

Welcome to CS420!

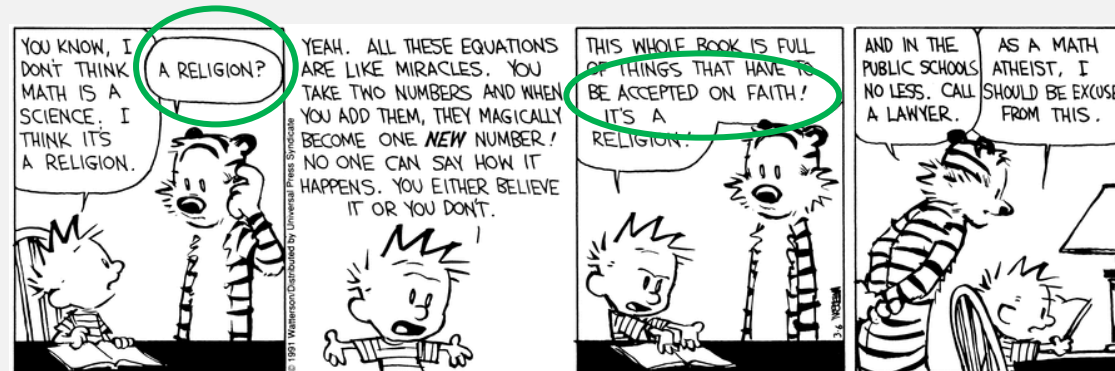
Intro to Theory of Computation

UMass Boston Computer Science

Instructor: Stephen Chang

Spring 2023

Today's Theme:
Does CS 420 teach
anything useful???



Welcome to CS420!

Intro to Theory of Computation

UMass Boston Computer Science
Instructor: Stephen Chang
Spring 2023



What's this?


CS 420 Lecture Logistics

- I expect lecture to be interactive
 - Participation is a part of your grade
 - Also, it's the best way to learn!
- I may call on students randomly
 - It's ok to be wrong! – will not affect your grade
 - Also, it's the best way to learn!
- Please tell me your name before speaking
 - Sorry in advance if I get it wrong
 - It's the best way for me to learn!

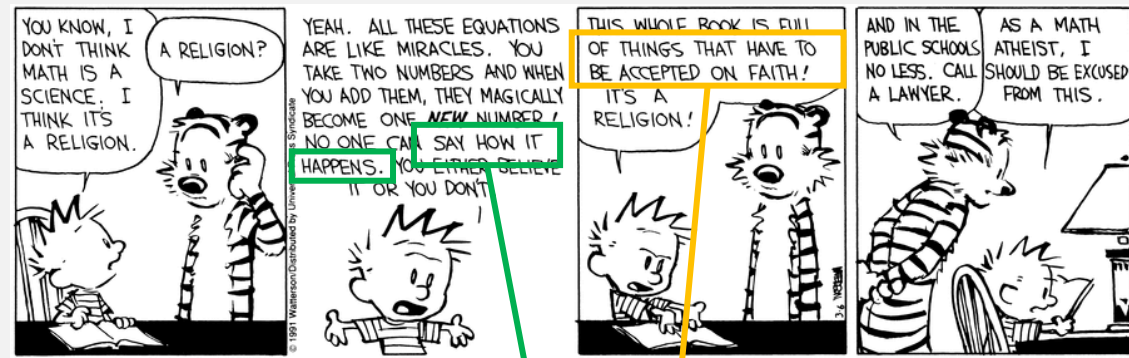
Welcome to CS420!
Intro to Theory of Computation

UMass Boston Computer Science
Instructor: Stephen Chang
Spring 2023

**How would you
define this?**



Computation Is ...



- $1 + 1 = ??$
- $= 2$

... some base definitions and assumptions (“axioms”),
e.g., “A **Number** is either **0**, or the **successor** of another **Number**” ...

- $11 + 11 = ??$
- $= 22$

... and rules that use the definitions and axioms (“algorithm”),
e.g., grade school arithmetic

- $9999999999 + 9999999999 = ??$
- $= 19999999998$

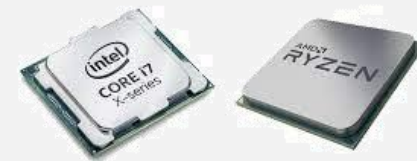
Computation rules can be executed by
hand, or by a **machine / automaton**



- $1 + 1 = ??$
- $= 10$

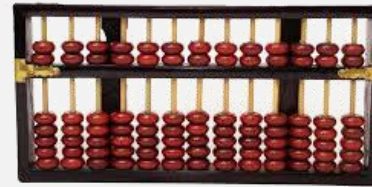
(binary)

There are many possible definitions
(models) of computation



(hint)

Many Different Kinds of Computation



This class is about:

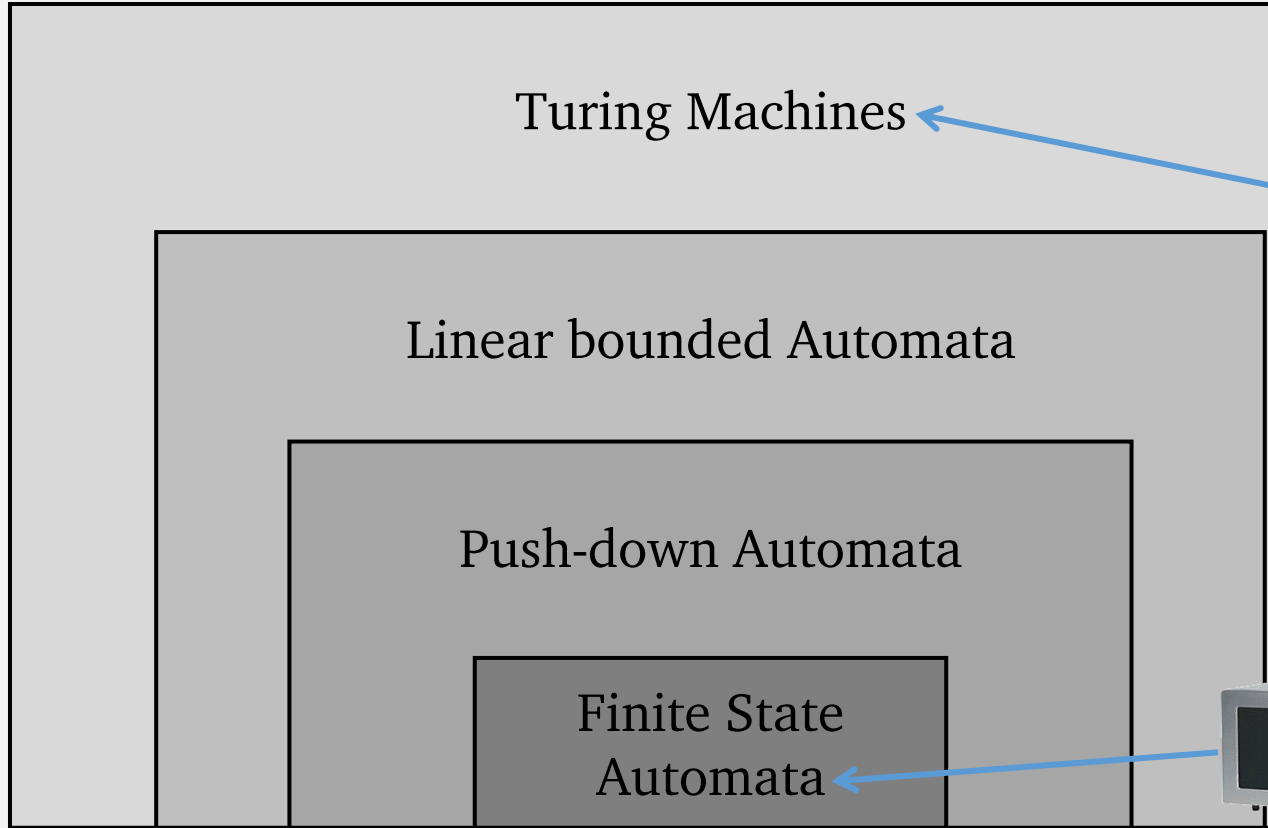
- formally defining models of computation,
- studying what they can and can't do,
- and studying their relation to each other!

(definitions + rules)

Models of Computation Hierarchy

... and get to here ...

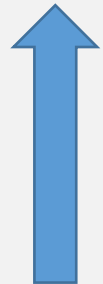
... and also look at what's out here??



```
1 import RPi.GPIO as GPIO
2 import time
3 import numpy as np
4 import cv2
5 from datetime import datetime
6 import os
7 import smtplib
8 from email.MIMEBase import MIMEBase
9 from email.MIMEText import MIMEText
10 from email import Encoders
11 gmail_user = "FROM MAIL ID@gmail.com" #Sender email address
12 gmail_pwd = "FROM MAIL PASSWORD" #Sender email password
13 to = "TO MAIL ID@gmail.com" #Receiver email address
14 subject = "Security Alert"
15 text = "There is some activity in your home. See the attached picture"
16
17
18 sensor = 4
19
20 GPIO.setmode(GPIO.BCM)
21 GPIO.setup(sensor, GPIO.IN, GPIO.PUD_DOWN)
22
```



More powerful
More complex
Less restricted



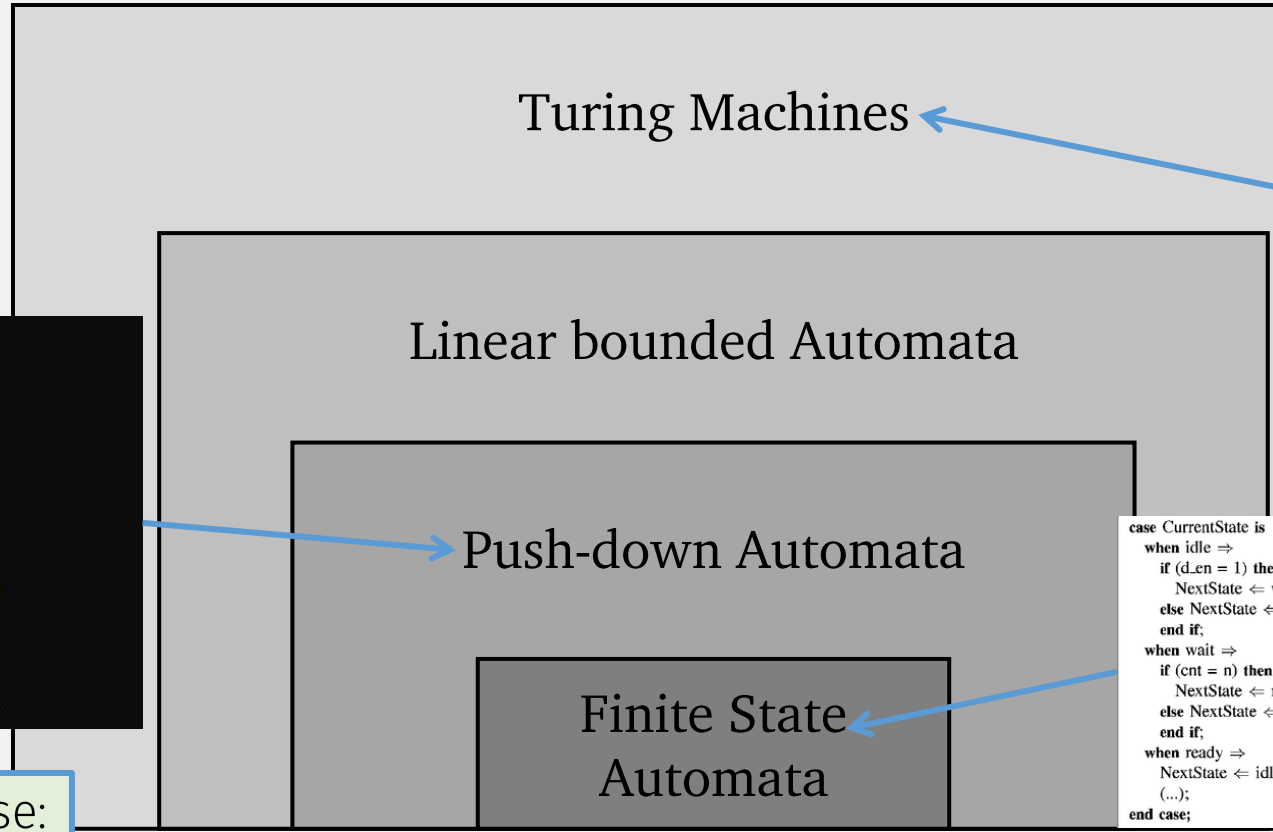
We'll start here ...

Computation = Programs!

```
Lesson 1 - CFG Grammar
E -> T E'
E' -> + T E'
E' ->
T -> F T'
T' -> * F T'
T' ->
F -> ( E )
F -> int

Lesson 2 - FA Lexer
NFA machine contains 11 total states

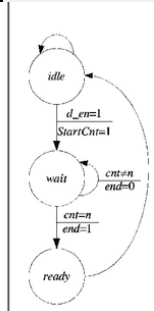
Lesson 3 - Runtime Parser
Reading expression "3+5*7"
NonTerminal E: , Line 1, Column 1
NonTerminal T: Line 1, Column 1
```



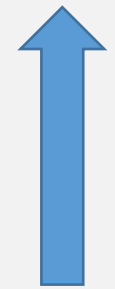
```
1 import RPi.GPIO as GPIO
2 import time
3 import numpy as np
4 import cv2
5 from datetime import datetime
6 import os
7 import smtplib
8 from email.MIMEText import MIMEText
9 from email.MIMEBase import MIMEBase
10 from email.MIMEText import MIMEText
11 from email import Encoders
12 gmail_user = "FROM MAIL ID@gmail.com" #Sender email address
13 gmail_pwd = "FROM MAIL PASSWORD" #Sender email password
14 to = "TO MAIL ID@gmail.com" #Receiver email address
15 subject = "Security Alert"
16 text = "There is some activity in your home. See the attached picture"
17
18 sensor = 4
19
20 GPIO.setmode(GPIO.BCM)
21 GPIO.setup(sensor, GPIO.IN, GPIO.PUD_DOWN)
22
```



```
case CurrentState is
when idle =>
if (d_en = 1) then
NextState <- wait; StartCnt <- 1;
else NextState <- idle;
end if;
when wait =>
if (cnt = n) then
NextState <- ready; end <- 1;
else NextState <- wait;
end if;
when ready =>
NextState <- idle;
(...);
end case;
```



More powerful
More complex
Less restricted



Remember for this course:

A model of computation = a class (set) of machines (each box above)

- Think of: a **class** of machines = a **Programming Language!**
- Think of: a **single** machine = a **Program!**

Welcome to CS420! *Models of*
Intro to Theory of Computation
& Programs

UMass Boston Computer Science

Instructor: Stephen Chang

Spring 2023

Welcome to CS420! *Models of*
Intro to Theory of Computation
& Programs

This class is about **formally** defining models of computation!

Spring 2023

“formally” = mathematically
(This is a math course, about programs!)

A (Mathematical) Theory is ...

Today's Theme:
CS 420 is useful?

Mathematical theory

From Wikipedia, the free encyclopedia

A **mathematical theory** is a **mathematical model** of a branch of mathematics that is based on a set of **axioms**. It can also simultaneously be a **body of knowledge** (e.g., based on known **axioms and definitions**), and so in this sense can refer to an area of mathematical research within the established framework.^{[1][2]}

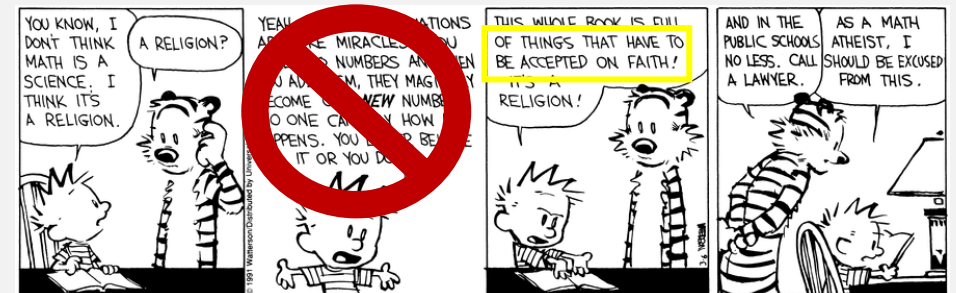
Explanatory depth is one of the most significant theoretical virtues in mathematics. For example, set theory has the ability to **systematize and explain** number theory and geometry/analysis. Despite the widely logical necessity (and self-evidence) of arithmetic truths such as $1 < 3$, $2 + 2 = 4$, $6 - 1 = 5$, and so on, a theory that just postulates an infinite blizzard of such truths would be inadequate. Rather an adequate theory is one in which such truths are derived from explanatorily prior axioms, such as the Peano Axioms or set theoretic axioms, which lie at the foundation of ZFC axiomatic set theory.

The singular accomplishment of axiomatic set theory is its ability to give a foundation for the derivation of the entirety of classical mathematics from a handful of axioms. The reason set theory is so prized is because of its explanatory depth. So a mathematical theory which just postulates an infinity of arithmetic truths without explanatory depth would not be a serious competitor to Peano arithmetic or Zermelo-Fraenkel set theory.^{[3][4]}

... a mathematical model, i.e., **axioms and definitions**, of some domain, e.g. computation ...

... that **explains (predicts)** some real-world phenomena ...

i.e., a theory must be useful in practice!

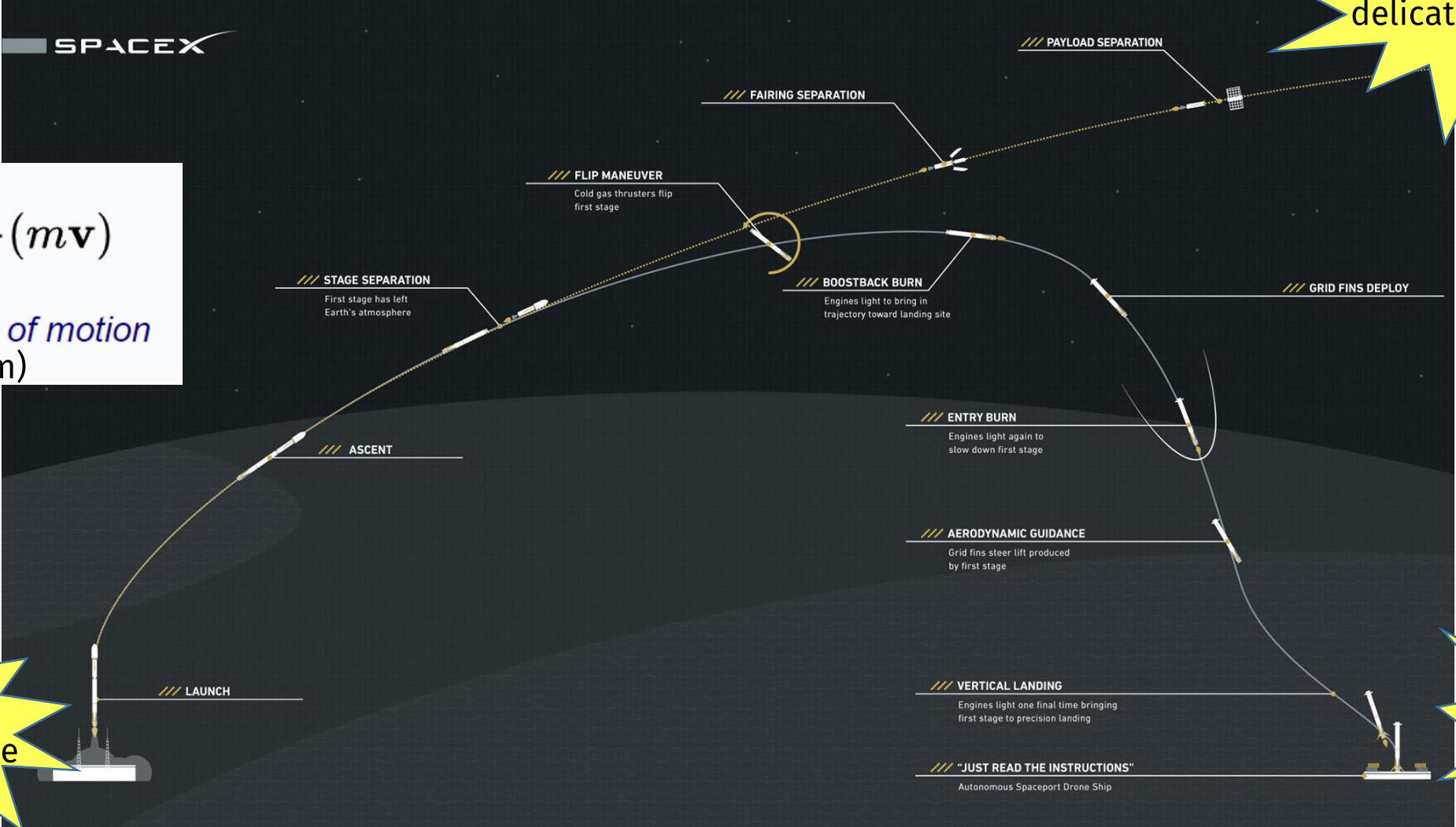


Example: Theory of Classical Mechanics

Drop off \$1b of delicate stuff here

$$\mathbf{F} = \frac{d}{dt}(m\mathbf{v})$$

Second law of motion
(axiom)



Do 1000 ton explosion here

Predicts landing exactly here

Why make predictions about computation?

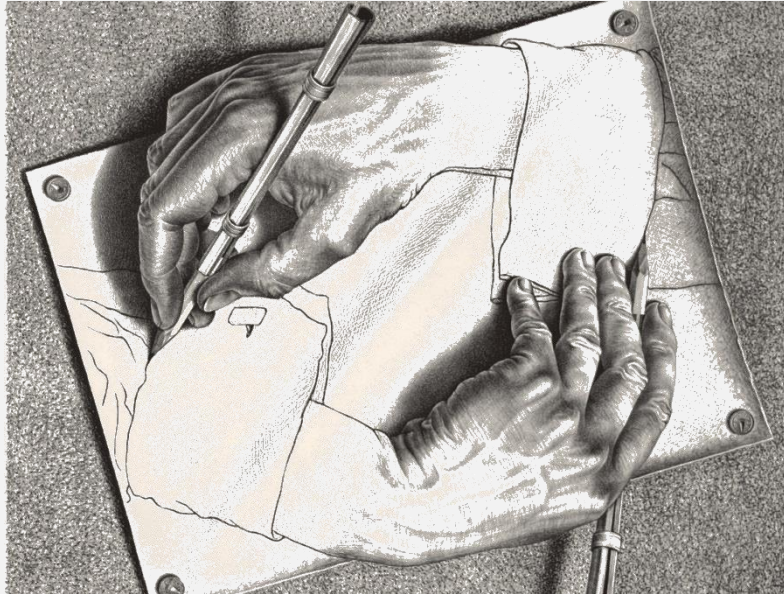
```
function check(n)
{ // check if the number n is a prime
  var factor; // if the checked number is not a prime, this is its first factor
  var c;
  factor = 0;
  // try to divide the checked number by all numbers till its square root
  for (c=2; (c <= Math.sqrt(n)); c++)
  {
    if (n%c == 0) // is n divisible by c ?
      { factor = c; break }
  }
  return (factor);
} // end of check function

function communicate()
{ // communicate with the user
  var i; // i is the checked number
  var factor; // if the checked number is not a prime, this is its first factor
  i = document.primetest.number.value; // get the checked number
  // is it a valid input?
  if ((i != N(i)) || (i <= 0) || (Math.floor(i) != i))
    { alert ("The checked object should be a whole positive number"); }
  else
  {
    factor = check (i);
    if (factor == 0)
      { alert (i + " is a prime number"); }
    else
      { alert (i + " is not a prime number. i =" + factor + "X" + i/factor) }
  }
} // end of communicate function
```



Predict result without running a program?

Can we make predictions about computation?



It's tricky: **Trying to predict computation requires computation!**

Can we make predictions about computation?

- The **Halting Lemma** says:



- And **Rice's Theorem** says:

- “all non-trivial, semantic properties of programs are undecidable”

- Actually:

- it depends on the computation model!



Predicting What Some Programs Will Do ...

microsoft.com/en-us/research/project/slam/

SLAM is a project for checking that software satisfies critical behavioral properties of the interfaces it uses and to aid software engineers in designing interfaces and software that ensure reliable and correct functioning. Static Driver Verifier is a tool in the Windows Driver Development Kit that uses the SLAM verification engine.

"Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification we're building tools that can do actual proof about the software and how it works in order to guarantee the reliability." **Bill Gates, April 18, 2002. [Keynote address at WinHec 2002](#)**

Holy grail of CS!

SLAM
`if(!node->); i ++ vis; proc; end() node){`

Static Driver Verifier Research Platform README

Overview of Static Driver Verifier Research Platform

Static Driver Verifier (SDV) is a compile-time static verification tool, included in the Windows Driver Kit (WDK). The SDV Research Platform (SDVRP) is an extension to SDV that allows you to adapt SDV to:

- Support additional frameworks (or APIs) and write custom SLIC rules for this framework.
- Experiment with the model checking step.

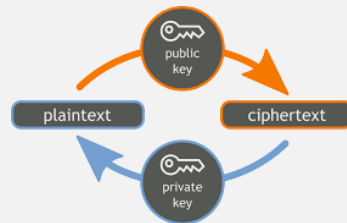
Today's Theme:
CS 420 is useful ... for
advanced programmers?



Knowing What Computers Can't Do is Still Useful!

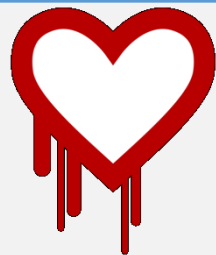
In Cryptography:

- Perfect secrecy is impossible in practice
- But with slightly imperfect secrecy (i.e., a computationally bounded adversary) we get:

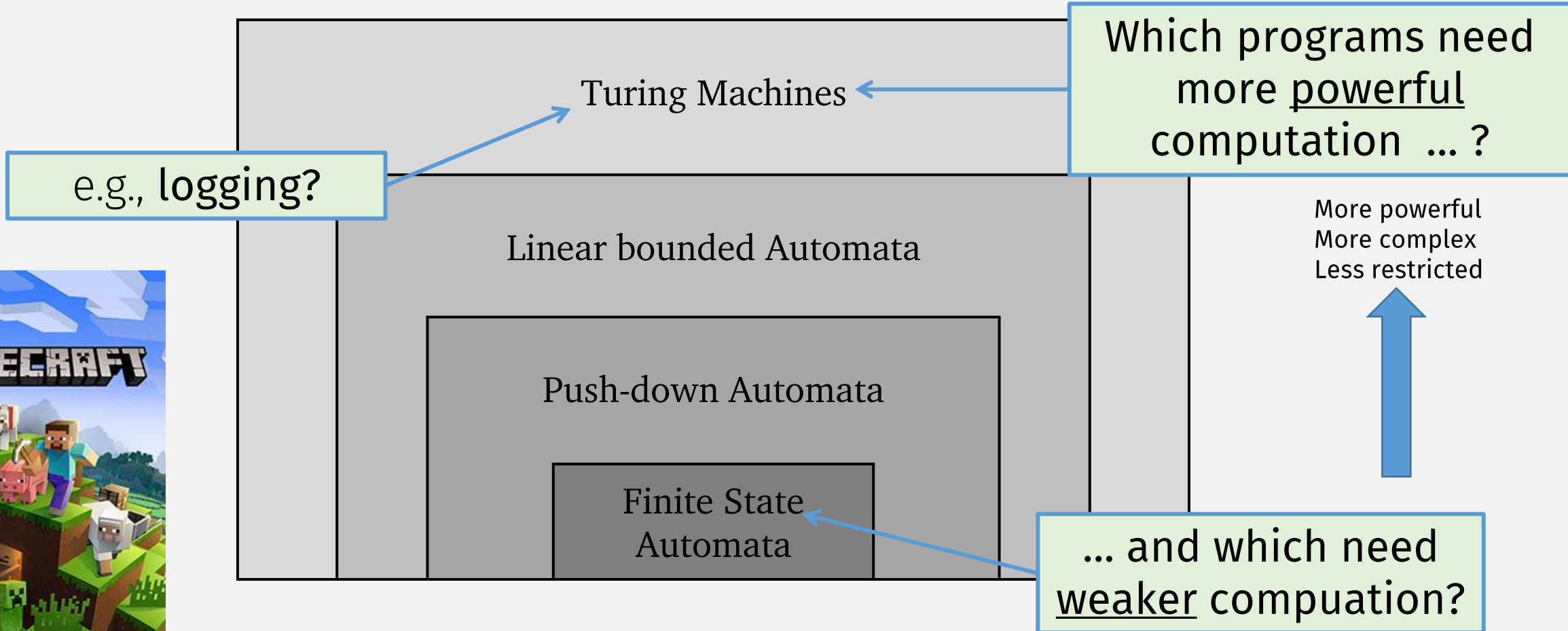


- But, even with a strong math foundation, there are still problems:

• Caused by programmers who don't understand theory of computation!



Which Computation Model to Choose?



The Computing Power of Logs?

Log4Shell

From Wikipedia, the free encyclopedia

Log4Shell (CVE-2021-44228) was a [zero-day](#) vulnerability in [Log4j](#), a popular [Java logging framework](#), involving [arbitrary code execution](#).^{[2][3]} The vulnerability has existed unnoticed since 2013 and was privately disclosed to [the Apache Software Foundation](#), of which Log4j is a project, by Chen Zhaojun of [Alibaba Cloud's](#) security team on 24 November 2021, and was publicly disclosed on 9 December 2021.^{[1][4][5][6]} Apache gave Log4Shell a [CVSS](#) severity rating of 10, the highest available score.^[7] The exploit is simple to execute and is estimated to affect hundreds of millions of devices.^{[6][8]}

The vulnerability takes advantage of Log4j's allowing requests to arbitrary [LDAP](#) and [JNDI](#) servers,^{[2][9][10]} allowing attackers to execute arbitrary Java code on a server or other computer, or leak sensitive information.^[5] A list of its affected software projects has been published by the [Apache Security Team](#).^[11] Affected commercial services include [Amazon Web Services](#),^[12] [Cloudflare](#), [iCloud](#),^[13] *[Minecraft: Java Edition](#)*,^[14] [Steam](#), [Tencent QQ](#) and many others.^{[9][15][16]} According to [Wiz](#) and [EY](#), the vulnerability affected 93% of enterprise cloud environments.^[17]

Log4Shell	
CVE identifier(s)	CVE-2021-44228 ↗
Date discovered	24 November 2021; 57 days ago
Date patched	6 December 2021; 45 days ago
Discoverer	Chen Zhaojun of the Alibaba Cloud Security Team ^[1]
Affected software	Applications logging user input using Log4j 2

The Computing Power of Fonts?



BIZ & IT TECH SCIENCE POLICY CARS GAMING & C

IN THE WILD —

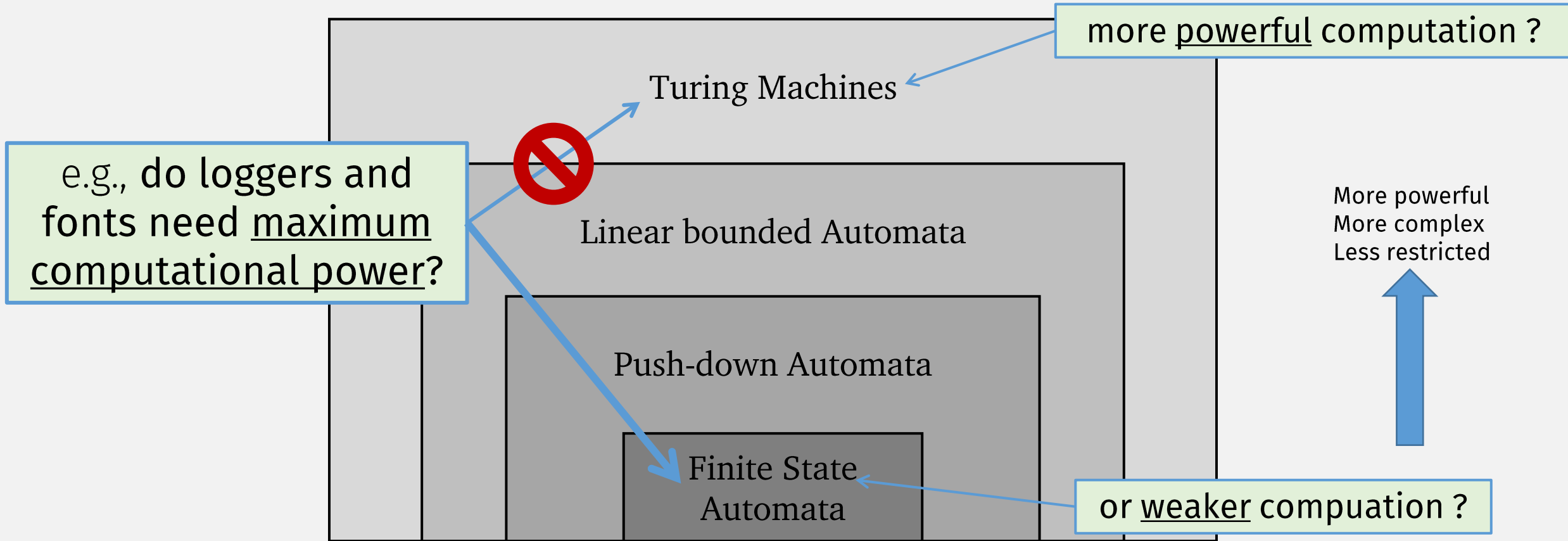
Windows code-execution zeroday is under active exploit, Microsoft warns

There's no patch available now. Here's what to do until Microsoft issues one.

DAN GOODIN - 3/23/2020, 3:40 PM

The **font-parsing remote code-execution vulnerability** is being used in "limited targeted attacks," against Windows 7 systems, the software maker said in an **advisory published on Monday morning**. The security flaw exists in the Adobe Type Manager Library, a Windows DLL file that a wide variety of apps use to manage and render fonts available from Adobe Systems. The vulnerability consists of two code-execution flaws that can be triggered by the improper handling of maliciously crafted master fonts in the Adobe Type 1 Postscript format. Attackers can exploit them by convincing a target to open a booby-trapped document or viewing it in the Windows preview pane.

Which Computation Model to Choose?



Today's Theme:
CS 420 is useful ... for all programmers!

A (Mathematical) Theory Is ...

Mathematical theory

From Wikipedia, the free encyclopedia

A **mathematical theory** is a **mathematical model** of a branch of mathematics that is based on a set of **axioms**. It can also simultaneously be a **body of knowledge** (e.g., based on known **axioms and definitions**), and so in this sense can refer to an area of mathematical research within the established framework.^{[1][2]}

Explanatory depth is one of the most significant theoretical virtues in mathematics. For example, set theory has the ability to **systematize and explain** number theory and geometry/analysis. Despite the widely logical necessity (and self-evidence) of arithmetic truths such as $1 < 3$, $2 + 2 = 4$, $6 - 1 = 5$, and so on, a theory that just postulates an infinite blizzard of such truths would be inadequate. Rather an adequate theory is one in which such truths are derived from explanatorily prior axioms, such as the Peano Axioms or set theoretic axioms, which lie at the foundation of ZFC axiomatic set theory.

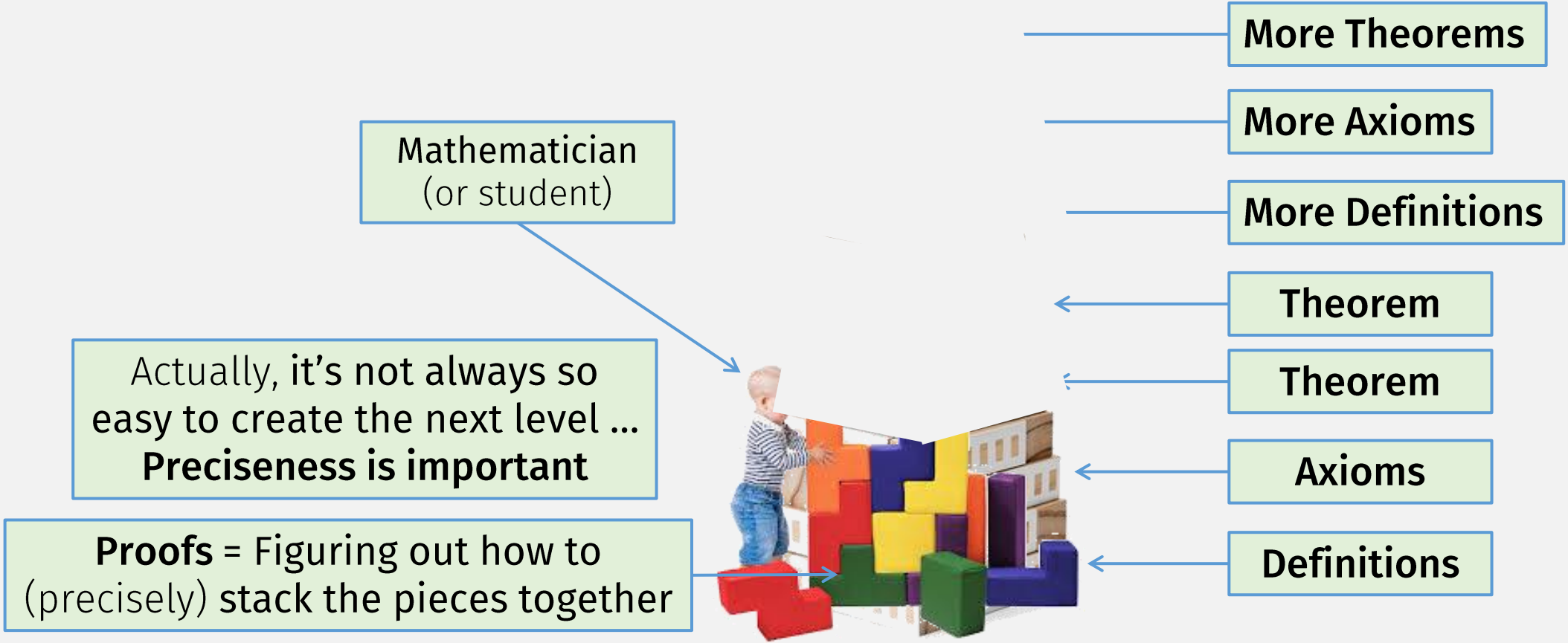
The singular accomplishment of axiomatic set theory is its ability to give **a foundation for the derivation of the entirety of classical mathematics** from a handful of axioms. The reason set theory is so prized is because of its explanatory depth. So a mathematical theory which just postulates an infinity of arithmetic truths without explanatory depth would not be a serious competitor to Peano arithmetic or Zermelo-Fraenkel set theory.^{[3][4]}

... a mathematical model, i.e., **axioms and definitions**, of some domain, e.g. computers ...

... that **explains (predicts)** some real-world phenomena ...

... and can **derive** additional results (**lemmas & theorems**) ...

How Mathematics Works



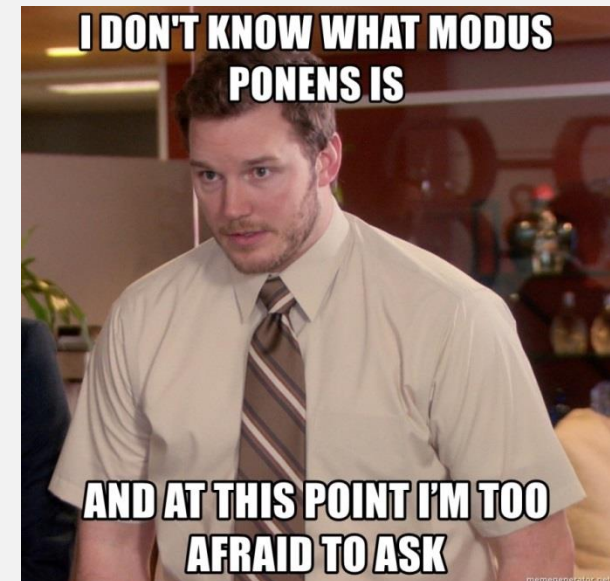
Precisely Fitting Blocks Together: The “Modus Ponens” Inference Rule

Premises (if we can show these statements are true)

- If P then Q
- P is TRUE

Conclusion (then we can say that this is also true)

- Q must also be TRUE



Kinds of Mathematical Proof

Deductive Proof

- Start with known facts and statements
- Use logical **inference rules** (like modus ponens) to prove new facts and statements

Deductive Proof Example

Prove the following:

- If: $x \geq 4$, then $2^x \geq x^2$ ← Given (inputs)
- And: x is the sum of the squares of four positive integers
- Then: $2^x \geq x^2$ ← Need to show this (need to get this output)

Deductive Proof Example

Prove: If $x \geq 4$, then $2^x \geq x^2$ and x is the sum of the squares of four positive integers then $2^x \geq x^2$

Proof:

Statement

1.

Justification

6. $2^x \geq x^2$

Need to prove this ...

Deductive Proof Example

Prove: If $x \geq 4$, then $2^x \geq x^2$ and x is the sum of the squares of four positive integers then $2^x \geq x^2$

... using known facts

Proof:

Statement

- 1. $x = a^2 + b^2 + c^2 + d^2$
- 2. $a \geq 1; b \geq 1; c \geq 1; d \geq 1$

Justification

- 1. Given
- 2. Given

6. $2^x \geq x^2$

Deductive Proof Example

Prove: If $x \geq 4$, then $2^x \geq x^2$ and x is the sum of the squares of four positive integers then $2^x \geq x^2$

Proof:

Statement

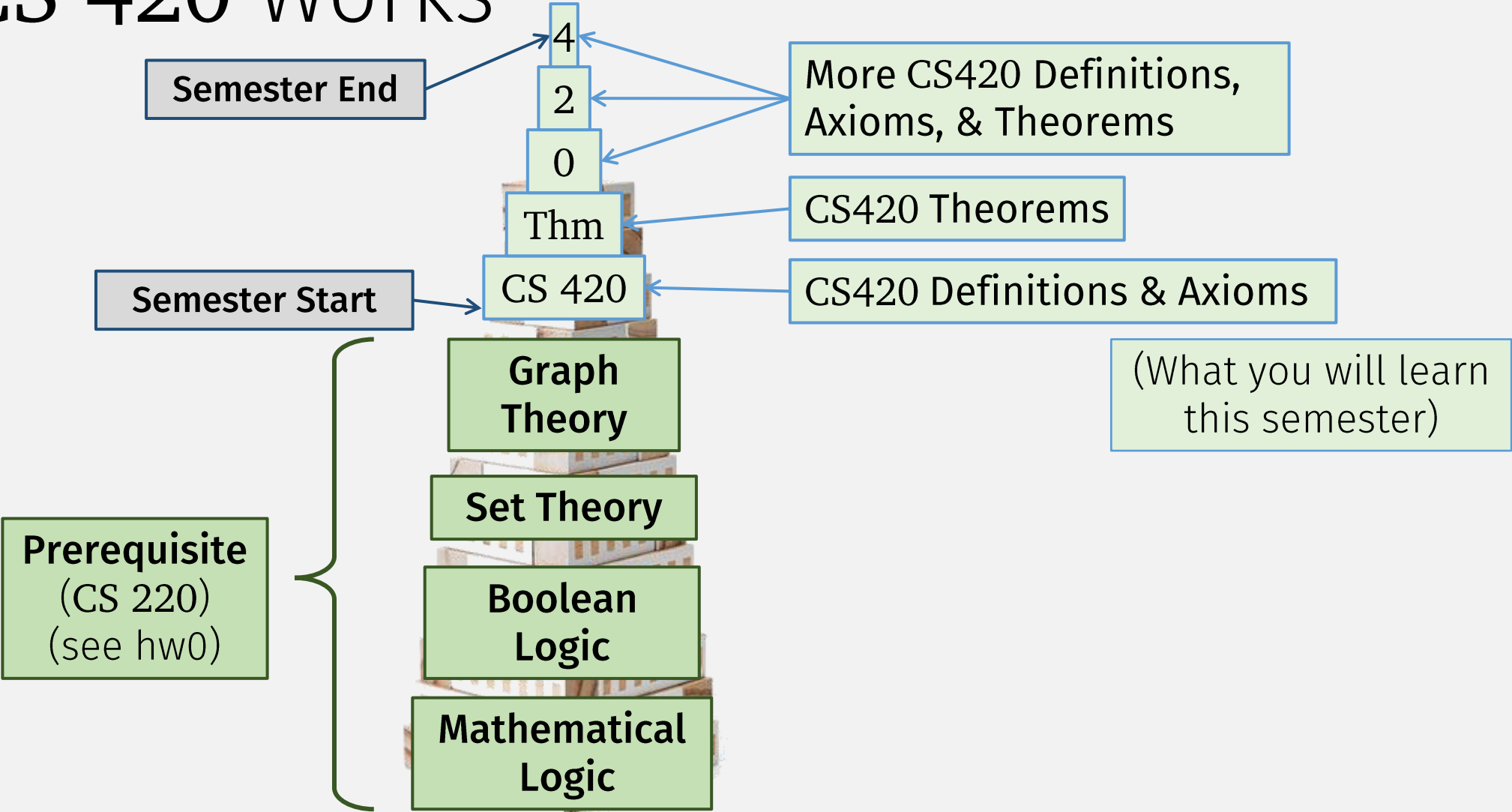
1. $x = a^2 + b^2 + c^2 + d^2$
2. $a \geq 1; b \geq 1; c \geq 1; d \geq 1$
3. $a^2 \geq 1; b^2 \geq 1; c^2 \geq 1; d^2 \geq 1$
4. $x \geq 4$
5. If $x \geq 4$, then $2^x \geq x^2$
6. $2^x \geq x^2$

Justification

1. Given
2. Given
3. By Step (2) & arithmetic laws
4. (1), (3), and arithmetic
5. Given
6. (4) and (5)

Modus ponens

How CS 420 Works



A Word of Advice

Important:
Do not fall behind
in this course



To prove a (new) theorem ...

... need to know all axioms,
definitions, and (previous)
theorems below it

Another Word of Advice

HW 1, Problem 1

Prove that $ABC = XYZ$



Q Prove that $ABC = XYZ$

A Not-From-CS421-
Spring2023 Theorem

Google Search

“Blocks” from outside the course won’t work in the proof

Remember:

Preciseness is important

(Proofs must connect facts from this course exactly)

HW problems *graded* on how you got to the answer, not on the final answer itself!

... can be used to **prove** (new) theorems in this course

Only axioms, definitions, and theorems from this course...

How to Do Well in this Course

- Learn the “building blocks”
 - I.e., axioms, definitions, and theorems
- To solve a problem (prove a new theorem) ...
... think about how to (precisely) combine existing “blocks”
- HW problems graded on how you got to the answer, not on the final answer itself!
- Don't Fall Behind!
 - Start HW Early (HW 0 due Sunday 11:59pm EST)
- Participate and Engage
 - Lecture
 - Office Hours
 - Message Boards (piazza)

Textbooks

- Sipser. *Intro to Theory of Computation*, 3rd ed.
 - Hopcroft, Motwani, Ullman. *Intro to Automata Theory, Languages, and Computation*, 3rd ed.
- Recommended but not required,
 - slides and lecture should be self-contained,
 - Readings to accompany lectures will be posted

All course info available on web site:
<https://www.cs.umb.edu/~stchang/cs420/s23>

Grading

- **HW: 80%**
 - Weekly: Out Monday, In Sunday
 - Approx. 12 assignments
 - Lowest grade dropped
- **Quizzes: 5%**
 - End of every lecture
 - To help everyone keep up
- **Participation: 15%**
 - Lecture, office hours, piazza
- **No exams**
- **A range: 90-100**
- **B range: 80-90**
- **C range: 70-80**
- **D range: 60-70**
- **F: < 60**

All course info available on web site:
<https://www.cs.umb.edu/~stchang/cs420/s23>

Late HW

- Is bad ... try not to do it please
 - Grades get delayed
 - Can't discuss solutions
 - You fall behind!
- Late Policy: **3 late days** to use during the semester

HW Collaboration Policy

Allowed

- Discussing HW with classmates (but must cite)
- Using other resources, e.g., youtube, other books, etc.
- Writing up answers on your own, from scratch, in your own words

Not Allowed

- Submitting someone else's answer
- It's still someone else's answer if:
 - variables are changed,
 - words are omitted,
 - or sentences rearranged ...
- Using sites like Chegg, CourseHero, Bartleby, Study, etc.
- Can't use theorems or definitions not from this course

Honesty Policy

- 1st offense: zero on problem
- 2nd offense: zero on hw, reported to school
- 3rd offense+: F for course

Regret policy

- If you self-report an honesty violation, you'll only receive a zero on the problem and we move on.

All Up to Date Course Info

Survey, Schedule, Office Hours, HWs, ...

See course website:

<https://www.cs.umb.edu/~stchang/cs420/s23/>

Check-In Quiz 1/23

(see gradescope)