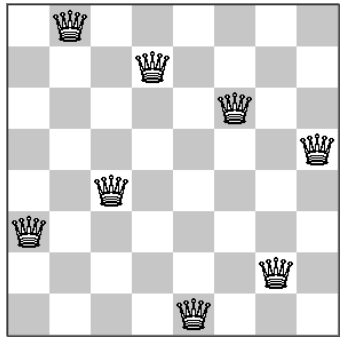
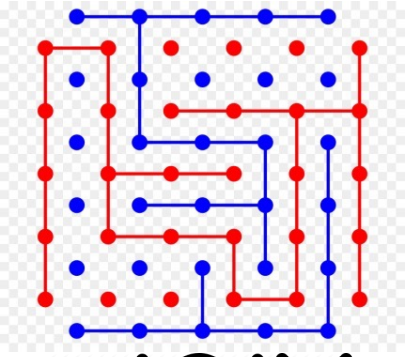
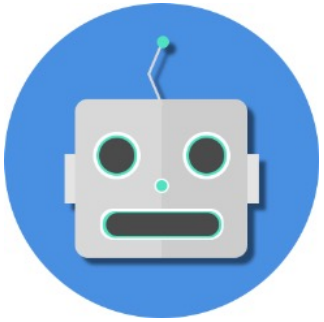
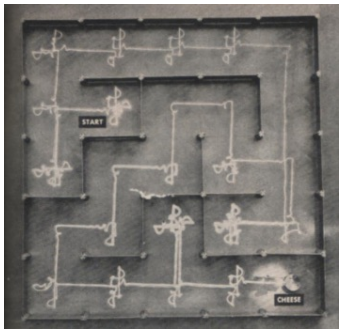
Artificial Intelligence:

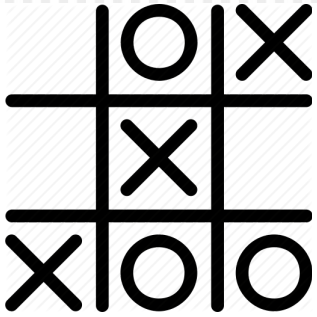
Past, Present and Future











Chee Wei Tan

Mathematical Logic in AI

John McCarthy

Mathematical Logic in Artificial Intelligence

THIS ARTICLE concerns computer programs that represent information about their problem domains in mathematical logical languages and use logical inference to decide what actions are appropriate to achieve their goals.

Mathematical logic is not a single language. There are many kinds of mathematical logic, and even choosing a kind does not specify the language. The language is determined by declaring what nonlogical symbols will be used and what sentences will be taken as axioms. The nonlogical symbols are those that concern the concrete subject matter to be stored in a computer's data base—for example, information about objects and their locations and motions.

Whatever the choice of symbols, all kinds of mathematical logic share two ideas. First, it must be mathematically definite what strings of symbols are considered formulas of the logic. Second, it must be mathematically definite what inferences of new formulas from old ones are allowed. These ideas permit the writing of computer programs that decide what combinations of symbols are sentences and what inferences are allowed in a particular logical language.

Mathematical logic has become an important branch of mathematics, and most logicians work on problems arising from the internal development of the subject. Mathematical logic has also been applied to studying the foundations of mathematics, and there it has had its greatest success. Its founders, Aristotle, Leibniz, Boole, and

John McCarthy is professor of computer science and Charles M. Pigott Professor of Engineering at Stanford University.

Vol. 117, No. 1, Artificial Intelligence (Winter, 1988), pp. 297-311

Frege, also wished to apply it to making reasoning about human affairs more rigorous. Indeed, Leibniz was explicit about his goal of replacing argument with calculation. However, expressing knowledge and reasoning about the commonsense world in mathematical logic has entailed difficulties that seem to require extensions of the basic concepts of logic, and these extensions are only beginning to develop.

If a computer is to store facts about the world and reason with them, it needs a precise language. The program must be based on a precise idea of what reasoning is allowed—that is, how new formulas may be derived from old. It was natural in the beginning to try to use mathematical logical language to express what an intelligent computer program "knows" that is relevant to the problems we want it to solve and to make the program use logical inference in order to decide what to do. The first proposal to use logic in artificial intelligence for expressing what a program knows and how it should reason was in a paper I wrote in 1960. The problem of proving logical formulas as a domain for AI had already been studied. In this paper I said:

The *advice taker* is a proposed program for solving problems by manipulating sentences in formal languages. The main difference between it and other programs or proposed programs for manipulating formal languages (the *Logic Theory Machine* of Newell, Simon and Shaw and the Geometry Program of Herbert Gelernter) is that in the previous programs the formal system was the subject matter but the heuristics were all embodied in the program. In this program the procedures will be described as much as possible in the language itself and, in particular, the heuristics are all so described.

The main advantage we expect the *advice taker* to have is that its behavior will be improvable merely by making statements to it, telling it about its symbolic environment and what is wanted from it. To make these statements will require little if any knowledge of the program or the previous knowledge of the *advice taker*. One will be able to assume that the *advice taker* will have available to it a fairly wide class of immediate logical consequences of anything it is told and its previous knowledge. This property is expected to have much in common with what makes us describe certain humans as having *common sense*. We shall therefore say that *a program has common sense if it automatically deduces for itself a sufficiently wide class of immediate consequences of anything it is told and what it already knows.*[1]

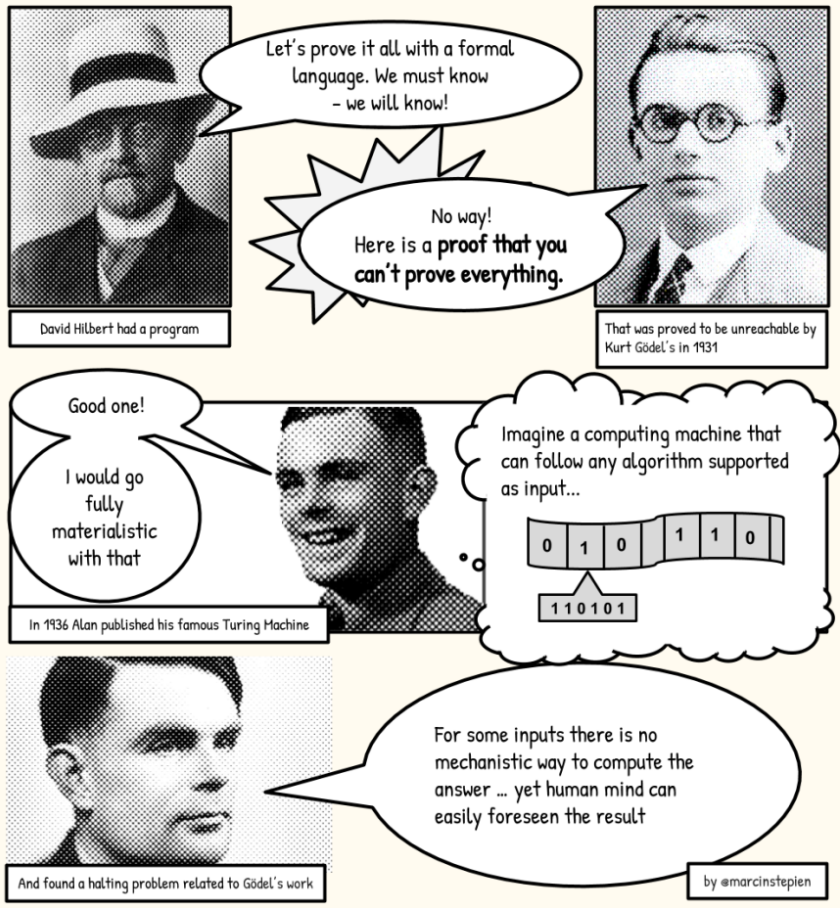
Alan Turing: Machine and Game

- Alan Turing is widely viewed as "Father of Computer Science". His 1937 Ph.D. thesis studied Hilbert's "Entscheidungsproblem" (Decision problem), in which he described a computing machine (now known as *Turing's Machine*) as a *thought experiment*: An "algorithm" can be run by a finite set of rules on this machine to compute anything that is computable.
- Turing was also instrumental in cracking the Nazi Enigma crypto-system during the Second World War. He is also known for "*Turing Test*" used in Artificial Intelligence. We will explore the *Turing Test* (The Imitation Game) after midterm.
- 2014 movie of Turing: "The Imitation Game"
- https://www.youtube.com/watch?v=nuPZUUED5uk

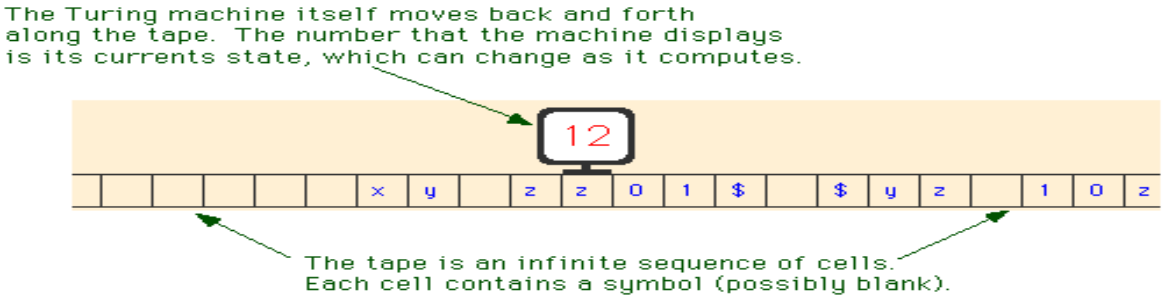
https://www.bankofengland.co.uk/banknotes/50-pound-note-nominations

What is a Turing Machine?

- 1. Imagine a computer that writes everything down in a form that is completely specified using one symbol (e.g., letter/number) at a time.
- 2. The computer follows a finite set of rules that are referred to once a symbol is written down.
- 3. Rules are stated such that at any given time only a single rule is active and hence no ambiguity can arise. Each rule activates another rule depending on what letter/number is currently read.



https://pbs.twimg.com/media/DsEy_F5WwAIF81F.png



Turing Machine: “Hello World” Program

If read 1,

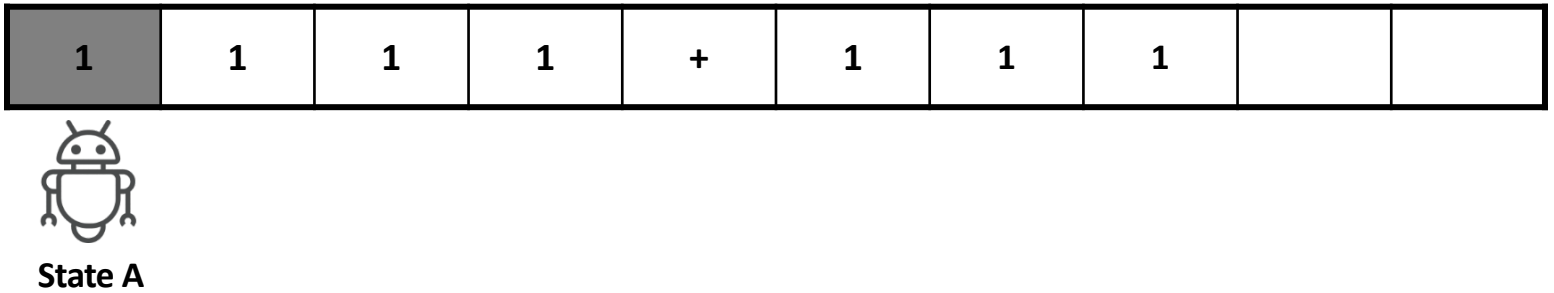
- **State A:**
  1. **Erase the 1.**
  2. **Scan next cell on right**
  3. **Go to state B**
- **State B:**
  1. **Scan next cell on right.**
  2. **Stay in State B.**

If read +,

- **State B:**
  1. **Erase the +.**
  2. **Print 1.**
  3. **Stop**

Eight cells on the paper tape are marked 111+111, signifying the addition of 4 and 3 in the “unary” system in which an integer *n* is symbolized by *n* 1’s.

Start by assuming that the machine is in State A.



M. Gardner, Can Machines Think?, Chapter 9 in Mathematical Circus, pp. 102-104

Turing Machine: “Hello World” Program

If read 1,

- **State A:**
  1. **Erase the 1.**
  2. **Scan next cell on right**
  3. **Go to state B**
- **State B:**
  1. **Scan next cell on right.**
  2. **Stay in State B.**

If read +,

- **State B:**
  1. **Erase the +.**
  2. **Print 1.**
  3. **Stop**

	1	1	1	+	1	1	1		

State A

M. Gardner, Can Machines Think?, Chapter 9 in Mathematical Circus, pp. 102-104

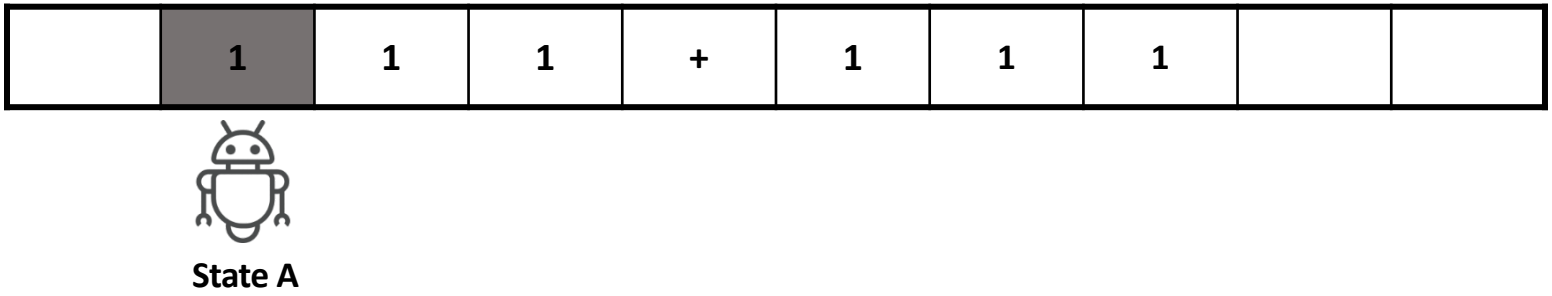
Turing Machine: “Hello World” Program

If read 1,

- **State A:**
  1. **Erase the 1.**
  2. **Scan next cell on right**
  3. **Go to state B**
- **State B:**
  1. **Scan next cell on right.**
  2. **Stay in State B.**

If read +,

- **State B:**
  1. **Erase the +.**
  2. **Print 1.**
  3. **Stop**



M. Gardner, Can Machines Think?, Chapter 9 in Mathematical Circus, pp. 102-104

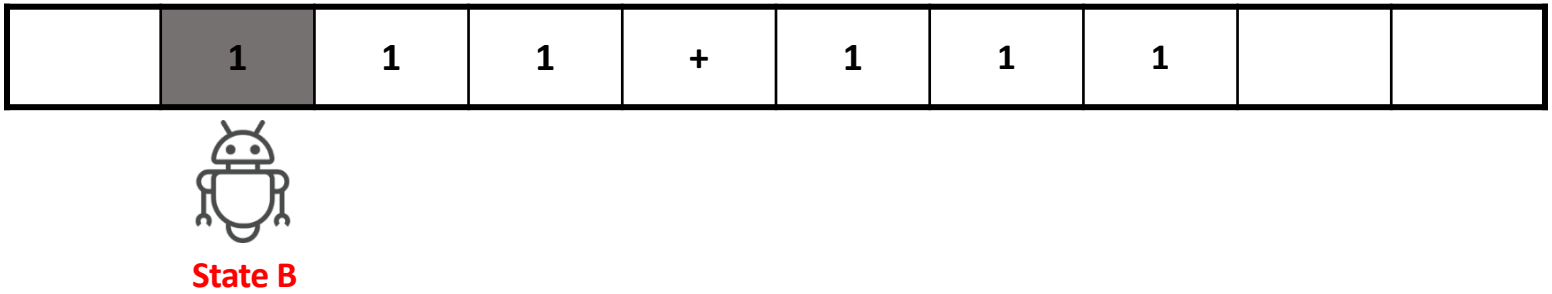
Turing Machine: “Hello World” Program

If read 1,

- **State A:**
  1. **Erase the 1.**
  2. **Scan next cell on right**
  3. **Go to state B**
- **State B:**
  1. **Scan next cell on right.**
  2. **Stay in State B.**

If read +,

- **State B:**
  1. **Erase the +.**
  2. **Print 1.**
  3. **Stop**



M. Gardner, Can Machines Think?, Chapter 9 in Mathematical Circus, pp. 102-104

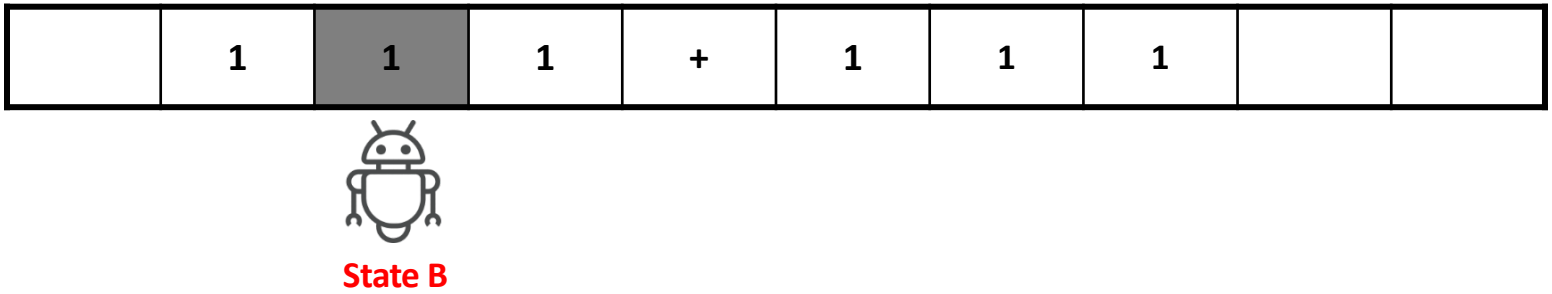
Turing Machine: “Hello World” Program

If read 1,

- **State A:**
  1. **Erase the 1.**
  2. **Scan next cell on right**
  3. **Go to state B**
- **State B:**
  1. **Scan next cell on right.**
  2. **Stay in State B.**

If read +,

- **State B:**
  1. **Erase the +.**
  2. **Print 1.**
  3. **Stop**



M. Gardner, Can Machines Think?, Chapter 9 in Mathematical Circus, pp. 102-104

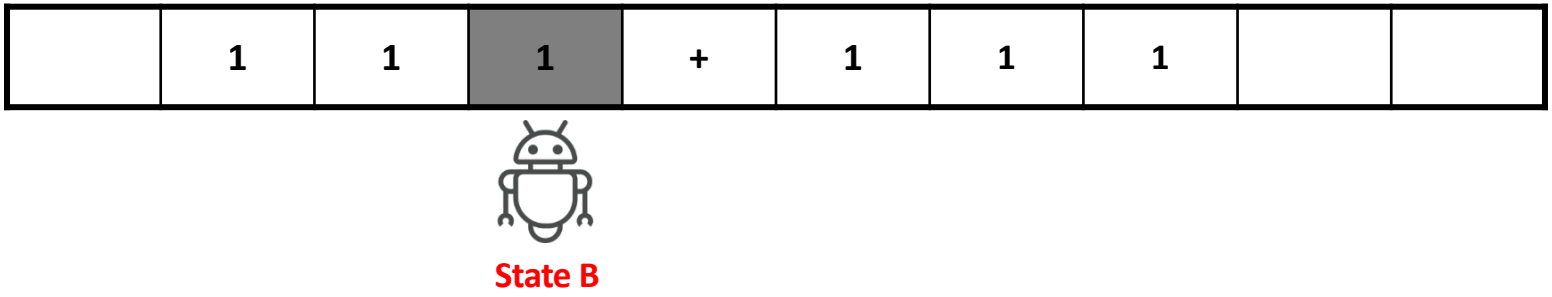
Turing Machine: “Hello World” Program

If read 1,

- **State A:**
  1. **Erase the 1.**
  2. **Scan next cell on right**
  3. **Go to state B**
- **State B:**
  1. **Scan next cell on right.**
  2. **Stay in State B.**

If read +,

- **State B:**
  1. **Erase the +.**
  2. **Print 1.**
  3. **Stop**



M. Gardner, Can Machines Think?, Chapter 9 in Mathematical Circus, pp. 102-104

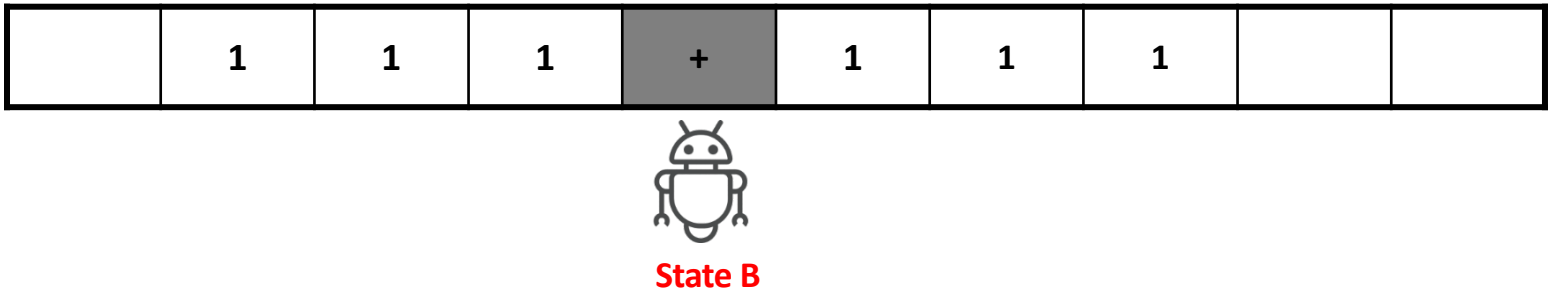
Turing Machine: “Hello World” Program

If read 1,

- **State A:**
  1. **Erase the 1.**
  2. **Scan next cell on right**
  3. **Go to state B**
- **State B:**
  1. **Scan next cell on right.**
  2. **Stay in State B.**

If read +,

- **State B:**
  1. **Erase the +.**
  2. **Print 1.**
  3. **Stop**



M. Gardner, Can Machines Think?, Chapter 9 in Mathematical Circus, pp. 102-104

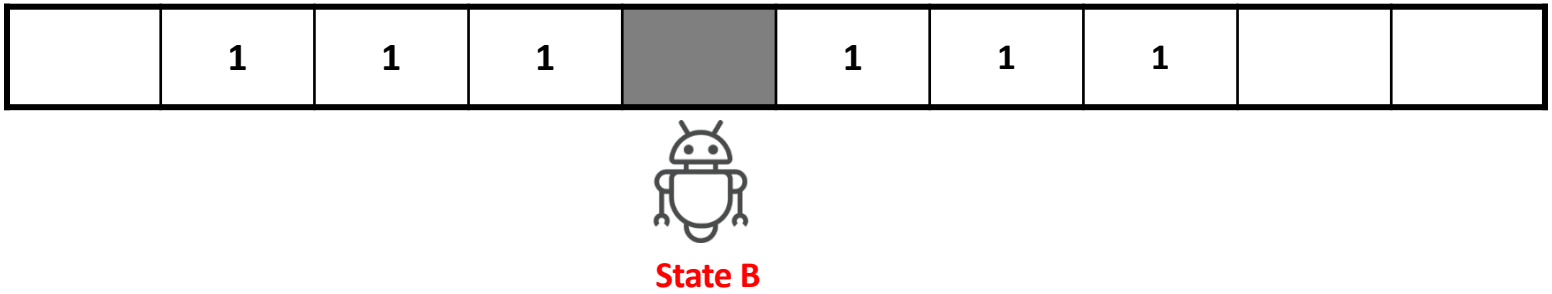
Turing Machine: “Hello World” Program

If read 1,

- **State A:**
  1. **Erase the 1.**
  2. **Scan next cell on right**
  3. **Go to state B**
- **State B:**
  1. **Scan next cell on right.**
  2. **Stay in State B.**

If read +,

- **State B:**
  1. **Erase the +.**
  2. **Print 1.**
  3. **Stop**



M. Gardner, Can Machines Think?, Chapter 9 in Mathematical Circus, pp. 102-104

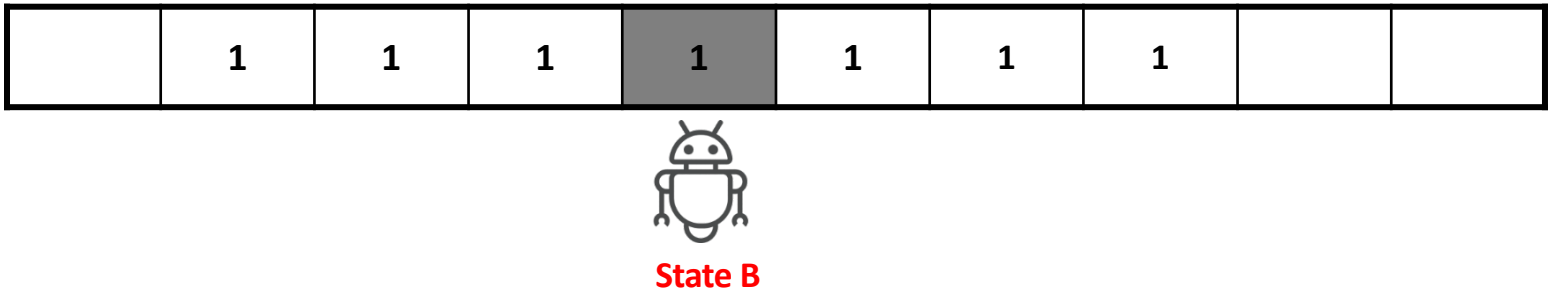
Turing Machine: “Hello World” Program

If read 1,

- **State A:**
  1. **Erase the 1.**
  2. **Scan next cell on right**
  3. **Go to state B**
- **State B:**
  1. **Scan next cell on right.**
  2. **Stay in State B.**

If read +,

- **State B:**
  1. **Erase the +.**
  2. **Print 1.**
  3. **Stop**



M. Gardner, Can Machines Think?, Chapter 9 in Mathematical Circus, pp. 102-104

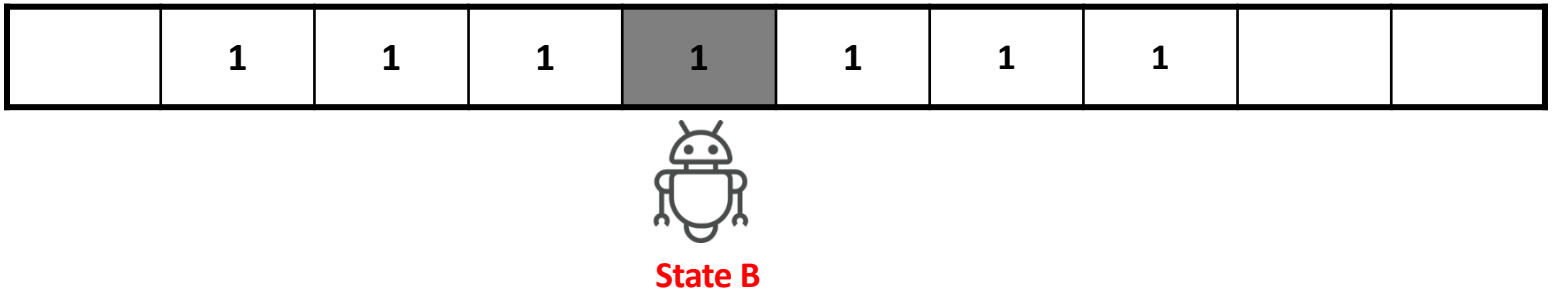
Turing Machine: “Hello World” Program

If read 1,

- **State A:**
  1. **Erase the 1.**
  2. **Scan next cell on right**
  3. **Go to state B**
- **State B:**
  1. **Scan next cell on right.**
  2. **Stay in State B.**

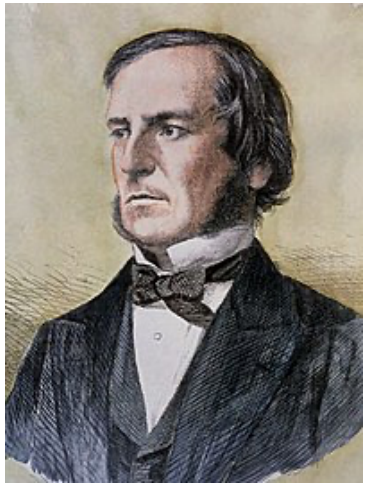
If read +,

- **State B:**
  1. **Erase the +.**
  2. **Print 1.**
  3. **Stop**



M. Gardner, Can Machines Think?, Chapter 9 in Mathematical Circus, pp. 102-104

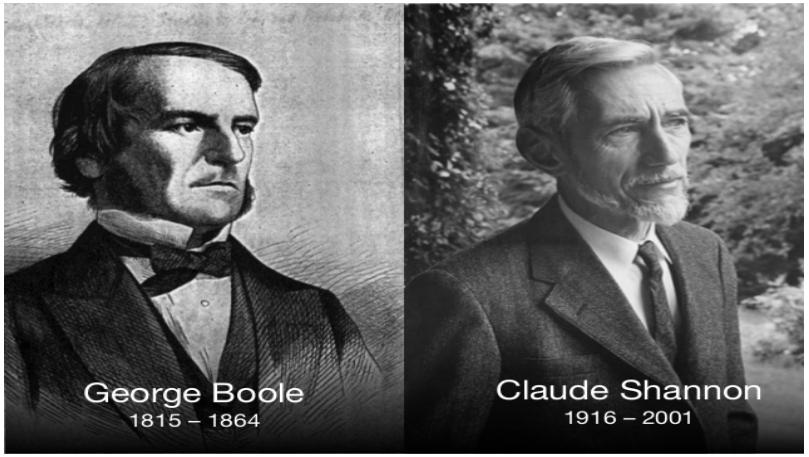
Boolean Logic and Boolean Algebra



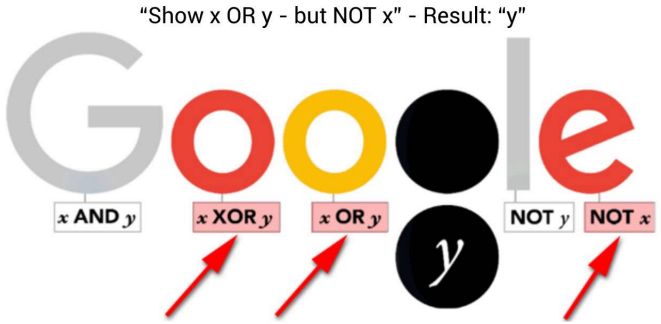
George Boole, English mathematician who established modern symbolic logic is self-taught. He wrote in 1854 "*An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities*", where his algebra of logic, now called Boolean algebra, pointed out that if his '1' were taken as truth and his '0' as falsehood, the calculus could be applied to statements that are either true or false. This leads to Propositional Calculus.

http://sydneypadua.com/2dgoggles/wp-content/uploads/2015/11/boole2.jpg

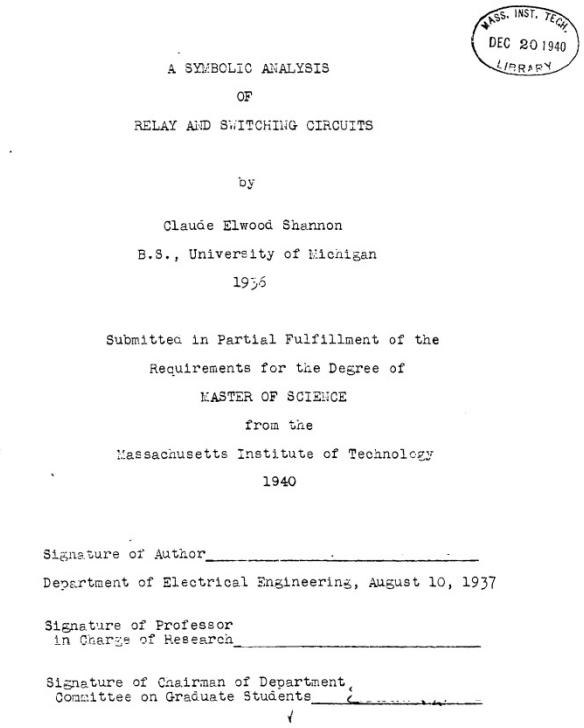
Boolean Logic and Boolean Algebra



The Genius of George Boole (2015), 40:30
https://www.youtube.com/watch?v=Hljir_TyTEw



Google's doodles of 200th Birthday of Georgle Boole:
https://www.google.com/doodles/george-booles-200th-birthday

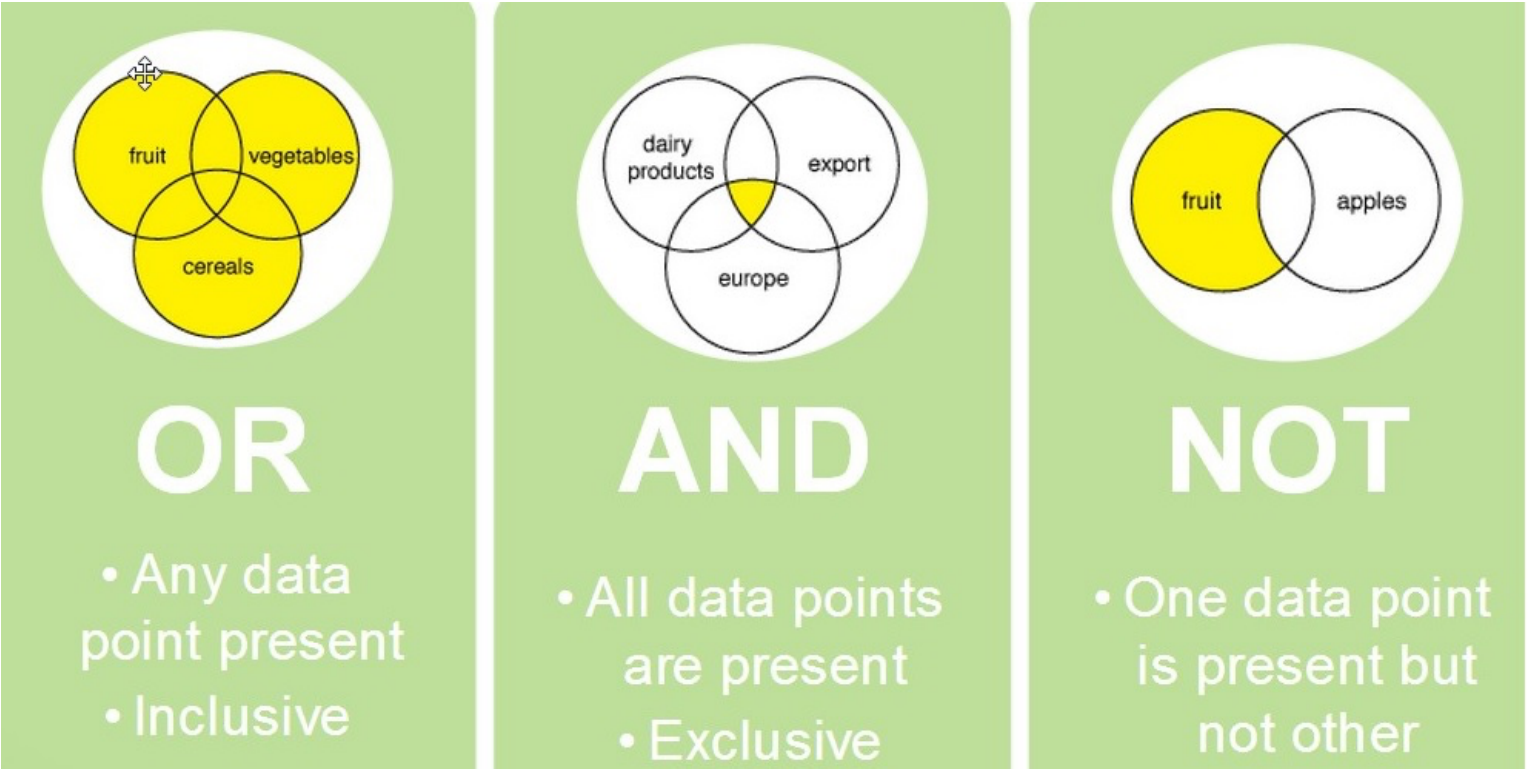


Shannon, at the age of 21, wrote in 1937 "*possibly the most important, and also the most noted, master's thesis of the century*" that describes fully the building blocks to build a logical computer (He stumbled across Boole's work in a philosophy class).

Shannon demonstrated that Boolean algebra is the algebra of sets, which can be physically realized in switching circuits and networks to manipulate any logical tasks. Let's see his adder later.

Boolean Logic and Boolean Algebra

- This is the calculus concerned with true or false statements connected by the following three binary relations:



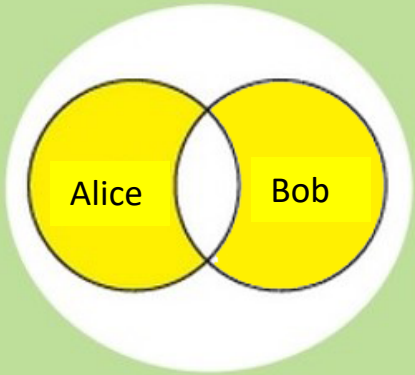
- These three logic relations can build the entire universe of logic
- A Venn diagram is a convenient tool to understand logic

Boolean Logic and Boolean Algebra

- A natural duality correspondence between set theory and logic operators

Set Theory	Logic
$A \cup B$	p or q
$A \cap B$	p and q
$A = B$	$p \leftrightarrow q$
$A \subseteq B$	$p \rightarrow q$
$(A \cup B)' = A' \cap B'$	$(p \text{ or } q)' \leftrightarrow p' \text{ and } q'$
$(A \cap B)' = A' \cup B'$	$(p \text{ and } q)' \leftrightarrow p' \text{ or } q'$

Boolean Logic and Boolean Algebra



EXCLUSIVE OR

- Data is true if and only if one operand is true and the other is false

Example of Exclusive–Or Statement that is valid

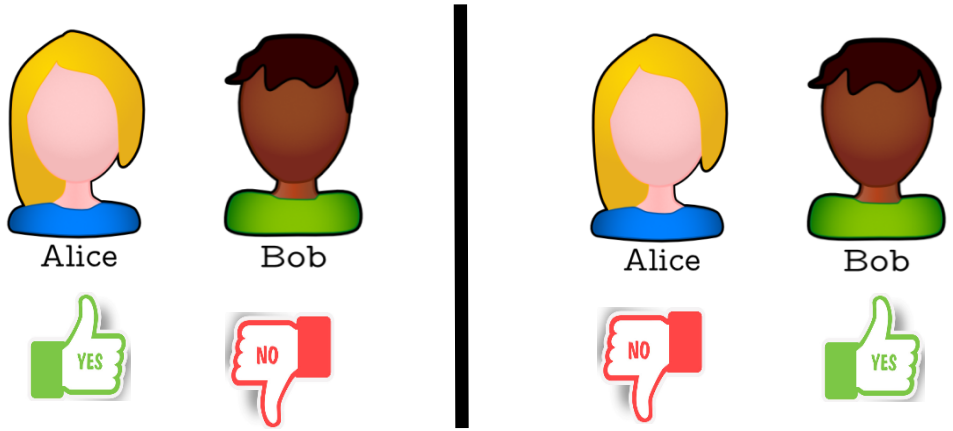
Either I vote for Alice or I vote for Bob in the election with a single vote.

I did not vote for Alice and I did not vote for Bob. (False)

I voted for Alice and I did not vote for Bob. (True)

I did not vote for Alice and I voted for Bob. (True)

I voted for Alice and I voted for Bob. (False)



Example 1

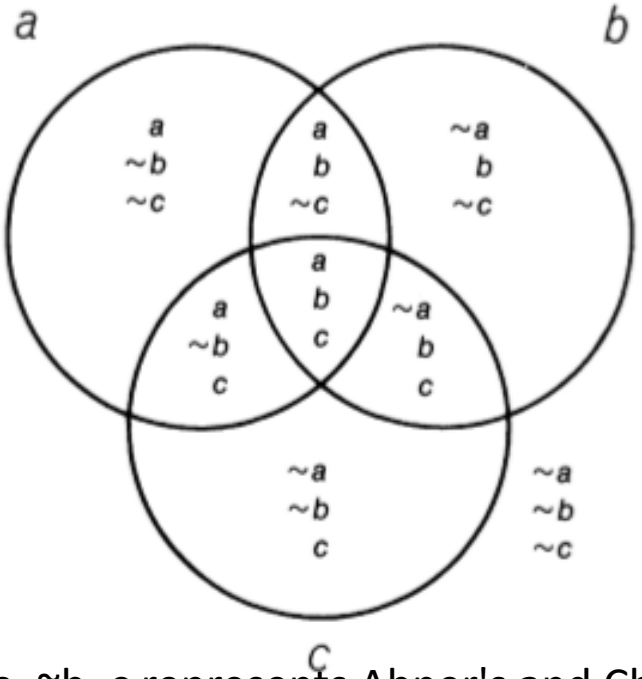
To see how easily the Venn circles solve certain types of logic puzzles, consider the following premises about three businessmen, Abner, Bill, and Charley, who lunch together every working day:

1. If Abner orders a martini, so does Bill.
2. Either Bill or Charley always orders a martini, but never both at the same lunch.
3. Either Abner or Charley or both always order a martini.
4. If Charley orders a martini, so does Abner.

M. Gardner, Boolean Algebra, Chapter 8 in Mathematical Circus, pp. 87-101

Example 1 (cont.)

- The eight areas of the overlapping circles shown in following diagram are labeled to show all possible combinations of truth values for a, b, c, which stand for Abner, Bill, and Charley.

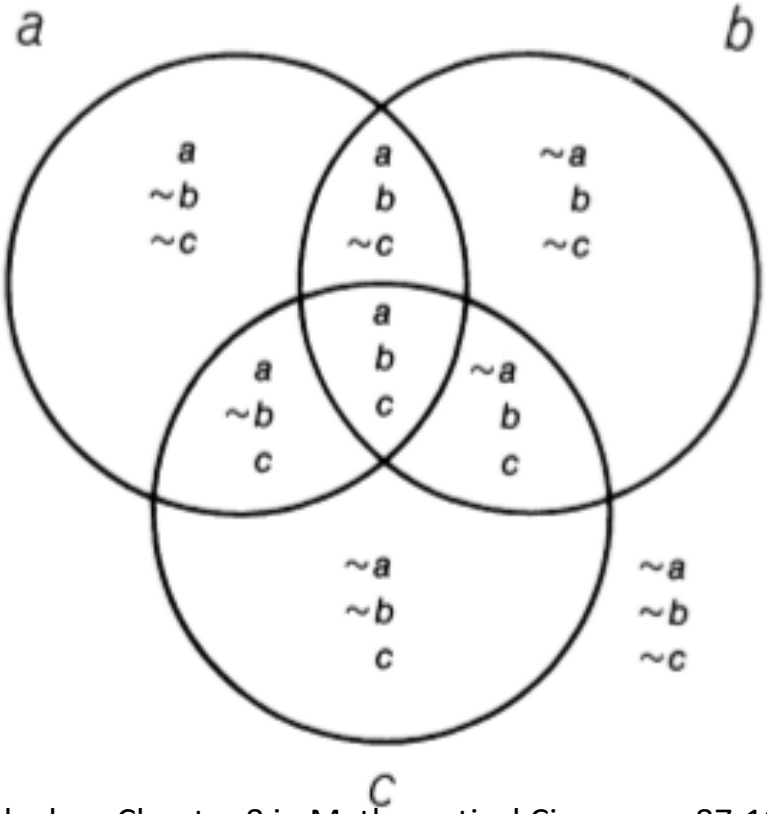


Thus the area marked a, ~b, c represents Abner's and Charley's having martinis while Bill does not.

M. Gardner, Boolean Algebra, Chapter 8 in Mathematical Circus, pp. 87-101

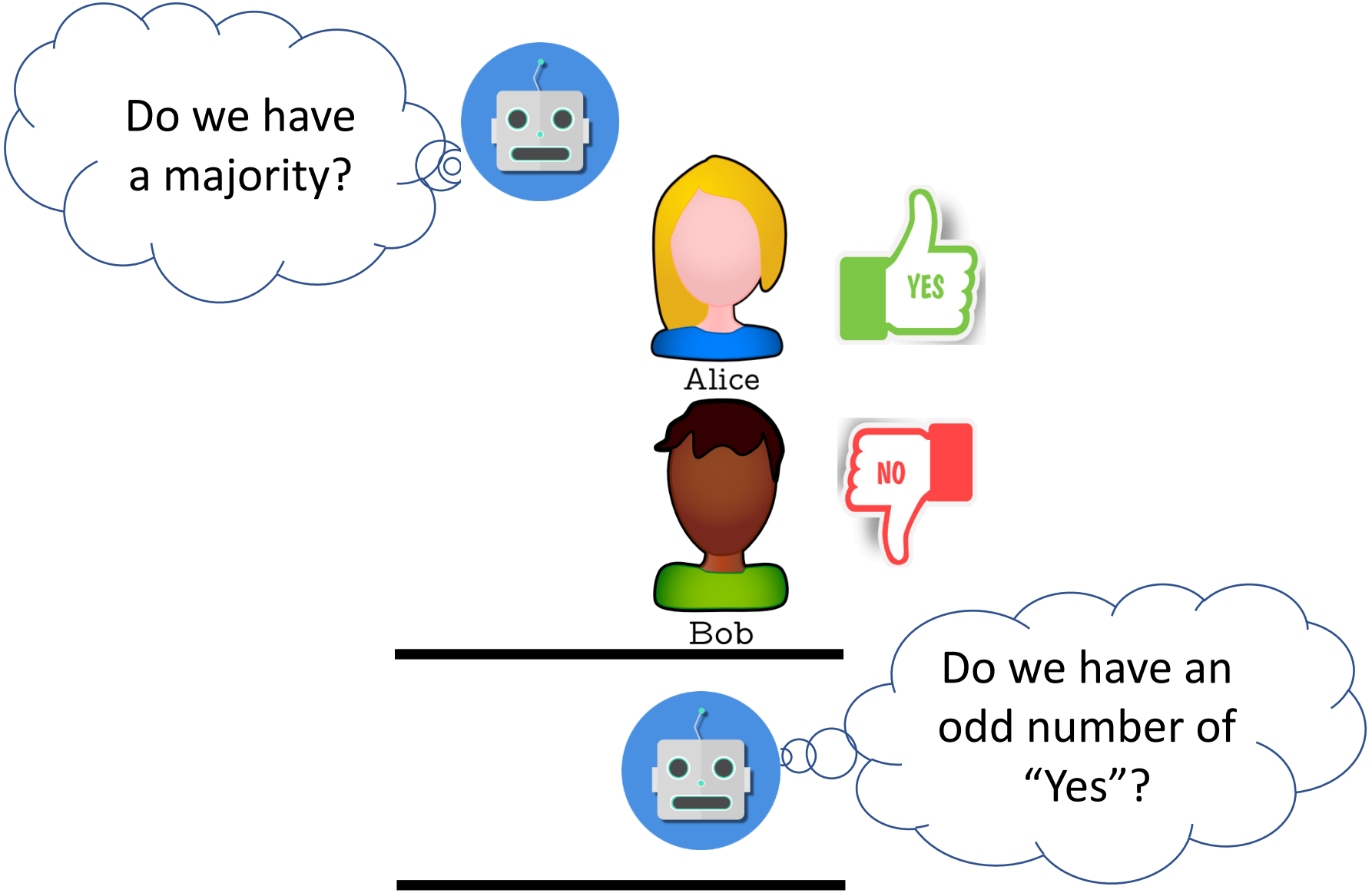
Example 1 (cont.)

See if you can shade the areas declared empty by the four premises and then examine the result to determine who will order martinis if you lunch with the three men.

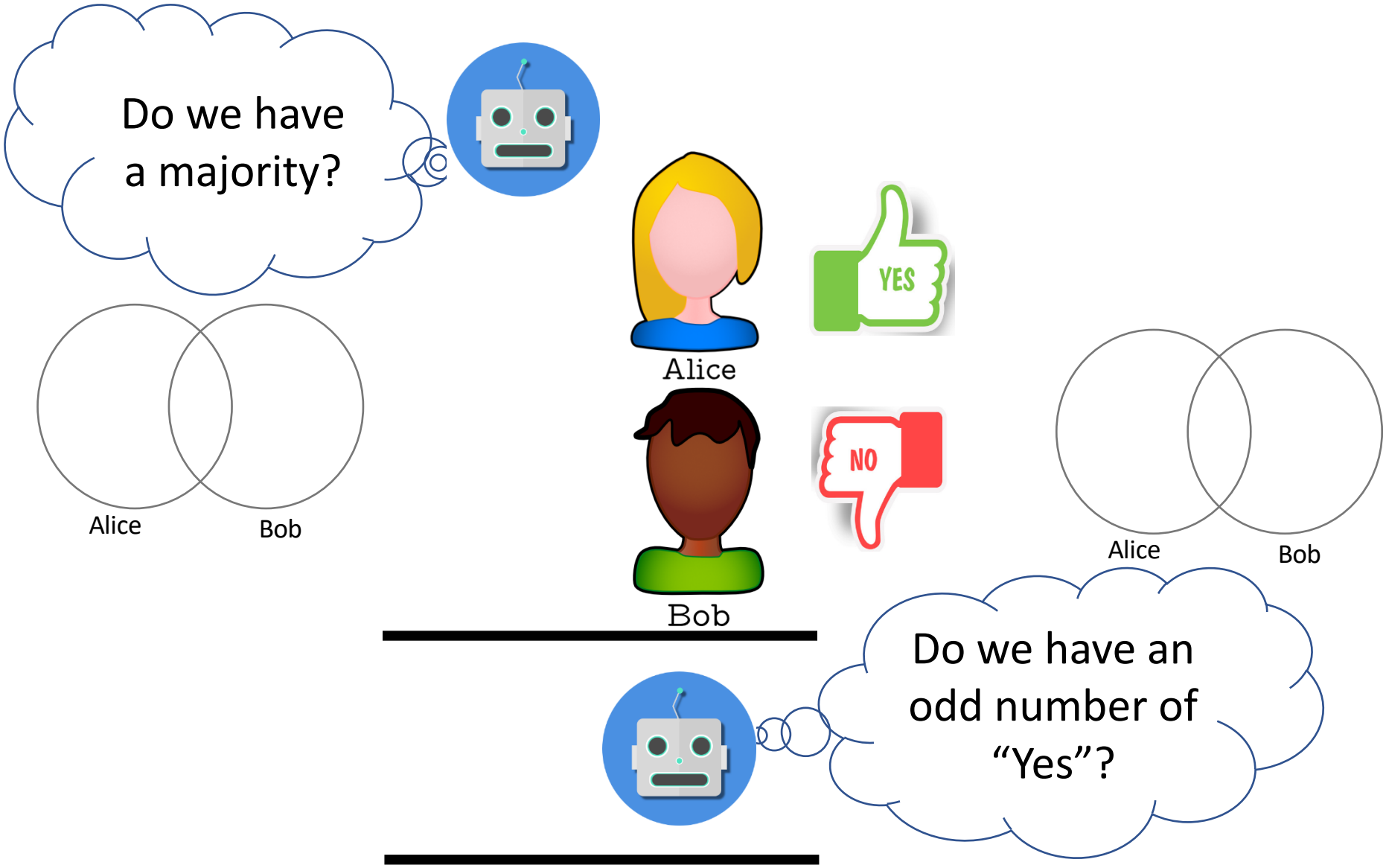


M. Gardner, Boolean Algebra, Chapter 8 in Mathematical Circus, pp. 87-101

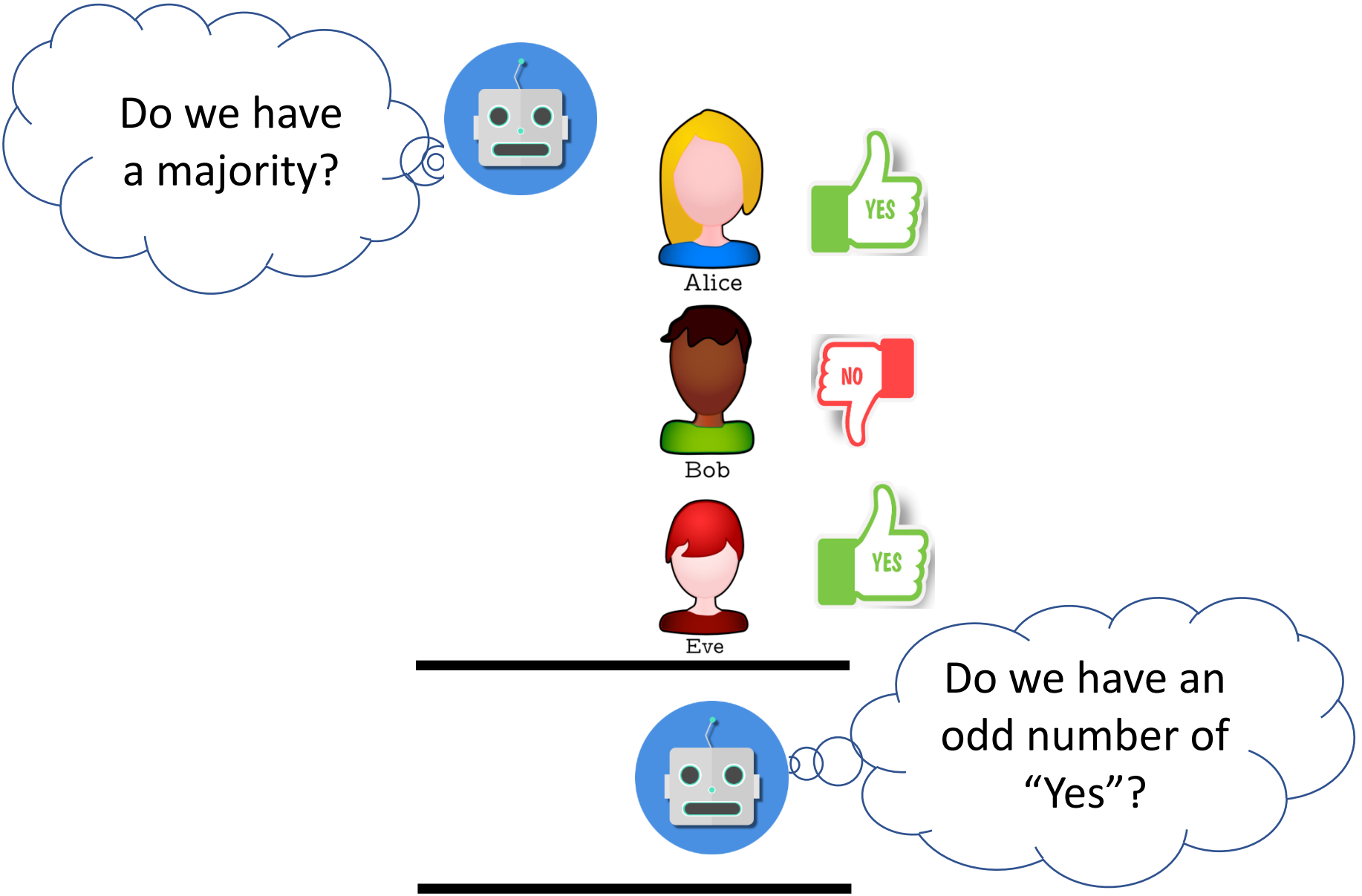
Boolean Logic and Shannon's Adder



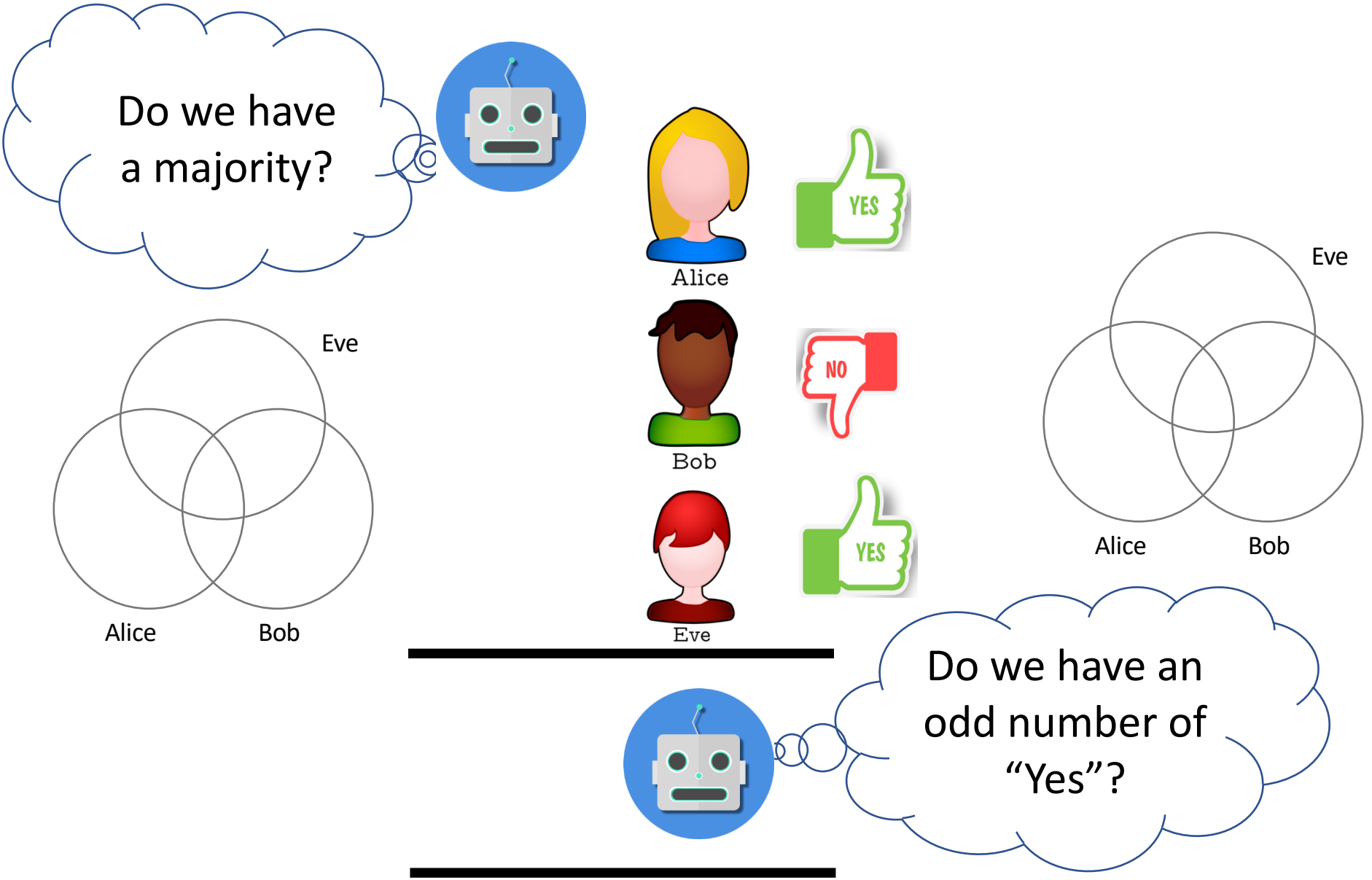
Boolean Logic and Shannon's Adder



Boolean Logic and Shannon's Adder



Boolean Logic and Shannon's Adder



Boolean Logic and Shannon's Adder

Carry: Do we have a majority?

Adding Binary Digits (Bit)

0+0 = Sum:0, Carry:0

1+0 = Sum:1, Carry:0

1+1 = Sum:0, Carry:1

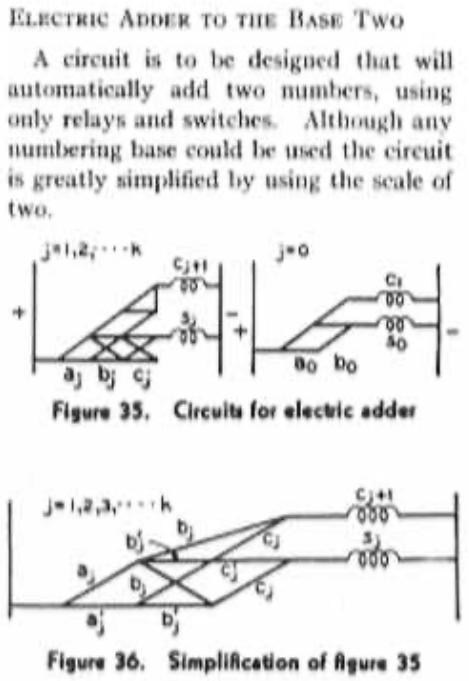
'1' is logically yes

'0' is logically no

```
  1 1 1
    1 0 1 1
+   0 1 0 1
-----------
  1 0 0 0 0
```

Sum: Do we have an odd number of "Yes"?

Shannon's Adder (1937)



ELECTRIC ADDER TO THE BASE TWO

A circuit is to be designed that will automatically add two numbers, using only relays and switches. Although any numbering base could be used the circuit is greatly simplified by using the scale of two.

Figure 35. Circuits for electric adder

Figure 36. Simplification of figure 35

As an aside, *Shannon's adder* can be used to *build the entire universe of logic*, i.e., it is the basic building block of logic!

Example 2: Revisiting Elementary School Math

Consider the following set of eight numbers: 1, 2, 3, 5, 6, 10, 15, 30. They are the factors of 30, including 1 and 30 as factors.

We interpret "union" as the least common multiple of any pair of those numbers. "Intersection" of a pair is taken to be their greatest common divisor.

What is the greatest common divisor of 6 and 10?

What is the least common multiple of 6 and 10?

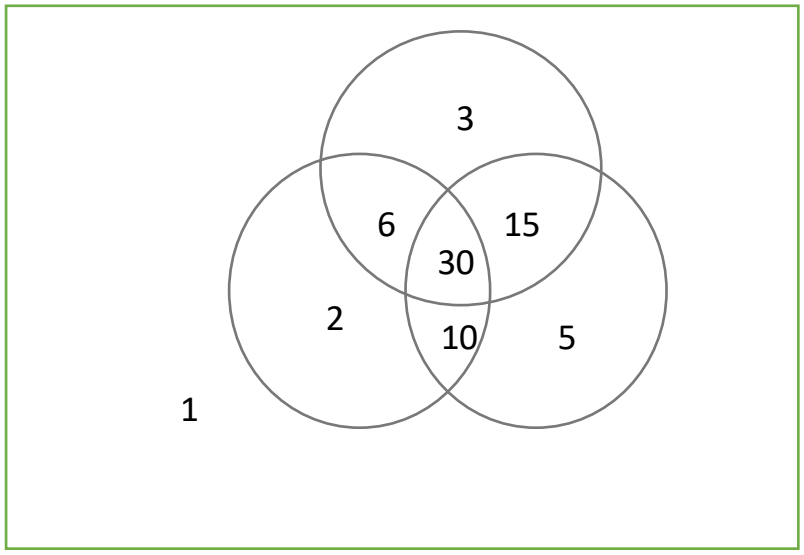
M. Gardner, Boolean Algebra, Chapter 8 in Mathematical Circus, pp. 87-101

Bunitskiy, E., "Some applications of mathematical logic to the theory of the greatest common divisor and least common multiple" (in Russian), Vestnik Opytnoy Jiziki i elem. mat., no. 274, 1899.

Example 2: Revisiting Elementary School Math

Set inclusion becomes the relation "is a factor of." The universal set is 30, the null set 1. The complement of a number *a* is 30/a. With these novel interpretations of the Boolean relations, it turns out that we have a consistent Boolean structure! All the theorems of Boolean algebra have their counterparts in this curious system based on the factors of 30.

30/15 = 2, 2/2 = 1
30/10 = 3, 3/3 = 1
30/6 = 5, 5/5 = 1
30/3 = 10, 10/5 = 2 , 2/2 =1
30/2 = 15, 15/3 = 5 , 5/5 =1
30/5 = 6 , 6/3 = 2 , 2/2 =1
30/1 = 30
30/30 = 1



M. Gardner, Boolean Algebra, Chapter 8 in Mathematical Circus, pp. 87-101

Example 2: Revisiting Elementary School Math

The numbers 1, 2, *3,* 5, 6, 7, 10, 14, 15, 21, 30, 35, 42, 70, 105, 210, the 16 factors of 210, also form a Boolean algebra when interpreted in the same way, although of course 210 is now the universal set and the complement of *a* is 210/a.

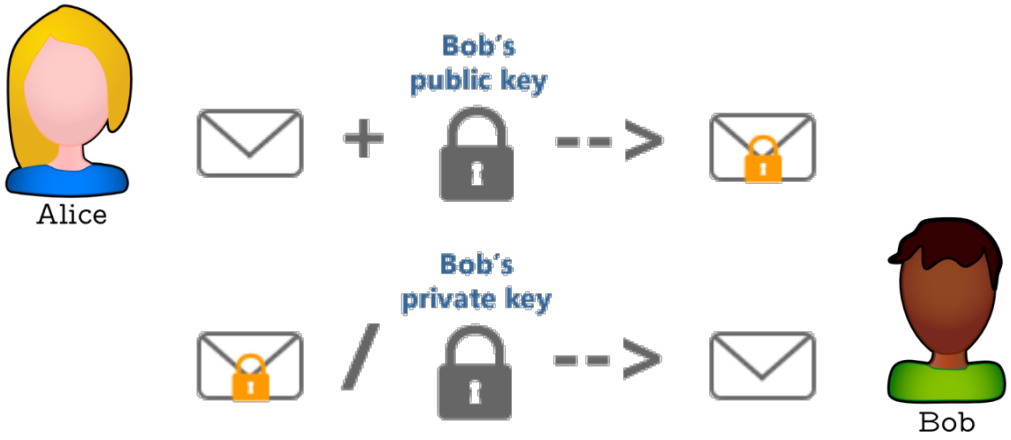
Can you discover a simple way to generate sets of 2^n numbers, where n is any positive integer, that will form Boolean systems of this peculiar kind?

If yes, you have an algorithm!

M. Gardner, Boolean Algebra, Chapter 8 in Mathematical Circus, pp. 87-101

Millenium Prize Challenge

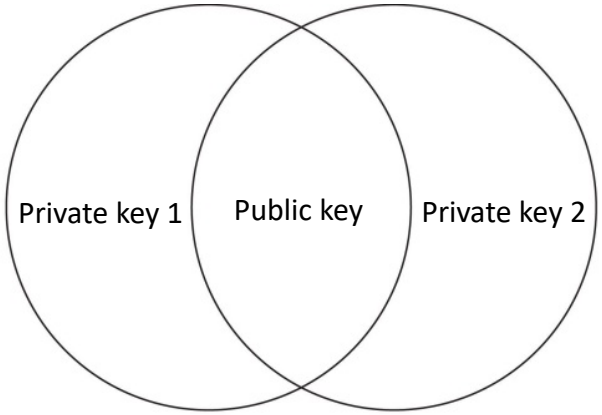
- An encryption method is presented with the novel property that publicly revealing an encryption key does not thereby reveal the corresponding decryption key. This has two important consequences:
  - Couriers or other secure means are not needed to transmit keys.
  - A message can be 'signed' using a privately held decryption key. Anyone can verify this signature using the corresponding publicly revealed encryption key...



Rivest, Shamir, Adleman; *A method for obtaining digital signatures and public-key cryptosystems*; Communications of the ACM, Feb 1978.

Millenium Prize Challenge

- The heart of the RSA crypto-system is that, for each number n, there exists prime numbers p and q such that $n = p \times q$. Find these two primes, given only n
- The RSA crypto-system is a real-life practical application of a Boolean algebra system that has commercial and consequential value



- Suppose my public key is 35, break this RSA crypto-system using the simple algorithm you devised. What if my public key is 143?
- How many digits do current RSA public key have?

https://en.wikipedia.org/wiki/RSA_Factoring_Challenge

https://en.wikipedia.org/wiki/The_Magic_Words_are_Squeamish_Ossifrage