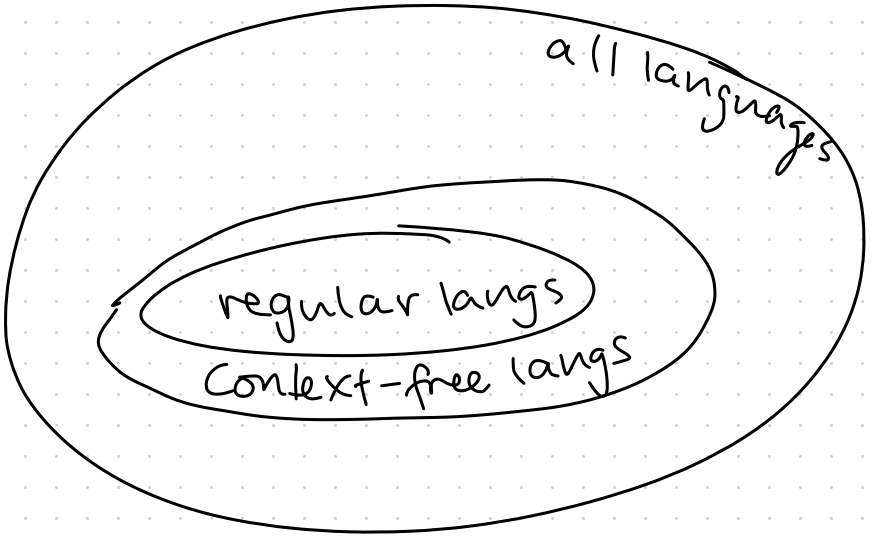
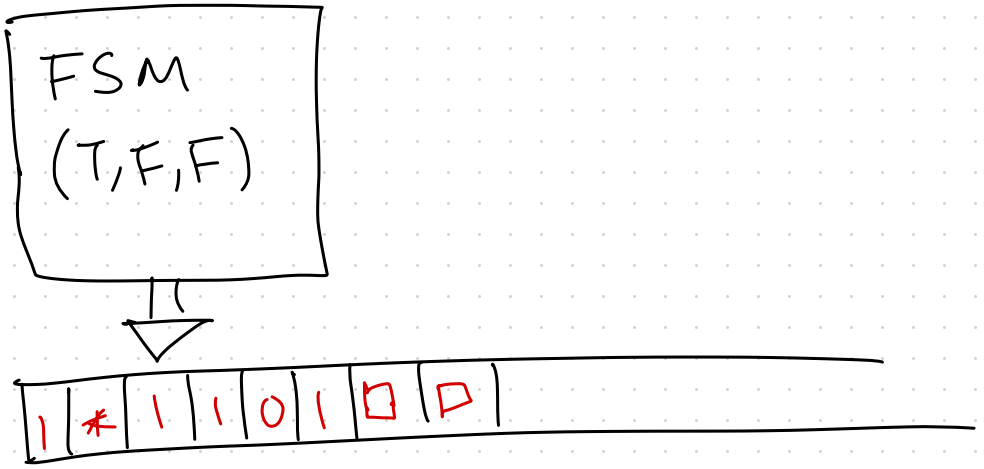
last time: CFGs
- add recursion
- what did we miss?
  - equivalent to DFA/NFA: pushdown automaton
  - non-context free languages
    $\{0^n 1^n 0^n : n \geq 0\}$
  - language transformations



	languages	machine
sequencing branching repetition	regular	DFA/NFA
recursion	context-free	pushdown automaton
memory	decidable	Turing machine



At each step:

Read symbol currently pointing at on tape

Based on symbol + current state:

- write symbol at current position
- change state
- move 1 step L or R

ex TM to recognize $\{0^n 1^n 0^n : n \geq 0\}$

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

Γ = tape alphabet

$\square \in \Gamma$ = blank symbol

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

Q = states

once a TM enters q_{acc} or q_{rej}, halts

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

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

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

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

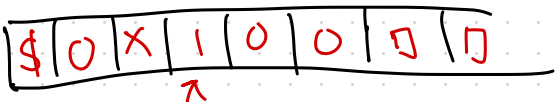
$\Sigma = \{0, 1\}$

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

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

any unspecified transition goes to reject ←

(start, 1) ?



start

reject

seek 1

verify

seek 0

reset

basic algorithm:
- match i^{th} 0, 1, 0
- mark 1st set of 0s w/ $
- mark 1s and 2nd set of 0s w/ x

example run on 001100

(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!**

does (start, 1) = reject
still make sense?

(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 + position + state)

↑ finite ↑ infinite

Church-Turing Thesis:

Turing machines are equivalent to all reasonable models of computation

Polynomial

$5x^2 + z^3 - 2y = 0$

solution for x, z, y ints?

1970: undecidable

Given a input w, a TM can either:

- accept } halt
- reject }
- loop forever

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

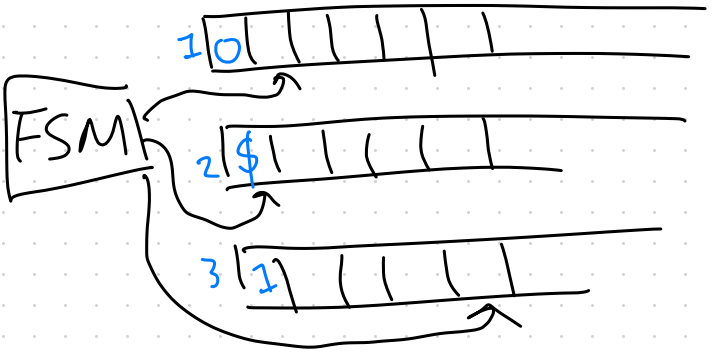
- A language L is decidable if there

is a TM that accepts every string in L 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.

Equivalence of TM w/ other models of computation

- what if we allow TM to stay put?
  - replace any "stay put" w/ 2move:
    - Move R
    - move L
- what if we want multiple tapes?

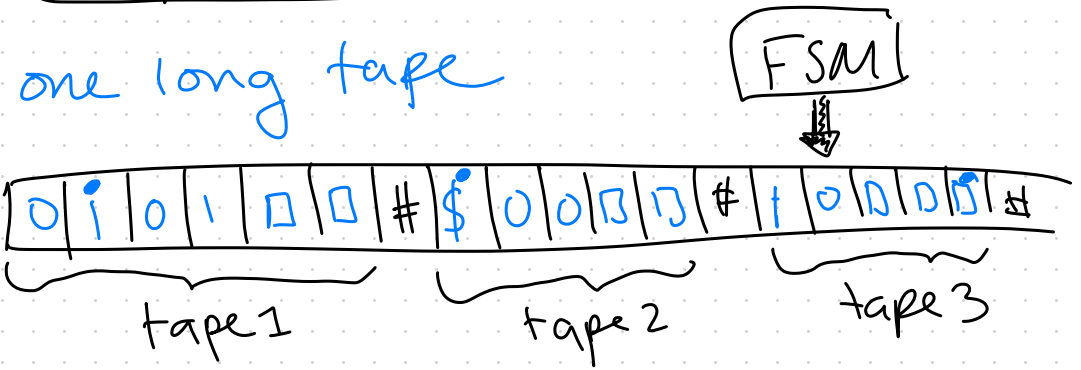


$\{0, 1, \sqcup, \$\}$

$\{(0,0,0),$

① tape alphabet = tuples

(2) one long tape



marked alphabet

the description of a TM can be input to a TM

$\Rightarrow$ a TM can be input to itself