# CISC 4090
# Theory of Computation

Turing Machines, continued:
Transducers, MultiTape, NonDeterminism

Professor Daniel Leeds
dleeds@fordham.edu
JMH 332

## "Turing recognizable" vs. "Decidable"

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

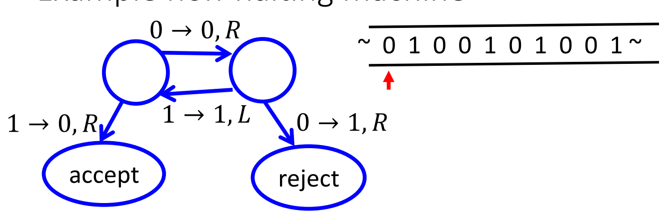
Language is **Turing recognizable** if some Turing machine recognizes it
- Also called "recursively enumerable"

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!



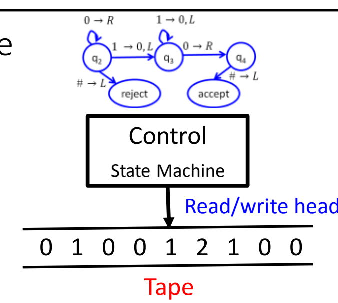
## Turing machine structure

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=$\{0^{2^n} \mid n \geq 0\}$

***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

5

## 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!

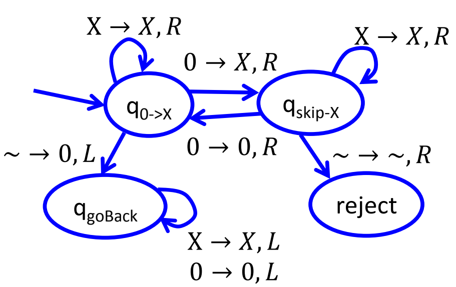
Round 1

~ ~ ~ 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 ~ ~ ~

## (Incomplete) State machine for alternating 0 removal




**NOTE: This partial solution requires additional states (and transitions) to handle which there is exactly one 0 on the tape (leading to accept!)**

## Language D=$\{a^i b^j c^k \mid k=i \times j \text{ and } i,j,k>0\}$

Multiplication on a Turing Machine!

Consider 2x3=6

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

8

## 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

9

## "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 y b b ***c*** c c c c c ~ ~
~ ~ X a y b ***b*** X c c c c c ~ ~
…
~ ~ X a y y b X ***c*** c c c c ~ ~

10

## 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

11

## 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

12

## 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



## 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



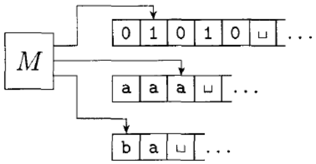
## Many equivalent variants of TM

- 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

## 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






## 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



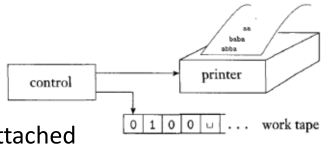
## Equivalence of Deterministic and Nondeterministic TMs

- 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



## 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

## Proof of enumerator equivalence

If enumerator E enumerates language A, TM M recognizes it

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

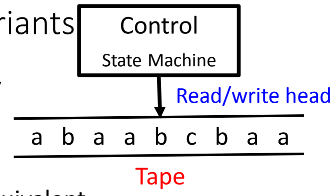
If TM M recognizes language, A, can construct enumerator E for A:

- 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



## 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 $10^{th}$ 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



## 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



## 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