Theory of computation unit

- Languages, finite automata, and regular expressions (last week Wednesday)
- Equivalence of models of computation (last week Friday)
- The lambda calculus (Monday)
- Computability and tractability (**today**)
- Review for Test 3 (Friday)

Today:

- There are some things computers can't do, and we can prove it
  - The halting problem
  - Reductions
- There are some things which, as far as we know, computers can't do efficiently.
  - Define "efficiently"
  - $\mathcal{P}$ vs $\mathcal{NP}$; Hamiltonian cycle
  - Reductions
  - $\mathcal{NP}$-completeness

Test 3: Final exam block, Wednesday, May 6, 1:30 pm

Drawn from Lewis and Papadimitriou, *Elements of the Theory of Computation*, 2e 1998.

https://dilbert.com/search_results?terms=Illogical+Scientist

Undecidability has been so much a part of the culture of computer science since its beginnings, that it is easy to forget what a curious fact it is. Strictly speaking, once we accept the identification of problems with languages and algorithms as Turing machines, there are trivial reasons why there must be undecidable problems: There are more languages (uncountably many) than ways of deciding them [Turing machines/algorithms/computer programs]. Still, that such problems exist so close to our computational ambitions was a complete surprise when it was first observed in the 1930's. Undecidability is in some sense the most lethal form of complexity. *C Papadimitriou,* Computational Complexity, *pg 59*

- Suppose we have a program `halts`
- Build the program `diagonal`:

  `diagonal`(X) :
  a: if `halts`(X, X) then goto a else halt

- Does `diagonal`(`diagonal`) halt or diverge?

  `diagonal`(`diagonal`) halts $\longrightarrow$ `halts`(`diagonal`, `diagonal`) $\longrightarrow$ `diagonal`(`diagonal`) diverges
  `diagonal`(`diagonal`) diverges $\longrightarrow \sim$ `halts`(`diagonal`, `diagonal`) $\longrightarrow$ `diagonal`(`diagonal`) halts

This kind of argument should be familiar not only from your past exposure to computer science, but also from general twentieth century culture. *LP, pg 251*

Given an algorithm M, does M halt when given the empty string as its input? **This problem is undecidable.**

Proof. *The L be the language of discriptions of such algorithms, that is, algorithms that halt on empty input. Suppose an algorithm deciding this language exists; call it M_L.*

Build an algorithm, call it M_H, that takes as its input a machine M and input x to machine M. The algorithm M_H then constructs the description of an algorithm M_x which, when given the empty string as its input, calls/executes/acts like machine M on input W. Algorithm M_H then feeds M_x as input into algorithm M_L and returns the result of M_L as its result.

Note that by how M_H is constructed, M_x halts on the empty string if and only if M halts on x. Hence M_H decides the halting problem.

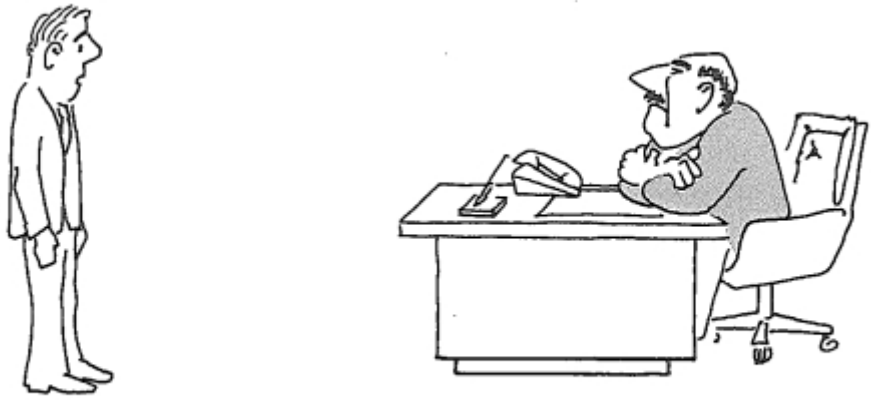
Contradiction. Since it is impossible for an algorithm to decide the halting problem, M_L cannot exist, and the given problem is undecidable. □

Let $\mathcal{P}$ be the set of languages (problems) for which there exist algorithms that decide that language within a number of steps bounded by a polynomial function of the size of the input string.

Let $\mathcal{NP}$ be the set of languages (problems) for which there exist algorithms that, when given a string an a certificate, verify that the string is in the language using that certificate, within a number of steps bounded by a polynomial function of the size of the input string.

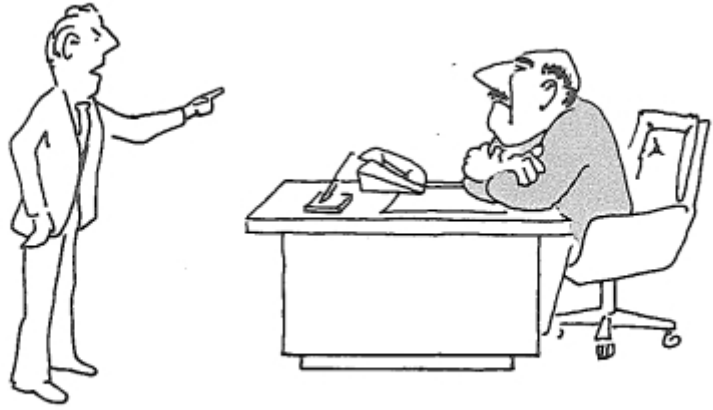
Define the **Hamiltonian cycle** problem: Given the description of a graph, decide whether that graph has a Hamiltonian cycle. This problem is in the set $\mathcal{NP}$.

Definte the **Traveling Salesman** problem: Given a set of points and the distances between each pair of points, find the minimum route among the points. ("Budget" version: Given those points and distances and a number B, find a route among all the points whose total distance is less than or equal to B.)



"I can't find an efficient algorithm, I guess I'm just too dumb."

Garey and Johnson, *Computers and Intractability*, Freeman, 1979; pg 2



"I can't find an efficient algorithm, because no such algorithm is possible!"

Garey and Johnson, *Computers and Intractability*, Freeman, 1979; pg 2

"I can't find an efficient algorithm, but neither can all these famous people."

Garey and Johnson, *Computers and Intractability*, Freeman, 1979; pg 3

For Thurs, Apr 30:

Submit semester reflection and feedback (see Canvas)