Chapter 9 Turing Machines

Definition of TM

TM as Accepters

TM as Transducers/Computer

Turing thesis

Turing’s Vision: the Birth of Computer Science

- Turing's theory forms the basis of computer science
- Combining abstract math theory and mechanical computation, Turing’s theory provided elegant principle on how to use machine to mimic human brain to solve math problem

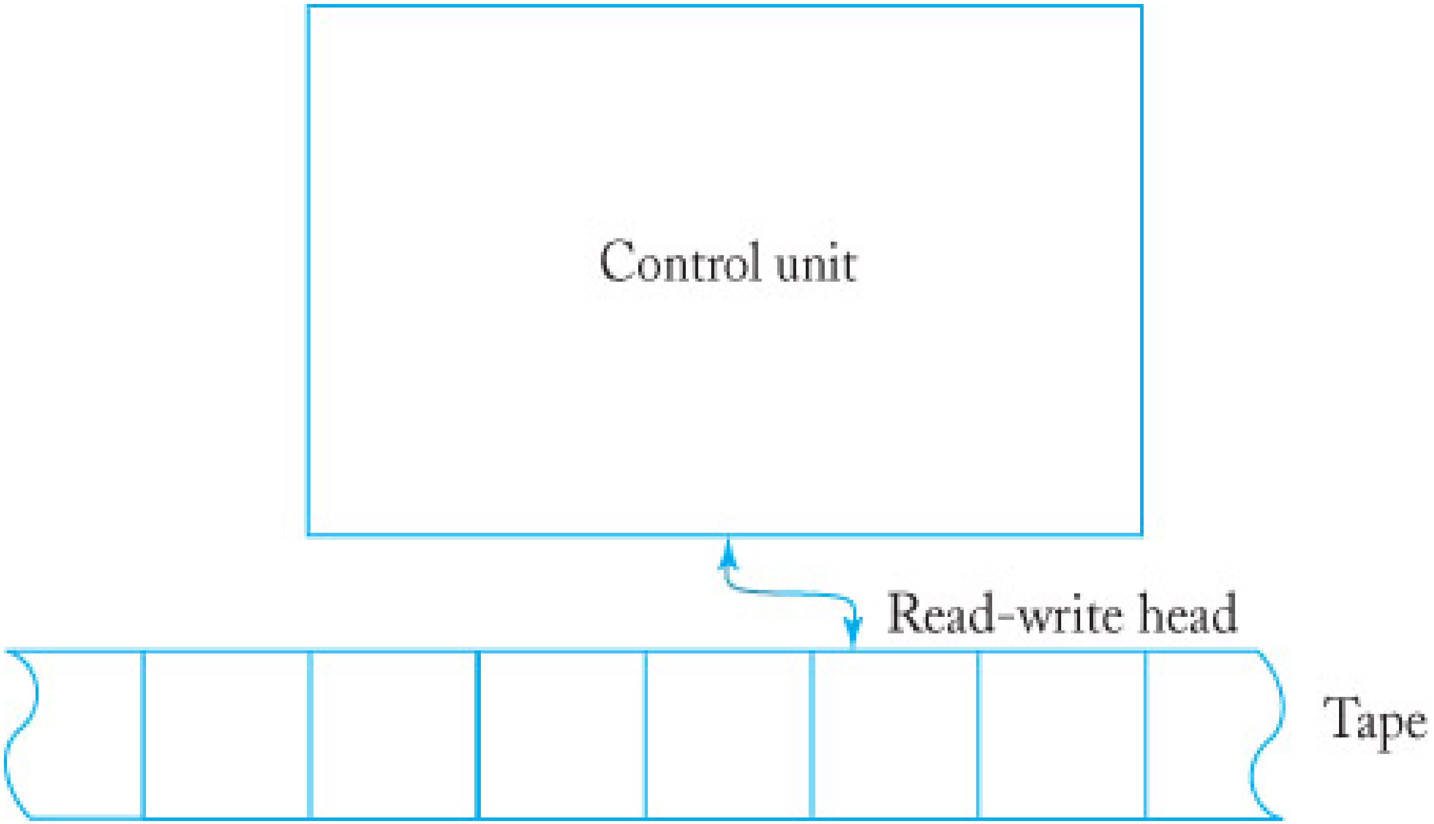
Computing Theory's Foundation

- In 1936, when he was just 24, Alan Turing wrote a remarkable paper in which he outlined the theory of computation, laying out the ideas that underlie all modern computers.
- 3 decision problems to explore the concept of undecidability:
  - To investigates theoretical computing machines, including **Turing machines**;
  - To explain universal machines; and
  - To prove that certain problems are undecidable

Fundamental ideas: automata

Automata	Storage	Accepter	Power level
FA	None	RL	Basic
PDA	Stack	CFL	More powerful
TM	Input/output Tape	Recursively Enumerable Language	Most powerful and flexible

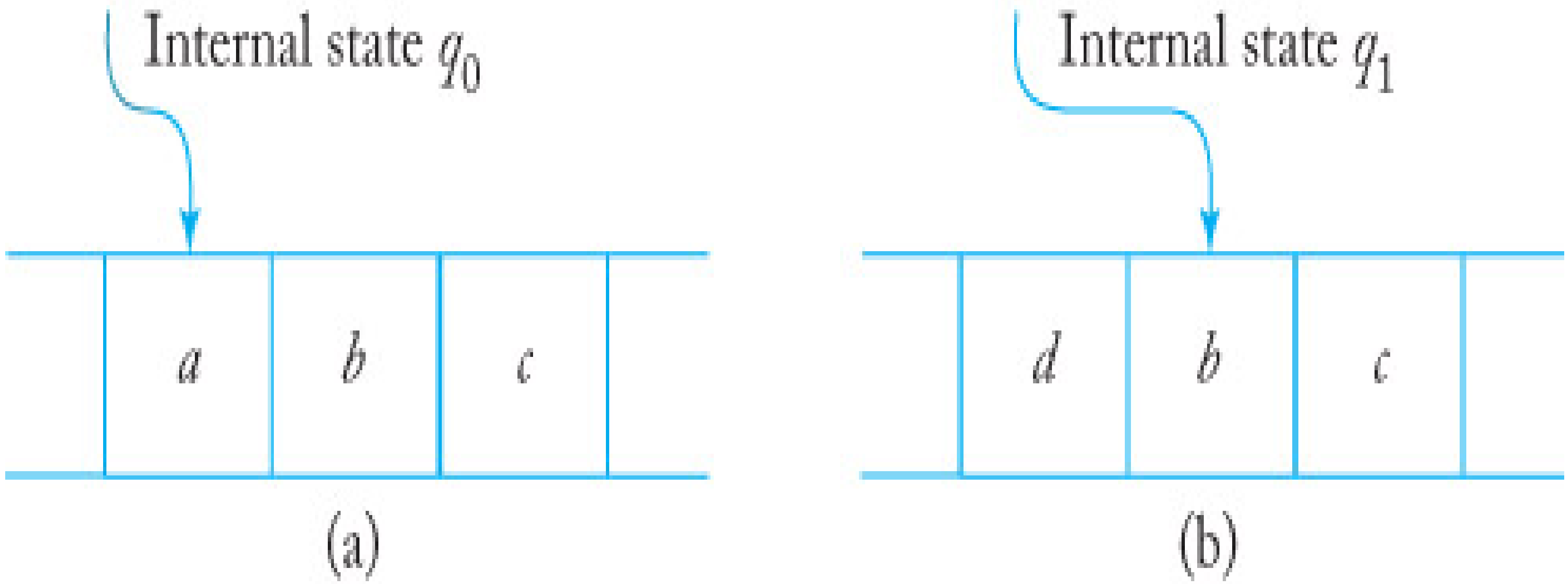
Visualization of a Turing machine



Definition 9.1: TM

- $M = (Q, \Sigma, \delta, \Gamma, q_0, \Box, F)$
- Q, Σ, q_0, and F are the same as they are in FA and PDA
- Γ is a finite set of symbols called the tape alphabet
- $\Box \in \Gamma$ is a special symbol called blank
- $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

Fig 9.2 Before and after the move $\delta\ (q_0, a) = (q_1, d, R)$

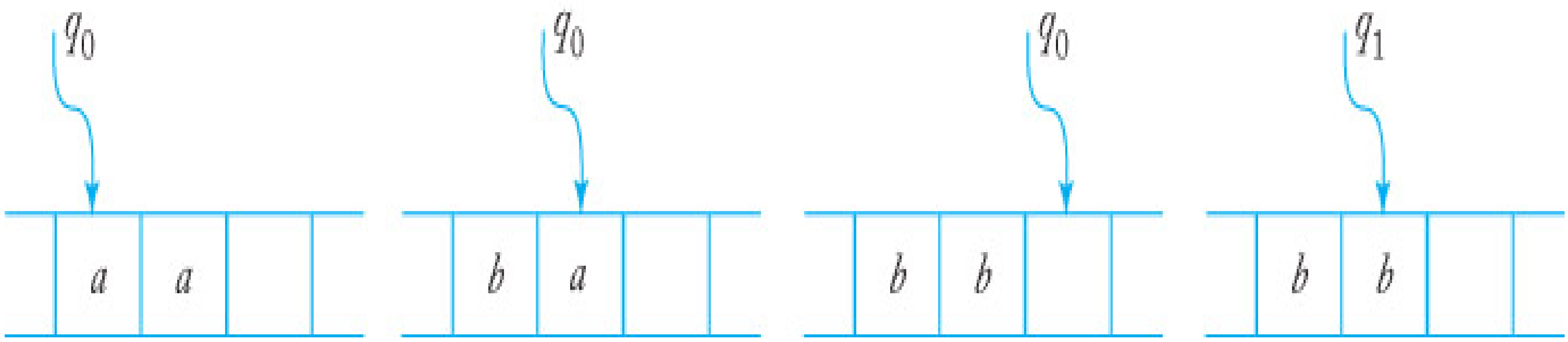


Example 9.2 - 1

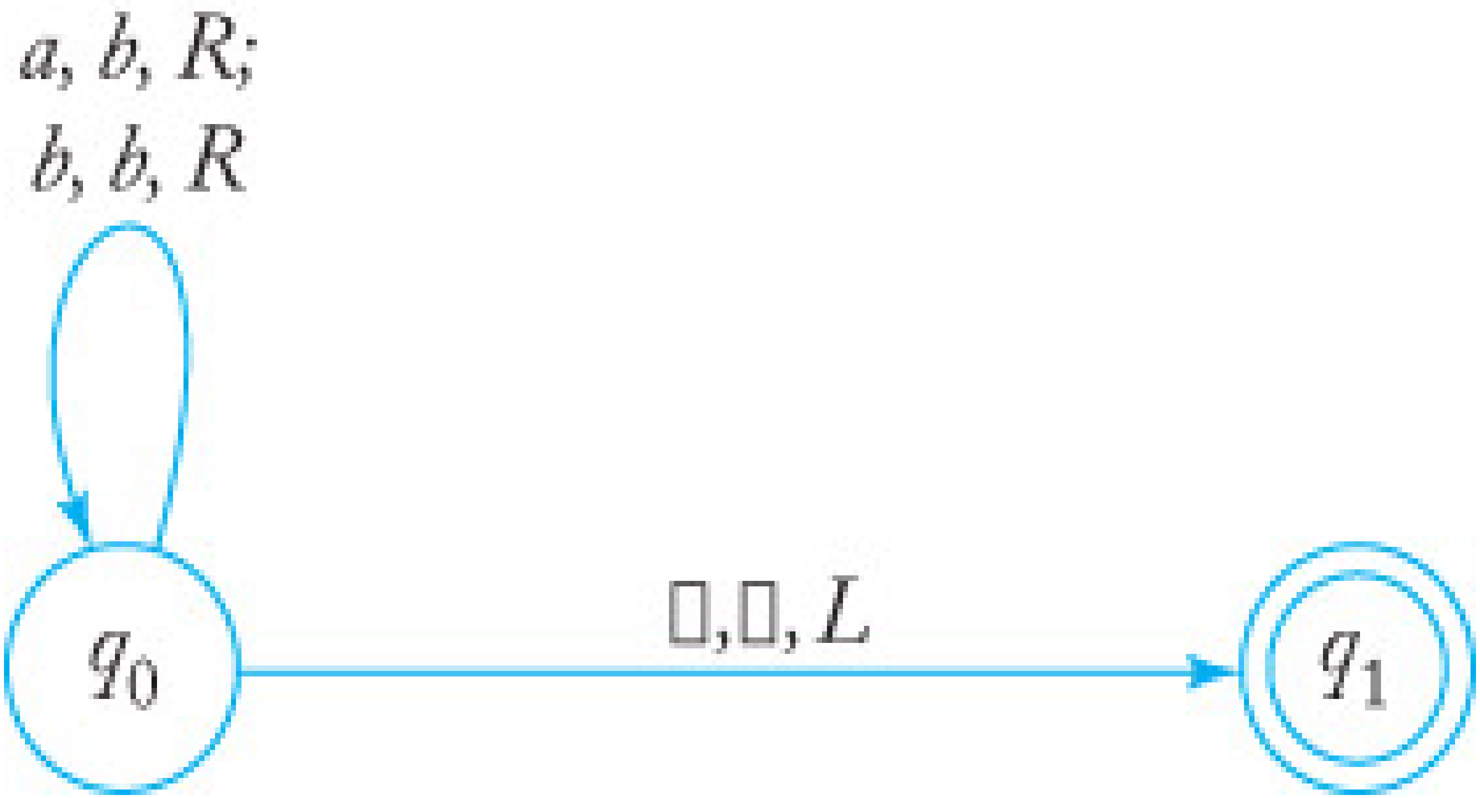
- $Q = \{q_0, q_1\}$,
- $\Sigma = \{a, b\}$,
- $\Gamma = \{a, b, \square\}$,
- $F = \{q_1\}$

Example 9.2 -2

$\delta (q_0, a) = (q_0, b, R)$, $\delta (q_0, b) = (q_0, b, R)$,
$\delta (q_0, \square) = (q_1, \square, L)$

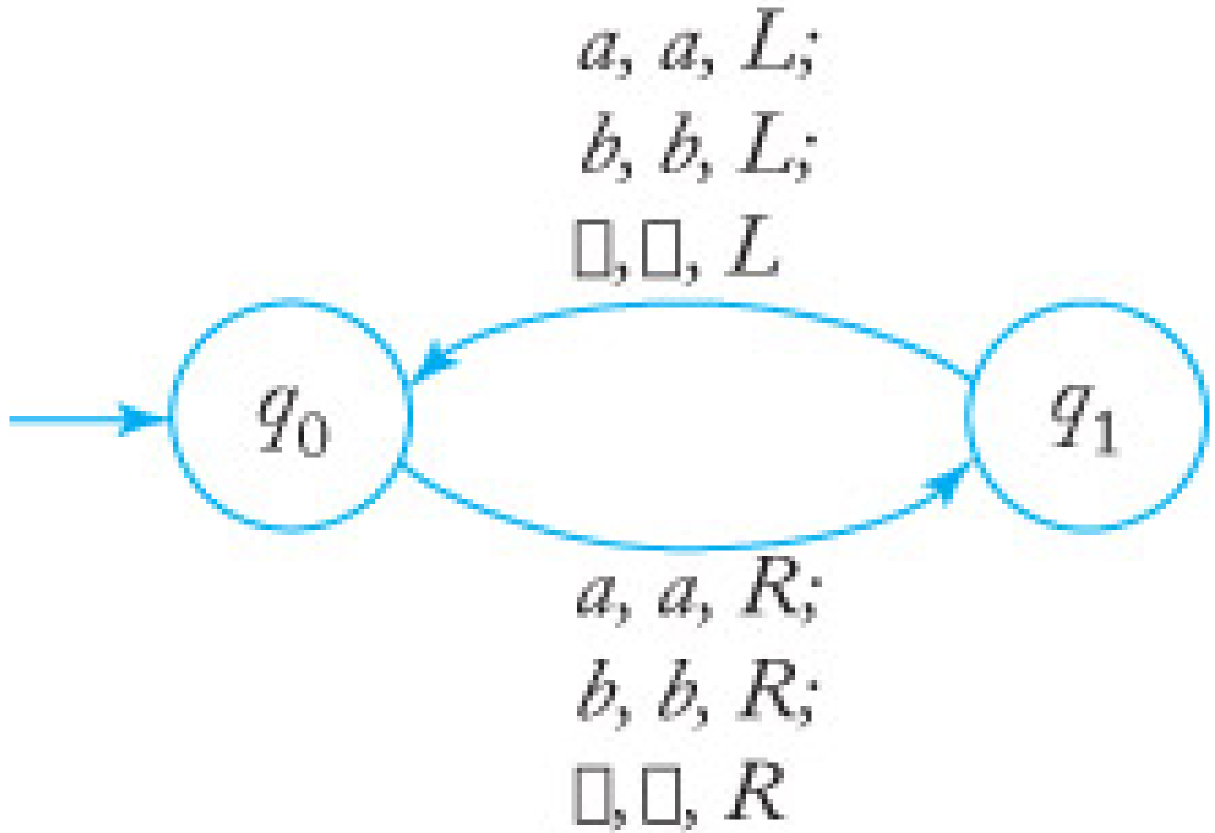


Example 9.2 -3 (Fig. 9.4) Transition Graph Representation



Example 9.3 (Fig. 9.5)

TM that does not halt = TM is in an infinite loop



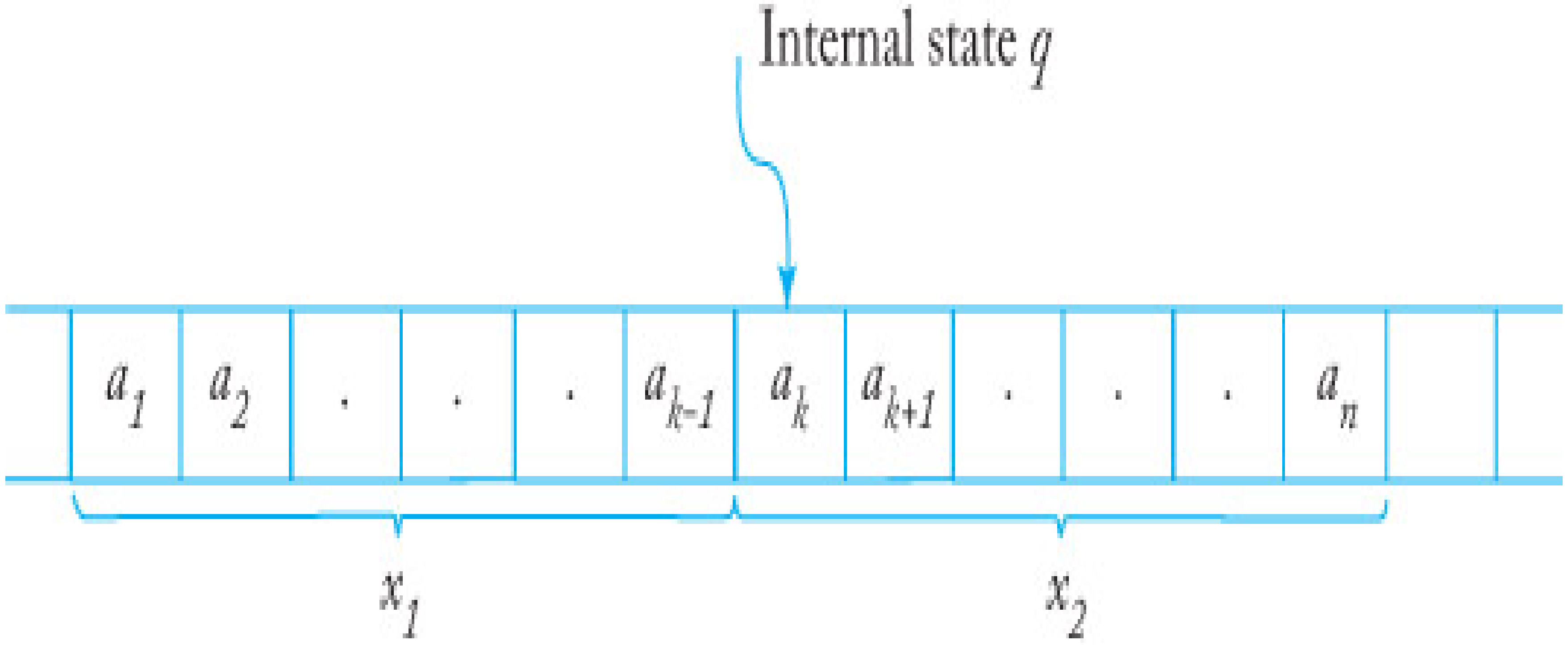
Every TM T over the alphabet Σ divides the set of strings Σ^* into three classes

1. ACCEPT (T) is the set of all strings leading to a final state. This is also called the language accepted by T.
2. REJECT (T) is the set of all strings that halt in a non-final state.
3. LOOP (T) is the set of all other strings that loop forever while running on T.

Standard Turing Machine

1. The TM has a tape that is unbounded in both directions allowing any number of left and right moves
2. The TM is deterministic in the sense that δ defines at most one move for each configuration
3. A tape for reading and writing (input and output)

Notation for a configuration: $x_1 q x_2$



A move from one configuration to another or an arbitrary number of moves

$$\vdash$$

$$\vdash^{*}$$

Definition 9.2 (page 228) Let M be a TM.

- Then any string $a_1 \ldots a_{k-1} q_1 a_k a_{k+1} \ldots a_n$ is an instantaneous description of M.
- A move
- $a_1 \ldots a_{k-1} q_1 a_k a_{k+1} \ldots a_n \vdash a_1 \ldots a_{k-1}\, b q_2 a_{k+1} \ldots a_n$
  - is possible iff $\delta(q_1, a_k) = (q_2, b, R)$
- M is said to halt if for any q_j and a for which $\delta(q_j, a)$ is undefined
- the sequence of configurations leading to a halt state will be called computation .

TM as Language Accepters

- Let M be a TM. Then the language accepted by M is
  - $L(M) = \{w \in \Sigma^+ : q_0\, w \vdash^* x_1 q_f x_2$ for some $q_f \in F$, x1, x2 $\in \Gamma^*\}$

FA $\Rightarrow$ TM

- Every RL has a TM that accepts it
  - Proof (using TG representations in both)
    - For each "a" on $\rightarrow$ in FA, change it to (a, a, R) on $\rightarrow$ in TM;
    - For initial state in FA, let it be the initial state in TM;
    - For final state in FA, let it be a state before going to a halt state with (□, □, R) on $\rightarrow$

Example 9.6
Design a TM for a RL

- $\Sigma = \{0, 1\}$
- Design a TM for RL L = L(00*)
  - Give a TG representation for FA
  - Give a TG representation for TM
  - Note, after obtained the TG for TM we may reduce the number of states
    - See text Example 9.6, L = L(00*), for an equivalent TM design with only 2 states.

Example 9.7 (page 230 - 231)

Design a TM for a CFL $L = \{a^n b^n : n \geq 1\}$

- Algorithm
  - Starting at the leftmost a, we check it off by replacing it with x.
  - We then let the read-write head travel right to find the leftmost b, with in turn is checked off by replacing it with y.
  - We go left again to the leftmost a, replace it with x, and so on.
  - Traveling back and forth, we match each a with a corresponding b.

Example 9.8
Design a TM for non-CFL $L = \{a^n b^n c^n: n \geq 1\}$

- Algorithm
  - Very similar with the algorithm in Example 9.7 although that one is CFL and this is not
  - We match each a, b, c by replacing them in order by x, y, z respectively.
  - At the end, we check that all original symbols have been rewritten.

TM as Transducers/Computers

- TM can do more than as a language accepter
- We can view a TM transducer M as an implementation of a function f defined by
  - $w' = f(w)$
  - Provided that $q_0 w \vdash^*_M q_f w'$, $q_f \in F$

Definition 9.4 (page 232) Turing-computable

- Function f with domain D is said to be Turing-computable or just computable if there if there exists some TM M such that
  - $q_0 w \vdash^*_M q_f f(w),\ q_f \in F$
  - For all $w \in D$

Example 9.9

design a TM to add two positive integers: x and y

- Unary encoding
  - input x=3 and y=2 can be represented by
    - 111011, here 0 is a separator
  - Output x+y=5 can be represented by
    - 11111
  - See the sequence of instantaneous description for adding 111 to 11
    - $q_0 111011 \vdash^* q_4 111110$, q_4 is the final state

In-class Ex.

- Turing Machine as Computer:
  - (a) 3-interger adder;
  - (b) adder for any number of integers

Turing's Thesis

1. Anything that can be done on any existing digital computer can also be done on a TM.
2. No one has yet been able to suggest a problem, solvable by what we intuitively consider an algorithm, for which a TM cannot be written.
3. Alternative models have been proposed for mechanical computation, but none of them is more powerful than the TM model.

Definition 9.5
Algorithm

- An algorithm for a function $f: D \to R$ is a Turing machine M, which given as input any $d \in D$ on its tape, eventually halts with the correct answer $f(d) \in R$ on its tape.
  - $q_0 d \vdash^*_M q_f\, f(d)$, $q_f \in F$
  - For all $d \in D$

References

- https://mitpress.mit.edu/books/turings-vision
- https://www.newscientist.com/article/mg23130803-200-how-alan-turing-found-machine-thinking-in-the-human-mind/
- https://en.wikipedia.org/wiki/Alan_Turing