Lecture 9:
Introduction to Turing Machines

Lecture Outline

- Summary of Machine/Language Results So Far
- What is a Turing Machine ?
- A Bit of History: What was Turing Trying to Do ?

Reading

Chapter 3.1 - 3.3 of Brookshear

Chapter 6.4 of Revesz

Chapter 24 of Cohen.

Chapter 7 of Hopcroft and Ullman

Hodges, A. (1992) Alan Turing: The Enigma. Vintage, London.

Summary of Language/Machine Results So Far

- Regular Grammars/Finite Automata

  1. For any RG G there is a FA M such that $L(M) = L(G)$.
  2. For any FA M there is a RG G such that $L(G) = L(M)$.

  The languages accepted by **deterministic** finite automata are exactly those accepted by **nondeterministic** finite automata.

  There are languages which are not regular – e.g. $x^n y^n$.

- Context-free Grammars/Pushdown Automata

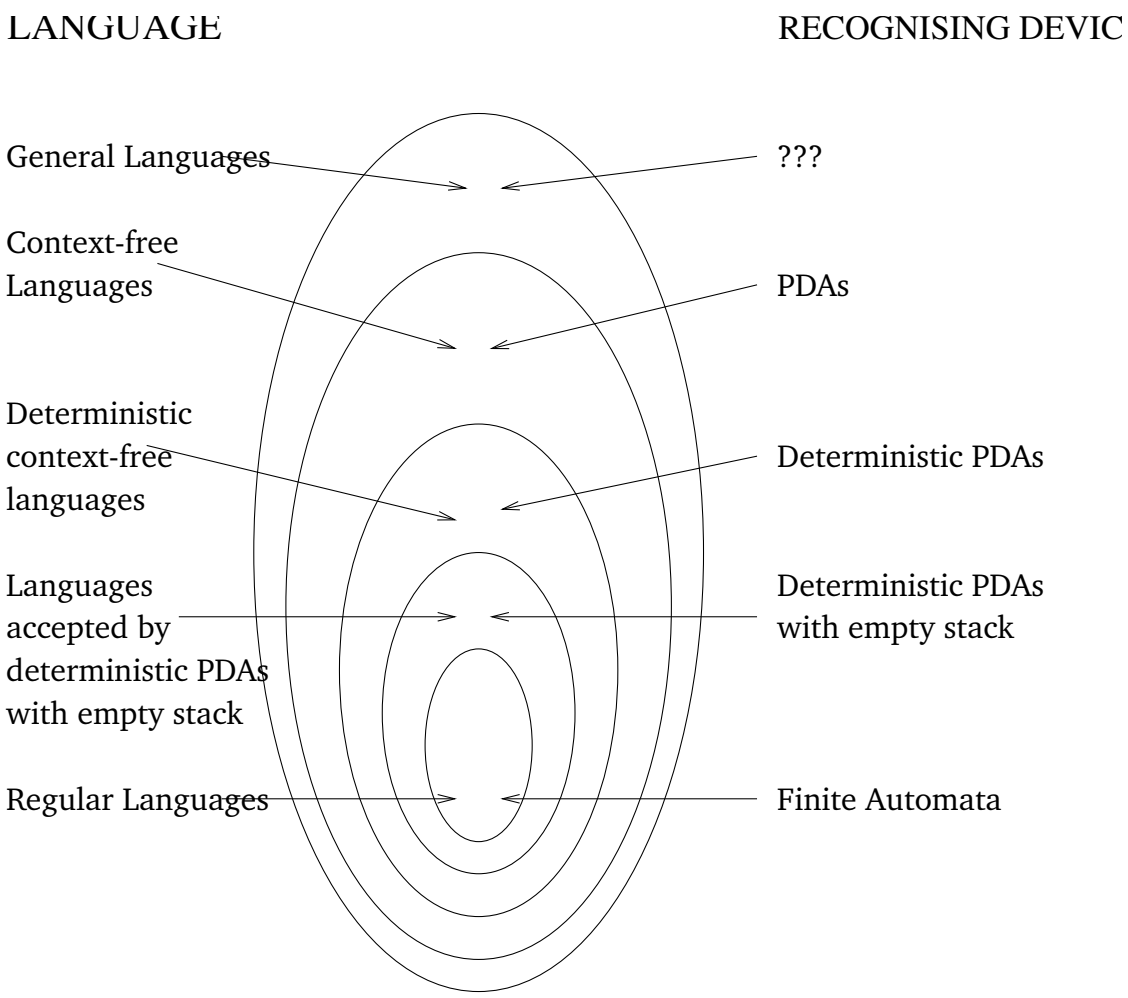
  1. For any CFG G there is a PDA M such that $L(M) = L(G)$.
  2. For any PDA M there is a CFG G such that $L(G) = L(M)$.

  The languages accepted by **deterministic** pushdown automata (DPDA) are **NOT** the same as those accepted by **nondeterministic** pushdown automata (PDA).

  The languages accepted by deterministic pushdown automata with empty stack are **NOT** the same as those accepted by **deterministic** pushdown automata.

  There are languages which are not context-free – e.g. $x^n y^n z^n$.

Summary of Language/Machine Results So Far

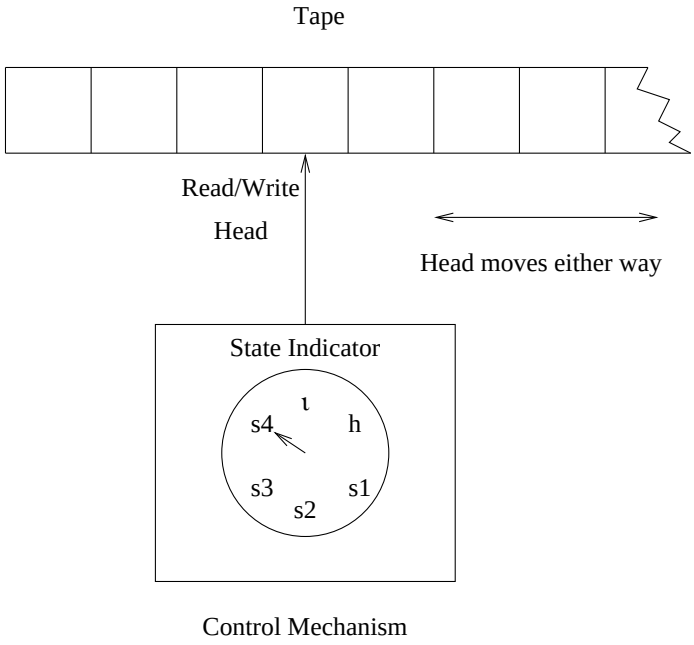


What is a Turing Machine ?

A **Turing machine** (named for its inventor Alan Matheson Turing) has both similarities and dissimilarities to finite automata and pushdown automata.

Similarities:

- it consists of
  1. a finite state control mechanism,
  2. a tape with a fixed left end but stretching indefinitely to the right,
  3. a tape head;
- there is a finite alphabet of input symbols in which any input to the machine must be coded;
- there is an initial state.



What is a Turing Machine ? (cont)

Dissimilarities:

- A Turing machine may **write** to its tape as well as read from it (FAs and PDAs may only read from their input tape).

  This means that there is no need for a stack, since the tape serves the dual purpose of input device and auxiliary storage.

  Further the TM is not limited to pushing and popping symbols on a stack: it can scan the tape reading and/or modifying symbols in any position.

- There is a single **halt** state, distinct from the initial state, in which, if reached, computation ceases.
  By contrast, recall:
  1. FAs and PDAs have a **set** of accept states any of which may be identical to the initial state;
  2. FAs and PDAs may or may not cease computation when they reach an accept state.

- The set of tape symbols includes the input alphabet and additional symbols that the machine can use for internal purposes (like the stack symbols of the PDA which include input symbols and special stack markers).

  In particular we assume the blank symbol is in the tape alphabet, but not in the input alphabet, and we denote it by Δ.

  All cells on the tape not otherwise specified are assumed to contain blanks.

Transitions in a Turing Machine

- Turing machines have two sorts of operations only:
  1. **write** operations: replace a symbol on a tape with another one and shift to a new state (without moving the tape head);
  2. **move** operations: move tape head right and shift to new state or move tape head left and shift to a new state.

- At any time the machine's behaviour is determined by the current state and the current symbol under the tape head.

- Letting
  1. S be a set of states,
  2. $h \in S$ be the halt state,
  3. Γ be a set of tape symbols
  4. L (left) and R (right) be two symbols not in Γ

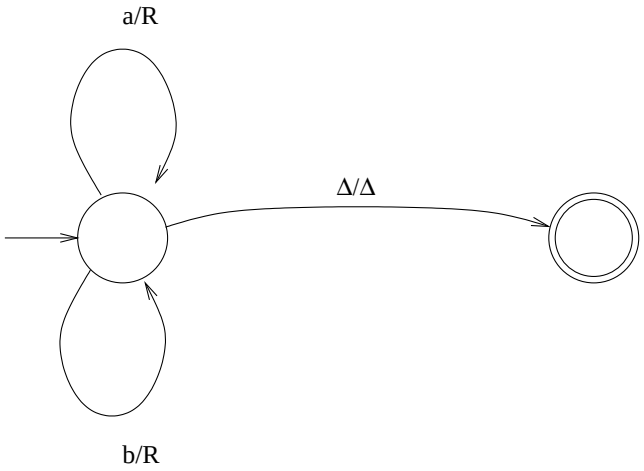
  then a Turing machine's transition function is a function

$$\delta : (S - \{h\} \times \Gamma) \rightarrow (S \times (\Gamma \cup \{L, R\}))$$

- There are three possibilities here which we interpret as follows:
  1. $\delta(p, x) = (q, y)$ means
     If the machine is in state p and the current tape symbol is x then replace x with y and shift to state q
  2. $\delta(p, x) = (q, L)$ means
     If the machine is in state p and the current tape symbol is x then move the tape head one cell to the left and shift to state q
  3. $\delta(p, x) = (q, R)$ means
     If the machine is in state p and the current tape symbol is x then move the tape head one cell to the right and shift to state q

Transition Diagrams for Turing Machines

- As with FAs and PDAs, a Turing machine's transitions may be represented diagrammatically using a transition diagram.
- These diagrams differ from those for FAs and PDAs in one respect – the arc labels take the form s/A, where s is the symbol read off the tape and A is the action to be performed:
  - if A is t then s is to be replaced by t on the tape;
  - if A is L then the tape head is to move left;
  - if A is R then the tape head is to move right.
- Here is a transition diagram for a simple Turing machine that reads a's and b's, moving right after each read, until it encounters a blank (Δ) at which point it rewrites the blank with a blank and then halts in that cell.



A Formal Definition of a Turing Machine

- Since transitions in a Turing machine are defined by a **function**, it follows that the machine is deterministic: given any state (except the halt state) and a tape symbol there is only one possible transition.

- Normally a Turing machine starts in its initial state and executes transitions until its halt state is reached.

  Note:

  1. the halt state may never be reached because the machine goes into a nonterminating loop;
  2. the machine may terminate abnormally if it tries to move its tape head off the left hand end of the tape.

- Formally, a Turing Machine is defined as a sextuple

$$(S, \Sigma, \Gamma, \delta, \iota, h)$$

  where:

  1. S is a finite collection of states;
  2. Σ is a finite set of nonblank symbols called the machine's alphabet;
  3. Γ is a finite set of symbols, including the symbols in Σ, called the machine's tape symbols;
  4. δ is the machine's transition function (as described above);
  5. $\iota \in S$ is the machine's initial state;
  6. $h \in S$ is the machine's halt state

Turing Machines – An Example

Design a Turing machine with tape symbols x, y, and Δ that will search its tape for the pattern $xyxy$ and halt if and only if that pattern is found.

A Bit of History: What was Turing Trying to Do ?

- Turing presented his ideas about computing machines in 1936.

  This work predates work on finite automata and pushdown automata.

- He was not trying to extend simpler models, as we have been doing, but rather was trying to express in the simplest way the essence of computation.

- Turing's model was a human being with pencil and paper attempting to carry out a computation.
  - Such person could concentrate on a limited section of the paper at a given time on which the marks written could be viewed as single symbol.
  - The person could distinguish only a finite number of such symbols.
  - At any time the person could either modify a section of paper on which they had written or move to a new section of paper.
  - What the person did would depend on the symbol in the current section and on the person's state of mind.
  - The person could be assumed to be capable of only a finite number of states of mind.
  - To avoid artificial restriction, the amount of paper available for the computation could be assumed to be unlimited.
- No one has since discovered a more powerful model of computation and the consensus is that there is none:

  Turing's Thesis: *The computational power of Turing machines is as great as any possible computational system.*

- Most interestingly, Turing's ideas predated the invention of the digital computer and it was directly from his mathematical models that the first computers were built.

  This is a powerful vindication of the value of theoretical work.