#0\#6\#\cdots$$

This is much like the encodings of graphs we saw back at the start of the course. (We will need standard encodings for the initial, accepting, and rejecting states.) So let us commit to some such encoding scheme for Turing machines; as usual, the details hardly matter, as long as we can unambiguously recover the Turing machine from its string encoding. [1]

As we saw in Assignment 1, any such encoding can be converted into a binary encoding over the alphabet $\{0, 1\}$. If $\mathcal{M}$ is a Turing machine, let's denote by $< \mathcal{M} >$ the binary encoding of $\mathcal{M}$.

Now suppose $w \in \Sigma^*$, where Σ is the input alphabet of $\mathcal{M}$. We can also encode a pair $(\mathcal{M}, w)$ by a binary string, so let us write $< \mathcal{M}, w >$ for the binary encoding of such a pair. In general, we will use this notation to represent the binary encoding of any object. So, later in these notes, you will see $< q >$ used to represent the binary encoding of the state q. Again, it doesn't matter *what* encoding scheme we use for this, as long as we are always able to figure out what object is represented from the encoding.

3 The Universal Turing Machine

Consider the set $L_{TM} \subseteq \{0, 1\}^*$ of all strings $< \mathcal{M}, w >$, such that $w \in \Sigma^*$ and $\mathcal{M}$ accepts w. In other words, an element of L_{TM} is (the encoding of) a pair consisting of a machine and a string, with the restriction that the machine accepts the string.

Theorem 1 *L_{TM} is a Turing-recognizable language.*

Let's pause a moment to see what this theorem is *really* saying. It states that there is a Turing machine $\mathcal{U}$ that behaves as follows: If you give $\mathcal{U}$ a machine $\mathcal{M}$ together with a string w, it will tell you what $\mathcal{M}$ does when it is run on w. In other word $\mathcal{U}$ is a Turing machine that can simulate *every* Turing machine–so it is called a *universal* Turing machine.

[1] This includes being able to tell whether the string really does encode a Turing machine. It may be that the format is incorrect—perhaps the number of fields is not divisible by 5. It might also be that the list of quintuples gives two different results for $\delta(q, a)$ for some state a and input a. But all of these things are easily checked, and we can check them with a Turing machine.

Turing machines as we have introduced them are special-purpose computers, each running a single program. The universal Turing machine is a general-purpose programmable computer: give it a program (the specification of the Turing machine $\mathcal{M}$) and an input, and it will 'run' $\mathcal{M}$ on w.

4 Proof outline.

Before we proceed to a more detailed argument, we point out that the theorem MUST be true if you believe the Church-Turing thesis. Because we certainly have a pencil-and-paper algorithm for simulating a given Turing machine on a given input–you carried out this algorithm in the first homework assignment. So assuming that such pencil-and-paper algorithms can be realized by Turing machines, our universal TM must exist.

Here is a more detailed, but still 'high-level' argument. Rather than describe the construction of $\mathcal{U}$ in detail, we will describe at a higher level the construction of a *3-tape* TM $\mathcal{U}_3$ that recognizes L_{TM}. We know from the previous lecture that this 3-tape machine can in turn be simulated by a 1-tape machine, so putting the pieces together will yield a one-tape universal TM.

$\mathcal{U}_3$ begins with its input on its first tape.

In the first phase, it 'unpacks' the input, so that the input word w is on the first tape, the list of quintuples of $\mathcal{M}$ is on the second tape, and the state on the third tape. The current position on each tape is at the leftmost symbol of the tape contents.

It then repeatedly executes the following steps: (a) Find the quintuple on tape 2 whose first two components match the state on tape 3 and the current input symbol on tape 1. (b) Replace the current symbol on tape 1 by the fourth component of the quintuple, and the state on tape 3 by the third component of the quintuple, and move the current position on the first tape to the right or the left according to the fifth component of the quintuple. (c) Return the current position on tapes 2 and 3 to the start of the tape contents. Observe that the contents of tape 2 never change during this process.

If the transition located on tape 2 directs $\mathcal{M}$ to enter the accept or reject state, $\mathcal{U}_3$ does likewise. Thus $\mathcal{U}_3$ accepts the input $< \mathcal{M}, w >$ if and only if $\mathcal{M}$ accepts w. (The accompanying figures illustrate how $\mathcal{U}_3$ simulates $\mathcal{M}$.)

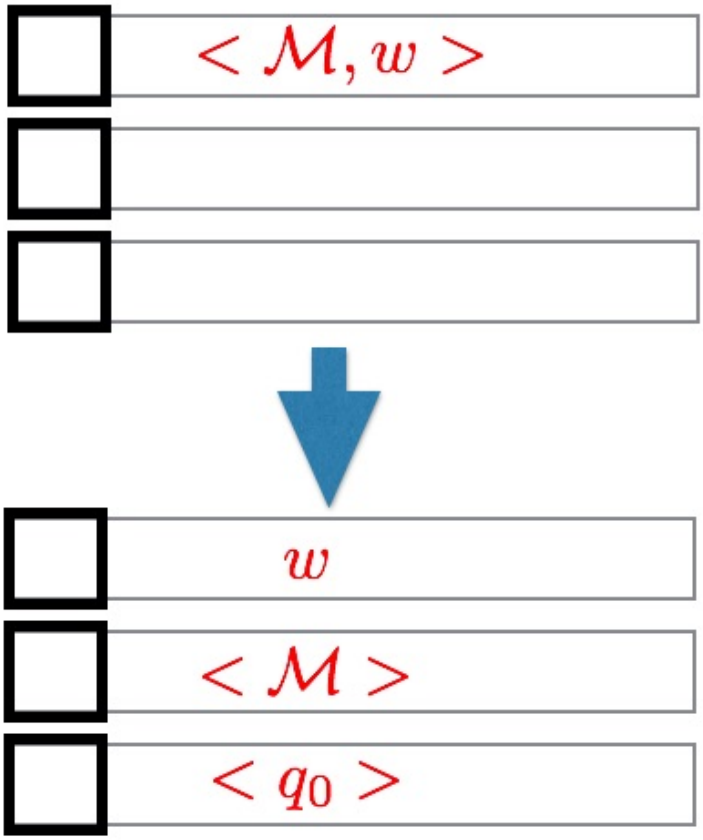

Figure 1: The initial configuration of $\mathcal{U}_3$ (top) and the configuration just after the unpacking phase is completed. The state q_0 is the initial state of $\mathcal{M}$.

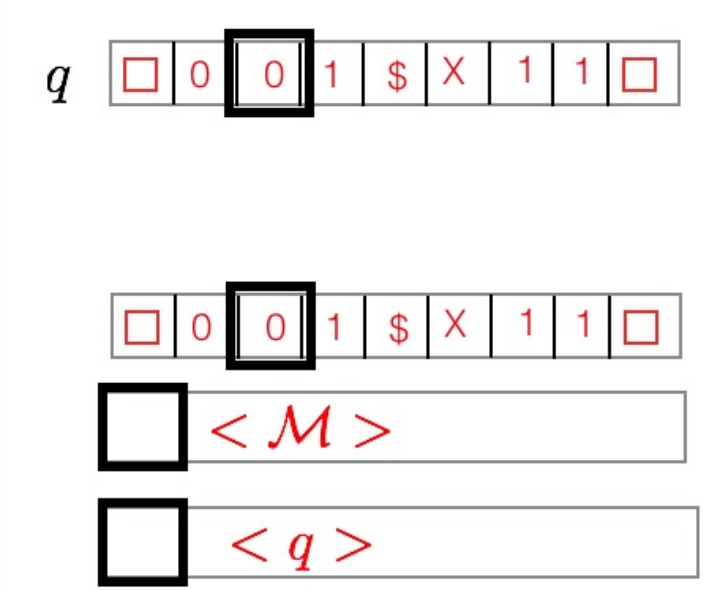

Figure 2: The configuration of $\mathcal{M}$ (top) at one moment during the run of $\mathcal{M}$ on w, and the corresponding configuration of the simulation of this run by $\mathcal{U}_3$ (bottom). Observe that the first tape of $\mathcal{U}_3$ is identical to the tape of $\mathcal{M}$, including the location of the read head.

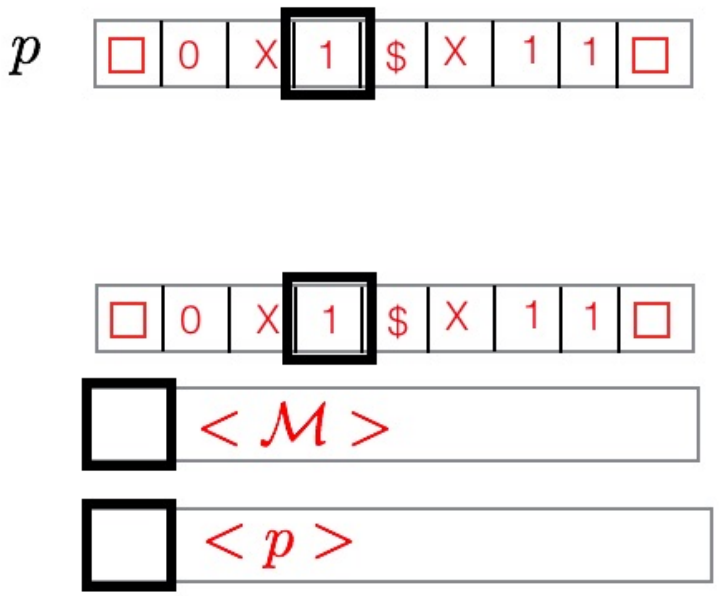

Figure 3: The configuration of $\mathcal{M}$ immediately following the one illustrated in Figure 2, and the corresponding configuration of $\mathcal{U}_3$. $\mathcal{M}$ has executed the transition $\delta(q, 0) = (p, X, R)$. To simulate this, $\mathcal{U}_3$ has to search the table of transitions of $\mathcal{M}$ stored on tape 2, modify tape 1, write the new state on tape 3, and return the reading heads for tapes 2 and 3 to the start. Observe that the contents of tape 2 never change during the simulation.