COMS W3261
Computer Science Theory
Lecture 13
Variants of Turing Machines

Alexander Roth

2014 – 10 – 15

Outline

1. Review
2. Programming techniques for Turing machines
3. Models of computation equivalent to Turing machines

1 Review: Recursive and Recursively Enumerable Languages

- A language L is *recursively enumerate* if $L = L(M)$ for some TM M.
- We sometimes say a language is Turing-recognizable if some TM recognizes it.
- We say that a language L is *recursive* if $L = L(M)$ for some Turing machine M such that
  1. If w is in L, then M accepts w and therefore halts.
  2. If w is not in L, then M eventually halts but never enters an accepting state.
- A language L is said to be *decidable* if it is a recursive language.
- A language L is said to be *undecidable* if it is not a recursive language.

2 Programming Techniques for Turing Machines

- Turing machines are exactly as powerful as conventional computers.
- To make the behavior of a Turing machine clearer, we use the finite- state control of a Turing machine to hold a finite amount of data. One way to do this is to use states with multiple files, where one field represents a position in the Turing machine program. and the other fields hold data elements. The number of fields in a state is always finite.
- Another way to make the behavior of a Turing machine clearer, is to think of the tape as having several tracks.
- We can also group states into "subroutines". A subroutine has its own start state, and another state which can server as a "return" state.

3 Models of Computation Equivalent to Turing Machines

- Many variants of Turing machines have been defined such as:
  - Turing machines with a semi-infinite input tape.
  - Multitape Turing machines.
  - Turing machines with tapes having multiple tracks.
  - Nondeterministic Turing machines.
- All these machines are equivalent to our definition of a deterministic Turing machines.
- Other universal models of computation:
  - Chomsky type 0 grammars. A type 0 grammar is like a context-free grammar (V, T, P, S) except that productions can be of the form $\alpha \to \beta$ where α is a string of nonterminals and terminals with at least one nonterminal and β is any string of nonterminals and nonterminals.
  - Lambda calculus.
  - Pushdown automata with two or more stacks.
  - Two-counter machines.
  - Random access machines.
  - Most programming languages.
  - Real computers with an arbitrary amount of energy.
- Again, all these models are computationally equivalent to our definition of a deterministic Turing machine.

Class Notes

L_d – The Diagonalization Language.

Let w_1,w_2,w_3 be an enumeration of all binary strings. Let M_1,M_2, M_3,... be an enumeration of all TM's.

Let $L_d = \{w \mid w \notin L(M_1)\}$ for some i.

Time Complexity of a TM

$T(n)$ = maximum number of moves made by M in processing an input of length over all inputs length n.