COMP-330
Theory of Computation

Fall 2019 -- Prof. Claude Crépeau

Lecture 1 : Introduction

COMP-330
Theory of Computation

Fall 2019 -- Prof. Claude Crépeau

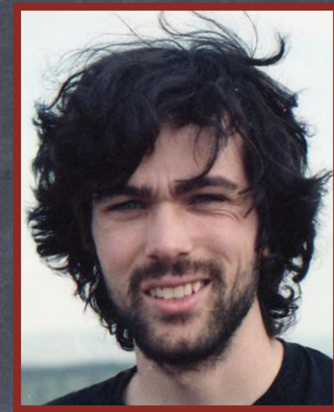
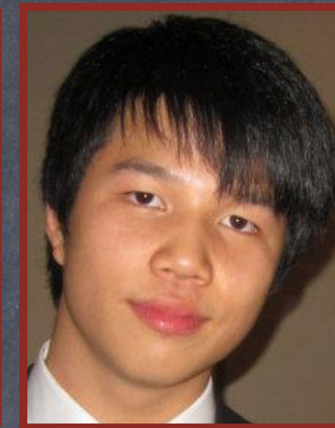
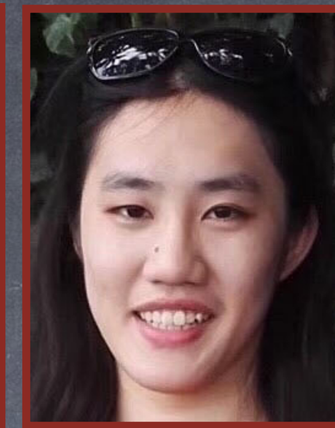
COURSE OUTLINE

COMP 330 Fall 2019

- **Class Schedule** :
  Tuesday-Thursday 13:05-14:25 MAASS 112

- **Instructor** :
  Prof. Claude Crépeau

- **Office** : Room 110N,
  McConnell Eng. Building
  phone: (514) 398-4716
  email: crepeau@cs.mcgill.ca

- **2019 T.A.s** :

Pouriya Alikhani	pouriya.alikhani@mail.mcgill.ca
Pierre-William Breau	pierre-william.breau@mail.mcgill.ca
Anirudha Jita	anirudha.jitani@mail.mcgill.ca
Justin Li	juan.y.li@mail.mcgill.ca
Yanjia Li	yanjia.li@mail.mcgill.ca
Shiquan Zhang	shiquan.zhang@mail.mcgill.ca

- **Office Hours** :

Claude : Wednesday 13:00-16:00 ENGMC 110N

Pouriya : Friday 13:00-14:00 ENGTR 3090

Pierre-William : Monday 15:00-16:00 ENGTR 3110

Anirudha : Monday 16:00-17:00 ENGTR 3090

Justin : Tuesday 15:00-16:00 ENGTR 3110

Yanjia : Friday 10:00-11:00 ENGTR 3110

Shiquan : Thursday 15:00-16:00 ENGTR 3110

COMP-330 Fall 2019 — Weekly Schedule

Mon 10:00	**Tue** 10:00	**Wed** 10:00	**Thu** 10:00	Yanjia TR-3110
Mon 10:30	**Tue** 10:30	**Wed** 10:30	**Thu** 10:30	
Mon 11:00	**Tue** 11:00	**Wed** 11:00	**Thu** 11:00	**Fri** 11:00
Mon 11:30	**Tue** 11:30	**Wed** 11:30	**Thu** 11:30	**Fri** 11:30
Mon 12:00	**Tue** 12:00	**Wed** 12:00	**Thu** 12:00	**Fri** 12:00
Mon 12:30	**Tue** 12:30	**Wed** 12:30	**Thu** 12:30	**Fri** 12:30
Mon 13:00	Claude MA-112 course	Claude MC-110N office hours	Claude MA-112 course	Pouriya TR-3090
Mon 13:30				
Mon 14:00				**Fri** 14:00
Mon 14:30	**Tue** 14:30		**Thu** 14:30	**Fri** 14:30
Pierre-W. TR-3110	Justin TR-3110		Shiquan TR-3110	**Fri** 15:00
				Fri 15:30
Anirudha TR-3090	**Tue** 16:00	**Wed** 16:00	**Thu** 16:00	**Fri** 16:00
	Tue 16:30	**Wed** 16:30	**Thu** 16:30	**Fri** 16:30

MC = MCENG = McConnell • TR = ENGTR = Trottier

COMMUNICATION

www:

http://crypto.cs.mcgill.ca/~crepeau/COMP330/

email:

cs330@cs.mcgill.ca

FaceBook:

COMP 330 Fall 2019

COMP-330 Fall 2019 — Extra help !

CSUS Helpdesk

HOURS: 12pm – 5pm (mon-fri)
LOCATION: Trottier 3090

WHO ARE WE? WHAT DO WE DO?

- U2 and U3 students who have taken this course and want to help you!
- We are a FREE drop-in tutoring service, perfect for study help, and guidance on assignments.
- We provide review sessions for midterms and finals for intro courses!

COMP-330 Fall 2019 — Weekly Schedule

Mon 10:00	**Tue** 10:00	**Wed** 10:00	**Thu** 10:00	Yanjia TR-3110
Mon 10:30	**Tue** 10:30	**Wed** 10:30	**Thu** 10:30	
Mon 11:00	**Tue** 11:00	**Wed** 11:00	**Thu** 11:00	**Fri** 11:00
Mon 11:30	**Tue** 11:30	**Wed** 11:30	**Thu** 11:30	**Fri** 11:30
CSUS Helpdesk TR-3090	**CSUS**	**CSUS**	**CSUS**	
	Claude MA-112	Claude MC-110N office hours	Claude MA-112	Pouriya TR-3090
				CSUS Helpdesk TR-3090
Pierre-W. TR-3110	Justin TR-3110		Shiquan TR-3110	
Anirudha TR-3090	**Helpdesk TR-3090**	**Helpdesk TR-3090**	**Helpdesk TR-3090**	

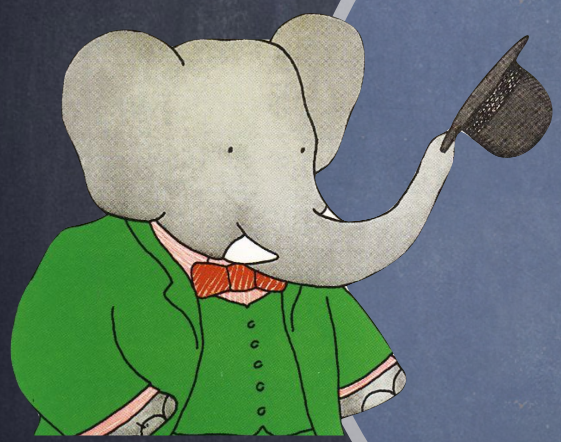
MC = MCENG = McConnell • TR = ENGTR = Trottier

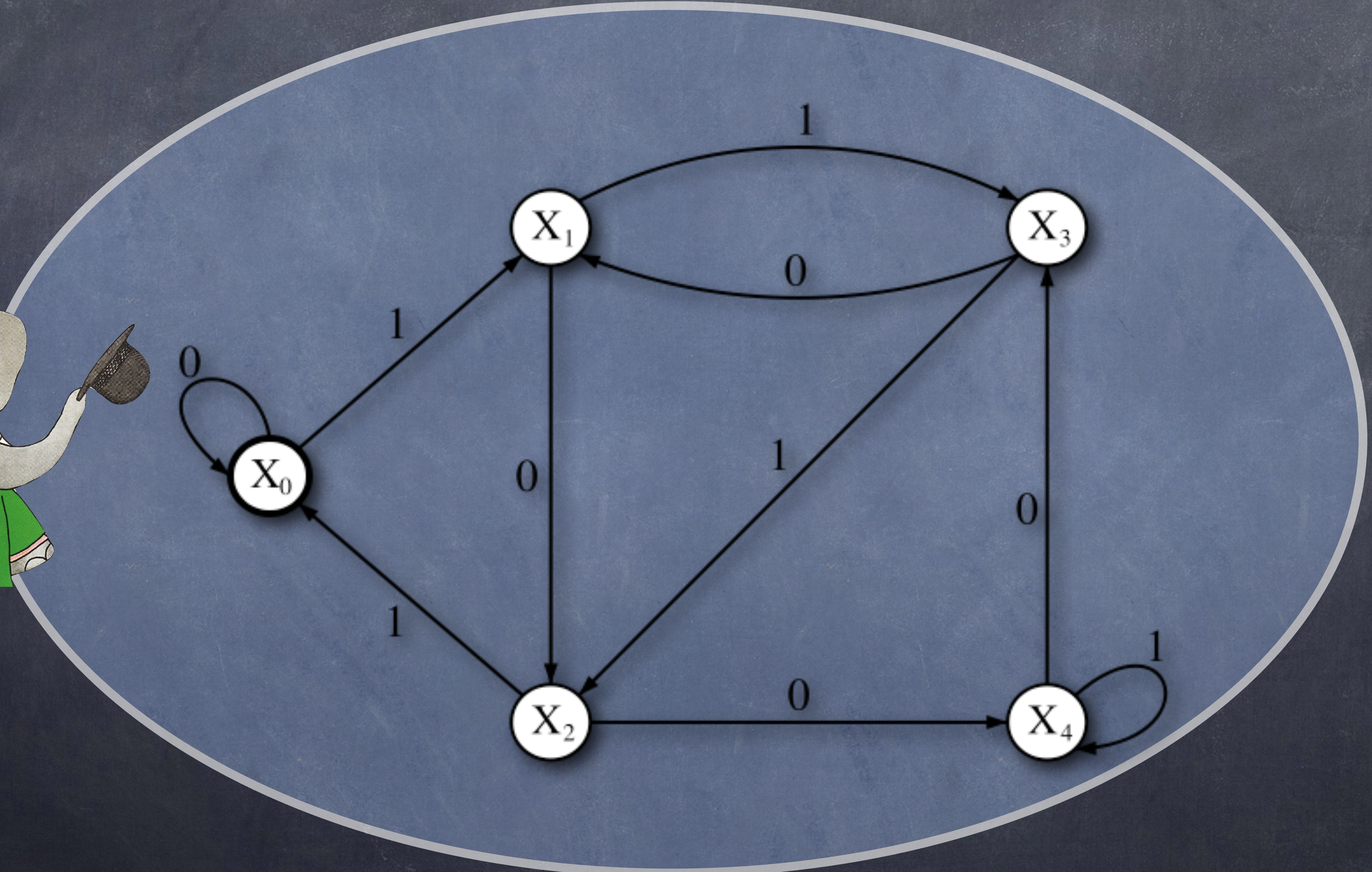
COMP 330 Fall 2019

Description: (3 credits, 3 hours).

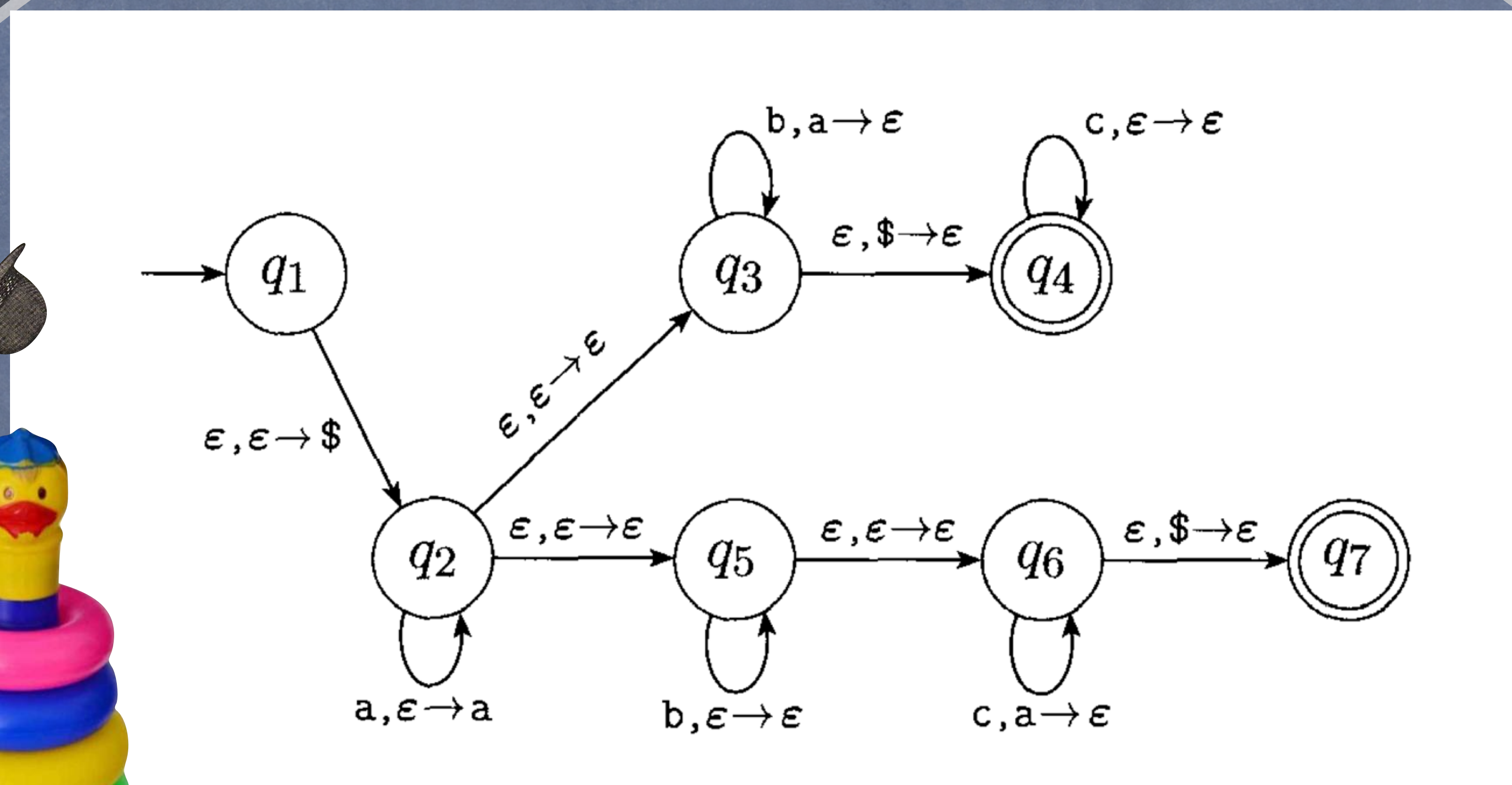
- We study models of computation of increasing power.
- We begin with finite automata and regular languages.
- The next phase deals with context-free languages invented by linguists and now an essential aspect of every modern programming language.
- Finally we explore the limits of computability with the study of recursive sets, enumerable sets, self-reproducing programs and undecidability theory.

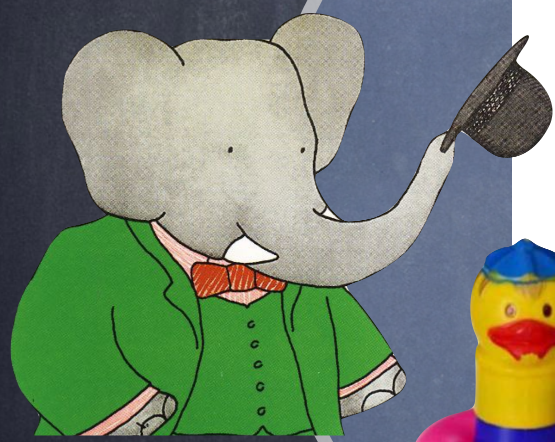
Part 1: Regular expressions & Deterministic Finite Automata



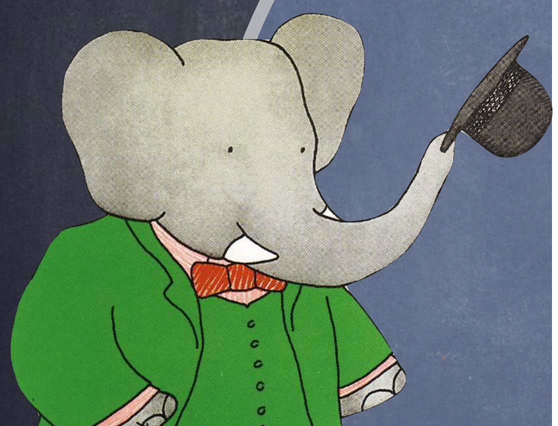


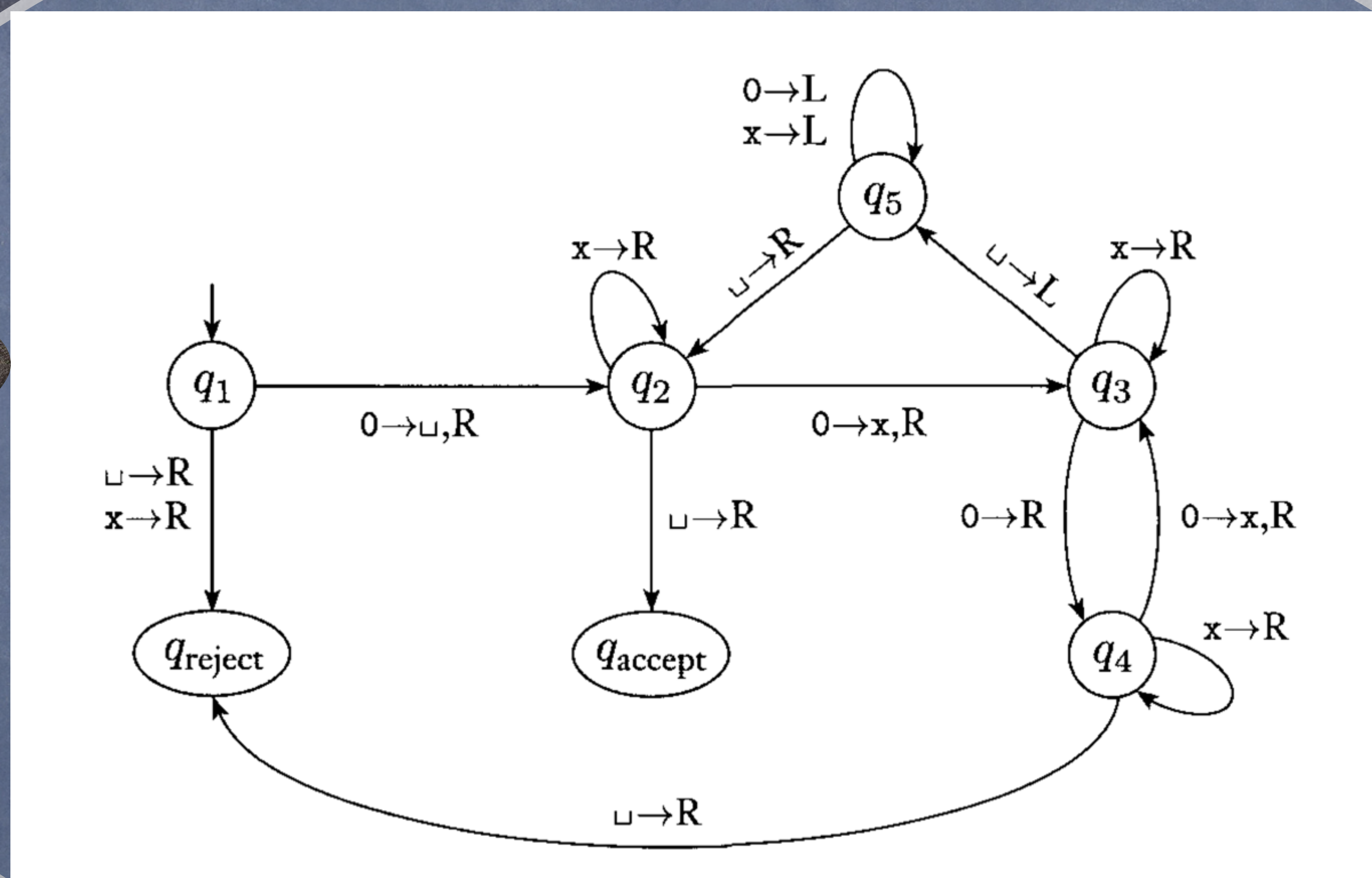
Part 2: Context-free Language & Pushdown Automata

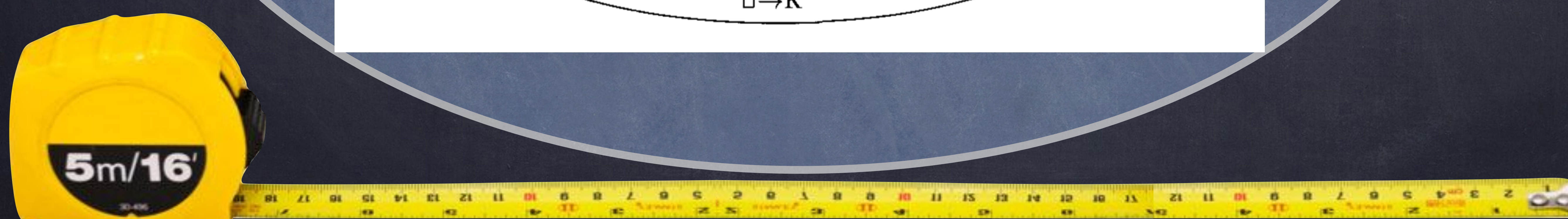




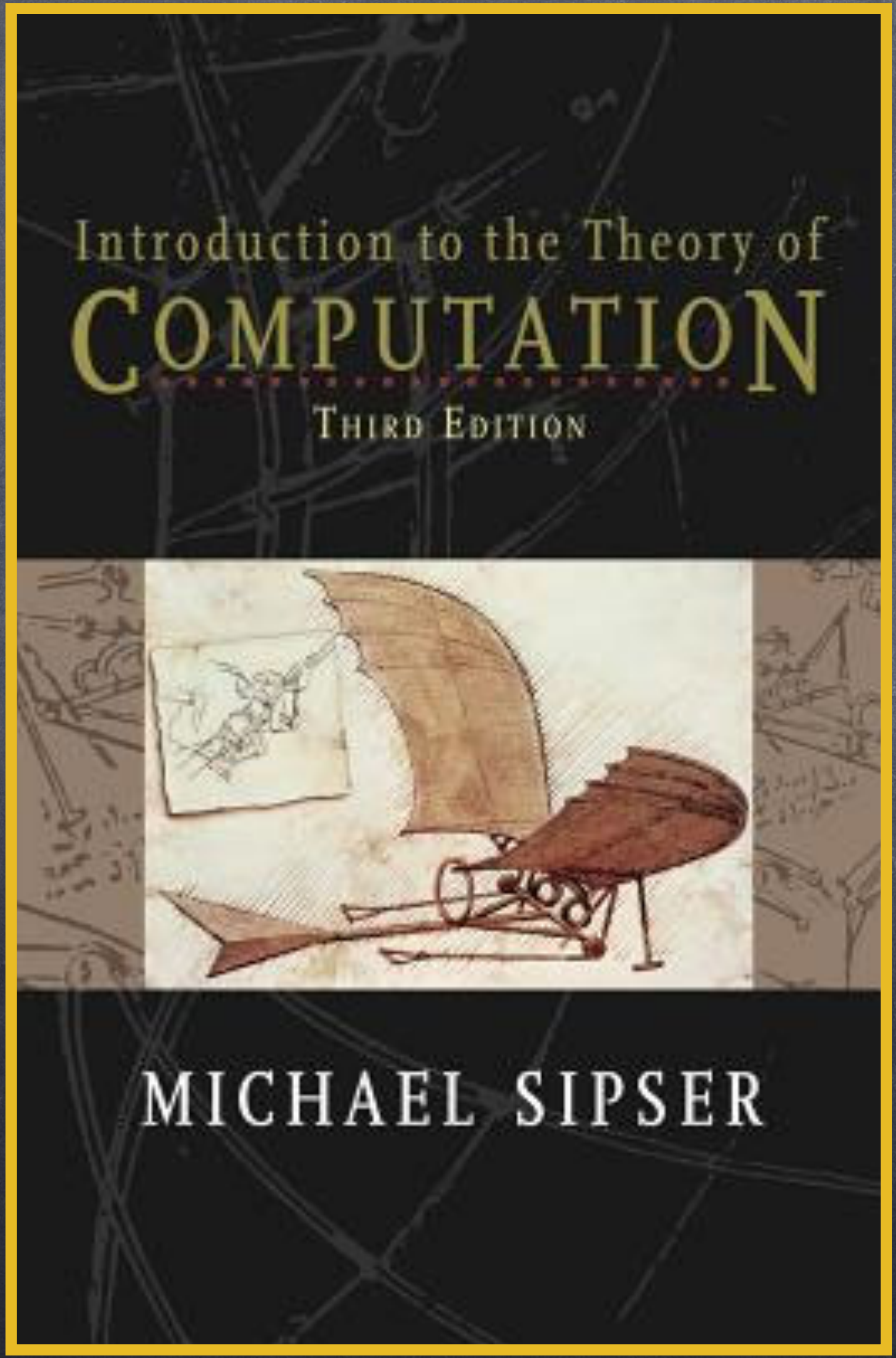
Part 3: Turing Machines, Computability & Complexity







Mandatory Textbook



COMP 330 Fall 2019: Lectures Schedule

1. Introduction
   1.5. Some basic mathematics
2. Regular expressions, DFAs
3. Nondeterministic finite automata
4. Determinization
5. Closure properties, Kleene's theorem
6. The pumping lemma
7. The pumping lemma
8. Minimization
9. Lexical analysis
10. Duality
11. Myhill-Nerode theorem
12. Labelled transition systems
13. MIDTERM
14. Context-free languages
15. Pushdown automata
16. Parsing
17. The pumping lemma for CFLs
18. Introduction to computability
19. Models of computation

    Basic computability theory

20. Reducibility, undecidability and Rice's theorem
21. Undecidable problems about CFGs
22. Post Correspondence Problem
23. *Validity of FOL is RE / Gödel's and Tarski's thms*
24. *Universality / The recursion theorem*
25. *Degrees of undecidability*
26. Introduction to complexity

COMP 330 Fall 2019

- **Evaluation**:
There will be
• 4 assigments worth 40%,
• a midterm exam worth 10%, and
• a final exam worth 50%
of your final grade.

COMP 330 Fall 2019

- In accord with McGill University's Charter of Students' Rights, students in this course have the right to submit in English or in French any written work that is to be graded.
- En vertu de la chartre des droits des étudiants de l'université McGill, les étudiants de ce cours ont le droit de soumettre leurs travaux écrits en anglais ou en français, à leur guise.

COMP 330 Fall 2019

- **Academic integrity** : McGill University values academic integrity. Therefore all students must understand the meaning and consequences of cheating, plagiarism and other academic offences under the Code of Student Conduct and Disciplinary Procedures (see http://www.mcgill.ca/students/srr/honest for more info).

- **Honnêteté académique** : L'université McGill attache une grande importance à l'honnêteté académique. Il incombe par conséquent à tous les étudiants de comprendre ce que l'on entend par tricherie, plagiat et autres infractions académiques, ainsi que les conséquences que peuvent avoir de telles actions, selon le Code de conduite de l'étudiant et des procédures disciplinaires (pour de plus amples renseignements, consultez http://www.mcgill.ca/students/srr/honest).

COMP-330
Theory of Computation

Fall 2019 -- Prof. Claude Crépeau

COURSE OUTLINE

COMP 330 Fall 2017: Lectures Schedule

<u>**1-2. Introduction**</u>
1.5. Some basic mathematics
2-3. Deterministic finite automata +Closure properties,
3-4. Nondeterministic finite automata
5. Minimization+ Myhill-Nerode theorem
6. Determinization+Kleene's theorem
7. Regular Expressions+GNFA
8. Regular Expressions and Languages
9-10. The pumping lemma
11. Duality
12. Labelled transition systems
13. MIDTERM
14. Context-free languages
15. Pushdown automata
16. Parsing
17. The pumping lemma for CFLs
18. Introduction to computability
19. Models of computation Basic computability theory
20. Reducibility, undecidability and Rice's theorem
21. Undecidable problems about CFGs
22. Post Correspondence Problem
23. Validity of FOL is RE / Gödel's and Tarski's thms
24. Universality / The recursion theorem
25. Degrees of undecidability
26. Introduction to complexity

INTRODUCTION

Paris, 1900

Paris, 1900

- German mathematician David Hilbert presented ten problems in mathematics from a list of 23 (1, 2, 6, 7, 8, 13, 16, 19, 21 and 22).

Paris, 1900

- German mathematician David Hilbert presented ten problems in mathematics from a list of 23 (1, 2, 6, 7, 8, 13, 16, 19, 21 and 22).
- Speaking on 8 August 1900, at the Paris 2nd International Congress of Mathematicians, at La Sorbonne. The full list was published later.

Paris, 1900

- German mathematician David Hilbert presented ten problems in mathematics from a list of 23 (1, 2, 6, 7, 8, 13, 16, 19, 21 and 22).
- Speaking on 8 August 1900, at the Paris 2nd International Congress of Mathematicians, at La Sorbonne. The full list was published later.
- The problems were all unsolved at the time, and several of them turned out to be very influential for 20th century mathematics.

Fundamental questions ?

Fundamental questions ?

- Can we prove all the mathematical statements that we can formulate ?
  (Hilbert's 2nd problem)
  Example: for all natural integers x,y,z and n>2
  $x^n+y^n \neq z^n$ (Fermat's last theorem)

Fundamental questions ?

- Can we prove all the mathematical statements that we can formulate ? (Hilbert's 2nd problem)
  Example: for all natural integers x,y,z and n>2
  $x^n+y^n \neq z^n$ (Fermat's last theorem)

- Certainly, there are many mathematical problems that we do not know how to solve.

Fundamental questions ?

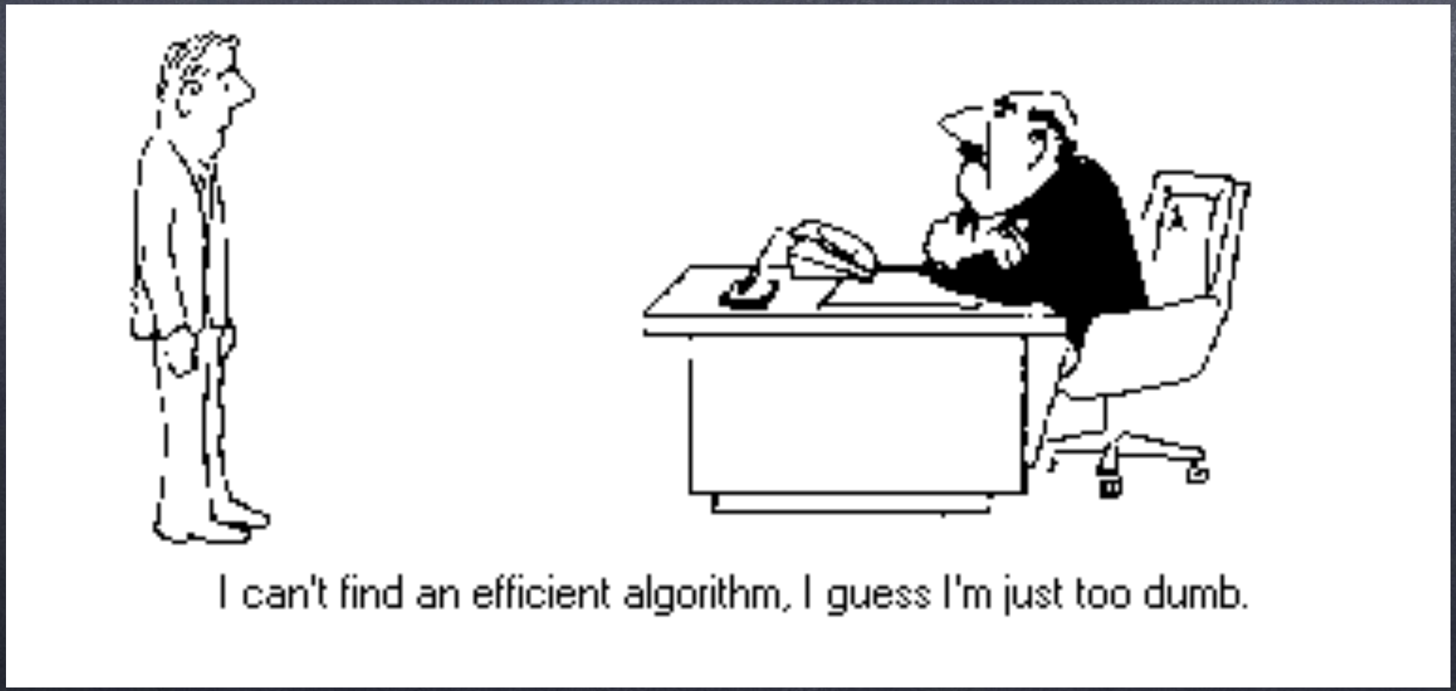
- Can we prove all the mathematical statements that we can formulate ? (Hilbert's 2nd problem)
  Example: for all natural integers x,y,z and n>2
  $x^n+y^n \neq z^n$ (Fermat's last theorem)
- Certainly, there are many mathematical problems that we do not know how to solve.
- Is this just because we are not smart enough to find a solution ?

Fundamental questions ?

- Can we prove all the mathematical statements that we can formulate ? (Hilbert's 2nd problem)
  Example: for all natural integers x,y,z and n>2
  $x^n+y^n \neq z^n$ (Fermat's last theorem)
- Certainly, there are many mathematical problems that we do not know how to solve.
- Is this just because we are not smart enough to find a solution ?
- Or, is there something deeper going on ?

Computer Science version of these issues

- If my boss / supervisor / teacher formulates a problem to be solved urgently, can I write a program to efficiently solve this problem ???



Simple Puzzle

95% of people cannot solve this!

$$\frac{🍎}{🍌+🍍} + \frac{🍌}{🍍+🍎} + \frac{🍍}{🍎+🍌} = 4$$

Can you find positive whole values for 🍎, 🍌, and 🍍?

Simple Puzzle

~~95%~~ of people cannot solve this!
99.9999%

🍎/(🍌+🍍) + 🍌/(🍍+🍎) + 🍍/(🍎+🍌) = 4

Can you find positive whole values for 🍎, 🍌, and 🍍?

Simple Puzzle

~~95%~~ of people cannot solve this!

99.9999%

$$\frac{🍎}{🍌+🍍} + \frac{🍌}{🍍+🍎} + \frac{🍍}{🍎+🍌} = 4$$

Can you find positive whole values for 🍎, 🍌, and 🍍?

🍎=154476802108746166441951315019919837485664325669565431700026634898253202035277999

🍌=36875131794129999827197811565225474825492979968971970996283137471637224634055579

🍍=4373612677928697257861252602371390152816537558161613618621437993378423467772036

Simple Puzzle

Bremner and MacLeod looked at what happens when we replace the 4 with something else. When you try to represent 178 in this way, you'll need 398,605,460 digits. If you try 896, you'll be up to *trillions* of digits…

~~95%~~ of people cannot solve this!

99.9999%

$$\frac{🍎}{🍌+🍍} + \frac{🍌}{🍍+🍎} + \frac{🍍}{🍎+🍌} = 4$$

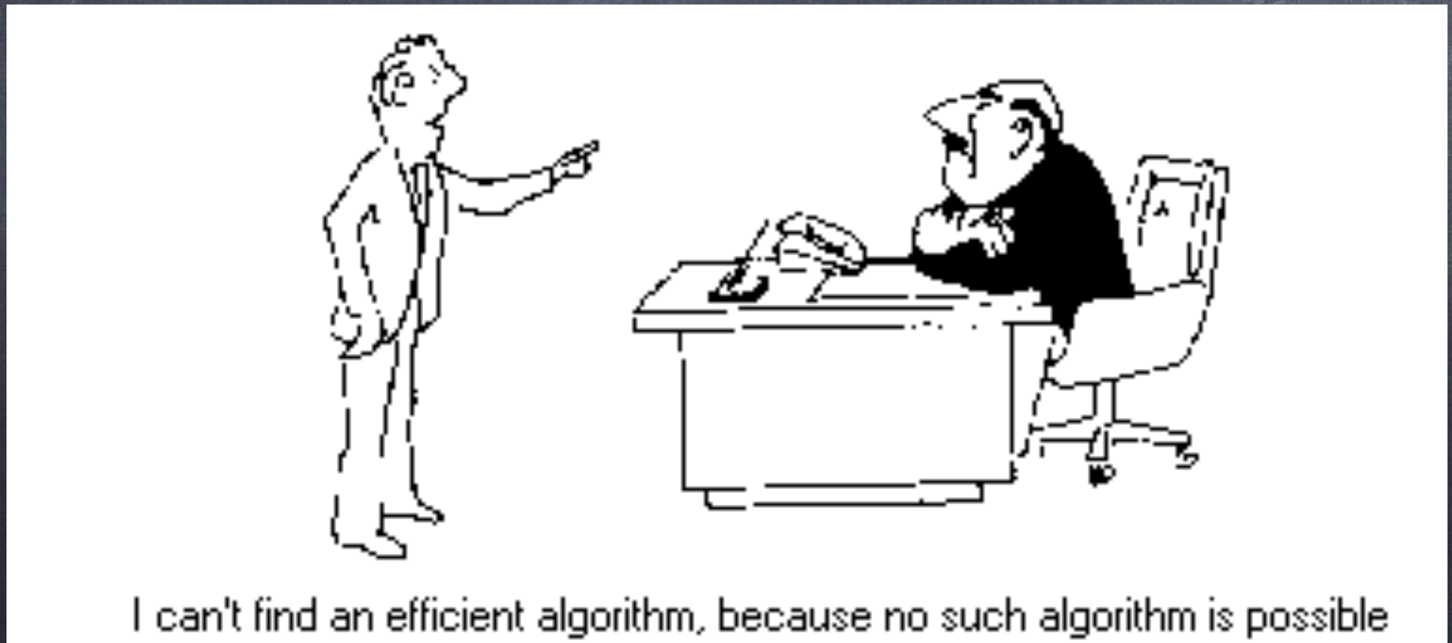
Can you find positive whole values for 🍎, 🍌, and 🍍?

🍎=154476802108746166441951315019919837485664325669565431700026634898253202035277999

🍌=36875131794129999827197811565225474825492979968971970996283137471637224634055579

🍍=4373612677928697257861252602371390152816537558161613618621437993378423467772036

Computer Science version of these issues

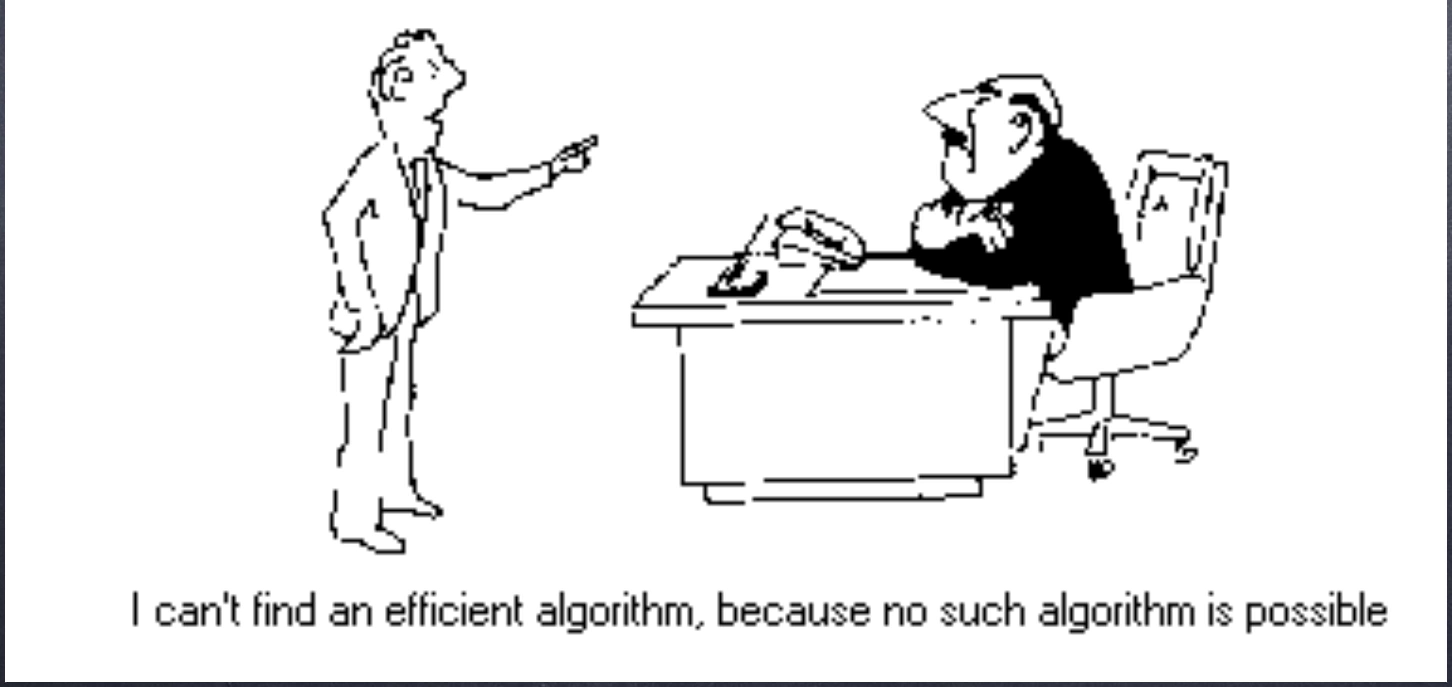


I can't find an efficient algorithm, because no such algorithm is possible

Computer Science version of these issues

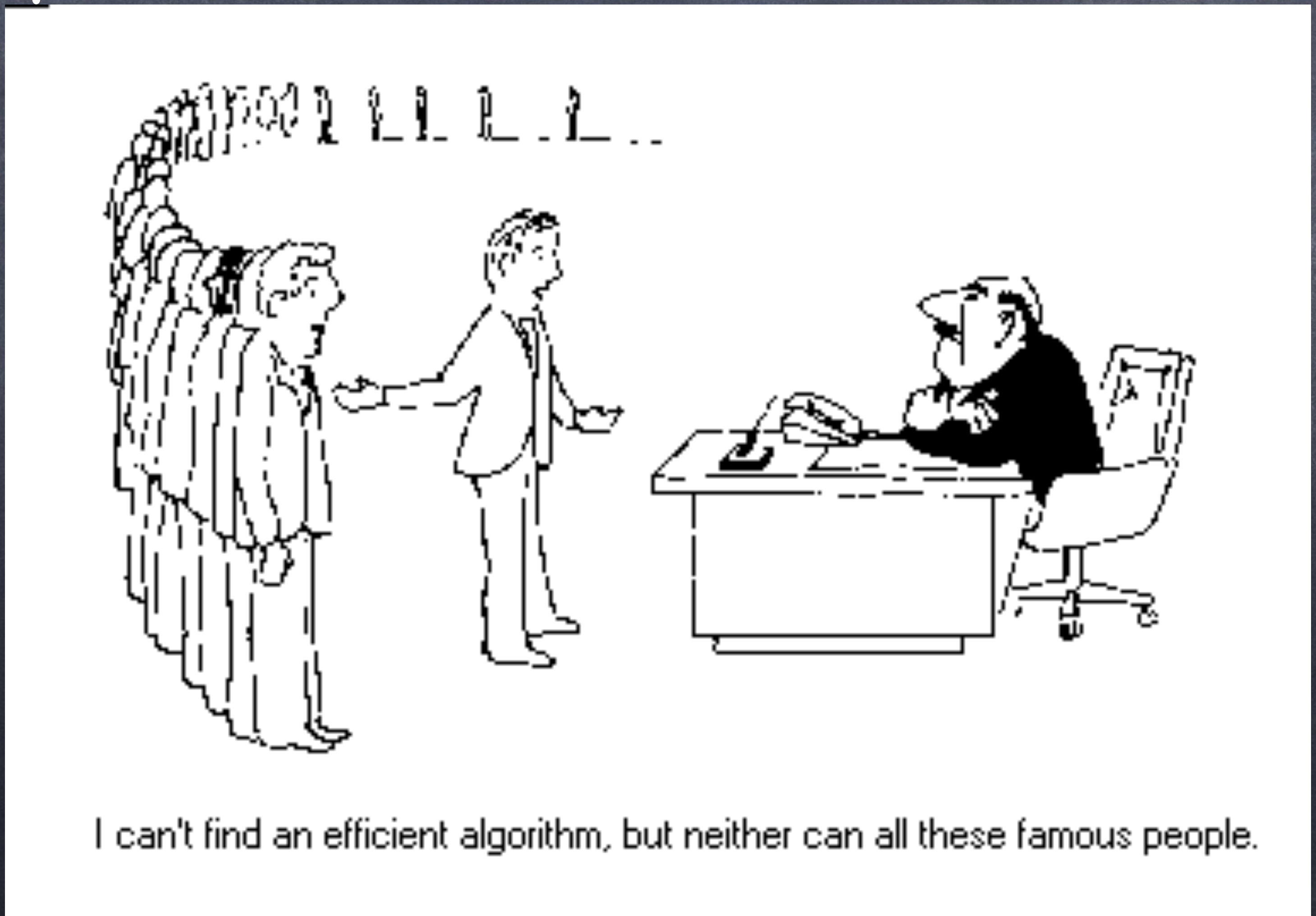
- Are there some problems that cannot be solved at all ? and, are there problems that cannot be solved efficiently ??

  (related to Hilbert's 10th problem)



Computer Science version of these issues

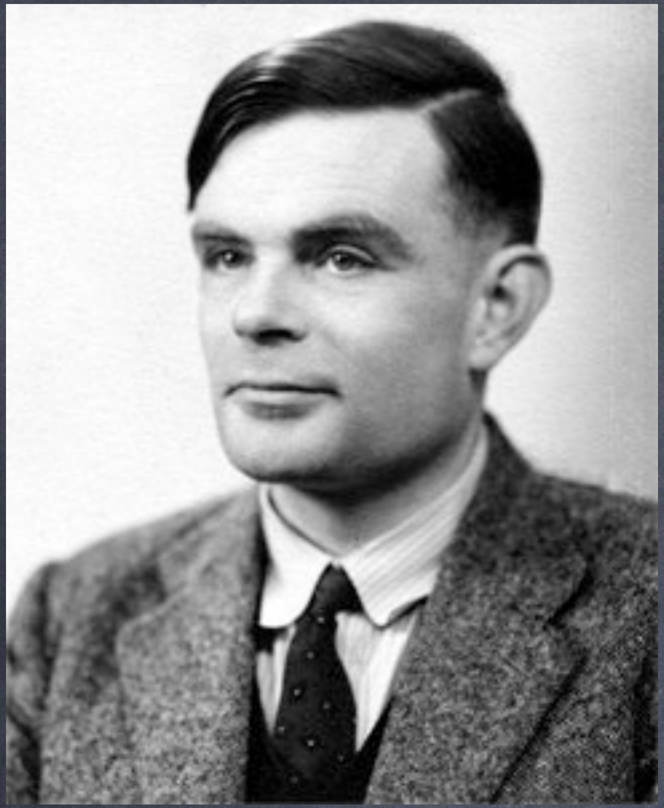
- If my boss / supervisor / teacher formulates a problem to be solved urgently, can I write a program to efficiently solve this problem ???



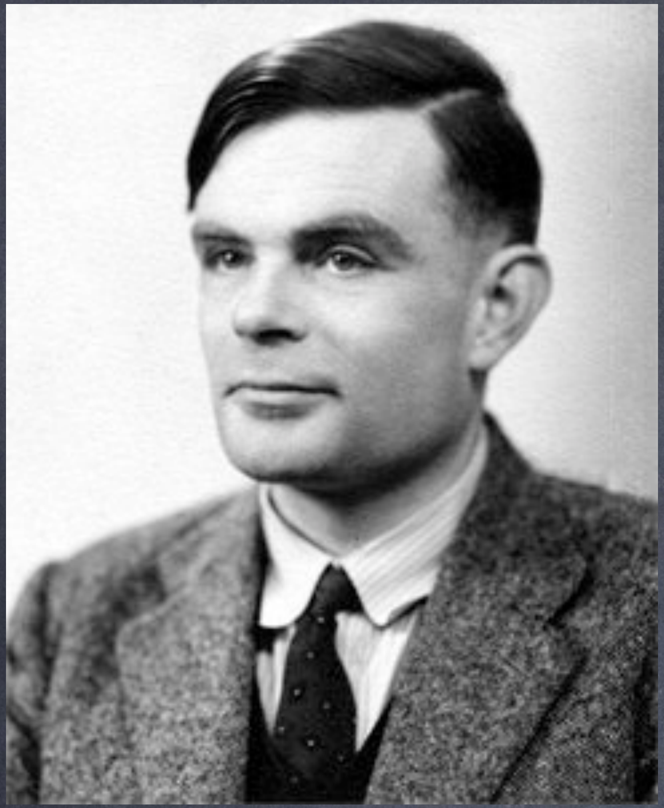
Kurt Gödel

Kurt Gödel

- In 1931, he proved that any formalization of mathematics contains some statements that cannot be proved or disproved.



Alan Turing



Alan Turing

- In 1934, he formalized the notion of **decidability** of a language by a computer.

A Language

A Language

- Let Σ be a finite alphabet. (ex: {0,1})

A Language

- Let Σ be a finite alphabet. (ex: {0,1})
- Let Σ^* be all sequences of elements from this alphabet. (ex: 0, 1, 00000, 0101010101,...)

A Language

- Let Σ be a finite alphabet. (ex: $\{0,1\}$)
- Let Σ^* be all sequences of elements from this alphabet. (ex: 0, 1, 00000, 0101010101,...)
- A language L is any subset of Σ^*.

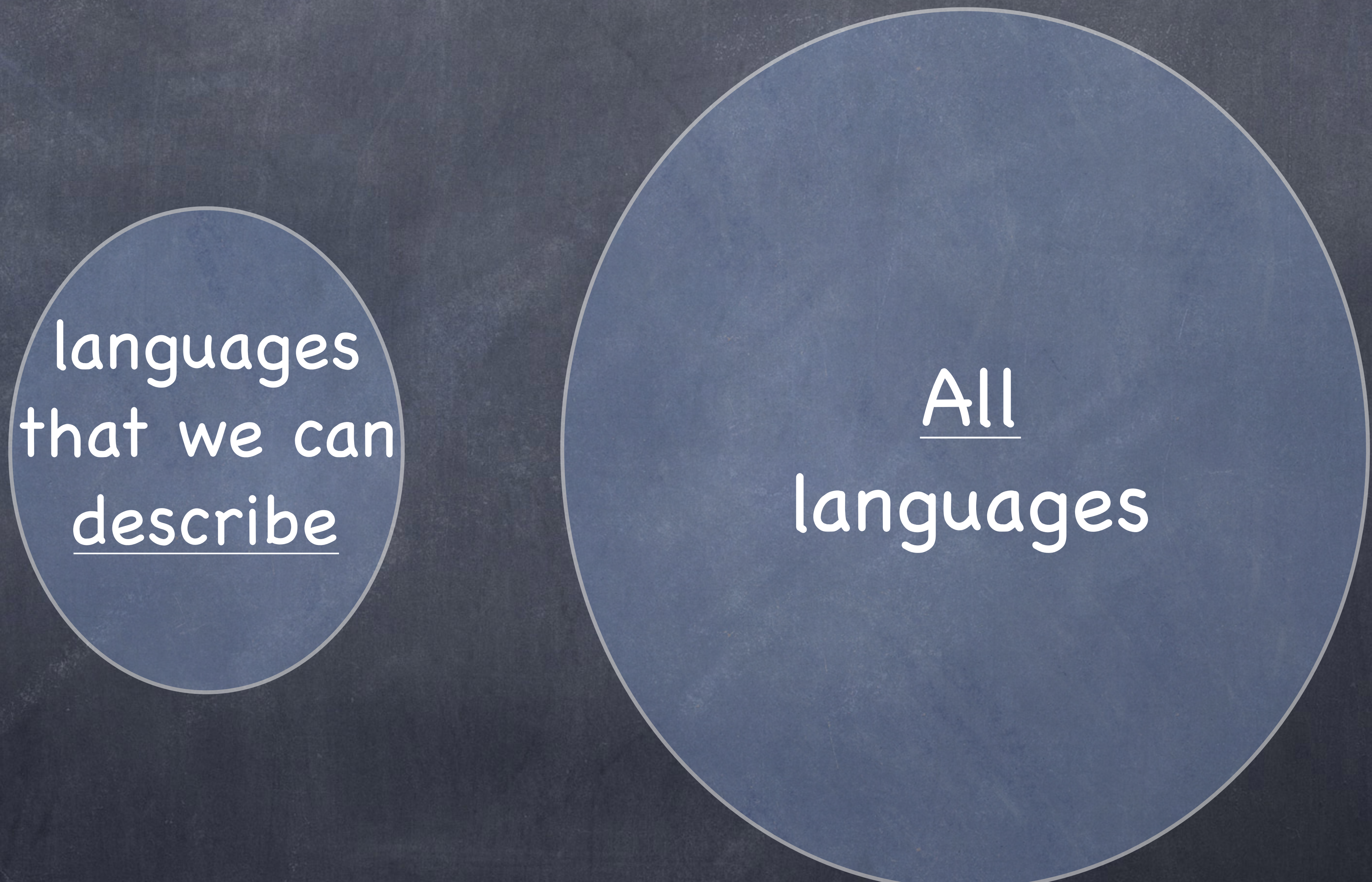
A Language

- Let Σ be a finite alphabet. (ex: {0,1})
- Let Σ* be all sequences of elements from this alphabet. (ex: 0, 1, 00000, 0101010101,...)
- A language L is any subset of Σ*.
- An algorithm **A** <u>decides</u> a language L if **A** answers **Yes** when $x \in L$ and **No** when $x \notin L$.

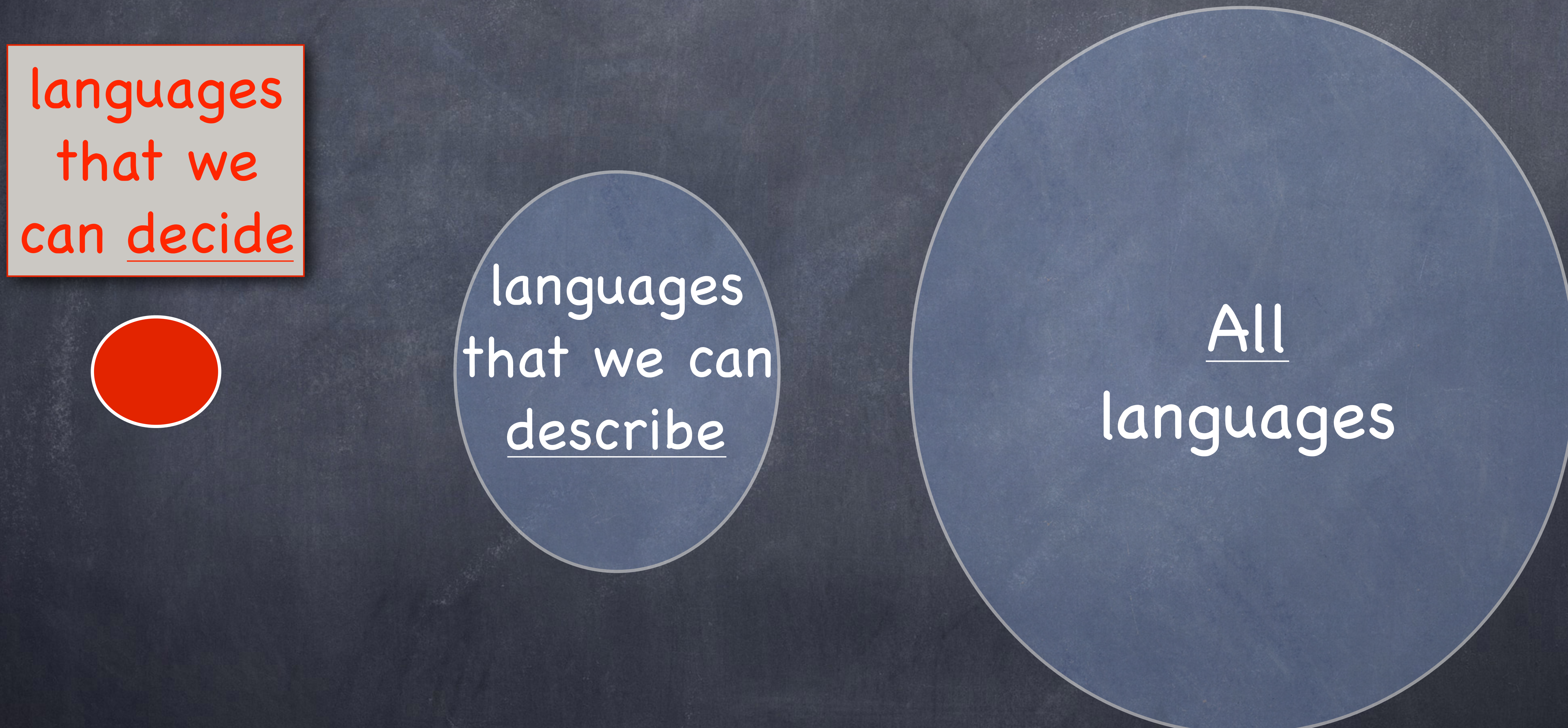
Comparing Cardinalities

Comparing Cardinalities

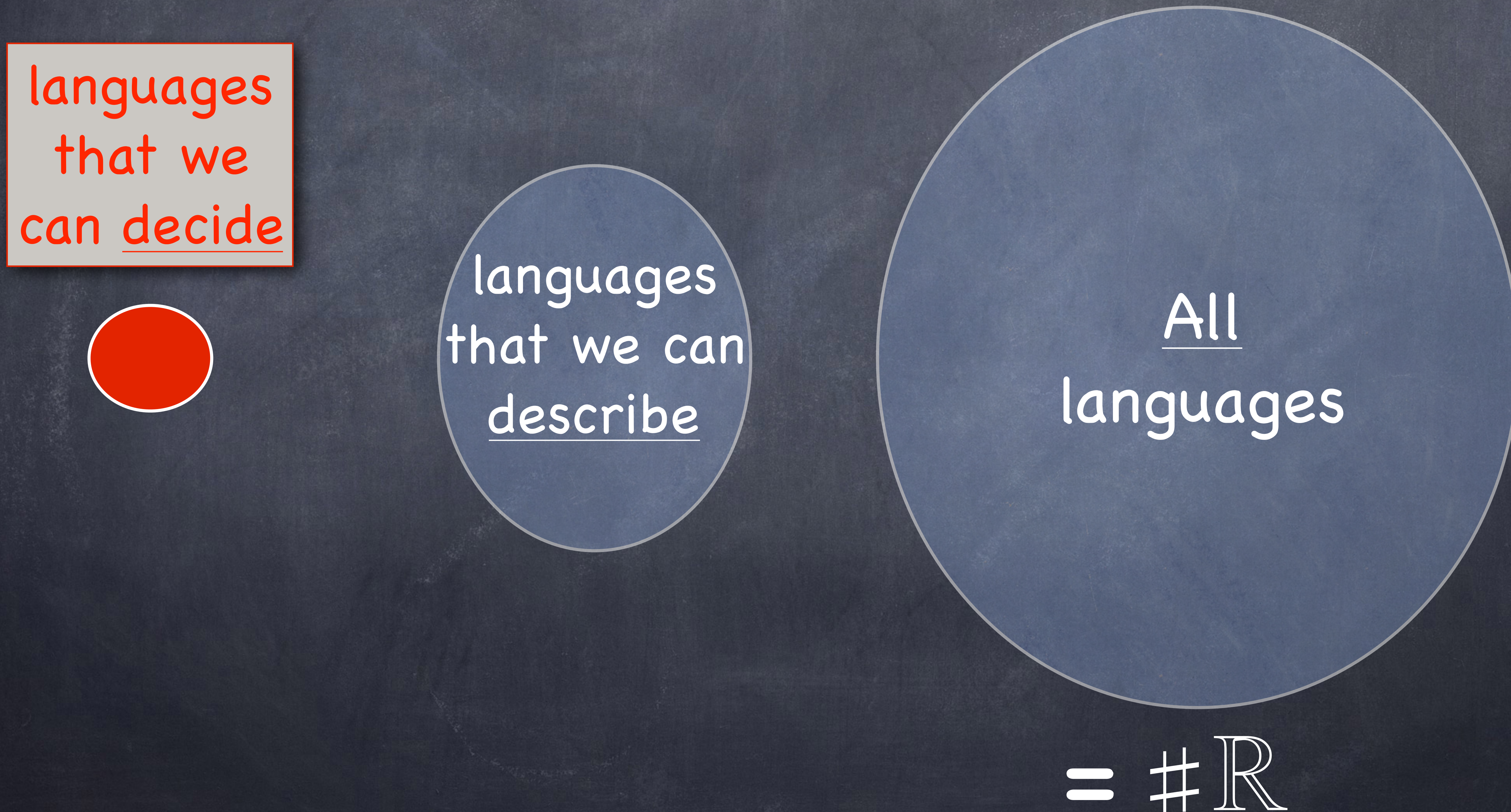
Comparing Cardinalities



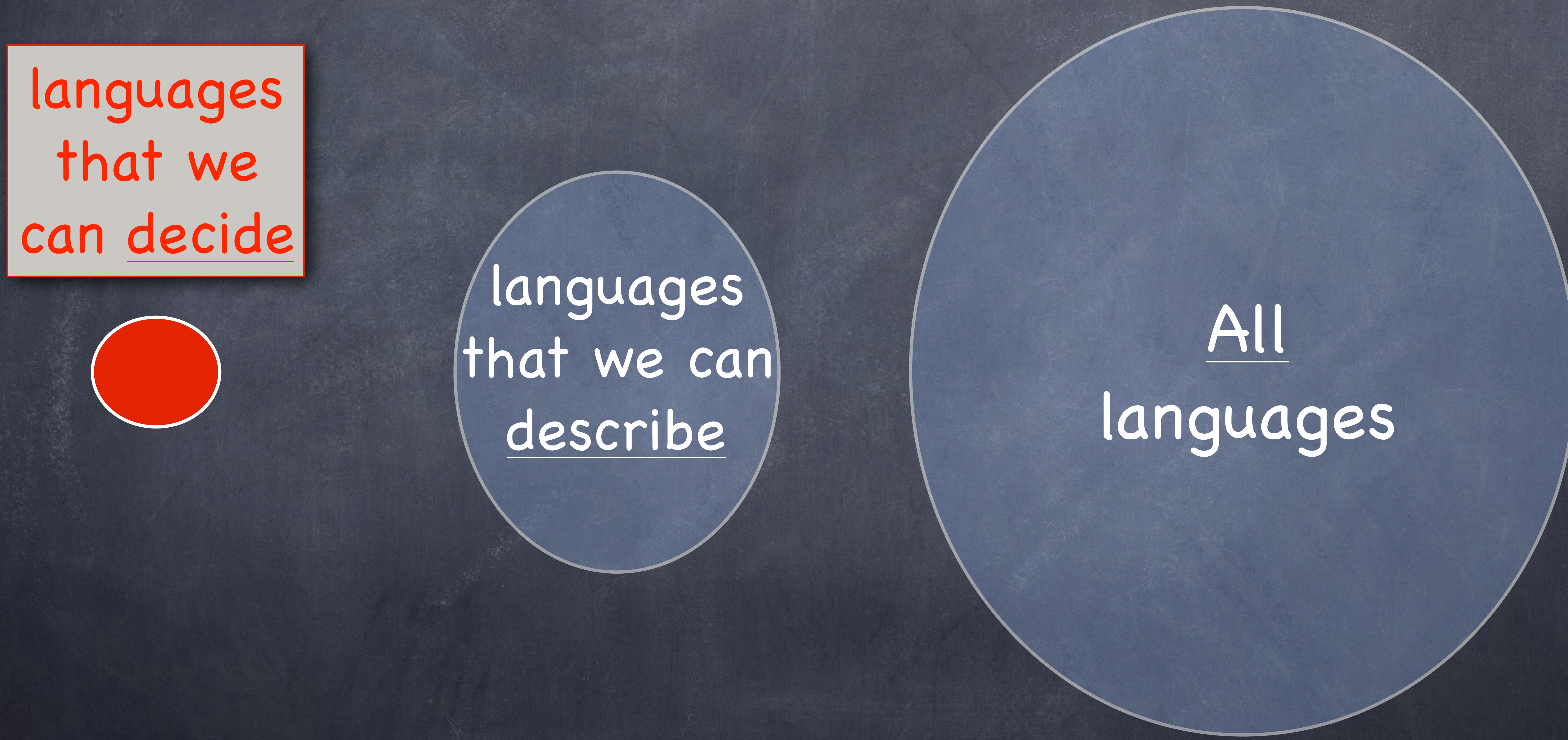
Comparing Cardinalities



Comparing Cardinalities

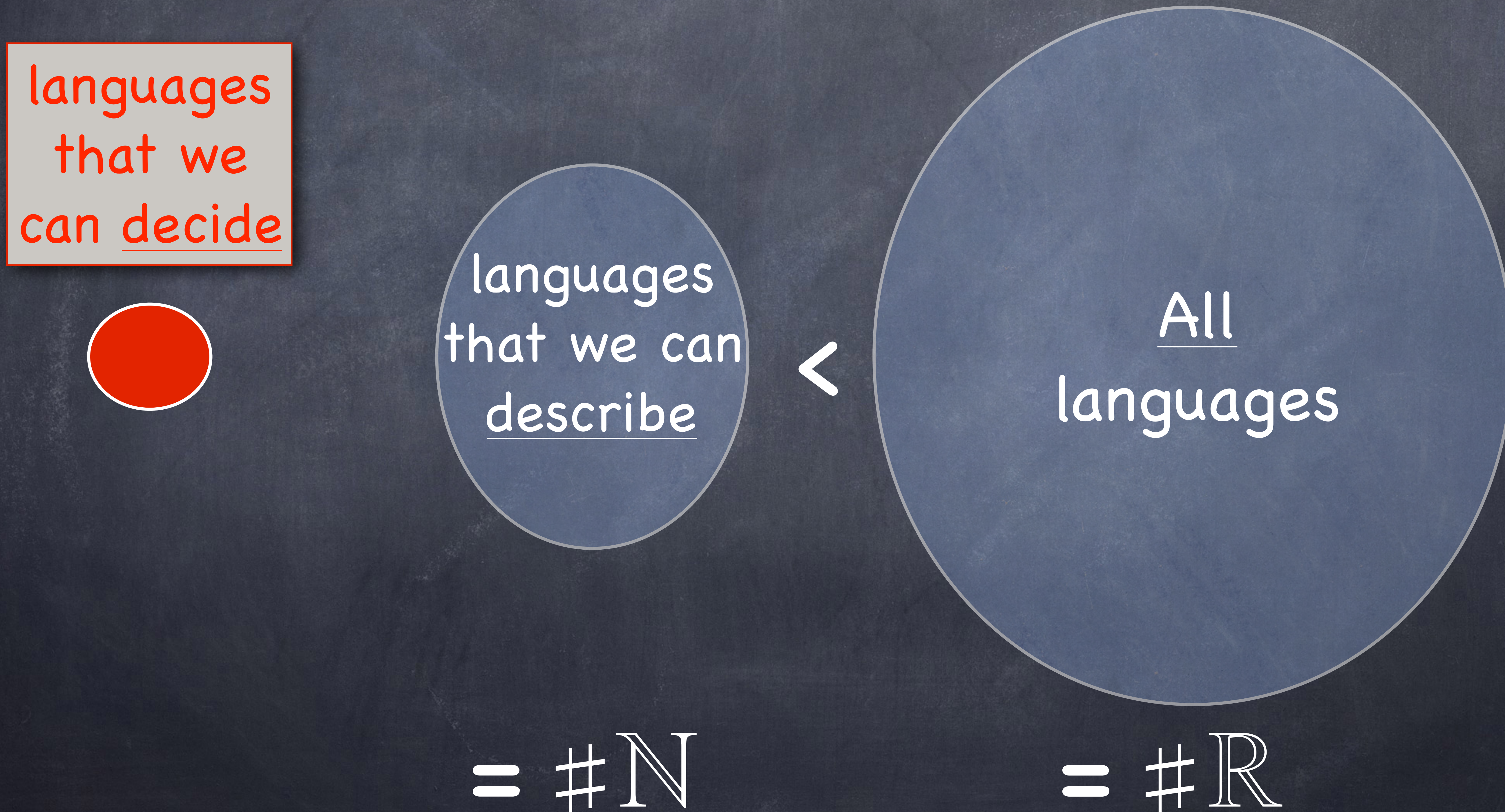


Comparing Cardinalities

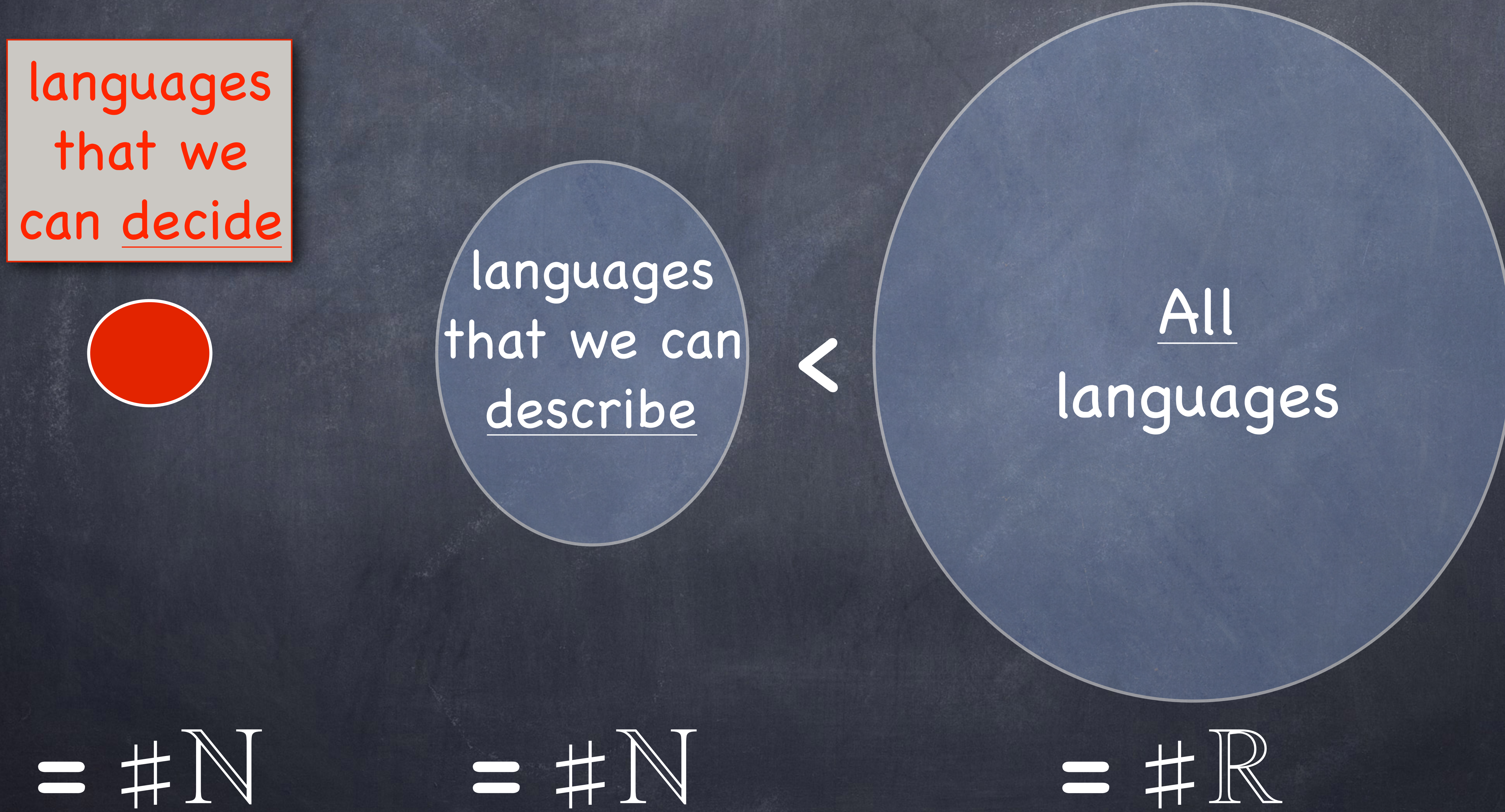


$= \#\mathbb{N}$ $= \#\mathbb{R}$

Comparing Cardinalities



Comparing Cardinalities



Comparing Cardinalities

languages that we can decide

= languages that we can describe < All languages

$= \#\mathbb{N}$ $\quad = \#\mathbb{N}$ $\quad = \#\mathbb{R}$

Alonzo Church

Alonzo Church

- In 1936, he proved that certain **languages** cannot be **decided** by any algorithm whatsoever… (even some language that we can describe precisely)

Emil Post

Emil Post

- In 1946, he gave a very natural example of an **undecidable** language...

(**PCP**) Post Correspondence Problem

(**PCP**) Post Correspondence Problem

aaa	a	bbb	aa		b
bb	bb	a	a	bb	

- An instance of **PCP** with 6 tiles.

(**PCP**) Post Correspondence Problem

aaa	a	bbb	aa		b
bb	bb	a	a	bb	

- An instance of **PCP** with 6 tiles.
- A solution to **PCP**

(**PCP**) Post Correspondence Problem

aaa	a	bbb	aa		b
bb	bb	a	a	bb	

- An instance of **PCP** with 6 tiles.

- A solution to **PCP**

aa
a

(**PCP**) Post Correspondence Problem

aaa	a	bbb	aa		b
bb	bb	a	a	bb	

- An instance of **PCP** with 6 tiles.
- A solution to **PCP**

aa	bbb
a	a

(**PCP**) Post Correspondence Problem

aaa	a	bbb	aa		b
bb	bb	a	a	bb	

- An instance of **PCP** with 6 tiles.
- A solution to **PCP**

aa	bbb	b
a	a	

(**PCP**) Post Correspondence Problem

aaa	a	bbb	aa		b
bb	bb	a	a	bb	

- An instance of **PCP** with 6 tiles.

- A solution to **PCP**

aa	bbb	b	
a	a		bb

(**PCP**) Post Correspondence Problem

aaa	a	bbb	aa		b
bb	bb	a	a	bb	

- An instance of **PCP** with 6 tiles.

- A solution to **PCP**

aa	bbb	b		
a	a		bb	bb

(**PCP**) Post Correspondence Problem

aaa	a	bbb	aa		b
bb	bb	a	a	bb	

- An instance of **PCP** with 6 tiles.
- A solution to **PCP**

aa	bbb	b		
a	a		bb	bb

Post
Correspondence Problem

Post Correspondence Problem

u_1	u_2	u_3	. . .	u_n
v_1	v_2	v_3		v_n

- Given n tiles, u_1/v_1 … u_n/v_n
  where each u_i or v_i is a sequence of letters.

Post Correspondence Problem

u_1	u_2	u_3	. . .	u_n
v_1	v_2	v_3		v_n

- Given n tiles, u_1/v_1 … u_n/v_n
  where each u_i or v_i is a sequence of letters.

- Is there a k and a sequence $<i_1,i_2,i_3,\ldots,i_k>$
  (with each $1 \leq i_j \leq n$) such that

$$u_{i_1} | u_{i_2} | u_{i_3} | \ldots | u_{i_k} = v_{i_1} | v_{i_2} | v_{i_3} | \ldots | v_{i_k} ?$$

A Solution to Post Correspondence Problem

u_1	u_2	u_3	. . .	u_n
v_1	v_2	v_3		v_n

u_{i_1}	u_{i_2}	u_{i_3}	u_{i_4}	u_{i_5}	. . .	u_{i_k}
v_{i_1}	v_{i_2}	v_{i_3}	v_{i_4}	v_{i_5}		v_{i_k}

A Solution to Post Correspondence Problem

u_1	u_2	u_3	. . .	u_n
v_1	v_2	v_3		v_n

- A solution is of this form:

u_{i_1}	u_{i_2}	u_{i_3}	u_{i_4}	u_{i_5}	. . .	u_{i_k}
v_{i_1}	v_{i_2}	v_{i_3}	v_{i_4}	v_{i_5}		v_{i_k}

A Solution to Post Correspondence Problem

$$\boxed{\begin{array}{c} u_1 \\ v_1 \end{array}} \; \boxed{\begin{array}{c} u_2 \\ v_2 \end{array}} \; \boxed{\begin{array}{c} u_3 \\ v_3 \end{array}} \quad \cdots \quad \boxed{\begin{array}{c} u_n \\ v_n \end{array}}$$

- A solution is of this form:

$$\boxed{\begin{array}{c} u_{i_1} \\ v_{i_1} \end{array}} \; \boxed{\begin{array}{c} u_{i_2} \\ v_{i_2} \end{array}} \; \boxed{\begin{array}{c} u_{i_3} \\ v_{i_3} \end{array}} \; \boxed{\begin{array}{c} u_{i_4} \\ v_{i_4} \end{array}} \; \boxed{\begin{array}{c} u_{i_5} \\ v_{i_5} \end{array}} \quad \cdots \quad \boxed{\begin{array}{c} u_{i_k} \\ v_{i_k} \end{array}}$$

with the top and bottom strings identical when we concatenate all the substrings.

A Solution to Post Correspondence Problem

$\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} \begin{bmatrix} u_2 \\ v_2 \end{bmatrix} \begin{bmatrix} u_3 \\ v_3 \end{bmatrix} \quad \cdots \quad \begin{bmatrix} u_n \\ v_n \end{bmatrix}$

- A solution is of this form:

$\begin{bmatrix} u_{i_1} \\ v_{i_1} \end{bmatrix} \begin{bmatrix} u_{i_2} \\ v_{i_2} \end{bmatrix} \begin{bmatrix} u_{i_3} \\ v_{i_3} \end{bmatrix} \begin{bmatrix} u_{i_4} \\ v_{i_4} \end{bmatrix} \begin{bmatrix} u_{i_5} \\ v_{i_5} \end{bmatrix} \quad \cdots \quad \begin{bmatrix} u_{i_k} \\ v_{i_k} \end{bmatrix}$

with the top and bottom strings identical when we concatenate all the substrings.

COMP-330
Theory of Computation

Fall 2019 -- Prof. Claude Crépeau

Lecture 1 :
Introduction