Turing Machine 03

Thumrongsak Kosiyatrakul
tkosiyat@cs.pitt.edu

Combining Turing Machines

- A Turing machine represents an algorithm
- Generally an algorithm can be described as a number of smaller algorithms working in combination
- Similarly, we can combine several Turing machines into a larger one
- Example, two Turing machines T_1 and T_2 sharing the same tape:
  - When T_1 finishes (either in the accept or reject state), T_2 takes over
  - This new machine is represented by T_1T_2

Combining Turing Machines

- Suppose we have two Turing machines:
  - $T_1 = (Q_1, \Sigma, \Gamma, \delta_1, q^1_{\text{start}}, q^1_{\text{accept}}, q^1_{\text{reject}})$ and
  - $T_2 = (Q_2, \Sigma, \Gamma, \delta_2, q^2_{\text{start}}, q^2_{\text{accept}}, q^2_{\text{reject}})$
- Let $T = (Q, \Sigma, \Gamma, \delta, q_{\text{start}}, q_{\text{accept}}, q_{\text{reject}})$ be T_1T_2 which can be constructed as follows:
  - $Q = Q_1 \cup Q_2$ (states of T_2 are relabeled if necessary)
  - Initial state of T is the initial state of T_1 ($q_{\text{start}} = q^1_{\text{start}}$)
  - $\delta = \delta_1 \cup \delta_2$ except those of T_1 that go to q^1_{accept} and q^1_{reject}
  - A transition $q \xrightarrow{x \to y, D} q^1_a$ in T_1 becomes $q \xrightarrow{x \to y, D} q^2_s$ where q^1_a is the accept state of T_1 and q^2_s is the start state of T_2
    - If T_1 enter its accept state, T_2 takes over. The moves that cause T to accept are precisely those that case T_2 to accept
    - However, if T_1 enter the reject state and halt, so does T

Example

- Suppose we want to create a machine that recognize a palindrome (e.g., racecar)
- Suppose we have the following Turing machines:
  - $Copy$: From $\sqcup x$ to $\sqcup x \sqcup x$
  - NB: Moves tape head to the next blank symbol to the right
  - PB: Moves tape head to the next blank symbol to the left
  - R: Reverses the content of the tape from $\sqcup x$ to $\sqcup x^r$
    - x^r is the reverse of a string x
  - $Equal$: Compare two strings separated by a blank symbol
- For simplicity, we put the blank symbol on the first square of the tape to indicate the left-end of the tape.

Example

- Let the content of a tape be $\sqcup x$ where x is a string
- The following machine will accept if x is a palindrome:

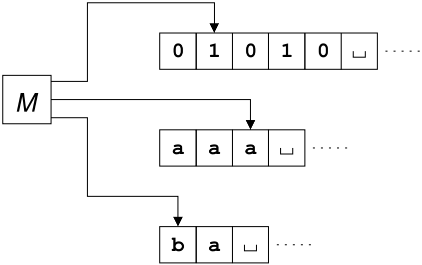
$$Copy \rightarrow NB \rightarrow R \rightarrow PB \rightarrow Equal$$

- Step by Step:

Machine	Tape
Start	$\overset{\downarrow}{\sqcup} x$
$Copy$	$\overset{\downarrow}{\sqcup} x \sqcup x$
NB	$\sqcup x \overset{\downarrow}{\sqcup} x$
R	$\sqcup x \overset{\downarrow}{\sqcup} x^r$
PB	$\overset{\downarrow}{\sqcup} x \sqcup x^r$
$Equal$	$\sqcup x \sqcup x^r \overset{\downarrow}{\sqcup}$

Multitape Turing Machines

- A Turing machine can have multiple tape and tape heads:



- All tape heads can read then write and move in a single Turing machine step

Multitape Turing Machines

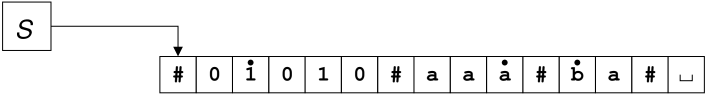
- Transition function need to control/make decision based on symbols read from all tapes
- Example: A transition function of a three-tape TM:

$$\delta(q, x, y, z) \to (r, a, b, c, R, L, R)$$

  - Current state is q, the first tape reads x, the second tape reads y, and the third tape reads z
  - Change the current state to r
  - Write a on to the first tape, write b onto the second tape, and write c onto the third tape
  - Move the first tape head to the right direction, move the second tape head to the left direction, and move the third tape head to the right direction
- Multitape TMs are suitable for algorithms in which several kinds of data are involved

Multitape to One-Tape

- A multitape Turing machine from previous slide, can be convert into one-tape Turing machine as shown below:



- Use # symbol to separate content between tapes
- Use $\dot{x}$ to indicate the current position of each tape head
- One move of multitape machine will be equal to several moves of one-tape machine
- For example,

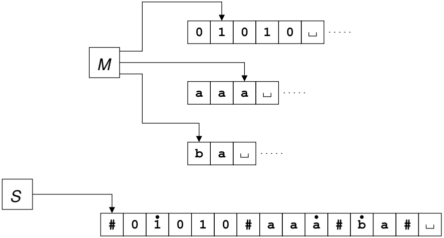
$$\delta(q, 1, a, b) \rightarrow (r, 0, b, a, L, L, R)$$

  will be

  1. Move to the next • on the right, write 0, move to the left square, write • over the symbol, and move to the right square
  2. Move to the next • on the right, write b, move to the left square, write • over the symbol, and move to the right square
  3. Move to the next • on the right, write b, move to the right square, write • over the symbol, and move to the left-end

Multitape to One-Tape

- Recall that a tape will be filled with blank after the last symbol of the string on the tape



- From the above multi-tape TM, if the second tape head needs to move tot he right direction, it should be on top of a blank symbol
  - But on a single-tape TM, it will be on top of the # symbol
  - Single-tape TM must **insert** the blank symbol with a dot at the #
- Every multitape TM has an equivalent single-tape TM (slower)

Nondeterministic Turing Machine

- Similar to Nondeterministic Finite Automata (NFA)
  - Processing one input symbol results in one or more machine.

$$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\})$$

  - Computation is a tree similar to NFA
- For a nondeterministic Turing machine (NTM):
  - If a branch is in the accept state, the machine accepts the input string
  - If all branches are in the reject state, the machine rejects the input string
  - If no branch is in the accept state and at least one branch enter an infinite loop, the machine loops indefinitely on the input string

Theorem 3.16

Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

Proof Idea

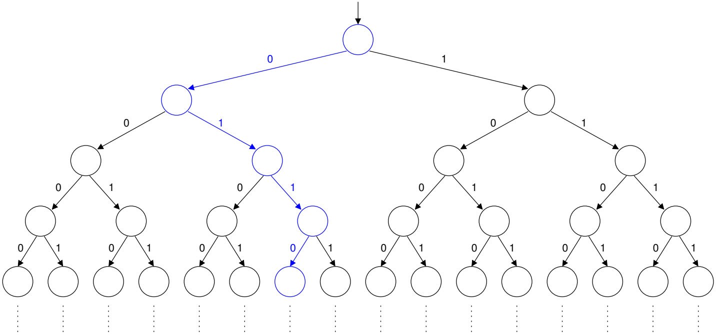
Theorem 3.16 Rewording

For every nondeterministic TM $T = (Q, \Sigma, \Gamma, \delta, q_{\text{start}}, q_{\text{accept}}, q_{\text{reject}})$, there is an ordinary (deterministic) TM $T' = (Q', \Sigma, \Gamma', \delta', q'_{\text{start}}, q'_{\text{accept}}, q'_{\text{reject}})$ with $L(T') = L(T)$.

- Recall that $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\})$
  - Processing a tape alphabet at a state may result in multiple machines
  - The upper bound of the number of machines is
    $$|Q| \times |\Gamma| \times |\{\mathsf{L}, \mathsf{R}\}|$$
  - For simplicity, assume that for every combination of nonhalting state and tape symbol, there are exactly two moves (split to two machines)

Proof Idea

- Computational Tree of a TM on an input



- The branch in blue behaves like a deterministic TM
- The move follows the path in blue can be represented by 0110
  - Use 0110 as a guideline to simulate a branch

Proof Idea

- Suppose a deterministic TM picks a branch and simulate it
  - If that branch ends in the accept state, the NTM accepts the input string
  - If that branch ends in the reject state, no conclusion
    - If another branch is in the accept state, NTM accepts the input string
    - If all other branches are in the reject state, NTM rejects the input string
    - If no branch is in the accept state and at least one branch enter infinite loop, NTM loops indefinitely on the input string
  - If that branch enter infinite loop, the simulation will not end
    - We do not always know that a TM has enter an infinite loop
    - Even if we know that it enters an infinite loop, we still cannot conclude whether NTM accepts or rejects the input string
- Machine T' that simulate an NTM will have to test all possible moves (level order, breadth first search)

Proof Idea

- Machine T' consists of four tapes
  - Tape 1 will be the input string and its contents never change
  - Tape 2 contain the binary string that represents the sequence of moves we are currently testing. (e.g., $0110\sqcup$)
  - Tape 3 is the working tape of a copy of NTM
  - Tape 4 keeps track of all possible reject sequences
- If a sequence of moves result in the accept state, T' accepts the input string
- If all possible sequence of the same length end in the reject state, T' rejects the input string
- If NTM loops indefinitely on the input string, the simulation will also loop indefinitely
- Since T' is a multitape Turing machine, there is an equivalent single-tape Turing machine

Universal Turing Machine

- A universal TM is a TM that can run another TM on an input string
- Imagine a multi-tape TM:
  - Tape 1 contains the formal definition of a TM M, followed by a # symbol, and an input string w
  - Tape 2 will be a working tape for TM M
  - Tape 3 will be used to keep track of the current state of TM M
- Initially:
  - Copy input string w to tape 2
  - Put the start state of TM M onto tape 3
- To run a step, simply search for $\delta(q, a)$ in the formal definition of TM M
  - q is the current state of tape 3
  - a is the symbol under the second tape head

  and update tapes 2 and 3 until tape 3 contains q_{accept} or q_{reject}
- A universal TM will loop indefinitely if the TM that it is running loop indefinitely

The Church-Turing Thesis

- To say that the Turing machine is a general model of computation means that any algorithmic procedure that can be carried out at all, by a human computer or a team of humans or an electronic computer, can be carried out by a Turing machine.
- Note that a Turing machine depends on low-level operations
  - A complex algorithm is simply a series of simple instruction (e.g., assembly) that involve
    - sophisticated logic (state machine) or
    - complex bookkeeping (tape/memory) strategies
- **An algorithm is a procedure that can be carried out by a Turing machine**.