LECTURE 14: TURING MACHINES

CSC 250: Theory of Computation

Attendance

Fill this out

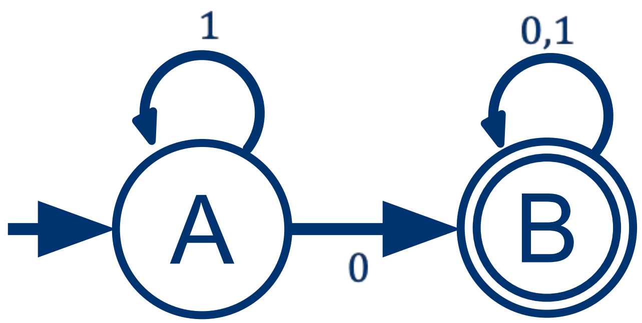
Attendance Form 14

ONLY do it if you are in class!!!

So far, four approaches…

Regular Expressions

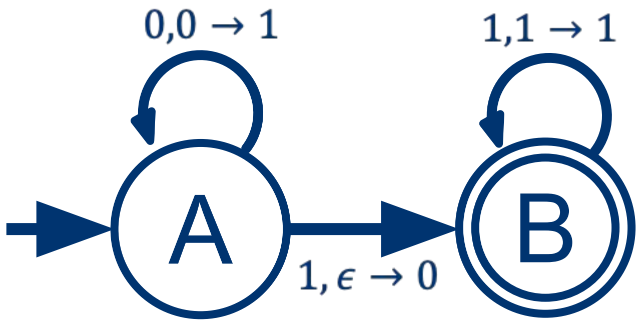
$$\Sigma^*1\Sigma^*$$



Finite Automata

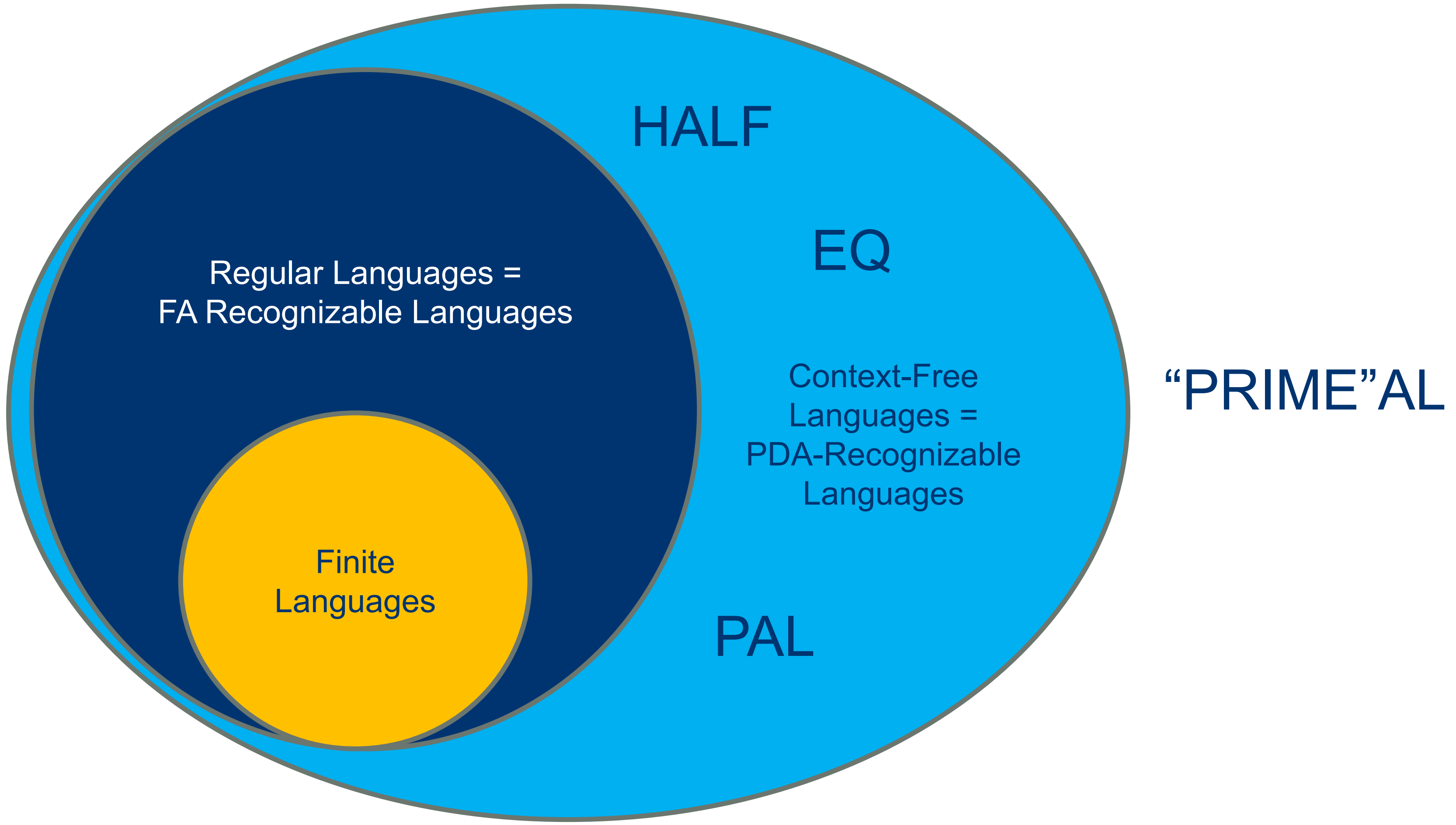
Context-Free Grammars

$$S \to 01S$$

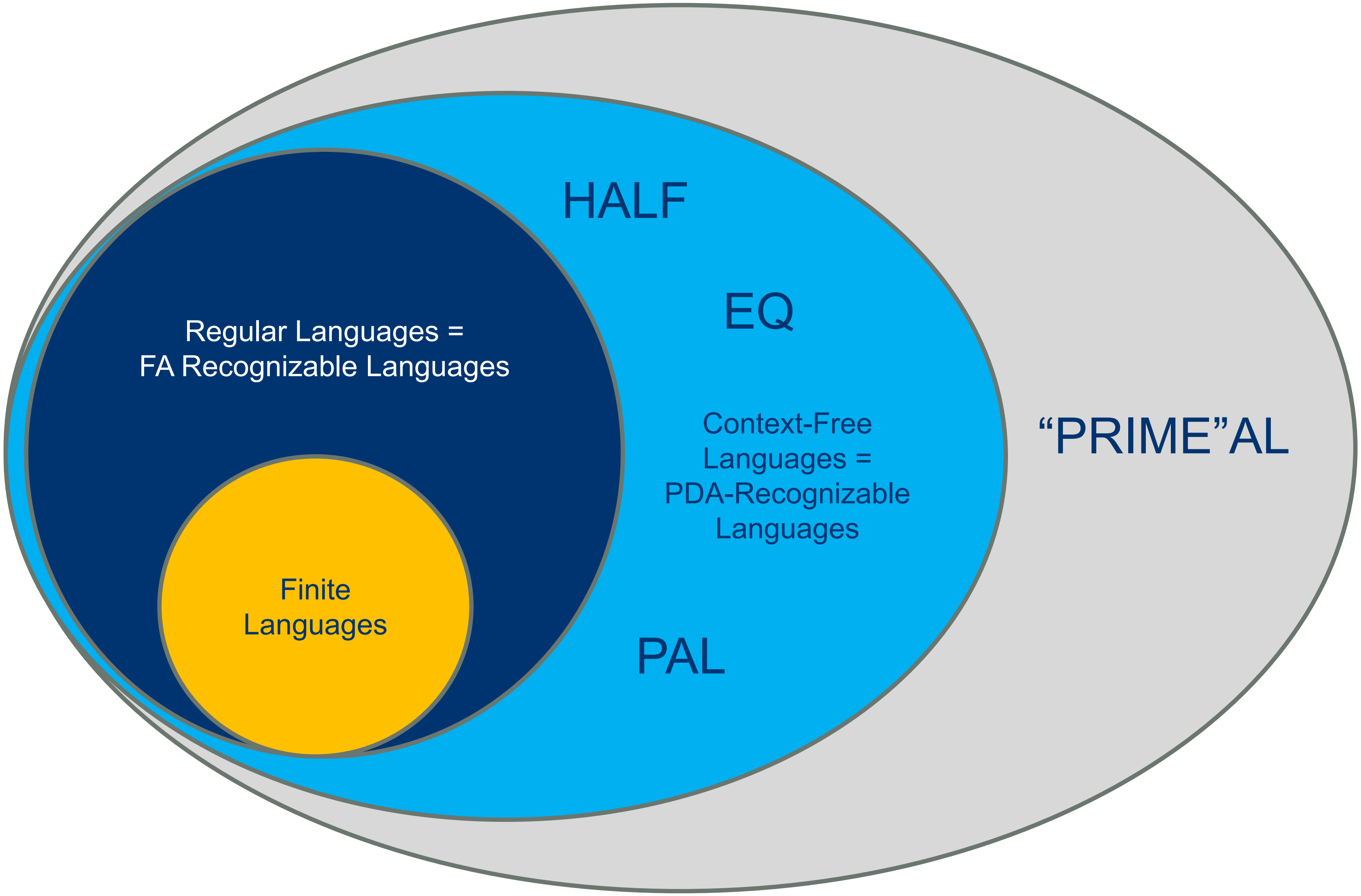


Push-Down Automata

What we know…


HALF
EQ
Regular Languages =
FA Recognizable Languages
Context-Free
Languages =
PDA-Recognizable
Languages
"PRIME"AL
Finite
Languages
PAL

What we hope…


HALF
EQ
Regular Languages =
FA Recognizable Languages
Context-Free
Languages =
PDA-Recognizable
Languages
"PRIME"AL
Finite
Languages
PAL

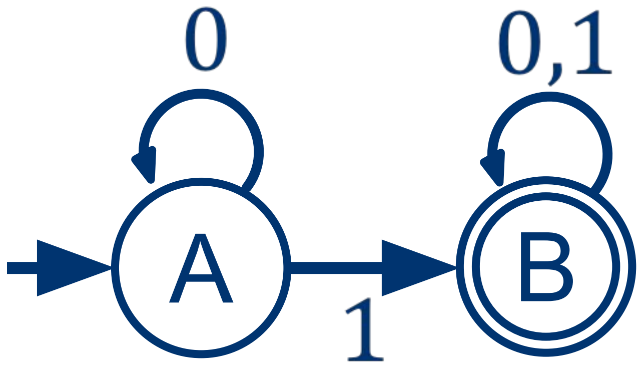
Backing up to FAs

Why couldn't we recognize

HALF, EQ, PAL, etc.

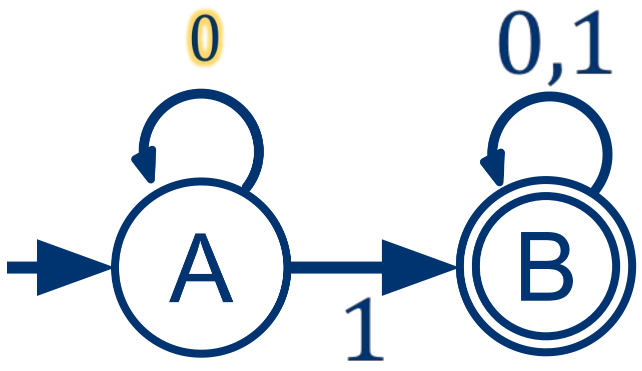
with a finite automaton?

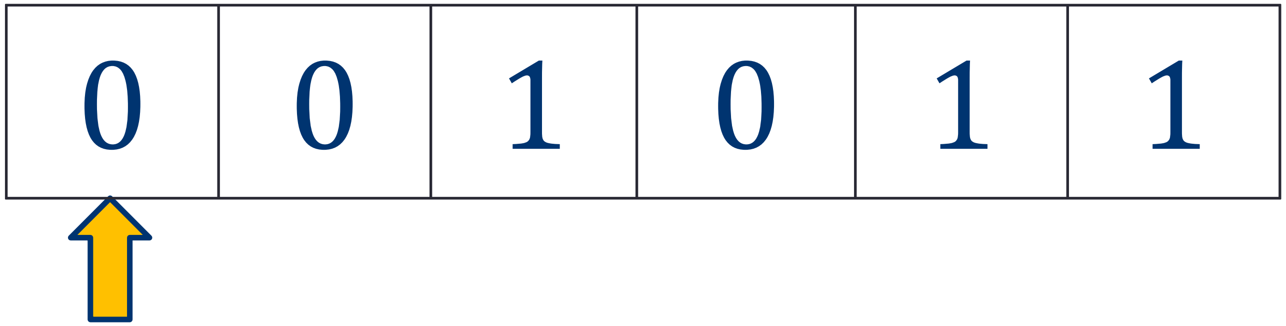
Another way to think about FAs



0	0	1	0	1	1

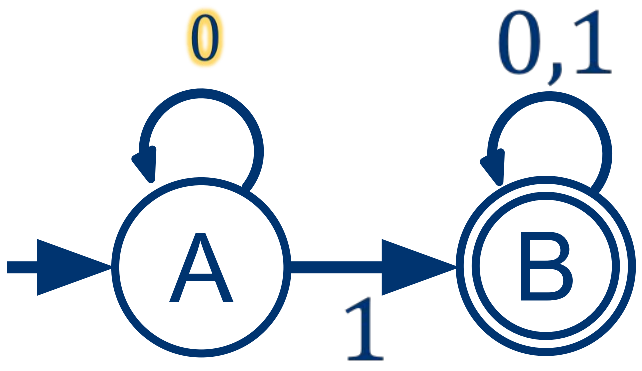
Another way to think about FAs

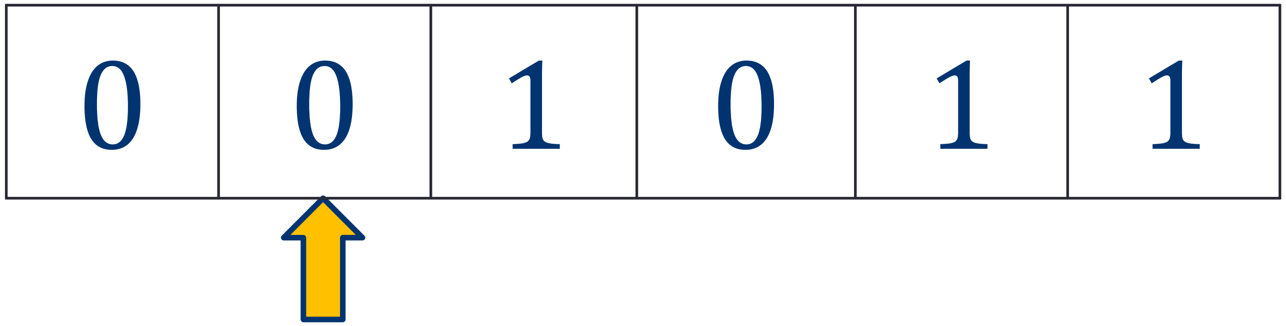




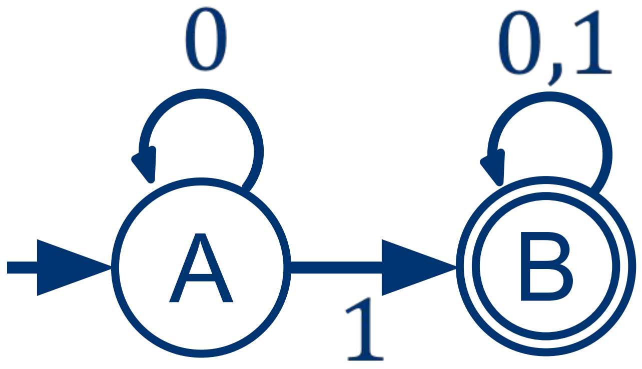
0	0	1	0	1	1

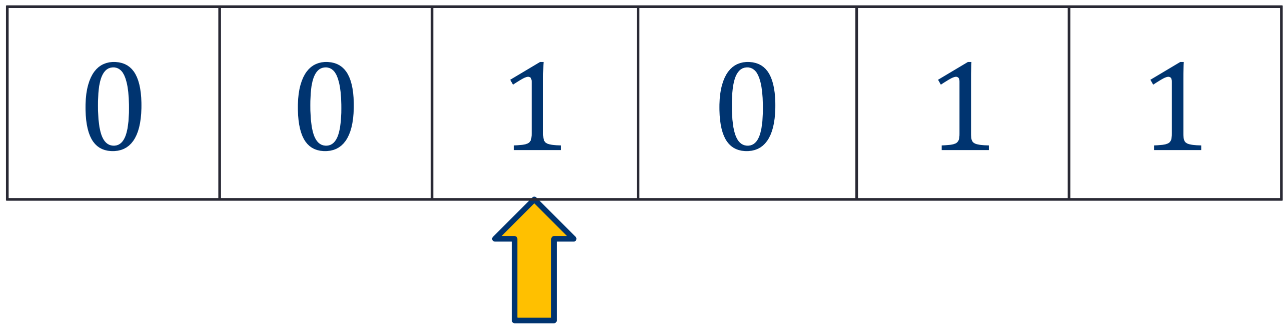
Another way to think about FAs



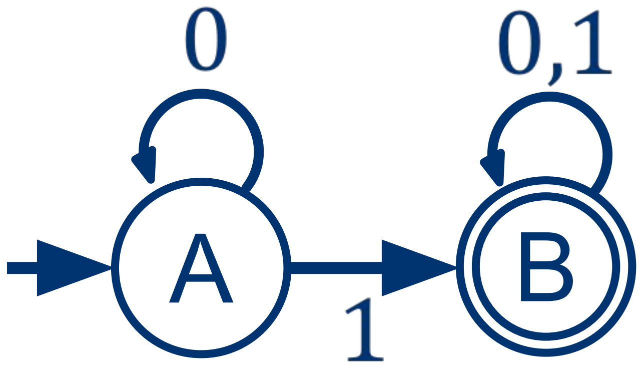


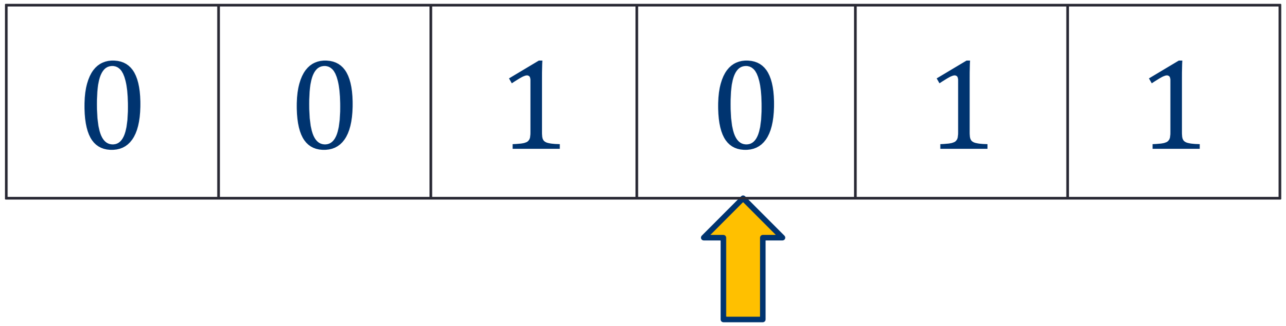
Another way to think about FAs



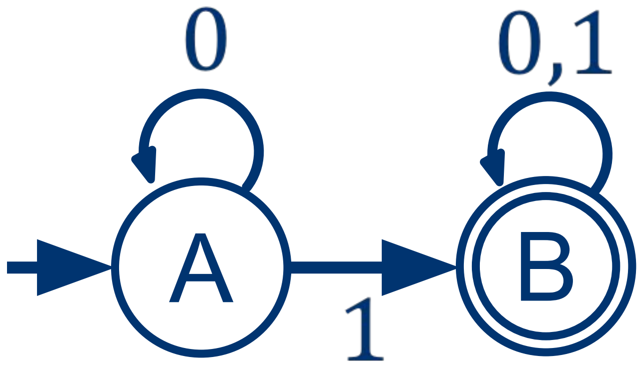


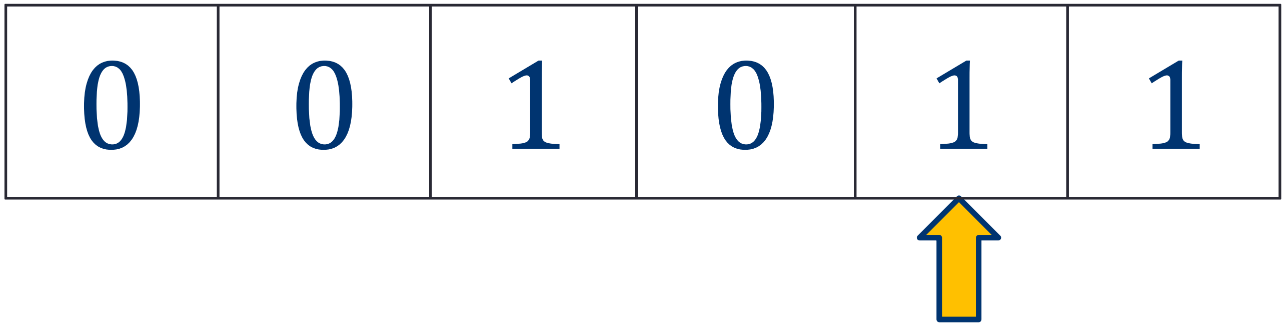
Another way to think about FAs



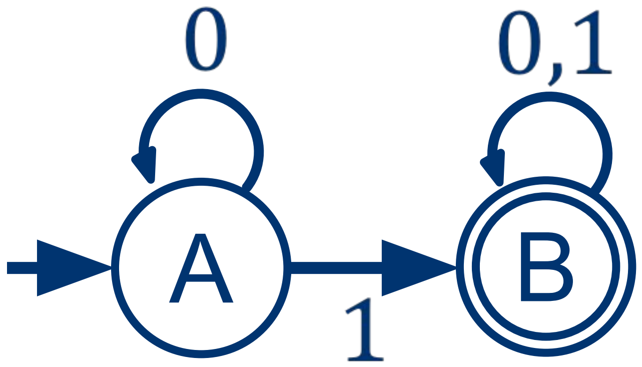


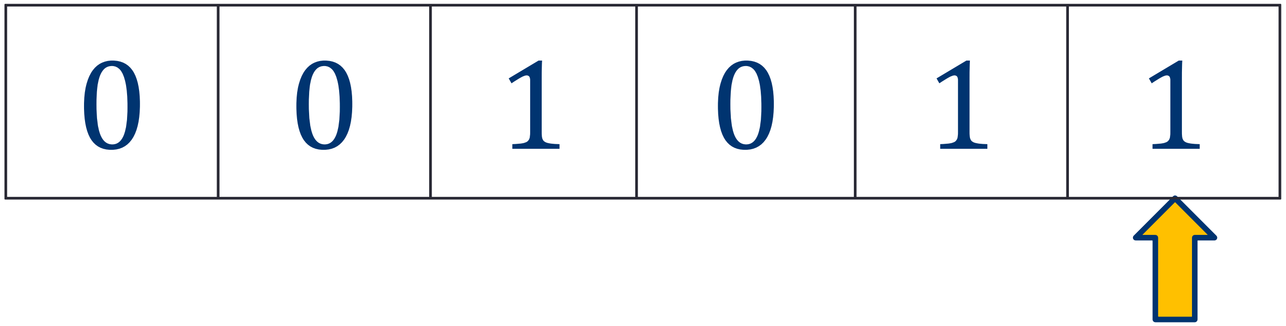
Another way to think about FAs



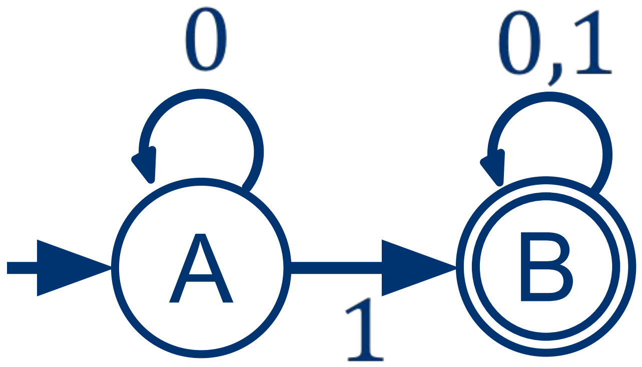


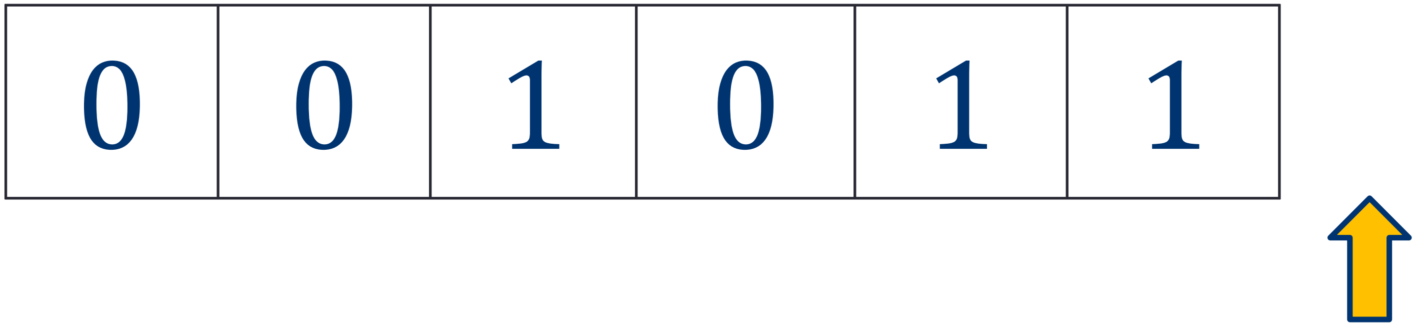
Another way to think about FAs





Another way to think about FAs





Discussion

Notice anything?

What if we wanted to do this

- **Goal**: determine if the following string is in PAL

0	0	1	0	1	0	0

Three new abilities

- Move both forward **and backward**
- **Write** new symbols to the tape
- **Halt** at any point on the tape (stop and return an answer)

“Turing Machine”

Formal definition

A TM is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where:

- Q is a finite set of states
- Σ is the input alphabet (doesn't include $\square$)
- Γ is the tape alphabet (includes $\square$, Σ, and maybe more)
- δ: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function
- $q_0 \in Q$, is the start state
- $q_{\text{accept}} \in Q$ is the accepting state
- $q_{\text{reject}} \in Q$ is the rejecting state, $q_{\text{reject}} \neq q_{\text{accept}}$

A note on stopping conditions

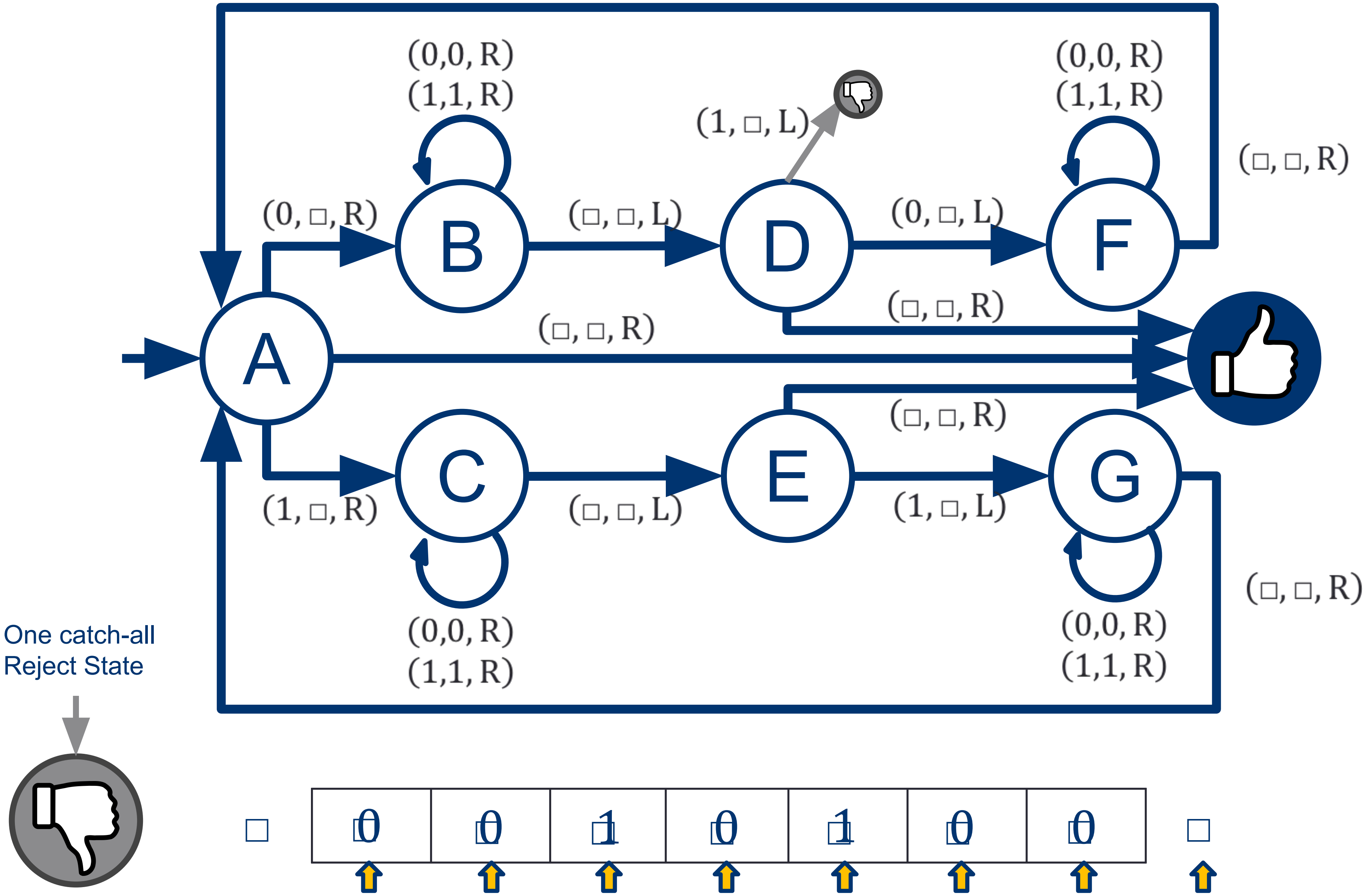
- A TM ***recognizes*** a language L iff it accepts all the strings in L (and nothing else)
- However, a TM doesn't HAVE to stop (it could loop)
- A TM ***decides*** a language L iff it accepts all strings in L <u>and</u> rejects all strings not in L
  - Tomorrow, we'll talk about TMs that can do this
  - For today, we won't worry about it

Formal definition

A TM is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where:

- Q is a finite set of states
- Σ is the input alphabet (doesn't include $\square$)
- Γ is the tape alphabet (includes $\square$, Σ, and maybe more)
- $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function
- $q_0 \in Q$, is the start state
- $q_{accept} \in Q$ is the accepting state
- $q_{reject} \in Q$ is the rejecting state, $q_{reject} \neq q_{accept}$

Goal: determine if a string is in PAL



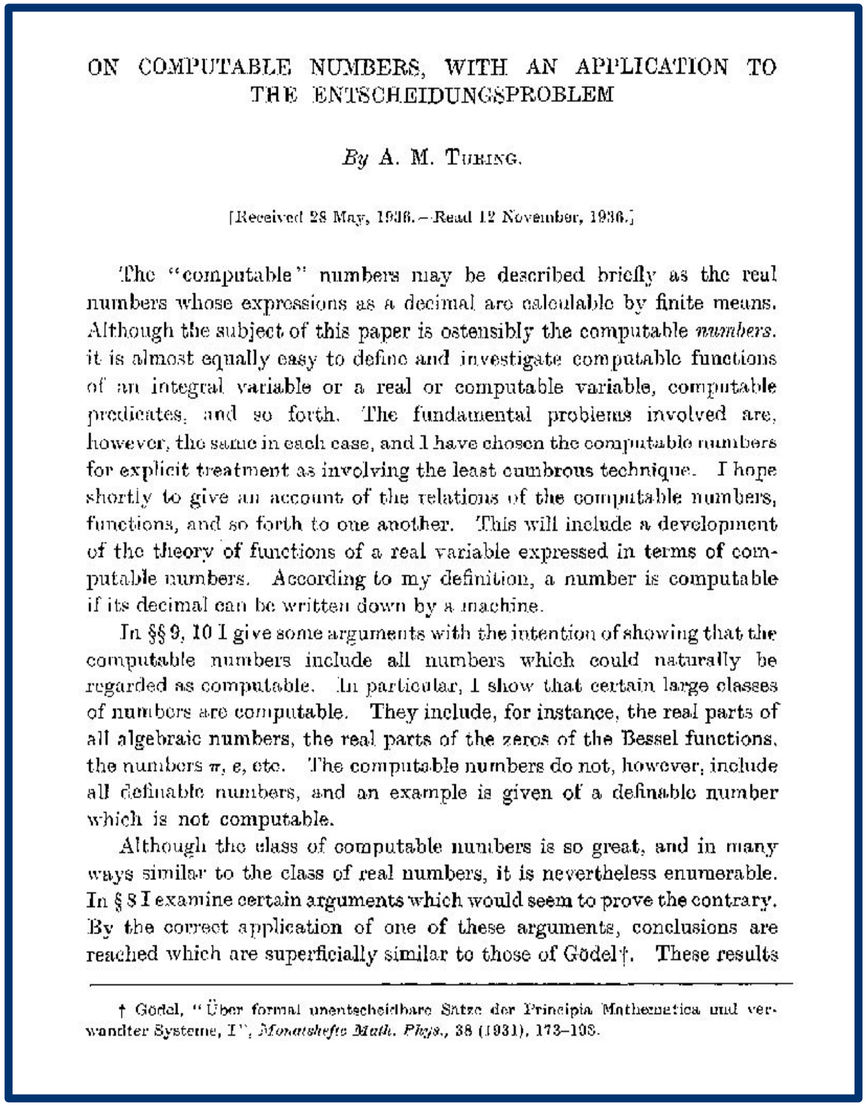
TMs are pretty powerful

- Can recognize languages that REs/Fas/CFGs/PDAs can’t
- Can search for non-regular/non-CF patterns
- Can even perform mathematical operations (+, -, *, /, etc.)
- Maybe the most interesting thing…

a Turing machine can **emulate**

another Turing machine

Universal TM



ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. Turing.

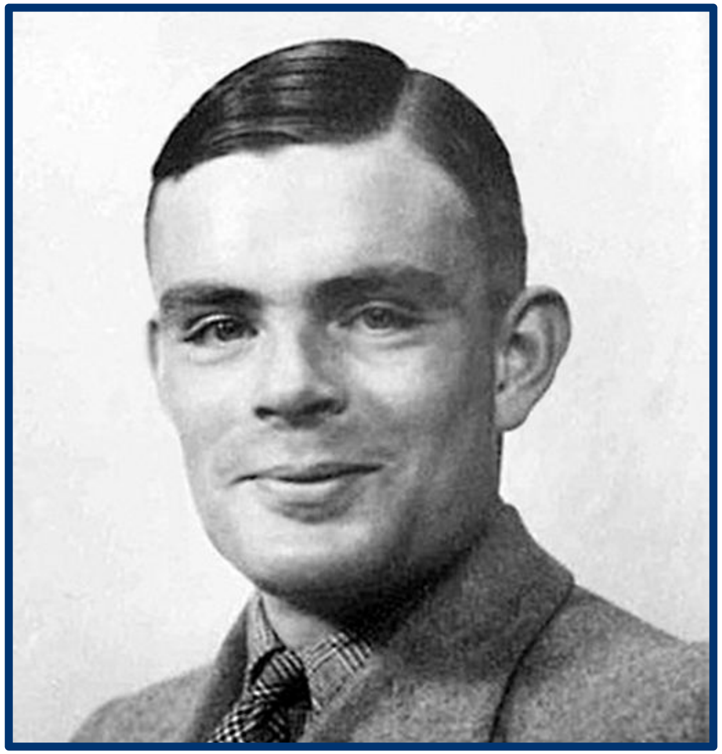
[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*. it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.



6. *The universal computing machine.*

It is possible to invent a single machine which can be used to compute any computable sequence.

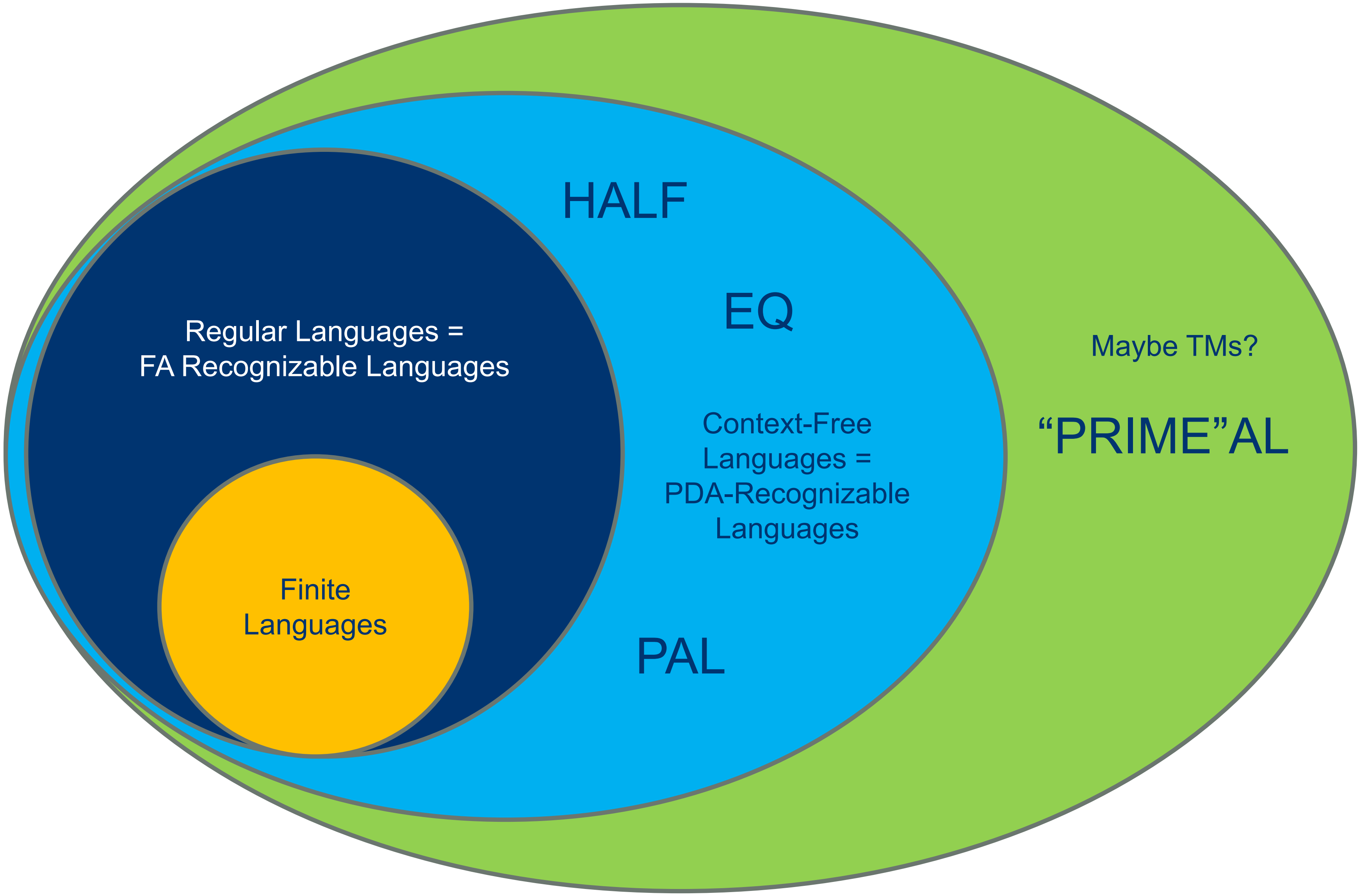
Discussion

How would this work?

But how can that be?

- TMs have to have a finite number of states, so how could one machine possibly emulate **all** other machines?

Cool!


HALF
EQ
Maybe TMs?
Regular Languages =
FA Recognizable Languages
Context-Free
Languages =
PDA-Recognizable
Languages
“PRIME”AL
Finite
Languages
PAL

Limitations of TMs

- Suppose we have a TM that never halts
  - Can we build another TM that can detect this?
  - In other words, can we make an infinite loop detector?

This would be pretty useful!

We could prove / disprove Goldbach's conjecture:

all even numbers 4 or greater
are the sum of two primes

by building a TM the does the following:

```
i = 2
while(i > 1):
 if 2*i is not a sum of two primes
    then HALT
 i = i+1
```

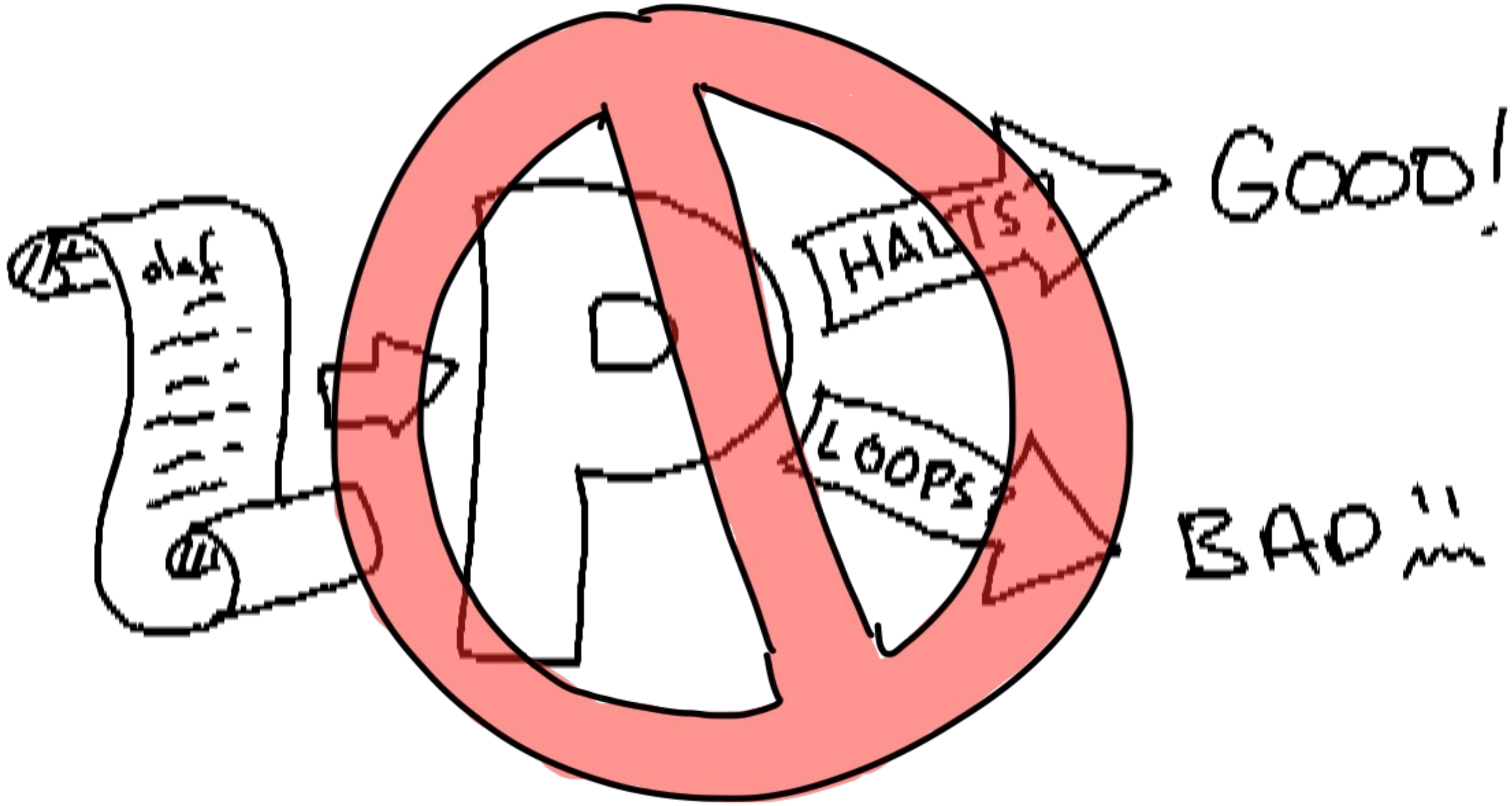
and asking "**does it halt**?"

“Scooping the Loop Snooper”

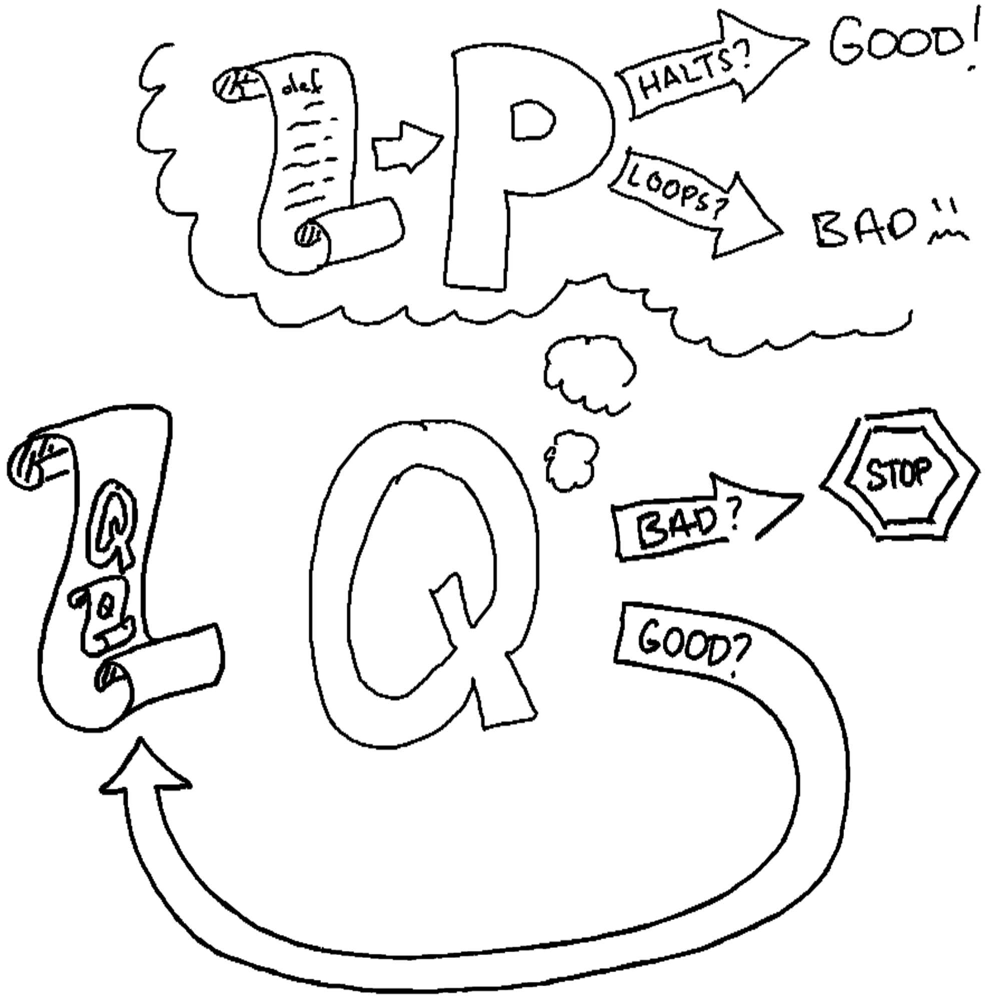
a poetic proof

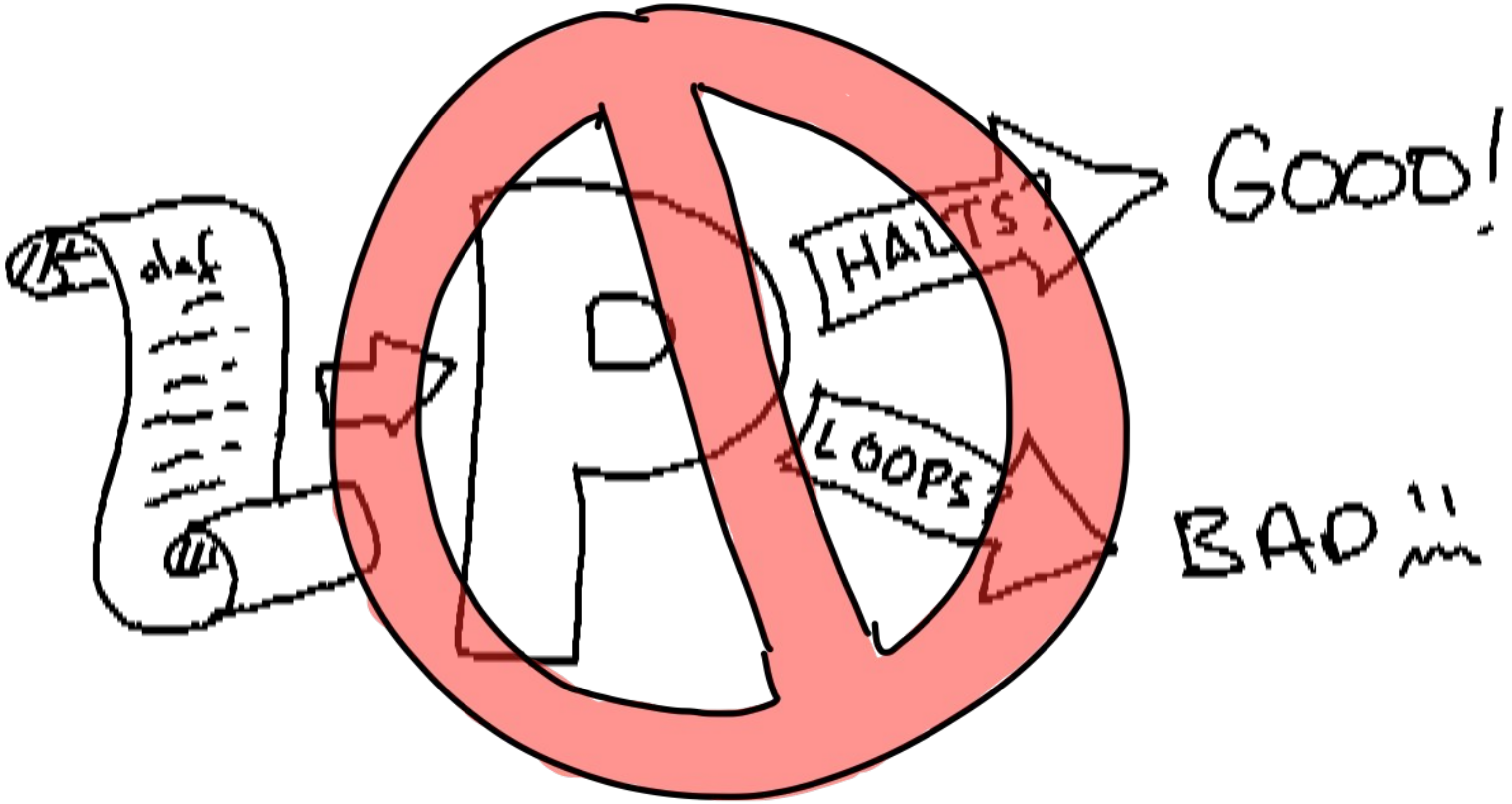
by Geoffrey K. Pullum

P


def
P
HALTS?
LOOPS?
GOOD!
BAD :(

Q


def
P
HALTS?
GOOD!
LOOPS?
BAD!!
Q
BAD?
STOP
GOOD?


def
P
HALTS?
LOOPS?
GOOD!
BAD ☹

Up next

- Decidability
- A4 due Thursday at 11:59PM EST