Chinese Rooms and Exponential Growth:
More on the Computational Metaphor
A Few Questions (Besides Turing’s)

• The computational metaphor, and the “embodied mind” objection
• The “necessity of evolution” objection
Searle’s “Chinese Room”
Replies to Searle’s Thought Experiment

• The Systems Reply
• The Robot Reply
• The Brain-Simulator Reply
• The Combination Reply
• The Other-Minds Reply
• The Many-Mansions Reply
Some Additional Things to Consider about the “Chinese Room”

- Degrees of consciousness (animals? That thermostat again?)
- The “danger of common sense” argument
If Machines Can’t Think, What Good Is the Computational Metaphor?
An alternative way of thinking about the “computational metaphor”: direct physical combinations of computational and biological materials
**What Does It Mean for a Problem to Be Hard?**

Some problems may not have a solution, or may simply be ill-defined.

*Examples:*
- Write a computer program that will pass the Turing test. (Turing)
- Define (and/or teach) virtue. (Plato)

Some problems may be simply impossible given the resources. (Unlike the first class of problems, we at least know that these are both well-defined and unsolvable.)

*Examples:*
- Using a straightedge and compass, and given an angle theta, construct an angle of magnitude theta/3.

- Write a program which, given any computer program P and number N as input, determines whether P will ever halt when run on input N. (Turing)
Some problems may be impossible to solve with complete accuracy, but they can be approached by using approximations, guesswork, or heuristics. What this means is that perhaps all the "solutions" will be wrong (and we try to make most of the solutions as "right" as possible); or it might mean that some solutions, but not all, will be right.

Examples:

Given a two-dimensional scene projected on a retina or camera plane, deduce the three-dimensional scene (set of objects) that produced this two-dimensional projection.

Given a finite set of sentences, determine the formal structure of a context-free grammar that generated those sentences.
Some problems may be defined in such a way that they can only (or best) be approached by techniques that incorporate some notion of uncertainty, probability, or vagueness:

*Examples:*

- Was there life on Mars at some past time?
- If I have to place a bet on a future event (e.g., whether the Rockies will win the pennant), how should I bet?
- Is this object (person, animal) a threat?
- Is this shape: 0 an ellipse? Is it "close" to an ellipse?

Some problems may be completely solvable in principle; we could even write an algorithm to solve them. But this algorithm would take so long to run (or equivalently would require so much space) in most "standard" cases that we are forced to use more approximate (and hence unsure) means to approach the problem.

Given a configuration of a chess board, find the best move for the player whose turn it is.

Given a map of the U.S., and 100 cities (including Boulder), find the shortest "complete tour" of the cities, beginning and ending in Boulder, and visiting each of the other cities exactly once.
Some problems—fortunately—are "easy" in the sense that we can write a program to solve them, and the program will typically run in a reasonable time.

Given a positive number, find its square root.
Given a context-free grammar G, produce a sentence using that grammar.
Given 100 linear equations in 100 variables, determine whether those equations have a solution, and if so, what it is.
Computational Notions of “Hardness” of Problems

• Order-of-growth notation: how does a problem’s difficulty “scale up” to the large cases?
• Polynomial vs. exponential-time problems
• NP-Complete problems
• Parallelism as a general strategy