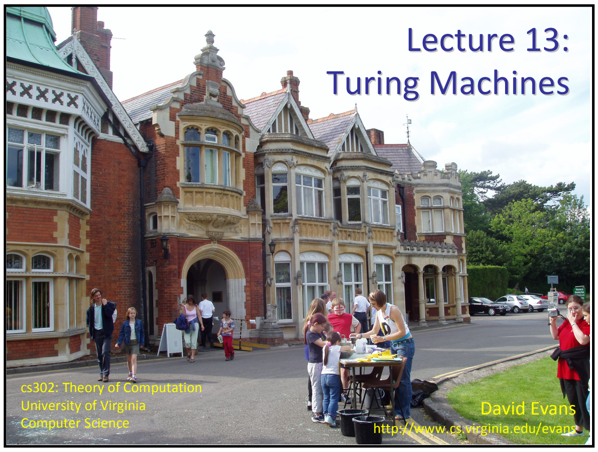
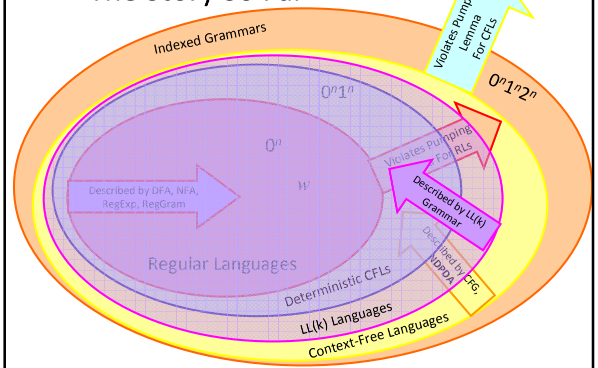
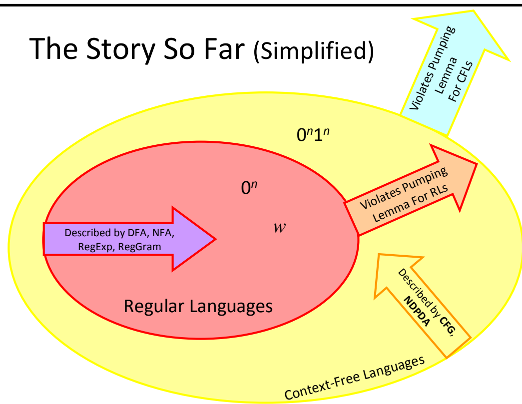



## The Story So Far



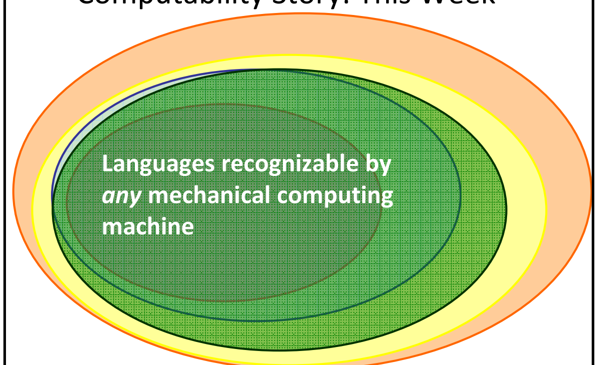



## The Story So Far (Simplified)



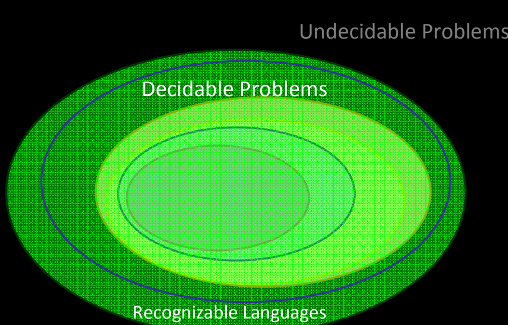



## Computability Story: This Week



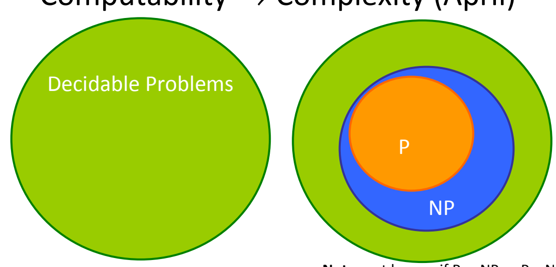



## Computability Story: Next Week






## Computability $\rightarrow$ Complexity (April)




**Note:** not known if P $\subset$ NP or P = NP

Problems that can be solved by a computer (eventually).

Problems that can be solved by a computer in a ***reasonable*** time.

# Exam 1



# Exam 1

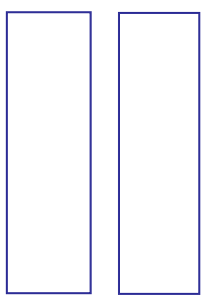
- Problem 4c: Prove that the language $\{0^n 1^{n^2}\}$ is not context-free.

Lengths of strings in $L$:
$n = 0$ $0 + 0^2 = 0$
$n = 1$ $1 + 1^2 = 2$
$n = 2$ $2 + 2^2 = 6$
$n = 3$ $3 + 3^2 = 12$
...
$n = k$ $k + k^2$

Pumping lemma for CFLs says there must be some way of picking $s = uvxyz$ such that $m = |v| + |y| > 0$ and $uv^ixy^iz$ in $L$ for all $i$.
So, increasing $i$ by 1 adds $m$ symbols to the string, which must produce a string of a length that is not the length of a string in $L$.



# Recognizing $\{0^n 1^{n^2}\}$



DPDA with **two** stacks?

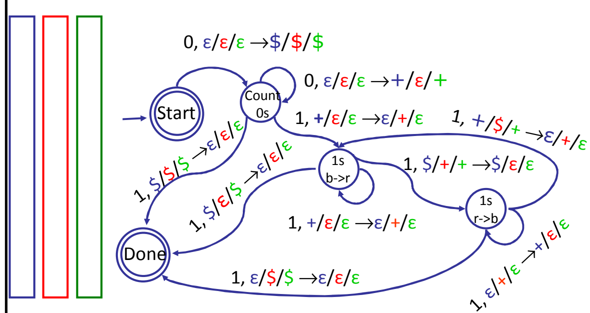
DPDA with **three** stacks?

...

?



# 3-Stack DPDA Recognizing $\{0^n 1^{n^2}\}$

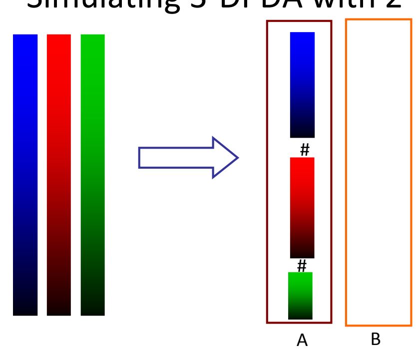




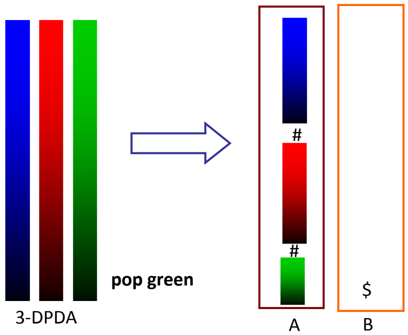

# Can it be done with 2 Stacks?



# Simulating 3-DPDA with 2-DPDA



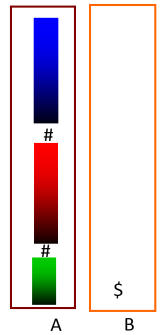
## Simulating 3-DPDA with 2-DPDA+




push$_B$($)
push$_B$(pop$_A$())
...
push$_B$(pop$_A$(#))
push$_B$(pop$_A$())
...
push$_B$(pop$_A$(#))
**res = pop$_A$()**
push$_A$(pop$_B$(#))
push$_A$(pop$_B$())
...
pop$_B$($)



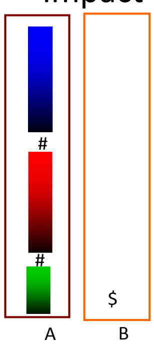
## 2-DPDA + Forced ε-Transitions




Need to do lots of stack manipulation to simulate 3-DPDA on one transition: need transitions with no input symbol (but not nondeterminism!)



## Impact of Forced ε-Transitions




What is the impact of adding non-input consuming transitions?

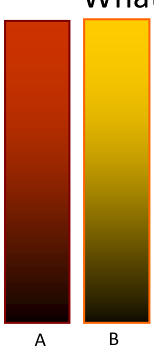
DPDA in length $n$ input:
runs for $\leq n$ steps
DPDA+ε in length $n$ input:
**can run forever**!



Is there any computing machine we can't simulate with a 2-DPDA+?



## What about an NDPDA?

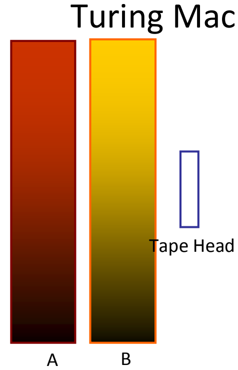



Use one stack to simulate the NDPDA's stack.

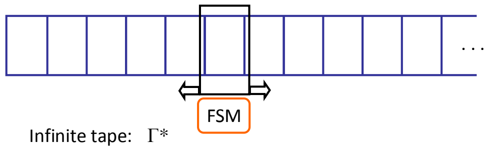
Use the other stack to keep track of nondeterminism points: copy of stack and decisions left to make.



## Turing Machine?



## Turing Machine



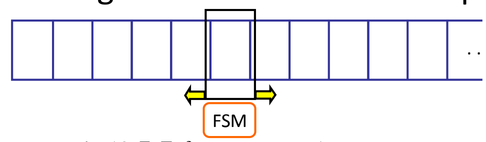

Infinite tape: $\Gamma^*$

Tape head: read current square on tape, write into current square, move one square **left** or **right**

FSM: like PDA, except:
 transitions also include direction (**left**/**right**)
 final accepting and rejecting states



## Turing Machine Formal Description




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

$Q$: finite set of states

$\Sigma$: input alphabet (cannot include blank symbol, _)

$\Gamma$: tape alphabet, includes $\Sigma$ and _

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

$q_0$: start state, $q_0 \in Q$

$q_{accept}$: accepting state, $q_{accept} \in Q$

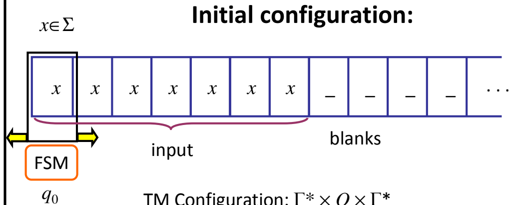
$q_{reject}$: rejecting state, $q_{reject} \in Q$

(Sipser's notation)



## Turing Machine Computing Model

**Initial configuration:**




TM Configuration: $\Gamma^* \times Q \times \Gamma^*$

tape contents left of head | current FSM state | tape contents head and right



## TM Computing Model

$$\delta^*: \Gamma^* \times Q \times \Gamma^* \rightarrow \Gamma^* \times Q \times \Gamma^*$$

The $q_{accept}$ and $q_{reject}$ states are final:

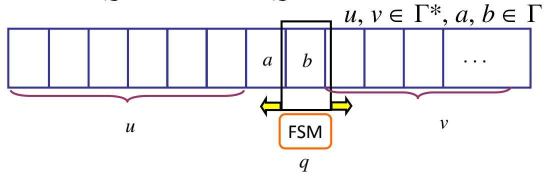
$$\delta^*(L, q_{accept}, R) \rightarrow (L, q_{accept}, R)$$
$$\delta^*(L, q_{reject}, R) \rightarrow (L, q_{reject}, R)$$



## TM Computing Model

$$\delta^*: \Gamma^* \times Q \times \Gamma^* \rightarrow \Gamma^* \times Q \times \Gamma^*$$

$u, v \in \Gamma^*, a, b \in \Gamma$




$\delta^*(ua, q, bv) = (uac, q_r, v)$ if $\delta(q, b) = (q_r, c, \mathbf{R})$

$\delta^*(ua, q, bv) = (u, q_r, acv)$ if $\delta(q, b) = (q_r, c, \mathbf{L})$

Also: need a rule to cover what happens at left edge of tape



## Thursday's Class

- Robustness of TM model
- Church-Turing Thesis