COS 487 Notes on the Recursion Theorem
Fall 2012
Tarjan

We are interested in Turing machines as algorithms to compute functions: if *M* is a Turing machine, *M*(*x*) denotes the output of machine *M* on input *x*, which is the value at *x* of the function computed by *M*.

Question: Can we construct a machine that produces a description of itself? Such a machine can be turned into a self-reproducing machine. More generally, is there a way to give a Turing machine access to its own description?

Here is a simple construction of a machine that writes a description of itself on an empty tape. Let *M* be a one-input Turing machine. A diagonal machine *D*.*M* for *M* has the following behavior: On an initially empty tape, *D*.*M* runs *M* on <*M*>. The following is an implementation of *D*.*M*. Write <*M*>. Transfer control to *M*. A diagonal machine *D* has the following behavior: *D*(<*M*>) = <*D*.*M*>). That is, given a description <*M*> of a Turing machine *M*, *D* writes a description of a diagonal machine *D*.*M* for M. The following is an implementation of *D*. On input <*M*>, construct a machine *D*.*M* that writes <*M*> on an empty tape and then runs *M*, and write a description <*D*.*M*>.

Now let SELF be the machine such that *D*(<*D*>) = <SELF>. That is, let SELF be the machine whose description is written by *D* when run on input <*D*>. What does SELF do? On empty tape, it writes <*D*> and then runs *D* on <*D*>, producing *D*(<*D*>) = <SELF>. That is, SELF writes a description of itself!

Recursion Theorem: For any Turing machine *T* that computes a function of two inputs, the first of which is a Turing machine description, there is a Turing machine *R*.*T* such that *R*.*T*(*x*) = *T*(<*R*.*T*>, *x*). Furthermore, there is a Turing machine *R* such that *R*(<*T*>) = <*R*.*T*>. That is, the proof of the Recursion Theorem can be made constructive.

Proof:

We start by generalizing *D*.*M* and *D* to machines *M* with two inputs. Let *M* be a two-input Turing machine. A diagonal machine *D*.*M* for *M* has the following behavior: *D*.*M*(*x*) = *M*(<*M*>, *x*). The following is an implementation of *D*.*M*. On input *x*: write <*M*> next to x, then run *M* on <*M*>, *x*. This implementation builds both a machine to write <*M*>, and a copy of *M* itself, into its finite control. An alternative is to write two copies of <*M*> and then run *U* on <*M*>, <*M*>, *x*, where *U* is a universal two-input Turing machine. This implementation is more generic in that the only part that depends on <*M*> is the part that writes <*M*>, but either implementation will do.

A diagonal machine *D* has the following behavior: *D*(<*M*>) = <*D*.*M*>. That is, on input <*M*>, it produces a description of a diagonal machine for *M*. The following is an implementation of *D*. On input <*M*>: construct *D*.*M*, where *D*.*M* is implemented as in the preceding paragraph, and write <*D*.*M*>.

A meta machine *E*.*T* for *T* has the following behavior: *E*.*T*(<*M*>, *x*) = *T*(*D*(<*M*>), *x*). The following is an implementation of *E*.*T*. On input <*M*>, *x*: run *D* on <*M*> to write <*D*.*M*>, then run *T* on <*D*.*M*>, *x*).

Let *R*.*T* be the machine such that *D*(<*E*.*T*>) = <*R*.*T*>; that is, the machine whose description *D* produces on input <*E*.*T*>, where *E*.*T* is some meta machine for *T*. Then *R*.*T*(*x*) = *E*.*T*(<*E*.*T*>, *x*) = *T*(*D*(<*E*.*T*>), *x*) = *T*(<*R*.*T*>, *x*), as desired.

[End of proof]

To make this proof constructive, we build a meta machine *E* such that *E*(<*T*>) = <*E*.*T*>. That is, on input <*T*>, *E* produces a description of *E*.*T*. An implementation of *E* is as follows. On input <*T*>: construct *E*.*T*, where *E*.*T* is implemented as described above, and write <*E*.*T*>. Given *E*, we build a machine *R* such that *R*(<*T*>) = *D*(*E*(<*T*>)) = <*R*.*T*>.

Instead of directly constructing a machine that writes a description of itself, as we did at the beginning of these notes, we can obtain such a machine by applying the recursion theorem to the trivial machine HALT that halts immediately. Indeed, let SELF’ be such that <SELF’> = *R*(<HALT>). That is, SELF’ is the machine whose description is written by *R* on input <HALT>. Then SELF’ on empty input writes <SELF’> and halts. (Why?)