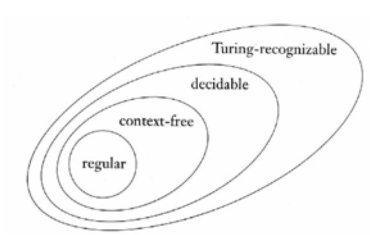
Outline

– Language Hierarchy
– Definition of Turing Machine
– TM Variants and Equivalence
– Decidability
– Reducibility

Outline

– Language Hierarchy
– Definition of Turing Machine
– TM Variants and Equivalence
– Decidability
– Reducibility

Language Hierarchy



•Regular: finite memory

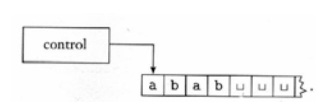
•CFG/PDA: infinite memory but in stack space

•TM: infinite and unrestricted memory

–TM Decidable/Recursive

–TM Recognizable/Recursively Enumerable

Semantics of TM



Alan Turing (1912-1954)

- Not a real machine, but a model of computation
- Components:
  - 1-way infinite tape: unlimited memory
    - Store input, output, and intermediate results
    - Infinite cells
    - Each cell has a symbol from a finite alphabet
  - Tape head:
    - Point to one cell
    - Read or write a symbol to that cell
    - move left or right

States of a TM

- Initial state:
  - Head on leftmost cell
  - input on the tape
  - Blank everywhere else
- Accept state
- Reject state
- Loop
- Accept or reject immediately

Formal Definition

A **Turing machine** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, and Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet, where the *blank* symbol $\sqcup \notin \Sigma$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state.

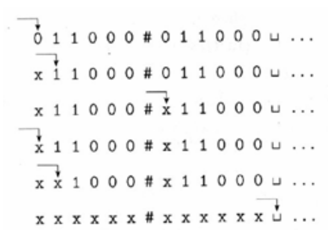
Example of transition function:

$$\delta(q, a) = (p, b, L)$$
$$\delta(q, a) = (p, b, R)$$

An Example

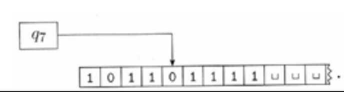
$B = \{w\#w | w \in \{0,1\}^*\}$, and $B = L(M_1)$

- The tape changing:



Configuration

- A configuration of TM:
  - Current state
  - Symbols on tape
  - Head of location
- A formal specification of a configuration:
  - uqv, where
  - u, v are strings on Γ, and uv is the current content on taps
  - q is current state
  - head is in the first symbol of v.
  - ex: 1011 q_7 01111



Configuration

- For two configurations:

  uaq_ibv and uq_jacv, where

  $a, b, c \in \Gamma$, and $u, v \in \Gamma^*$

  uaq_ibv **yields** uq_jacv if $\delta(q_i, b) = (q_j, c, L)$

  uaq_ibv **yields** $uacq_jv$ if $\delta(q_i, b) = (q_j, c, R)$

- Two special cases:
  - the leftmost cell
    - q_ibv yields q_jcv for $\delta(q_i, b) = (q_j, c, L)$
    - q_ibv yields cq_jv for $\delta(q_i, b) = (q_j, c, R)$
  - on the cell with blank symbol
  - uaq_i is equivalent to $uaq_i\sqcup$

Languages

- Turing-recognizable Languages:
  - For a $L \subseteq \Gamma^*$, exists a M such that M **recognizes** L
  - "Recognize" means accept, reject, or loop
- Turing-decidable languages:
  - For a $L \subseteq \Gamma^*$, exists a M such that M **decides** L
  - "Decide" means halting: either accept or reject
- Turing-decidable $\subset$ Turing-recognizable
  - Halting Problem is Turing-recognizable, but not decidable.
- Not all languages are Turing-recognizable
  - There are some languages cannot be recognized by a TM.
    - Complement of Halting problem is Turing-unrecognizable

Configuration

- Initial configuration with input *w:* q_0w
- Accepting configuration: $uq_{accept}v$
- *Rejecting configuration:* $uq_{reject}v$
- $uq_{accept}v$ and $uq_{reject}v$ do not yield any other configurations
  - Immediate effect of accepting/rejecting
  - Halting configurations
- For a TM *M, a* string $w \in L(M)$ if there is a sequence of configurations $C_1, C_2, \ldots C_k$ such that:
  - $C_1 = q_0w$
  - C_i yields C_{i+1} for $1 \le i \le k$
  - $C_k = uq_{accept}v$, $u, v \in \Gamma^*$

An example

$A = L(M_2)$, where $A = \{0^{2^n} | n \ge 0\}$

- Semantical description:

  For an input string w:

  { sweep left to the right along the tape, crossing off every other 0

  if *tape contains single 0*

  { return accepted;}

  elseif *tape contains odd number and more than one of 0s*

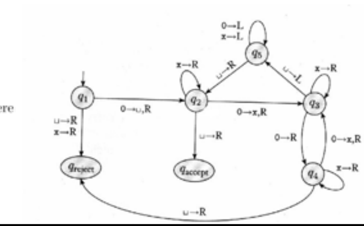
  { return (rejected);}

  else *go back to leftmost cell;*

  }

- Formal description:

  $M_2 = \{Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject}\}$, where
  - $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$
  - $\Sigma = \{0\}$
  - $\Gamma = \{0, x, \sqcup\}$
  - δ: state transition diagram



Outline

– Language Hierarchy
– Definition of Turing Machine
– TM Variants and Equivalence
– Decidability
– Reducibility

Simple variant

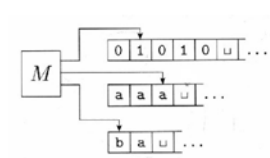
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, RR, LL\}$
- They are equivalent in recognizing language:
  – They can be simulated by original the TM
  – The difference is not significant

TM Variants

- Multitape TM
- Nondeterministic TM
- Enumerators
- Equivalence:All have same power
  – Recognize the same class of languages
  – Can be simulated by an ordinary TM

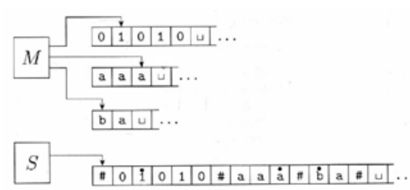
Multitape TM

- A multitape TM is identical to ordinary TM except:
  – k tapes, where $k \geq 1$
  – Each tap has its own head
  – $\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k$
  – $\delta(q_i, a_1, a_2, \ldots, a_k) = (q_j, b_1, b_2, \ldots, b_k, L, R, \ldots, R)$



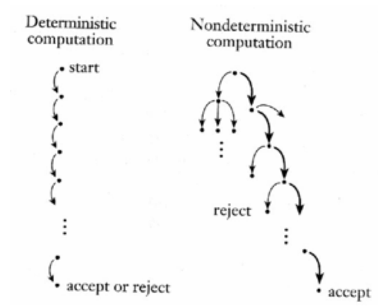
Multitape TM

- <u>Theorem: each multitape TM has an equivalent single tape TM</u>
  - Put # in a single tape for demarcation of original *k* tapes.
  - Each movement of *M* is simulated by a series movement of *S* on each segment.
  - For a right-move on the rightmost cell of *i*th tape in *M*, *S* write blank symbol in *(i+1)th* #, and right-shifts all symbols after that one cell.



NTM

- A computation single path and multi-path in a tree:



Nondeterministic TM

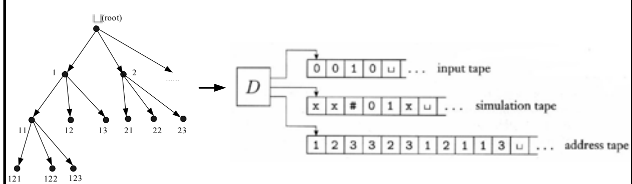
- A nondeterministic TM is identical to an ordinary TM except:
  - $\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$
  - At any point the head has several possibilities to read/write/move.
- In deterministic TM, a computation is a single path with sequence of configurations.
- In nondeterministic TM, a computation is a tree or a directed acyclic graph.
  - A NTM accepts an input string if there exists a path leading to an accept state.
  - If all paths lead to reject state, then this input is rejected.

Nodeterminism

- Is nondeterministic model always equivalent to a deterministic model?
  - Yes, for FA
  - No, for PDA
    - Some CFL cannot be recognized by any DPDA.
  - Yes, for TM!

NTM

- Theorem: *Every NTM has an equivalent DTM.*



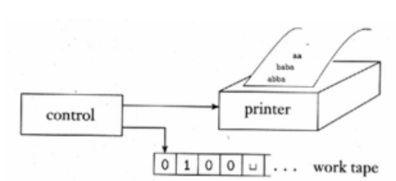
- For a computing tree of a NTM *N* with an input *w*, simulated with a 3-tape DTM *M*:
  - 1st tape: input *w*
  - 2nd tape: tape of a computing path with *N*
  - 3rd tape: node address (finite)

Enumerator

- Theorem: *A language is Turing-recognizable iff some enumerator enumerates it.*
  - For a language, if *E* enumerates it, then construct a TM *M* works as:
    - Run *E*. Every time that *E* outputs a string, compare it with input *w*.
    - If *w* appears in the output of *E*, *accept*.
  - For a language recognized by a TM *M*, construct *E* such that:
    - Run *M* for *i* steps on each input, *s*1, *s*2, …, *si*.
    - If any computations accept, print out the corresponding *sj*.
    - Repeat the above two steps with all possible inputs
- An enumerator can be regarded as a 2-tape TM.
  - Write accepted list on the 2nd tape.

Enumerator

- Semantically, an enumerator is a TM with an attached printer.
- Every time the TM wants to add a string to its output list, it sends the string to the printer.
- The language enumerated by an enumerator *E* is the collection of all the strings that *E* eventually prints out.

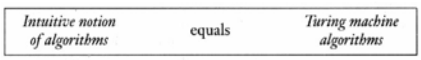


Other Variants

- Write-twice TM
  - Each cell on tape can only be written twice
- Write-once TM
  - Each cell on tape can only be written once
- TM with doubly infinite tape
  - Two-way infinite tape
- Universal TM
  - A TM that takes input of description of another TM.

Thesis

- Church-Turing Thesis:
  - *<u>Any algorithm can be expressed as a TM</u>*
  - Formally defines an algorithm:



- Extended Church-Turing Thesis:
  - *<u>Any polynomial-time algorithm can be expressed as a TM that operates in polynomial time.</u>*
  - A polynomial-time algorithm: number of element operations is a polynomial function of input length.
  - A polynomial-time TM: number of state transition is a polynomial function of input length.

Outline

- Language Hierarchy
- Definition of Turing Machine
- TM Variants and Equivalence
- Decidability
- Reducibility

Describing TM

- Formal description
  - specifying Turing machine's states, transition function, and so on.
- Implementation description
  - using natural language to describe the way that the Turing machine moves its head and the way that it stores data on its tape.
- High-level description
  - using natural language describe an algorithm, ignoring the implementation model.

Solvability

- Solvable:
  - an algorithm to solve it,
  - a TM decides it.
- Unsolvable:
  - not algorithm to solve it
  - no TM can decide it.

Decidable Language

$A_{\mathsf{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts } w\}$

- Acceptance problem:
  - Whether a particular DFA *B* accepts a given input string *w*.
- Membership problem:
  - Another way to say: whether <*B*,*w*> is a member of A_{DFA}.
- Theorem*: A_{DFA} is a decidable language.*

$M =$ "On input $\langle B, w\rangle$, where B is a DFA and w is a string:

1. Simulate B on input w.
2. If the simulation ends in an accept state, *accept*; otherwise, *reject*."

Decidable Language

$A_{\mathsf{REX}} = \{\langle R, w\rangle \mid R \text{ is a regular expression that generates } w\}$

- Theorem: *A_{REX} is a decidable language.*

$P =$ "On input $\langle R, w\rangle$, where R is a regular expression and w is a string:

1. Convert regular expression R to an equivalent DFA A.
2. Run TM M for deciding A_{DFA} on input $\langle A, w\rangle$.
3. If M accepts, *accept*; otherwise, *reject*."

Decidable Language

$A_{\mathsf{NFA}} = \{\langle B, w\rangle \mid B \text{ is an NFA that accepts } w\}.$

- Theorem*: A_{NFA} is a decidable language.*

$N =$ "On input $\langle B, w\rangle$, where B is an NFA and w is a string:

1. Convert NFA B to an equivalent DFA C.
2. Run TM M for deciding A_{DFA} (as a "procedure") on input $\langle C, w\rangle$.
3. If M accepts, *accept*; otherwise, *reject*."

Decidable Language

$E_{\mathsf{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$

- Emptiness test problem:
  - Whether the language of a particular DFA is empty.
- Theorem: *E_{DFA} is a decidable language.*

$T =$ "On input $\langle A\rangle$, where A is a DFA:

1. Mark the start state of A.
2. Repeat Step 3 until no new states get marked.
3. Mark any state that has a transition coming into it from any state that is already marked.
4. If no accept state is marked, *accept*; otherwise, *reject*."

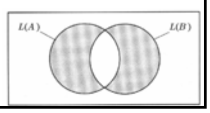
Decidable Language

$EQ_{\mathsf{DFA}} = \{ \langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \}$

- Equivalence problem:
  - Test whether two DFAs recognize the same language.
- Theorem: *EQ_{DFA} is a decidable language.*

$F =$ "On input $\langle A, B\rangle$, where A and B are DFAs:

1. Construct DFA $C = (A \cap \overline{B}) \cup (\overline{A} \cap B)$.
2. Run TM T for deciding E_{DFA} on input $\langle C\rangle$.
3. If T accepts, *accept*; otherwise, *reject*."



Halting Problem

$A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$

Theorem: *A_{TM} is Turing-recognizable*.

$U =$ "On input $\langle M, w\rangle$, where M is a TM and w is a string:

1. Simulate M on input w.
2. If M ever enters its accept state, *accept*; if M ever enters its reject state, *reject*."

- U is an example of universal TM.
- U keeps looping if M neither accepts or rejects.

Other Problems

- A_{CFG} is decidable.
- E_{CFG} is decidable.
- EQ_{CFG} is undecidable.
  - CFG is not closed in intersection and complementation.

- A_{TM} is undecidable.
  - Halting problem
- E_{TM} is undecidable.
- EQ_{TM} is undecidable.

Halting Problem

- Theorem: *A_{TM} is undecidable.*
  - Can be proved by recursive theorem.

Suppose H is a decider for A_{TM}:

$$H(\langle M, w\rangle) = \begin{cases} \textit{accept} & \text{if } M \text{ accepts } w \\ \textit{reject} & \text{if } M \text{ does not accept } w \end{cases}$$

$D =$ "On input $\langle M\rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. If H accepts, *reject* and if H rejects, *accept*."

$$D(\langle \mathsf{M}\rangle) = \begin{cases} \textit{accept} & \text{if } \mathsf{M} \text{ does not accept } \langle \mathsf{M}\rangle \\ \textit{reject} & \text{if } \mathsf{M} \text{ accepts } \langle \mathsf{M}\rangle \end{cases}$$

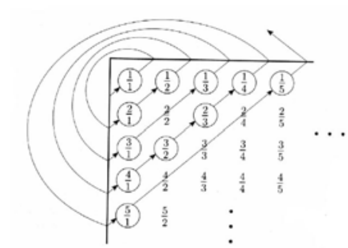
$$D(\langle D\rangle) = \begin{cases} \textit{accept} & \text{if } D \text{ does not accept } \langle D\rangle \\ \textit{reject} & \text{if } D \text{ accepts } \langle D\rangle \end{cases}$$

Unrecognizable

- Theorem: *There are languages that cannot recognized by any TM.*
  - The set of TMs are countable
    - Q, Σ, and Γ are all finite sets
    - Number of transition functions is countable.
  - The set of languages is uncountable.
    - $w \in \Gamma^*$
    - $L \subseteq \Gamma^*$
    - $L \in \mathcal{P}(\Gamma^*)$, $\mathcal{P}(\Gamma^*)$ is uncountable
      - Diagonalization method to prove this

Countable

- Set of position rational numbers is countable: $\{m/n, m, n \in \mathcal{N}\}$



Countable and Uncountable

- Two infinite sets *A* and *B* are the **same size** if there is a correspondence from A to B.
  - A correspondence is a one-to-one and onto function: $f : A \to B$
  - one-to-one: $f(a) \neq f(b)$ whenever $a \neq b$
  - Onto: $\forall b \in B, \exists a \in A, f(a) = b$
- A set is **countable** if either it is finite or it has the same size as N={1,2,3…}; otherwise it is **uncountable**.

Uncountable

- Set of real numbers *R* is uncountable:

Assume that a correspondence f existed between $\mathcal{N}$ and $\mathcal{R}$.

n	$f(n)$
1	$3.\underline{1}4159\cdots$
2	$55.5\underline{5}555\cdots$
3	$0.12\underline{3}45\cdots$
4	$0.500\underline{0}0\cdots$
$\vdots$	$\vdots$

We can find an x, $0 < x < 1$, so that the i-th digit following the decimal point of x is different from that of $f(i)$; for example, $x = 0.4641\cdots$ is a possible choice.

Uncountable

- The set of all languages over an alphabet is uncountable.
  – Think that a real number is a string over alphabet of {. , 0,1,2,3,4,5,6,7,8,9}
  – Similar diagonalization way to prove with general alphabet

$\overline{A_{\mathsf{TM}}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ does not accept } w\}$

- Theorem: $\overline{A_{\mathsf{TM}}}$ *is not Turing-recognizable*
  – If $\overline{A_{\mathsf{TM}}}$ is Turing-recognizable, and A_{TM} is Turing-recognizable, then A_{TM} must be decidable.—contradiction!

- Theorem: *A language is decidable iff both it and its complement language are Turing-recognizable.*
  – If A is decided by M_1, then :
    - M_2="on input w:
      1. Run M_1 on w.
      2. If M_1 rejects, *accept*; if M_1 accepts, *reject*. "
    – M_2 decides $\overline{A}$
  – If A and $\overline{A}$ are Turing-recognizable:

    Let M_1 be a recognizer for A and M_2 be a recognizer for $\overline{A}$.

    $M =$ "On input w:

    1. Run both M_1 and M_2 on input w in parallel. (M takes turns simulating one step of each machine until one of them halts.)
    2. If M_1 accepts, *accept* and if M_2 accepts, *reject*."

Outline

– Language Hierarchy
– Definition of Turing Machine
– TM Variants and Equivalence
– Decidability
– Reducibility

Reducibility

- Semantics
- Reduce A_{TM} to $HALT_{TM}$
- PCP Problem
- Mapping Reducibility