ARTIFICIAL INTELLIGENCE

The Very Idea

Vasant G. Honavar
Dorothy Foehr Huck and J. Lloyd Huck Chair in Biomedical Data Sciences and Artificial Intelligence
Professor of Data Sciences, Informatics, Computer Science, Bioinformatics & Genomics and Neuroscience
Director, Artificial Intelligence Research Laboratory
Director, Center for Artificial Intelligence Foundations and Scientific Applications
Director of Strategic Initiatives, Institute for Computational and Data Sciences
Pennsylvania State University

vhonavar@psu.edu
http://faculty.ist.psu.edu/vhonavar
http://ailab.ist.psu.edu

On computation

- Thomas Hobbes, the British philosopher claimed that reasoning or thinking – is a form of computation

- Gottfried Leibniz, the German philosopher, mathematician and scientist dreamed of machines that could reason

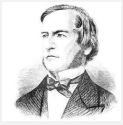
- George Boole, the British logician developed Boolean Algebra – a language for expressing assertions and deciding their Truth based on the Truth of assumptions

- David Hilbert, the German philosopher posed the decision problem: Is there an algorithm that can decide whether a claim is provable from assumptions

- Alan Turing invented the Turing Machine to solve Hilbert's decision problem and formalized what it means to compute

- Now we look at computation in a bit more detail

Boolean Algebra



- Boolean algebra is like the algebra you learned in high school except
  - Variables in the expressions or formulae, instead of numbers, represent logical propositions that are either True or False
  - True is often denoted by 1 and False by 0
  - We use 1 and 0 interchangeably with True and False respectively
  - The arithmetic operations are replaced by the symbols $\wedge, \vee, \neg$ denoting logical AND, OR, and NOT operations

Semantics of AND, OR, and NOT

X	Y	$X \wedge Y$
False	False	False
False	True	False
True	False	False
True	True	True

X	Y	$X \vee Y$
False	False	False
False	True	True
True	False	True
True	True	True

X	$\neg X$
False	True
True	True

- $X \wedge Y$ is True if *both* X and Y are True
  - This generalizes to any number of variables
- $X \vee Y$ is True if either X or Y or *both* are True
  - This generalizes to any number of variables
- $\neg X$ is True if and only if X is False

Boolean algebra at work

- Suppose I want to say:
  - If it is sunny outside AND I have completed my work then (and only then) I will go for a run.
- To express this in Boolean algebra, I introduce three Boolean variables
  - S to represent "it is sunny outside"
  - W to represent "I have completed my work"
  - R to represent "I will go for a run"
- Now $R \equiv S \Lambda W$ consisely expresses what I wanted to say.

Boolean Algebra at work

- Suppose I want to say: "I get home early from work if I get to leave work early or the traffic is light"
  - Suppose
    - H denotes "I get home early from work"
    - T denotes "the traffic is light"
    - L denotes "I get to leave early"
  - Now I can say $H \equiv T \vee L$
- Suppose I want to say: "If (and only if) it is not sunny, I will play chess"
  - Suppose
    - S denotes "it is sunny" and
    - C denotes "I will play chess"
  - Now I can say $\neg S \equiv C$

Graphical notation for Boolean Logic operators

AND OR NOT

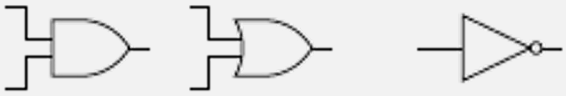


Figure 4.1: Graphical notation for 2-input AND, 2-input OR, and NOT Boolean operators

- The notation can be generalized to
  - multi-input AND and
  - multi-input OR operators
- $X \wedge Y \wedge Z$ is $True$ if and only if X, Y and Z are *all* $True$
- $X \vee Y \vee Z$ is $True$ if and only if one or more variable(s) among X, Y and Z are $True$

Boolean function composition

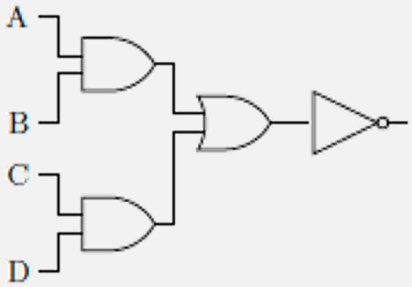


Figure 4.2: A compler Boolean function composed of $\wedge$, $\vee$ and $\neg$

$$\neg((A \wedge B) \vee (C \wedge D))$$

Boolean algebra - completeness

- The operators ∧,∨, ¬ are complete for Boolean algebra
  - Any conceivable Boolean function can be expressed using only the operations ∧,∨, ¬
  - The meaning of any complex Boolean formula can be understood in terms of
    - the meanings of the Boolean variables and
    - the meaning of the logical operators

A	B	C	D	$f(A,B,C,D)$
False	False	False	False	True
False	False	False	True	True
False	False	True	False	True
False	False	True	True	False
False	True	False	False	True
False	True	False	True	True
False	True	True	False	True
False	True	True	True	False
True	False	False	False	True
True	False	False	True	True
True	False	True	False	True
True	False	True	True	False
True	True	False	False	False
True	True	False	True	False
True	True	True	False	False
True	True	True	True	False

Boolean functions provide compact encodings of Truth Tables

$$f(A,B,C,D) \equiv \neg((A \wedge B) \vee (C \wedge D))$$

Modern computer design relies on composition of complex Boolean functions from elementary Boolean functions, e.g., ∧,∨, ¬

Top Hat Question

- Suppose $A = True, B = False, C = True, D = False$
- Is $\big(A \wedge C \wedge (\neg B \vee D)\big)$ True or False?

Exercise

$(A \wedge C \wedge (\neg B \vee D))$

Suppose we denote
True by 1

False by 0

A	B	C	D	$(A \wedge C \wedge (\neg B \vee D))$
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Exercise

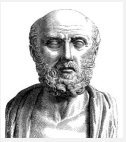
$(A \wedge C \wedge (\neg B \vee D))$

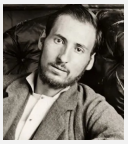
Suppose we denote
True by 1

False by 0

A	B	C	D	$(A \wedge C \wedge (\neg B \vee D))$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

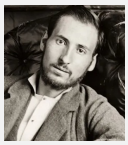
Minds, Brains, and Neurons



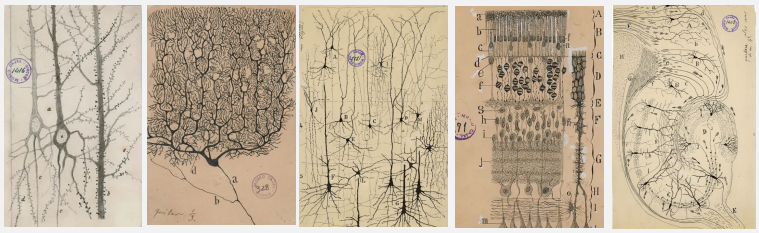


- While his contemporaries, including Aristotle, believed that the mind resided in the heart, Hippocrates argued, over 2500 years ago, that the brain is the seat of thought, sensation, emotion and cognition
- Spanish anatomist Santiago Ramón y Cajal (1852-1935) who came to be considered the father of neuroscience
  - Observed the first images of neurons using a staining technique invented by Golgi
  - Created more than 2,900 drawings of neurons and neural circuits
  - Observed that neurons are the building blocks of brains
  - Noted that insect neurons sometimes exceeded the complexity of human neurons

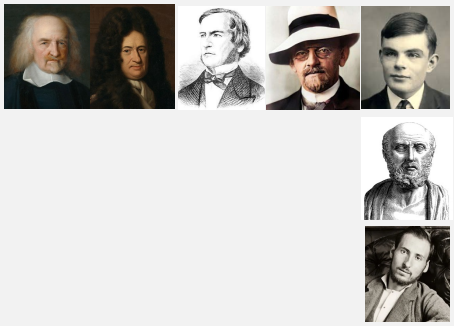
The birth of neuroscience



- Santiago Ramón y Cajal in the early 1900s
  - Introduced connectionism
    - The idea that cognitive abilities are a function of the way neurons are connected to form networks
    - The basis of deep neural networks in modern AI

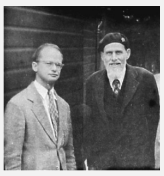


From Boolean Logic to McCulloch-Pitts Neuron

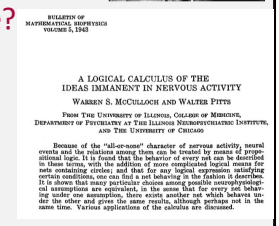


- Hobbes, Leibniz, Hilbert, Turing: Cognition can be modeled by computation
- Hippocrates: Brain is the substrate of the mind
- Cajal:
  - Neurons are the building blocks of brains
  - Cognition is a function of neural networks
- Then the logical question is what is it that neurons and networks of neurons compute?

From Boolean Logic to McCulloch-Pitts Neurons



- UIC Professor and neurophysiologist Warren McCulloch became friends with a homeless teenager, a self-taught logician Walter Pitts working odd jobs at U of Chicago
- McCulloch and Pitts
  - Asked What and how does the brain compute?
  - Introduced the first mathematical model of a neuron – The McCulloch-Pitts Neuron
  - Established the computational power of networks of networks of neurons
- Initially ignored, the work had a major impact on Von Neumann's design of the first digital computer
- Today's computers are largely based on Von Neuman's design



BULLETIN OF
MATHEMATICAL BIOPHYSICS
VOLUME 5, 1943

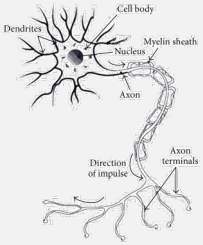
A LOGICAL CALCULUS OF THE
IDEAS IMMANENT IN NERVOUS ACTIVITY

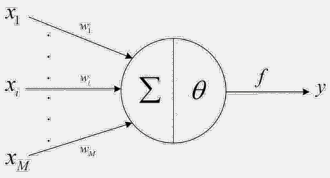
WARREN S. MCCULLOCH AND WALTER PITTS

FROM THE UNIVERSITY OF ILLINOIS, COLLEGE OF MEDICINE,
DEPARTMENT OF PSYCHIATRY AT THE ILLINOIS NEUROPSYCHIATRIC INSTITUTE,
AND THE UNIVERSITY OF CHICAGO

Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing circles; and that for any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes. It is shown that many particular choices among possible neurophysiological assumptions are equivalent, in the sense that for every set behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time. Various applications of the calculus are discussed.

McCulloch-Pitts Neurons



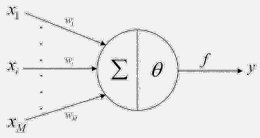


- A neuron receives input from other neurons through synapses
- When the voltage inside the cell body exceeds a threshold, the neuron fires
- The signal travels down the axon and reaches its destination
- The firing releases the charge accumulated in the cell body …

$$y = 1 \; if \sum_{i=1}^{M} w_i \, x_i \geq \theta$$

$$y = 0 \text{ otherwise}$$

McCulloch-Pitts Neuron: Example



$$y = 1 \; if \sum_{i=1}^{M} w_i \, x_i \geq \theta$$

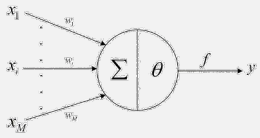
$$y = 0 \text{ otherwise}$$

Suppose $M = 2$, $w_1 = 1$, $w_2 = 1$, and $\theta = 2$

x_1	x_2	$s = w_1 x_1 + w_2 x_2$	$s \geq \theta$?	y
0	0	0	*No*	0
0	1	1	*No*	0
1	0	1	*No*	0
1	1	2	*Yes*	1

- What logical function does the neuron compute?
- Logical AND

McCulloch-Pitts Neuron: Example



$$y = 1 \; if \sum_{i=1}^{M} w_i \, x_i \geq \theta$$

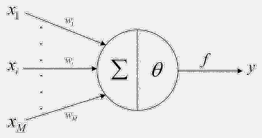
$y = 0$ otherwise

Suppose $M = 2$, $w_1 = 1$, $w_2 = 1$, and $\theta = 1$

x_1	x_2	$s = w_1 x_1 + w_2 x_2$	$s \geq \theta$?	y
0	0	0	*No*	0
0	1	1	*Yes*	1
1	0	1	*Yes*	1
1	1	2	*Yes*	1

- What logical function does the neuron compute?
- Logical OR

McCulloch-Pitts Neuron: Example



$$y = 1 \; if \sum_{i=1}^{M} w_i \, x_i \geq \theta$$

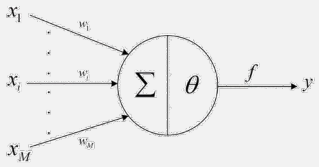
$$y = 0 \text{ otherwise}$$

Suppose $M = 1$, $w_1 = -1$, and $\theta = -0.5$

x_1	$s = w_1 x_1$	$s \geq \theta$?	y
1	-1	*No*	0
0	0	*Yes*	1

- What logical function does the neuron compute?
- Logical NOT

McCulloch-Pitts Neurons



$$y = 1 \; if \; \sum_{i=1}^{M} w_i \, x_i \geq \theta$$

$$y = 0 \text{ otherwise}$$

- A McCulloch Pitts neuron with the appropriately chosen weights and threshold can compute
  - Logical AND of its Boolean inputs
  - Logical OR of its Boolean inputs
  - Logical NOT
- Boolean algebra tells us that any Boolean function can be constructed using AND, OR and NOT
- An appropriately wired network of McCulloch Pitts neurons can realize any Boolean function

Exercise: McCulloch-Pitts Neuron

$$y = 1 \; if \sum_{i=1}^{M} w_i \, x_i \geq \theta \qquad y = 0 \text{ otherwise}$$

Suppose $M = 3$, $w_1 = 1$, $w_2 = 1$, $w_3 = 1$ and $\theta = 2$

x_1	x_2	x_3	$\sum_{i=1}^{M} w_i \, x_i$	$\sum_{i=1}^{M} w_i \, x_i \geq \theta$	y
0	0	0	0	No	0
0	0	1	1	No	0
0	1	0	1	No	0
0	1	1	2	Yes	1
1	0	0	1	No	0
1	0	1	2	Yes	1
1	1	0	2	Yes	1
1	1	1	3	Yes	1

Exercise: McCulloch-Pitts Neuron

$$y = 1 \; if \sum_{i=1}^{M} w_i \, x_i \geq \theta \qquad y = 0 \text{ otherwise}$$

Suppose $M = 3$, $w_1 = 1$, $w_2 = 1$, $w_3 = 1$ and $\theta = 2$

x_1	x_2	x_3	$\sum_{i=1}^{M} w_i \, x_i$	$\sum_{i=1}^{M} w_i \, x_i \geq \theta$	y
0	0	0	0		
0	0	1	1		
0	1	0	1		
0	1	1	2		
1	0	0	1		
1	0	1	2		
1	1	0	2		
1	1	1	3		

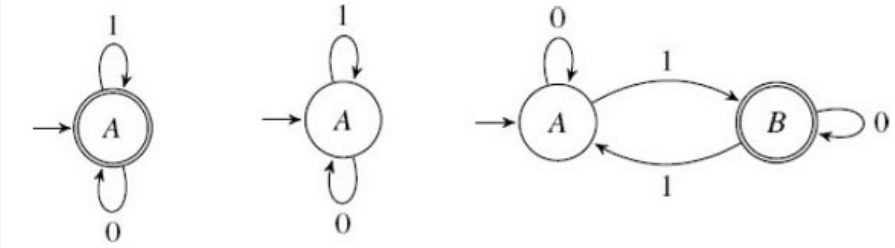
From Boolean Functions to Finite Automata

- If we add a time delay between input and output of Boolean circuits and wire them together to get finite automata that are more powerful than Boolean circuits

Finite automata

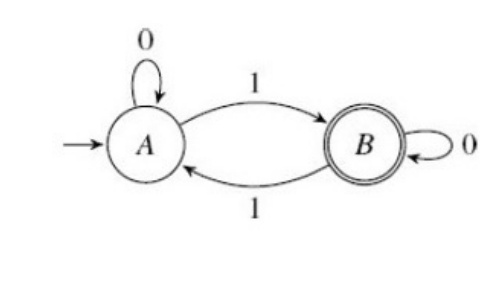
- Can be in one of a finite set of possible states including
  - the start state
  - one or more accept state(s)
- Input to the machine is a string or a sequence of symbols over a finite alphabet
- The machine starts in the start state and reads the first symbol of the input string
- A state transition table tells the machine what state to move to given the state it is currently in and the symbol it has just read
- The process halts when the machine reaches the end of its input string
- The machine `accepts' the string if the machine halts in an `accept state'. Otherwise the machine rejects the string

Examples of finite automata



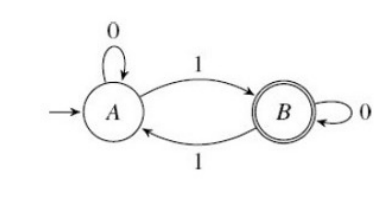
- We denote “accept” states by concentric circles
- The first automaton accepts all strings made of 0s and 1s
- The second automaton accepts no strings
- What does the third automaton do?

Exercise: What strings does the automaton accept?



- What happens with string 101 as input?
- What happens with string 100 as input?
- What about with string 1011 as input?

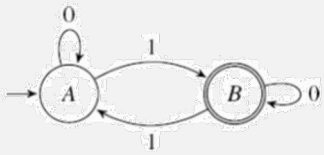
The automaton accepts strings with an odd number of 1s



- Suppose the input string is always finite
- We switch states only when the machine reads a 1
- We start in a non-accepting state A
- So long as the string is finite, the machine must end up in
  - A after reading an even number of 1s
  - B after reading an odd number of 1s
- What if the input string is infinite? The machine would run for ever.

Language of a finite automaton

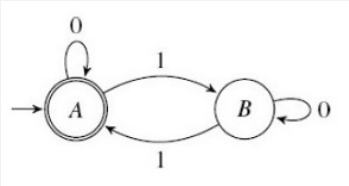
- The set of all strings accepted by a machine is called the language of the machine
- What is the language of the finite automaton shown below?



- The set of all strings over the alphabet $\{0,1\}$ that contain an odd number of 1's

Language of a finite automaton

- The set of all strings accepted by a machine is called the language of the machine
- What is the language of the finite automaton shown below?



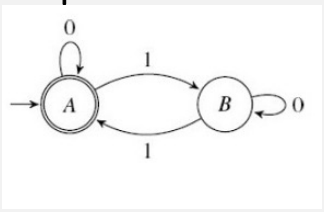
- The set of all strings over the alphabet $\{0,1\}$ that contain an even number of 1's

Language of a finite automaton

- The set of all strings accepted by a machine is called the language of the machine
- The languages accepted by finite automata, are called regular languages
- Can you give an example of a regular language?
  - The set of all binary strings
  - The set of binary strings with an odd number of 1s
  - …

Finite automaton solves decision problems

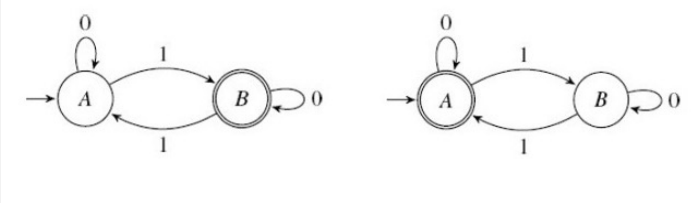
- Decision problem: Is the input string belong to the language of the automaton or not?
  - Answer can be Yes or No



- Does the automaton shown accept
  - 110
  - 101
  - 100
  - 111
  - 1101
  - 1001

Finite automaton solves decision problems

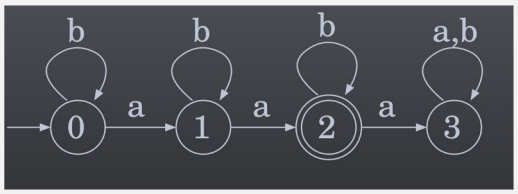
- Decision problem: Is the input string belong to the language of the automaton or not?
  - Answer can be Yes or No
  - Interchanging the accept and non-accept states of the machine corresponds to negating the question



  - There exist languages whose membership cannot be decided by finite automata but can be decided by more complex machines, e.g., Turing Machines

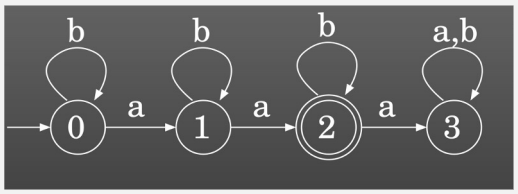
Exercise

What is the language of the finite automaton shown below?



Exercise

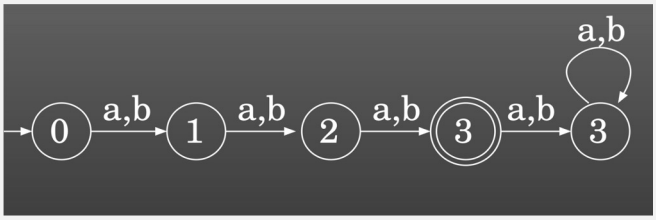
What is the language of the finite automaton shown below?



The set of all strings over the alphabet $\{a, b\}$ that contain exactly two occurrences of a anywhere in the string

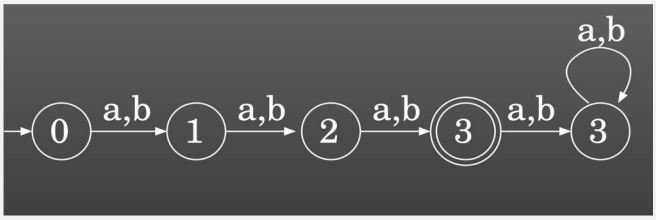
Exercise

- What is the language of the automaton shown below?



Exercise

What is the language of the automaton shown below?



The set of all strings of length 3

Exercises

- Construct a finite automaton that accepts all strings over alphabet $\{0, 1\}$ that end with a 0.
- Construct a finite automaton that accepts all strings over alphabet {0, 1} that start with a 0 and end with a 0.
- Construct a finite automaton that accepts all strings over the alphabet $\{a, b\}$ that end with abb
- Construct a finite automaton that accepts all strings over $\{0, 1\}$ that begin with 111
- Construct a finite automaton that accepts all strings $\{0, 1\}$ that begin or end with 111
- Construct a finite automaton that accepts all strings $\{0, 1\}$ that begin and end with 111

Turing Machine

- Hilbert and Ackerman (1928)
  - Decision Problem: "Is there an effective procedure, that allows us to decide, by means of finitely many operations, whether any given theorem is provable from a given set of premises?"
- Alan Turing (1936)
  - What is an effective procedure?
  - An effective procedure is an algorithm – step-by-step sequence of instructions that can be executed by a human or a machine
  - What kind of machine?
  - Turing Machine!

Turing's observations

Based on his observation of human computers at work, Turing concluded that a computer must be able to:

- Read the inputs of the computation (e.g., "455 + 376")
- Execute the appropriate algorithm
  - A sequence of simple steps like adding two digits in the 1's position
  - or steps in the computation which Turing called the states of mind of the (human) computer
  - which later got abbreviated as simply states
- Write results to keep track of the results of each stage.

Turing's observations

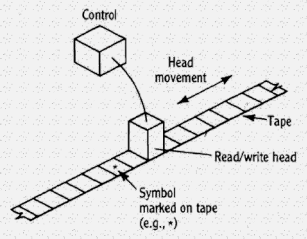
- The two-dimensional nature of the paper is irrelevant
- There is no reason to read more than one symbol at a time
- There is no reason to write more than one symbol at a time
- There is a need for the computer to step through states that keep track of the stage of the computation being carried out
  - Example: the digits in the one's position have been added and carry written

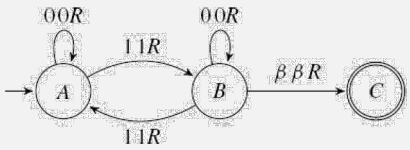
Turing's claim

- An effective procedure is simply an algorithm that tells a machine exactly what to do at any given step
- From the standpoint of executing such an algorithm human and the machine are functionally equivalent
  - Given the same input, they both can be seen as carrying out identical steps ending up with identical outputs
- The physical substrate used to implement the machine is immaterial
- All that is needed is the ability to manipulate symbols based on the syntax of the input
- One can build computers out of silicon, tinker toys, …

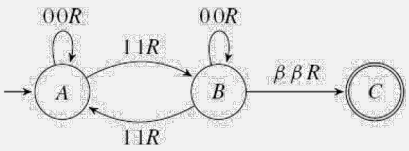
From Finite Automata to Turing Machine

- A finite automaton can be designed such that
  - The machine changes its state based on
    - Its current state and the input symbol that is read at the current position of the head
    - Writes a symbol on the tape
    - Moves the read/write head to the left or right or stays where it is





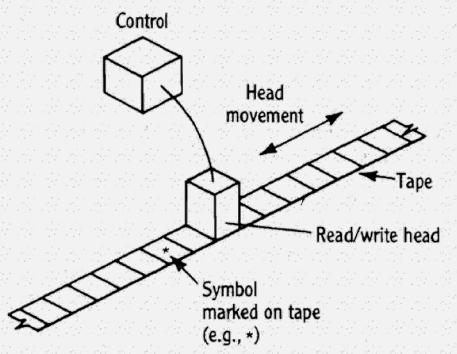
From Finite Automata to Turing Machine



- In state A, if you read a 0, write a 0, move R, and stay in A
- In state A, if you read a 1, write a 1, move R, and enter B
- In state B, if you read a 1, write a 1, move R, and enter A
- In state B, if you read a 0, write a 0, move R, and stay in B
- In state B, if you read a blank (β), write a blank, move R, and enter C

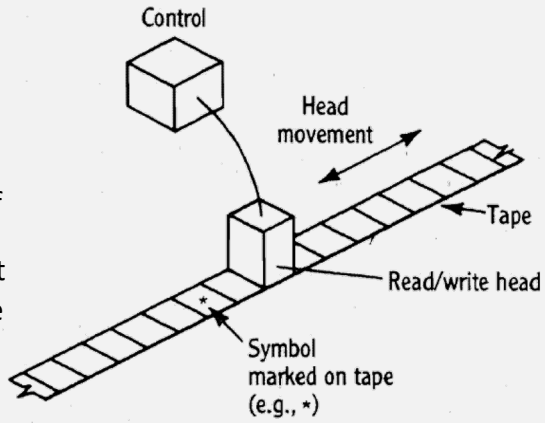
Turing Machine

- The tape is assumed to be infinite in both directions
- Ensures that there is enough space to write intermediate results
- At any given step
  - only a finite portion of the tape is used
  - the rest of the tape is blank



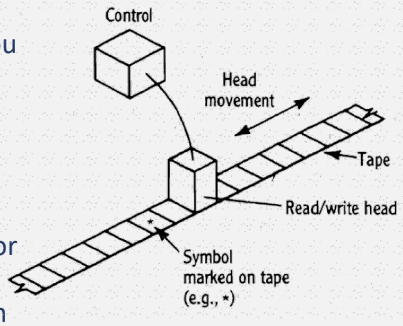
Universal Turing Machine

- Turing showed that there is a Universal Turing Machine that can simulate any other Turing Machine
  - Input: The program of the target Turing machine, and its input
  - Output: Output of the target machine



Turing Machine

- Defines an effective procedure
  - ✓An effective procedure is an algorithm executed by the Turing machine
- If you agree with Turing's definition, you have no choice but to accept that any algorithm whatsoever can be implemented by a Turing Machine
- If you disagree with the definition, the burden is on you
  - to devise an alternative definition or
  - at least show an example of a process that intuitively looks like an algorithm but cannot be realized by a Turing Machine



Church-Turing Thesis

- Many tried to investigate alternative formalizations of effective procedures
  - Alonzo Church proposed λ −calculus
  - Emil Post proposed Post productions
  - Andrey Markov proposed Markov Algorithms
  - Stephen Kleene proposed general recursive functions

- All of them were shown to be equivalent to Turing Machines
  - Church-Turing Thesis: Any computation that any of the formalisms could perform could be performed by the Turing Machine and vice versa

Church-Turing Thesis

- All the attempts at alternative formalizations of effective procedures were shown to be equivalent to Turing machines
  - Any computation that any of the formalisms could perform could be performed by the Turing Machine and vice versa
- Church-Turing Thesis: Any function that can be computed by an algorithm can be computed by a Turing Machine
  - What does it mean for a function to be computed by an algorithm?
    - Executing a finite sequence of instructions on a finite input producing a finite output
    - Transforming one finite sequence of letters into another sequence of letters according to a set of rules in a finite number of steps

Implications of Church-Turing Thesis

- Any computer that is sufficiently powerful (Turing equivalent) can execute any algorithm or program whatsoever
- Any program for one computer can be translated to an equivalent program for another computer
  - Why?
- Programs written in one programming language can be translated into programs written in any other programming language
  - Why?
- The design of modern computers, programming languages, software .. rests on Church-Turing Thesis

Limits of computing machines

- There exist problems that are unsolvable by Turing machines
  - Halting problem: Deciding whether an arbitrary program eventually terminates on *all* inputs
  - Theorem proving in logic: Deciding whether a theorem logically follows from a set of axioms
    - If the theorem is true, there is a finite proof
    - If the theorem is false, it is possible that the machine runs for ever
      - there is no way to tell if it is
        - because the theorem is false or
        - because the machine needs more time
  - Hilbert's decision problem is semi-decidable

Can we build computers that are more powerful than Turing machines?

- Maybe, if we can allow the machines to take advantage of something Turing machine can't
  - Infinite precision numbers
    - There is no physical substrate capable of supporting infinite precision
  - Quantum phenomena – machines that make use of quantum as opposed to deterministic states
    - There is no evidence that quantum computers can circumvent the limitations of Turing machines although they can be far more efficient than conventional computers on some problems
- We don't know if we can build computers that are more powerful than Turing Machines
- No one has succeeded to date in doing so

Implications of Church-Turing Thesis for AI

- So long as intelligence has an algorithmic description, there are many ways to realize it that are functionally equivalent
  - Brains
  - Digital computers
  - Quantum computers
  - DNA computers
  - And many other types of computers….

Implications of limits of computation for AI

- Position 1: To the extent that intelligence can be modeled by computation, there will be some truths that are unknowable by both humans and machines
- Position 2: Powers of mind exceed those of computers – Tantamount to saying that minds have no finite algorithmic description
- Position 3: Semi-decidability is a property of machines that reason deductively. Most of our knowledge of the world comes from inductive and not deductive methods so the limits of deductive methods are only partially relevant to AI