COSC 341
Theory of Computing
Lecture 12
More on Turing Machines

Stephen Cranefield
stephen.cranefield@otago.ac.nz

Lecture slides (mostly) by Michael Albert

Keywords: Acceptance conditions, computing functions

Turing machines as function computers

- How do we "compute a function" with a TM?
- Let f be a function whose domain is some subset of Σ^* and whose co-domain is Σ^*.
- Then $\mathbf{M}$ computes f if, when the tape is initialised to w in the domain of f, running $\mathbf{M}$ leaves the tape in final state $w\#f(w)$ (or possibly just $f(w)$) and halts and accepts.
- On any other input, f should halt and reject, or halt abnormally, or run forever.

Let’s build two machines

- “Compute 2^k” : input a^k, output $a^k\#b^{2^k}$
- “Convert to binary” : input a^k, output $a^k\#w$ where $w \in \{0,1\}^*$ is the representation of k in binary.

Variations on acceptance

To accept you must halt. Is it necessary to designate accepting and non-accepting final states?

Theorem

The collections of languages accepted by Turing machines with designated accepting final states and those accepted by Turing machines by halting (at all) are the same.

- One direction clear - make all halting states accepting.
- In the other direction, to eliminate non-accepting halts, just replace the (undefined) transitions from them (which cause the halt) by transitions to a new state, TO INFINITY, which moves the head rightwards forever.

One more machine

Accept the language PRIME consisting of all sequences a^k where k is a prime number.