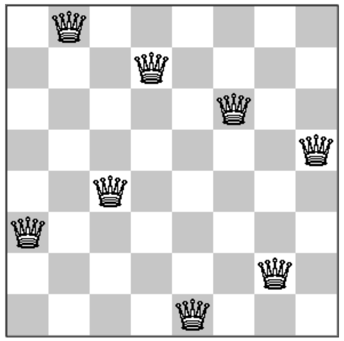
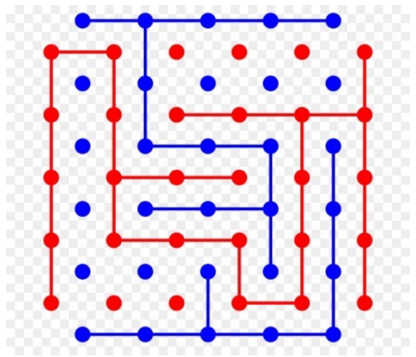
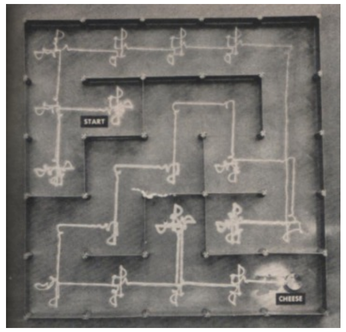
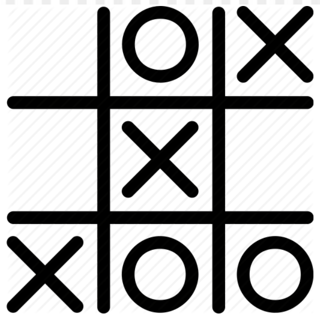
Artificial Intelligence:

Past, Present and Future



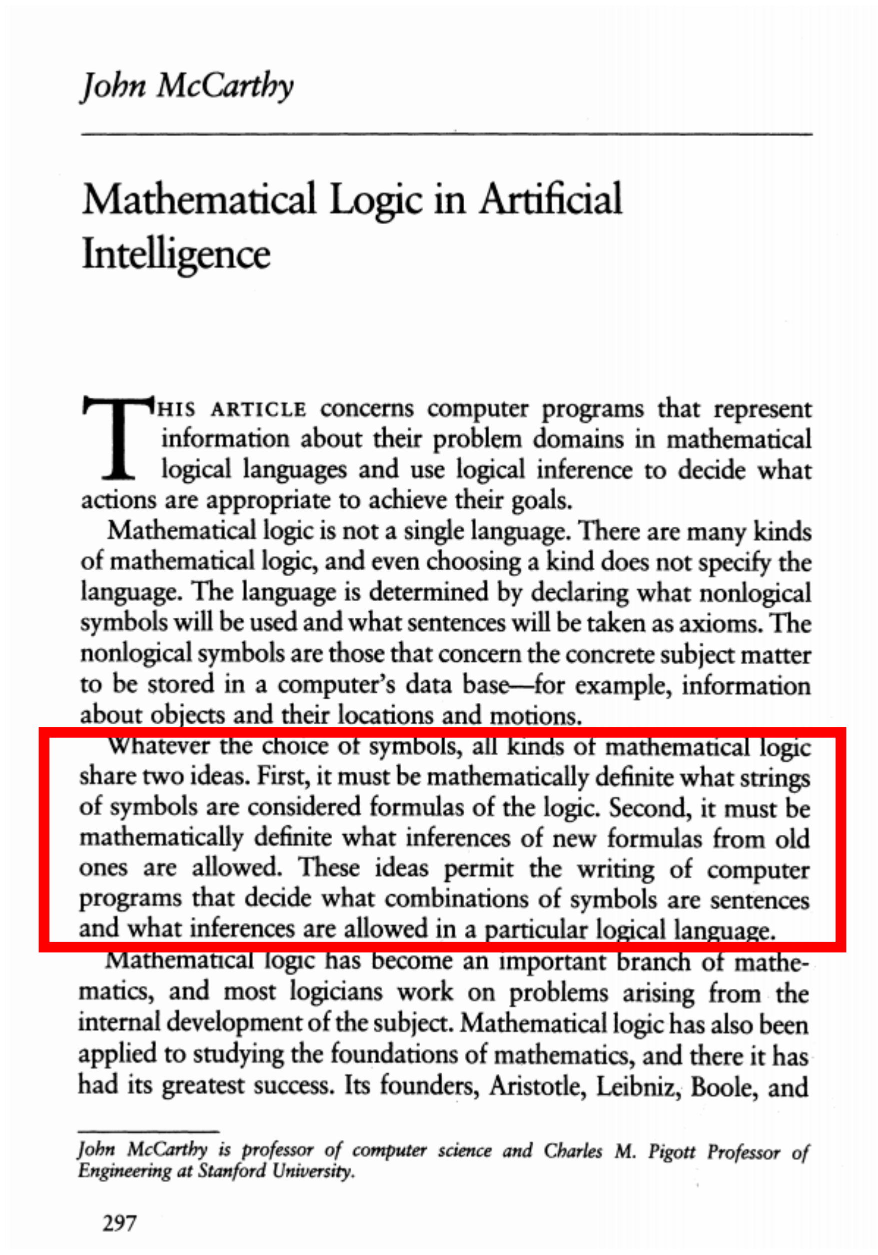






Chee Wei Tan

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.

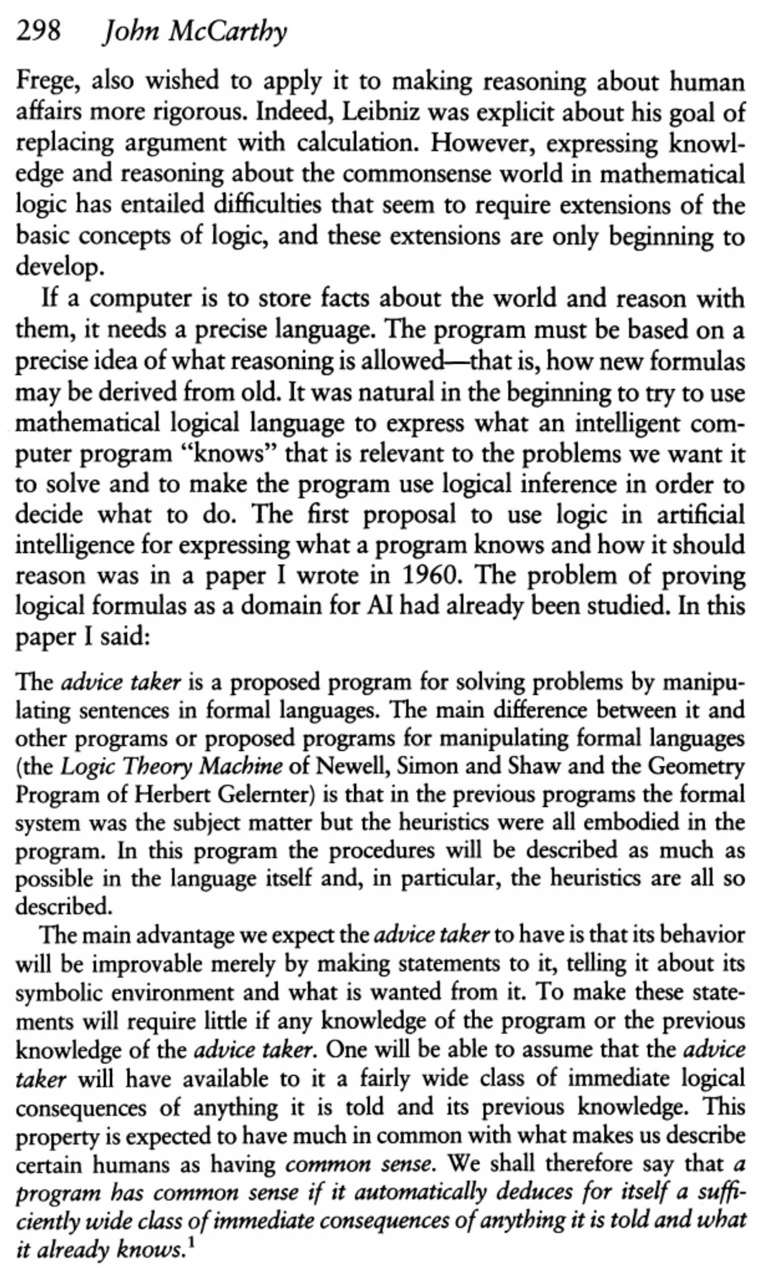
Whatever the choice of symbols, all kinds of mathematical logic share two ideas. First, it must be mathematically definite what strings of symbols are considered formulas of the logic. Second, it must be mathematically definite what inferences of new formulas from old ones are allowed. These ideas permit 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.

297

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



298 *John McCarthy*

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

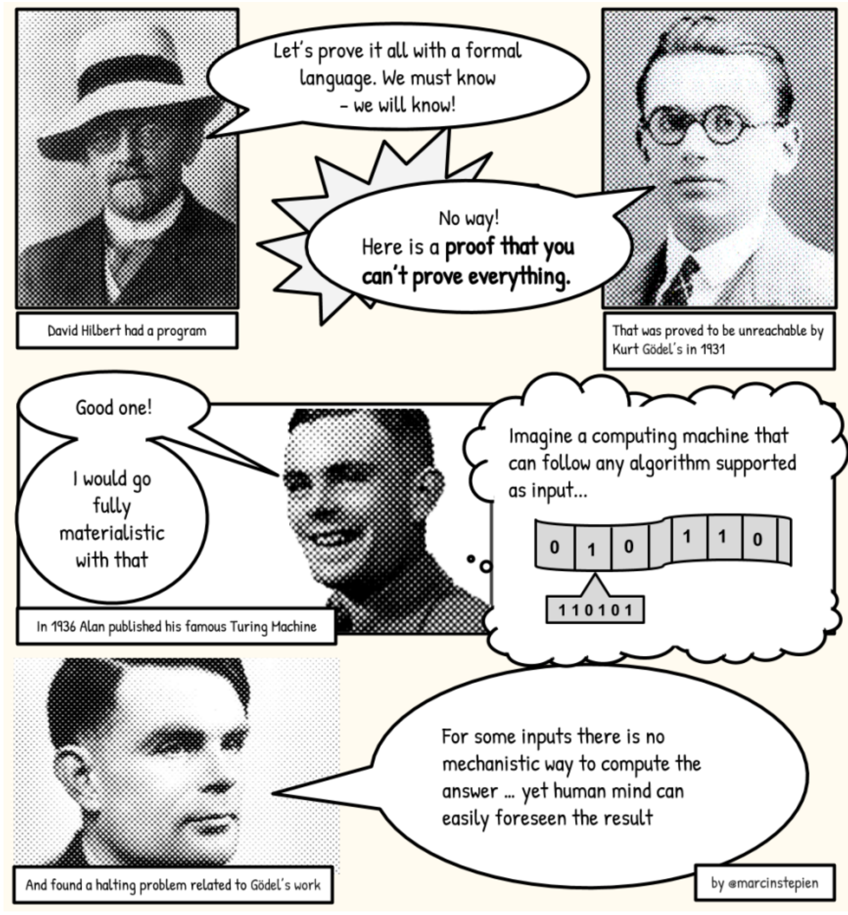
BANK OF ENGLAND
Alan Turing Banknote Concept
Bank of England
Fifty Pounds
"This is only a foretaste of what is to come and only the shadow of what is going to be"
Alan Turing (1912-1954)
©The Governor and Company of the Bank of England 2019

- 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

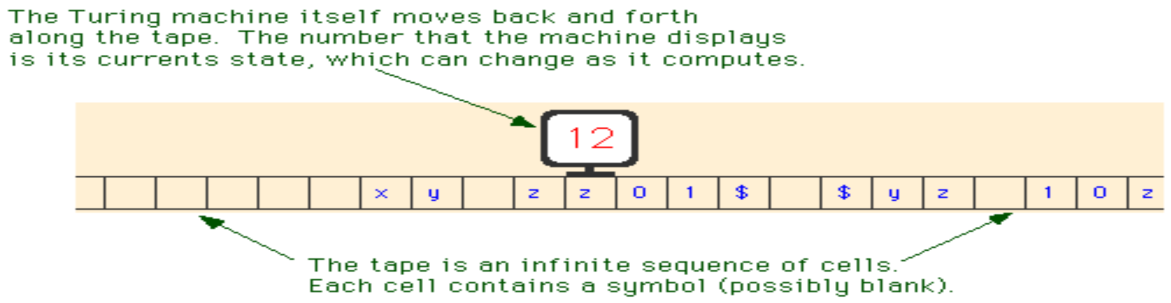
https://www.bankofengland.co.uk/banknotes/50-pound-note-nominations

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.


Let's prove it all with a formal language. We must know – we will know!
David Hilbert had a program
No way! Here is a proof that you can't prove everything.
That was proved to be unreachable by Kurt Gödel's in 1931
Good one!
I would go fully materialistic with that
In 1936 Alan published his famous Turing Machine
Imagine a computing machine that can follow any algorithm supported as input...
0 1 0 1 1 0
110101
For some inputs there is no mechanistic way to compute the answer ... yet human mind can easily foreseen the result
And found a halting problem related to Gödel's work
by @marcinstepien

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


The Turing machine itself moves back and forth along the tape. The number that the machine displays is its currents state, which can change as it computes.
12
x y z z 0 1 $ $ y z 1 0 z
The tape is an infinite sequence of cells. Each cell contains a symbol (possibly blank).

Turing Machine: “Hello World” Program

If read 1,

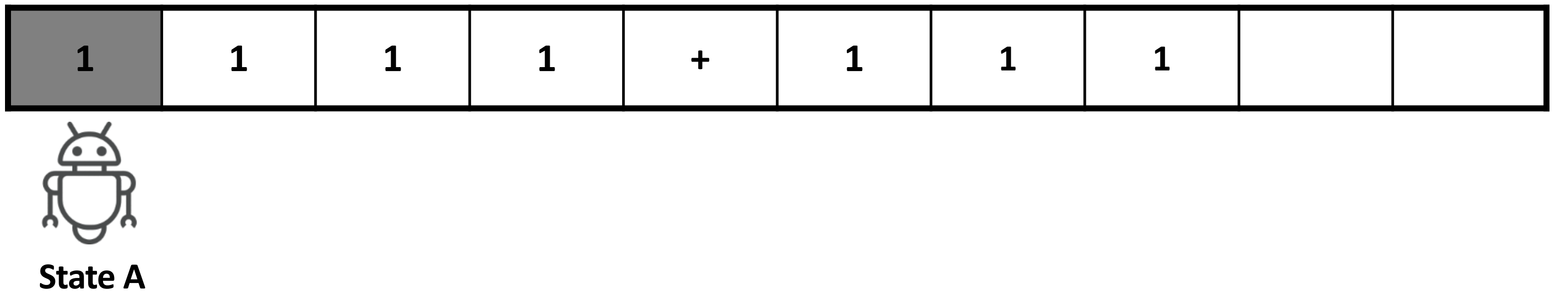
- **State A:**
  1. **Erase the 1.**
  2. **Scan next cell on right**
  3. **Go to state B**
- **State B:**
  1. **Scan next cell on right.**
  2. **Stay in State B.**

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

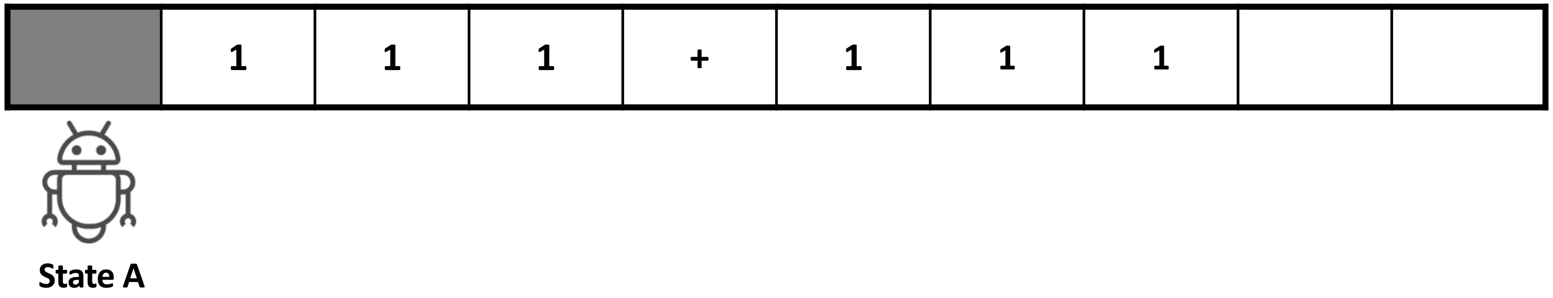
Turing Machine: “Hello World” Program

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



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

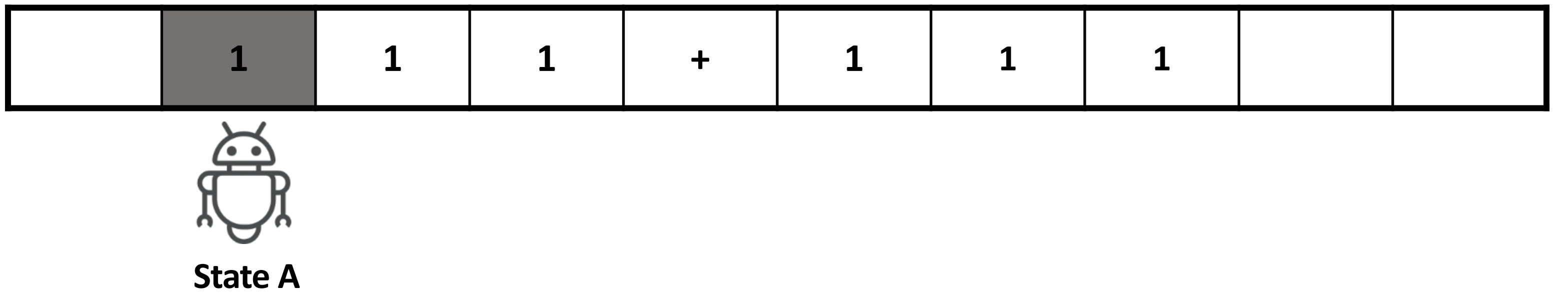
Turing Machine: “Hello World” Program

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

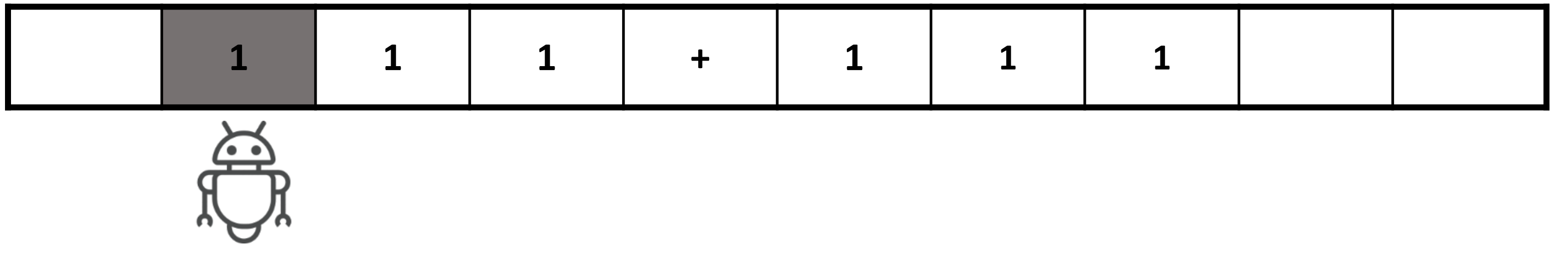
Turing Machine: “Hello World” Program

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**



State B

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

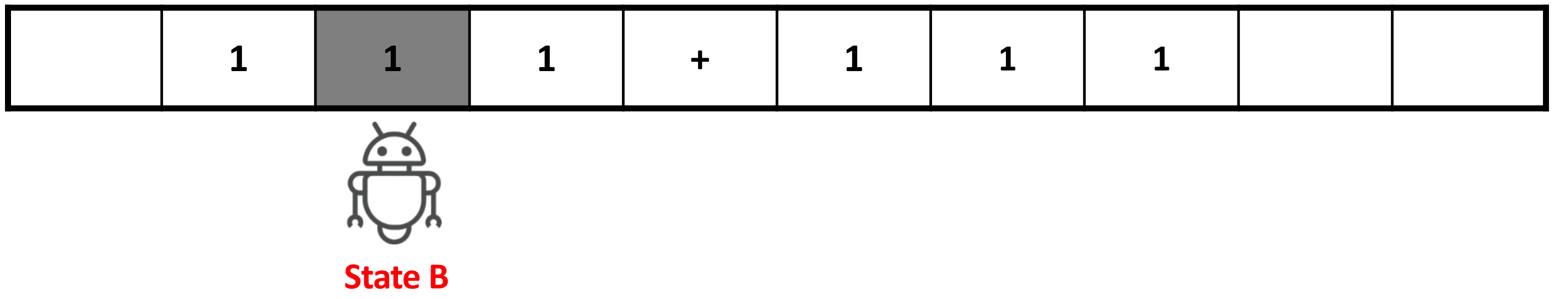
Turing Machine: “Hello World” Program

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

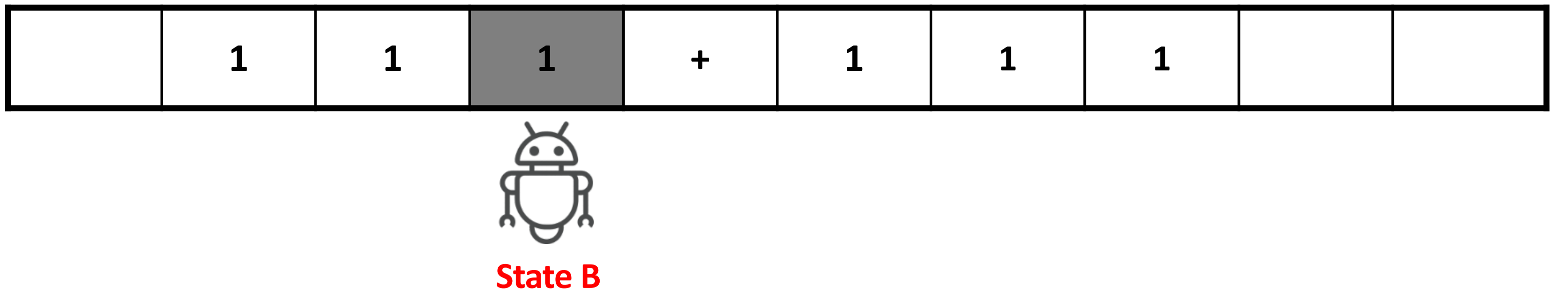
Turing Machine: “Hello World” Program

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

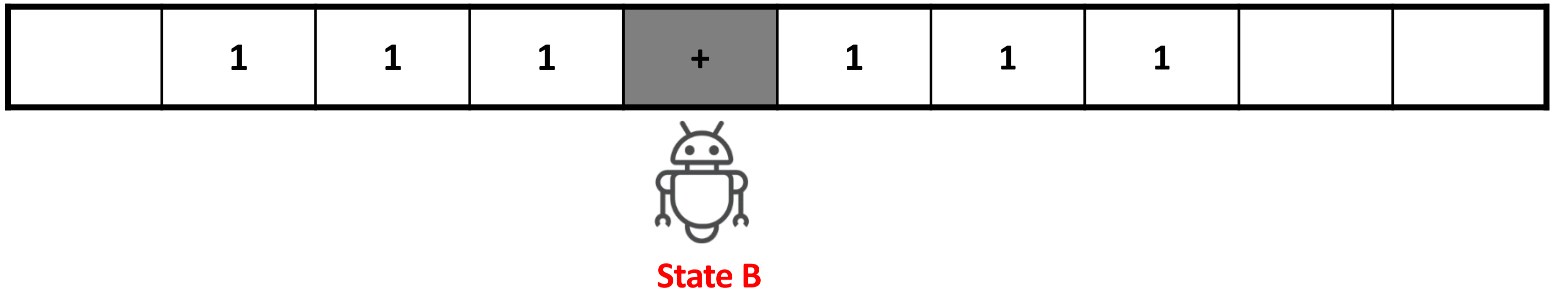
Turing Machine: “Hello World” Program

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

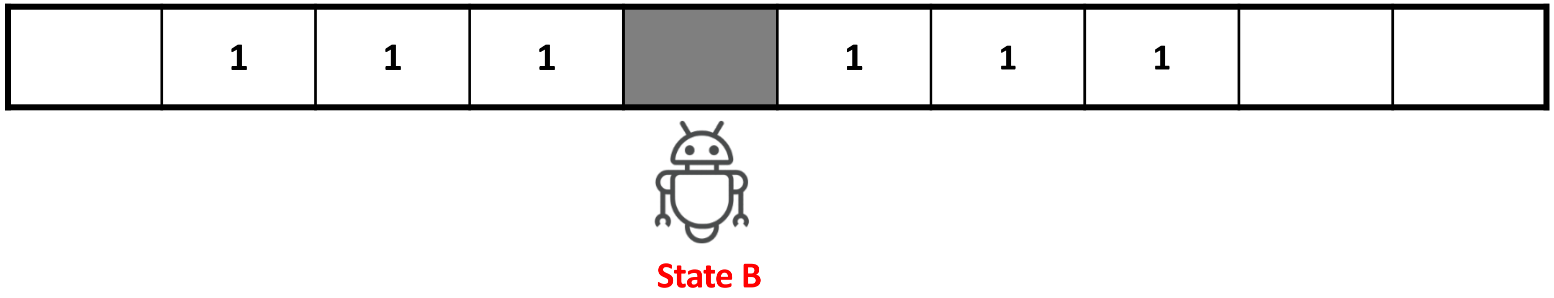
Turing Machine: "Hello World" Program

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

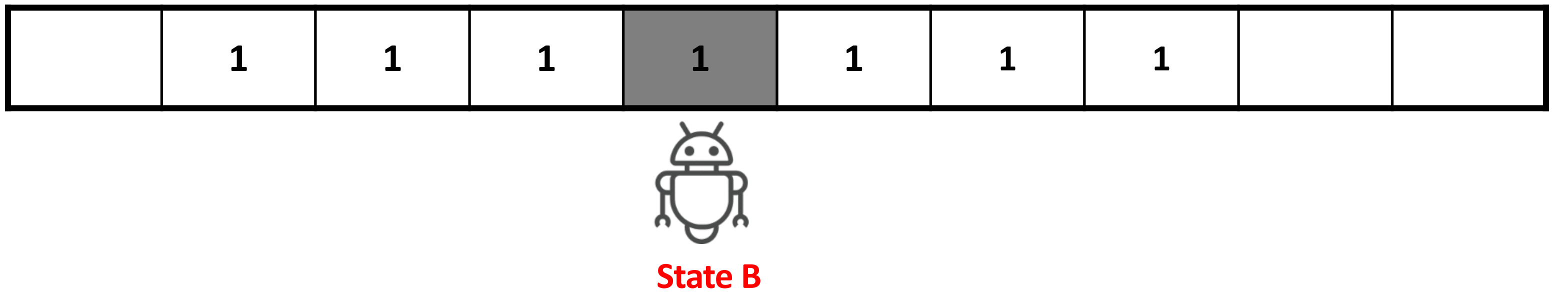
Turing Machine: “Hello World” Program

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

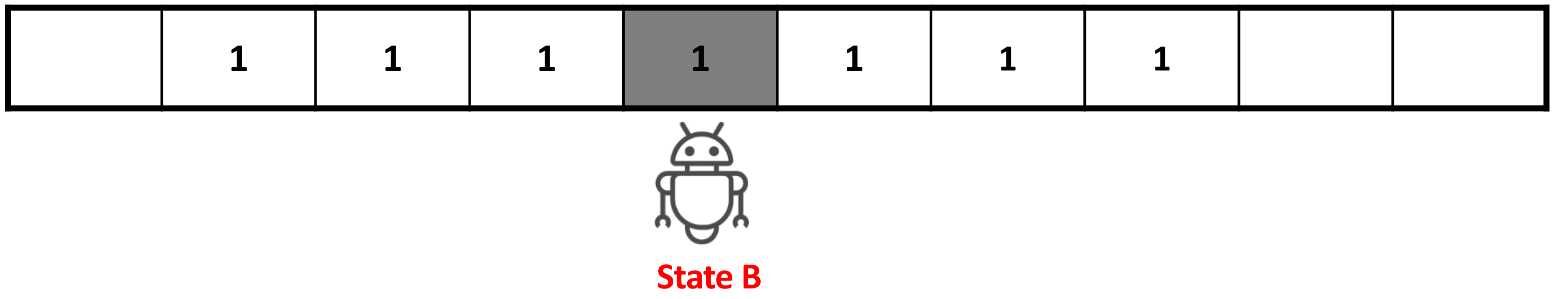
Turing Machine: “Hello World” Program

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.**

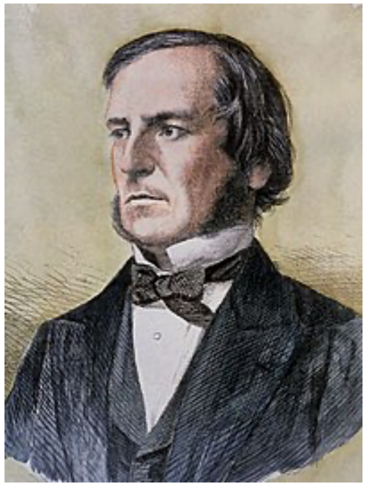
If read +,

- **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



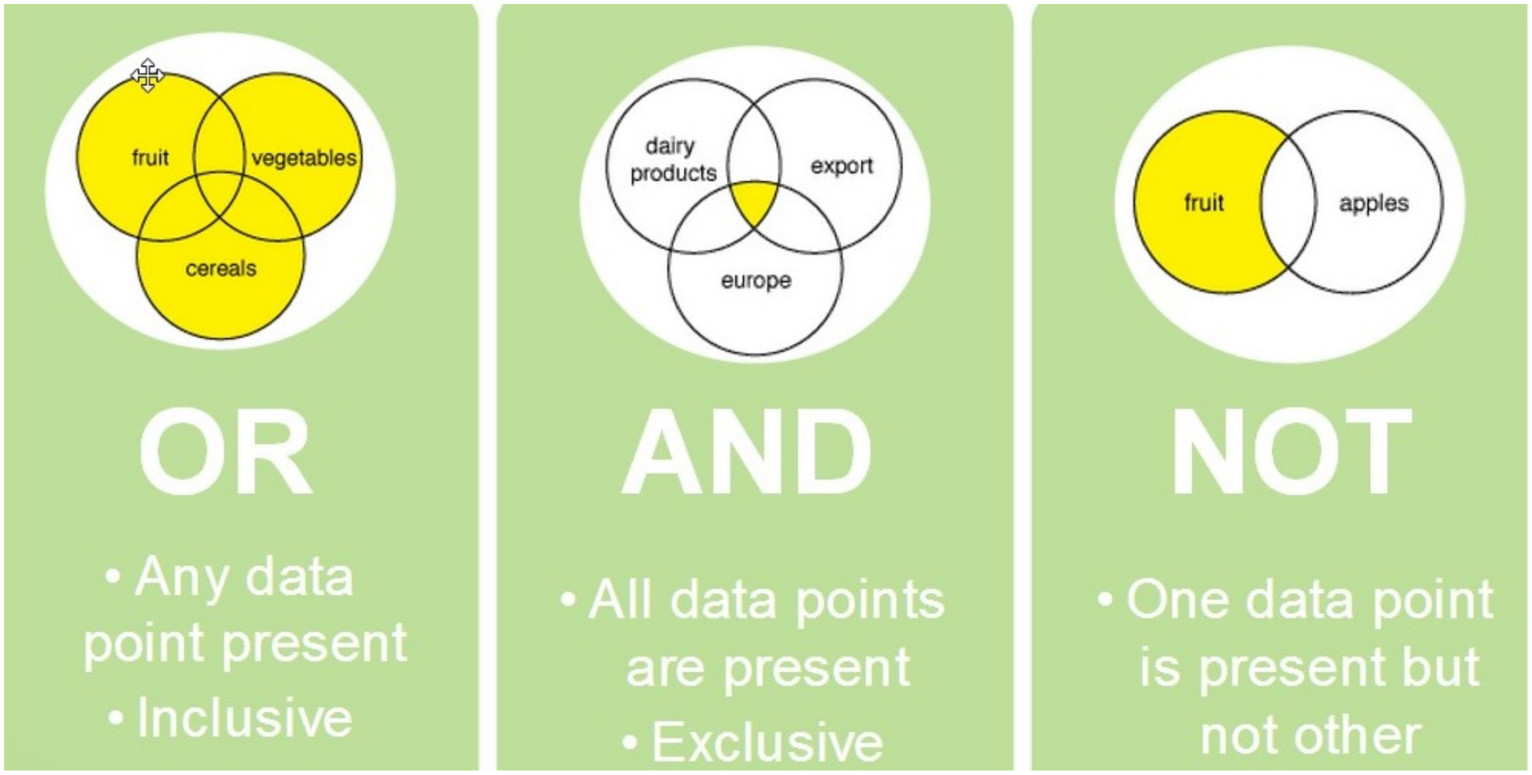
LADY LOVELACE AND MR. BABBAGE WILL BE DOWN SHORTLY, WOULD YOU LIKE COFFEE, OR TEA?
YES.
YOU WANT COFFEE AND TEA?
NO.
SO YOU WANT COFFEE?
NO!
SO YOU WANT TEA!
YES.
"BE A FOOTMAN!" THEY SAID. "YOU HAVE THE LEGS FOR IT!" THEY SAID. "THERE'S NO THINKING AT ALL!" THEY SAID.
HOW DO YOU LIKE...

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

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



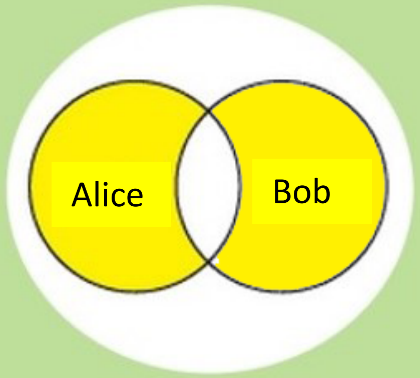
- These three logic relations can build the entire universe of logic
- A Venn diagram is a convenient tool to understand logic

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

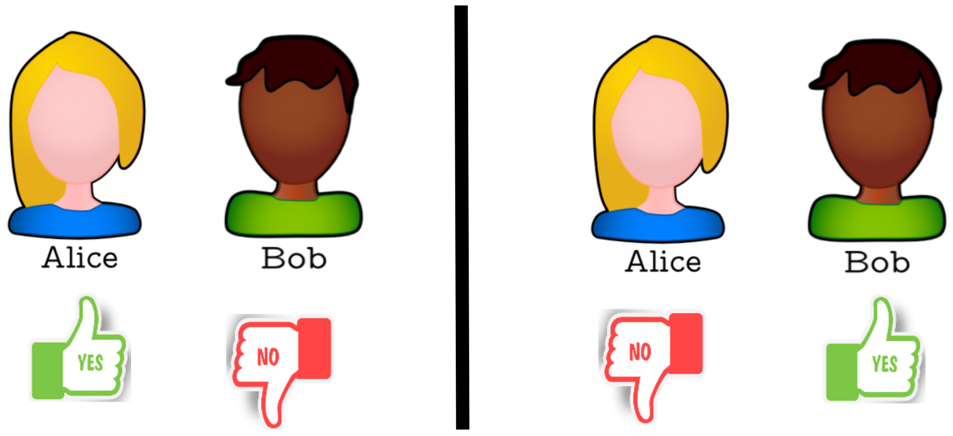
Either I vote for Alice or I vote for Bob in the election with a single vote.

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

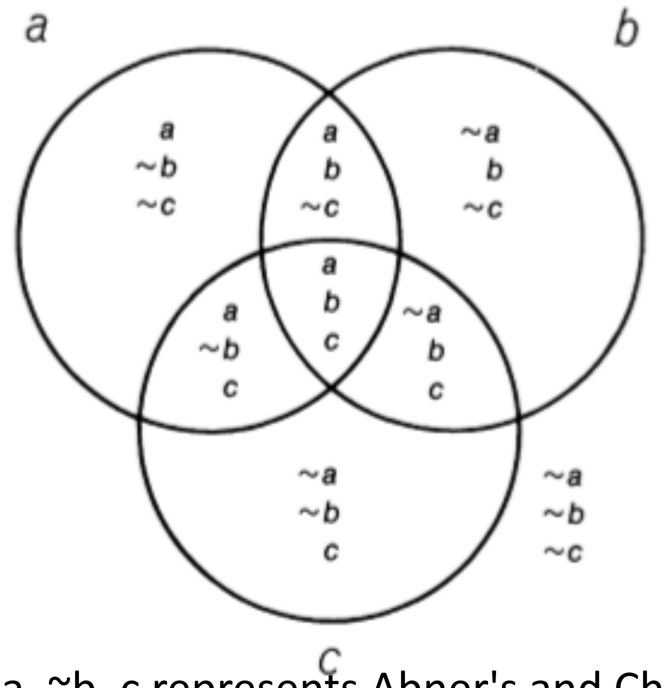
To see how easily the Venn circles solve certain types of logic puzzles, consider the following premises about three businessmen, Abner, Bill, and Charley, who lunch together every working day:

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.

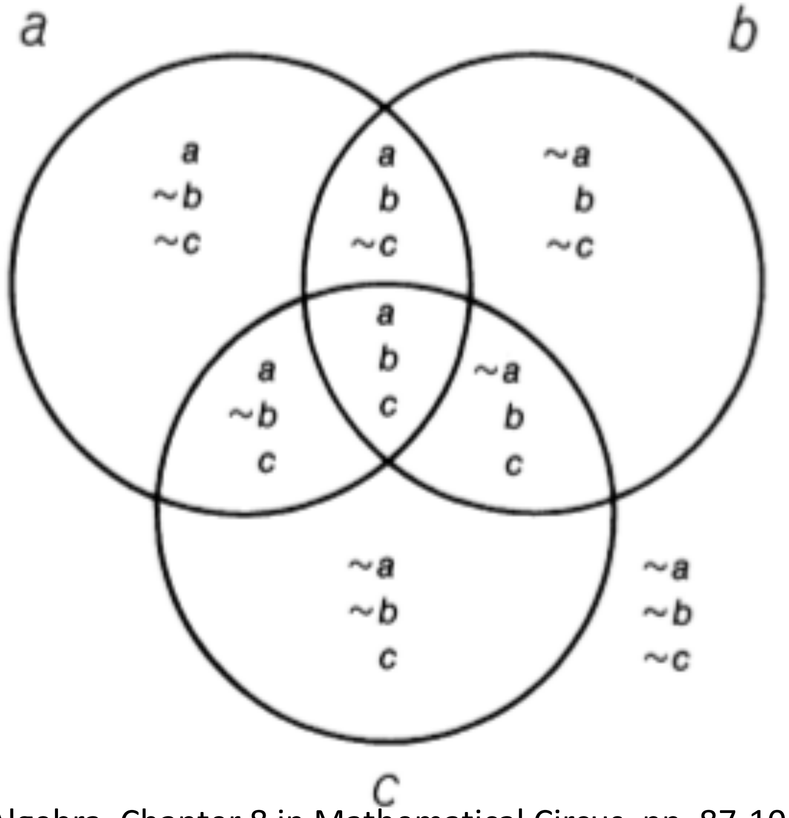


Thus the area marked a, ~b, c represents Abner's and Charley's having martinis while Bill does not.

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