Turing Machines Variants

Variants of Turing Machines

- Many kinds of variants
  - Multiple tapes or nondeterminism
  - Recognize the same class of languages

- Robustness
  - Invariance of results due to changes in design
  - Turing machines have a large degree of robustness
    - Many choices in design can be changed while still producing the same results

- Example
  - Modify transition function to account for transitions that do not move on tape
  - Does not change language recognized by the TM
    - We can simply convert between the types, which means the language is the same
  - To show that to TM models are equivalent
    - Show that one can simulate the other

Multitape Turing Machines

- TM with multiple infinite memory tapes
  - One tape head each
  - Input is initialized to tape 1, with the other tapes blank

- Transition function allows for **some** or **all** of the tapes to move simultaneously

$$\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k$$

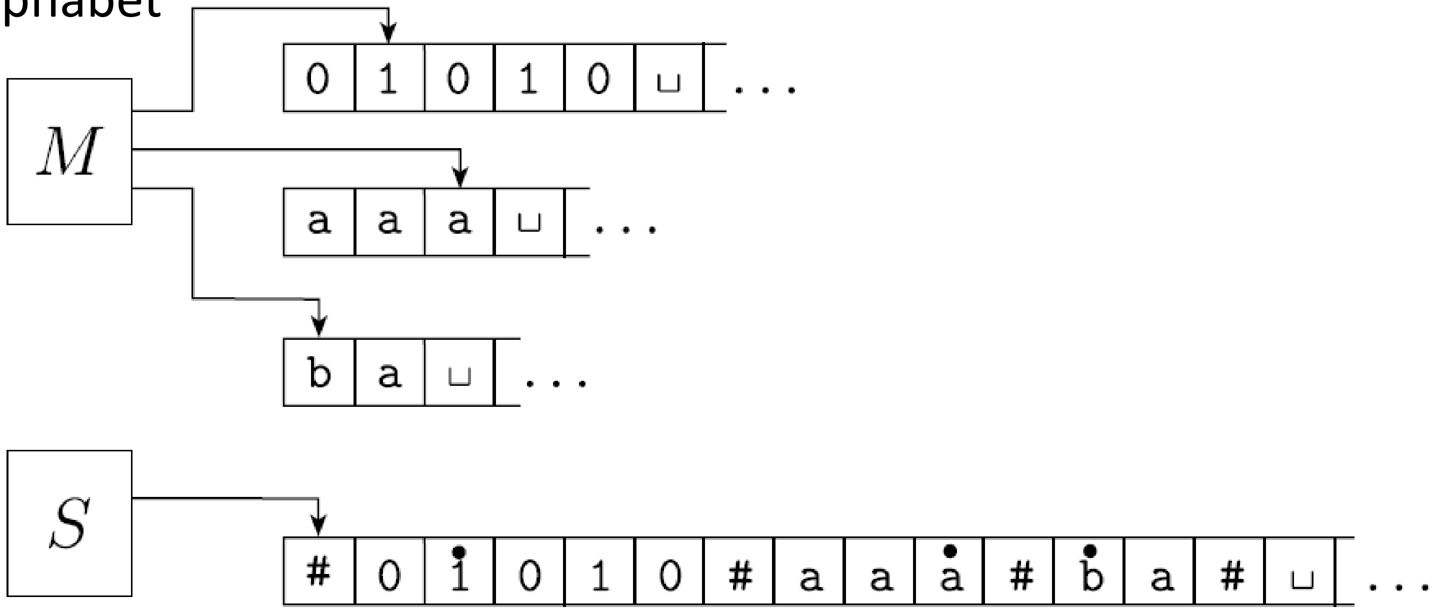
  - k = number of tapes

$$\delta(q_i, a_1, \dots, a_k) = (q_j, b_1, \dots, b_k, L, R, \dots, L)$$

- Theorem:
  - Every multitape TM has an equivalent single-tape TM
  - Equivalent in power, recognizes the same languages
  - A language is Turing-recognizable iff some multi-tape TM recognizes it

Example: Convert Multi-Tape TM to Single-Tape

- Convert mutitape TM M to single-tape TM S
  - Simulate M with S
- If M has k tapes, S can simulate the effects of k tapes on its single infinite tape.
  - Creates virtual tapes and tape heads
  - Use a special symbol (#) to delimit the sections
  - Add modified tape alphabet symbols used to track the tape head of each section
    - Use a “dotted” version of the tape alphabet



Example: Convert Multiple TM to Single-Tape

- Procedure
  1. Given word w = $w_1 \ldots w_n$, TM S initializes the following tape contents:

$$\#\dot{w}_1 w_2 \ldots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \ldots \#$$

  2. The tape head takes a single scan pass to determine the location of the “dotted” symbols
  3. Tape head makes a second pass and updates contents based on <u>transition function of TM M</u>
  4. If one of the virtual heads moves onto the delimiter symbol (#)
     - Shift all affected tape contents over

Nondeterministic Turing Machines

- TM can be explicitly written nondeterministically
  - Multiple possibilities for the same transition
    - Multiple paths, accept if any path reaches accept state
  - Transition Function

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

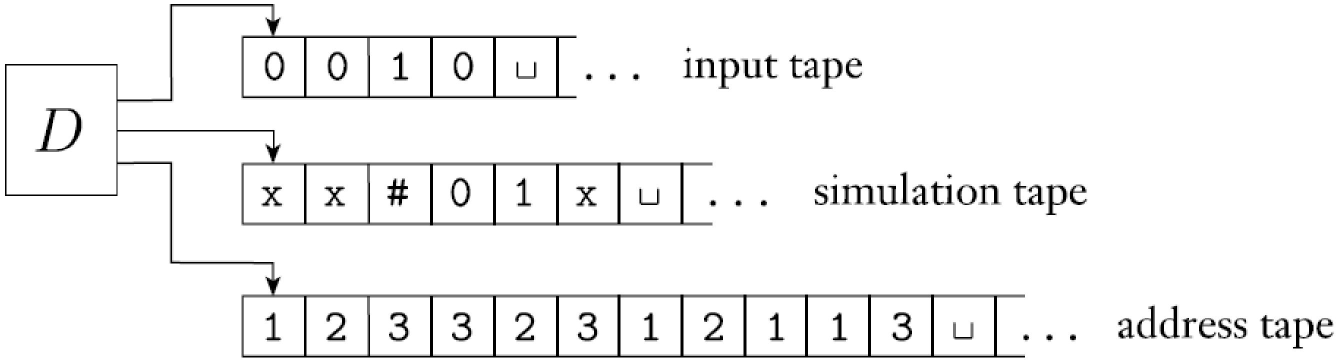
- Theorem
  - Every nondeterministic TM has an equivalent deterministic TM
  - A language is Turing-recognizable iff some nondeterministic TM recognizes it.
- Nondeterministic Decider
  - All branches must halt.
  - A language is Turing-decidable iff some nondeterministic TM decides it.

Example: Nondeterministic TM to Deterministic

- Simulate nondeterministic TM N with a deterministic TM D.
  - D tries all branches of N
  - If D finds any accept branch, D accepts
- Visualize N's computation on an input as a tree
  - D is designed to search tree for an accepting configuration
    - Do not do a **depth-first search** (trace a path 1 at a time)
      - Tracing a path all the way down may never halt.
    - Instead do a **breadth-first search** (trace all branches to the same depth before going to the next depth level)
      - Guarantees halting if an accept state is reached by any branch

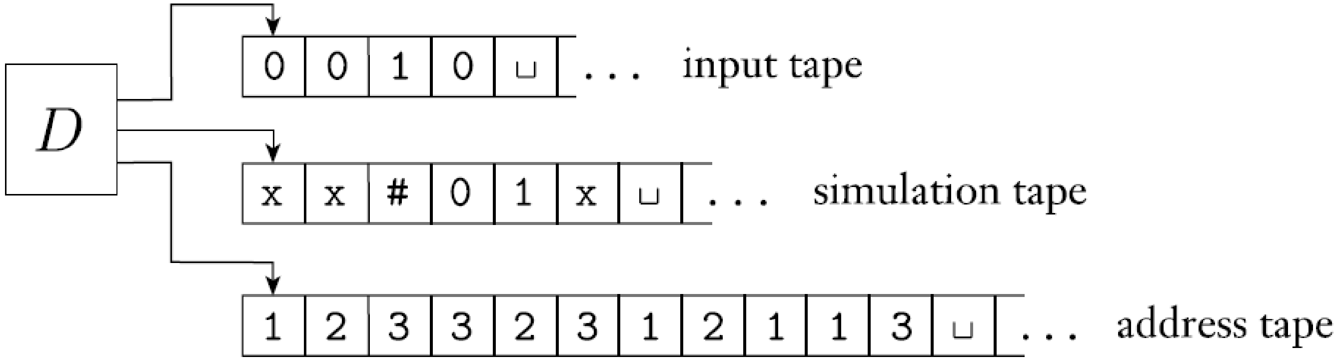
Example: Nondeterministic TM to Deterministic

- Deterministic TM D will have 3 tapes:
  - Tape 1: input string, never altered
  - Tape 2: simulation tape, copy of one of N's branch tapes
  - Tape 3: address tape, tracks location on N's computation tree
    - Each number represents which child to continue to from the current node
    - Ex: 231,
      - starting from root node,
      - go to the 2nd child,
      - then from that node, go to the 3rd child
      - Final end at that node's lst child



Example: Nondeterministic TM to Deterministic

- Procedure
  1. Initialize tape 1 with input, other tapes are blank
  2. Copy tape 1 to tape 2, initialize tape 3 with ε
  3. Use tape 2 to simulate a branch of N
     - Use tape 3 and N's computation tree to move along inputs of tape 2
     - If invalid transition is found, go to step 4
     - If all are valid, accept
  4. Replace string on tape 3 with next string
     1. Go back to step 2



Enumerators

- A TM attached to a printer
  - Every time the TM accepts a string it is sent to the printer

- Starts with a blank input on tape
  - If TM does not halt, may print an infinite number of strings

  - L(Enumerator) = all printable words
    - Recursively enumerable language

- Theorem
  - Language is Turing-recognizable
    iff some enumerator enumerates it.