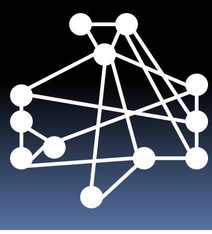
15-251: Great Theoretical Ideas in Computer Science
Fall 2014; Lecture 23
November 18, 2014

Efficient Reductions

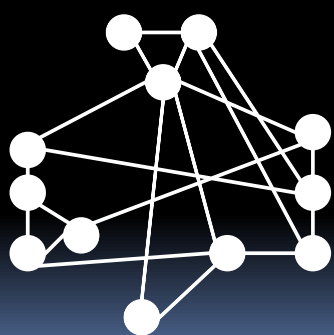


Reductions

How to tell a mathematician from and an engineer:

- Put an empty kettle in the middle of the kitchen floor and tell your subjects to boil some water.
- The engineer will fill the kettle with water, put it on the stove, and turn the flame on. The mathematician will do the same thing.
- Next, put the kettle already filled with water on the stove, and ask the subjects to boil the water.
- The engineer will turn the flame on.
- The mathematician will empty the kettle and put it in the middle of the kitchen floor… thereby reducing the problem to one that has already been solved!

A Graph Named "Gadget"



K-Coloring

We define a k-coloring of a graph:

Each node gets colored with one color

At most k different colors are used

If two nodes have an edge between them they must have different colors

A graph is called k-colorable if and only if it has a k-coloring

A 2-CRAYOLA Question!

Is Gadget 2-colorable?

No, it contains a triangle

A 2-CRAYOLA Question!

Given a graph G, how can we decide if it is 2-colorable?

Answer: Enumerate all 2^n possible colorings to look for a valid 2-color

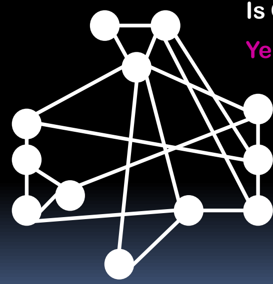
How can we efficiently decide if G is 2-colorable (aka bipartite)?

<u>Theorem</u>: G contains an odd cycle if and only if G is not 2-colorable

Efficient 2-coloring algorithm:

- To 2-color a connected graph G, pick an arbitrary node v, and color it white
- Color all v's neighbors black
- Color all their uncolored neighbors white, and so on
- If the algorithm terminates without a color conflict, output the 2-coloring
- Else, output graph is not 2-colorable (the conflict proves no 2-coloring is possible, and there is an odd cycle)

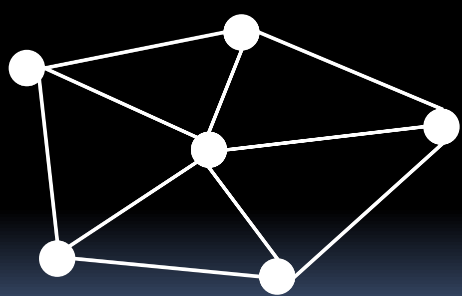
A 3-CRAYOLA Question!



Is Gadget 3-colorable?

Yes!

A 3-CRAYOLA Question!



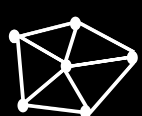
Is the "wheel" 3-colorable?

3-Coloring Is Decidable by Brute Force

Try out all 3^n colorings until you determine if G has a 3-coloring

Best known algorithms take c^n time for some $c > 1$ (exponential runtime)

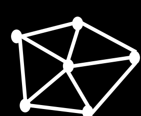
A 3-CRAYOLA Oracle



YES/NO

3-Colorability Oracle

Better 3-CRAYOLA Oracle

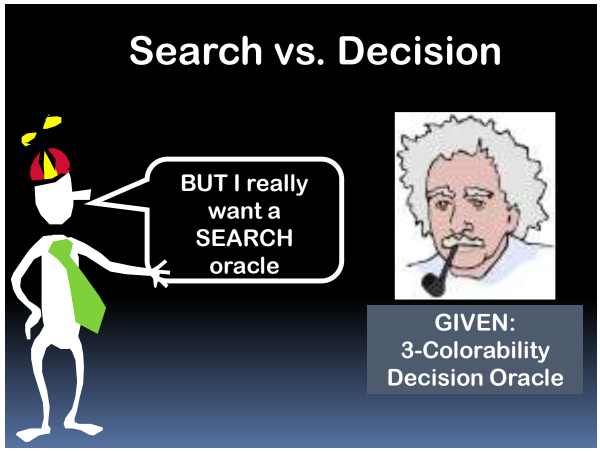


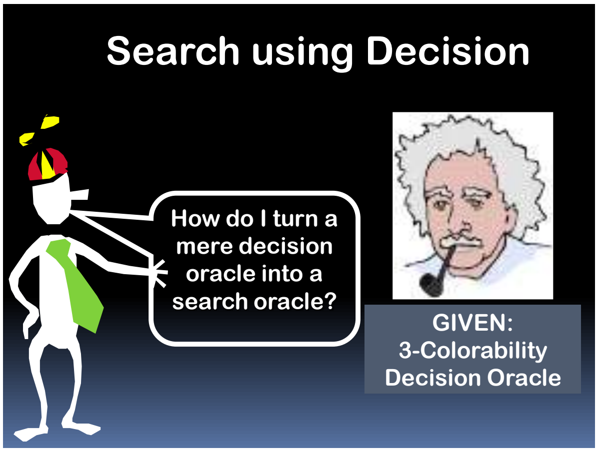
NO, or

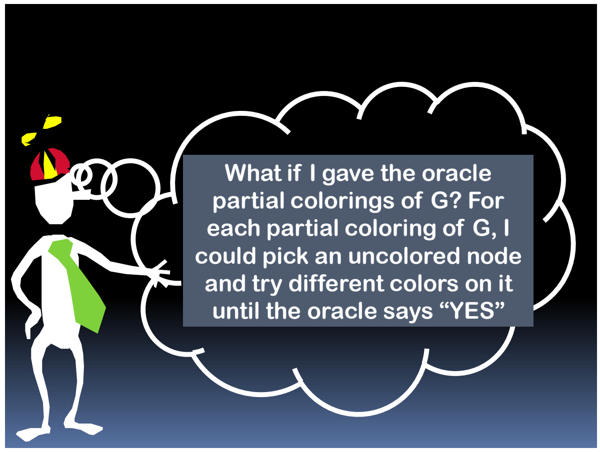
YES here is how: gives 3-coloring of the nodes

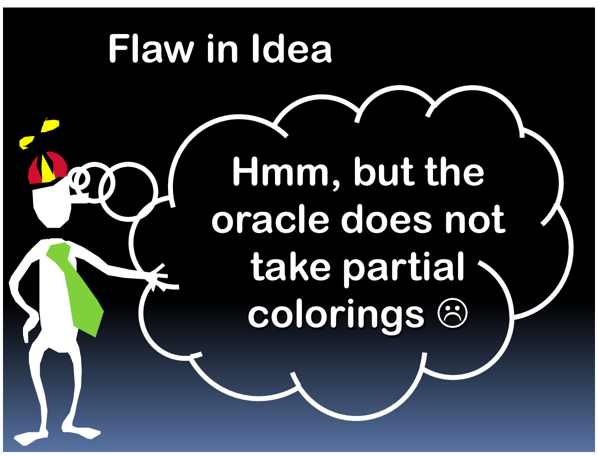
3-Colorability Search Oracle

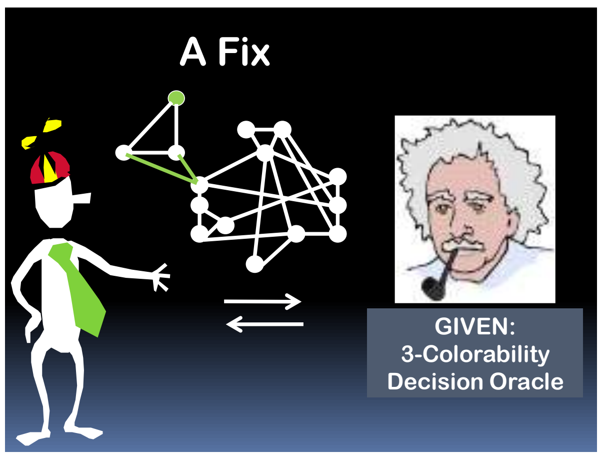

3-Colorability
Search Oracle
3-Colorability
Decision Oracle


Search vs. Decision
BUT I really want a SEARCH oracle
GIVEN:
3-Colorability
Decision Oracle

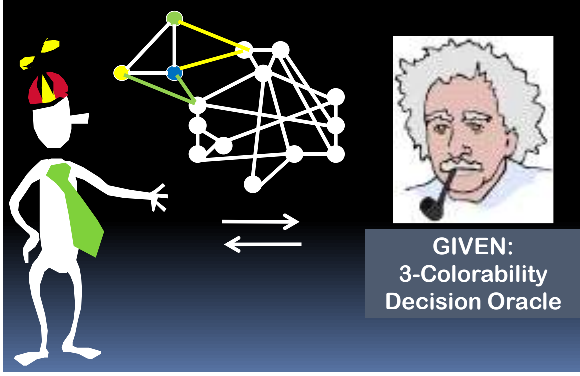

Search using Decision
How do I turn a mere decision oracle into a search oracle?
GIVEN:
3-Colorability
Decision Oracle


What if I gave the oracle partial colorings of G? For each partial coloring of G, I could pick an uncolored node and try different colors on it until the oracle says "YES"


Flaw in Idea
Hmm, but the oracle does not take partial colorings ☹


A Fix
GIVEN:
3-Colorability
Decision Oracle

A Fix



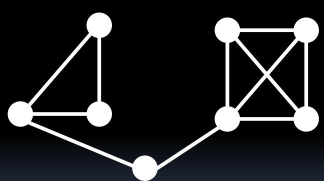
GIVEN:
3-Colorability
Decision Oracle

Let's now look at two other problems:

1. K-Clique
2. K-Independent Set

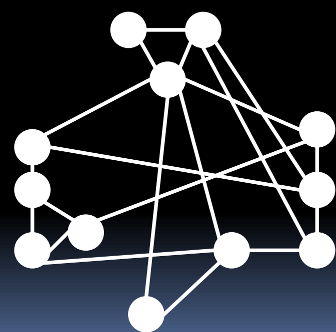
K-Cliques

A K-clique is a set of K nodes with all K(K-1)/2 possible edges between them



This graph contains a 4-clique

Largest clique in gadget graph?



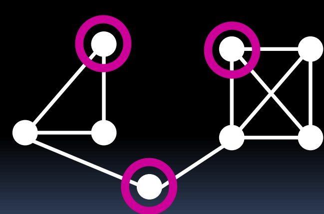
Given: (G, k)
Question: Does G contain a k-clique?

BRUTE FORCE: Try out all n choose k possible locations for the k clique

No substantially faster algorithm known!

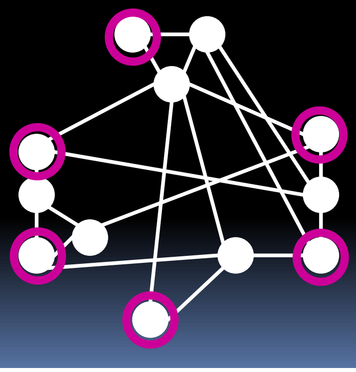
Independent Set

An independent set is a set of nodes with no edges between them



This graph contains an independent set of size 3

Independent set in gadget graph



Given: (G, k)
Question: Does G contain an independent set of size k?

BRUTE FORCE: Try out all n choose k possible locations for the k independent set

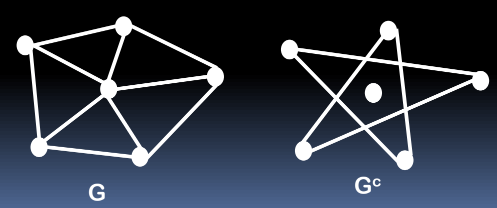
No substantially faster algorithm known!

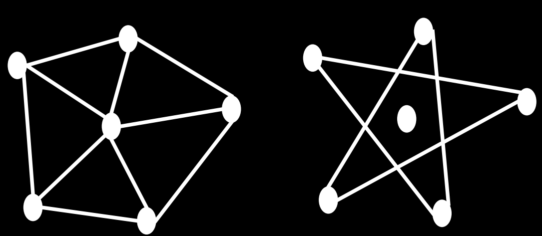
Clique / Independent Set

Two problems that are cosmetically different, but the same substance-wise

Complement of G

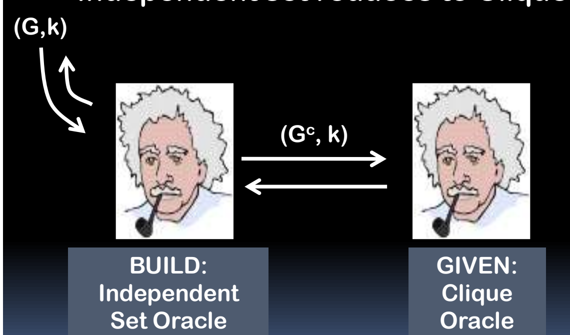
Given a graph G, let G^c, the complement of G, be the graph obtained by the rule that two nodes in G^c are connected if and only if the corresponding nodes of G are not connected

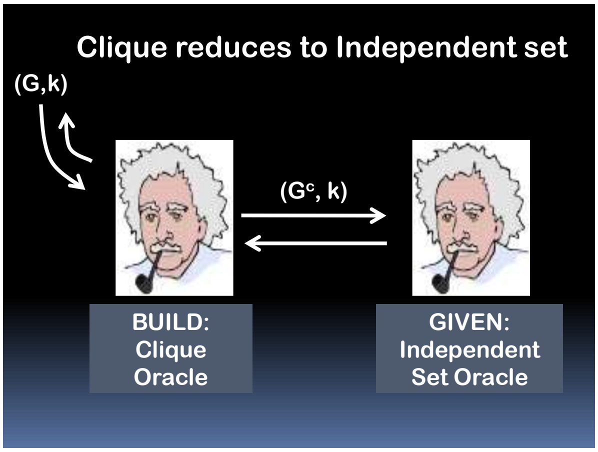


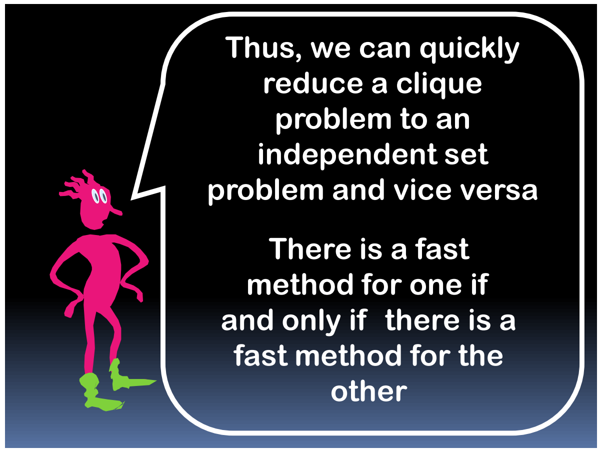


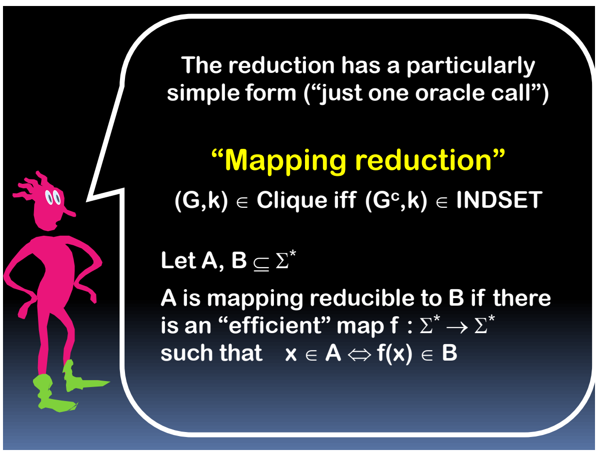
G has a k-clique $\Leftrightarrow$ G^c has an independent set of size k

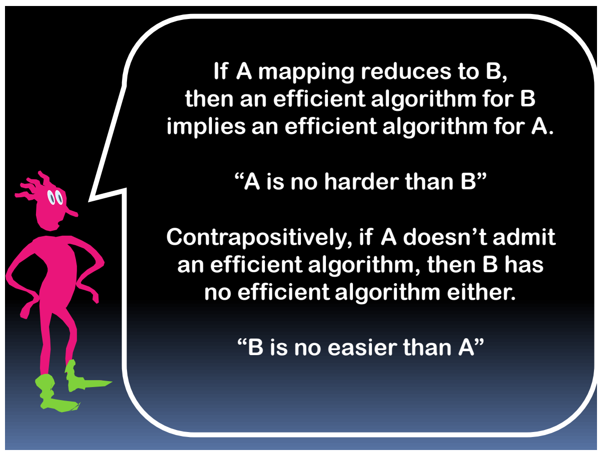
Independent set reduces to Clique

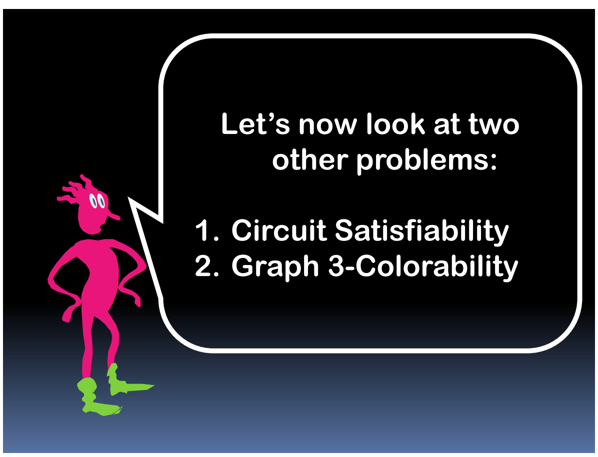


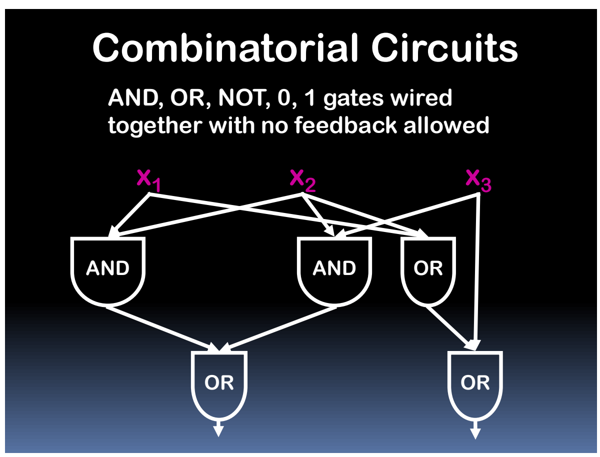

Clique reduces to Independent set
(G,k)
(G^c, k)
BUILD: Clique Oracle
GIVEN: Independent Set Oracle


Thus, we can quickly reduce a clique problem to an independent set problem and vice versa
There is a fast method for one if and only if there is a fast method for the other


The reduction has a particularly simple form ("just one oracle call")
"Mapping reduction"
(G,k) ∈ Clique iff (G^c,k) ∈ INDSET
Let A, B ⊆ Σ*
A is mapping reducible to B if there is an "efficient" map f : Σ* → Σ*
such that x ∈ A ⇔ f(x) ∈ B


If A mapping reduces to B, then an efficient algorithm for B implies an efficient algorithm for A.
"A is no harder than B"
Contrapositively, if A doesn't admit an efficient algorithm, then B has no efficient algorithm either.
"B is no easier than A"

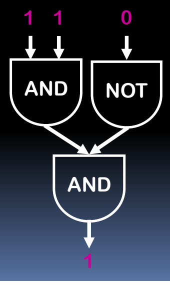

Let's now look at two other problems:
1. Circuit Satisfiability
2. Graph 3-Colorability


Combinatorial Circuits
AND, OR, NOT, 0, 1 gates wired together with no feedback allowed
x1
x2
x3
AND
AND
OR
OR
OR

Circuit-Satisfiability

Given a circuit with n-inputs and one output, is there a way to assign 0-1 values to the input wires so that the output value is 1 (true)?

Yes, this circuit is satisfiable: 110



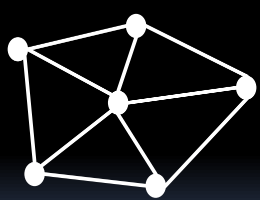
Circuit-Satisfiability

Given: A circuit with n-inputs and one output, is there a way to assign 0-1 values to the input s so that the output value is 1 (true)?

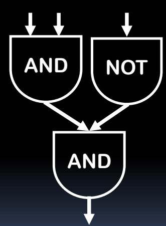
BRUTE FORCE: Try out all 2^n assignments

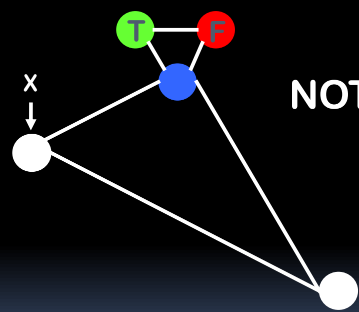
Again, no substantially faster algorithm known (not even c^n time for any constant $c < 2$)

3-Colorability

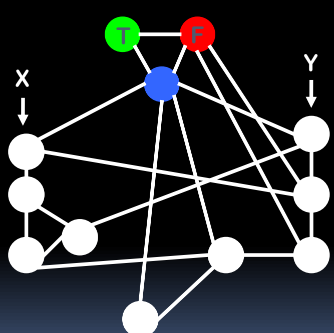


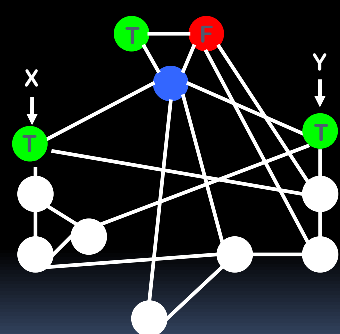
Circuit Satisfiability

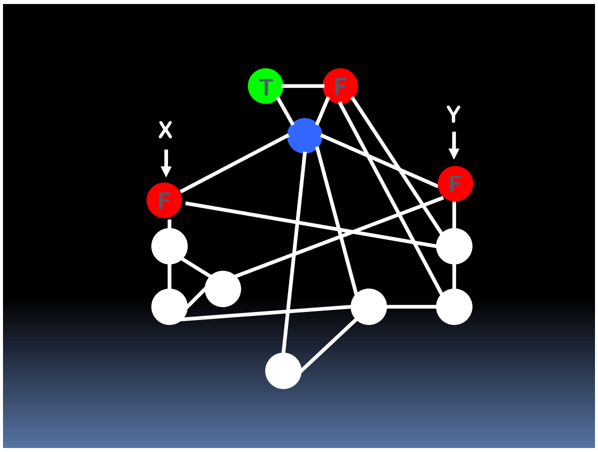


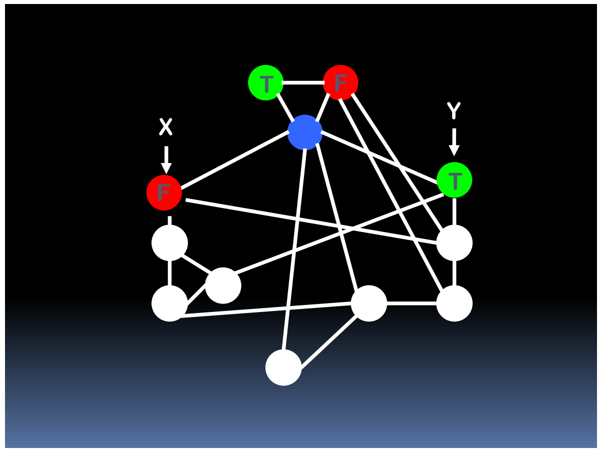


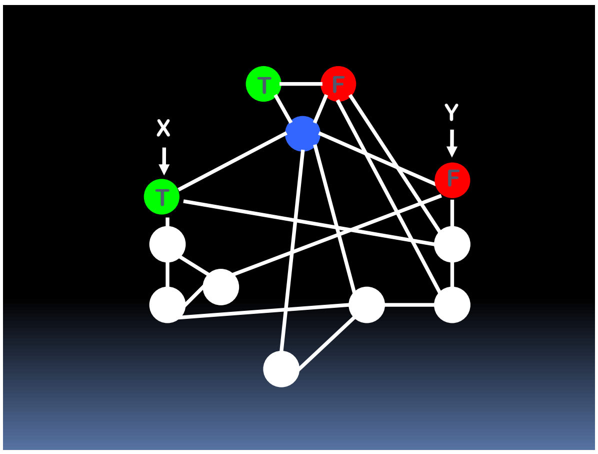
NOT gate!

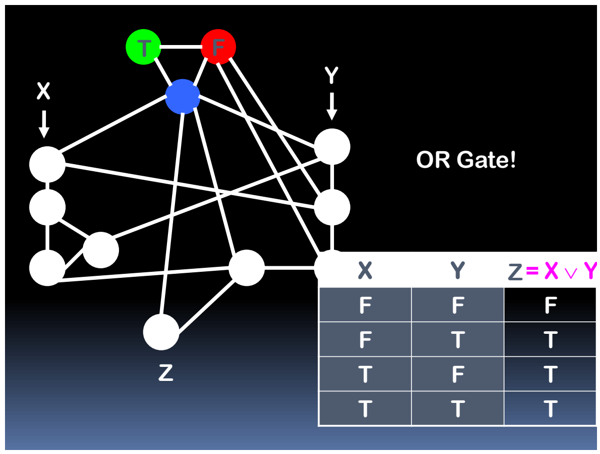




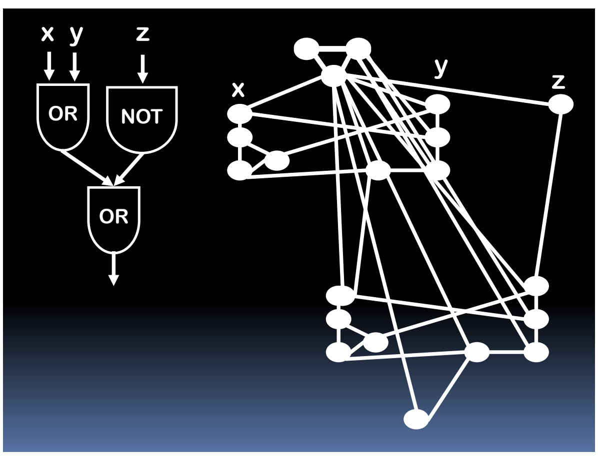


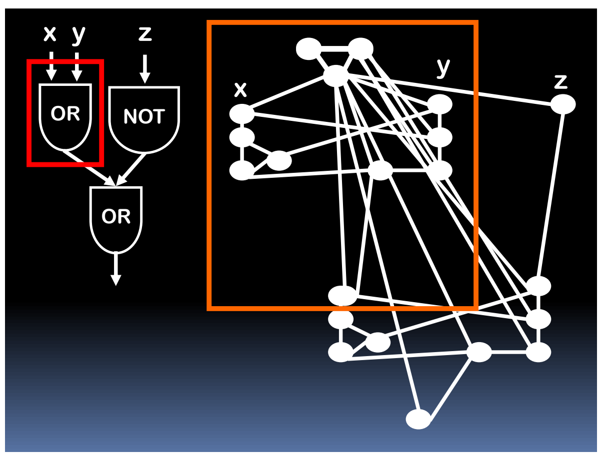


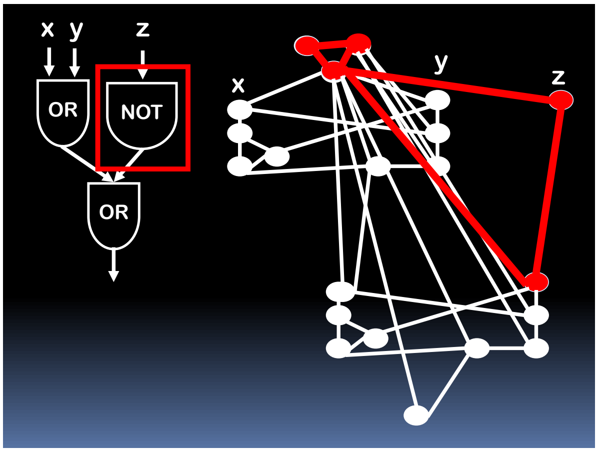


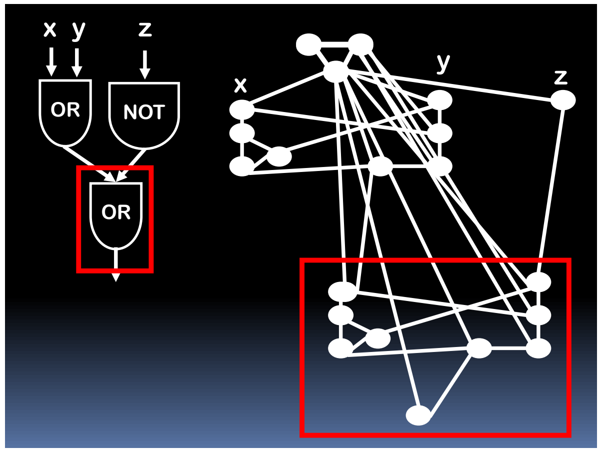


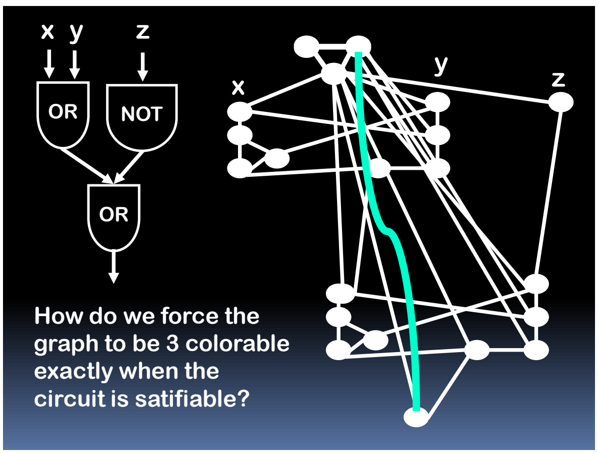
X	Y	Z = X ∨ Y
F	F	F
F	T	T
T	F	T
T	T	T

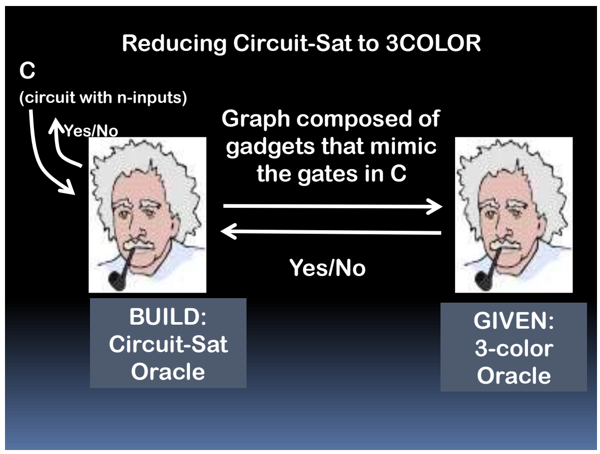






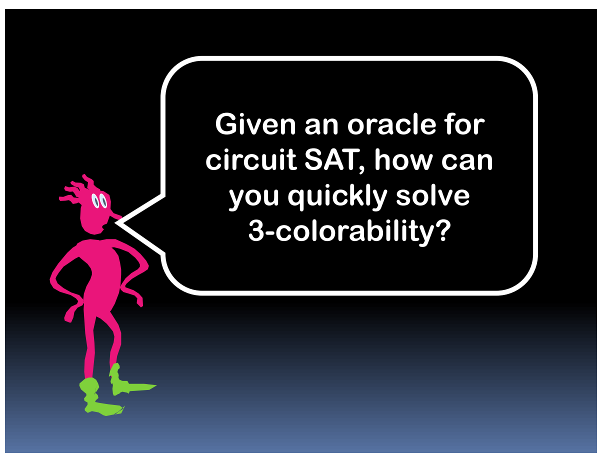


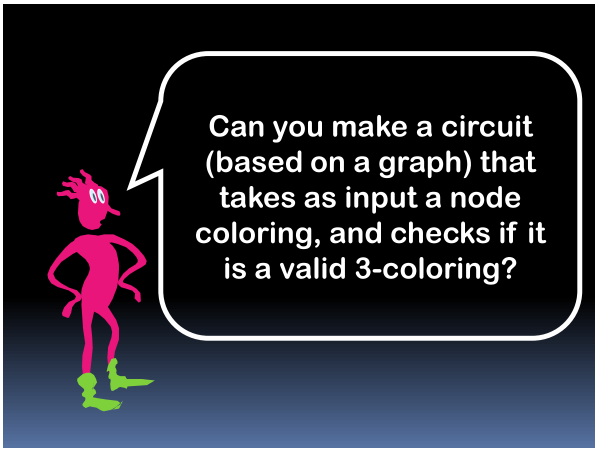


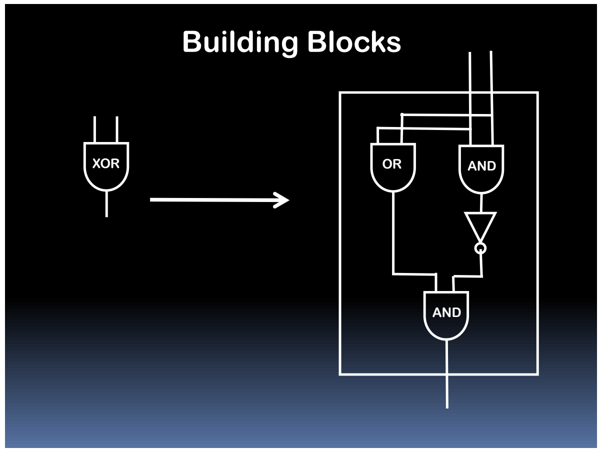


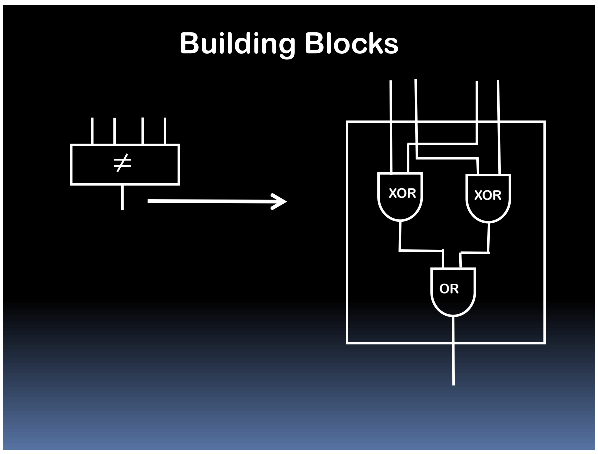
Upshot:

You can quickly transform a method to decide 3-coloring into a method to decide circuit satisfiability!







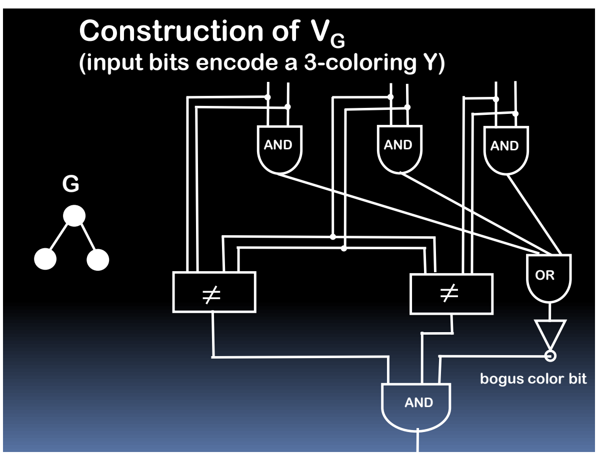


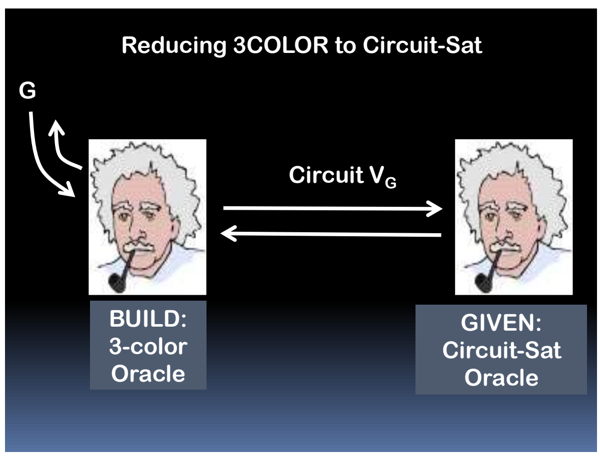
Circuit $V_G(\,)$

Let $V_G(Y)$ be a circuit constructed for a graph G, that takes as input an assignment of colors to nodes Y, and **verifies** that Y is a valid 3 coloring of G. I.e., $V_G(Y) = 1$ iff Y is a 3 coloring of G

Y is expressed as a 2n bit sequence

Given G, we can construct $V_G(Y)$ in time O(n+m)





Circuit-SAT / 3-Colorability

Two problems that are cosmetically different,
but each is “mapping reducible” to the other

Circuit-SAT / 3-Colorability

Clique / Independent Set

Given an oracle for circuit SAT, how can you quickly solve k-clique?

<u>Hint</u>: Similar to 3-coloring case.

Given an oracle for k-clique, one can build oracle for circuit SAT.
In fact, CircuitSAT mapping reduces to k-clique
(We won’t prove it, but it is convenient to go through 3SAT. May show reduction from 3SAT to k-clique in next lecture/recitation)

Circuit-SAT / 3-Colorability

Clique / Independent Set

These four problems are efficiently reducible to each other

FACT: No one knows a way to solve any of the 4 problems that is fast on all instances.
But if one of them has such an algorithm, then all of them do!

3SAT

Instance: A *Boolean formula* $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$

- Variables $x_1, x_2, \ldots, x_n$ and
- *clauses* $C_1, C_2, \ldots, C_m$,

  each C_j is of the OR of *up to* 3 literals,eg. $(x_1 \vee \overline{x_3} \vee x_{17})$

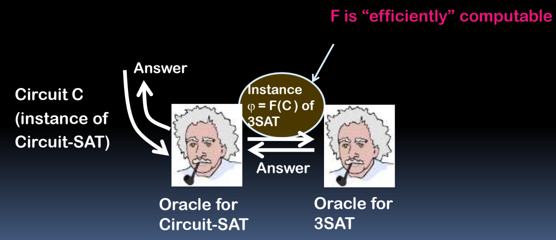
Decision problem: Is there a Boolean assignment to the variables that satisfies φ (i.e., all the clauses)

3SAT is a special case of Circuit-SAT (Why?)

So SAT reduces to Circuit-SAT

(given Circuit-SAT oracle can build one for 3SAT)

Reducing Circuit-SAT to 3SAT

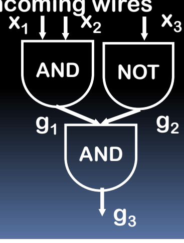


How?

Write a 3CNF formula to simulate the circuit.

- Variable for each wire of the circuit
- Clauses for each gate, to ensure correct computation of bit on outgoing wire(s) from the (two) bits on incoming wires

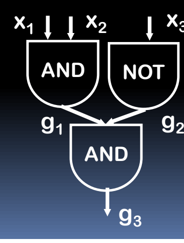
$(w_i \vee w_j) \wedge (\overline{w_i} \vee \overline{w_j})$

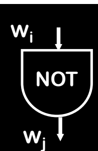


Write a 3CNF formula to simulate the circuit.

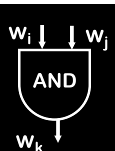
- Variable for each wire of the circuit
- Clauses for each gate, to ensure correct computation of bit on outgoing wire(s) from the (two) bits on incoming wires

$(w_i \vee w_j \vee \overline{w_k}) \wedge (w_i \vee \overline{w_j} \vee \overline{w_k}) \wedge (\overline{w_i} \vee w_j \vee \overline{w_k}) \wedge (\overline{w_i} \vee \overline{w_j} \vee w_k)$





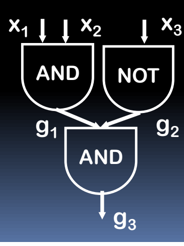
$(w_i \vee w_j) \wedge (\overline{w_i} \vee \overline{w_j})$



$(w_i \vee w_j \vee \overline{w_k}) \wedge (w_i \vee \overline{w_j} \vee \overline{w_k}) \wedge (\overline{w_i} \vee w_j \vee \overline{w_k}) \wedge (\overline{w_i} \vee \overline{w_j} \vee w_k)$

How do we ensure using our formula that circuit outputs True ?

Add the unary clause g_3



Summary

Many problems that appear different on the surface can be efficiently reduced to each other, revealing a deeper similarity.

Reductions are one of the most versatile and powerful tools in theoretical computer science to understand and relate the computational complexity of problems.