What we have discussed in this course

COS116, Spring 2010

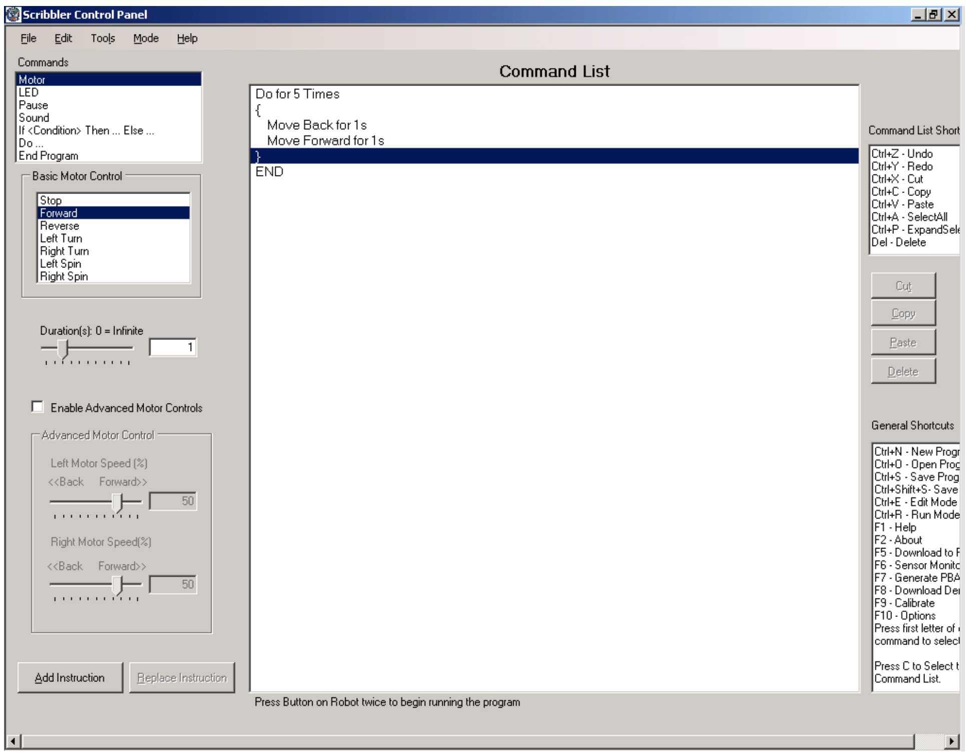
Adam Finkelstein

Roughly three parts of the course

Lectures 1-10, Lab 1-5: Expand notion of “computation”.

- Scribbler
- Pseudocode
- Game of life, cellular automata,
  physical systems (weather, twister..)
- Web, networks, websearch, datamining.
- Turing-Post programs (universal programs, undecidability)
- Digital sound and music

Controlling Scribbler's behavior



Scribbler Control Panel

(uses "pseudocode")

Pseudocode: Workaround for Computing's Tower of Babel; shares features with most programming languages. (Main features: basic arithmetic operations; "conditional branching" (if-then-else); loops (do for; do while ())

Steps in solving a computational task

- Design an algorithm: A precise,unambiguous description for how to compute a solution.
- Express algorithm in pseudocode.
- Turn pseudocode into computer program.

Creating new worlds ("simulation")

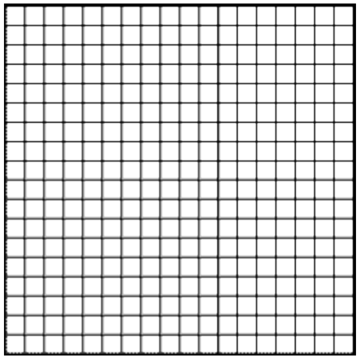
(game of life, weather, twisters…)

Steps:

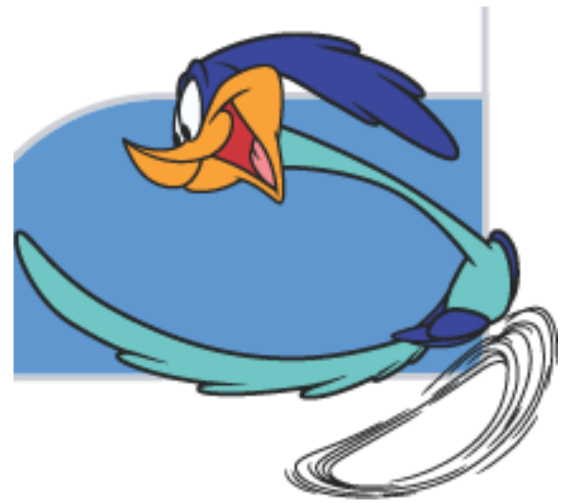
• Figure out the "rules" for particles' actions (how their 'state" evolves with time)

• Figure out how to write the code for changes undergone by a particle in one time step.

• Simulation = one big "Do for" loop.



How do we measure the "speed" of an algorithm?



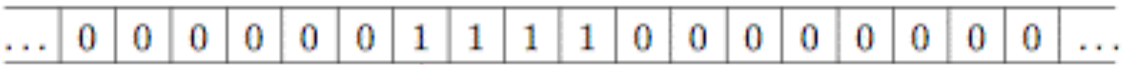
- Ideally, should be independent of:
  – machine
  – technology

Answer: Count number of elementary steps.

Example: Binary search on a sorted array of n numbers takes *4 log n* steps.

(Also studied other notions of "search" including data mining and web search.)

What are limits of computation?



- 1 dimensional unlimited scratchpad ("infinite")
- Only symbols are 0/1 (tape has a finite number of 1s)
- Can only scan/write one symbol per step
- Program looks like
- We believe this simple model can simulate all physically realizable computational models. ("Church Turing Thesis.")

1. PRINT 0
2. GO LEFT
3. GO TO STEP 1 IF 1 SCANNED
4. PRINT 1
5. GO RIGHT
6. GO TO STEP 5 IF 1 SCANNED
7. PRINT 1
8. GO RIGHT
9. GO TO STEP 1 IF 1 SCANNED
10. STOP

The Doubling Program

Examples of “undecidable” problems

- Given a starting configuration of Game of Life, and a specific cell index, say (11, 15), decided if that cell ever gets occupied.
- Given a program and an input for it, decide if the program ever halts when run on that input.
- Given a mathematical statement, decide if it has a proof in the standard axiomatic framework for math.

Other ideas encountered: proof by contradiction, self-reproducing programs, the possibility that many simply described systems may have no succinct “theories.”

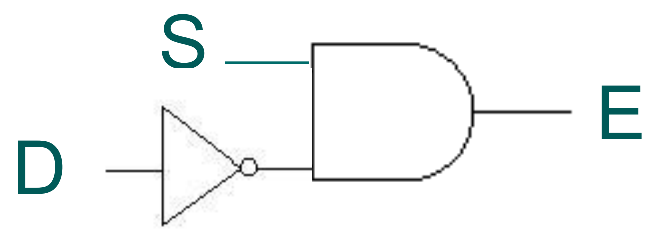
Lectures 11-16; Labs 6-7: Looking inside current computers

- Boolean logic
- Circuits (combinational, and sequential)
- Finite state machine (the “controller”)
- CPUs and computer organization.
- Silicon chips; microprocessors; Moore’s Law
- Caching and Multitasking

Boolean logic
(Three Representations)

Boolean Expression $E = S \text{ AND } \overline{D}$

Boolean Circuit



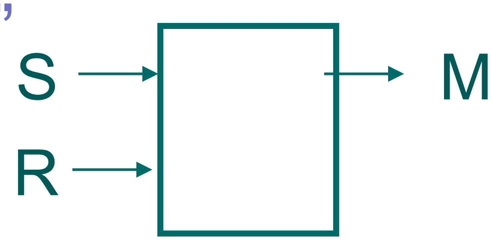
Truth table:

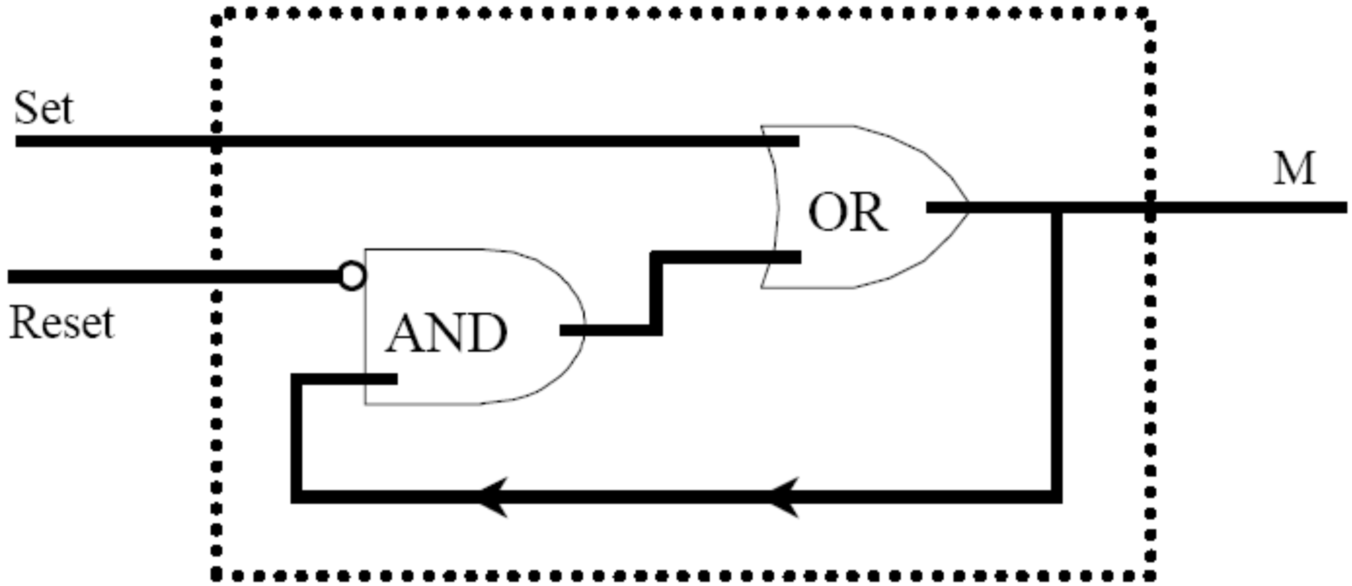
Value of E for every possible D, S.

TRUE=1; FALSE= 0.

D	S	E
0	0	0
0	1	1
1	0	0
1	1	0

How circuits get “memory”



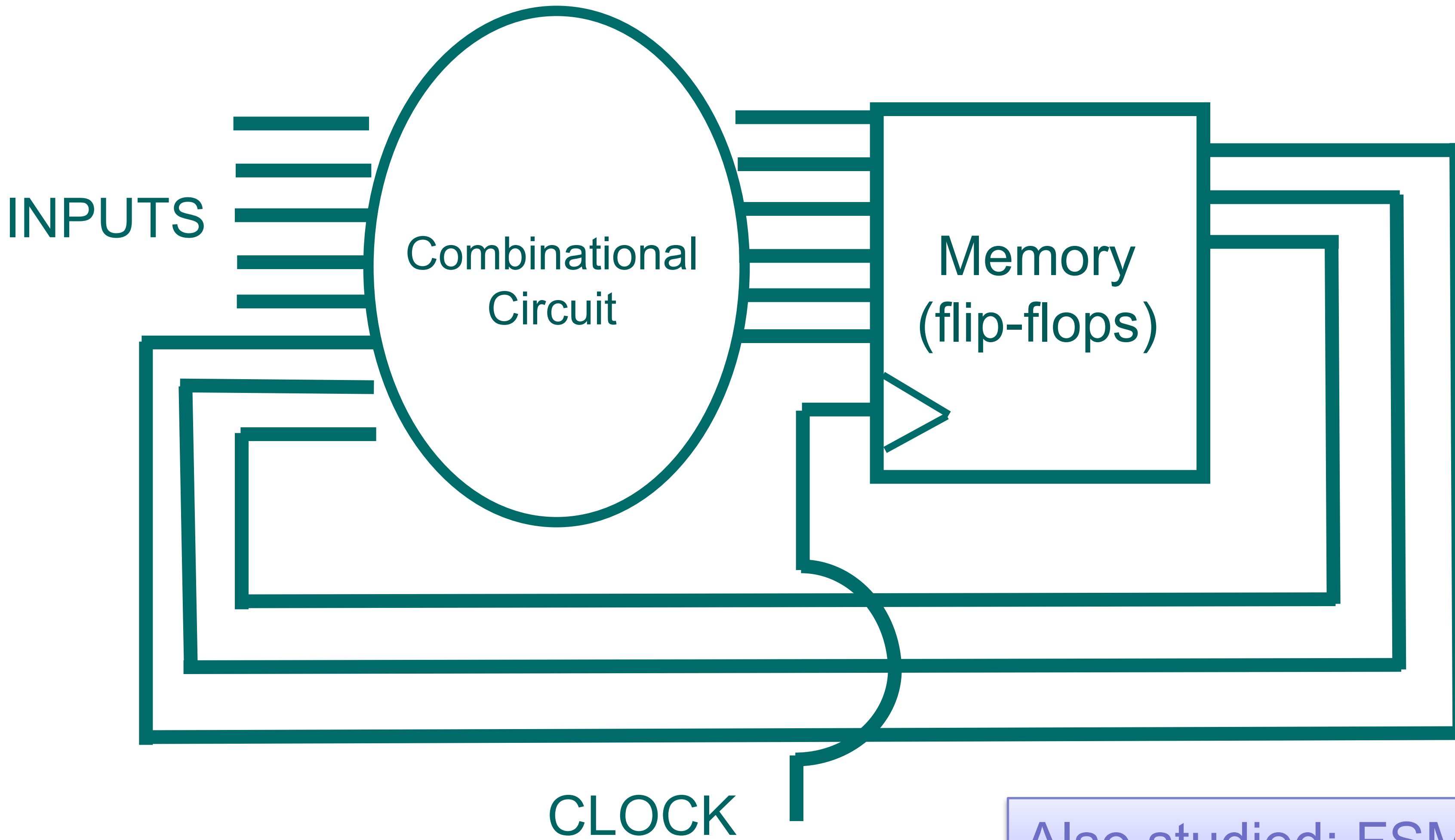


R-S Flip Flop

- M becomes 1 if Set is turned on
- M becomes 0 if Reset is turned on
- Otherwise (if both are 0), M just remembers its value

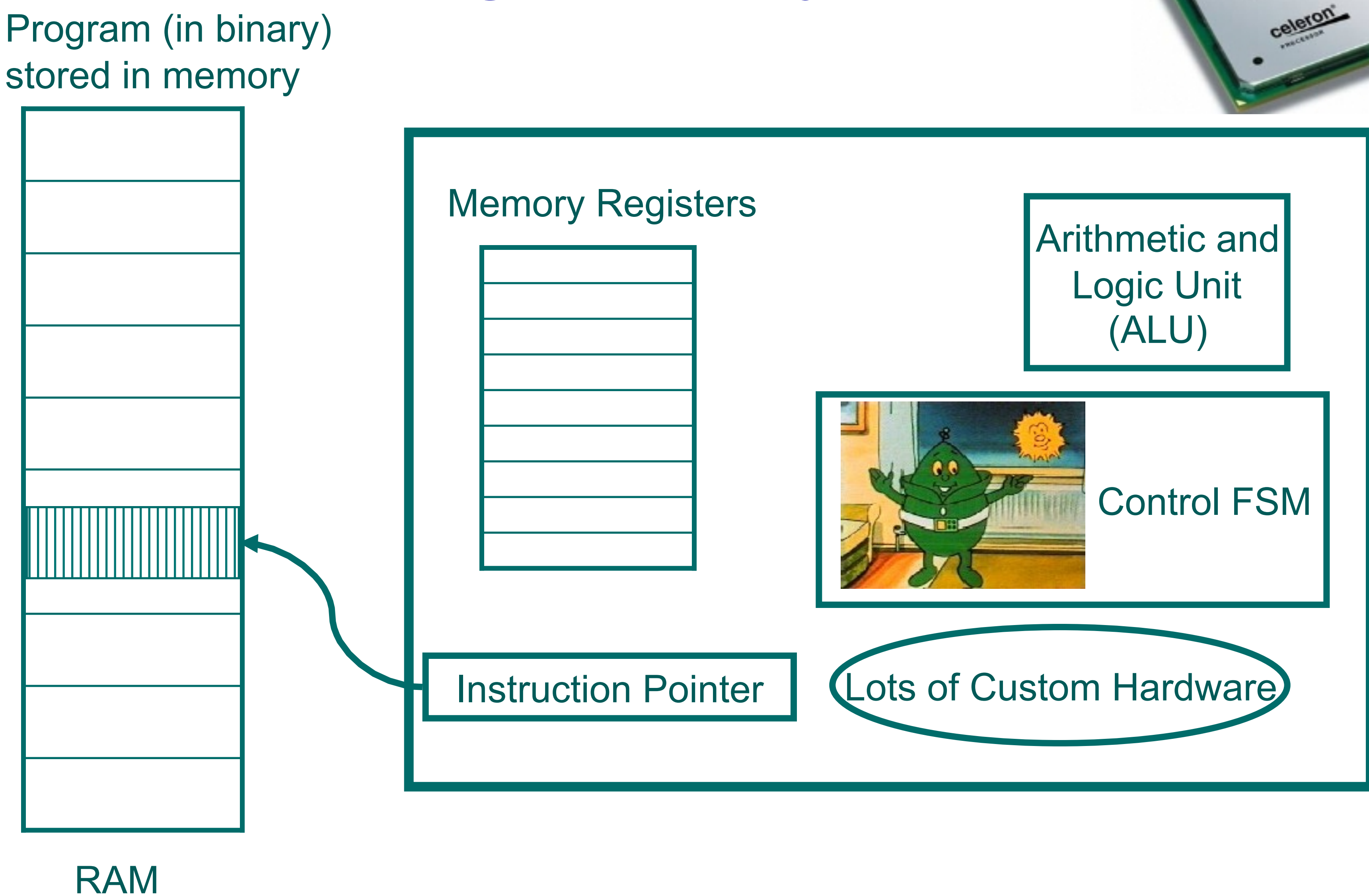
Synchronous Sequential Circuit

(aka Clocked Sequential Circuit)



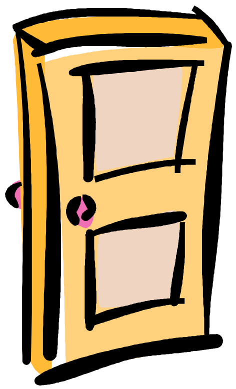
Also studied: FSMs

Modern Computer (simplified view): FSM controlling a memory bank



~~Librarian~~ Computer arrangement

Reserves / Disk



"Most popular" shelf:
20% most popular books

Memory

CPU

Top 4% / Cache

"80-20 Rule"

Often, today's computers have even more levels of caching

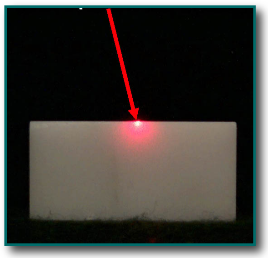
Internet: Main themes

1. Building reliability on top of unreliable protocols.
   (retransmission, timeout, etc.)

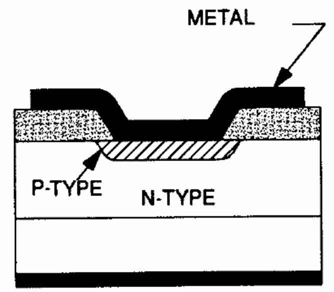
2. Decentralized control.
   (cs.princeton.edu : DNS
   physical routing)

3. Reliance on kindness of strangers.

Science behind modern computers



Lasers; used in chip manufacturing, fiber optic cables



Semiconductors: rely on quantum mechanics.
Chips manufactured by a “photography”-like technique.

Touched upon: Moore’s law (hows and whys)

Lectures 16-24; Labs 7-10: New Concepts that arose from study of computation

- WWW and the Internet
- Efficient computations, P vs NP, NP-completeness
- Cryptography; Zero Knowledge Proofs
- Viruses/Worms/Zombies/Cybersecurity
- Machine Learning
- Artificial intelligence.

The P vs NP Question

- P: problems for which solutions can be found in polynomial time (n^c where c is a fixed integer and n is "input size"). Example: Rumor Mill

- NP: Decision problems for which a "yes" solution can be verified in polynomial time.

- Question: Is P = NP?

  "Can we automate brilliance?"

  (Note: Choice of computational model --- Turing-Post, pseudocode, C++ etc. --- irrelevant.)

NP-complete problems: The "hardest" problems of NP.

Viruses and Worms

Automated ways of breaking in;

Use self-replicating programs.

Studied how and why people create these (“botnets”).

No real solution in sight except eternal vigilance

Cryptography

- Creating problems can be easier than solving them (eg “factoring”)
- Difference between seeing information and making sense of it (e.g., one-time pad, zero-knowledge proofs)
- Role of randomness in the above
- Ability of 2 complete strangers to exchange secret information (public key cryptosystems)

Machine learning, AI

Machine learning: less ambitious. Make computer do sort of intelligent tasks in very limited domains (understanding images, speech recognition, etc.)

Key idea: learning algorithm that is “trained” with large amounts of Data.

AI: try to create more general intelligence.

One measure of success: Turing Test.

Simulation argument for feasibility of AI.

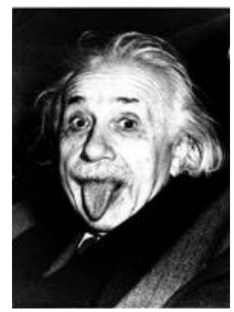
Searle’s argument why strong AI is impossible

Cryptography

Generally accepted fact about AI

Programming all necessary knowledge into computers is hopeless.
Only hope : General purpose Learning Algorithms

Many years of learning →



Approach already successful in restricted domains:
Deep Blue, Google, Automated Stock Trading, Checking X-rays.

Thoughts about Deep Blue

- Tremendous computing power (ability to “look ahead” several moves)
- Programmed by a team containing chess grandmasters.
- Had access to huge database of past chess games.
- Used machine learning tools on database to hone its skills.

“Human-machine computing”

Another example of human-computer computing…

Olde dream: “central repository of knowledge; all facts at your finger-tips.

Google

How it happened:

100s of millions of people created “content” for their own pleasure.

Powerful algorithms were used to extract meaningful info out of this, and have it instantly available.

“Second Life”

- Online community where everybody acquires an “avatar.” (Piece of code; point-and-click programming as in Scribbler.)
- Avatar customizable but follows laws of physics in imaginary world (remember: weather simulation)

Weird 2nd life facts

- Ability to buy/sell. ("Linden dollars")
- Budding markets in real estate, avatar skins, clothes, entertainment, "teaching" avatars new skills, etc.
- Emerging political systems

An interesting viewpoint: Second-Lifers are teaching the computer what "human life" is.

(Analogies: Chess database and Deep Blue,
WWW and Google.)

The most interesting question in the computational universe

Not:

"Will computers ever be conscious?"

But:

Where will all this take **us**?

(and our science, society, politics,…)

Administrivia

- One last reading (Searle) to be posted this afternoon.

- One final blogging assignment (due May 7): Write 2-3 paragraphs about AI, your expectations about it before you took this course, how they were shaped by this course, and the Searle article.

- Review sessions, probably afternoon of May 7.

- Final Exam Sunday May 16

Good luck with the final and have a great summer!
Enjoy your time in the computational universe!