TURING MACHINES

Chap. 9

SUMMARY

- Introduce a *Turing machine (TM)*, a finite-state control unit to which is attached a one-dimensional, unbounded tape.
- Though a TM is still a very simple structure, it turns out to be *very powerful* and lets us *solve many problems that cannot be solved with a pushdown automaton*.
- This leads to *Turing's Thesis*, which claims that Turing Machines are the *most general types of automata*, in principle as powerful as any computer.

Learning Objectives

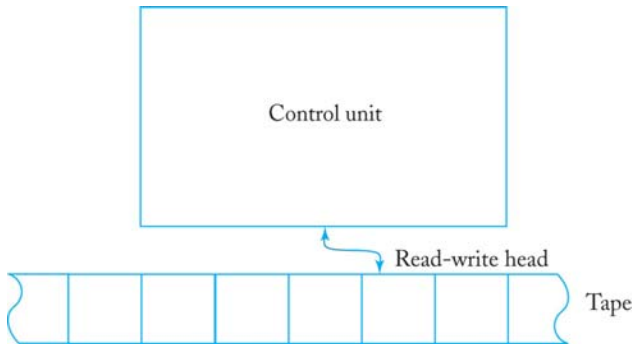
- Describe the components of a standard Turing Machine.
- State whether an input string is accepted by a Turing Machine.
- Construct a Turing Machine to *accept a specific language*.
- Trace the operation of a Turing Machine transducer given a sample input string.
- Construct a Turing Machine to *compute a simple function*.
- State Turing's Thesis and discuss the circumstantial evidence supporting it.

The Standard Turing Machine

- A standard Turing machine has *unlimited storage* in the form of a *tape* consisting of an infinite number of cells, with each cell storing one symbol.
- The *read-write head* can travel in both directions (Left/Right), processing one symbol per move.
- A *deterministic* control function causes the machine to change states and possibly overwrite the tape contents.
- Input string is surrounded by blanks, so the input alphabet is considered a proper subset of the tape alphabet: □*w*□.

Diagram of a Standard Turing Machine

In a standard Turing machine, the tape acts as the input, output, and storage medium.



Definition of a Turing Machine

- Definition 9.1: A *Turing Machine* M is defined by:

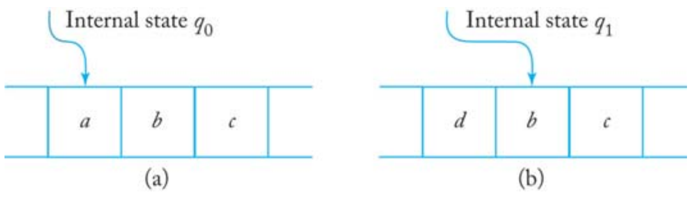
  M = $(Q, \Sigma, \Gamma, \delta, q_0, \square, F)$, where
  - Q : a finite set of internal states
  - Σ : an input alphabet, $\Sigma \subseteq \Gamma - \{\square\}$,
  - Γ : a *tape alphabet*
  - δ : a transition function defined by $\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
  - $\square \in \Gamma$: a special symbol, called the *blank*
  - $q_0 \in Q$: an initial state
  - $F \subseteq Q$: a set of final states
- Output of δ consists of a new state, new tape symbol that replaces the old one, and move symbol that indicates the direction of head move to the *location of the next symbol to be read* (L or R).
- δ is a partial function, so that some (state, symbol) input combinations may be undefined.

Sample Turing Machine Transition

- Example 9.1: the sample transition rule:

  $\delta(q_0, a) = (q_1, d, R)$

- According to this rule, when the control unit is in state q_0 and the tape symbol is a, the new state is q_1, the symbol d replaces a on the tape, and the read-write head moves one cell to the *right.*



The situation (a) before the move and (b) after the move.

A Sample Turing Machine

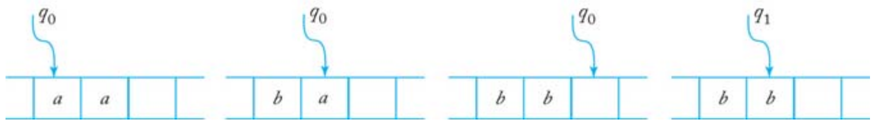
- Example 9.2: Consider the Turing machine defined by

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

  with initial state q_0 and transition function given by:

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

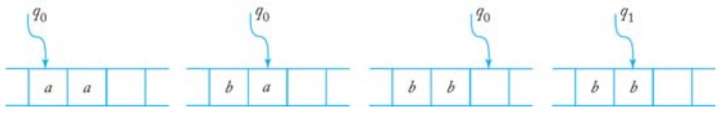
- The machine starts in q_0 and, as long as it reads a's, will replace them with b's and continue moving to the *right*, but b's will not be modified.
- When it encounters the 1st a blank, the control unit switches states to q_1 and moves one cell to the left, then *halt* in final state q_1 .
- The machine halts whenever it reaches a configuration for which δ is not defined (in this case, state q_1)



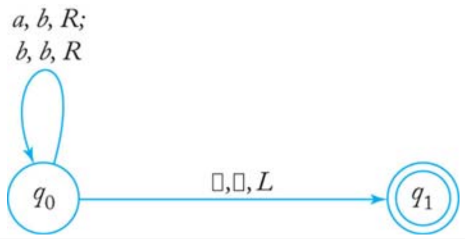
A Sample Turing Machine (cont.)

Example 9.2 (cont.): several stages of processes with initial contents '*aa*' in the tape.

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

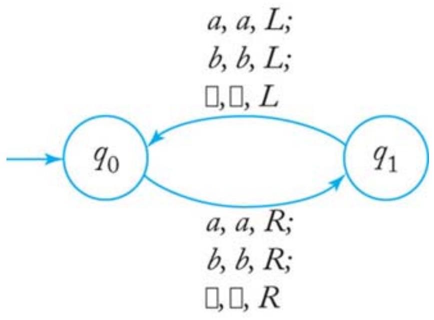


In a Turing machine *transition graph*, each edge is labeled with three items: *current tape symbol, new tape symbol*, and *direction of the head move.*



A Turing Machine that Never Halts

- Example 9.3: It is possible for a Turing Machine to never halt on certain inputs, e.g.) input string *ab*.
- The machine *runs forever* –in an *infinite loop*- with the read-write head moving alternately right and left, but making no modifications to the tape.



Summary: A Standard Turing Machine

Main features of TM model, called a *standard TM:*

1. The TM has a *tape* that is *unbounded in both directions*, allowing any number of *left* and *right moves*.

2. The Turing machine is *deterministic* in the sense that δ defines *at most one move* for each configuration.

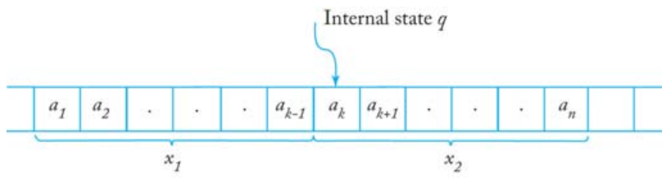
3. There is *no special input file*. We assume that at the initial time the tape has some specified content. Some of this may be considered input. Similarly, there is *no special output device*. Whenever the machine halts, some or all of the contents of the tape may be viewed as output.

Configuration of a Standard TM

- Notation of a configuration of TM, determined by the current state, the contents of the tape, and the position of the read-write head:

$$x_1 q x_2 \quad \text{or} \quad a_1 a_2 \ldots a_{k-1} q a_k a_{k+1} \ldots a_n$$

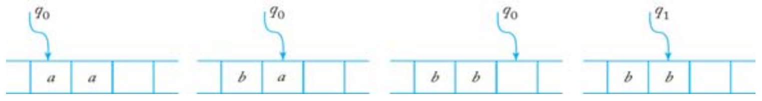
-- an instantaneous description of TM in state q below.



Example: Configuration of a Standard TM

- Example 9.4: the sequence of instantaneous description

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



$\lambda q_0 aa \vdash \quad bq_0a \quad \vdash \quad bbq_0\square \quad \vdash \quad bq_1b.$

- A move from one configuration to another is denoted by $\vdash$:

  $\delta(q_1, c) = (q_2, e, R) \quad \Rightarrow \quad abq_1cd \vdash abeq_2d$

- Example 9.5:

  $q_0aa \vdash bq_0a \vdash bbq_0\square \vdash bq_1b$ *or* $q_0aa \vdash^* bq_1b$

Definition: Configuration of a Standard TM

- Definition 9.2: Let M = $(Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ be a Turing Machine. Then, any string $a_1 \cdots a_{k-1}q_1a_ka_{k+1} \cdots a_n$, with $a_i \in \Gamma$ and $q_1 \in Q$, is an *instantaneous description* of M. A move

  $$a_1 \ldots a_{k-1}q_1a_ka_{k+1} \cdots a_n \vdash a_1 \cdots a_{k-1}bq_2a_{k+1} \cdots a_n$$

  is possible if and only if $\delta(q_1, a_k) = (q_2, b, R)$.

  A move $a_1 \cdots a_{k-1}q_1a_ka_{k+1} \cdots a_n \vdash a_1 \cdots q_2a_{k-1}ba_{k+1} \cdots a_n$

  is possible if and only if $\delta(q_1, a_k) = (q_2, b, L)$.

  M is said to *halt* starting from some *initial configuration* $x_1q_ix_2$ if

  $$x_1q_ix_2 \vdash^* y_1q_jay_2$$

  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 a *computation.*

- The situation that a TM never halts, proceeding in an endless loop: $x_1qx_2 \vdash^* \infty$

The Language Accepted by a TM

- Turing machines can be viewed as language accepters.
- Definition 9.3: Let $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ be a TM.
  Then the language accepted by M is

  $L(M) = \{w \in \Sigma^+ \mid q_0 w \vdash^* x_1 q_f x_2$ for some $q_f \in F$, $x_1, x_2 \in \Gamma^*\}$.

  i.e. the *set of all strings* which cause the machine to *halt in a final state*, when started in its standard initial configuration (q_0, leftmost input symbol)

- A string is *rejected* if
  - The machine *halts in a nonfinal state*, or
  - The machine *never halts (i.e. infinite loop)*

Example: The Language Accepted by a TM

- Example 9.6: For $\Sigma=\{0, 1\}$, design a TM M s.t. L(M) = 00* in REX.

Idea: Starting at the left end of the input,

we read each symbol and check that it is a 0.

If symbol = 0, continue by moving right.

If symbol = □ without encountering anything but 0,

$\rightarrow$ halt in the final state and *accept* the string.

If symbol = 1 anywhere, the string $\notin$ L (00*),

$\rightarrow$ halt in a non-final state. i.e. *reject* the string.

$M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ where

$Q = \{q_0, q_1\}$, $\Sigma=\{a, b\}$, $\Gamma=\{a, b, \square\}$, $F=\{q_1\}$, and the transition rules

$\delta(q_0, 0) = (q_0, 0, R)$, $\delta(q_0, \square) = (q_1, \square, R$ (or L)).

Example: The Language Accepted by a TM (cont.)

- Example 9.7: For $\Sigma=\{a, b\}$, a TM M s.t. $L(M) = \{a^n b^n \mid n \geq 1\}$

  $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ where

  $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $\Sigma=\{a, b\}$, $\Gamma =\{a, b, x, y, \Box\}$, $F=\{q_4\}$.
- the transitions that replaces the leftmost *a* with *x*, then move the head right to the first *b*, replacing it with *y*. When the *y* is written, the machine enters q_2, indicating *a* is successfully paired with *b* :

  $\delta(q_0, a) = (q_1, x, R)$, $\delta(q_1, a) = (q_1, a, R)$,

  $\delta(q_1, y) = (q_1, y, R)$, $\delta(q_1, b) = (q_2, y, L)$,
- the transitions that reverses the direction until *x* is encountered, repositions the head over the leftmost *a*, and returns control to the initial state : $\delta(q_2, y) = (q_2, y, L)$, $\delta(q_2, a) = (q_2, a, L)$, $\delta(q_2, x) = (q_0, x, R)$,

  Then, back in q_0 and ready for the next *a* and *b*.
- Thus, after 1st & 2nd passes, it carries out the partial computation:

  $q_0aa \dots abb \dots b \vdash^* xq_0a \dots ayb \dots b \vdash^* xxq_0 \dots ayy \dots b$, etc.

Example: The Language Accepted by a TM (cont.)

- Example 9.7(cont.): For $\Sigma=\{a, b\}$, a TM M s.t. $L(M) = \{a^n b^n \mid n \geq 1\}$

  $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ where

  $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $\Sigma=\{a, b\}$, $\Gamma =\{a, b, x, y, \Box\}$, $F=\{q_4\}$.
- Thus, after 1st & 2nd passes, it carries out the partial computation:

  $q_0aa \dots abb \dots b \vdash^* xq_0a \dots ayb \dots b \vdash^* xxq_0 \dots ayy \dots b$, etc.
- For $w = a^n b^n \in L(M)$, M *stops* only when there are *no more a's to be erased*. To terminate, a final check is made to see if all *a*'s and *b*'s have been replaced (to detect input where *a* follows *b*):

  $\delta(q_0, y) = (q_3, y, R)$, // all *a*'s are erased.

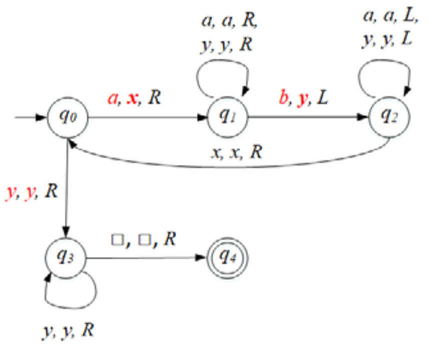
  $\delta(q_3, y) = (q_3, y, R)$,

  $\delta(q_3, \Box) = (q_4, \Box, R)$.
- For $w \notin L(M)$, the computation will *halt* in a *nonfinal state*.

e.g.) $w = a^n b^m \notin L(M)$, $n > m$, M eventually encounters a blank in q_1.
It will halt because no transition $\delta(q_1, \Box)$ is defined for this case.
$w=a^n b^m \notin L(M)$, $n < m$: $\delta(q_3, b)$ undefined but M halts. -- slide #14

Example: The Language Accepted by a TM (cont.)

- Example 9.7(cont.): L(M) = $\{a^n b^n \mid n \geq 1\}$



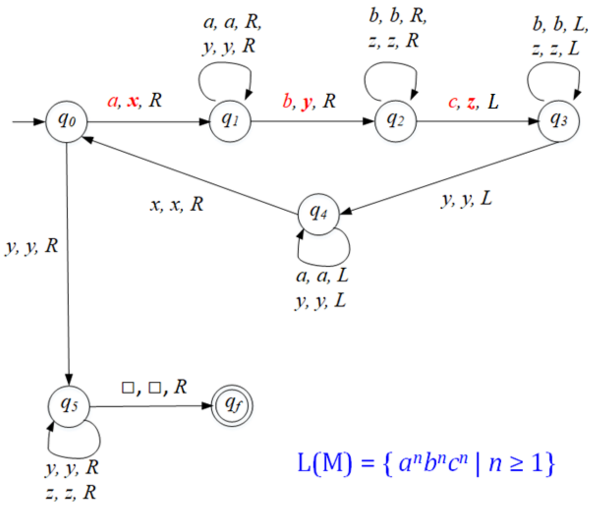
- The transitions with the input *'aabb'*:

$q_0aabb \vdash xq_1abb \vdash xaq_1bb \vdash xq_2ayb$
$\vdash q_2xayb \vdash xq_0ayb \vdash xxq_1yb$
$\vdash xxyq_1b \vdash xxq_2yy \vdash xq_2xyy$
$\vdash xxq_0yy \vdash xxyq_3y \vdash xxyyq_3\,\square$
$\vdash xxyy\,\square q_4\,\square.$

So, aabb$\in$L(M).

Example: The Language Accepted by a TM (cont.)

- Example 9.8: For $\Sigma=\{a, b, c\}$, a TM M s.t. L(M) = $\{\, a^n b^n c^n \mid n \geq 1\}$



L(M) = $\{\, a^n b^n c^n \mid n \geq 1\}$

Turing Machines as Transducers

- Turing Machines provide an *abstract model for digital computers*, acting as a transducer that transforms input into output.
- A *Turing machine transducer* implements a function that treats the *original contents (w)* of the tape as its *input* and the *final contents (w')* of the tape as its *output*.

  $w' = f(w)$ if $q_0 w \vdash^*_M q_f w'$ for some final state q_f.

- Definition 9.4: A function f with domain D is said to be *Turing-computable* or just *computable* if there exists some Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ such that

  $q_0 w \vdash^*_M q_f f(w),\ q_f \in F \quad \forall w \in D$

i.e. A function is *Turing-computable* if it can be carried out by a Turing machine capable of processing all values in the function domain:

Example: Turing Machine Transducer

- Example 9.9: Given two positive integers x and y in unary notation, separated by a single zero, the TM below computes the function $x + y$.
- E.g.) unary notation of x: $w(x) \in \{1\}^+$ s.t. $|w(x)| = x$.
- The transducer has $Q = \{ q_0, q_1, q_2, q_3, q_4 \}$ with initial state q_0 and final state q_4
- The defined values of the transition function are

  $\delta(q_0, 1) = (q_0, 1, R)$ $\quad$ $\delta(q_0, 0) = (q_1, 1, R)$

  $\delta(q_1, 1) = (q_1, 1, R)$ $\quad$ $\delta(q_1, \Box) = (q_2, \Box, L)$

  $\delta(q_2, 1) = (q_3, 0, L)$ $\quad$ $\delta(q_3, 1) = (q_3, 1, L)$

  $\delta(q_3, \Box) = (q_4, \Box, R)$

- When the machine halts, the read-write head is positioned on the leftmost symbol of the unary representation of x + y: $q_0 w(x) 0 w(y) \vdash^* q_f w(x+y) 0$.

Example: Turing Machine Transducer

- Example 9.9(cont.): $q_0 w(x)0w(y) \vdash^* q_f w(x+y)0$.
- The defined values of the transition function are

$$\delta(q_0, 1) = (q_0, 1, R) \qquad \delta(q_0, 0) = (q_1, 1, R)$$
$$\delta(q_1, 1) = (q_1, 1, R) \qquad \delta(q_1, \square) = (q_2, \square, L)$$
$$\delta(q_2, 1) = (q_3, 0, L) \qquad \delta(q_3, 1) = (q_3, 1, L)$$
$$\delta(q_3, \square) = (q_4, \square, R)$$

- f(3,2)=3+2 =5: inputs 3 = 111, 2 = 11 → w = 111011.

$q_0 111011 \vdash 1q_0 11011 \vdash 11q_0 1011 \vdash 111q_0 011$

$\vdash 1111q_1 11 \vdash 11111q_1 1 \vdash 111111q_1 \square$

$\vdash 11111q_2 1 \vdash 1111q_3 10$

$\vdash^* q_3 \square 111110 \vdash q_4 111110$

Example: Turing-Computable Function

- Example 9.10: Design a TM that copies strings of 1's, i.e. TM that performs the computation $q_0 w \vdash^* q_f ww,\ \forall w \in \{1\}^+$.
- Intuitive Process:

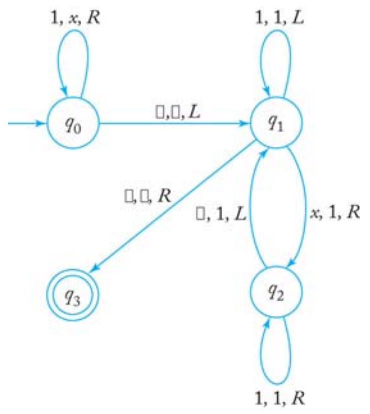
  1. Replace *every* 1 by *x*.

  Repeat

  2. Find the *rightmost x* and replace it with 1.

  3. Travel to the *right end of* the current *nonblank region* and *create 1* there.

  Until there are *no more x's*.



- Computation: $q_0 11 \vdash xq_0 1 \vdash xxq_0 \square \vdash xq_1 x\square$

$\vdash x1q_2 \square \vdash xq_1 11 \vdash q_1 x11 \vdash 1q_2 11 \vdash 11q_2 1 \vdash 111q_2 \square$

$\vdash 11q_1 11 \vdash 1q_1 111 \vdash q_1 1111 \vdash q_1 \square 1111 \vdash q_3 1111$

Example: Turing-Computable Function

- Example 9.11: For integers $x, y > 0$ in unary notation, construct a TM that halts in q_f if $x \geq y$; otherwise halt in a nonfinal state q_n.

  More specifically, $q_0w(x)0w(y) \vdash^* q_fw(x)0w(y)$ if $x \geq y$,

  $q_0w(x)0w(y) \vdash^* q_nw(x)0w(y)$ if $x < y$.

  i.e. Comparer of x and y
- Idea: Match each 1 on the left of the dividing 0 with the 1 on the right and replace them with x.

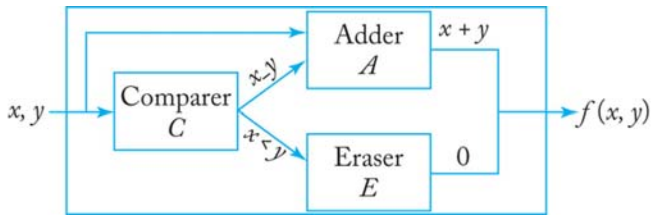
  At the end of the matching,

  either $xx\ldots110xx\ldots x\square$ if $x > y$; or $xx\ldots0xx\ldots x11\square$ if $x < y$.

  Enter q_f; Enter q_n.
- TM can be programmed to make decisions based on arithmetic comparisons.

Combining Turing Machines

- By *combining* Turing Machines that perform simple tasks, *complex algorithms* can be implemented.
- For example, assume the existence of a machine to compare two numbers (comparer), one to add two numbers (adder), and one to erase the input (eraser).
- Figure shows the diagram of a Turing Machine that computes the function $f(x, y) = x + y$ (if $x \geq y$); 0 (if $x < y$)



Example: Combining Turing Machines

- Example 9.12: $f(x, y) = x + y$ (if $x \geq y$); 0 (if $x < y$)
- TM for the comparer C in Example 9.11.
- TM for the adder A in Example 9.9.
- TM for the eraser E, Construct E.

The computations to be done by C are

$q_{C,0}w(x)0w(y) \vdash^* q_{A,0}w(x)0w(y)$ if x ≥ y,

and $q_{C,0}w(x)0w(y) \vdash^* q_{E,0}w(x)0w(y)$ if x < y.

C starts either A or E where $q_{A,0}$ and $q_{E,0}$ as the initial states of A & E.

The computation by A: $q_{A,0}w(x)0w(y) \vdash^* q_{A,f}\, w(x + y)0$,

The computation by E: $q_{E,0}w(x)0w(y) \vdash^* q_{E,f}\, 0$.

Example 9.13: Combining TMs

- Example 9.13: The macroinstruction

  if a then q_j else q_k.
- If the TM reads an a, then it go into state q_j regardless of its current state, without changing the tape content or moving the read-write head. If TM read a symbol $\neq a$, it go into state q_k without changing anything.
- The steps of a TM for its implementation:

  $\delta(q_i, a) = (q_{j0}, a, R)$ $\forall q_i \in Q$,

  $\delta(q_i, b) = (q_{k0}, b, R)$ $\forall q_i \in Q, \forall b \in \Gamma - \{a\}$

  $\delta(q_{j0}, c) = (q_j, c, L)$ $\forall c \in \Gamma$,

  $\delta(q_{k0}, c) = (q_k, c, L)$ $\forall c \in \Gamma$, where

  q_{j0} and q_{k0} are the new (intermediate) states, to handle complications arising from the changes of head position in each move in TM.
- In the macroinstruction, we want to change the state only, not the head position. So, let the head move right, entering into a state q_{j0} or q_{k0}, then go back to desired state q_j or q_k by moving the head left.

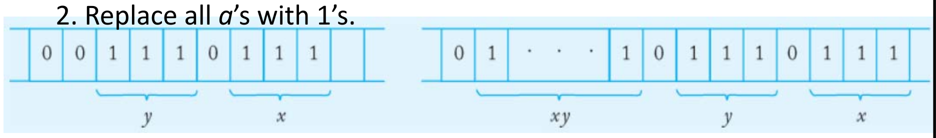
Example 9.14: Combining TMs

Example 9.14: Design a TM that multiplies *x, y* > 0 in unary notation:

$f(x, y) = x \cdot y$

- $x \cdot y = y + y + \ldots + y$, adding *y, x* times, so a TM can be constructed by combining TM of Adder (Ex.9.9) and a Copier (Ex.9.10)
- Assume that the initial/final tape contents in Fig. 9.10.
- The process of multiplication can then be visualized as a repeated copying of the multiplicand *y* for each 1 in the multiplier *x*, whereby the string *y* is added the appropriate number of times to the partially computed product.
- The main steps of process:

```
1. Repeat
      Find a '1' in x and replace it with another symbol 'a'.
      Replace the leftmost 0 by 0y.   // copy y after 0.
   Until x contains no more 1's.
2. Replace all a's with 1's.
```



Turing's Thesis

- How powerful are Turing machines?
- TM for a simple operation $\Rightarrow$ a complex operation by combining the block diagram/psudo-code of simple TMs?
- Ex 9.8: A TM for a non-CFL where $\neg\exists$ PDA. -- L = $\{ a^n b^n c^n \mid n \geq 1\}$
- Ex 9.9, 9.10, 9.11: TM for some simple arithmetic operations, string manipulations and simple comparisons by combining simple TMs.
- Hypothesis: More$^+$ complex operations by combining TMs

  $\Rightarrow$ TM is equal in power to a digital computer?
- Experiment: Take the machine language instruction set of a specific computer and design a TM that can perform all the instructions in the set! Possibly doable if the hypothesis is correct, but not a proof.
- Try to find some procedure for which we can write a computer program, but for which we can show that no TM can exist. If this were possible, we would have a basis for rejecting the hypothesis. But no counterexample yet! Unsuccessful !! $\rightarrow$ an evidence that it cannot be done.
- Every indication is that TMs are in principle as powerful as any computer.

Church-Turing Thesis

- How powerful are Turing machines?
- Turing Thesis: the conjecture by A.M. Turing and others in the mid-1930's.
- *Turing's Thesis* contends that *any computation* carried out by mechanical means can be performed by *some Turing Machine*.
- An acceptance of Turing's Thesis leads to a definition of an algorithm:
- Definition 9.5: An *algorithm* for a function $f : D \to R$ is a Turing Machine M, which given any $d \in D$ on its tape, eventually *halts* with the correct answer $f(d) \in R$ on its tape. Specifically, we can require that
  $q_0 d \vdash^*_M q_f f(d),\ q_f \in F,$ for any $d \in D$.

Evidence Supporting Turing's Thesis

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