CISC 4090
Theory of Computation

Turing machines

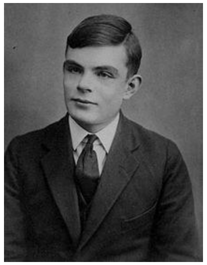
Professor Daniel Leeds
dleeds@fordham.edu
JMH 332

Alan Turing (1912-1954)

Father of Theoretical Computer Science

Key figure in Artificial Intelligence

Codebreaker for Britain in World War I

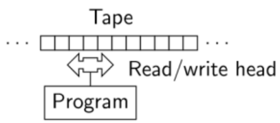


2

Turing machine

Simple theoretical machine

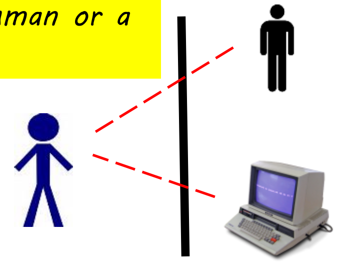
Can do anything a real computer can do!



3

Detour: "Turing test"

A computer is "intelligent" if human investigator can't tell if she's talking to a human or a computer

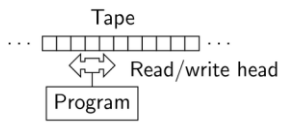


4

Turing machine

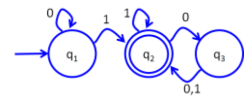
Simple theoretical machine

Can do anything a real computer can do!

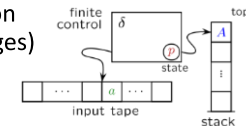


Review of machines

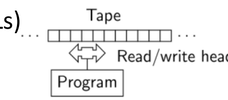
- Finite state automaton (Regular languages)



- Push down automaton (Context free languages)



- Turing machine (beyond CFLs)



Turing machine structure

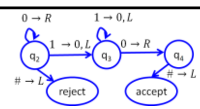
Infinite tape

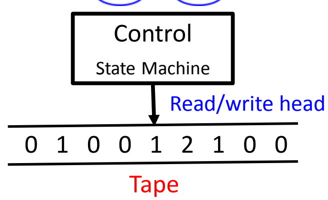
At each step

- Must move left/right on tape
- Can change state
- Can change tape content

When reaches accept or reject state, terminates and outputs “accept” or “reject”

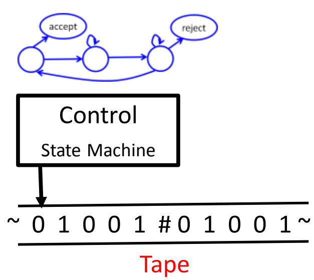
Can loop forever





A Turing Machine for B={w#w|$w \in \{0,1\}^*$}

Assume the string is written on the tape and you start at the beginning of the string. What can we do?



Strategy:

Find left-most 0-or-1 character in first word

If match left-most character in second word, X out both chars

Else reject

If no characters left, accept

~ ~ ~ 0 1 1 0 0 0 # 0 1 1 0 0 0 ~ ~ ~
~ ~ ~ X 1 1 0 0 0 # X 1 1 0 0 0 ~ ~ ~
~ ~ ~ X X 1 0 0 0 # X X 1 0 0 0 ~ ~ ~

How do we move this with single actions:
move-by-one and write?

Turing machine: the formal definition

7 tuple: $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

Q is set of states

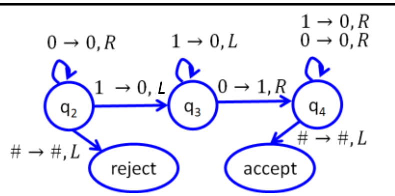
Σ is input alphabet

Γ is the tape alphabet; blank$\in \Gamma$ and $\Sigma \in \Gamma$

$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ transition function

Start, accept, and reject state: q_0, q_{accept}, q_{reject}

The transition function

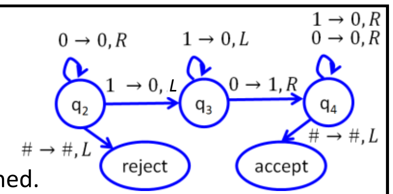


$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

Given state q and symbol a at present location on tape,
change to state r, change symbol on tape to b, move Left or Right

Change in: (state, tape content, head location)
– called "configuration"

The transition function



Example:

Start at q_2. Current position underlined.

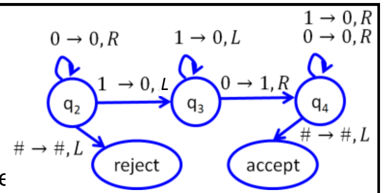
Step 0: q_2 ~ 0 0 $\underline{1}$ 1 # ~ ~ 0 0 1 1 # ~

Step 1: q_3 ~ 0 $\underline{0}$ 0 1 # ~ ~ 0 0 0 1 # ~

Step 2: q_4 ~ 0 1 $\underline{0}$ 1 # ~

Step 3: q_4 ~ 0 1 0 $\underline{1}$ # ~

The transition function



Example:

Start at q_2. Current position underline

Step 0: q_2 ~ 0 0 # 1 # ~ ~ 0 0 # 1 # ~

Step 1: q_2 ~ 0 0 # 1 # ~ ~ 0 0 # 1 # ~

Step 2: $q_?$ Tape: ? ? ?

Step 3: $q_?$ Tape: ? ? ?

16

Strategy: B={w#w | $w \in \{0,1\}^*$}

Find left-most 0-or-1 character in first word

If match left-most character in second word, X out both chars

Else reject

If no characters left, accept

~ ~ ~ 0 1 1 0 0 0 # 0 1 1 0 0 0 ~ ~ ~
~ ~ ~ X 1 1 0 0 0 # X 1 1 0 0 0 ~ ~ ~
~ ~ ~ X X 1 0 0 0 # X X 1 0 0 0 ~ ~ ~

18

Strategy: B={w#w | $w \in \{0,1\}^*$}

Define TM state sequence for each big-picture algorithmic step

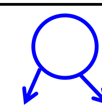
Given character s in left word

1. Move to right word
2. Check if first available symbol in right word == s
3. If match, keep going; else reject

~ ~ ~ 0 1 1 0 0 0 # 0 1 1 0 0 0 ~ ~ ~
~ ~ ~ X 1 1 0 0 0 # X 1 1 0 0 0 ~ ~ ~
~ ~ ~ X X 1 0 0 0 # X X 1 0 0 0 ~ ~ ~

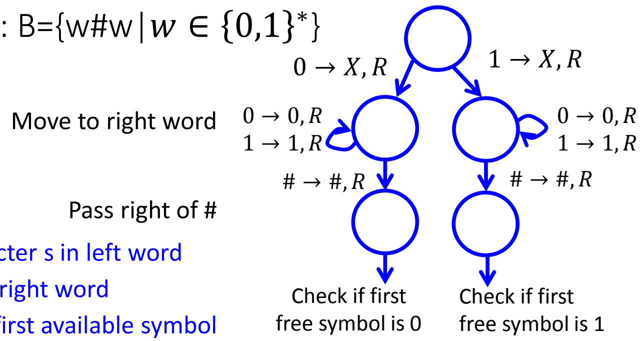
19

Strategy: B={w#w | $w \in \{0,1\}^*$}



20

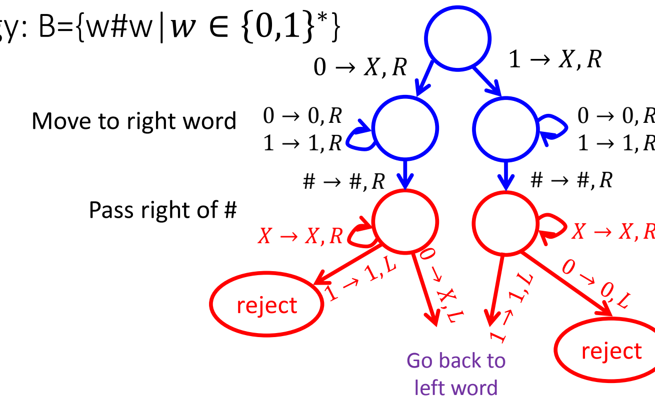
Strategy: B={w#w | $w \in \{0,1\}^*$}



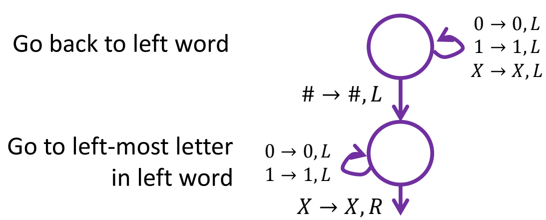
Given character s in left word

1. Move to right word
2. Check if first available symbol in right word == s
3. If match, keep going; else reject

Strategy: B={w#w | $w \in \{0,1\}^*$}



Strategy: B={w#w | $w \in \{0,1\}^*$}



Strategy: B={w#w | $w \in \{0,1\}^*$}

Typical big-picture solution

Find left-most 0-or-1 character in first word

If match left-most character in second word, X out both chars

Else reject

If no characters left, accept

~ ~ ~ 0 1 1 0 0 0 # 0 1 1 0 0 0 ~ ~ ~
~ ~ ~ X 1 1 0 0 0 # X 1 1 0 0 0 ~ ~ ~
~ ~ ~ X X 1 0 0 0 # X X 1 0 0 0 ~ ~ ~

Strategy: B={w#w | $w \in \{0,1\}^*$}

Find left-most 0-or-1 character in first word

If match left-most character in second word, X out both chars

Else reject

If no characters left, accept

~ ~ ~ 0 1 1 0 0 0 # 0 1 1 0 0 0 ~ ~ ~
~ ~ ~ X 1 1 0 0 0 # X 1 1 0 0 0 ~ ~ ~
~ ~ ~ X X 1 0 0 0 # X X 1 0 0 0 ~ ~ ~

Analysis: We will always get an answer (accept or reject), because problem gets smaller after each step

"Turing recognizable" vs. "Decidable"

L(M) – "language **recognized** by M" is set of strings M accepts

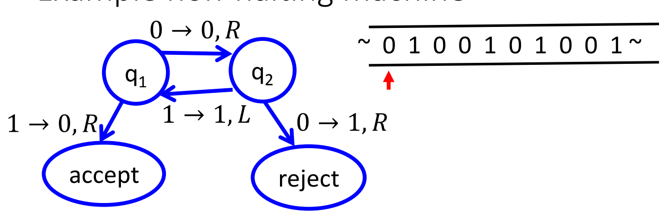
Language is **Turing recognizable** if some Turing machine recognizes it

- Also called "recursively enumerable"

Machine that halts on all inputs is a **decider**. A decider that recognizes language L is said to **decide** language L

Language is **Turing decidable,** or just **decidable**, if some Turing machine decides it

Example non-halting machine



Determining if a machine halts can be hard!

Turing machine structure

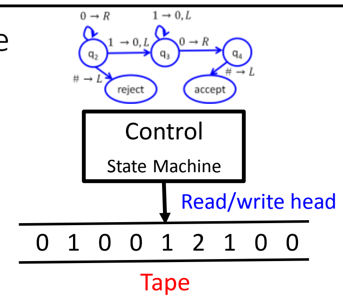
Infinite tape

At each step

- Must move left/right on tape
- Can change state
- Can change tape content

When reaches accept or reject state, terminates and outputs "accept" or "reject"

Can loop forever



Turing Machine for C={0^{2^n} | $n \geq 0$}

Recursive division by 2

Sweep left to right across tape, cross off every-other 0

If

- Exactly one 0: accept
- Odd number of 0s: reject
- Even number of 0s, return to front

Alternating 0s in action:

TM M2 "decides" language C

If you land on a location and want to cross it out, but it is a ~, you crossed out an even number of 0s – do another loop!

If you land on a location and want to skip over it, but it is a ~, you crossed out an odd number of 0s – reject!

Round 1

~ ~ ~ **0** 0 0 0 0 0 0 0 ~ ~ ~
~ ~ ~ X **0** 0 0 0 0 0 0 ~ ~ ~
~ ~ ~ X 0 **0** 0 0 0 0 0 ~ ~ ~
~ ~ ~ X 0 X **0** 0 0 0 0 ~ ~ ~
…
~ ~ ~ X 0 X 0 X 0 X **0** ~ ~ ~
~ ~ ~ X 0 X 0 X 0 X 0 **~** ~ ~
~ ~ ~ X 0 X 0 X 0 X **0** ~ ~ ~
…

Round 2

~ ~ ~ **X** 0 X 0 X 0 X 0 ~ ~ ~
~ ~ ~ X **0** X 0 X 0 X 0 ~ ~ ~
…
~ ~ ~ X X X 0 X X **X** 0 ~ ~ ~

Location of read head marked in **red**

Language D={$a^ib^jc^k$ | k=ixj and i,j,k>0}

Multiplication on a Turing Machine!

Consider 2x3=6

~ ~ ~ a a b b b c c c c c c ~ ~ ~

TM M3 to decide D={$a^ib^jc^k$ | k=ixj and i,j,k>0}

Scan string to confirm form is $a^+b^+c^+$

- if so: go back to front; if not: reject

X out first a, for each b, x off that b and x off one c

- If run out of c's but b's left: reject

Restore crossed out b's, repeat b—c loop for next a

- If all a's gone, check if any c's left
  - If c's left: reject; if no c's left: accept

"Multiply" in action:

TM M3 "decides" language D

Symbol X is an a or c that is gone for good

Symbol y is a b temporarily out of service as you go through all the other b's

Confirm $a^+b^+c^+$

~ ~ ***a*** a b b b c c c c c c ~ ~
~ ~ a ***a*** b b b c c c c c c ~ ~
…
~ ~ a a b b b c c c c c ***c*** ~ ~
…

(a,b) pair one

~ ~ ***a*** a b b b c c c c c c ~ ~
~ ~ X ***a*** b b b c c c c c c ~ ~
…

(a,b) pair two

~ ~ X a B b b ***c*** c c c c c ~ ~
~ ~ X a B b ***b*** X c c c c c ~ ~
…
~ ~ X a B B b X ***c*** c c c c ~ ~

Location of read head marked in **red**

Transducers: generating language

So far our machines accept/reject input

Transduction: Computers transform from input to output

- New TM: given *i* a's and *j* b's on tape, print out *ixj* c's

Transducer: Write c^k , k=ixj, given i a's, j b's,

Scan string to confirm form is a^+b^+

- if so: go back to front; if not: reject

X out first a, for each b, Y off that b and add c to the end

Restore crossed out b's, repeat b—c loop for next a

- If all a's gone, accept

"Transducer" in action:

~ ~ ***a*** a b b b ~ ~ ~ ~ ~ ~ ~ ~
~ ~ X ***a*** b b b ~ ~ ~ ~ ~ ~ ~ ~
…
~ ~ X a B b b c **~** ~ ~ ~ ~ ~
…
~ ~ X a B B **b** c c ~ ~ ~ ~ ~ ~
…
~ ~ X ***a*** b b b c c c ~ ~ ~ ~ ~
…
~ ~ X X B B B c c c c c c **~**

Symbol X is an a that is removed

Symbol y is a b temporarily out of service as you go through all the other b's

Location of read head marked in **red**

TM 4: Element distinctiveness

Given a list of strings over {0,1}, separated by #, accept if all strings are different:

Example: 01101#1011#00010

TM 4 solution

1. Place mark on top of left-most symbol. If it is blank: accept; if it is #: continue, otherwise: reject

2. Scan right to next # and place mark on it. If none encountered and reach blank: accept

3. Zig-zag to compare strings to right of each marked #

4. Move right-most marked # to the right. If no more #: move left-most # to its right and the right-most # to the right of the new first marked #. If no # available for second marked #: accept

5. Go to step 3

TM 4 solution: alternate description

1. Mark left-most un-removed word as wordA; if none available, accept
2. Move to right until reach new un-removed word (if reach blank, loop to step 1)
3. Mark new word as wordB
4. If wordA=wordB, reject; else temporarily remove wordB and continue
5. Loop to step 2

"Distinctiveness checker" in action:

$\underline{0}$ $\underline{1}$ – marked as wordA

0 1 – marked as wordB

X – removed fully

~~0~~ ~~1~~ – temporarily ignored

~ ~ **0** 0 1 # 1 1 # 1 0 1 ~ ~

~ ~ $\underline{0}$ **0** 1 # 1 1 # 1 0 1 ~ ~

…

~ ~ $\underline{0}$ $\underline{0}$ $\underline{1}$ # **1** 1 # 1 0 1 ~ ~

…

~ ~ $\underline{0}$ $\underline{0}$ $\underline{1}$ # *1* *1* **#** 1 0 1 ~ ~

…

~ ~ $\underline{0}$ $\underline{0}$ $\underline{1}$ # ~~1~~ ~~1~~ # *1* *0* *1* **~** ~

…

~ ~ X X X # $\underline{1}$ $\underline{1}$ # *1* **0** 1 ~ ~

Location of read head marked in **red**

Decidability

How do we know decidable?

- Simplify problem at each step toward goal
- Can prove formally – number of remaining symbols at each step

Showing language is Turing recognizable but not decidable is harder

42

Many equivalent variants of TM

- TM that can “stay put” on tape for a given transition
- TM with multiple tapes
- TM with non-deterministic transitions

Can select convenient alternative for current problem

43

“Stay put” TM equivalent to Traditional TM

Design TM to simulate Stay Put TM as follows:

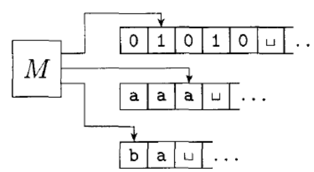
IF $\delta_{spTM}(q_i, a) = \left(q_j, b, L\right)$, THEN: $\delta_{TM}(q_i, a) = \left(q_j, b, L\right)$
IF $\delta_{spTM}(q_i, a) = \left(q_j, b, R\right)$, THEN: $\delta_{TM}(q_i, a) = \left(q_j, b, R\right)$

IF $\delta_{spTM}(q_i, a) = \left(q_j, b, StayPut\right)$,
THEN: $\delta_{TM}(q_i, a) = (q_{new}, b, R)$ and
$\delta_{TM}(q_{new}, z) = (q_{new}, z, L)\ \forall z \in \Gamma$

46

MultiTape TM

- Each tape has own ReadWrite Head
- Initially tape 1 has input string, all other tapes blank
- Transition does read/write on all heads at once



47

Equivalence of SingleTape and MultiTape TM

Convert *k* tape TM M to single tape TM S

- Contents of M's tapes separated by # on S's tape
- Mark current location on each tape

- Read stage: sweep through all *k* tapes to check input
- Write stage: sweep through all *k* tapes to write output **and** update marker (read head) locations
- Head location out of range?
  - Add new position to relevant tape, shift all other characters to right

48

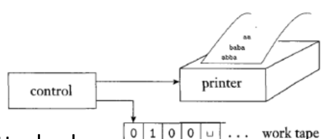
Equivalence of Deterministic and Nondeterministic TMs

- Try all possible non-deterministic branches – breadth first search
- DTM accepts if NTM accepts

- Can use three tapes: 1 for input, 1 for current branch, 1 to track tree position

49

Enumerators



Enumerator E is TM with printer attached

- TM can send strings to be output by printer
- Input tape starts blank
- Language enumerated by E is collection of strings printed
- E may print infinitely

Theorem: A language is Turing-recognizable iff some enumerator enumerates it

50

Proof of enumerator equivalence

If enumerator E enumerates language A, TM M recognizes it

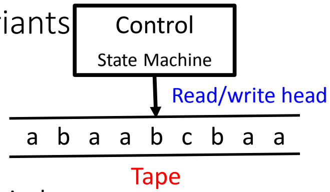
- For every w generated by E, M runs E and checks if w in output

If TM M recognizes language, A, can construct enumerator E for A:

- s1, s2, s3, … be list of all possible strings
- For i=1,2,…
  - Run M for i steps on s1, s2, …, si
  - If string accepted, print it

51

Common themes in TM variants



- Unlimited access to unlimited memory
- Finite work performed at each step

Note, all programming languages are equivalent

- Can write compiler for C++ in Java

An Algorithm

is a collection of simple instructions for carrying out some task

Hilbert's Problems

In 1900, David Hilbert proposed 23 mathematical problems

Problem #10

- Devise algorithm to determine if a polynomial has an integral root.
- Example: $6x^3yz^2+3xy^2-x^3-10$ has root x=5, y=3, z=0

General algorithm for Problem 10 does not exist!

Church-Turing Thesis

- Intuition of thesis: algorithm == corresponding Turing machine
- Algorithm described by TM also can be describe by λ–calculus (devised by Alonzo Church)

Hilbert's 10th problem

Is language D decidable, where D={p | p is polynomial with integral root}

Devise procedure:
- Try all ints, starting at 0: x=0, 1, -1, 2, -2, 3, -3, …
- You may never terminate – so not decidable

Exception: univariate case for root **is** decidable

56

Levels of description

For FA and PDA
- Formal or informal description of machine operation

For TM
- Formal or informal description of machine operation
- **OR** just describe algorithm
  - Assume TM confirms input follows proper tape string format

57

Graph connectivity problem

Let A be all strings representing graphs that are connected (any node can be reached by any other)

A={<G> | G is connected undirected graph}

Describe TM M to decide language

Algorithm:
1. Select and mark first node of G
2. Repeat below until no new nodes marked:
   - For each node in G, mark if it is attached to already-marked node
3. Scan all nodes of G – if all marked, accept; else, reject

58