Last time: CFGs → regular expressions

Still have sequencing, branching, repetition

added: recursion

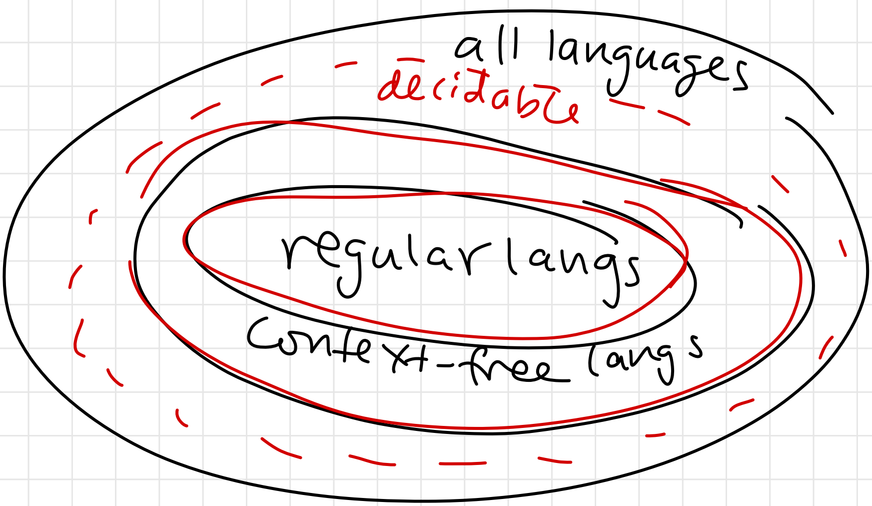
=> count

What are we going to skip?

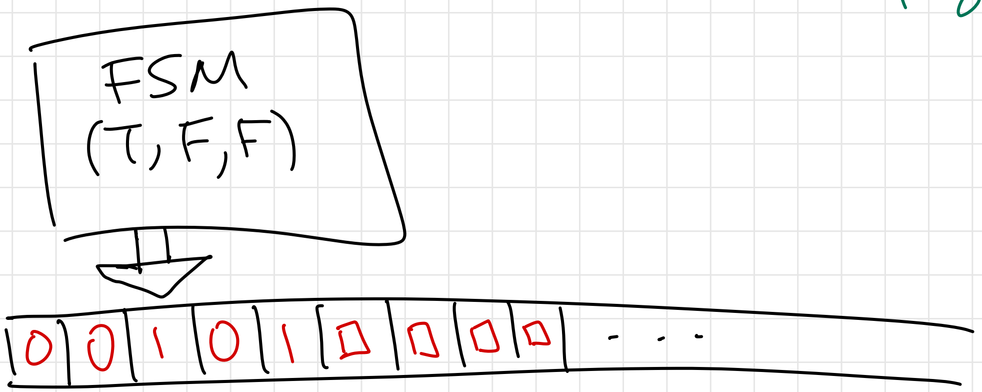
- analog to DFA/NFA: pushdown automata
- are there languages that are not context-free? ✓

  $\{0^n 1^n 0^n : n \geq 0\}$

- language transformations



computation	languages	machine
sequencing branching repetition	regular	DFA/NFA
recursion	context-free	pushdown automaton
(infinite) memory	decidable	Turing machine (Python program)



At each step:

- read symbol pointing to on tape — symbol ↓ a
- based symbol + current state, write a symbol at current position — δ
- move L or R

ex TM that ~~recognizes~~ decides $\{0^n 1^n 0^n : n \geq 0\}$

$TM = (\Gamma, \square, \Sigma, Q, q_{start}, q_{reject}, q_{accept}, \delta)$

Γ = tape alphabet

$\square \in \Gamma$ = blank symbol

$\Sigma \subseteq (\Gamma \setminus \square)$ = input alphabet

Q = states

$q_{start}, q_{accept}, q_{reject} \in Q$

once a TM enters q_{accept} or q_{reject}, it halts

$\delta(Q \setminus \{q_{accept}, q_{reject}\} \times \Gamma) \to Q \times \Gamma \times \{+1, -1\}$

R , L

$\Gamma = \{0, 1, \$, x, \square\}$ tape alphabet

$\Sigma = \{0, 1\}$

$Q = \{start, seek1, seek0, reset, verify, accept, reject\}$

$\delta(\ p\ , a) = (\ q\ , b, \delta)$	explanation
$\delta(\text{start}, 0) = (\text{seek1}, \$, +1)$	mark first 0 and scan right
$\delta(\text{start}, \text{x}) = (\text{verify}, \$, +1)$	looks like we're done, but let's make sure
$\delta(\text{seek1}, 0) = (\text{seek1}, 0, +1)$	scan rightward for 1
$\delta(\text{seek1}, \text{x}) = (\text{seek1}, \text{x}, +1)$	
$\delta(\text{seek1}, 1) = (\text{seek0}, \text{x}, +1)$	mark 1 and continue right
$\delta(\text{seek0}, 1) = (\text{seek0}, 1, +1)$	scan rightward for 0
$\delta(\text{seek0}, \text{x}) = (\text{seek0}, \text{x}, +1)$	
$\delta(\text{seek0}, 0) = (\text{reset}, \text{x}, +1)$	mark 0 and scan left
$\delta(\text{reset}, 0) = (\text{reset}, 0, -1)$	scan leftward for \$
$\delta(\text{reset}, 1) = (\text{reset}, 1, -1)$	
$\delta(\text{reset}, \text{x}) = (\text{reset}, \text{x}, -1)$	
$\delta(\text{reset}, \$) = (\text{start}, \$, +1)$	step right and start over
$\delta(\text{verify}, \text{x}) = (\text{verify}, \$, +1)$	scan right for any unmarked symbol
$\delta(\text{verify}, \square) = (\text{accept}, \square, -1)$	success!

any unspecified transition goes to reject

$\delta(\text{start}, 1) = \text{reject}$

$\{0^n 1^n 0^n : n \geq 0\}$

010

reject

start

seek1

verify

reset

seek0

alg: match i^{th} 0, 1, 0

(start, 001100)
⇒ (seek1, \$01100)
⇒ (seek1, \$01100)
⇒ (seek0, \$0x100)
⇒ (seek0, \$0x100)
⇒ (reset, \$0x1x0)
⇒ (reset, \$0x1x0)
⇒ (reset, \$0x1x0)
⇒ (reset, \$0x1x0)
⇒ (start, \$0x1x0)
⇒ (seek1, \$\$x1x0)
⇒ (seek1, \$\$x1x0)
⇒ (seek0, \$\$xxx0)
⇒ (seek0, \$\$xxx0)
⇒ (reset, \$\$xxxx)
⇒ (reset, \$\$xxxx)
⇒ (reset, \$\$xxxx)
⇒ (reset, \$\$xxxx)
⇒ (start, \$\$xxxx)
⇒ (verify, \$\$\$xxx)
⇒ (verify, \$\$\$\$xx)
⇒ (verify, \$\$\$\$\$x)
⇒ (verify, \$\$\$\$\$\$□)
⇒ (accept, \$\$\$\$\$\$) ⇒ **accept!**

match first 0,1,0

match second 0,1,0

in words, why does 00100 reject?

(start, 00100)
⇒ (seek1, \$0100)
⇒ (seek1, \$0100)
⇒ (seek0, \$0x00)
⇒ (reset, \$0xx0)
⇒ (reset, \$0xx0)
⇒ (reset, \$0xx0)
⇒ (start, \$0xx0)
⇒ (seek1, \$\$xx0)
⇒ (seek1, \$\$xx0)
⇒ (seek1, \$\$xx0) ⇒ **reject!**

State vs. Configuration

(tape contents and state)

↑ finite

⌈ finite too

⇑ infinite

(not Theorem)

Church-Turing Thesis:

TMs are equivalent to all reasonable models of computation.

Given an input w, a TM can:

- accept } halt
- reject
- loop forever ←

If a TM halts on all inputs, we call it a decider.

A language L is decidable if there is a TM that accepts every string in the language and rejects every string not in L.

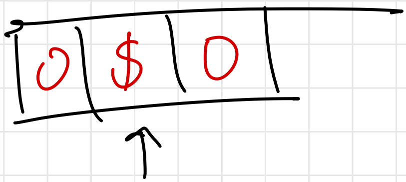
A language L is recognizable if there is a TM that accepts a string iff it is in L.

with table: diff ?

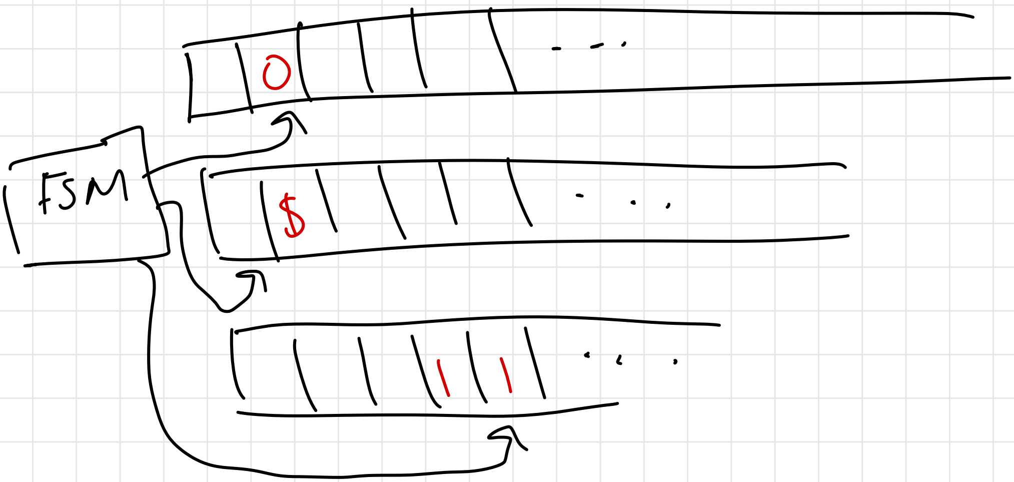
Equivalence of TM w/ "stronger" models of computation

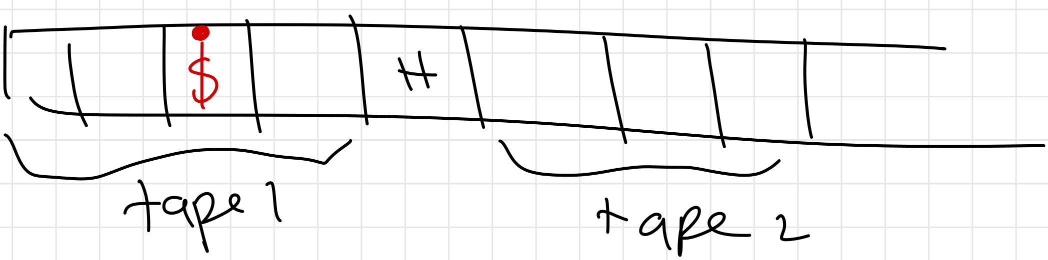
- stay put

move R
move L



- what if we want multiple tapes?





$\$, \dot{\$}$