Unit 7: Theory

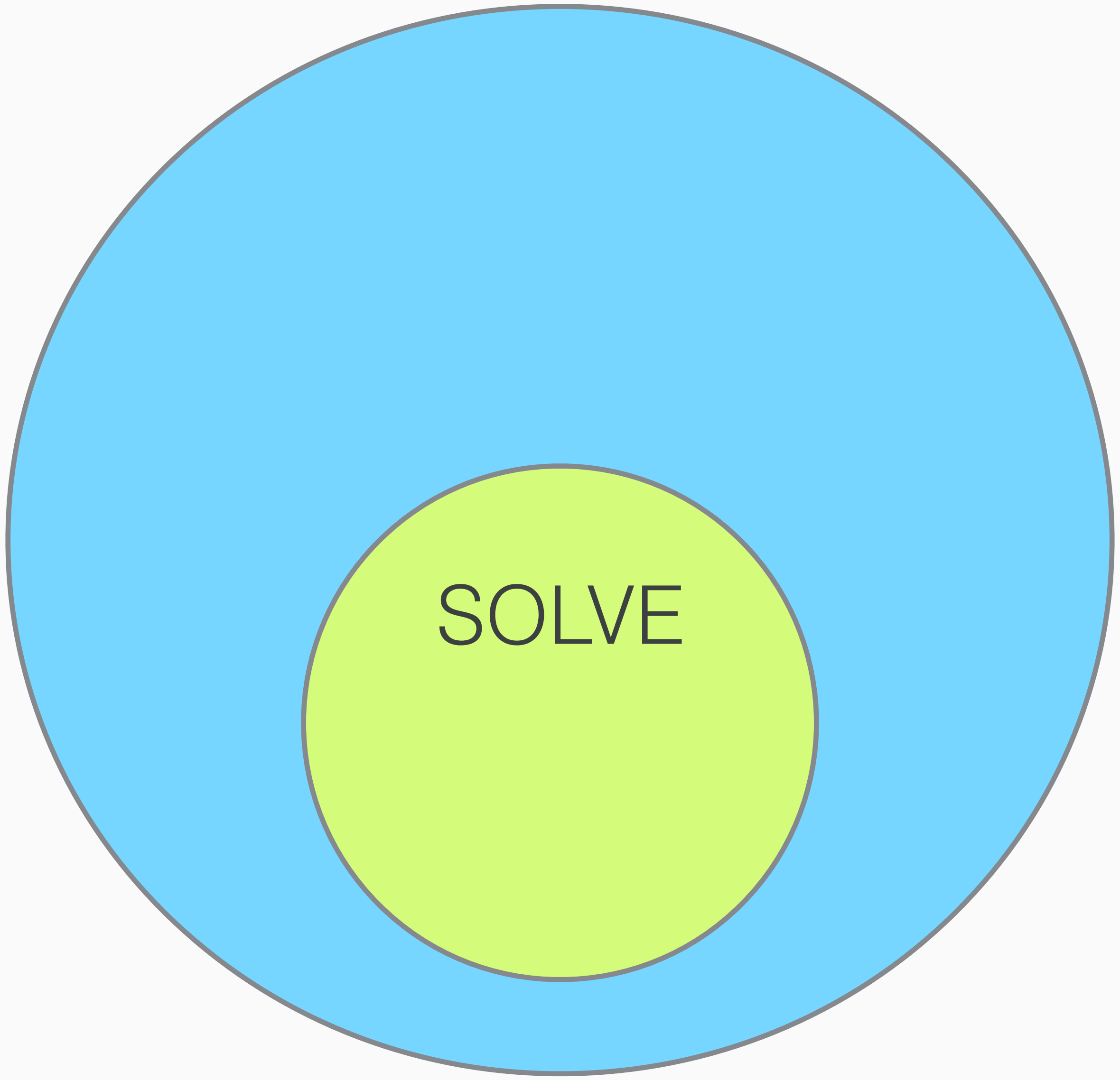
Dave Abel

March 18th, 2016

As Promised:

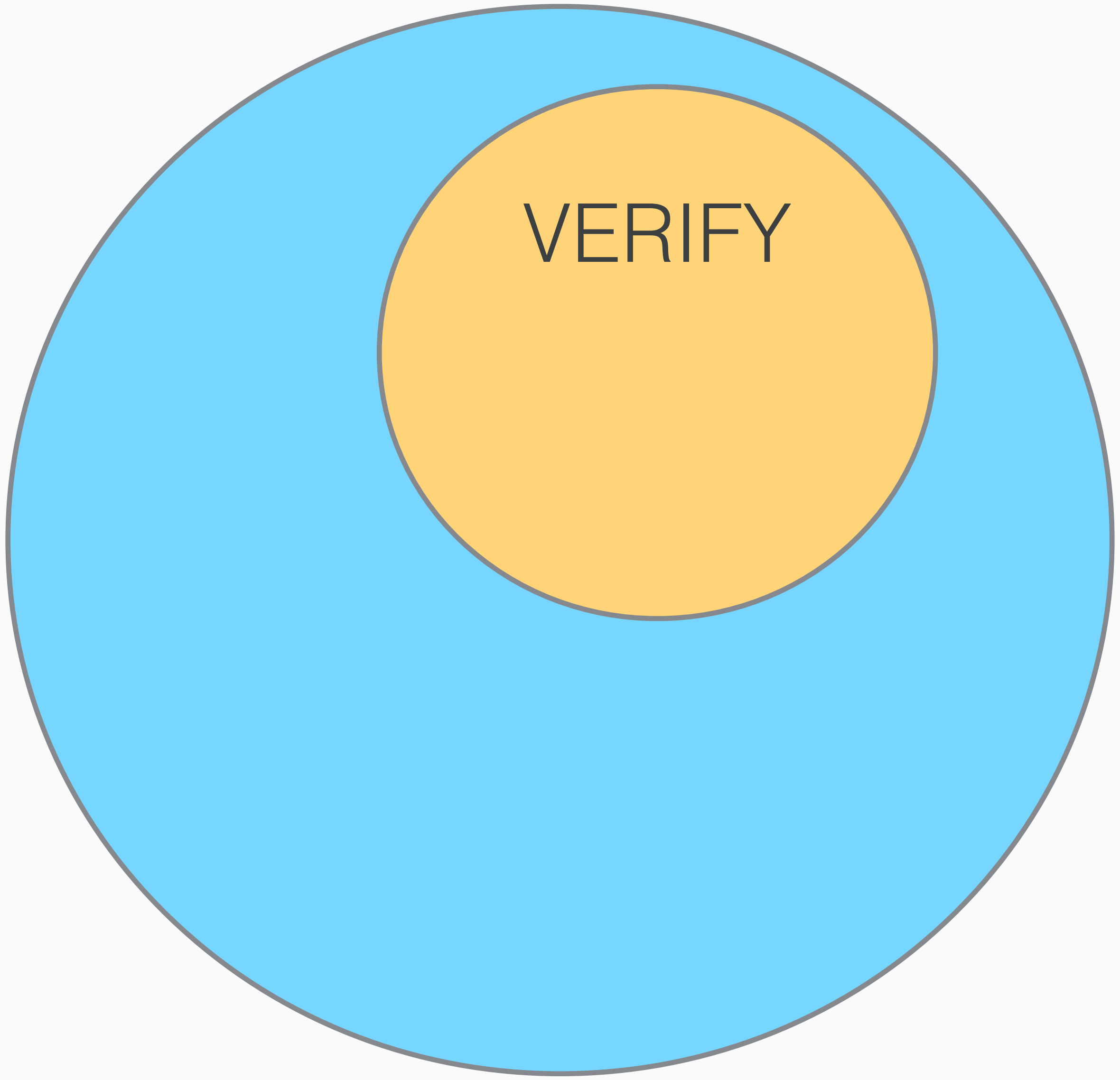
Any Midterm questions?

SOLVE



Problems that we can solve efficiently

VERIFY



Problems for which candidate solutions can be verified efficiently

This Is It!

Q: Given *any* problem known to be in VERIFY, do we know anything about its status relative to SOLVE?

A: Nope! (so far)

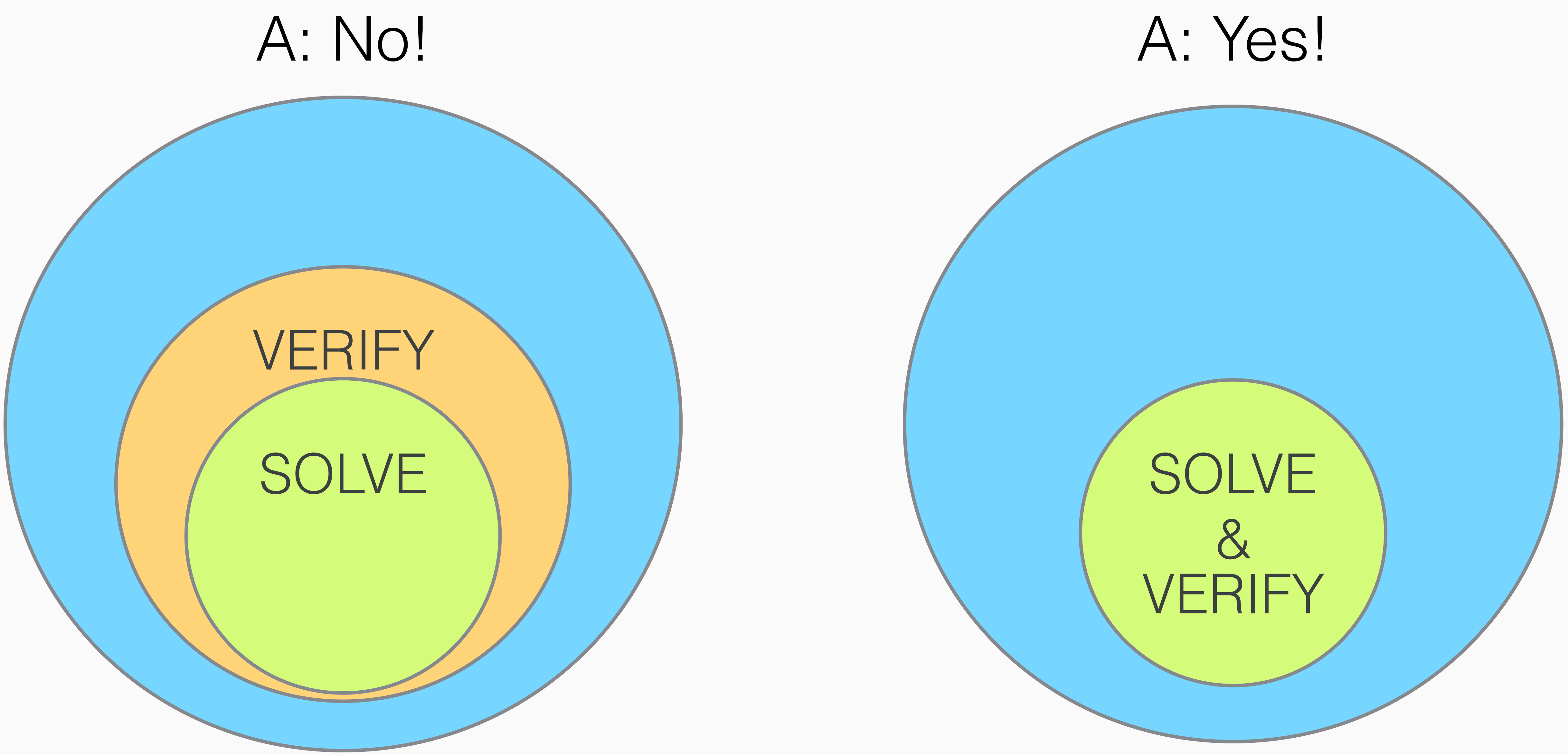
This is considered the most important unanswered question in all of computer science.

Rephrased

Q: If candidate solutions to a problem can be verified efficiently, can the problem also be solved efficiently?

This is considered the most important unanswered question in all of computer science.

SOLVE versus VERIFY

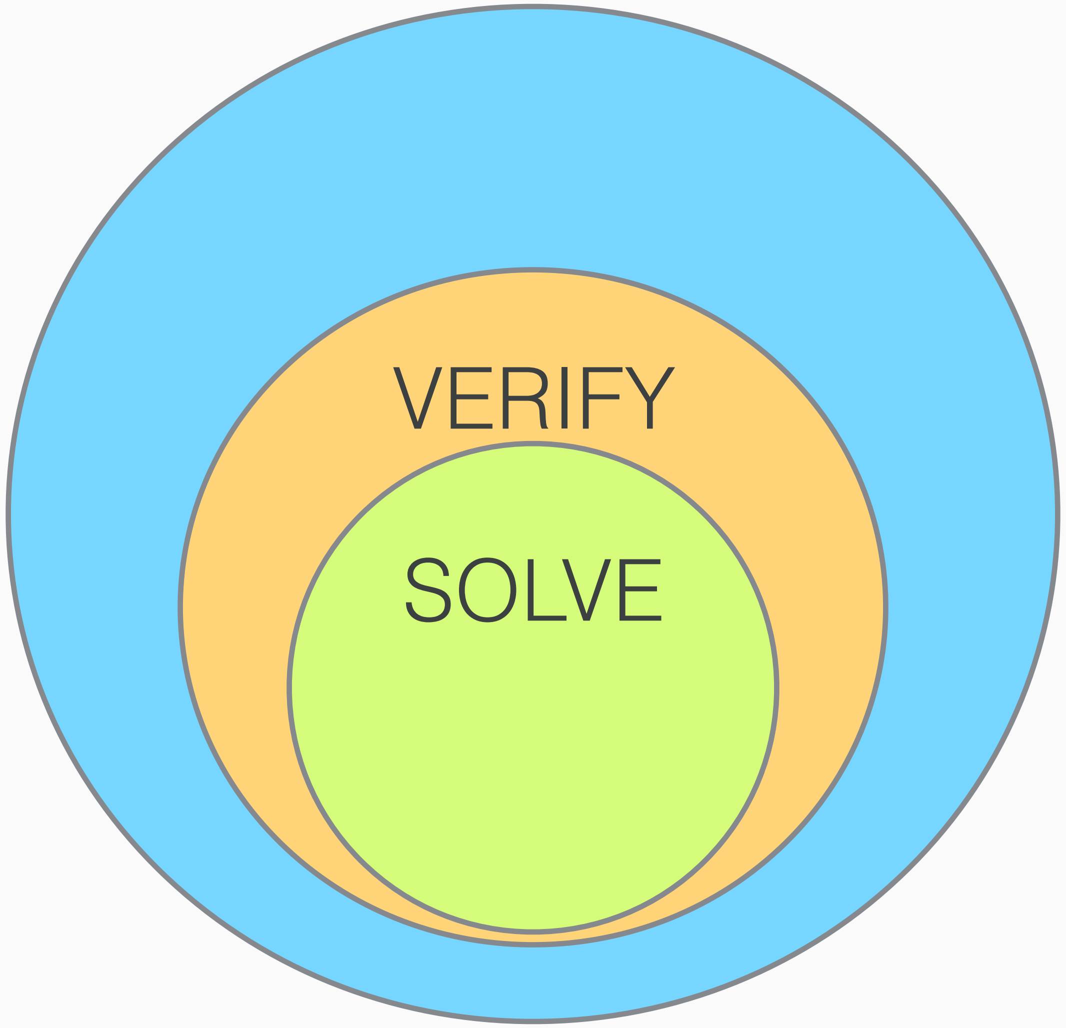


Q: If candidate solutions to a problem can be verified efficiently, can the problem also be solved efficiently?

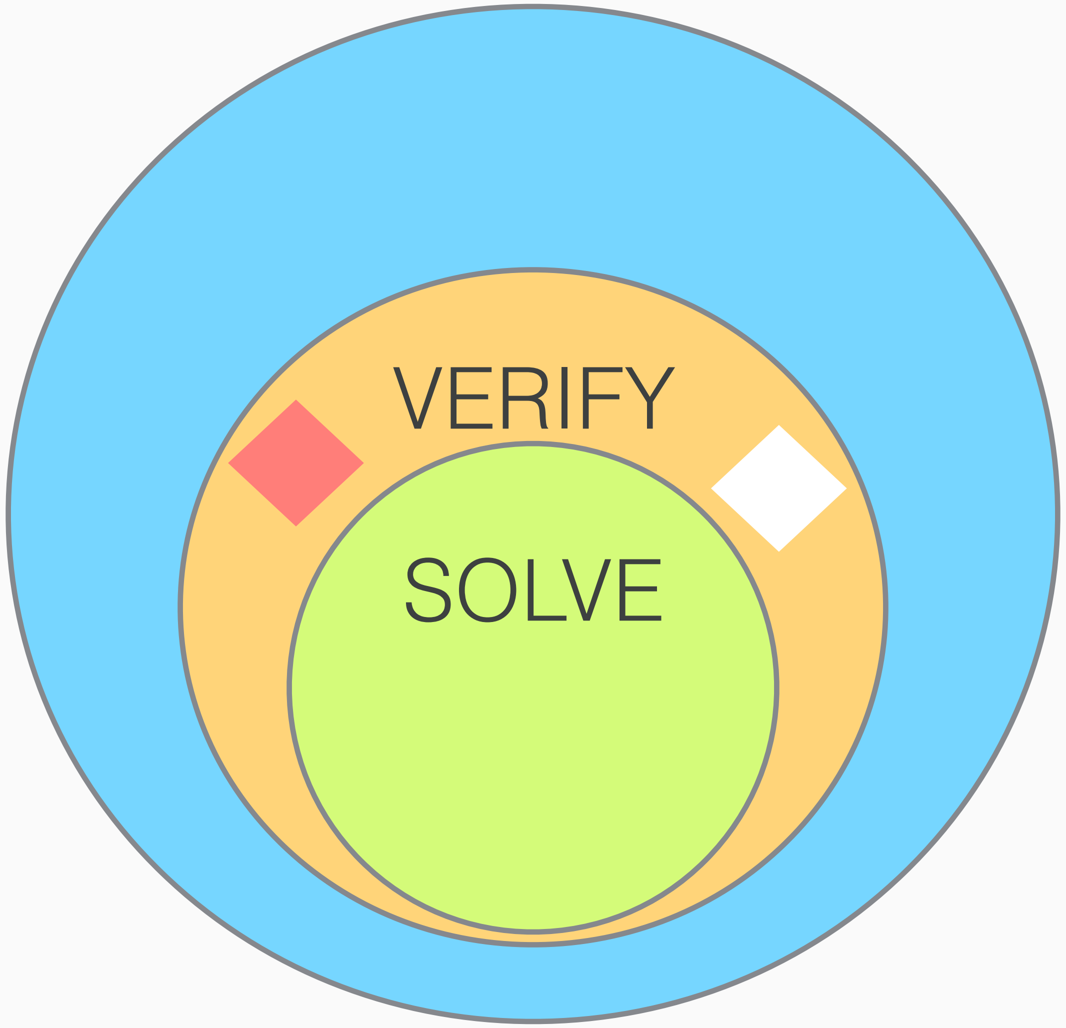
‘The P vs. NP problem has been called “one of the most important problems in contemporary mathematics and theoretical computer science” [60]. That is an understatement. Not only is P vs. NP the defining question of our field; it’s one of the deepest questions ever asked for which we’d know how to recognize an answer.’

-Prof. Scott Aaronson

Some Verification Examples



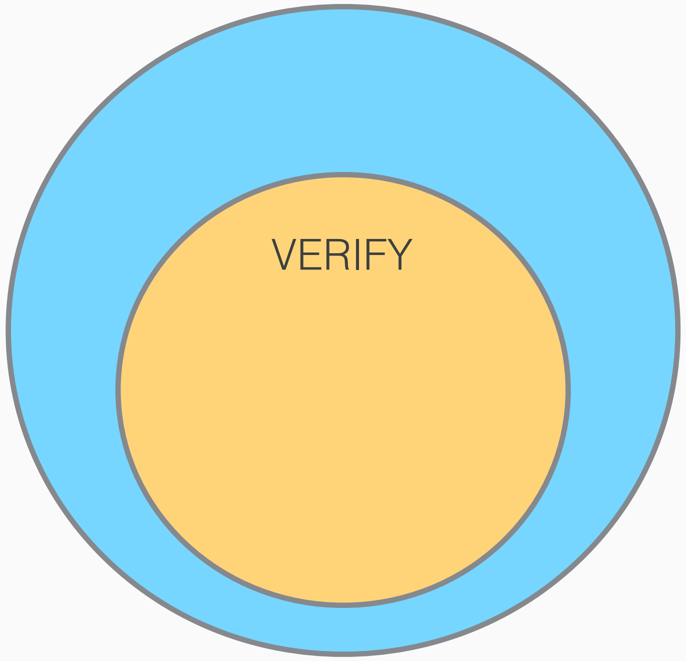
Some Verification Examples



(That we think might not be in SOLVE)

Some Verification Examples

- *Can this logical formula be true?*
  - (Also called Satisfiability)
- Subset Sum
- Circle + Line problems!



Can this logical formula be true?

- *Input:*
  - a logical formula
- *Output:*
  - True if there is some way we can make the formula true
  - False if there is no way we can make the formula true

Can this logical formula be true?

I give you: *AND(P,Q)*

Q: Can this be true?

(our tools are how we set the truth value of *P, Q*)

Can this logical formula be true?

I give you: *AND(P,Q)*

Q: Can this be true?

A: Sure! *P = True, Q = True*

(our tools are how we set the truth value of *P, Q*)

Verification: Can this logical formula be true?

Verification: Can this logical formula be true?

- *Input:*
  - a logical formula, an assignment to each variable
- *Output:*
  - True if the resulting sentence is True
  - False if the resulting sentence is False

Verification: Can this logical formula be true?

I give you: *AND(P,Q), P = True, Q = False*

Q: Is this True?

Verification: Can this logical formula be true?

I give you: *AND(P,Q), P = True, Q = False*

Q: Is this True?

No!

Can this logical formula be true?

Build The Truth Table! (Brute Force)

1. *Build the truth table for the logical formula*
2. *Check to see if it has any row that is True.*
3. *If it does, report True.*
4. *If not, report False.*

Build the Table!

Build the Table!

1. Build the Truth Table
2. Check to see if any row is True.
3. If so, report True.
4. Otherwise, report False.

Q: Growth Rate?

(*N* is number of variables..)

Build the Table!

Build the Table!

1. Build the Truth Table
2. Check to see if any row is True.
3. If so, report True.
4. Otherwise, report False.

Q: Growth Rate?

(*N* is number of variables..)

A: 2^N

P	*Q*	*R*	
T	T	T	
T	T	F	
T	F	T	
T	F	F	
F	T	T	
F	T	F	
F	F	T	
F	F	F	

Verification: Can this logical formula be true?

- *Input:*
  - a logical formula, an assignment to each variable
- *Output:*
  - True if the resulting sentence is True
  - False if the resulting sentence is False

Algorithm: plug in the assignments, check if True!

Verification: Can this logical formula be true?

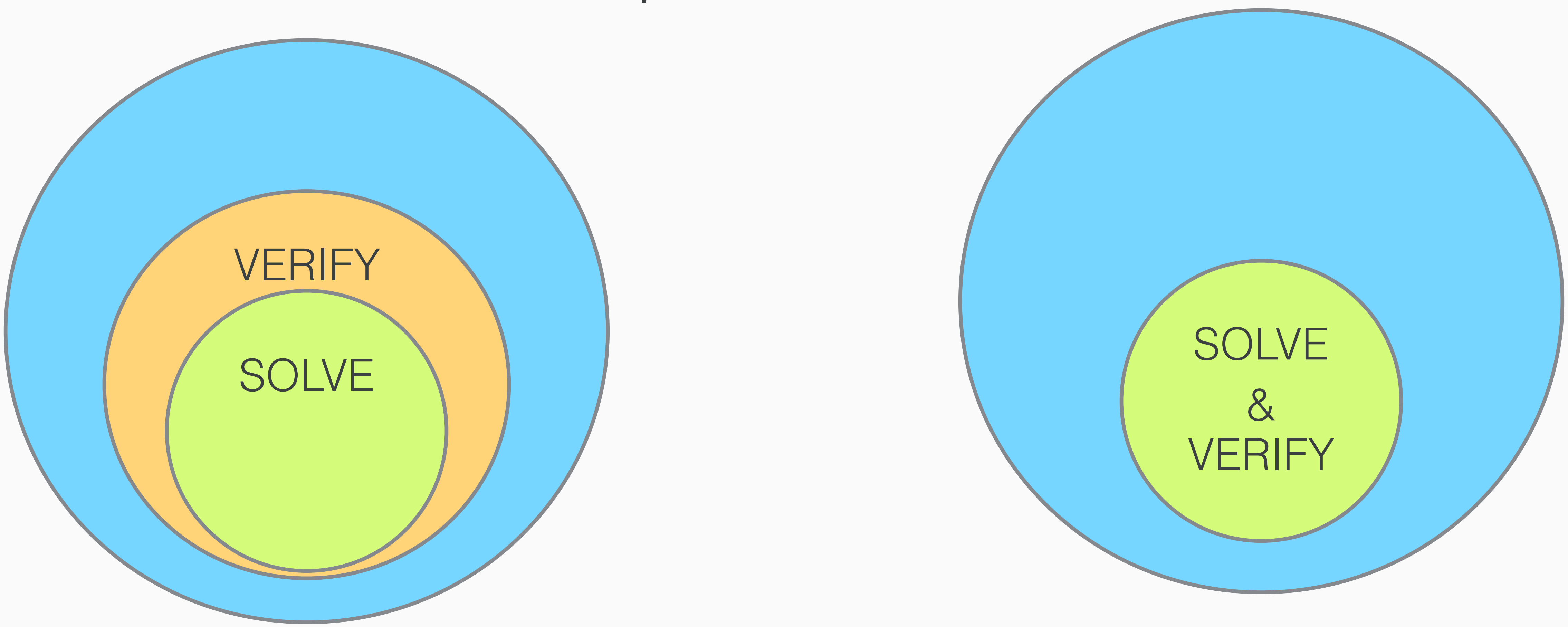
- *Input:*
  - a logical formula, an assignment to each variable
- *Output:*
  - True if the resulting sentence is True
  - False if the resulting sentence is False

Growth Rate: N

Algorithm: plug in the assignments, check if True!

Satisfiability (SAT)

- *Can **verify** with N operations*
- *Can **solve** with 2^N operations.*



A: No

A: Yes

Problem: Subset Sum

- *Input:*
  - Some numbers! Like: 5,2,14,6,12,1,3
- *Output:*
  - True if some of the given numbers add to 10
  - False if none of the numbers add to 10

Clicker Question!

- *Input:*
  - Some numbers! Like: 5,2,14,6,12,1,3
- *Output:*
  - True if some of the given numbers add to 10
  - False if none of the numbers add to 10

[A] TRUE

[B] FALSE

Q: What do we report for 5,2,14,6,12,1,3?

Clicker Answer!

- *Input:*
  - Some numbers! Like: 5,2,14,6,12,1,3
- *Output:*
  - True if some of the given numbers add to 10
  - False if none of the numbers add to 10

[A] TRUE

[B] FALSE

Q: What do we report for 5,2,14,6,12,1,3?

Verification: Subset Sum

- *Input:*
  - Some numbers! Like: 5,2,14,6,12,1,3
  - A proposed solution (e.g. 5,1,3)
- *Output:*
  - True if the proposed solution adds to 10
  - False if the proposed solution does not add to 10

Verification: Subset Sum

- *Input:*
  - Some numbers: 5,2,14,6,12,1,3
  - Proposed solution: 2,6,4
- *Output:*
  - True if 2,6,4 adds to 10
  - False if 2,6,4 adds to10

Verification: Subset Sum

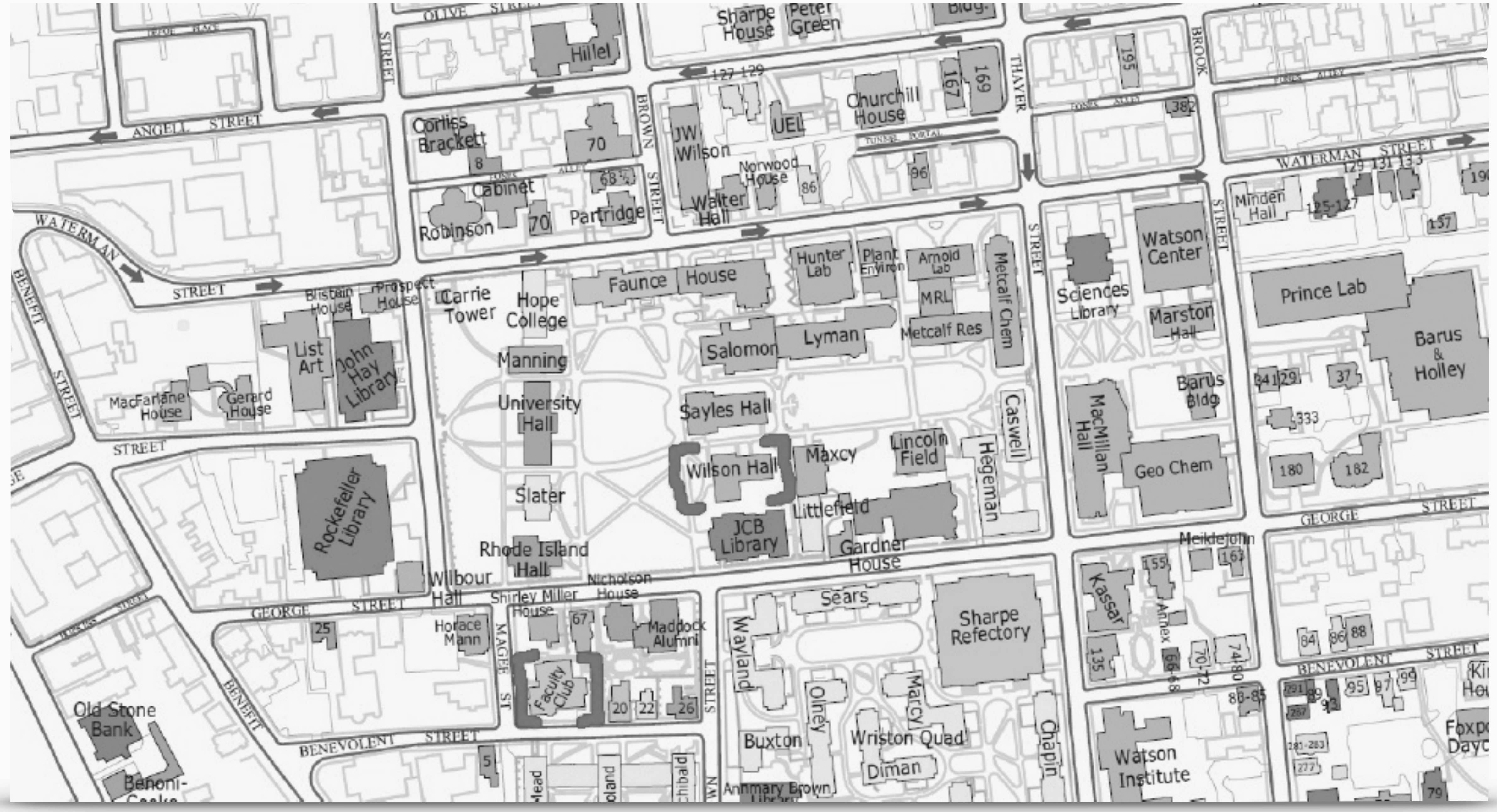
- *Input:*
  - Some numbers: 5,2,14,6,12,1,3
  - Proposed solution: 2,6,4
- *Output:*
  - True if 2,6,4 adds to 10
  - **False** if 2,6,4 adds to10

Verification: Subset Sum

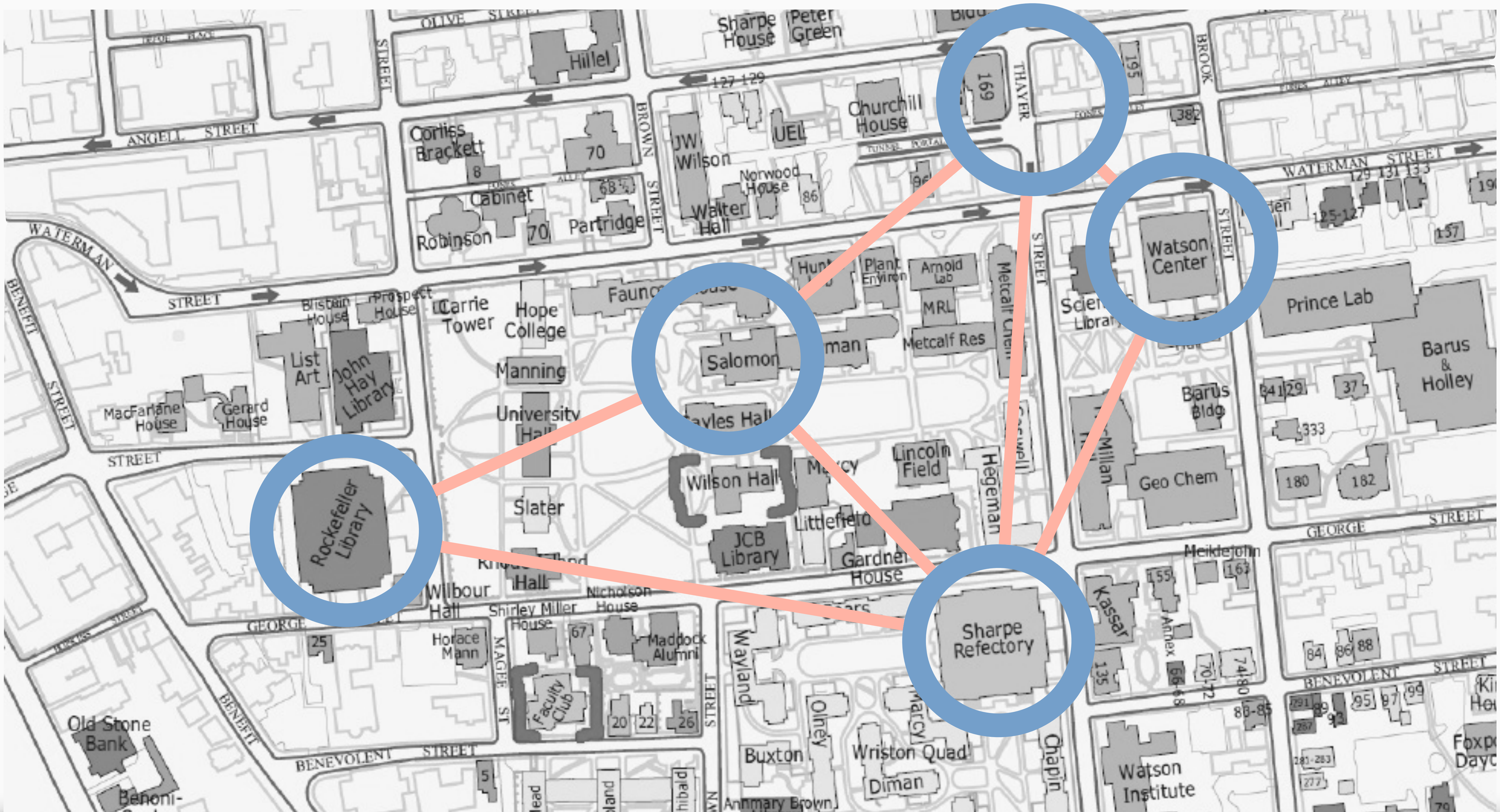
- *Input:*
  - Some numbers: 5,2,14,6,12,1,3
  - Proposed solution: 2,6,4
- *Output:*
  - True if 2,6,4 adds to 10
  - **False** if 2,6,4 adds to10

But: no known efficient solution for solving this problem!

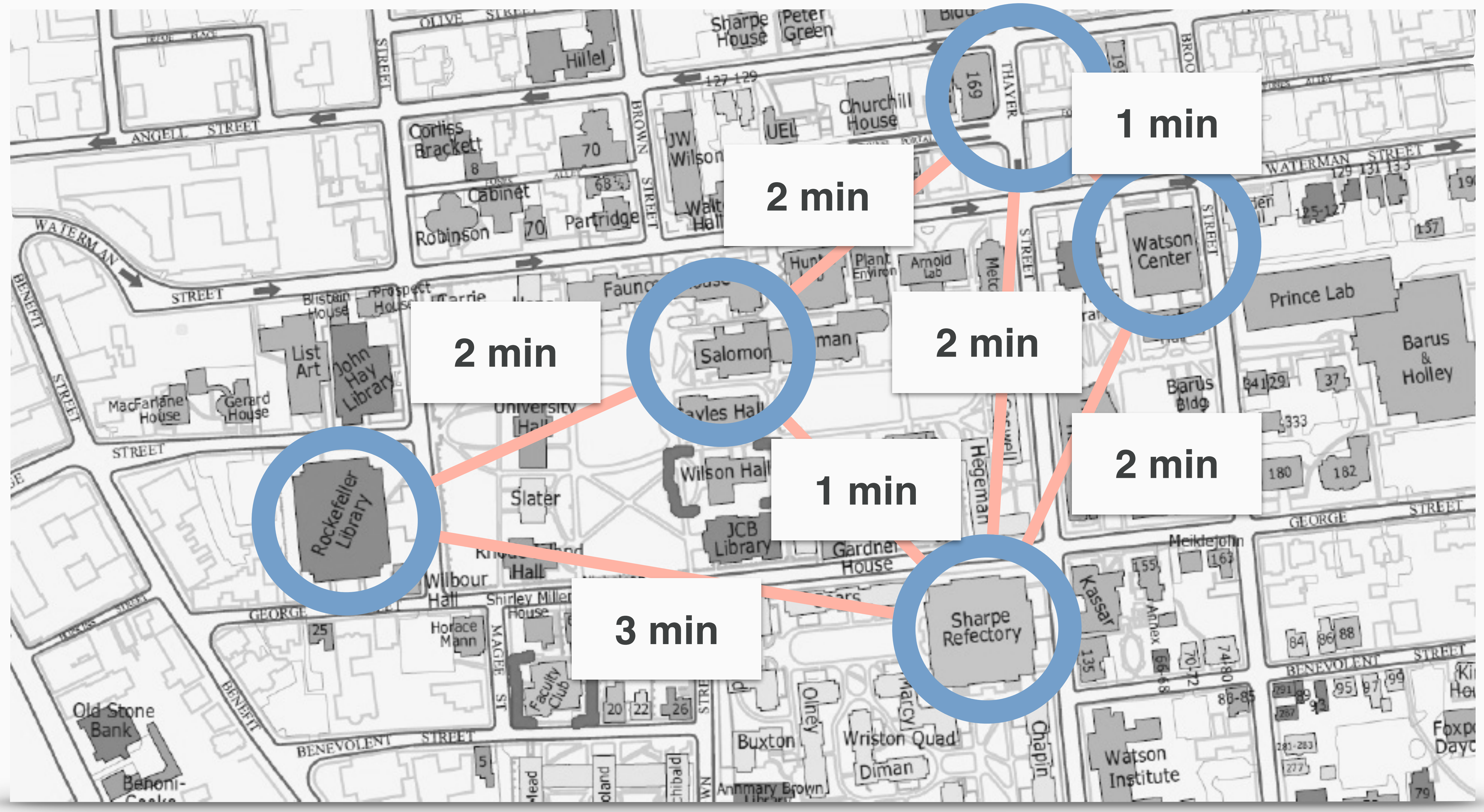
Circles + Lines!



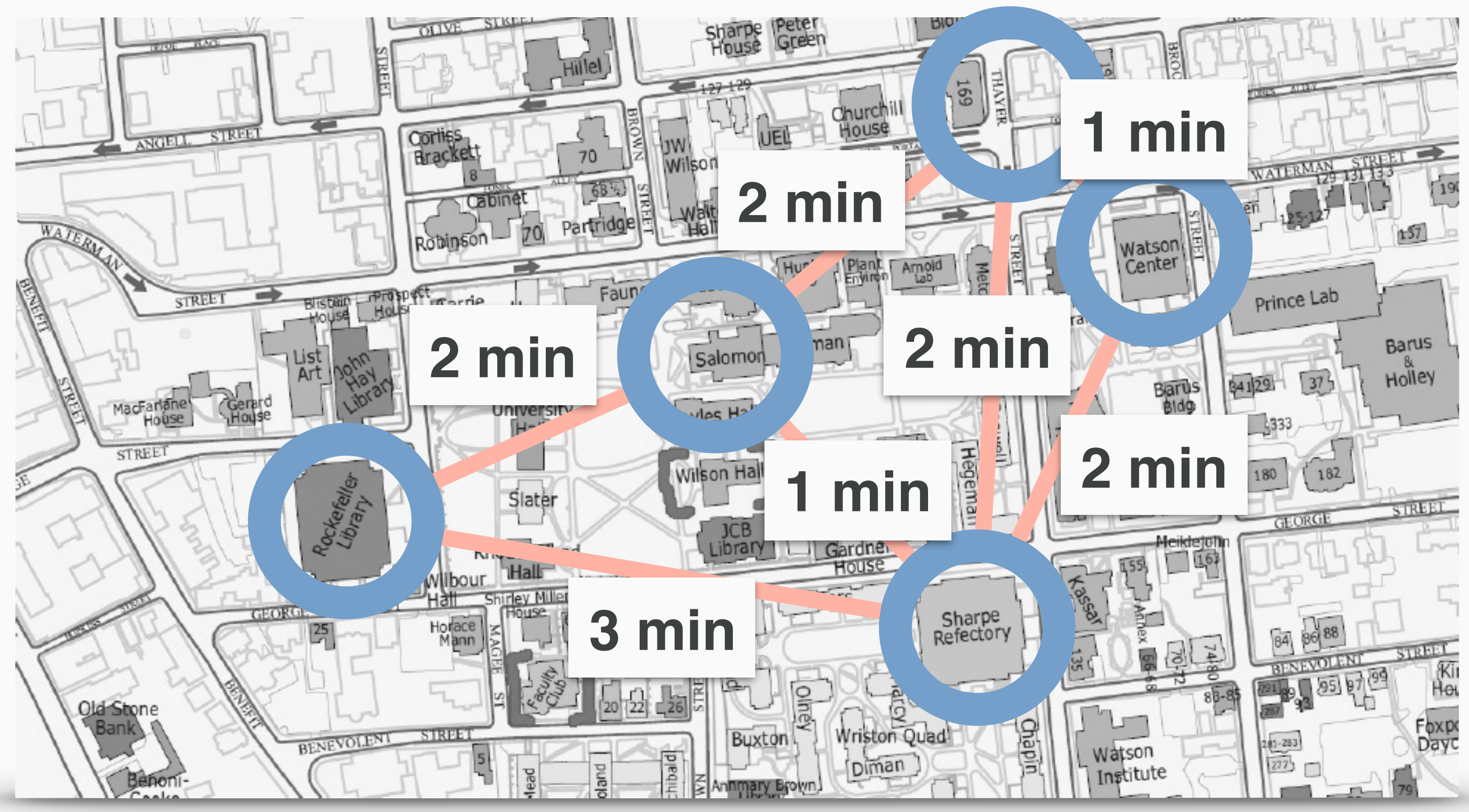
Circles + Lines!



Circles + Lines!



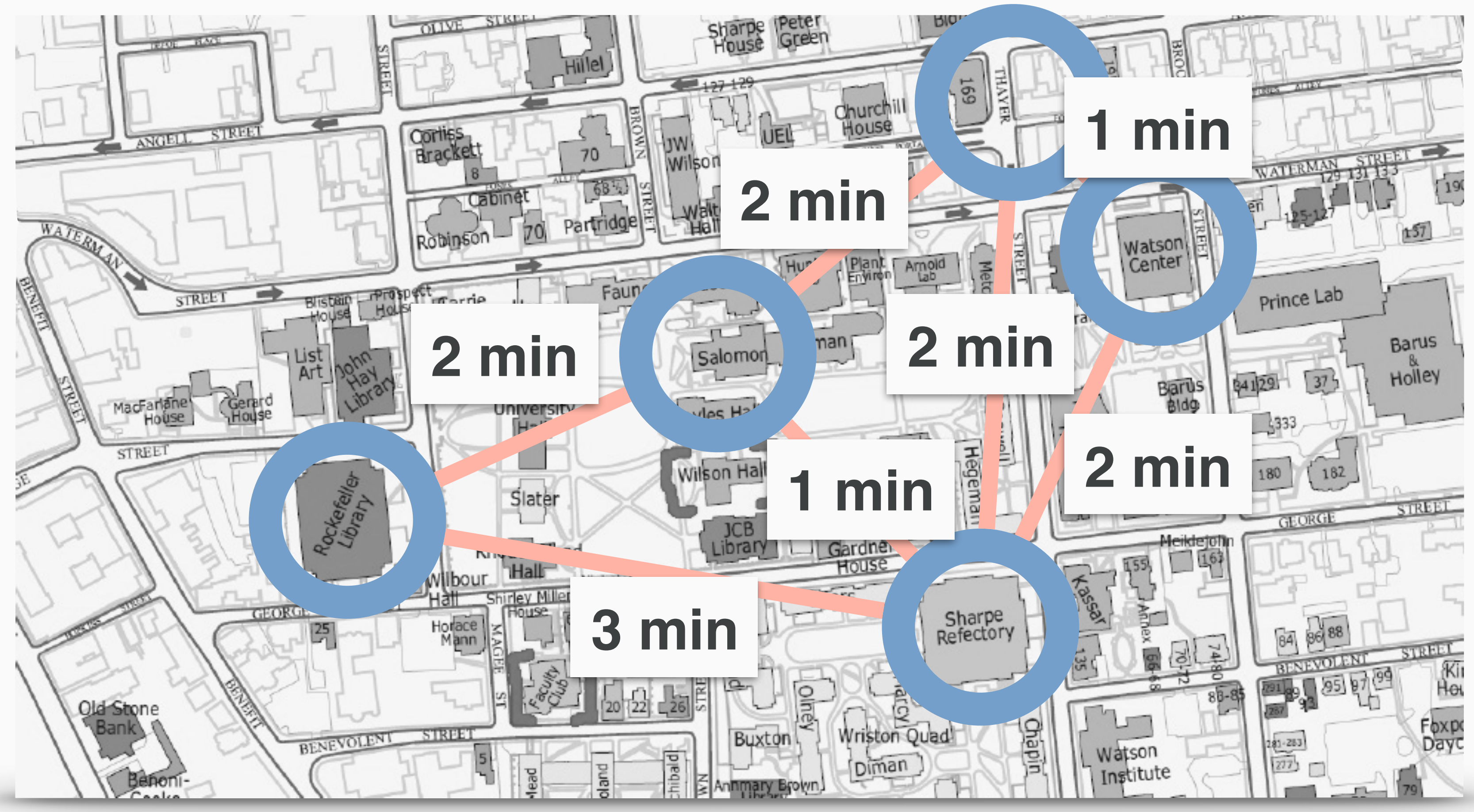
Circles + Lines!



Circle: an object (a location in this case)

Line: a relation between objects (tram, in this case)

“Graph”



Circle: an object (a location in this case)

Line: a relation between objects (tram, in this case)

Reflection

- Q: If candidate solutions to a problem can be verified efficiently, can the problem also be solved efficiently?
- Verification Examples:
  - Can this logical formula be true?
  - Subset Sum
  - Circle + Line problems!

A Disclaimer

Takeaway

There are problems that are unsolvable, period.

Overview: The Uncomputable

- Some problems *have no correct algorithm*.
- Closely related to:
  - Self-reference
  - Paradoxes
  - Infinity

Things That Describe Themselves

“I am a sentence consisting of eight words”

Things That Describe Themselves

Another view:

“X” Is a sentence with eight words in it.

Things That Describe Themselves

Another view:

“X” Is a sentence with eight words in it.

X = “The old man the boat”

“The old man the boat” Is a sentence with eight words in it.

Things That Describe Themselves

Another view:

“X” Is a sentence with eight words in it.

X = “The old man the boat”

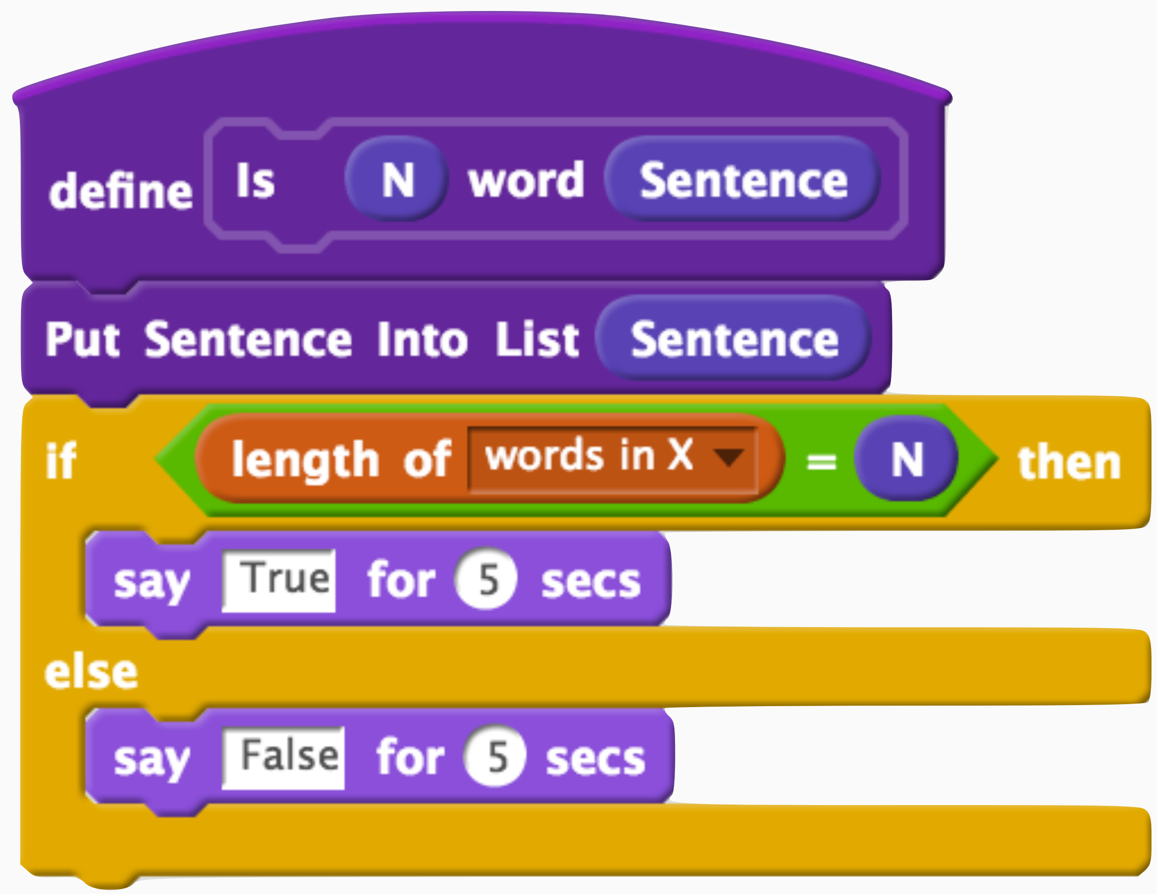
“The old man the boat” Is a sentence with eight words in it.

False

Things That Describe Themselves

Another view:

Does "sentence" have N words?



Things That Describe Themselves

Q: What about things that lie about themselves?

Things That Describe Themselves

Q: What about things that lie about themselves?

“This sentence is False”

Things That Describe Themselves

Q: What about things that lie about themselves?

"This sentence is False"

True

Things That Describe Themselves

Q: What about things that lie about themselves?

"This sentence is False"

True

Contradiction

Things That Describe Themselves

Q: What about things that lie about themselves?

"This sentence is False"

True

False

Contradiction

Things That Describe Themselves

Q: What about things that lie about themselves?

"This sentence is False"

(so the content of the sentence has to not be true)

True

False

Contradiction

True

Things That Describe Themselves

Q: What about things that lie about themselves?

"This sentence is False"

(so the content of the sentence has to not be true)

True

False

Contradiction

True

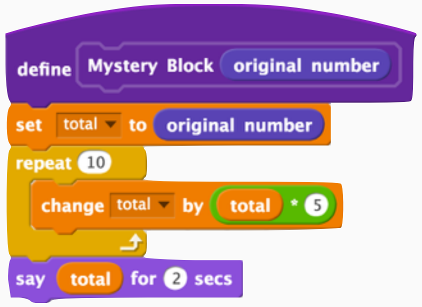
Contradiction

The Halting Problem

- *INPUT:*
    - A description of a program, input to the program.
- *OUTPUT:*
    - True if the program will halt when run on that input.
    - False if the program does not halt with that input.

The Halting Problem

- *INPUT:*



6

  - A **description** of a program, **input** to the program.

- *OUTPUT:*
  - True if the program will halt when run on that input.
  - False if the program does not halt with that input.

The Halting Problem

- Suppose we have a correct algorithm for this problem:

The Halting Problem

- Suppose we have a correct algorithm for this problem:

- Linear Search
- Dictionary
- “Snow”

The Halting Problem

- Suppose we have a correct algorithm for this problem:

The Halting Problem

- Suppose we have a correct algorithm for this problem:

- Random Search
- Dictionary
- “Flaboygle”

The Halting Problem

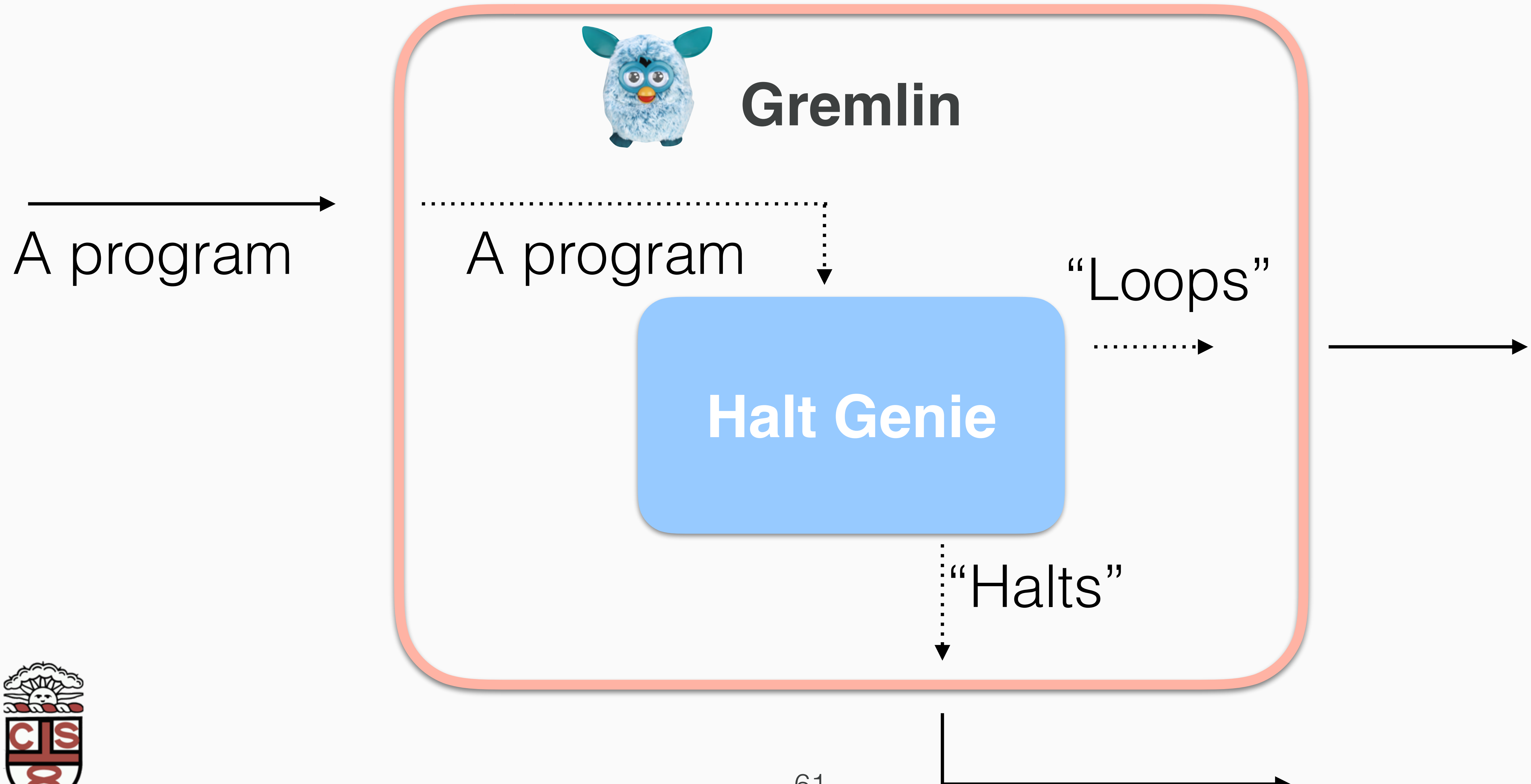
- Suppose we have a correct algorithm for this problem:

The Halting Problem

Now consider the wily Gremlin:

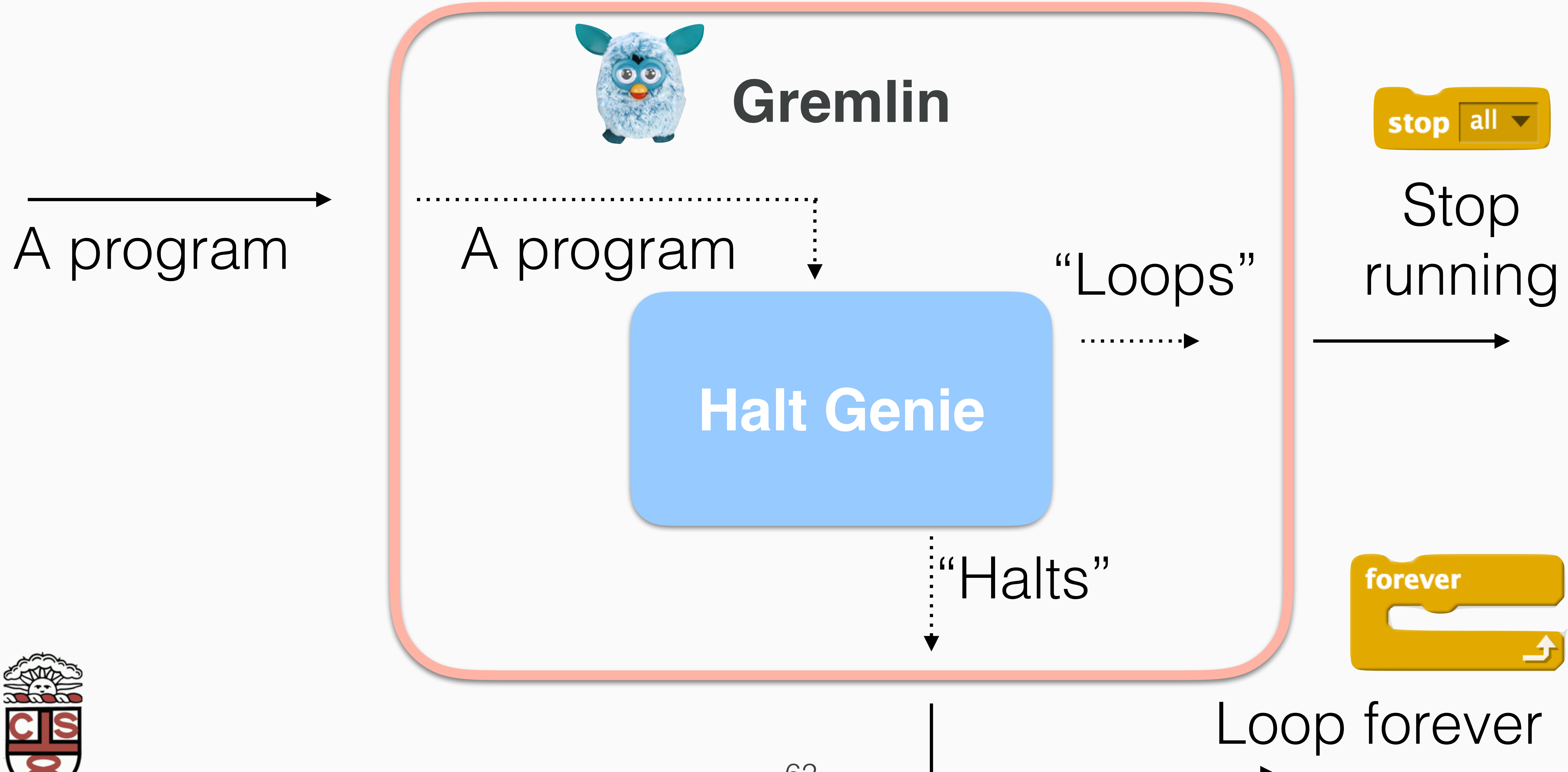
The Halting Problem

Now consider the wily Gremlin:



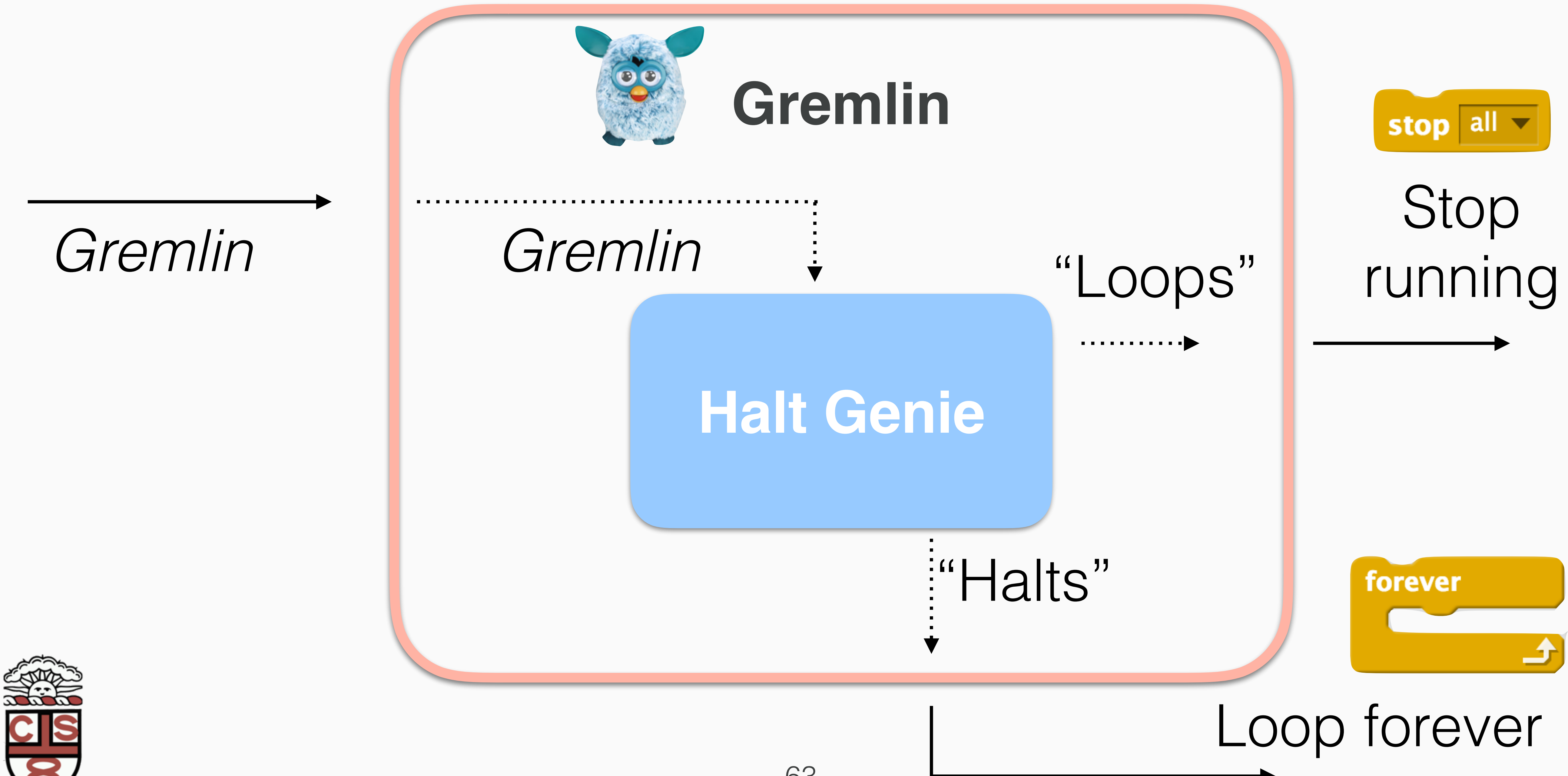
The Halting Problem

Now consider the wily Gremlin:



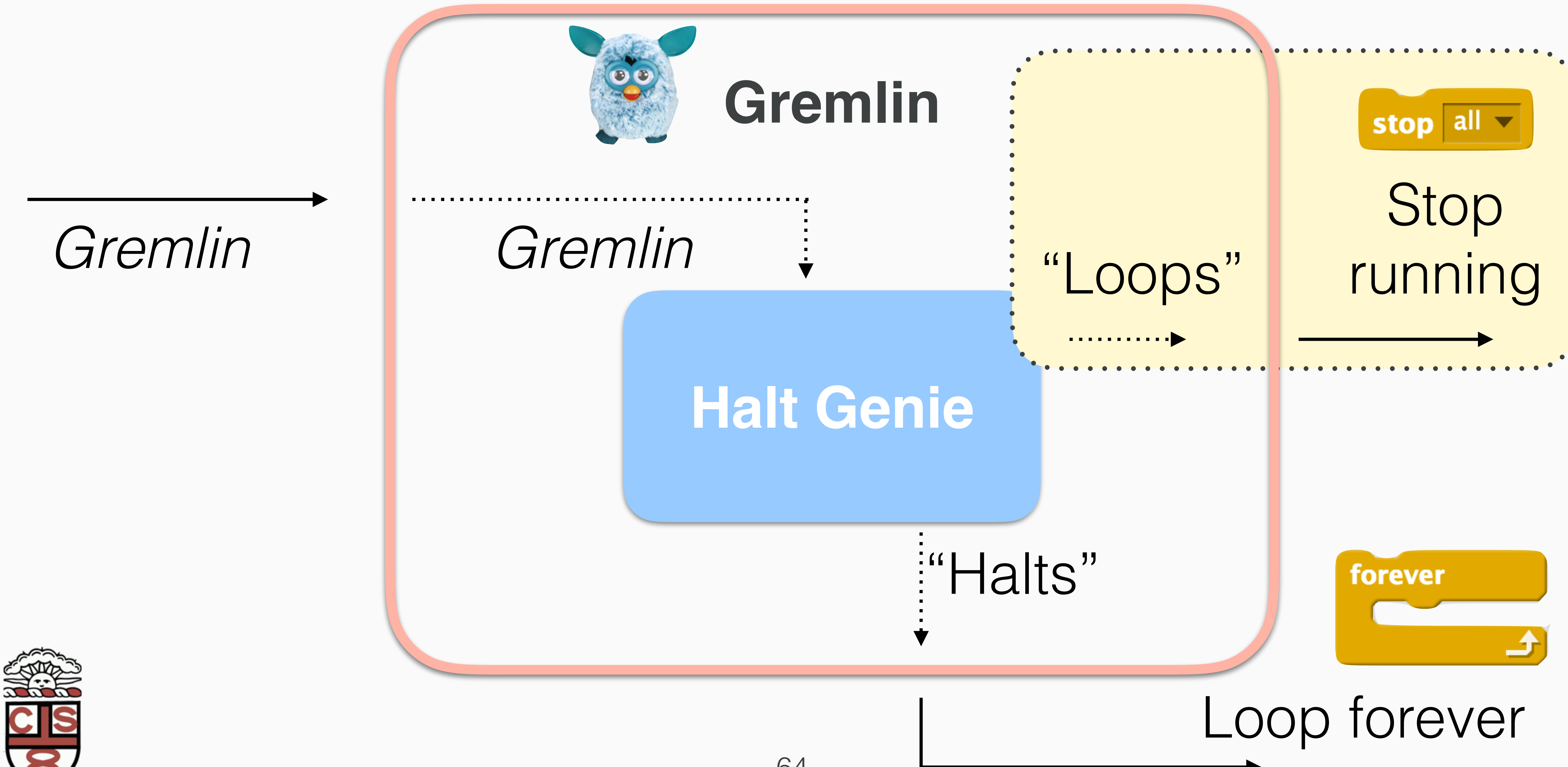
The Halting Problem

Now consider the wily Gremlin:



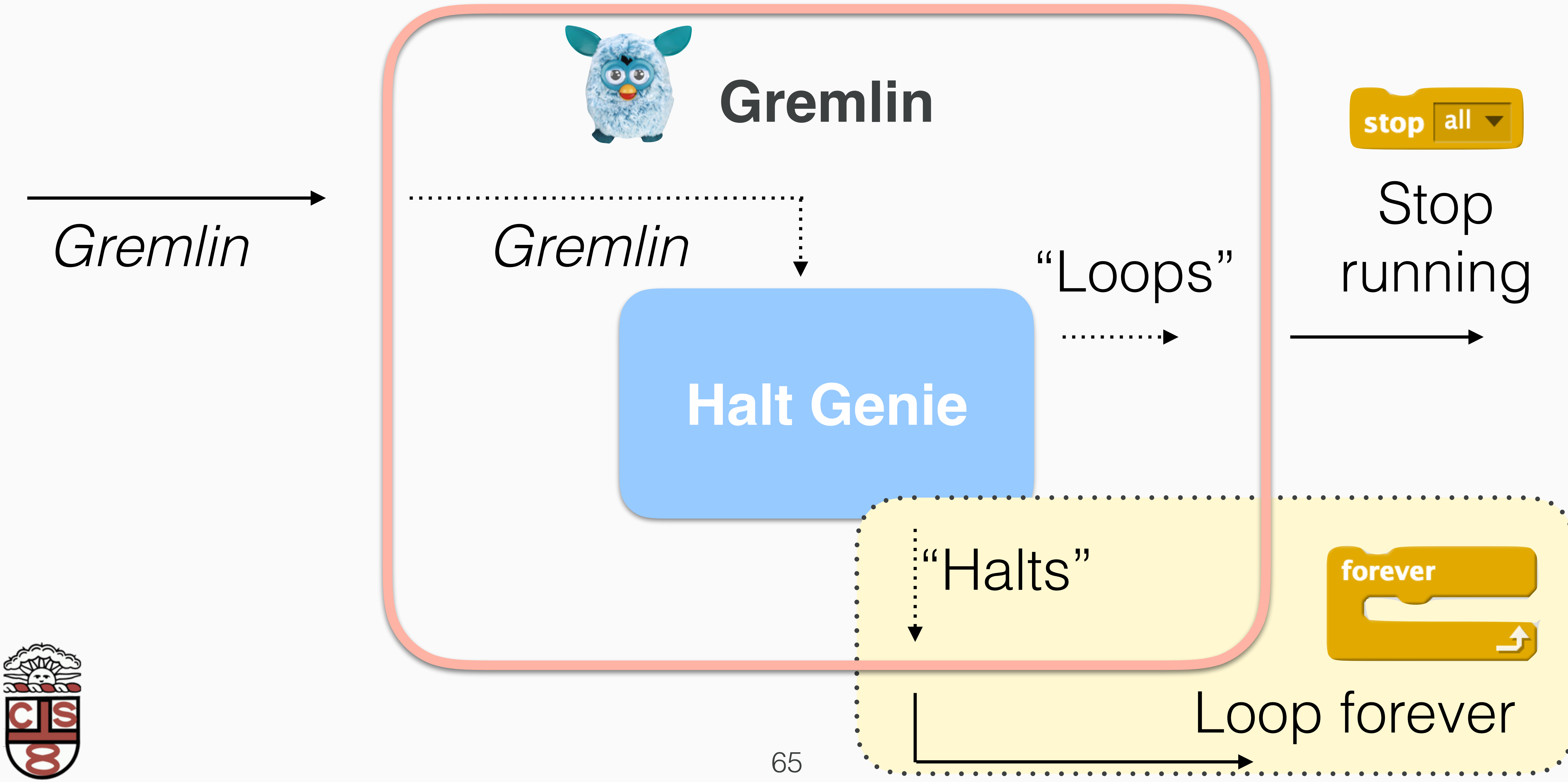
The Halting Problem

Now consider the wily Gremlin:



The Halting Problem

Now consider the wily Gremlin:



The Halting Problem

If a Halt Genie exists (i.e. a correct algorithm for the Halting Problem), then we get a contradiction!

The Halting Problem

If a Halt Genie exists (i.e. a correct algorithm for the Halting Problem), then we get a contradiction!

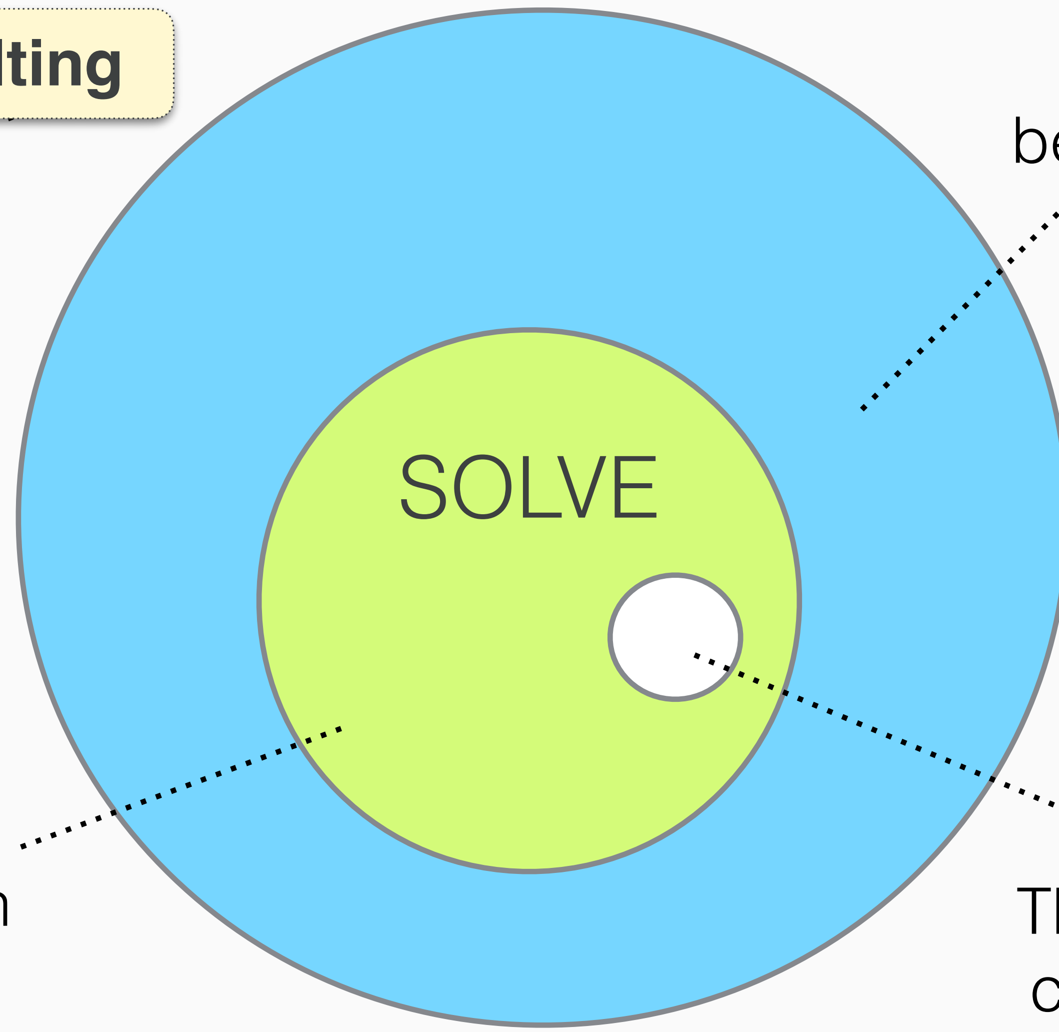
Therefore, no correct algorithm can exist for this problem.

Computability

Halting

Q: What, if anything, is out here?

A: The Halting Problem (and friends)



Things that can be computed, period.

Things a regular computer can compute before the sun goes supernova

Things a domino computer could compute before the sun goes supernova

Implications 1

- Closely related to Gödel's Incompleteness Theorems!
  - Mathematical statements that say of themselves, "I am not provable".
  - Result: there are mathematical statements that are unprovable, even though they're true.
  - Result: No mathematical system can prove certain properties about the system itself.
  - I.e. Can Logic prove that logic always makes the right conclusions?
- Gödel and Turing spoke at length about the relationship between consciousness, self-reference, and statements of this kind.
- Book: Gödel, Escher, Bach, an Eternal Golden Braid

Implications 2

- If we had a Halting Genie, we would have a pretty simple algorithm for solving any problem (even those that involve infinite loops).
  - Set it up so that, for *any question:*
    - "True" answers Halt
    - "False" answers Loop forever
  - Then, we just give the program to the Halting Genie. Boom! We have our answer.

Computable Numbers

- Consider:
  - 1,2,3,4,5….. ∞
- We could count twice as fast:
  - 2,4,6,8,10,… ∞
- Or even ten times as fast:
  - 10,20,30,40,… ∞

Computable Numbers

- Consider:
  - 1,2,3,4,5….. ∞
- We could count twice as fast:
  - 2,4,6,8,10,… ∞

    1-2-3-4-5
- Or even ten times as fast:
  - 10,20,30,40,… ∞

    1-2-3-4

But with both of these, we can still count numbers in the sequence

Computable Numbers

- Consider:
  - 1,2,3,4,5….. ∞
- We could count twice as fast:
  - 2,4,6,8,10,… ∞

    1-2-3-4-5
- Or even ten times as fast:
  - 10,20,30,40,… ∞

    1-2-3-4

But with both of these, we can still count numbers in the sequence

Q: What about decimals?

Computable Numbers

- Consider:
  - 1,2,3,4,5….. ∞
- We could count twice as fast:
  - 2,4,6,8,10,… ∞

    1-2-3-4-5
- Or even ten times as fast:
  - 10,20,30,40,… ∞

    1-2-3-4

But with both of these, we can still count numbers in the sequence

Q: What about decimals?

0.1

Computable Numbers

- Consider:
  - 1,2,3,4,5..... ∞
- We could count twice as fast:
  - 2,4,6,8,10,... ∞

    1-2-3-4-5
- Or even ten times as fast:
  - 10,20,30,40,... ∞

    1-2-3-4

But with both of these, we can still count numbers in the sequence

Q: What about decimals?

0.1

0.01

Computable Numbers

- Consider:
  - 1,2,3,4,5….. ∞
- We could count twice as fast:
  - 2,4,6,8,10,… ∞

    1-2-3-4-5
- Or even ten times as fast:
  - 10,20,30,40,… ∞

    1-2-3-4

But with both of these, we can still count numbers in the sequence

Q: What about decimals?

0.1

0.01

0.001….

Computable Numbers

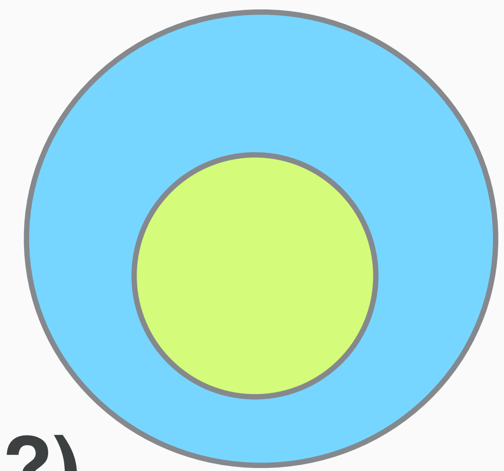
Things we can count	Things we can't count
1,2,3,4,5…	.01,.000001,.0000001…

Computable Numbers

Takeaway: there are numbers we cannot compute: those that are infinitely long decimals.

Reasoning: we can't even count them!

Reflection



- There are two notions of solvable:
  - **Computability (what's in the blue bubble?)**
    - What is solvable in principle (i.e. it doesn't matter how much time we have)
  - **Complexity (what's in SOLVE?):**
    - Given that we exist in *this* universe, and are subject to time constraints, how long do we have to sit around and wait for an answer?
- We're still trying to figure out which problems are in the second!