CSC 240

COMPLEXITY 3

COBHAM-EDMONDS THESIS

"Computational problems can be feasibly computed on some computational device only if they can be computed in polynomial time."

"Languages can be decided efficiently on some Turing machine only if they can be computed in $O(n^k)$."

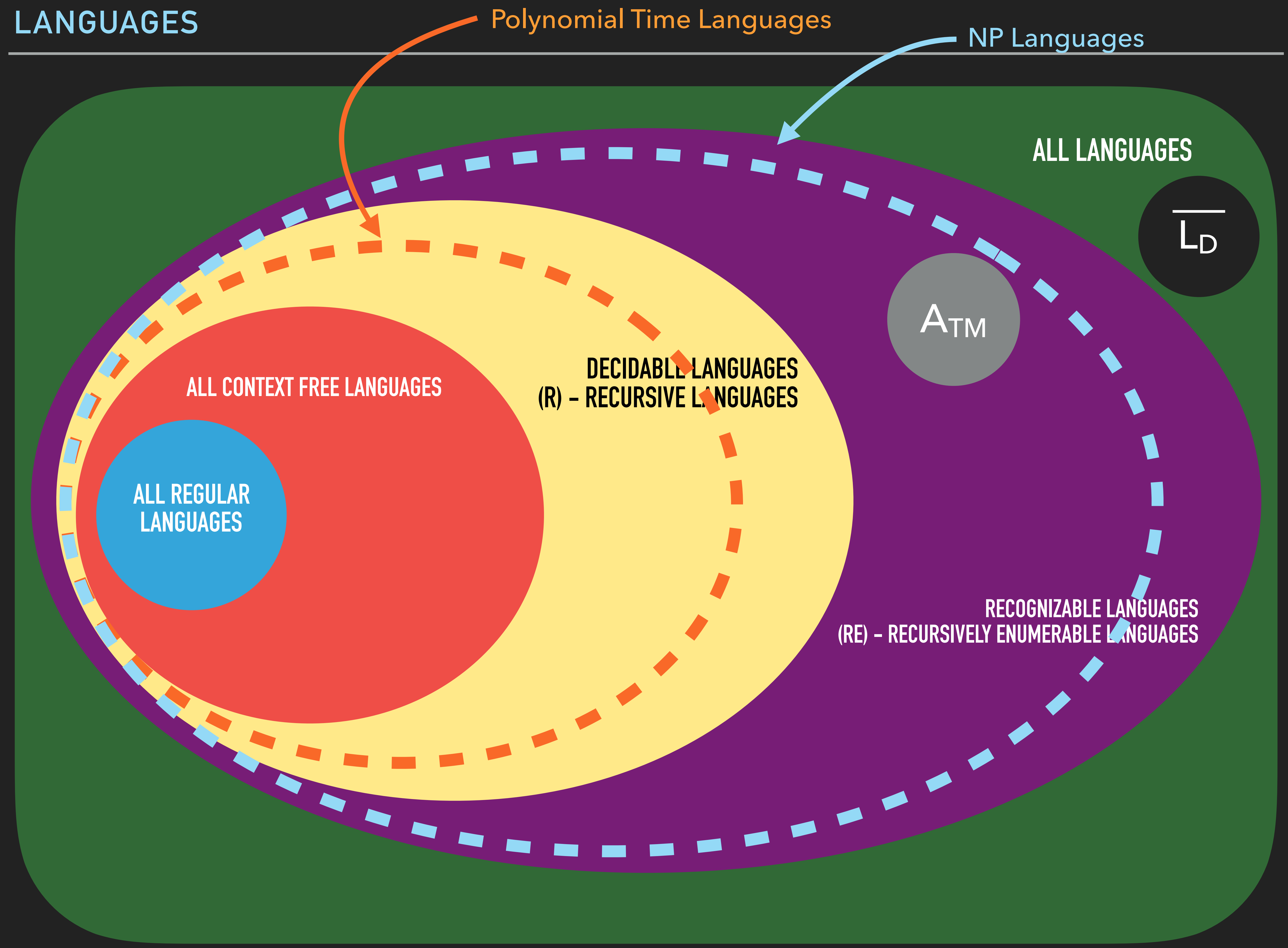
P = { L | There is a polynomial time decider for L }

- Complexity class P contains all problems that can be *decided* in polynomial time.
- Complexity class NP contains all problems that can be *verified* in polynomial time.

P = { L | There is a polynomial time decider for L }

NP = { L | There is a polynomial time verifier for L }

LANGUAGES



Polynomial Time Languages
NP Languages
ALL LANGUAGES
$\overline{L_D}$
A_{TM}
DECIDABLE LANGUAGES
(R) – RECURSIVE LANGUAGES
ALL CONTEXT FREE LANGUAGES
ALL REGULAR LANGUAGES
RECOGNIZABLE LANGUAGES
(RE) – RECURSIVELY ENUMERABLE LANGUAGES

$P \stackrel{?}{=} NP$

If a solution to a problem can be *verified* efficiently, does that mean there is *some* algorithm to *decide* that problem efficiently?

Proving this statement true or false will change the face of computer science.

NP COMPLEXITY

Are all NP problems equally difficult to solve?

If not, how can we rank their level of difficulty?

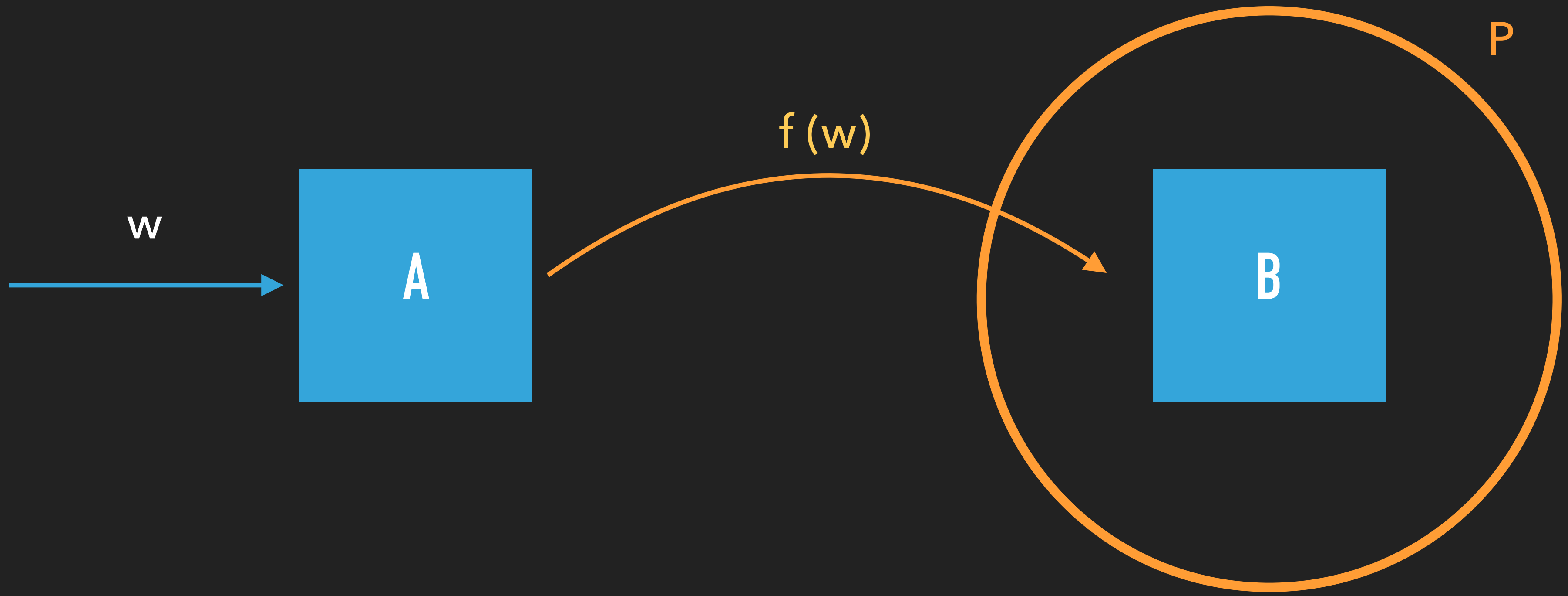
REDUCIBILITY

If we can take a problem, A, and convert it to another form, B, then if B has a solution, so does A.

1. If A is reducible to B, and B is decidable, then A is decidable.
2. If A is reducible to B, and A is undecidable, then B is undecidable.

EFFICIENT REDUCIBILITY

If we can take a problem, A, and *efficiently* convert it to another form, B, then if B has an *efficient* solution, so does A.

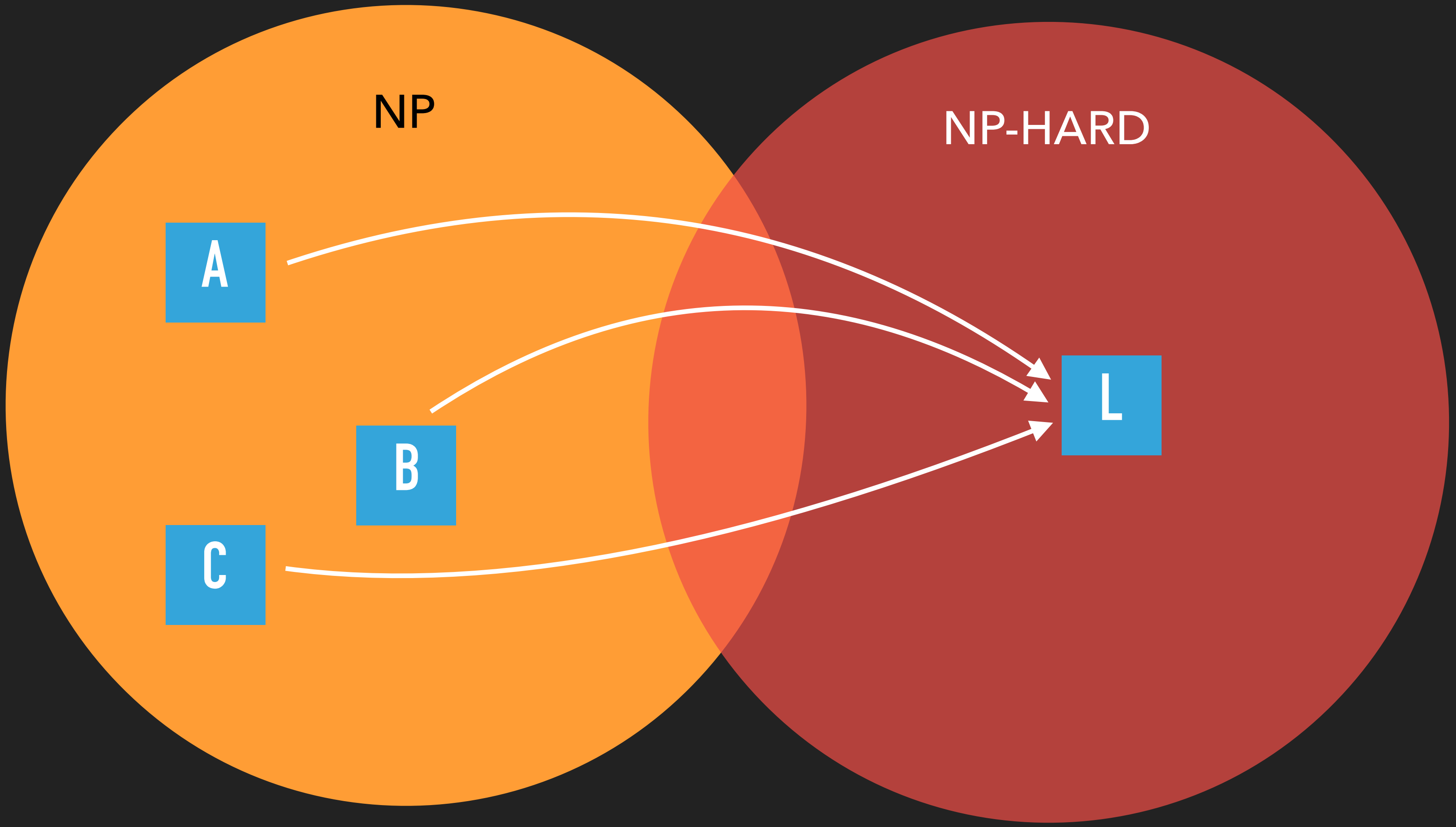


If A is reducible to B in polynomial time, and B is decidable in polynomial time, decidable, then A is also decidable in polynomial time.

$$A \leq_P B$$

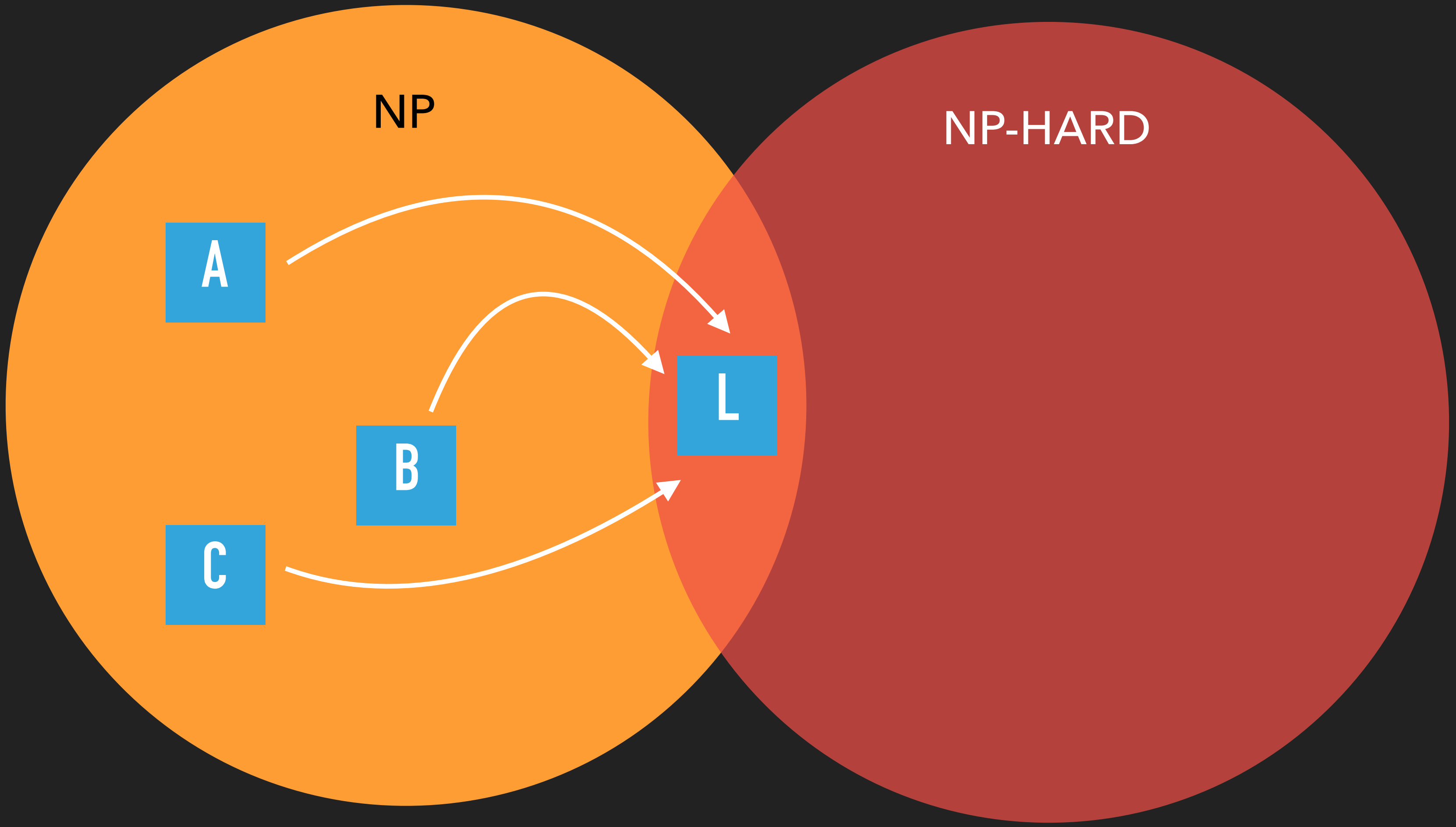
NP-HARD PROBLEMS

A language L is NP-Hard if every language in NP is reducible to it.



NP-COMPLETE PROBLEMS

A language L is NP-Complete if L is NP-Hard and L ∈ NP.



THE ULTIMATE QUESTION

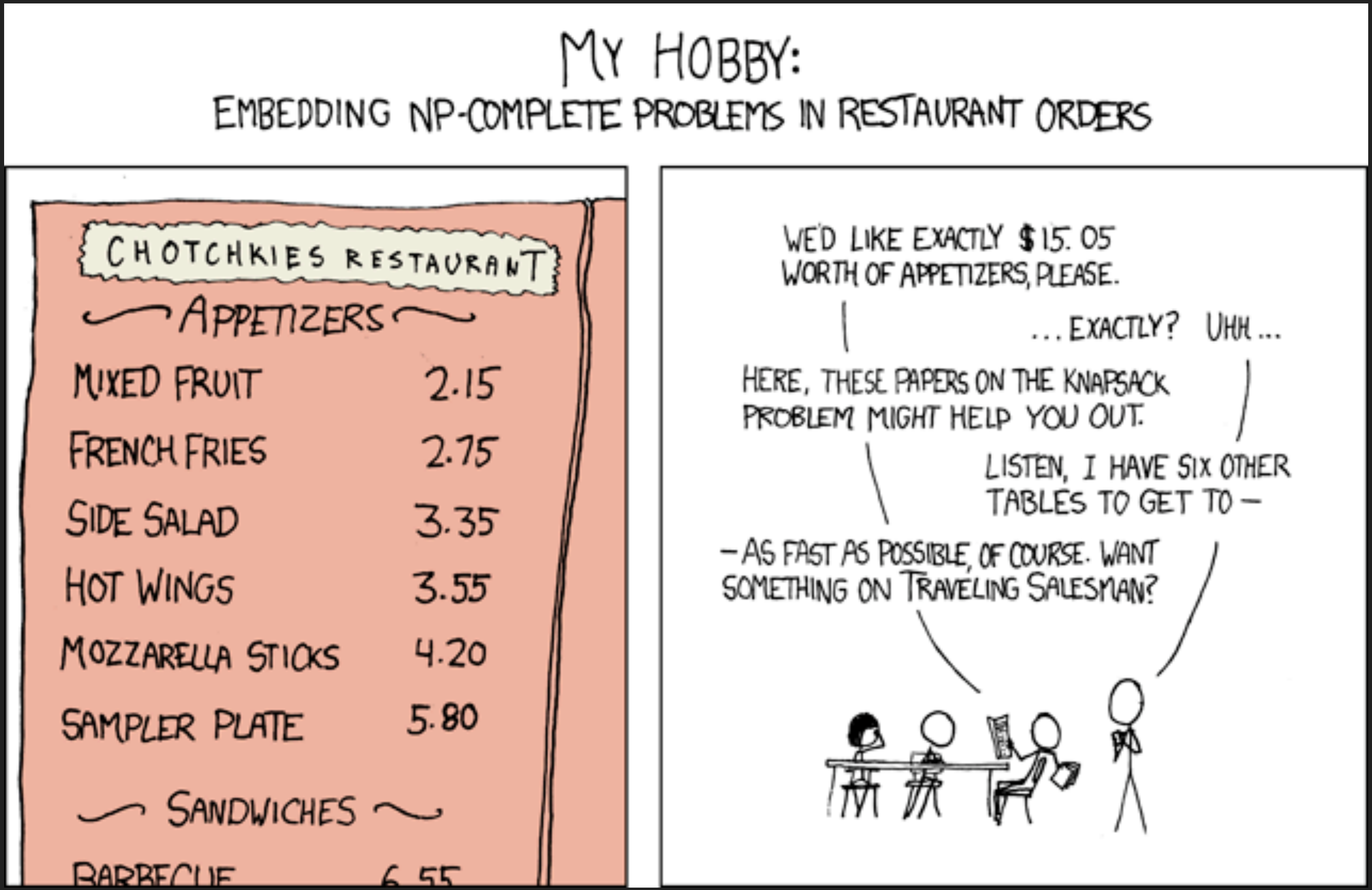
Fun Fact: If *any* NP-Complete language is in P, then P = NP

Fun Fact: If *any* NP-Complete language is not in P, then P ≠ NP

So to claim your $1,000,000 you just need to prove that any NP-Complete problem is or is not in P!
Easy Money!!!

I know an NP-Complete joke, but once you've heard one, you've heard them all…

THE ULTIMATE QUESTION


MY HOBBY:
EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS
CHOTCHKIES RESTAURANT
APPETIZERS
MIXED FRUIT 2.15
FRENCH FRIES 2.75
SIDE SALAD 3.35
HOT WINGS 3.55
MOZZARELLA STICKS 4.20
SAMPLER PLATE 5.80
SANDWICHES
BARBECUE 6.55
WE'D LIKE EXACTLY $15.05 WORTH OF APPETIZERS, PLEASE.
...EXACTLY? UHH...
HERE, THESE PAPERS ON THE KNAPSACK PROBLEM MIGHT HELP YOU OUT.
LISTEN, I HAVE SIX OTHER TABLES TO GET TO –
– AS FAST AS POSSIBLE, OF COURSE. WANT SOMETHING ON TRAVELING SALESMAN?

https://xkcd.com/287/

NP HARD PROBLEMS

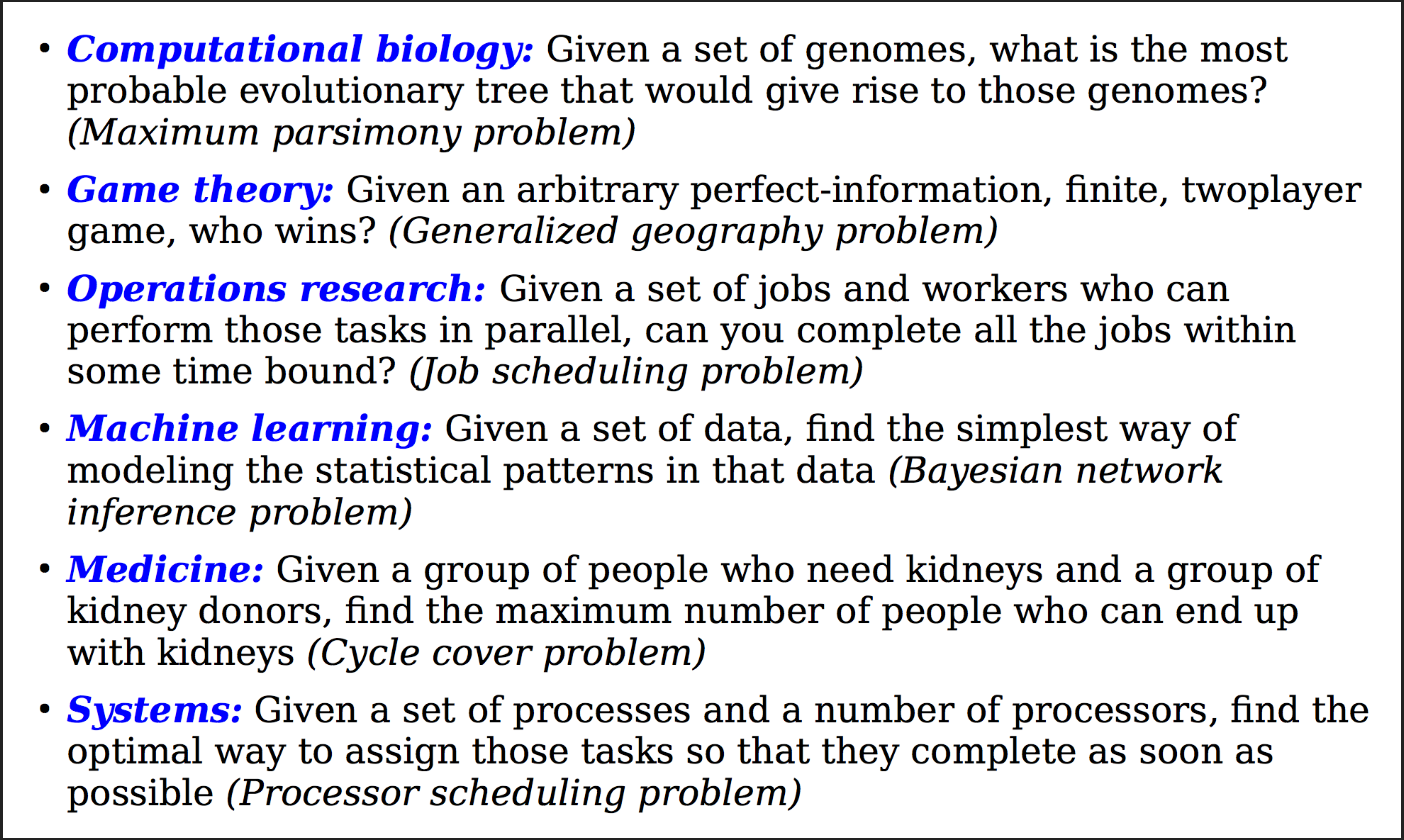
There are lots of problems that appear to be easy, but are actually NP-Hard.

This means that there is no known algorithm that can efficiently (in polynomial time) solve those problems for all inputs.

This means that if you do have to solve an NP-Hard problem, you have to be content with one of these options:

1. An approximate answer (local minimum vs the exact minimum)
2. A probabilistic answer
3. An algorithm that only works for a subset of possible inputs

SOME NP-HARD PROBLEMS



- ***Computational biology:*** Given a set of genomes, what is the most probable evolutionary tree that would give rise to those genomes? *(Maximum parsimony problem)*
- ***Game theory:*** Given an arbitrary perfect-information, finite, twoplayer game, who wins? *(Generalized geography problem)*
- ***Operations research:*** Given a set of jobs and workers who can perform those tasks in parallel, can you complete all the jobs within some time bound? *(Job scheduling problem)*
- ***Machine learning:*** Given a set of data, find the simplest way of modeling the statistical patterns in that data *(Bayesian network inference problem)*
- ***Medicine:*** Given a group of people who need kidneys and a group of kidney donors, find the maximum number of people who can end up with kidneys *(Cycle cover problem)*
- ***Systems:*** Given a set of processes and a number of processors, find the optimal way to assign those tasks so that they complete as soon as possible *(Processor scheduling problem)*

Keith Schwarz – Stanford