# COMS W3261
# Computer Science Theory
# Lecture 12
# Turing Machines

Alexander Roth

2014 – 10 – 13

## Outline

1. Turing machines
2. Algorithms and recursive languages
3. Some history
4. Models of computation equivalent to Turing machines

## 1 Turing Machines

- A Turing machine is a generalization of a finite automaton. At any moment in time, it can be pictured as a finite-state control with a tape head reading a symbol on a square of its infinite input tape. Initially, a finite length input string $a_1a_2 \dots a_n$ appears on the input tape with the tape head reading $a_1$ and the finite control in a designated initial state. The symbols of the input string appear in contiguous squares of the input tape. An infinite number of blank symbols are on the input tape to the left of $a_1$ and to the right of $a_n$.
- At the beginning of a move, a Turing machine reads the symbol on the square of the input tape under the tape head and consults the transition function (its "program") stored in its finite-state control. During the move it makes a state transition, replaces the symbol on the input tape with another tape symbol, and shifts the tape head one square to the left or one square to the right. If it has not entered a halting state, the Turing machine then makes another move.

- After a finite (but perhaps very long) number of moves the Turing machine may enter a final state and halt, in which case it is said to accept the input string $a_1 a_2 \dots a_n$ that was originally on the input tape. However, the Turing machine may instead enter a nonfinal state and halt, or it may make an infinite sequence of moves without ever entering a final state. In both cases, it does not accept the original input string.

- More formally, a nondeterministic Turing machine $M$ has seven components: $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$

  1. $Q$ is the finite set of states of the finite control.
  2. $\Sigma$ is the finite set of input symbols.
  3. $\Gamma$ is the finite set of tape symbols; $\Sigma$ is a subset of $\Gamma$.
  4. $\delta$ is the transition function. It maps $(Q \times \Gamma)$ to the subsets of $(Q \times \Gamma \times \{L, R\})$. If $(p, Y, D)$ is in $\delta(q, X)$ and $M$ is in state $q$ reading the symbol $X$ on the input tape, then $M$ can
     - go from state $q$ to state $p$,
     - replace the symbol $X$ on the input tape by the symbol $Y$, and
     - move its input head one square in the direction $D$ where $D$ can be either $L$ (for left) or $R$ (for right).

     M is *deterministic* if there is at most one element in $\delta(q, X)$ for any state $q$ and tape symbol $X$. Unless otherwise qualified, the term "Turing machine" will signify a deterministic Turing machine.
  5. $q_0$ is the start state.
  6. $B$ is the blank symbol. $B$ is in $\Gamma$ but not in $\Sigma$.
  7. $F$, a subset of $Q$, is the set of final accepting states. We assume there are no transitions from a final state so that when $M$ enters a final state it halts.

- During a sequence of moves we can represent the configuration of $M$ by an instantaneous description (ID) of the form $X_1 X_2 \dots X_{i-1} q X_i X_{i+1} \dots X_n$. This ID says $M$ is in state $q$ reading the tape symbol $X_i$. To the left of $X_i$ is the string of tape symbols $X_1 X_2 \dots X_{i-1}$. To the right of $X_i$ is the string of tape symbols $X_{i+1} \dots X_n$. We do not show blanks on the input tape unless necessary.

- The read head may be one input square past the right end of the input reading a blank in which case $M$ is in the ID

$$X_1 X_2 \dots X_n q B$$

- Moves by $M$ are modeled by the following changes to the ID:

1. If $(p, Y, L)$ is in $\delta(q, X_i)$, $M$ can move to the ID

   $$X_1X_2 \dots X_{i-2}pX_{i-1}YX_{i+1} \dots X_n$$

   If $i = 1$, then $M$ moves to the blank to the left of $X_1$ in which case the ID becomes

   $$pBYX_2 \dots X_n$$

   If $i = n$ and $Y = B$, then $M$ replaces $X_n$ by a blank and the ID becomes

   $$X_1X_2 \dots X_{n-2}pX_{n-1}$$

2. If $(p, Y, R)$ is in $\delta(q, X_i)$, $M$ can move to the ID

   $$X_1X_2 \dots X_{i-1}YpX_{i+1} \dots X_n$$

   If $i = n$, then $M$ moves to the blank to the right of $X_n$ in which case the ID becomes

   $$X_1X_2 \dots X_{n-1}YpB$$

   If $i = 1$ and $Y = B$, then $M$ replaces $X_1$ by a blank and the ID becomes

   $$pX_2 \dots X_n$$

- $L(M)$, the language accepted by $M$, is the set of strings $w$ in $\Sigma^*$ such that $q_0w \overset{*}{\vdash} \alpha p \beta$ for some state $p$ in $F$.
- We say that a language $L$ is *recursively enumerable* if $L = L(M)$ for some Turing machine $M$.
- See Example 8.2, p. 329, HMU for a TM that accepts the language $\{0^n1^n \mid n \geq 1\}$.

## 2 Algorithms and Recursive Languages

- 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 Turing machine that hatls on all inputs either in an accepting or nonaccepting state provides a precise definition for the term *algorihtm*. This formalizes our intuitive notion of an algorithm as a collection of simple instructions for carrying out some task.
- 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.

**Class Notes**

1. Given a CFG $G$, is $G$ ambiguous?
2. Given a CFG $G$, is there an equivalent unambiguous grammar?

## 3 Some History

- Hilbert's problems.
  - In 1900, David Hilbert, the eminent mathematician of his day, presented a famous list of twenty-three problems at the International Congress of Mathematicians in Paris.
  - Hilbert's tenth problem was to devise a process according to which it can be determined by a finite number of operations whether a polynomial with integer coefficients (Diophantine equation) has an integral root.
    * For example, the polynomial Diophantine equation

      $$6x^3yz^2 + 3xy^2 - x^3 - 10 = 0$$

      over the variables $x$, $y$, and $z$ has an integral root at $x = 5$, $y = 3$, and $z = 0$. The root is said to be integral because all variables are assigned integer values.
- In 1931 Kurt Gödel published his famous incompleteness theorems which basically said that in any reasonable system of formalizing the notion of provability in number theory, some true statements are unprovable. This shattered Hilbert's dream of finding a complete and consistent set of axioms for all of mathematics (Hilbert's second problem).
- In 1936 Alonzo Church wrote a paper in which he devised a notation called lambda calculus to define algorithms. Lambda calculus is the basis for the programming language Lisp.
- In 1936 Alan turing wrote a paper ("On computable numbers with an application to the Entsheidungsproblem", Proc. London Math. Society) in which he defined Turing machines and showed that the halting problem for Turing machines is undecidable.
  - The machine $M$ in Alan Turing's paper accepted by just halting – there is no final state. Let $H(M)$ be the set of inputs $w$ on which his machine hatls. Turing showed that set of pairs $(M, w)$ such that $w$ is in $H(M)$ is recursively enumerable but not recursive.
- The Church-Turing thesis hypothesizes that any function that can be computed ("effectively computable function") can be computed on a Turing machine.

- In 1970, Yuri Matiyasevich showed that no algorithm exists for testing whether a Diophantine equation has integral roots.

## 4 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 $\alpha$ is a string of nonterminals and terminals with at least one nonterminal and $\beta$ 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.