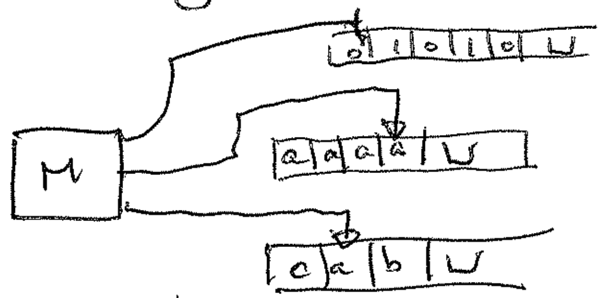
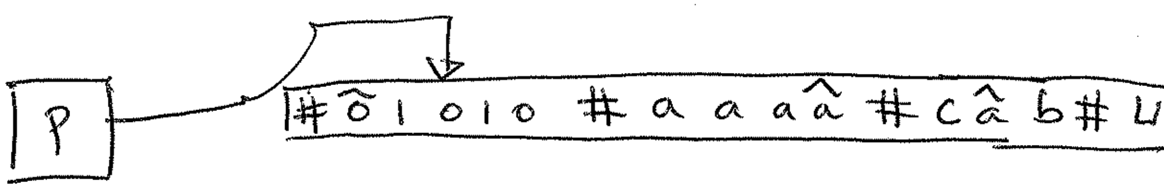
Variants of Turing Machines

Every multitape Turing Machine has an equivalent single-tape turing machine.





Equivalent power

Any two computational models that sastisfy certain reasonable requirements can simulate one another and hence are equivalent in power.

Definition of algorithm

It is a collection of simple instructions for carrying out some task.

polynomial $6x^4yz + 54x^2y^3z - 5$

Example:

Show that D_1 is Turing-recognizable

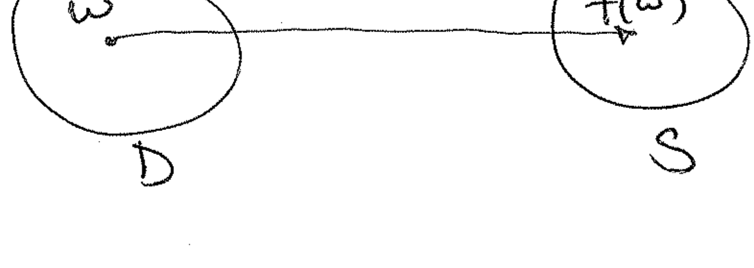
$D_1 = \{ p \mid p$ is a polynomial over x with an integral root $\}$

$M_1 =$ " The input is a polynomial p over the variable x

1. Evaluate p with x set to the values successively to $0, 1, -1, 2, -2, 3, -3, \ldots$ If at any point the polynomial evaluates to 0, accept "

Computing function with TM

A function $f(w)$ has domain D and range S. (result region)



Example: Desig a TM to solve the addition function defined as

$$f(x,y) = x + y$$

where x, y are (+) integers

Solution

unary notation

input string $x0y$

output string $xy0$

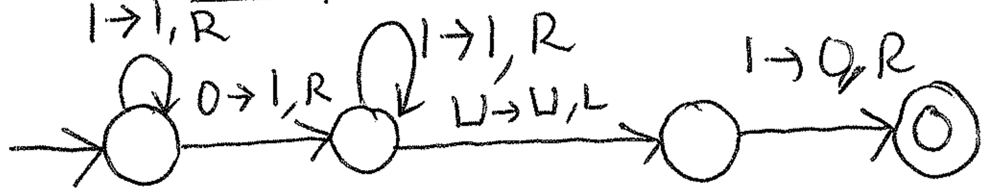
sample $2+3$

input

1	1	0	1	1	1	⊔	⊔

output

1	1	1	1	1	0	⊔	⊔



Example: Design a TM to solve $f(x) = 2x$ where x is an integer

Solution:

Input string x

output string xx

Sample

input

1	1	1	1	1	⊔

output

1	1	1	1	1	1	1	1	1	1	⊔

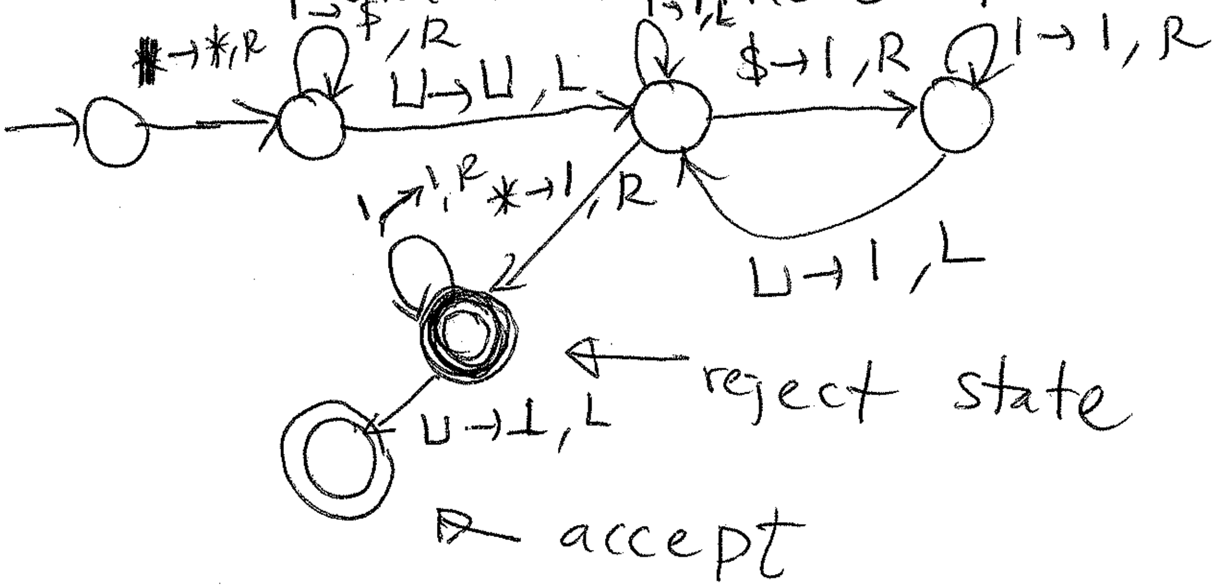
General description

Replace every 1 with $

Repeat

- find rightmost $, replace it with 1 go to right end, insert 1

until no more $ remain



Another example using functions

$$f(X,Y) = \begin{cases} 1 & \text{if } X > Y \\ 0 & \text{if } X \le Y \end{cases}$$

Solution

Input String $X\,0\,Y$

output 0 or 1

general description of TM

Repeat

Match a 1 from X with
a 1 from Y until
all of X or Y is matched

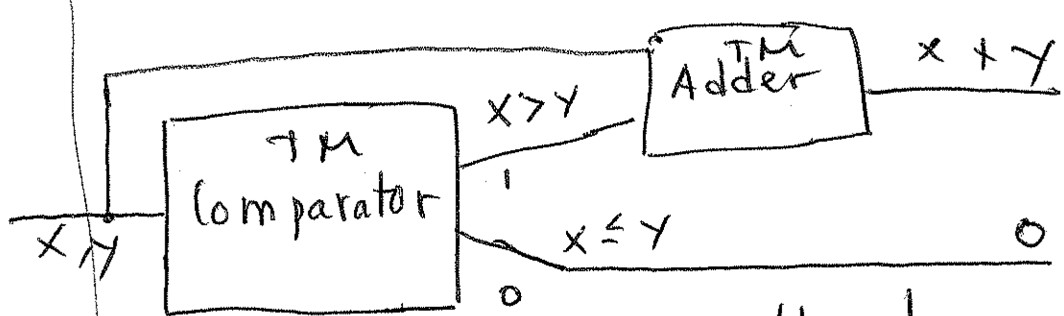
If a 1 or more from X
is not matched, erase tape,
write 1 $(X > Y)$

else
erase tape, write 0
$(X \le Y)$

Combining Turing Machines

Provide a general box
diagram that uses previously
defined TMs to solve

$$f(x,y) = \begin{cases} x+y & \text{if } x>y \\ 0 & \text{if } x \le y \end{cases}$$



Indirect method

Example : (Implementation level description)

Define a TM that decides $\{w \mid w$ contains an equal number of 0s and 1s $\}$

Q = " On input string w

(1) Scan the tape and mark the first 0 which has not been marked. If no unmarked 0 is found, go to stage (4). Otherwise move the head back to the front of the tape.

(2) Scan the tape and mark the first 1 which has not been marked. If no unmarked 1 is found, reject.

(3) Move the head back to the front of the tape and go to Stage 1.

(4) Move the head back to the front of the tape. Scan the tape to see if any unmarked 1s remain. If none are found, accept; otherwise, reject.

A language is recursively enumerable language or Turing-recognizable if some TM recognizes it.

If the TM either rejects or accepts, then the TM is called a <u>decider</u>.

A language is recursive or Turing-decidable or decidable if some TM <u>decides</u> it.

<u>Church-Turing Thesis</u>

Any algorithm can be expressed as a TM.

This formally describes an algorithm.

Decidability
Chapter 4

Problem Solving

Direct Method: (1) formulate the problem statement asking to show that a language is decidable and (2) construct a TM that decides that language

Indirect Method:

(1) Formulate the problem statement asking to show that a language L is decidable

(2) Find an expression of the language in terms of other decidable languages $L_1, \ldots, L_k$ by TMs $M_1, \ldots, M_k$

$$L = E(L_1, \ldots, L_k)$$

where E is constructed using only closure operators

(3) Construct a TM that decides the language L using the TM machines $M_1, \ldots, M_k$

Example:

The problem: consider the acceptance for DFAs

Direct Method:

Test whether a particular FA accepts a given string

$A_{DFA} = \{ \langle B, w \rangle \mid B \text{ is a DFA that accepts } w \}$

Construct a TM that decides A_{DFA}

M = "On input $\langle B, w \rangle$ where B is a DFA and w is a string

1. Simulate B on w as described in chapter 1
2. If the simulation ends in an accept state then accept, if it ends on a nonaccepting state then reject"

```
M: TM (B: DFA, w: string)
begin
    returns (simulate_DFA(B, w));
end;
```