

Discrete Mathematics (for Computer Science)

Part II: Computability

Zeyu Mi

TA: Zhuohao Shen

2022-11-16



What problems can a computer solve?

Is a number
prime?

Easy!

A prime number is a natural number greater than 1 whose only factors are 1 and itself

Is a number
prime?

Easy!

A prime number is a natural number greater than 1 whose only factors are 1 and itself

Is a number
prime?

We only need to verify there are no
its factors from 2 to itself-1

```
bool isPrime(int n){  
    for(int i=2;i<n;i++){  
        if(n%i==0) return false;  
    }  
    return true;  
}
```

*Is a person good
programmer?*

What is a good programmer?

Well-known reality:

Produce 1000 lines of bug-free code per hour.

Sleep less than 5 hours.

Have less hair than 99% people.

Is a person good
programmer?

What is a good programmer?

Well-known reality:

Produce 1000 lines of bug-free code per hour.

Sleep less than 5 hours.

Have less hair than 99% people.

Is a person good
programmer?

We need to analyze his/her code data,
sleep data and hair data to say if
he/she is qualified.

```
bool isGoodProgrammer  
(codedata, sleepdata, hairdata){  
    .....  
}
```

Does god exist?

Emmmm

Does god exist?

What if god exists? And what if not?
What should we take into computation.

I can't tell. I don't think a computer
can solve this problem.

Easy!

Output
Constraints

Input

Is a number
prime?

A prime number is a natural number greater than 1 whose only factors are 1 and itself

We only need to verify there are no its factors from 2 to itself-1

Solution

Code

```
bool isPrime(int n){  
    for(int i=2;i<n;i++){  
        if(n%i==0) return false;  
    }  
    return true;  
}
```

What's a good programmer?

Output
Constraints

"Person" is too
abstract for
computation

Produce 1000 lines of bug-free code per hour.
Sleep less than 5 hours.
Have less hair than 99% person.

Is a person good
programmer?

Input

We need to analyze his/her code data,
sleep data and hair data to say if
he/she is qualified.

Solution

```
bool isGoodProgrammer  
(codedata, sleepdata, hairdata){  
    .....  
}
```

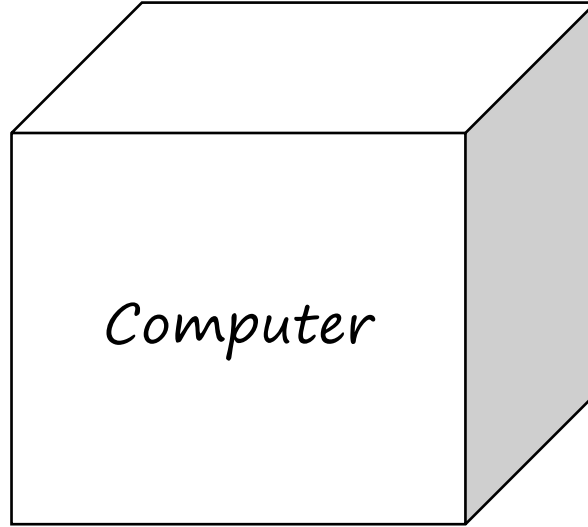
Code

Computational Problem (计算问题)

- **Problems for computation usually have:**
 - Well-defined input
 - Constraints that the output must satisfy.
- **Some problems are hard because they are hard to converted into computational problems**
 - Existence of god
- **We only focus on the computational problems**

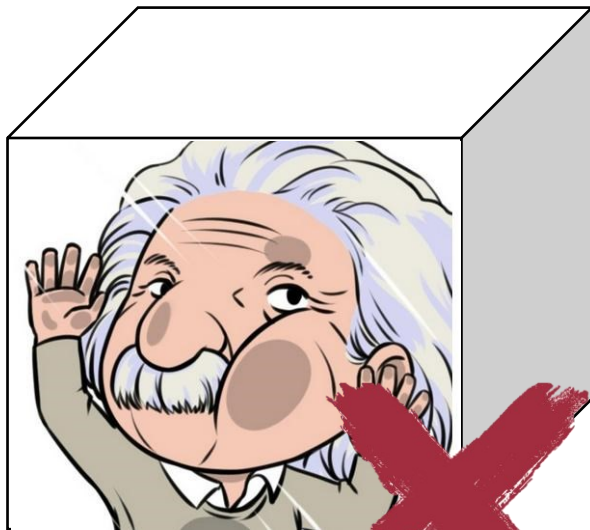
What about “computable”?

Input



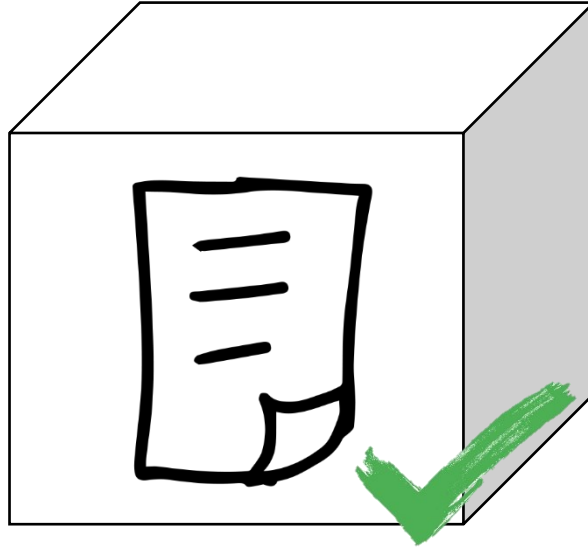
Output

Input

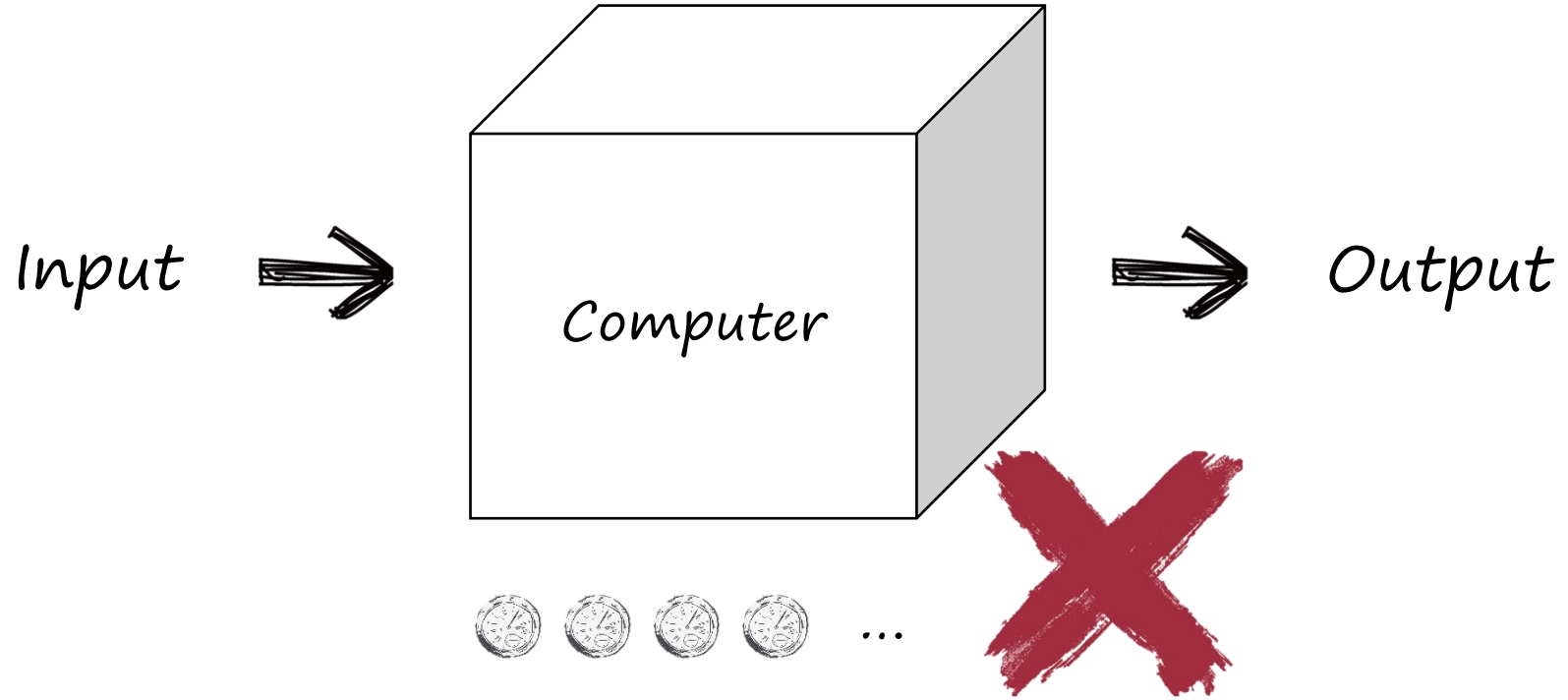


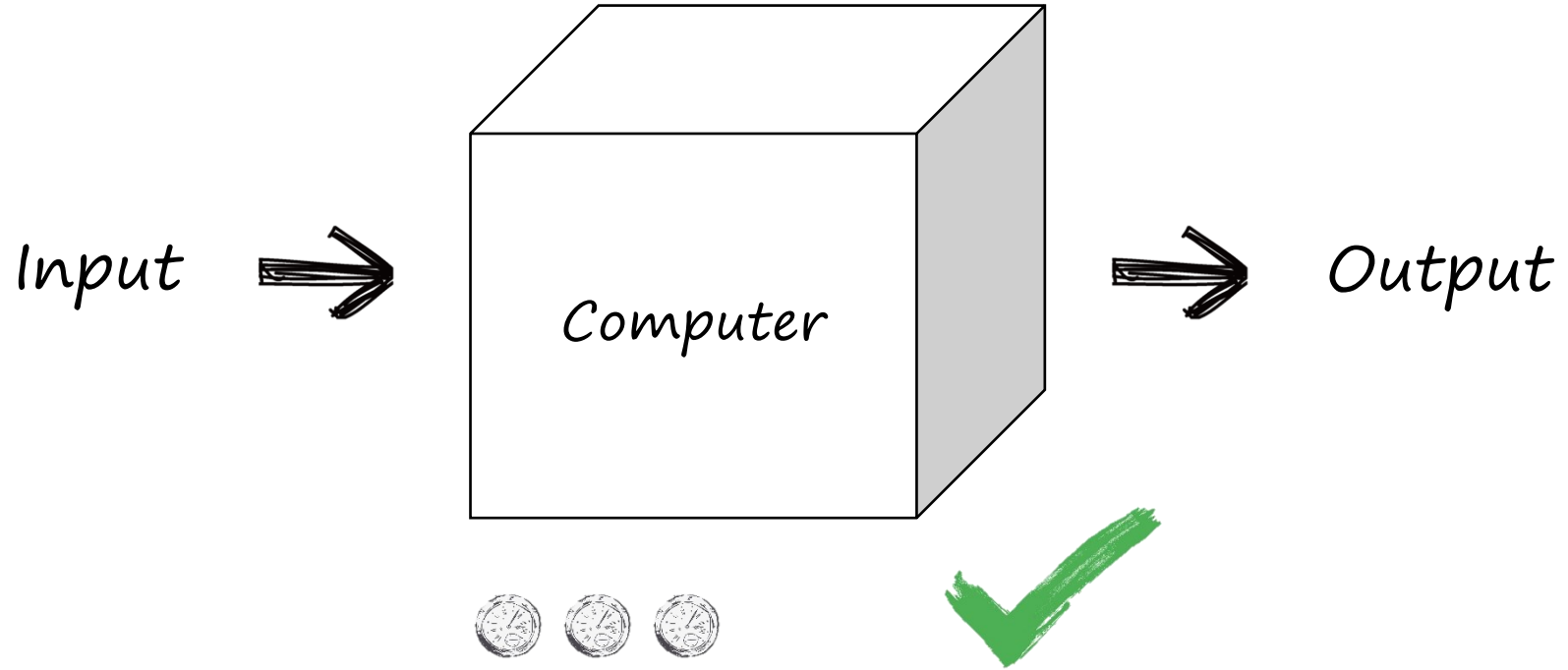
Output

Input



Output





Computable (可计算的)

- **A computational problem is computable if there is a procedure (or algorithm) for it which:**
 - Rigorously follows some finite instructions, requires no ingenuity.
 - Always finishes (terminates) after a finite number of steps.

*Are all the computational problem
computable?*

Entscheidungsproblem (判定问题)

All axioms of math (Finite)

There is no natural number
whose successor is 0
...

First-order logic (谓词逻辑)

$\forall x \in N (0 \neq S(x))$
...

A statement of
first-order logic

Input



Prover



True. The statement can be
deduced from the axioms



False. Otherwise.

How do we investigate computability?

Problem has large varieties.

Graph Theory
A web server
A recommendation system
....

Unify



How to represent a problem?

Computer has many details.

Instruction set architecture
Memory management
Device management
...

Simplify



How to represent computation?

How do we investigate computability?

2.1 Discussion on Problems

Part1. Intro & Set theory(I) : Basics & Formal Language.

Part2. Set Theory(II): Axiom system & Cardinality.

Part3. Capture Structures: Binary Relation & Function

2.2 Discussion on Computation

Part1. Turing Machine Basics.

Part2. Variants of Turing Machine. Church-Turing Thesis.

2.3 Discussion on Computability

Part1. The Language of Turing Machine. R & RE

Part2. Undecidability.

How do we investigate computability?

《数理逻辑与集合论》9.1-9.5
Stanford CS103 Finite Automata

2.1 Discussion on Problems

Part1. Intro & Set theory(I) : Basics & Formal Language.

Part2. Set Theory(II): Axiom system & Cardinality.

Part3. Capture Structures: Binary Relation & Function

2.2 Discussion on Computation

Part1. Turing Machine Basics.

Part2. Variants of Turing Machine. Church-Turing Thesis.

2.3 Discussion on computability

Part1. The Language of Turing Machine. R & RE

Part2. Undecidability.

What is a set?

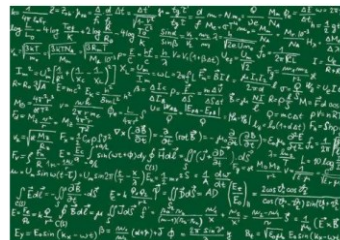
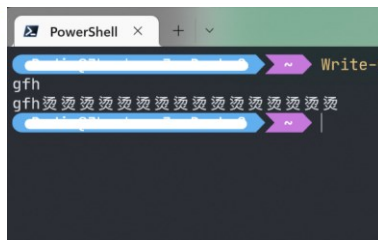
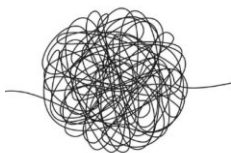
Emojis:



The hobbies of a talented man.



Or a bunch of things that even have no exact meaning



Definition

Set

A Set is an unordered collection of distinct objects, which may be anything (including other sets).

If a is an item of set A , we say that a belongs to A , writing as :

$$a \in A$$

If b is not an item of set A , we say that b doesn't belongs to A , writing as:

$$b \notin A$$

Representation

- **For simple sets, we can enumerate all the objects wrapped by curly braces**
 - {sing, jump, rap, basketball}
- **For some common sets, we use specific notations for them**

$\mathbb{N}$: All the natural number: $0, 1, 2, 3, \dots$

$\mathbb{Z}$: All the integer: $\dots, -2, -1, 0, 1, 2, \dots$

$\mathbb{Q}$: All the rational number

$\mathbb{R}$: All the real number

$\mathbb{C}$: All the complex number

$\emptyset$: Empty set, contains nothing. $\{\}$

U : The universal set, contains all the objects under *consideration*

Representation

For sets whose object has a certain property, we'll use set-builder notation

Set-Builder Notation

$$A = \{x|P(x)\} \text{ or } A = \{x: P(x)\}$$

Where P is a predicate, and A is the set of all elements which makes P true

$$\{x|x \text{ is a prime number}\} = \{2,3,5,7,11, \dots\}$$

Unorder (无序性)

The order of objects in a set doesn't matter



and



are identical

Distinction (相异性)

The objects in a set are distincted, or the repeated elements are ignored.



and



are identical

Deterministic (确定性)

For any set A and any object a , whether a belongs to A is determinate and mutually exclusive.

either $a \in A$ or $a \notin A$

Relations Between Sets

Equal

A and B are equal iff they have the same elements

$$A = B \Leftrightarrow (\forall x)(x \in A \leftrightarrow x \in B)$$

Subset

A is subset of B iff every element of A is element of B

$$A \subseteq B \Leftrightarrow (\forall x)(x \in A \rightarrow x \in B)$$

Proper Subset

A is proper subset of B iff A is subset of B and A is not equal to B

$$A \subset B \Leftrightarrow (A \subseteq B \wedge A \neq B)$$

Relations Between Sets

$$1 \in \{1, 2, 3\}$$

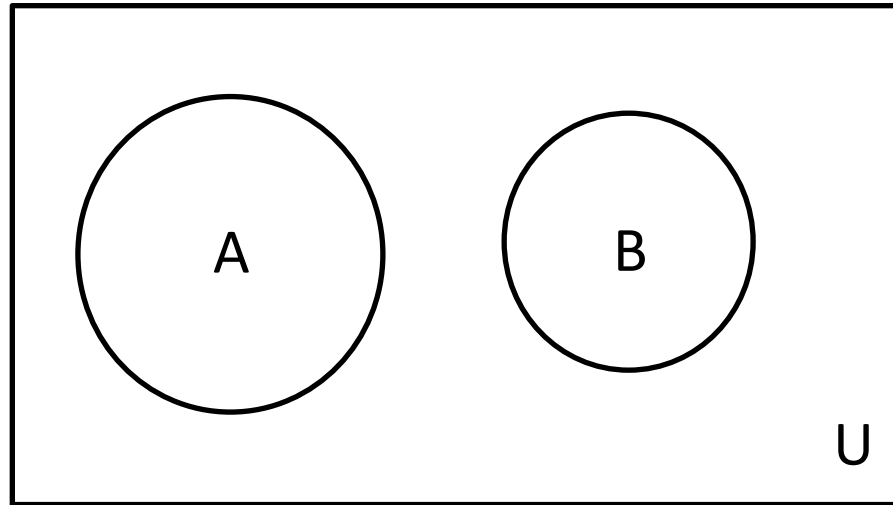
$$\{1\} \subset \{1, 2, 3\}$$

$$\emptyset \subset \{1, 2, 3\}$$

$$\emptyset \in \{\emptyset\} \text{ and } \emptyset \subset \{\emptyset\}$$

For any set A , $\emptyset \subseteq A$

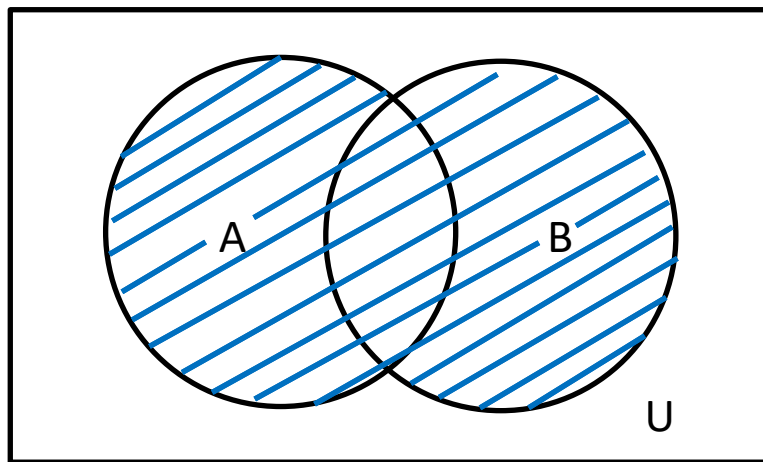
Venn Diagrams



Set Operation

Union(并)

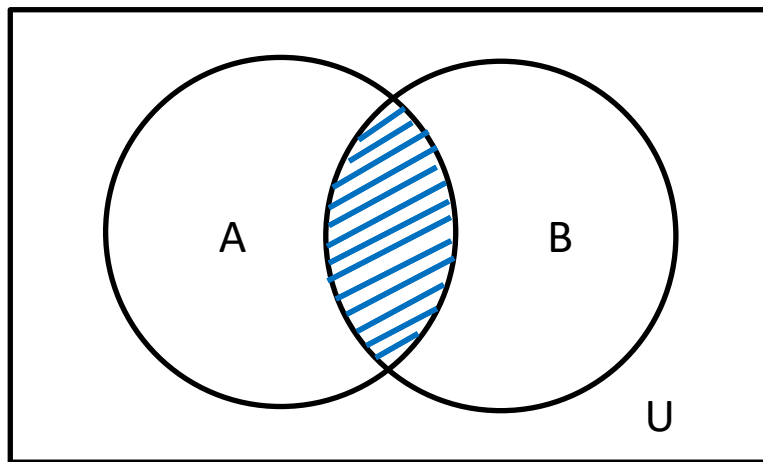
$$A \cup B = \{x | x \in A \vee x \in B\}$$



Intersection

Intersection(交)

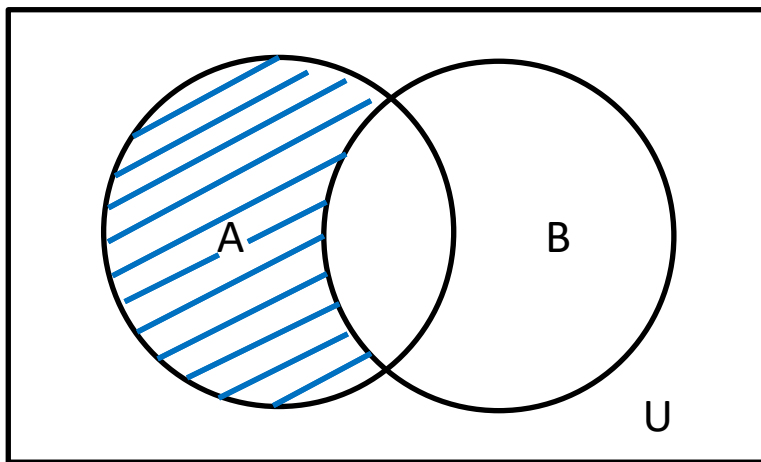
$$A \cap B = \{x | x \in A \wedge x \in B\}$$



Difference

Difference(差)

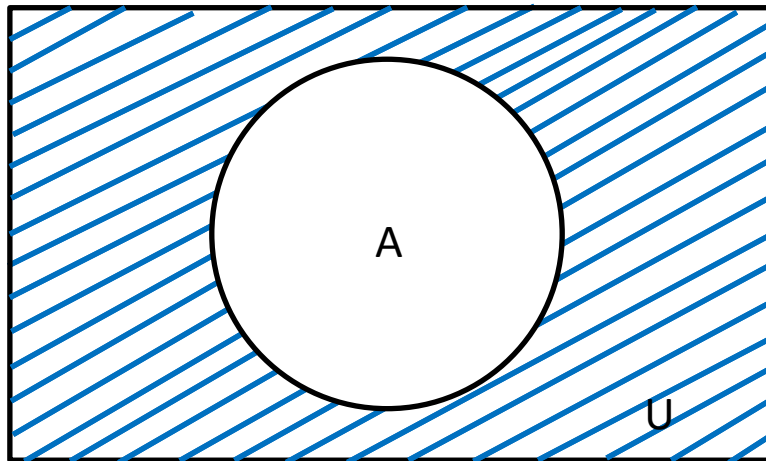
$$A - B = \{x | x \in A \wedge x \notin B\}$$



Complement

Complement(補)

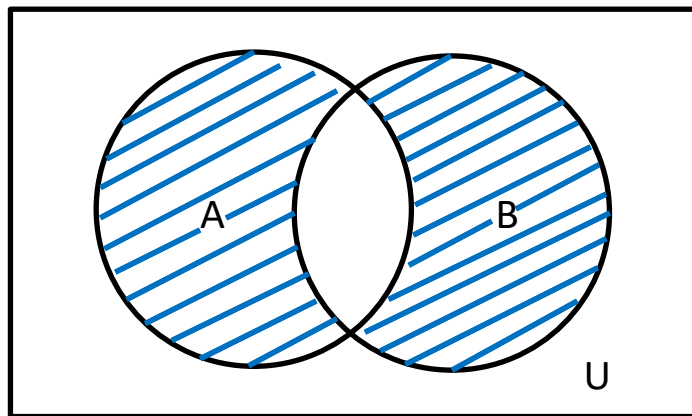
$$\neg A \text{ or } \bar{A} = \{x | x \in U \wedge x \notin A\}$$



Symmetric Difference

Symmetric Difference(对称差)

$$A \oplus B = \{x | x \in A \bar{\vee} x \in B\} = (A - B) \cup (B - A)$$



Generalized Union/Intersection

Generalized Union/Intersection(广义并/交)

$$\bigcup A = \{x | (\exists z)(z \in A \wedge x \in z)\}$$

$$\bigcap A = \{x | (\forall z)(z \in A \rightarrow x \in z)\}$$

$$A = \{\{a, b, c\}, \{a, b\}, \{b, c, d\}\}$$

$$\text{Then } \bigcup A = \{a, b, c, d\}, \bigcap A = \{b\}$$

Define that $\bigcup \emptyset = \emptyset$, $\bigcap \emptyset$ is undefined!

Set Identities (集合恒等式)

Identity laws (同一律)

$$A \cap U = A$$

$$A \cup \emptyset = A$$

Commutative laws (交换律)

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

Domination laws (零律)

$$A \cup U = U$$

$$A \cap \emptyset = \emptyset$$

Associative laws (结合律)

$$A \cup (B \cup C) = (A \cup B) \cup C$$

$$A \cap (B \cap C) = (A \cap B) \cap C$$

Idempotent laws (幂等律)

$$A \cup A = A$$

$$A \cap A = A$$

Distributive laws (分配律)

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

Set Identities

Complementation law (双补律)

$$\overline{(\overline{A})} = A$$

Complement laws (补余律)

$$A \cup \overline{A} = U$$

$$A \cap \overline{A} = \emptyset$$

Absorption laws (吸收律)

$$A \cup (A \cap B) = A$$

$$A \cap (A \cup B) = A$$

De Morgan's laws (摩根律)

$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$

$$A - (B \cap C) = (A - B) \cup (A - C)$$

$$A - (B \cup C) = (A - B) \cap (A - C)$$

Proof of Set Identities

Method 1: Using definition

Example: $A \cup B = B \Rightarrow A \subseteq B$

Analysis: We need to prove $A \subseteq B \Leftrightarrow \forall x(x \in A \rightarrow x \in B)$, so the proof structure is

for any x , assume $x \in A$, {...bunch of inference...}, we get $x \in B$

Proof:

For any x , assume $x \in A$, according to the definition of set union, $x \in A \cup B$. And $A \cup B = B$, so we get $x \in B$. Q.E.D.

Proof of Set Identities

Method 2: Using logical equivalent operation

Example: $A - (B \cup C) = (A - B) \cap (A - C)$

Proof:

$$\begin{aligned}x \in A - (B \cup C) &\Leftrightarrow (x \in A) \wedge (x \notin (B \cup C)) \\&\Leftrightarrow x \in A \wedge x \notin B \wedge x \notin C \\&\Leftrightarrow (x \in A \wedge x \notin B) \wedge (x \in A \wedge x \notin C) \\&\Leftrightarrow (x \in (A - B)) \wedge (x \in (A - C)) \\&\Leftrightarrow x \in ((A - B) \cap (A - C))\end{aligned}$$

Proof of Set Identities

Method 3: Deriving a set algebra using known identities or equations

Example: $\overline{A \cup (B \cap C)} = (\overline{C} \cup \overline{B}) \cap \overline{A}$

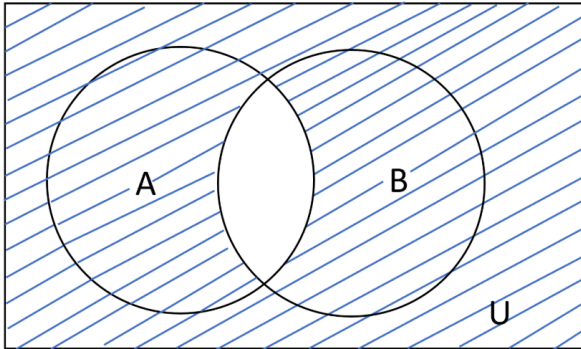
Proof:

$$\begin{aligned}\overline{A \cup (B \cap C)} &= \overline{A} \cap \overline{(B \cap C)} && \text{by the second De Morgan law} \\ &= \overline{A} \cap (\overline{B} \cup \overline{C}) && \text{by the first De Morgan law} \\ &= (\overline{B} \cup \overline{C}) \cap \overline{A} && \text{by the commutative law for intersections} \\ &= (\overline{C} \cup \overline{B}) \cap \overline{A} && \text{by the commutative law for intersections}\end{aligned}$$

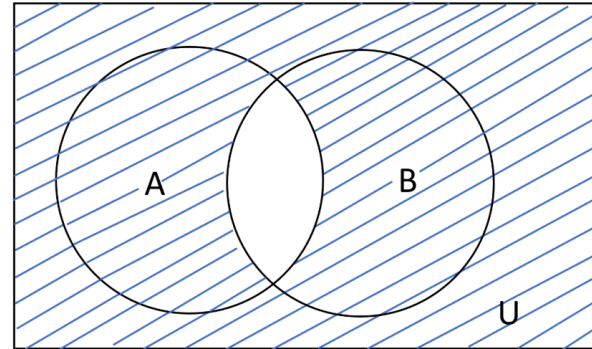
Proof of Set Identities

We can use venn diagram to help analyzing.

$$\overline{A \cap B}$$



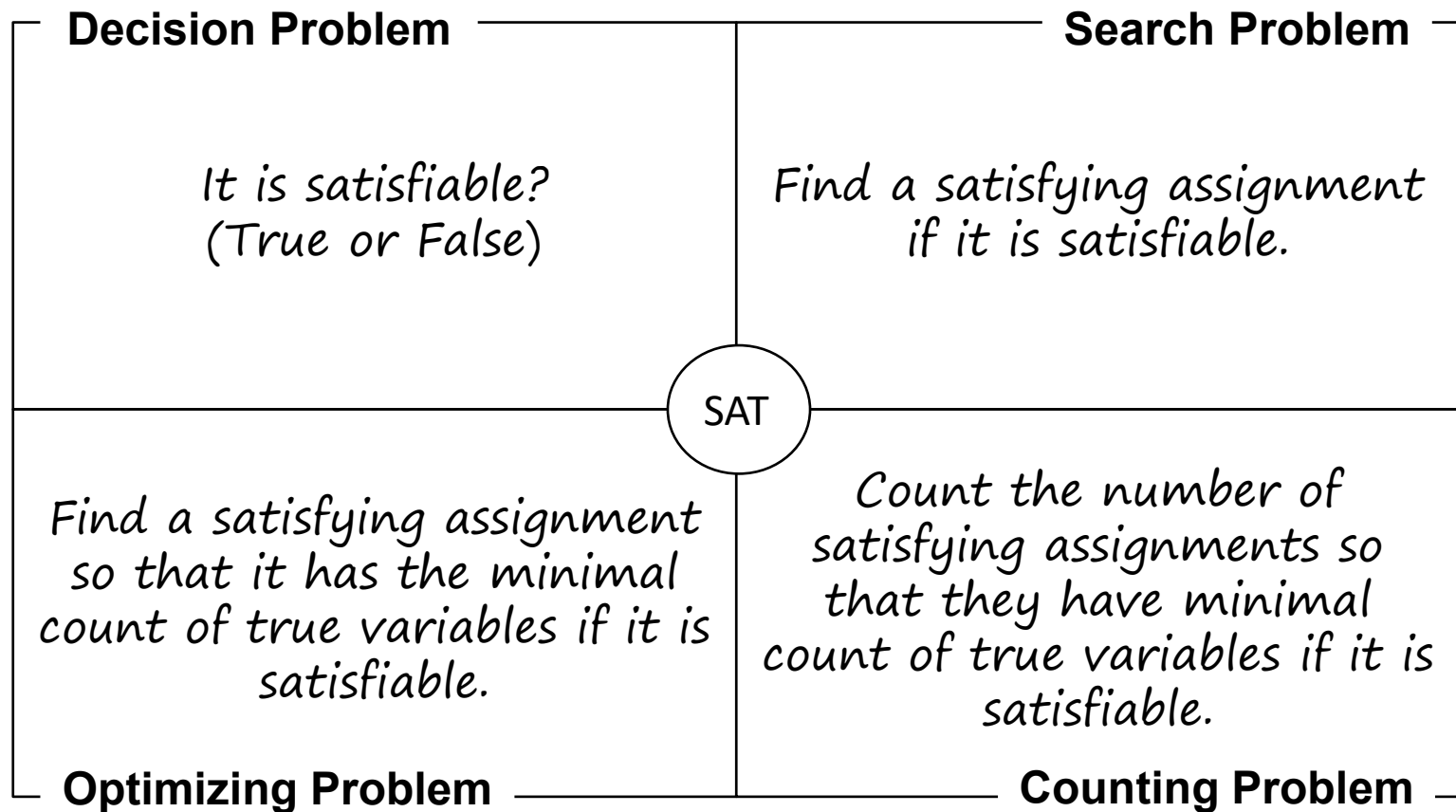
$$\overline{A} \cup \overline{B}$$



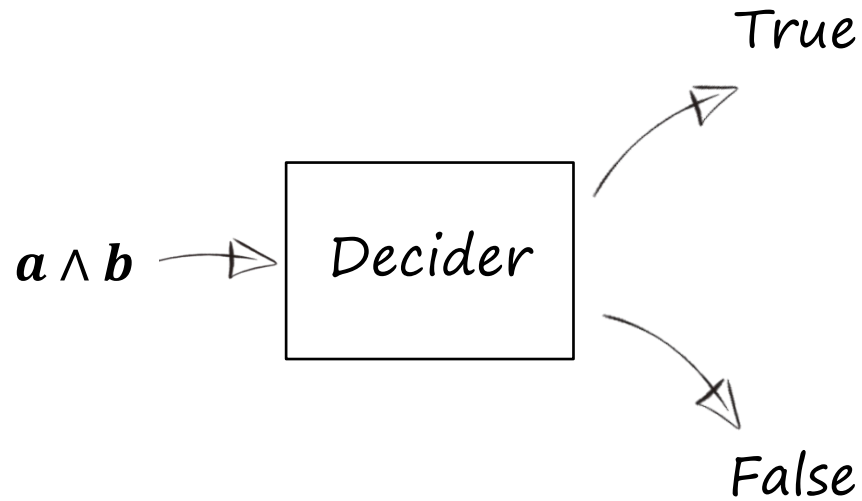
This is not a proof!!

Wait a minute,
What is the relationship between set theory and
computational problems?

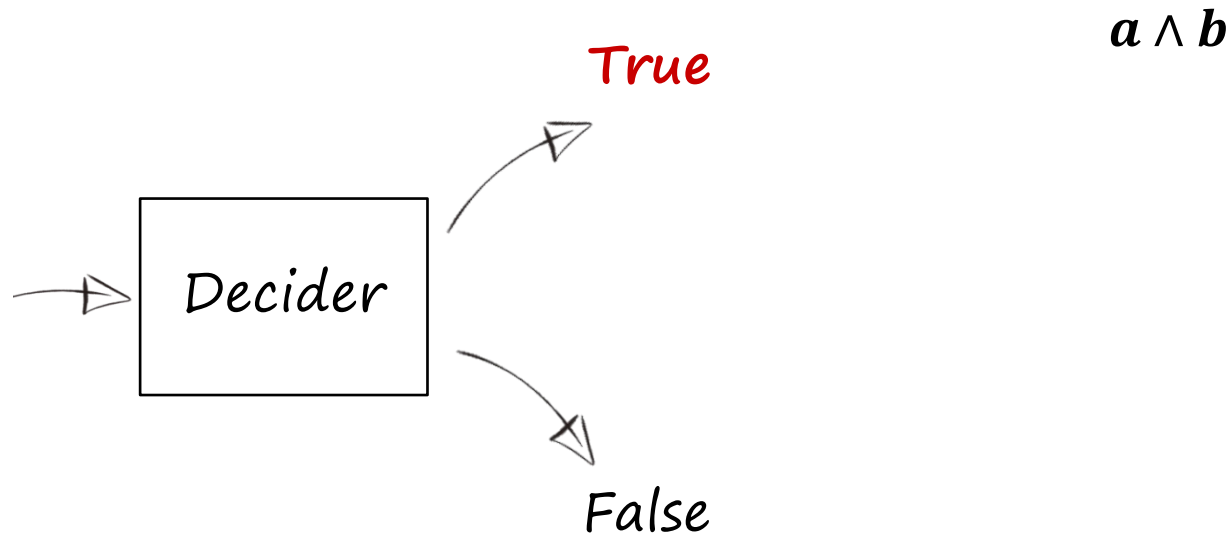
Types of Computational Problem



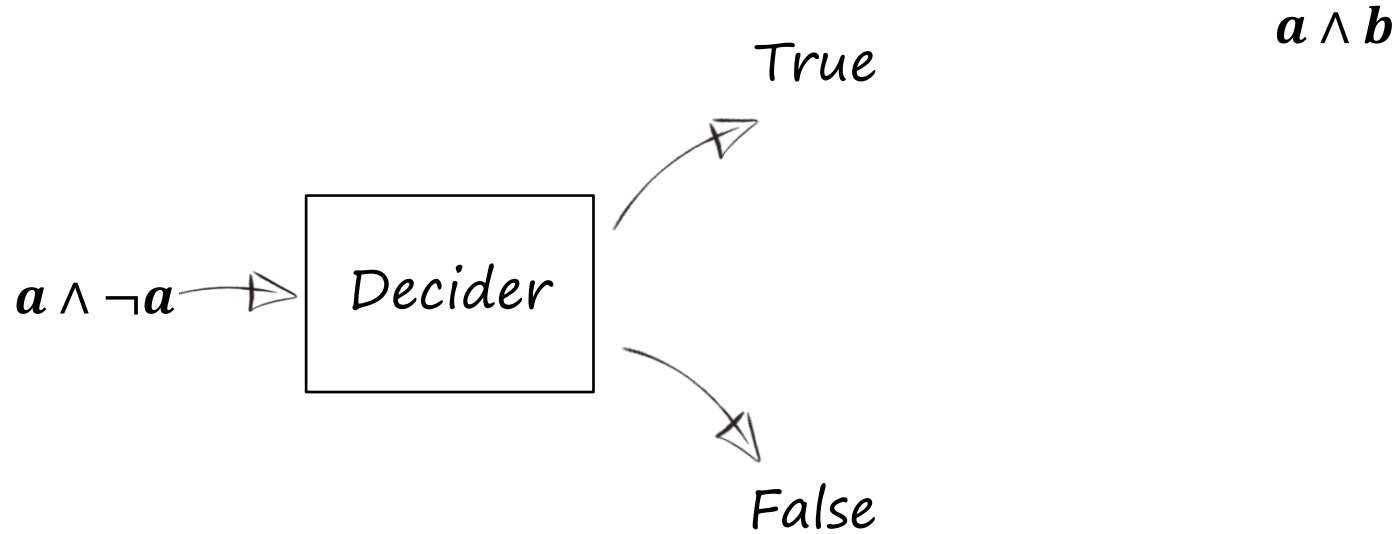
Solve Decision Problem



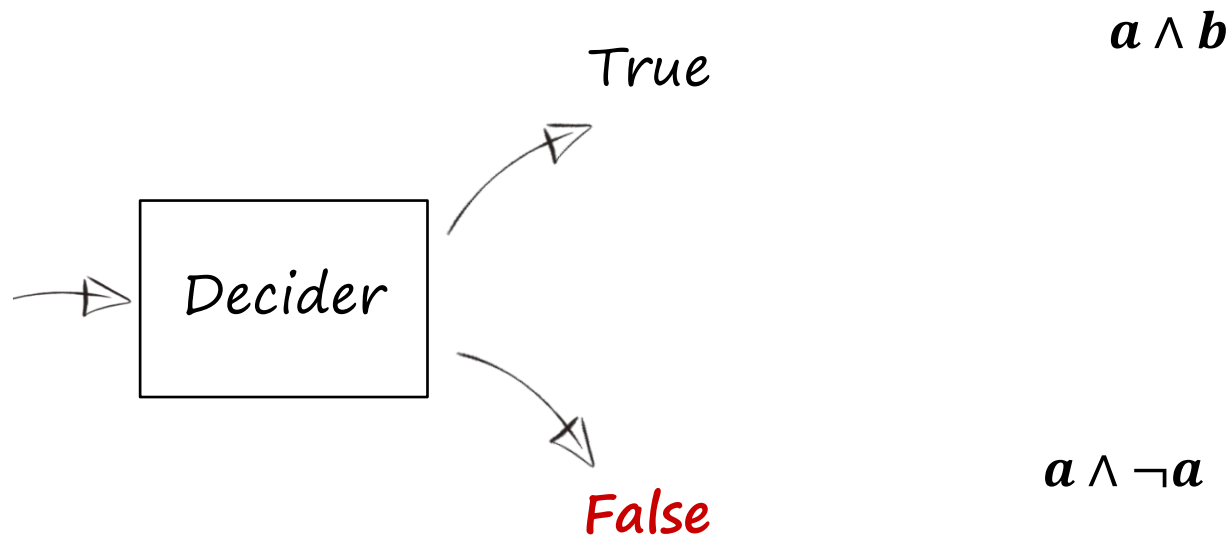
Solve Decision Problem



Solve Decision Problem



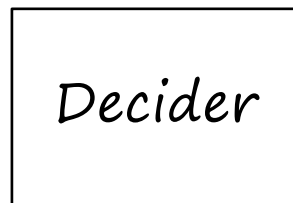
Solve Decision Problem



Solve Other Problems

Decision Problems

$a \wedge b$



True

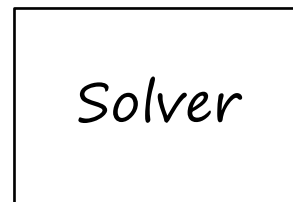


False

Results are only true or false.

Other Problems

$1+2$

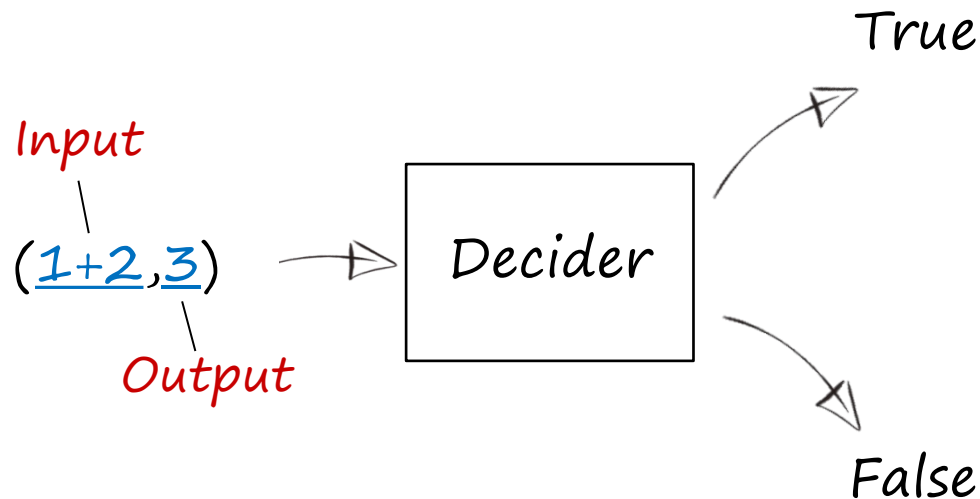


3

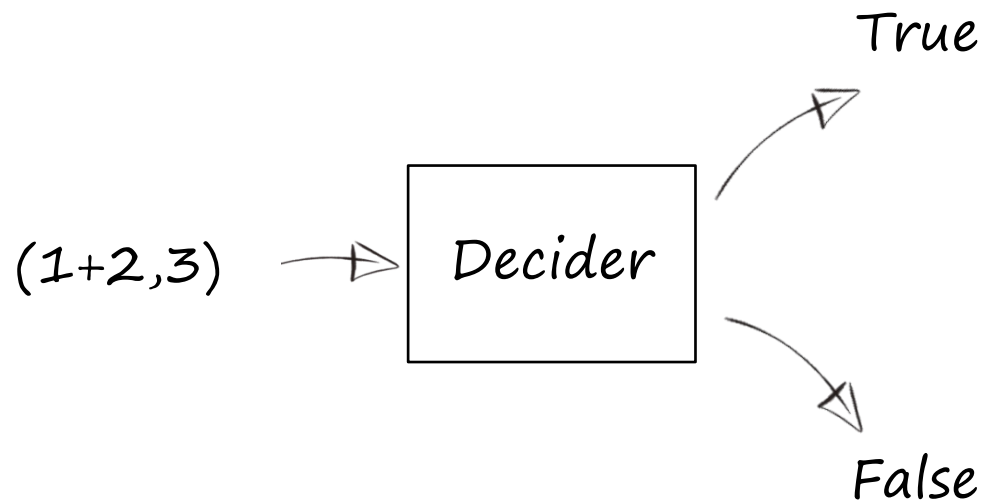
Results are not only true or false.

Solve Other Problems

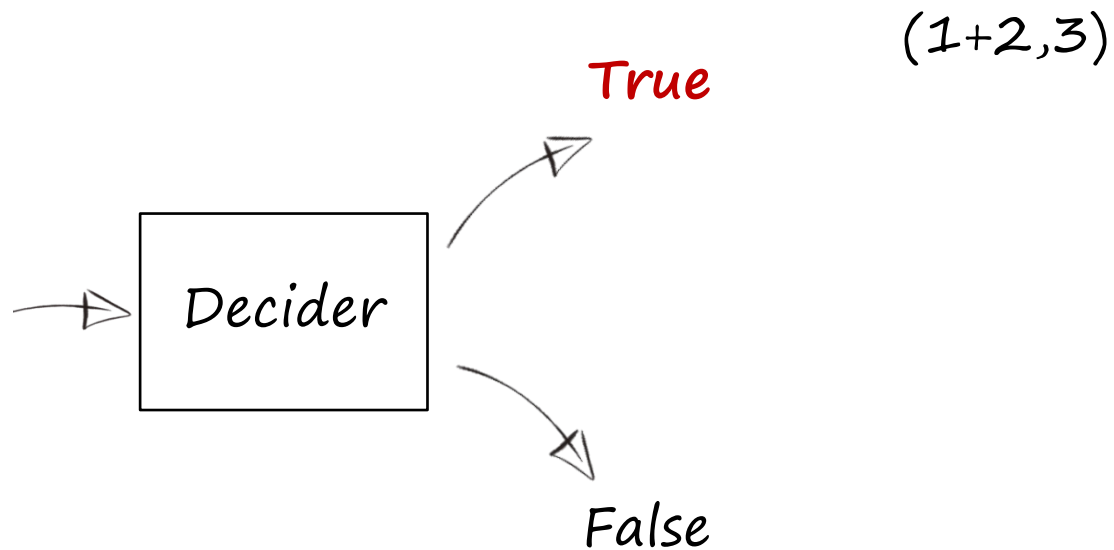
We can build a decider to judge whether an input/output pair is correct or not



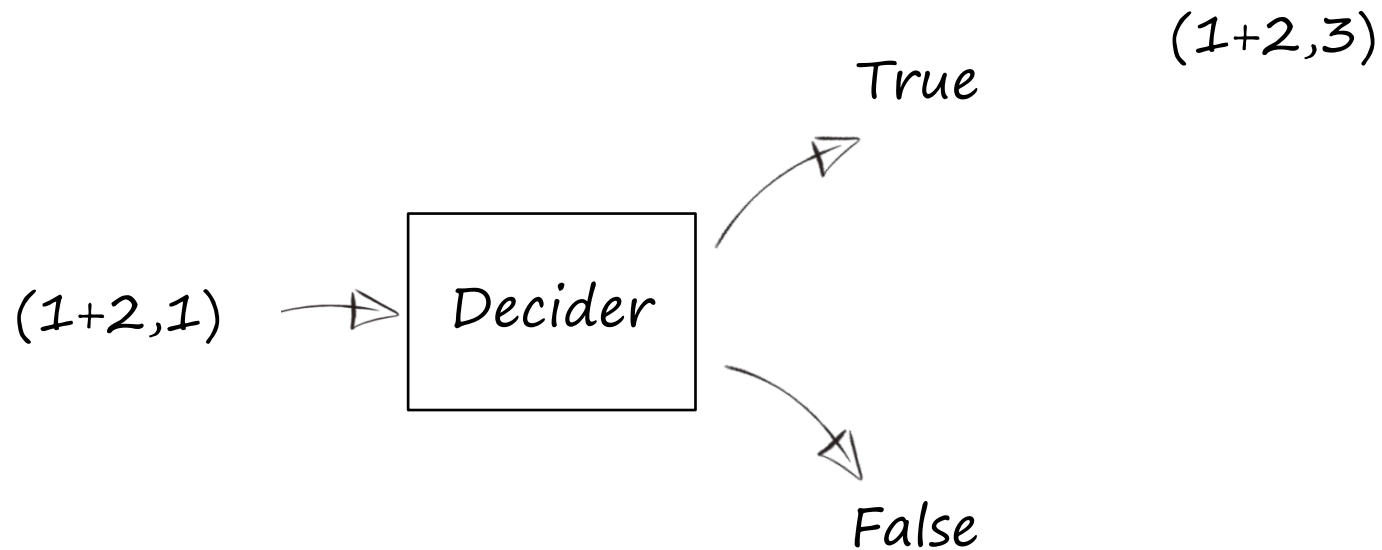
Solve Other Problems



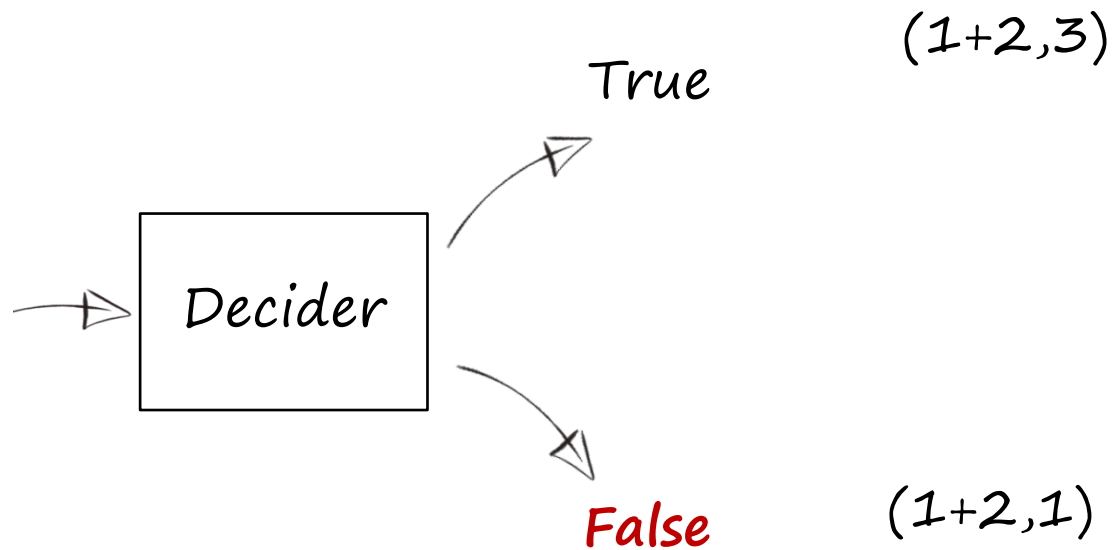
Solve Other Problems



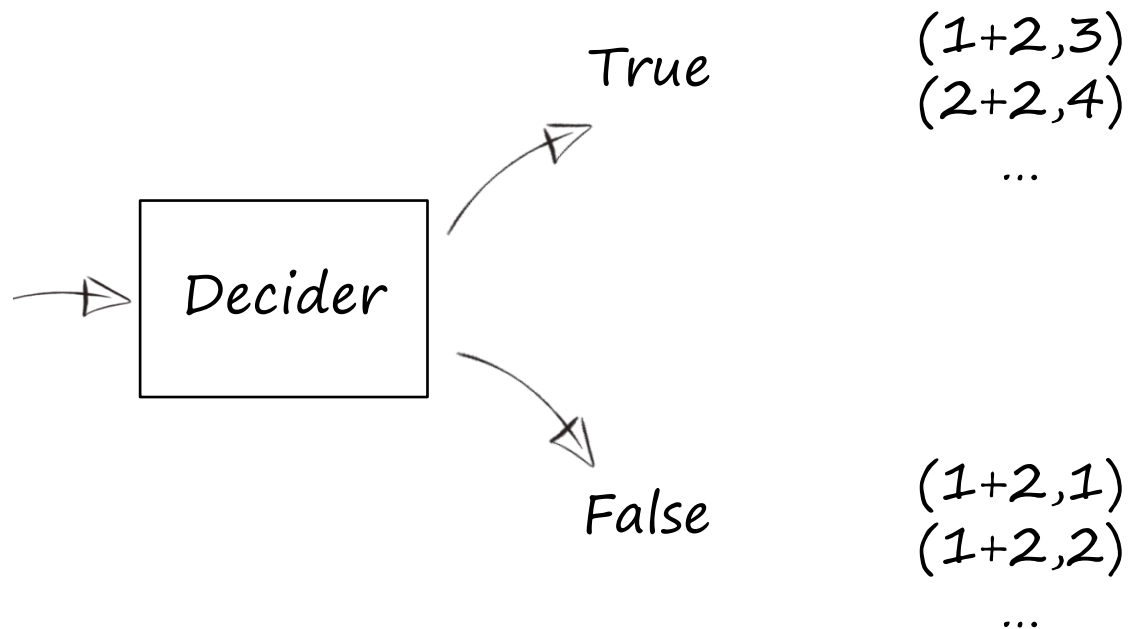
Solve Other Problems



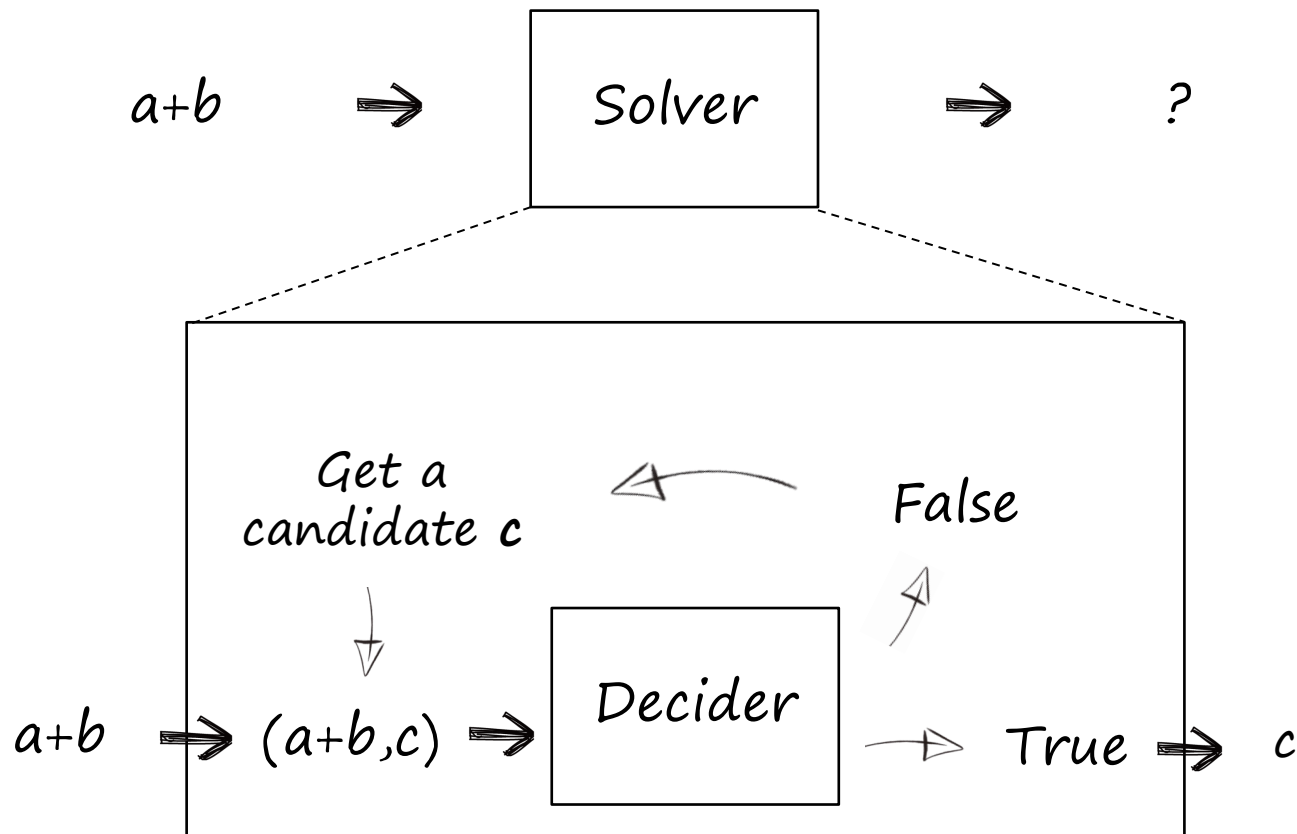
Solve Other Problems



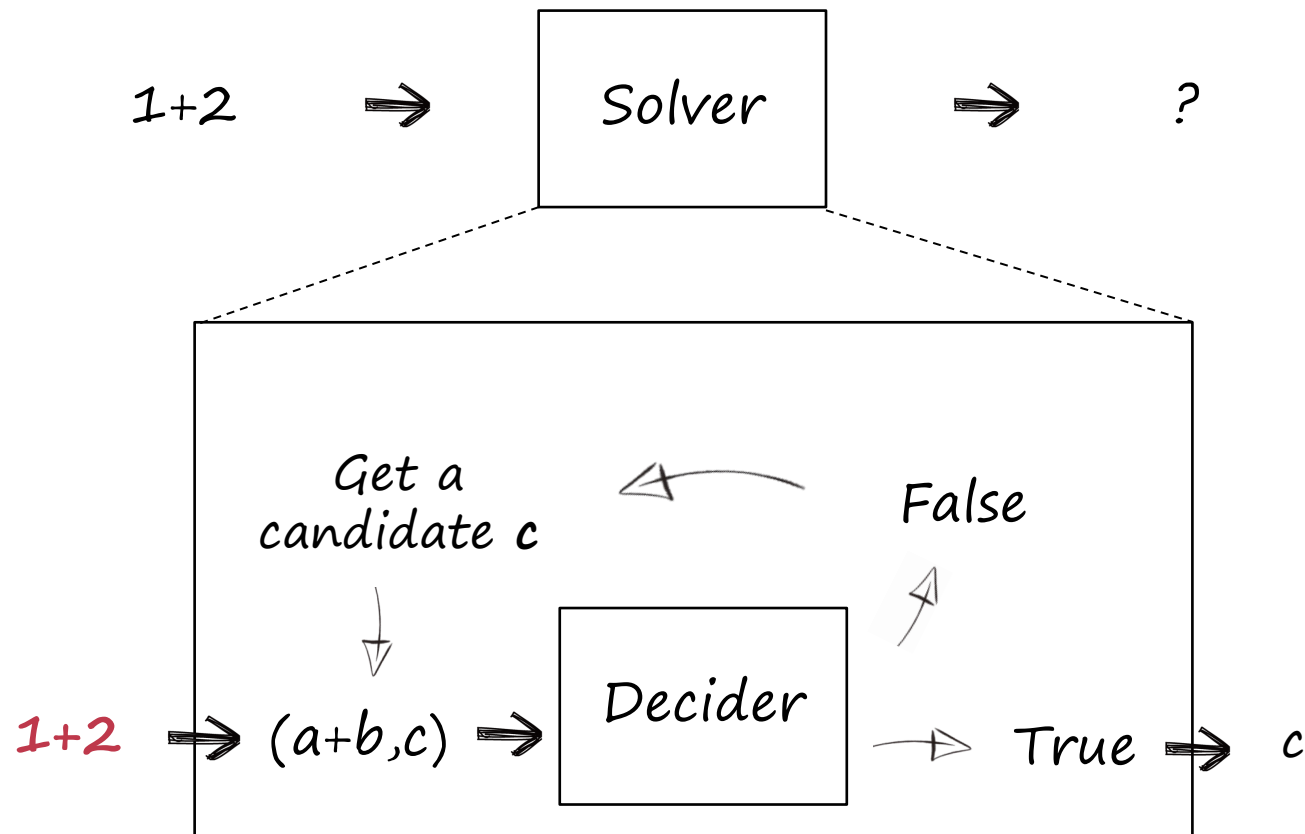
Solve Other Problems



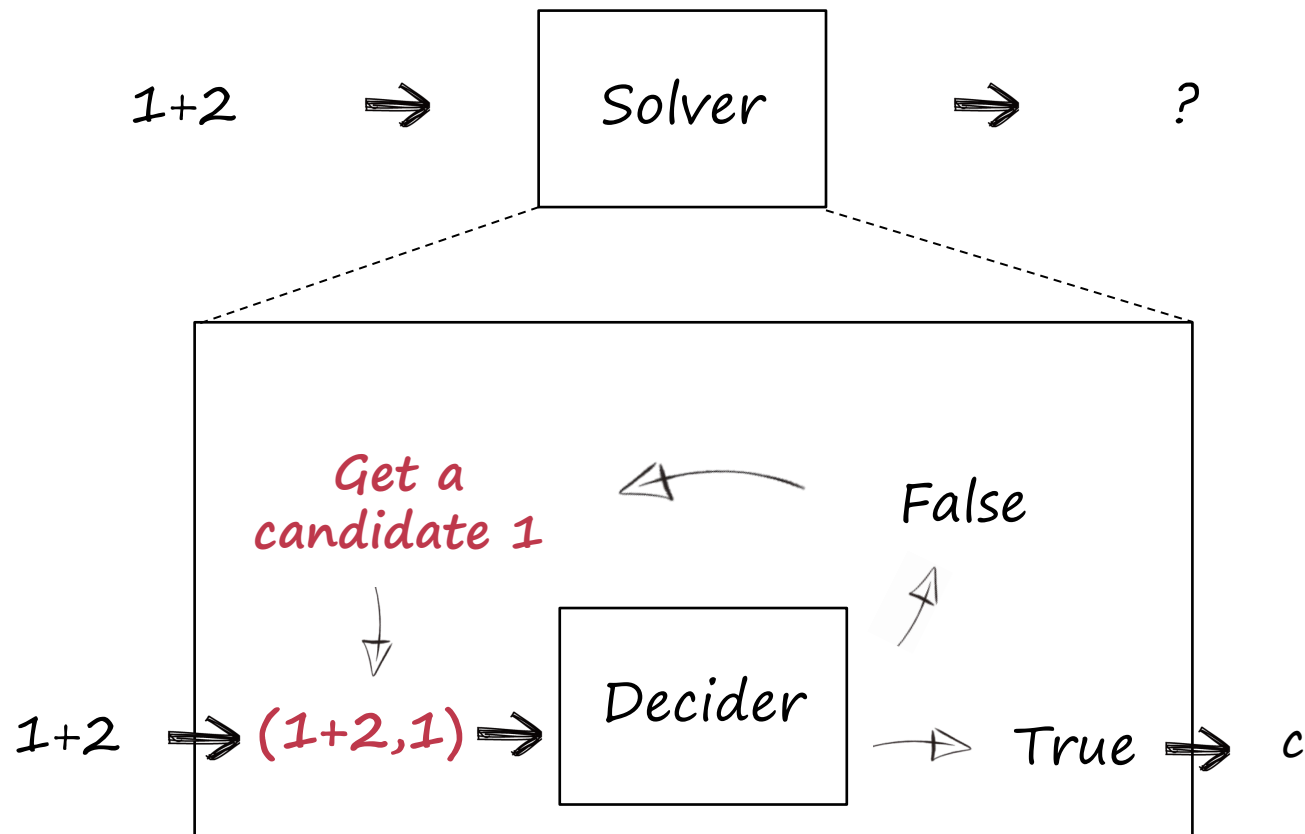
Solve Other Problems



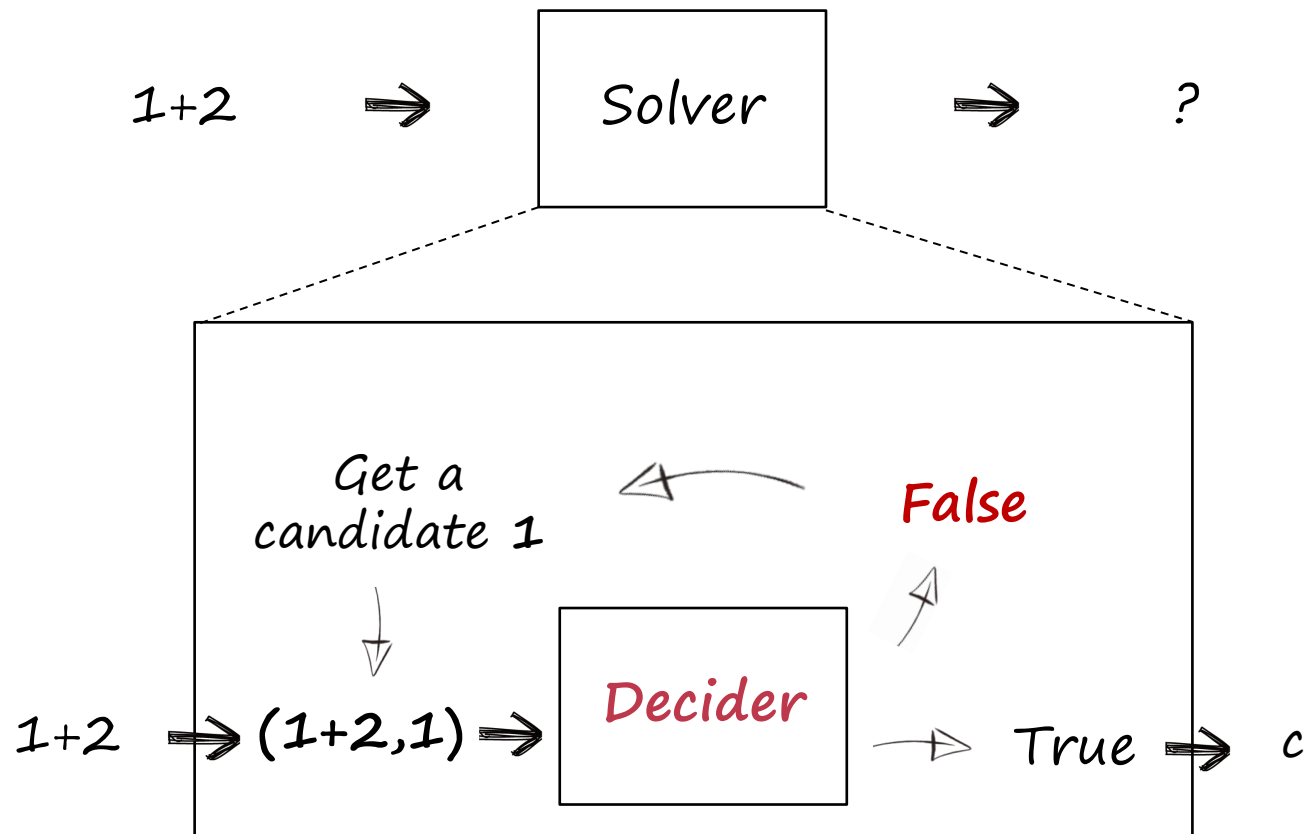
Solve Other Problems



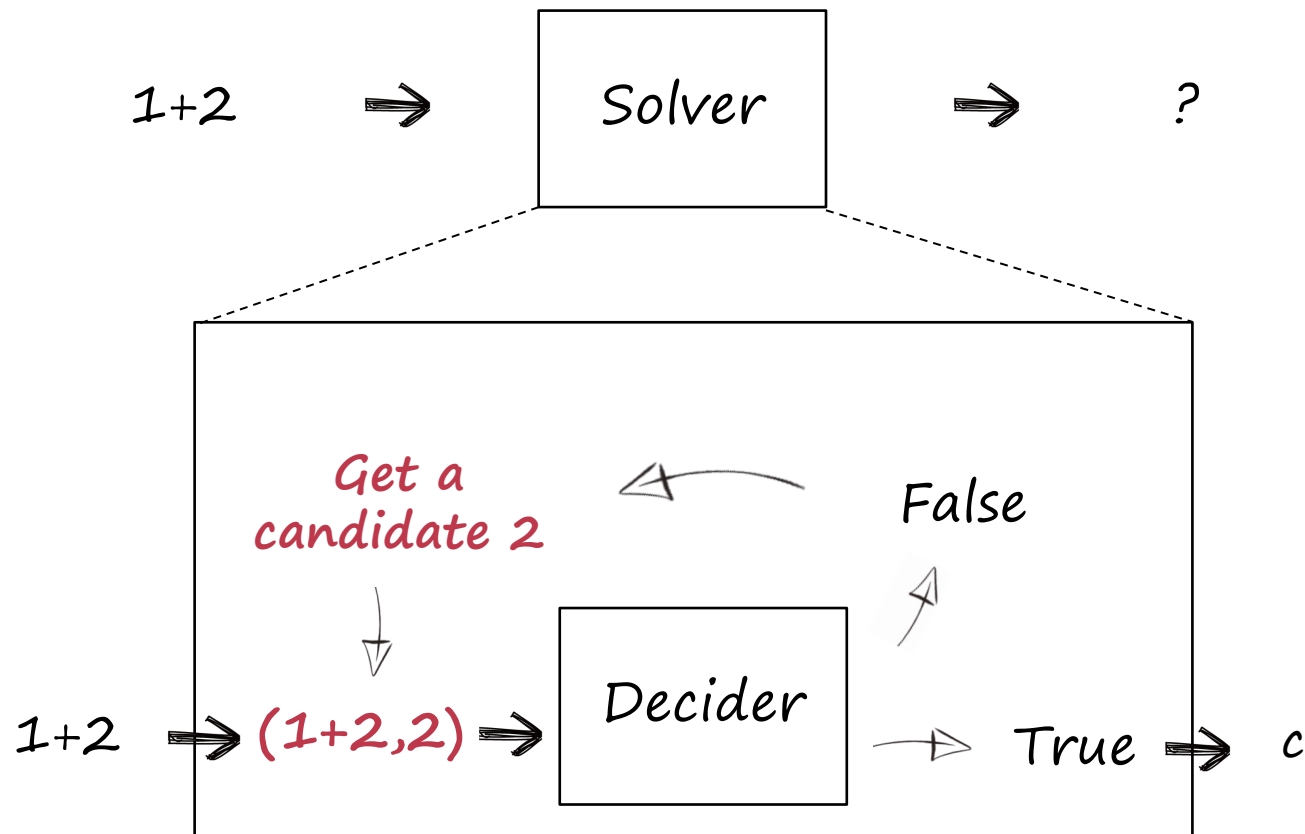
Solve Other Problems



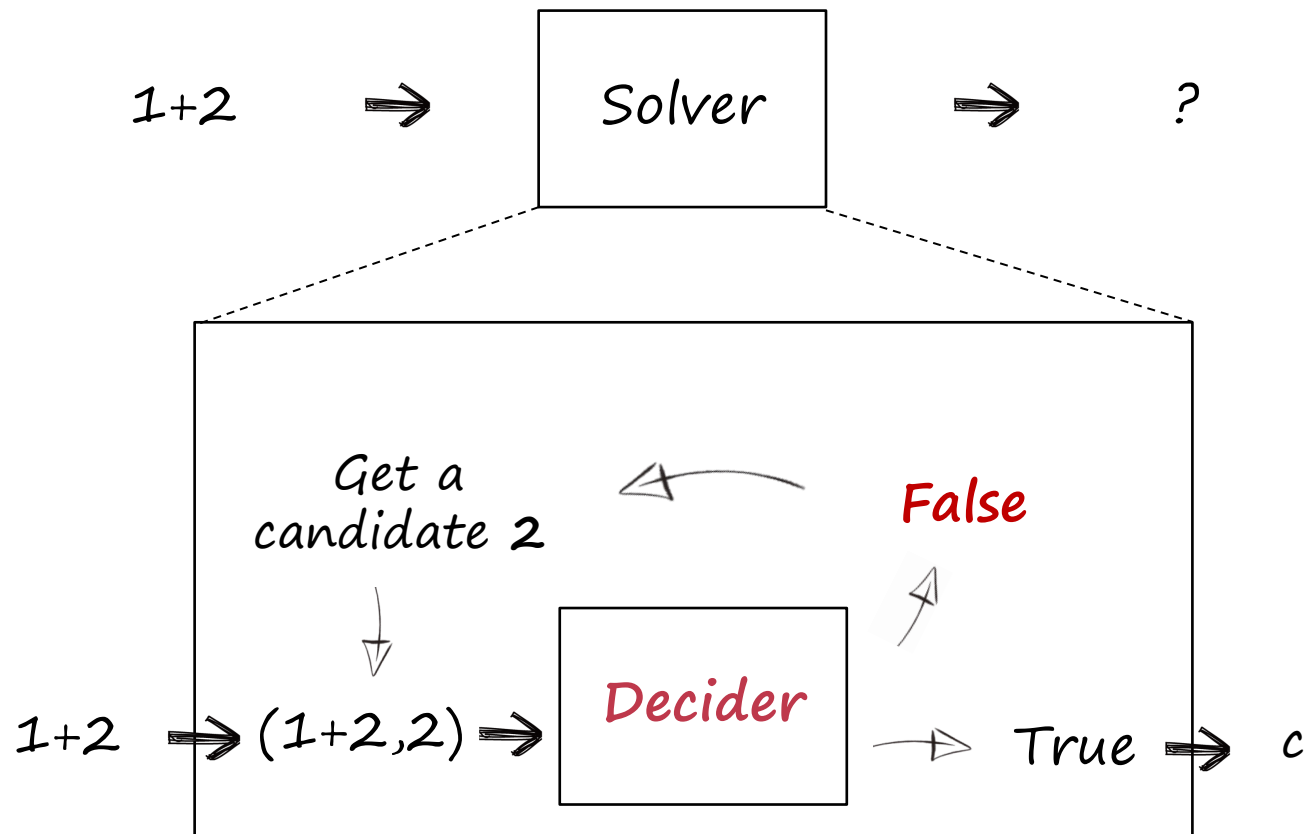
Solve Other Problems



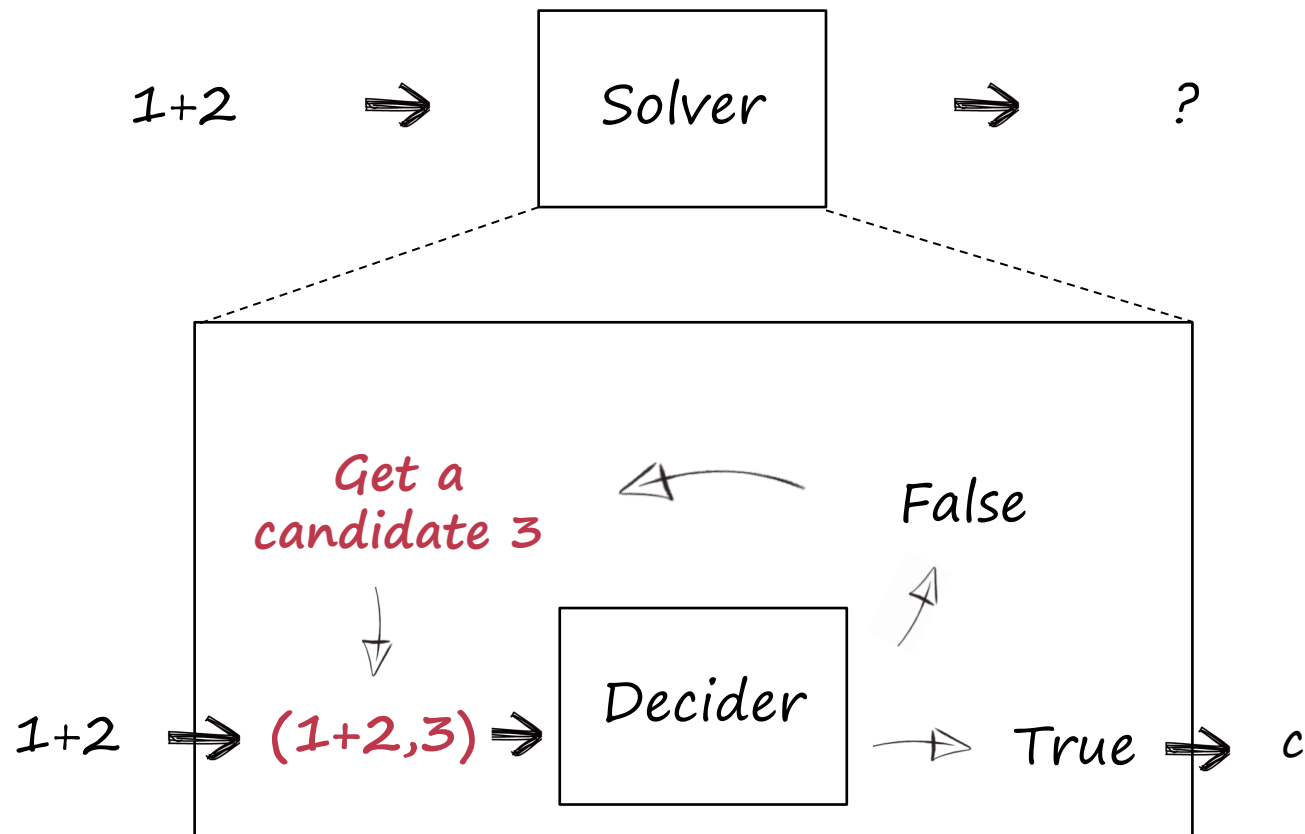
Solve Other Problems



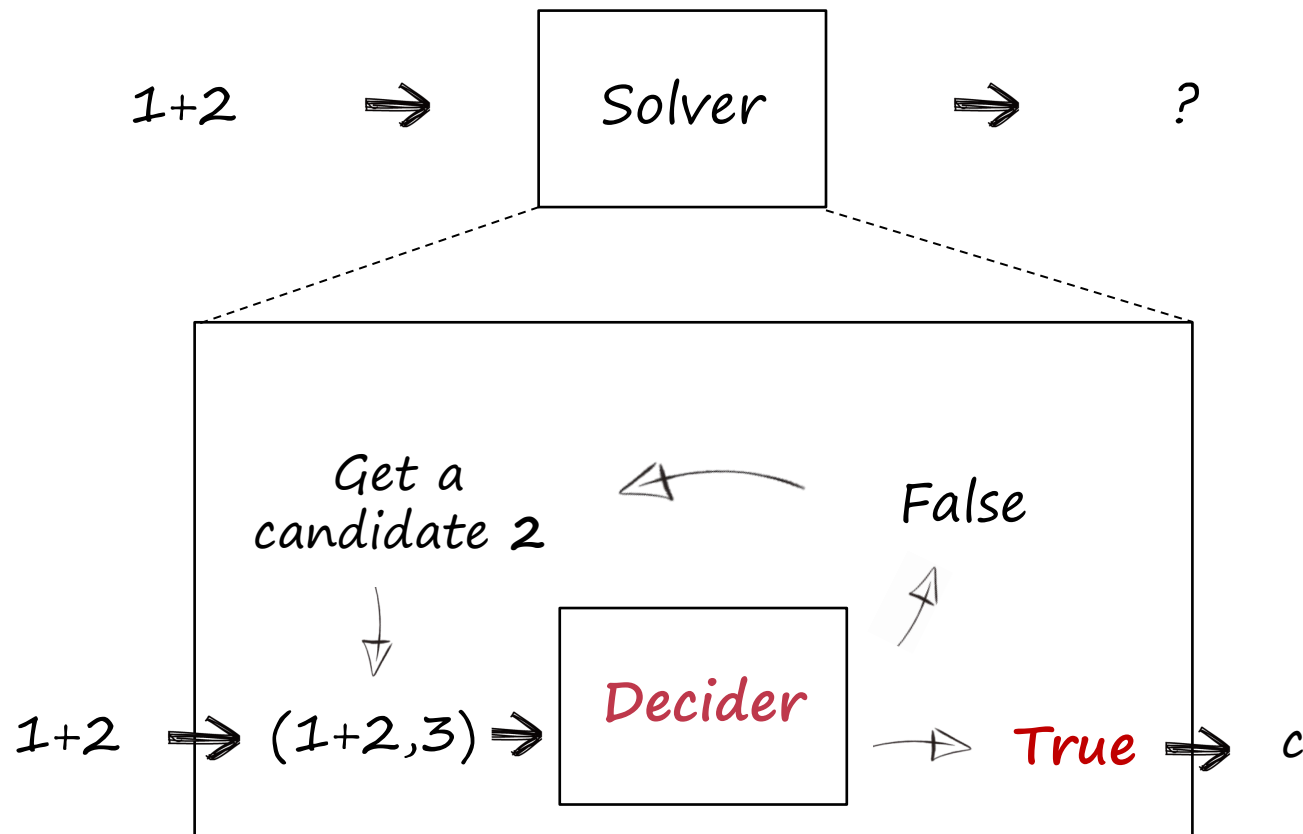
Solve Other Problems



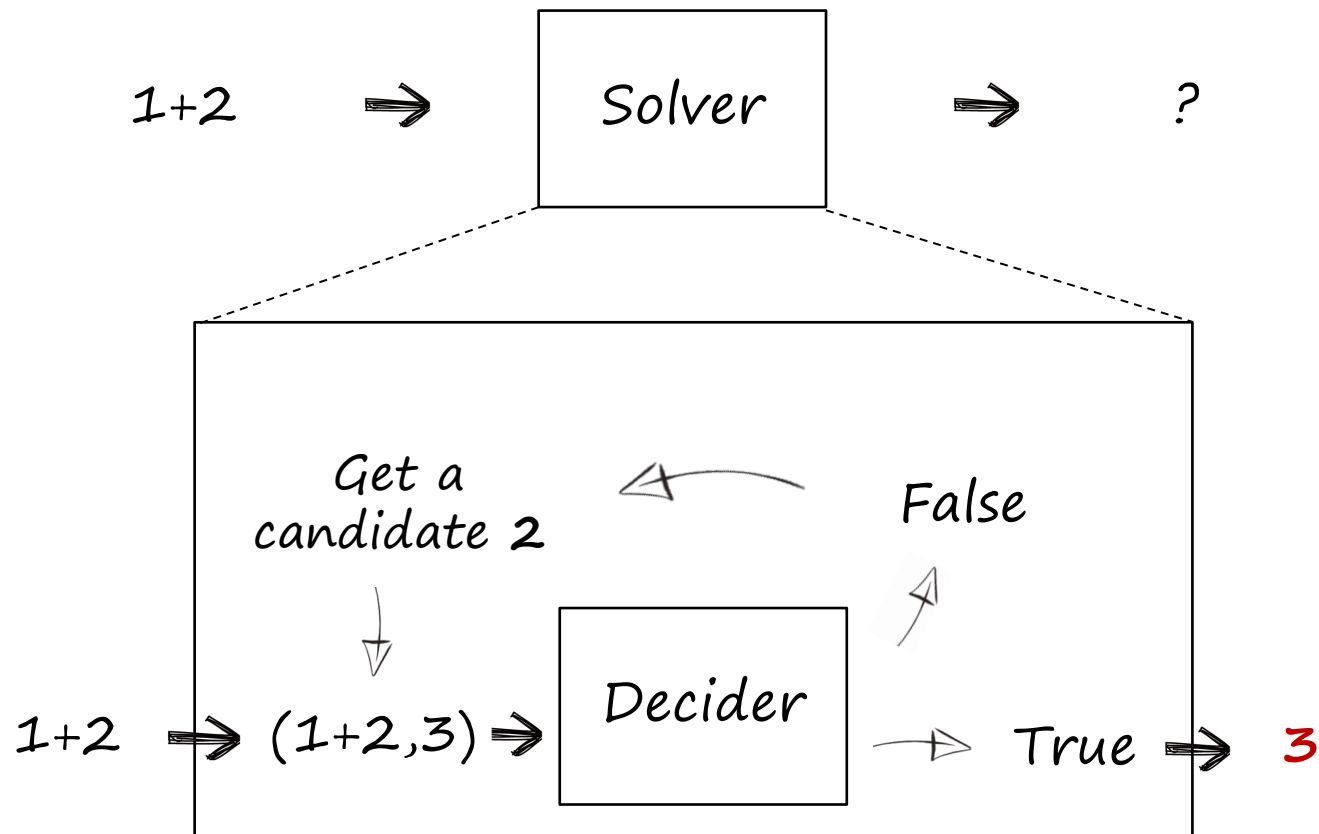
Solve Other Problems



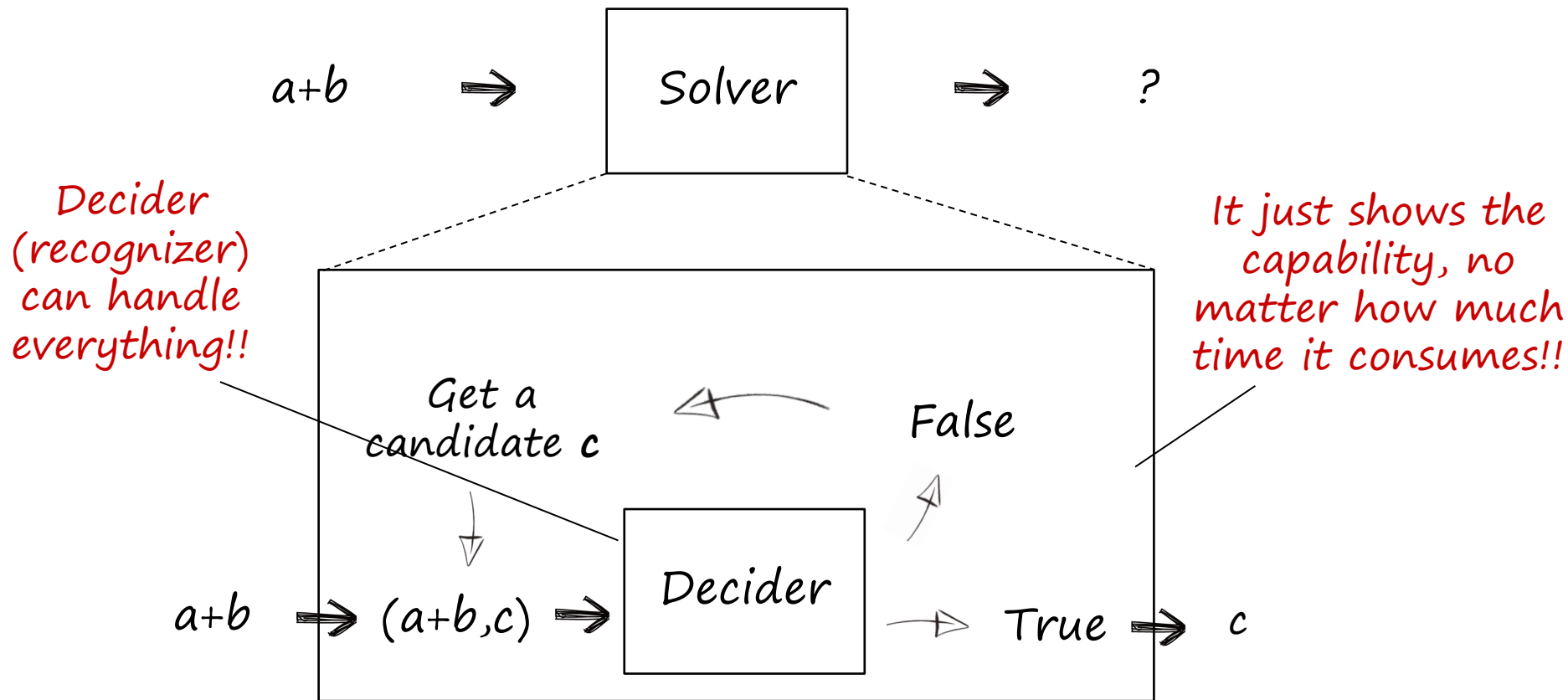
Solve Other Problems



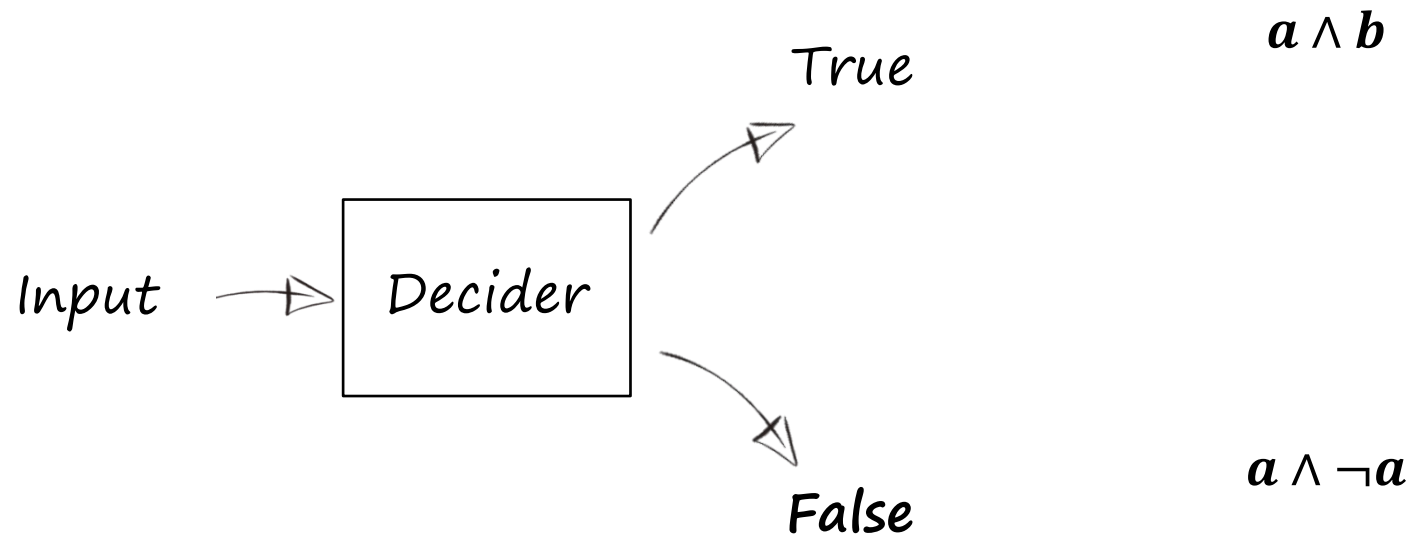
Solve Other Problems



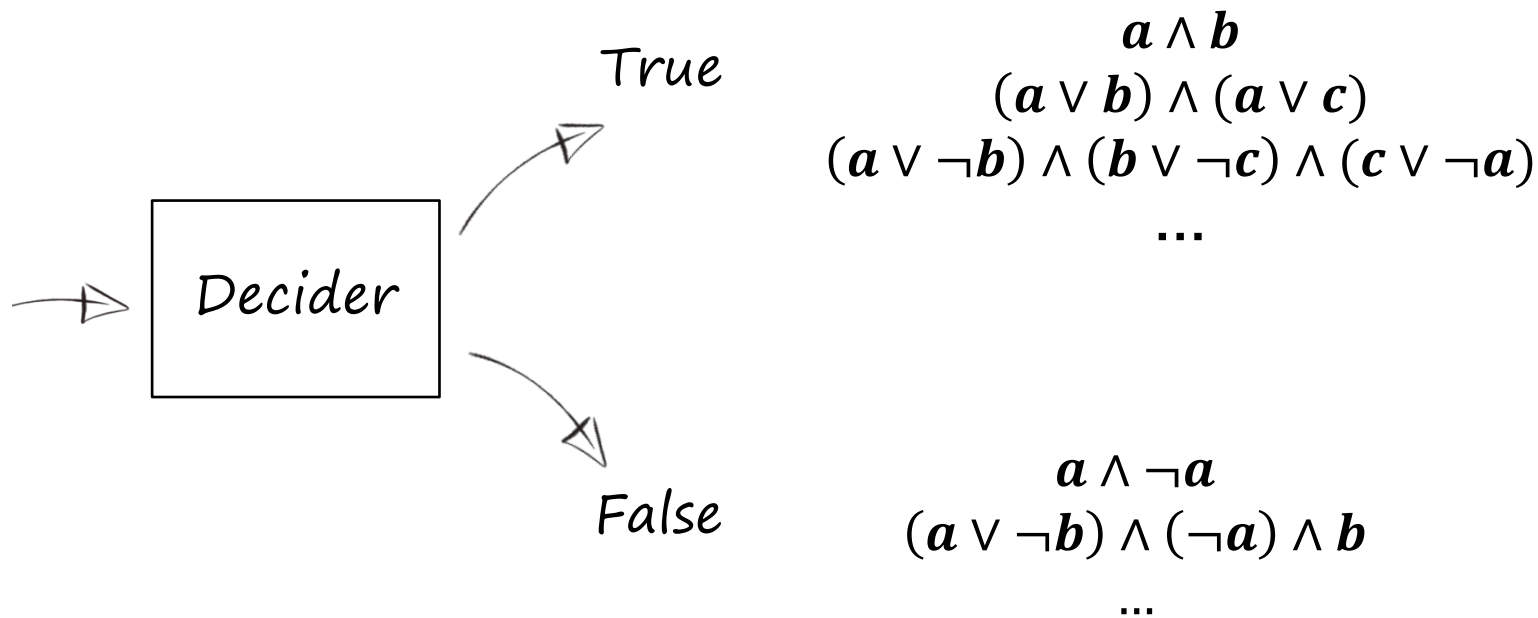
Solve Other Problems



Solve Decision Problem



Solve Decision Problem



Solve Decision Problem

Imaging that there is a god who knows all the Boolean formulas which are satisfiable, and he puts them all into a set

Set of all the Boolean formulas which are satisfiable, mark as S



$$a \wedge b$$

 a b

$$(a \vee b) \wedge (a \vee c)$$

$$(a \vee \neg b) \wedge (b \vee \neg c) \wedge (c \vee \neg a)$$

 $\dots$

$$a \wedge \neg a$$

Solve Decision Problem

A Boolean formula t is
satisfiable?



Is t in S ?



$a \wedge b$

a $(a \vee b) \wedge (a \vee c)$
 b
 $(a \vee \neg b) \wedge (b \vee \neg c) \wedge (c \vee \neg a)$

...

S

$a \wedge \neg a$

Solve Decision Problem

For any decision problem, we can define a predicate P :

$P(x)$: (x is a valid input) $\wedge$ (x satisfies the property the problem asks)

Then we can build the set:

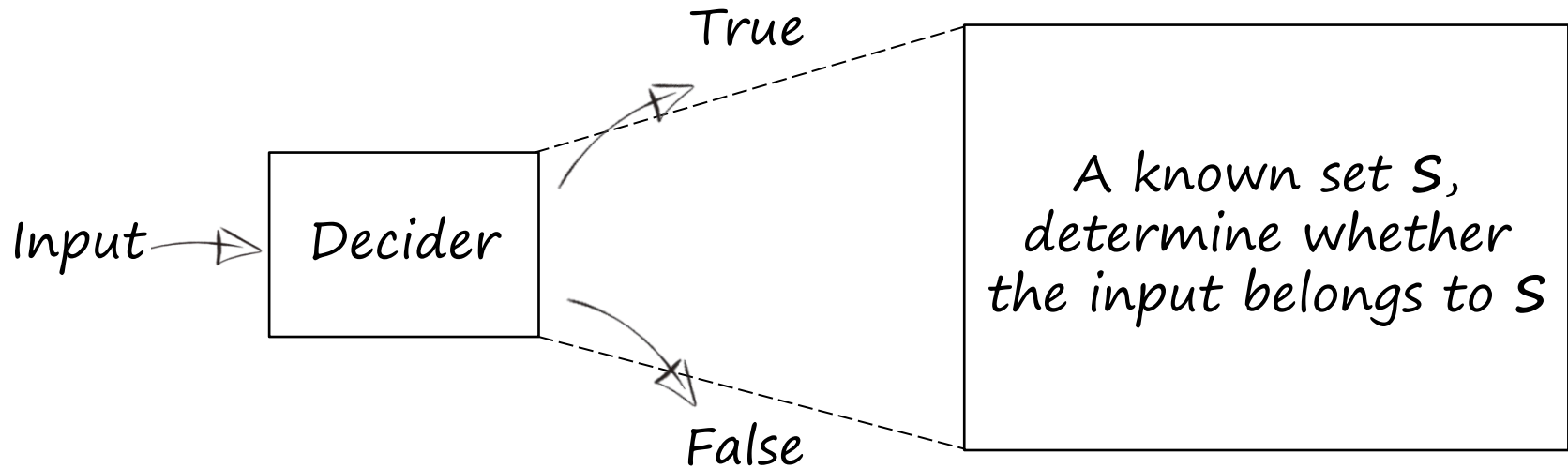
$$S = \{x | P(x)\}$$

Then the decision problem can be converted to:

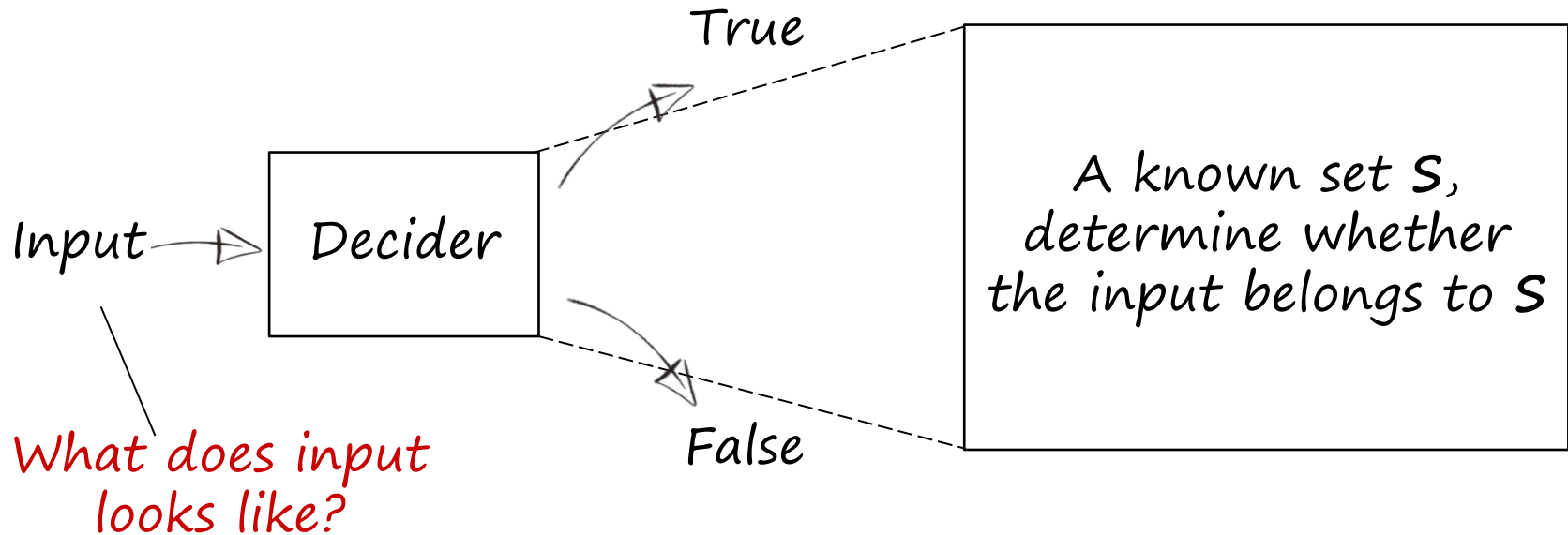
Take x as input and decide whether $x \in S$, that S is a known set.

This is called membership problem.

Solve Decision Problem



Solve Decision Problem

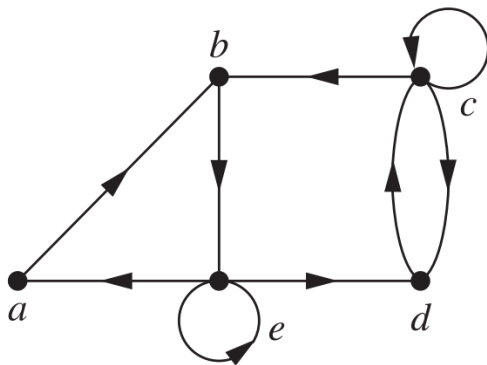


Input has large varieties

Structural
Data

```
{  
  "firstName": "Duke",  
  "lastName": "Java",  
  "streetAddress": "100 Internet Dr",  
  "phoneNumbers": [  
    { "Mobile": "111-111-1111" },  
    { "Home": "222-222-2222" }  
  ]  
}
```

Graph



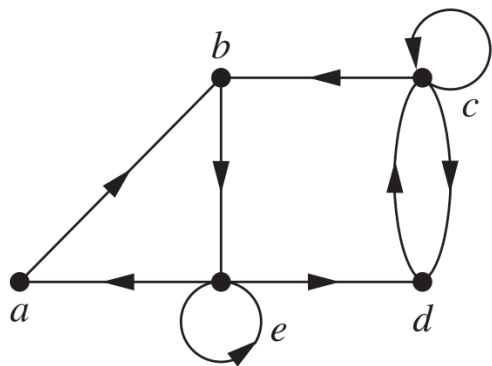
Sound



Image

But every input can be encoded as a string

Encode to string



$(a|b), (b|e), (c|b, c, d), (d|c), (e|a, d, e)$

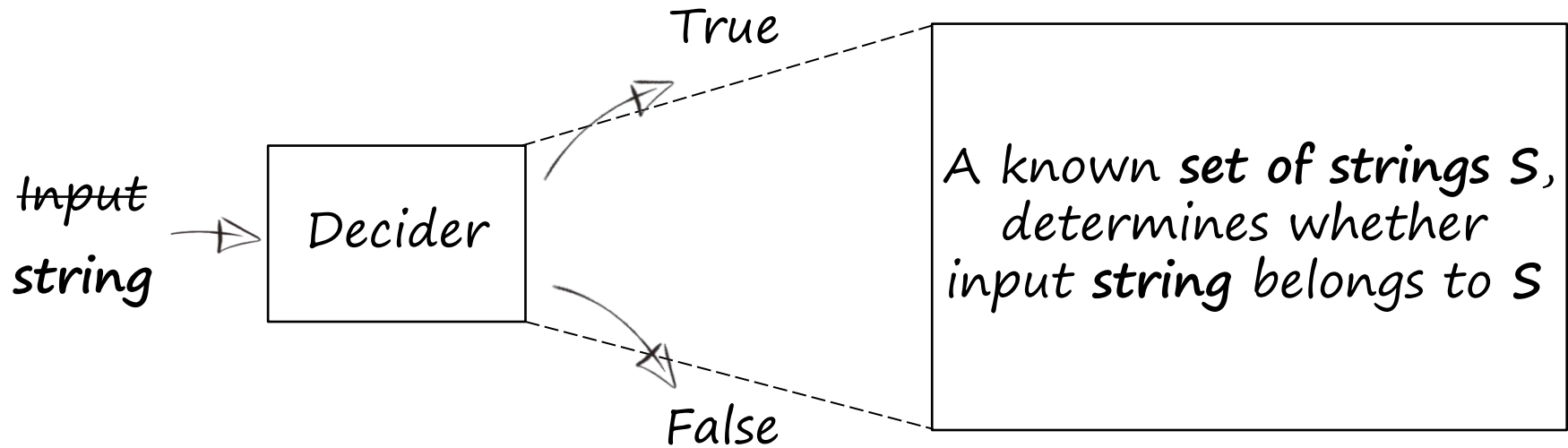


	118	118	120	120	119
	124	125	126	125	126
223	224	225	227	228	5
224	225	226	225	224	3
224	226	226	224	223	1
226	227	225	223	224	2
225	225	223	219	221	7



$(5 \times 5), R: 223, 224, 225, 227,$
 $228, 224, 225, 226, 225, 224,$
 $224, 226, 226, 224, 223, 226,$
 $227, 225, 223, 224, 225, 225,$
 $223, 219, 221, G: 134, 135,$
 $136, 135, 136, 132, 134, 135,$
 $134, 133, 132, 134, 135, 133,$
 $131, 133, 134, 132, 130, 132,$
 $132, 131, 129, 125, 127,$
 $B: 119, 119, 120, 120, 119,$
 $116, 116, 117, 116, 115, 114,$
 $116, 117, 115, 112, 115, 116,$
 $113, 111, 111, 113, 113, 108,$
 $104, 106$

Solve Decision Problem



Formal Language Theory

Alphabets are finite, non-empty sets of characters

Alphabet: Σ

$a, b, c \dots$
 $\wedge \vee \neg ()$

Characters are individual symbols

finite sequence

A string

$(a \vee b) \wedge (\neg b \vee c)$

Strings are finite sequence of characters

All strings

a, b, c
 $(a \vee b) \wedge (\neg b \vee c)$
 $\dots$

subset of Σ^*

A language

Languages are sets of strings.

$\vee \wedge abc$
 $a \wedge \neg a$
 $(a \vee b) \wedge (\neg b \vee c)$
 $\dots$

Set of all string : Σ^*

Language Recognizing

Throughout the whole course of computability, we'll focus on the membership problem of a language, or language recognizing, that is:

Given a language L and take w as input, determine whether $w \in L$.

~~What problems can a computer solve?~~



What languages can a computer recognize?

Next

The intuitive way to construct a set may lead to paradox.

- Russell's Paradox & ZF Axiom System

What's the relationship between the number of problems and the number of programs?

- Cardinality of a Set and Cantor's Theorem