Lecture 16: Turing Machines & Variants

CSCI 101
Spring, 2019

Kim Bruce

Definition

- Turing machine M is sixtuple (K, Σ, Γ, δ, s, H):
  - K is a finite set of states;
  - Σ is the input alphabet, which does not contain □;
    - □ represents "blank"
  - $\Gamma \supseteq \Sigma \cup \{\square\}$ is the tape alphabet.
  - $s \in K$ is the initial state;
  - $H \subseteq K$ is the set of halting states;
  - δ is ...

Definition (cont)

- δ is the transition function:

  $(K - H) \times \Gamma$ to $K \times \Gamma \times \{\rightarrow, \leftarrow\}$

  non-halting state × *tape char*   *state* × *tape char* × *action (R or L)*

- At each step, look at what is on tape and based on current state, move to new state, write replacement on tape, and move left or right.

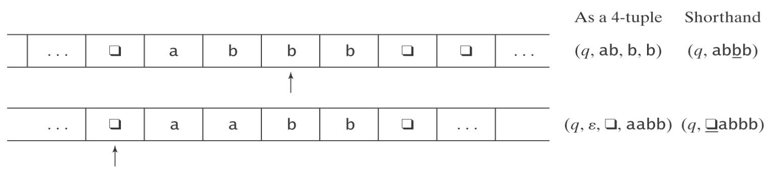
Configurations

- A configuration of a Turing machine M = (K, Σ, Γ, s, H, δ) is an element of:

$$K \times ((\Gamma - \{\square\})\, \Gamma^*) \cup \{\varepsilon\} \times \Gamma \times (\Gamma^* (\Gamma - \{\square\})) \cup \{\varepsilon\}$$

state   *up to scanned square*   scanned square   *after scanned square*

Examples:



Convenient shorthand:

(1) (q, ab, b, b) = (q, ab<u>b</u>b)

(2) (q, ε, □, aabb) = (q, <u>□</u>aabb)

Initial configuration is always (s, <u>□</u>w).

Configurations always finite!!

Computations

- $(q_1, w_1) \vdash_M (q_2, w_2)$ iff (q_2, w_2) follows from (q_1, w_1) via δ in one step.
  - A detailed definition can be given, but intuition clear.
  - $\vdash_M^*$ is the reflexive, transitive closure of $\vdash_M$.
- C_1 *yields* C_2 if $C_1 \vdash_M^* C_2$.
- A *path* is a sequence $C_0, C_1, \ldots, C_n$ s.t. C_0 is initial config and $C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \ldots \vdash_M C_n$.
- A *computation* by M is a path that halts.

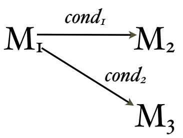
Programming TMs is Hard!

- Define some basic machines
- Symbol writing machines
  - For each x ∈ Γ, define Mx, written just x, to be a machine that writes x.
- Head moving machines
  - R: for each x ∈ Γ, δ(s, x) = (h, x, →)
  - L: for each x ∈ Γ, δ(s, x) = (h, x, ←)

Programming TMs

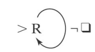
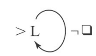
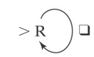
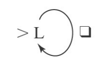
- Machines that simply halt:
  - h, which simply halts.
  - n, which halts and rejects.
  - y, which halts and accepts.

Combining

- M_1 M_2: Run M_1 until it halts, then start M_2 in its start state.
- 

  - Run M_1 until it halts, then based on condition, run M_2 or M_3

Useful Machines

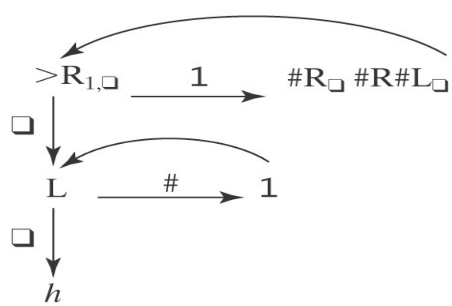
	Find the first blank square to the right of the current square.	$R_{❑}$
	Find the first blank square to the left of the current square.	$L_{❑}$
	Find the first nonblank square to the right of the current square.	$R_{¬❑}$
	Find the first nonblank square to the left of the current square	$L_{¬❑}$

Example

Input: ❑w $w \in \{1\}^*$
Output: ❑w^3

Example: ❑111❑❑❑❑❑❑❑❑❑❑❑❑❑❑

$M =$



TM recognizes languages

- Starting configuration: ❑w❑, w contains no ❑s
- Let M = (K, Σ, Γ, δ, s, {y, n}).
  - M *accepts* w iff (s, ❑w) $\vdash_M^*$ (y, w′) for some w′.
  - M *rejects* w iff (s, ❑w) $\vdash_M^*$ (n, w′) for some w′.

TM recognizes languages

- M *decides* a language $L \subseteq \Sigma^*$ iff for any string $w \in \Sigma^*$:
  - if $w \in L$ then M accepts w, and
  - if $w \notin L$ then M rejects w.
- A language L is *decidable* iff there is a Turing machine M that decides it. In this case, we will say that L is in **D**.

Example

$A^nB^nC^n = \{a^nb^nc^n : n \geq 0\}$
Example: ❑aabbcc❑ *Accepted*
Example: ❑aaccb❑ *Rejected*

1. Move right onto w. If the first character is ❑, halt and accept.
2. Loop:
   2.1. Mark off an a with a 1.
   2.2. Move right to the first b and mark it off with a 2.
   If there isn't one, or if there is a c first, halt and reject.
   2.3. Move right to the first c and mark it off with a 3.
   If there isn't one, or if there is an a first, halt and reject.
   2.4. Move all the way back to the left, then right again past all the 1's (the marked off a's).
   If there is another a, go back to the top of the loop.
   If there isn't, exit the loop.

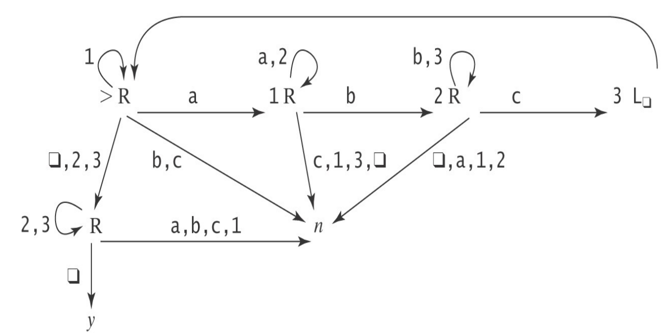
Example

$A^nB^nC^n = \{a^nb^nc^n : n \geq 0\}$
Example: ❑aabbcc❑ *Accepted*
Example: ❑aaccb❑ *Rejected*

3. All a's have found matching b's and c's and the read/ write head is just to the right of the region of marked off a's. Continue moving left to right to verify that all bs and cs have been marked. If they have, halt and accept. Otherwise halt and reject.

Deciding Example

$A^nB^nC^n = \{a^nb^nc^n : n \geq 0\}$
Example: ❑aabbcc❑ *Accepted*
Example: ❑aaccb❑ *Rejected*



Semi-Deciding Language

- M *semidecides* L iff, for any string $w \in \Sigma_M{}^*$:
  - $w \in L \rightarrow$ M accepts w
  - $w \notin L \rightarrow$ M does not accept w.
    M may either: reject or fail to halt.
- L is *semidecidable* iff there is a Turing machine that semidecides it.
- Let **SD** be the set of all *semidecidable* languages.

Example

- Let $L = b^*a(a \cup b)^*$
- We can build M to semidecide L:

1. Loop

   1.1 Move one square to the right.
   If the character under the read head is an a,
   halt and accept, otherwise repeat loop

- Accepts if in, but goes forever otherwise
  - Can easily be decided, too, but just not by this M

Computing Functions

- Let $M = (K, \Sigma, \Gamma, \delta, s, \{h\})$. Its initial configuration is $(s, ❑w)$.
  - Define $M(w) = z$ iff $(s, ❑w) \vdash_M{}^* (h, ❑z)$.
- Let $\Sigma' \subseteq \Sigma$ be M's output alphabet.
- Let $f : \Sigma^* \rightarrow \Sigma'^*$. Say M computes f iff $\forall w \in \Sigma^*$:
  - If w is an input on which f is defined: $M(w) = f(w)$.
  - Otherwise M(w) does not halt.

Recursive Functions

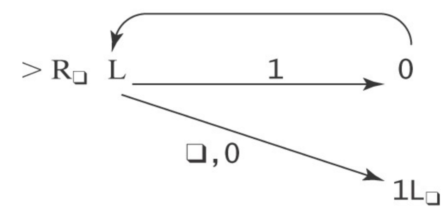
- A (total) function f is *recursive* or *computable* iff there is a Turing machine M that computes it and that always halts.

Numeric Functions

- Let $value_k(n)$ return the nonnegative integer that is encoded, base k, by the string n.
  - For example: $value_8(101) = 65$.
- M *computes* f from $\mathbb{N}^m$ to $\mathbb{N}$ iff, for some k:
  - $value_k(M(n_1;n_2;\ldots n_m)) = f(value_k(n_1), \ldots value_k(n_m))$.

Example

- Example: succ(n) = n + 1
- Represent n in binary. So $n \in 0 \cup 1\{0, 1\}^*$
- Input: ❑n❑❑❑ Output: ❑n+1❑
  ❑1111❑❑ Output: ❑10000❑



Decisions, decisions

- If L is decidable then so is its complement.
  - Why?
- If L and its complement are both semidecidable then L is decidable.

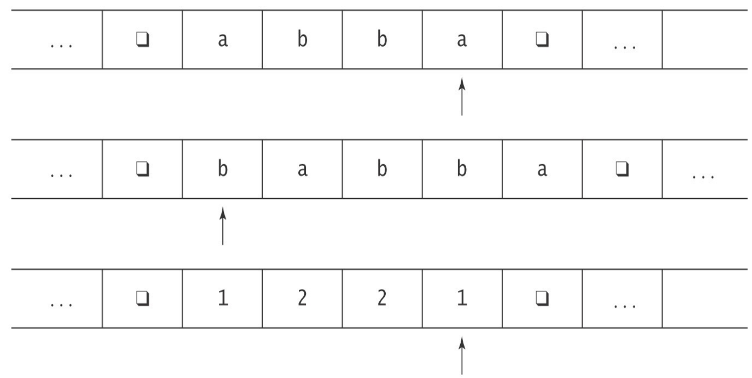
Why such a primitive model?

- TM's are more powerful than FSM, PDAs
- … and are much harder to work with than real computers
- Why?
  - Simplicity makes it easer to reason formally
  - Important that real computers NOT more powerful!

Extensions

- Claim: Every extended TM is equivalent to the basic machine.
- Possible extensions:
  - Multiple tapes
  - Nondeterministic

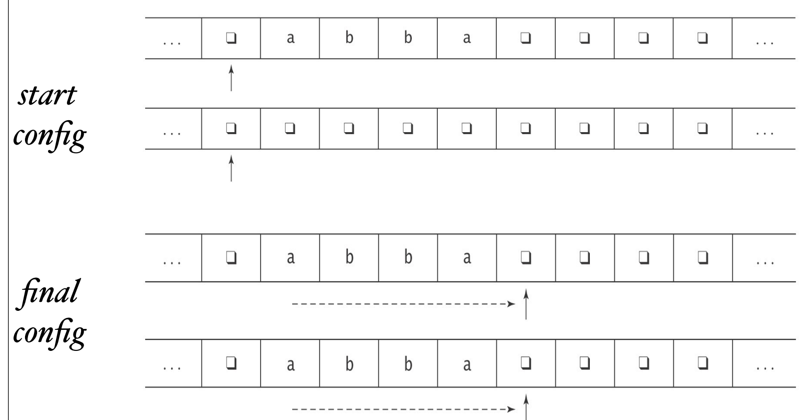
Multiple Tapes



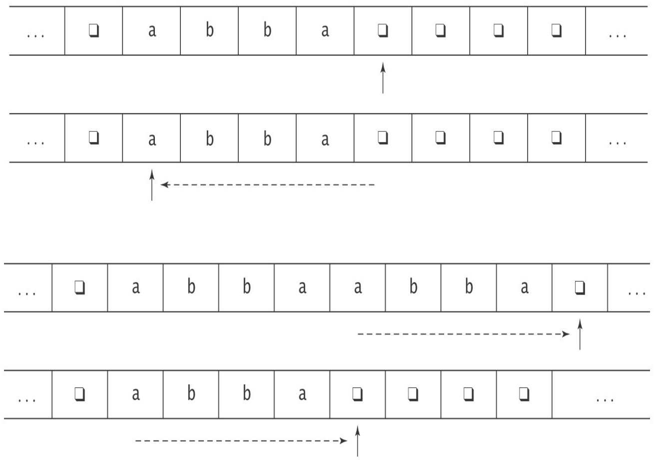
Multitape transitions

- The transition function for a k-tape Turing machine:
  - $$((K-H), \Gamma_1, \Gamma_2, \ldots, \Gamma_k) \quad \textit{to} \quad (K, \Gamma_1', \{\leftarrow, \rightarrow, \uparrow\}, \Gamma_2', \{\leftarrow, \rightarrow, \uparrow\}, \ldots, \Gamma_k', \{\leftarrow, \rightarrow, \uparrow\})$$
- Input: as before on tape 1, others blank.
- Output: as before on tape 1, others ignored.

Copying a String



Copying a String



Another Example

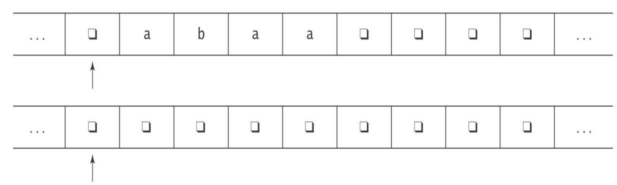
- Adding two numbers
  - Start w/both on first tape
  - Copy one to second tape
  - Move to right of both, start adding

No more power!

- **Theorem**: Let M be a k-tape Turing machine for some $k \geq 1$. Then there is a standard TM M' where $\Sigma \subseteq \Sigma'$, and:
  - On input x, M halts with output z on the first tape iff M' halts in the same state with z on its tape.
  - On input x, if M halts in n steps, M' halts in $O(n^2)$ steps.
- Proof: By construction.

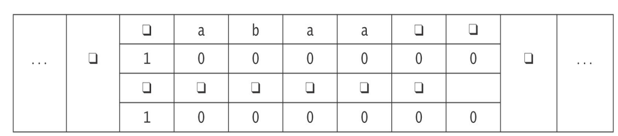
Representation

- Encode:



(a)

- As



(b)

- Alphabet (Σ') of M' = $\Gamma \cup (\Gamma \times \{0, 1\})^k$