Readings Summary

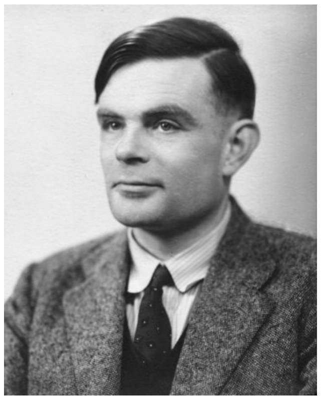
Today's assigned "reading" was to watch two talks from the History Panel session of the Computation and the Transformation of Practically Everything MIT150 symposium

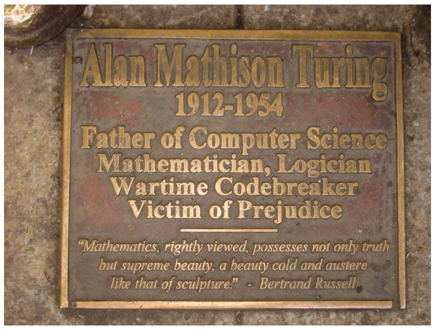
- Ed Lazowska tells us that
  - 1969 was an eventful year
  - Butler Lampson identifies three stages of computing:
    1. simulation (1940s–1970s)
    2. communication (1980s–2000s)
    3. embodiment (now)
  - computer scientists build complete systems (of ever increasing scale, integration and complexity)
- Pat Winston presents his view of both the history and "the way forward" in artificial intelligence, including his
  - strong story hypothesis
  - strong perception hypothesis
  - strong social animal hypothesis

Today's Learning Goals

1. To share the "big picture" story of computation, in terms of:
   - theory
   - realization in actual computing devices
2. To identify questions to ask of any knowledge representation (KR) scheme:
   - What "language" is used?
   - What knowledge is explicit?
   - What inference mechanism(s) are available?
3. To introduce the artificial intelligence approach to building and understanding intelligent systems

Alan Turing


June 23, 1912 – June 7, 1954


Alan Mathison Turing
1912-1954
Father of Computer Science
Mathematician, Logician
Wartime Codebreaker
Victim of Prejudice
"Mathematics, rightly viewed, possesses not only truth but supreme beauty, a beauty cold and austere like that of sculpture" - Bertrand Russell
Statue Plaque in Sackville Park

http://en.wikipedia.org/wiki/Alan_Turing

On December 24, 2013, the Queen granted Alan Turing a pardon under Britain's "Royal Prerogative of Mercy" for his 1952 conviction for the gross indecency of homosexuality. Following his conviction, he was chemically castrated. In addition, he lost his security clearance, which impacted his ability to continue the code-breaking work he was doing and his ability to travel. Alan Turing died in 1954 after eating an apple laced with cyanide. Some have speculated that the Apple logo, an apple with a bite out of it, is in honour of Turing. [1]

Models of Computation

Models of Computation

Bottom up:

- switch
- logic gate
- combinational circuit
- sequential circuit
- Finite State Machine (FSM)

The roots of computer science stem from the study of many alternative mathematical "models" of computation, and from the study of the classes of computations they can represent

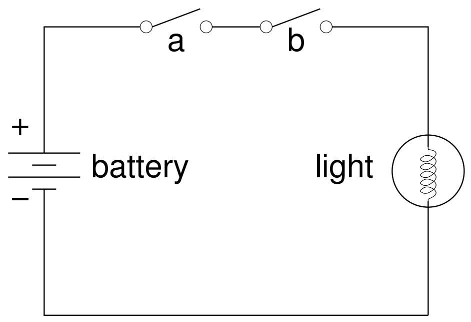
The elusive goal is to find an "ultimate" model, capable of representing all practical computations…

[1] Steve Jobs is on record as having said there was no connection between Turing and the Apple logo.

Question: Is a FSM the ultimate digital computing device?

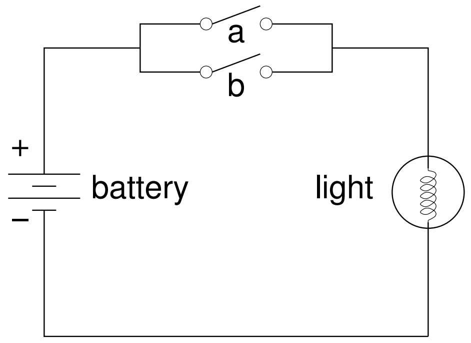
Let's look a bit more at switches…

"AND" Light Switch



switch a	switch b	light
closed	closed	on
closed	open	off
open	closed	off
open	open	off

"OR" Light Switch



switch a	switch b	light
closed	closed	on
closed	open	on
open	closed	on
open	open	off

Binary "Values"

on	off	
closed	open	
true	false	
T	*F*	*Formal logic*
high	low	
positive	negative	
1	*0*	*Boolean algebra*

The technology of switches has changed several times in my lifetime: (electro-mechanical) relays (used in telephone switching networks); vacuum tubes; (single canistor) transitor circuit elements; integrated circuits; and the myriad of subsequent transistor technologies including today's choice, CMOS.

The technology of switches continues to change (i.e., photonics and biological computing).

Switch Property "Wishlist"

- Small, simple, reliable
- Fast
  - High frequency (in ON/OFF cycles per second)
  - Rapid state transition ON to OFF and OFF to ON (i.e., minimal time in intermediate "invalid" state)
- Inexpensive (i.e., switches and connecting "wires" mass produced)
- Isolated (i.e., operation doesn't interfere with neighbouring switches)
- Low power consumption

The switch is the building block for digital computation. The good news is that with switches we can build devices that compute. What computation can be done with, say, 1–10 switches? Not that much (except for simple student exercises in introductory courses). What about with hundreds, thousands, or perhaps millions of switches? Well, things certainly did get more interesting when this became possible. Things really are interesting, however, when one deals with many billions of switches at a time (as is the case with current personal computers). What the future holds remains unknown. Moore's law continues with current technology. At the same time, there are other technologies with the potential to continue the exponential growth in switch density realizable in computational devices.

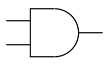
Consider DNA. DNA is the building block of life. DNA molecules are long but they are made up of just four bases: guanine, adenine, thymine and cytosine (denoted respectively as G, A, T and C). Understanding the role that DNA plays in all living things was one of the major intellectual accomplishments of the last century.

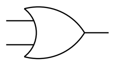
What is known about DNA can be considered profoundly disappointing or incredibly exciting, depending on one's perspective. It perhaps is disappointing if one was expecting something more magical or complex. On the other hand, it is incredibly exciting to begin to unravel what can be accomplished with DNA and the thousands and thousands of genes (which are sub-sequences of DNA) that code for proteins.

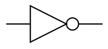
Switches are the computer scientist's DNA. For us, it's incredibly exciting to understand how to assemble switches into more complex and abstract structures (gates, combinational circuits, sequential circuits, FSMs) upon which modern computation is based.

Logic Gates

AND OR NOT Logic Gates







AND

input		output
0	0	0
0	1	0
1	0	0
1	1	1

NOT

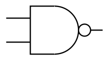
input	output
0	1
1	0

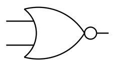
OR

input		output
0	0	0
0	1	1
1	0	1
1	1	1

AND, OR and NOT form a “universal” set of logic gates. That is, any combinational circuit can be implemented using these three and only these three logic gates.

Two More Logic Gates





NAND

input		output
0	0	1
0	1	1
1	0	1
1	1	0

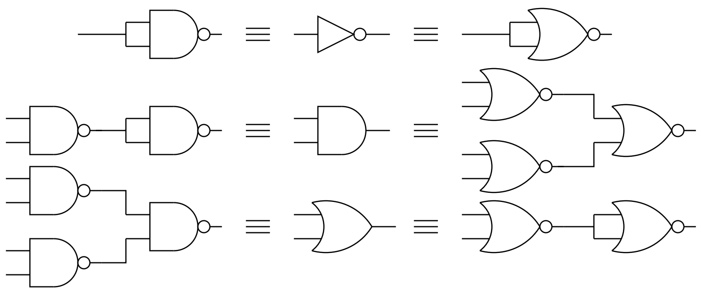
NOR

input		output
0	0	1
0	1	0
1	0	0
1	1	0

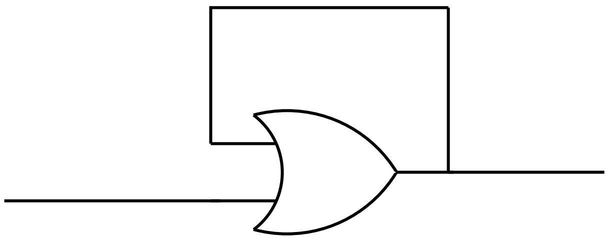
Two input NAND and NOR gates each also are "universal," as we now show.

2-input NAND and NOR Gates Each are Universal

NOT, AND and OR gates (middle) can be implemented using only NAND gates (left side) or only NOR gates (right side)



What about this?



Example 1: Apollo Guidance Computer (AGC)

To date, the United States is the only country to have successfully conducted manned missions to the moon. There were six manned Apollo landings (between 1969 and 1972), as follows:

Apollo 11	July 20, 1969
Apollo 12	November 24, 1969
Apollo 14	February 5, 1971
Apollo 15	August 7, 1971
Apollo 16	April 27, 1972
Apollo 17	December 11, 1972

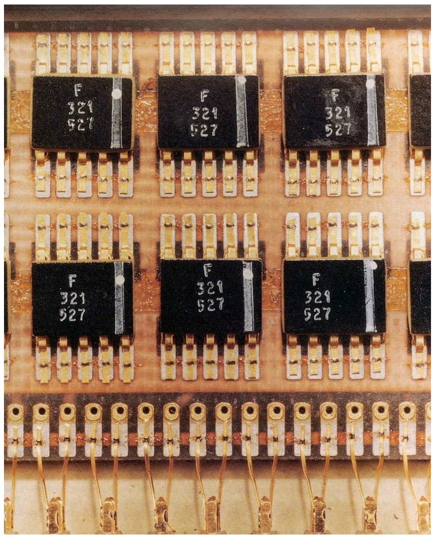
Each of these six flights to the moon had two AGCs, one in the Command Module and one in the Lunar Module. All six used the "Block II" version of the AGC. This version used 2,800 "single type" Integrated Circuits (ICs). Each IC had two 3-input NOR gates (for a total of 5,600 3 input NOR gates).

Example 1: Apollo Guidance Computer (AGC)

Invented by	MIT Instrumentation Laboratory
Manufactured by	Raytheon
Introduced	August, 1966
Discontinued	July, 1975
Processor technology	Discrete IC RTL based
Clock frequency	2.048 MHz
Memory	16–bit word
	2,048 words RAM (magnetic core memory)
	36,864 words ROM (core rope memory)
Power consumption	55W
Weight	32 kg (70 lb)
Dimensions	$61 \times 32 \times 17$ cm ($24 \times 12.5 \times 6.5$ inches)

http://en.wikipedia.org/wiki/Apollo_Guidance_Computer

Example 1 (cont'd): AGC Flatpack Integrated Circuits

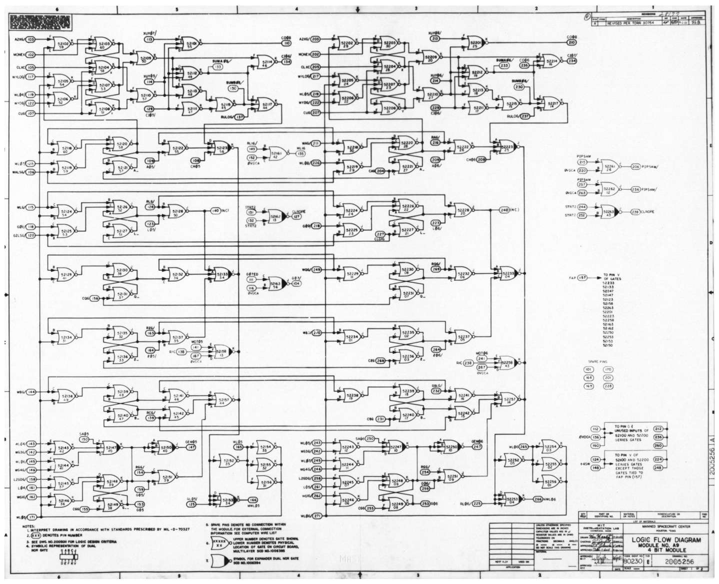


http://en.wikipedia.org/wiki/Apollo_Guidance_Computer

The "Block II" AGC ICs, manufactured by Fairchild Semiconductor, used resistor-transistor logic (RTL) in a "flatpack" arrangement. The ICs were inter-connected via wire wrap, and the wiring was then embedded in cast epoxy plastic.

Source: http://en.wikipedia.org/wiki/Apollo_Guidance_Computer

Example 1 (cont'd): AGC Module A9 Sheet 1 (of 2)



http://klabs.org/history/ech/agc_schematics/

Things to note in this schematic (as with others):

1. The only gate that appears in the design is a 3 input NOR gate. (Often, however, not all 3 inputs are used).
2. The design was drawn/checked/approved by individuals (whose signatures are found in the box in the lower right of the drawing).
3. Feedback loops exist. That is, there are examples where there exists a path from the output of a given gate back to an input of that same gate.

Four "Big Ideas" of Computer Science

Big Idea #1:

Finite State Machine (FSM)

Big Idea #2:
Turing Machine

Big Idea #3:
Computability

Big Idea #4:
Universality

Computability

Three attempts (in the first half of the 20th century) to formalize what it means for a function to be *"effectively calculable:"*

- λ–calculus (Alonzo Church)
- Turing machine (Alan Turing)
- General recursive (Alonzo Church, Stephen Kleene, J. Barkley Rosser)

All three formalisms were subsequently shown to be equivalent. A function is *"computable"* iff it can be computed by a Turing machine

"Effectively calculable" and *"computable"* need not be synonymous

The *Church–Turing hypothesis* states, *"everything effectively calculable is computable by a Turing machine"*

This cannot be formally proven. Nevertheless, it has near-universal acceptance (in Computer Science)

Universality

There exists a Turing machine that is able to simulate any other Turing machine. Such a machine is called a *universal Turing machine (UTM)*

This result, established by Turing in 1936, was considered astonishing at the time. We now take it for granted. But, many consider it to have been the fundamental theoretical breakthrough that led to the modern stored-program computer

Models of Computation: Summary

Formal models (computability, Turing machines, universality) provide the basis for modern computer science:

- Fundamental limits (on what can not be done, even given plenty of memory and time)
- Fundamental equivalence of computation models

- Representation of algorithms as data, rather than as machinery
- Programs, software, interpreters, compilers,...

There remain many practical issues to deal with:

- Cost: memory size, time performance, power consumption
- Programmability

Summary (cont'd):

- Computer science has theory and computer scientists build systems
- Switches are the "building block of life" in the world of digital computation
- The most general notion of *computability,* to date, is that of a Turing machine
- There are no technology gaps separating switches, from Turing machines, or from any current digital computer. There are "case–in–point" technologies for every step along the way
- Computers are the "Swiss army knife" of science. In particular, computers are ideal laboratory animals for testing theories of intelligence
- As Turing wrote in 1950, "We can only see a short distance ahead, but we can see plenty there that needs to be done"

Methodological Issues (Across Disciplines)

- What problems do each discipline address?
- What questions do each discipline ask about those problems?
- What tools and techniques do each discipline use to answer these questions?
- What defines "success" in each discipline?

AI deals with knowledge representation (KR). Intelligent systems need to represent both knowledge of the 3D world (external knowledge) and knowledge about themselves as problem solvers (internal knowledge).

Knowledge Representation (KR): Key Issues

Questions to ask of any KR scheme are:

1. What *"language"* is being used?
2. What *knowledge* is represented explicitly by a given set of expressions in the language?
3. What *inference mechanism* is available to make explicit knowledge which is implicit, but not yet explicit, in the given set of expressions?

Inference mechanisms can be thought of as methods to reason about (or calculate with) knowledge in order to make explicit that which is only implicit

Research Methodology

Artificial Intelligence

Artificial Intelligence...

...is the intelligent connection of perception to action

This definition of artificial intelligence is delightfully circular since the word "intelligence" appears twice. But, this is OK. Biology is the study of living things. Biology advances even though there never has been (and probably never will be) an accepted medical, legal and ethical definition of what it means for something to be alive.

Helen Keller

June 27, 1880 – June 1, 1968

with Anne Sullivan (circa 1930)

http://en.wikipedia.org/wiki/Helen_Keller
https://www.youtube.com/watch?v=Gv1uLfF35Uw

Consider the example of Helen Keller (1880–1968). In 1882, at 19 months of age, Keller caught a fever that was so fierce she nearly died. She survived but the fever left its mark — she could no longer see or hear. Because she could not hear she also found it very difficult to speak.

Her teacher was Anne Sullivan. Keller relied a great deal on Anne Sullivan, who accompanied her everywhere for almost fifty years. Without her faithful teacher Keller would probably have remained trapped within an isolated and confused world.

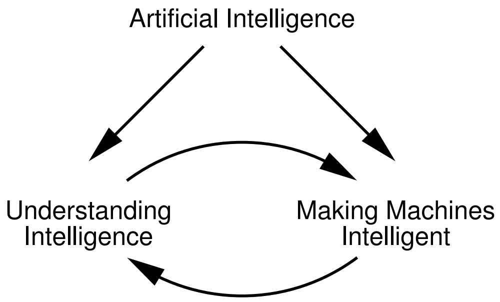
Given her life accomplishments, no one would suggest that Helen Keller was mentally disabled. But, her perceptual disabilities made it very difficult for her to learn. That she did learn is a great credit to her and to Anne Sullivan.

It is fair to say that someone like Helen Keller is blind (i.e., unable to see), deaf (i.e., unable to hear) and mute (i.e., unable to speak). It is not fair to call her dumb. (The word "dumb" has a double meaning in English. According to Webster, it means, "lacking the power of speech" and also "markedly lacking in intelligence: synonym *stupid*"). Given this, referring to someone who cannot speak as "dumb" now is considered offensive. In the same way, I would argue that referring to machines that cannot see or hear as "dumb" also is offensive. Intelligence requires both cognitive (i.e., mental) and perceptual abilities.

Here's a 3 minute, 1930 Vitaphone newsreel of Helen Keller with Anne Sullivan. The newsreel ends with Helen Keller speaking, "I am not dumb now."

There are two main motivations for being interested in artificial intelligence, a scientific one and an engineering one.

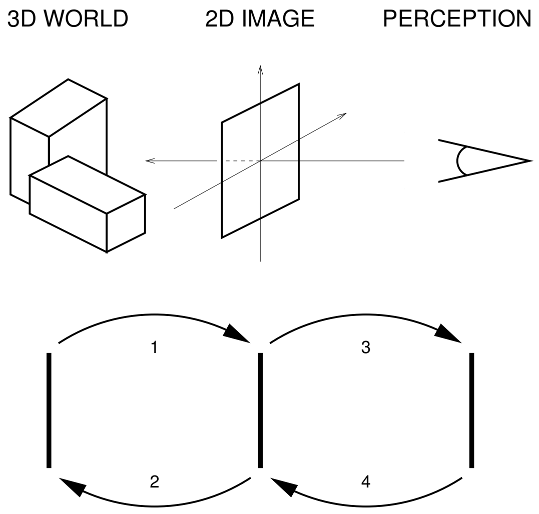
Artificial Intelligence



Let's look a bit deeper at computational perception, in this case computational vision.

The following diagram helps to identify various sub-fields related to computational vision.

Computational Vision: Domains and Mappings



The four numbered arrows identify four sub-areas.

Arrow 1:

Computer graphics deals with how the 3D world determines the 2D image. For objects of known shape, made of known materials, illuminated and viewed in a known way, computer graphics renders (i.e., synthesizes) the requisite image.

Arrow 2:

Computer vision deals with the inverse problem, namely given an image what can we say about the shape, and material composition of the objects in view and about the way they are illuminated and viewed.

Arrow 3:

Is perception of the 3D world direct? Our diagram takes view that visual perception of the 3D world is mediated by the 2D image. One can, of course, generate 2D images artifically that don't correspond to any possible 3D scene and study how these images are perceived. This, indeed is one of the things that perceptual psychologists do. Sometimes, "seeing is deceiving."

Arrow 4:

Different images can produce the identical perception. Examples of perceptual metamers are well known in areas such as lightness, colour, texture and motion. In general, arrow 4 deals with the question, "What is the equivalence class of images that give rise to identical perceptions?"

Computer Science: A Discipline with an Attitude

> The person who says it cannot be done should not interrupt the person doing it
>
> — Chinese proverb