# Fundamental Topics in Theoretical Computer Science

Chris Waites

# Motivation

Could I program any function?

Certain functions seem more difficult to compute – is this necessarily true?

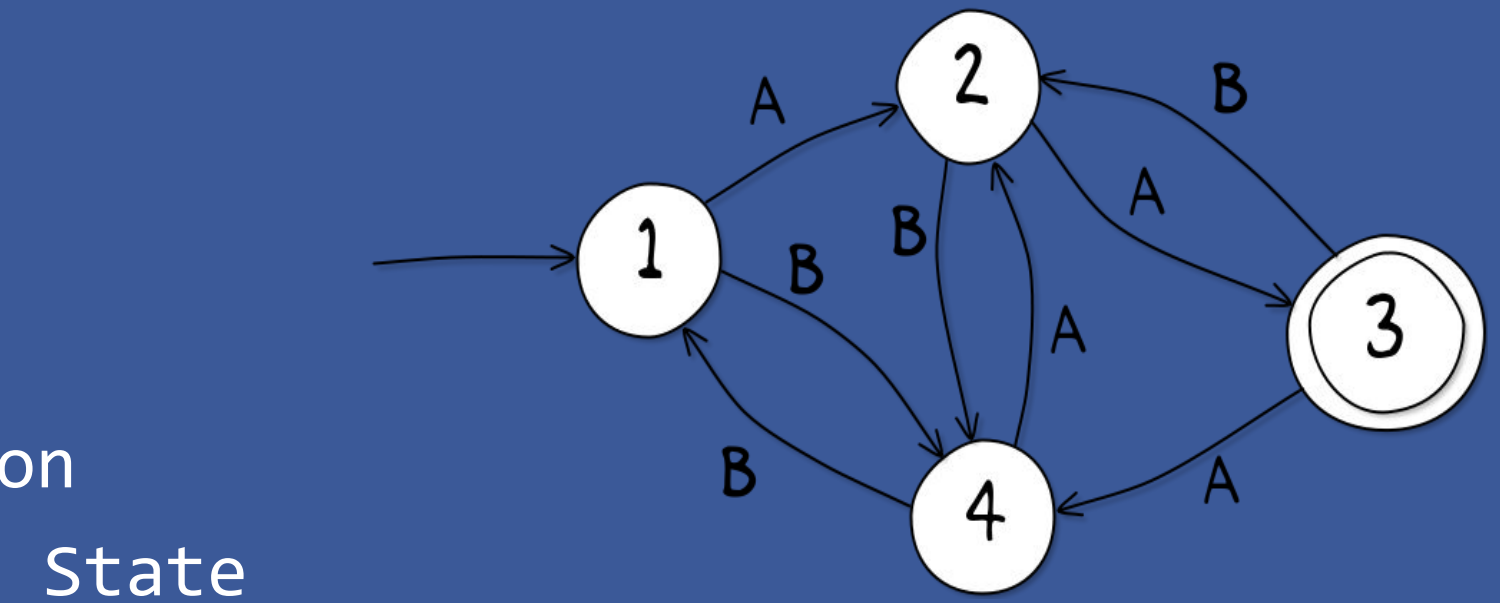
How can we rigorously address such questions?

# Discrete Finite Automata (DFA)

Set of States
- Initial State
- Accepting States

Set of Symbols

Transition Function
- State, Symbol -> State



# Regular Languages

Regular

- {’0’, ‘01’, ‘010’, ‘0101’, ‘01010’, ‘010101’, …}
- {‘0’, ‘10’, ‘100’, ’110’, ‘1000’, ‘1010’, …}
- $\{s_0, s_1, s_2, \ldots, s_k\}$
- Union/Intersection/Complement of Regular Languages

Irregular

- {’’, ‘10’, ‘1100’, ‘111000’, ‘11110000’, …}
- $\{‘0’^k : k \text{ power of } 2\}$
- $\{‘0’^k : k \text{ prime}\}$

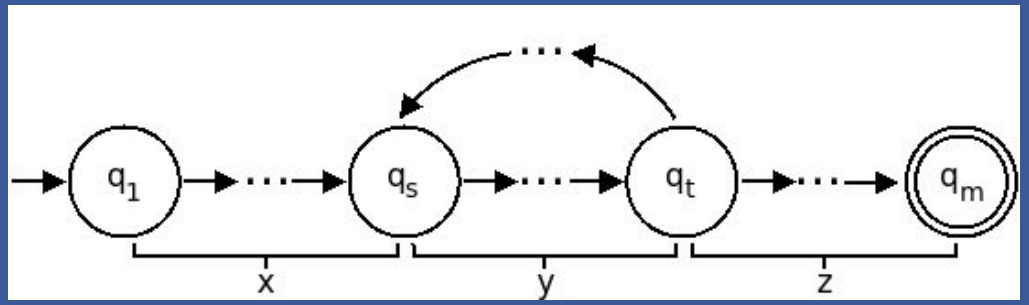
# Pumping Lemma

If L is a regular language, for all strings s $\epsilon$ L longer than p, s can be written as s = xyz s.t…

$|y| \geq 1$

$|xy| \leq p$

$xy^nz \epsilon L \; ; \; n \geq 0$

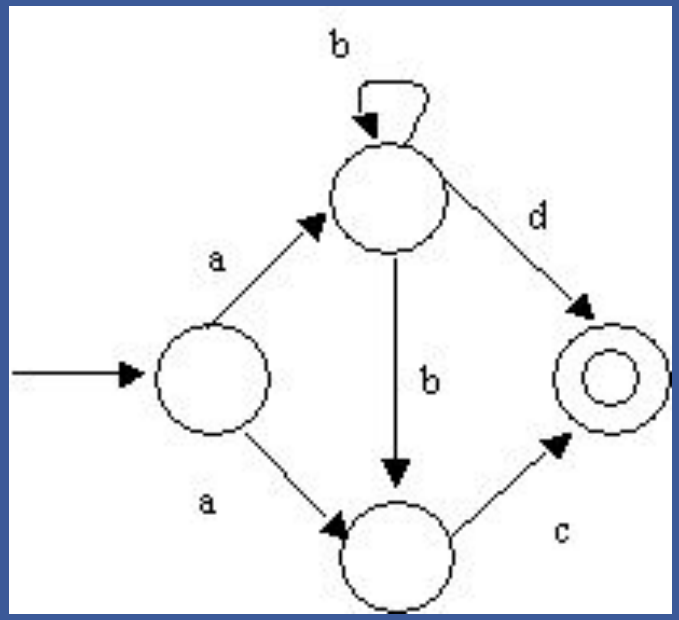


# Nondeterministic Finite Automata

“Seemingly stronger” model of computation

Rigorous Definition

DFA/NFA Equivalence



# The Turing Machine

Set of States
- Initial State
- Accepting States
- Rejecting States

Set of Symbols + {‘_’}

Transition Function
- State, Symbol -> Symbol, Action

# But Why?

Captures “what a human could do with pen and paper”

Equivalent to independently-developed models (General Recursive Functions, λ-Calculus)

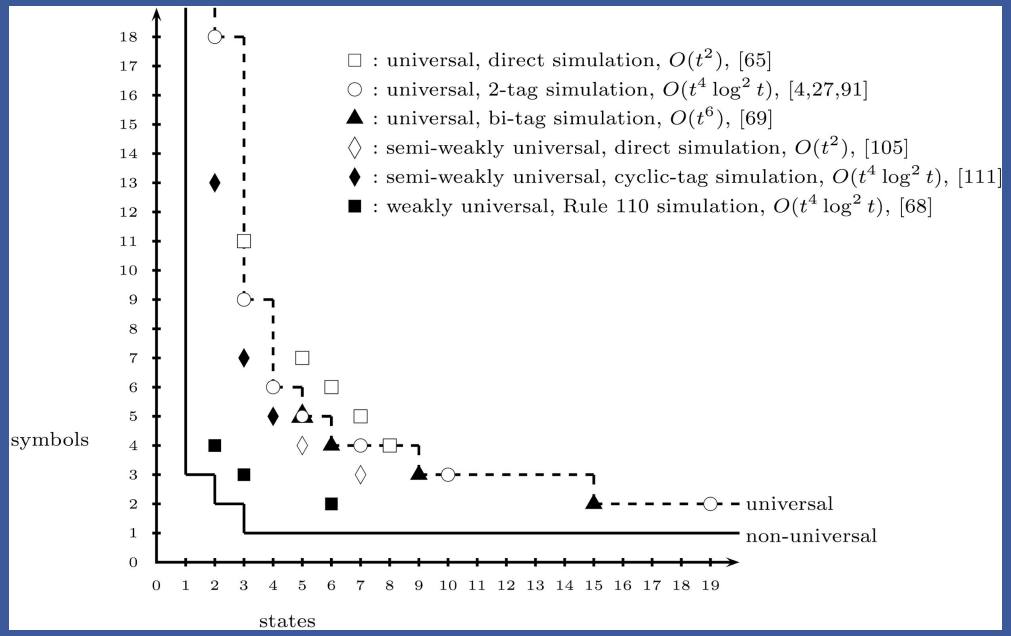
Languages accepted by Turing machines called “computable” functions

# Computable Functions

Anything you have written
- isPalindrome(X)
- isPrime(X)

Universal Turing Machine
- (TM Serialization)

Does fully understanding a function mean we can compute it?



# Halting Problem

Halt(<M>_X):
  Accept if M halts on X, otherwise reject

Halt’(<M>):
  Loop if M halts on <M>, otherwise accept

What will happen on Halt’(<Halt’>)?

# Approximate Halting Problem

```
approx-halt(<M>_X):
   returns H(<M>_X) for all X except one
```

Could this exist?

# Approximate Halting Problem

```
P(I):
  return approx-halt(<M>_X)

H(<M>_X):
  Construct P given <M> and X
  ans0 = P(“0”)
  ans1 = P(“1”)
  ans2 = P(“2”)
  return majority(ans0, ans1, ans2)
```

Therefore, approximate halting machine cannot exist

# Complexity Classes

A set of languages, characterized by some constraint

“The set of languages for which each there exists a Turing machine which accepts it *using at most k cells*”

“The set of languages for which each there exists a Turing machine which accepts it *using at most k transitions*”

# P vs. NP Problem

P

- “The set of languages for which each there exists a Turing machine which *accepts it in a polynomial number of transitions of input size*”

NP

- “The set of languages for which each there exists a *nondeterministic* Turing machine which accepts it in a polynomial number of transitions of input size”