# 15-780: Graduate AI

# *Lecture 3. FOL proofs; SAT*

*Geoff Gordon (this lecture)*
*Tuomas Sandholm*
*TAs Byron Boots, Sam Ganzfried*

# Admin

# HW1

- *Out today*
- *Due Tue, Feb. 3 (two weeks)*
  - *hand in hardcopy at beginning of class*
- *Covers propositional and FOL*
- *Don't leave it to the last minute!*

# Collaboration policy

- *OK to discuss general strategies*
- *What you hand in must be your own work*
  - *written with no access to notes from joint meetings, websites, etc.*
- *You must acknowledge all significant discussions, relevant websites, etc., on your HW*

# Late policy

- *You have 3 late days to use on HWs*
  - *these account for conference travel, holidays, illness, or any other reasons*
- *After late days, 75% for next day, 50% for next, 0% thereafter (but still must turn in)*
- *Day = 24 hrs, HWs due at 10:30AM*

# Office hours

- *Office hours start this week (see website for times)*
- *But, I have a conflict this week due to admissions; let me know by email if there is demand, and if so I can reschedule*

# Matlab tutorial

- *Thu 1/22, 4–5PM, Wean Hall 5409*

# Review

# In propositional logic

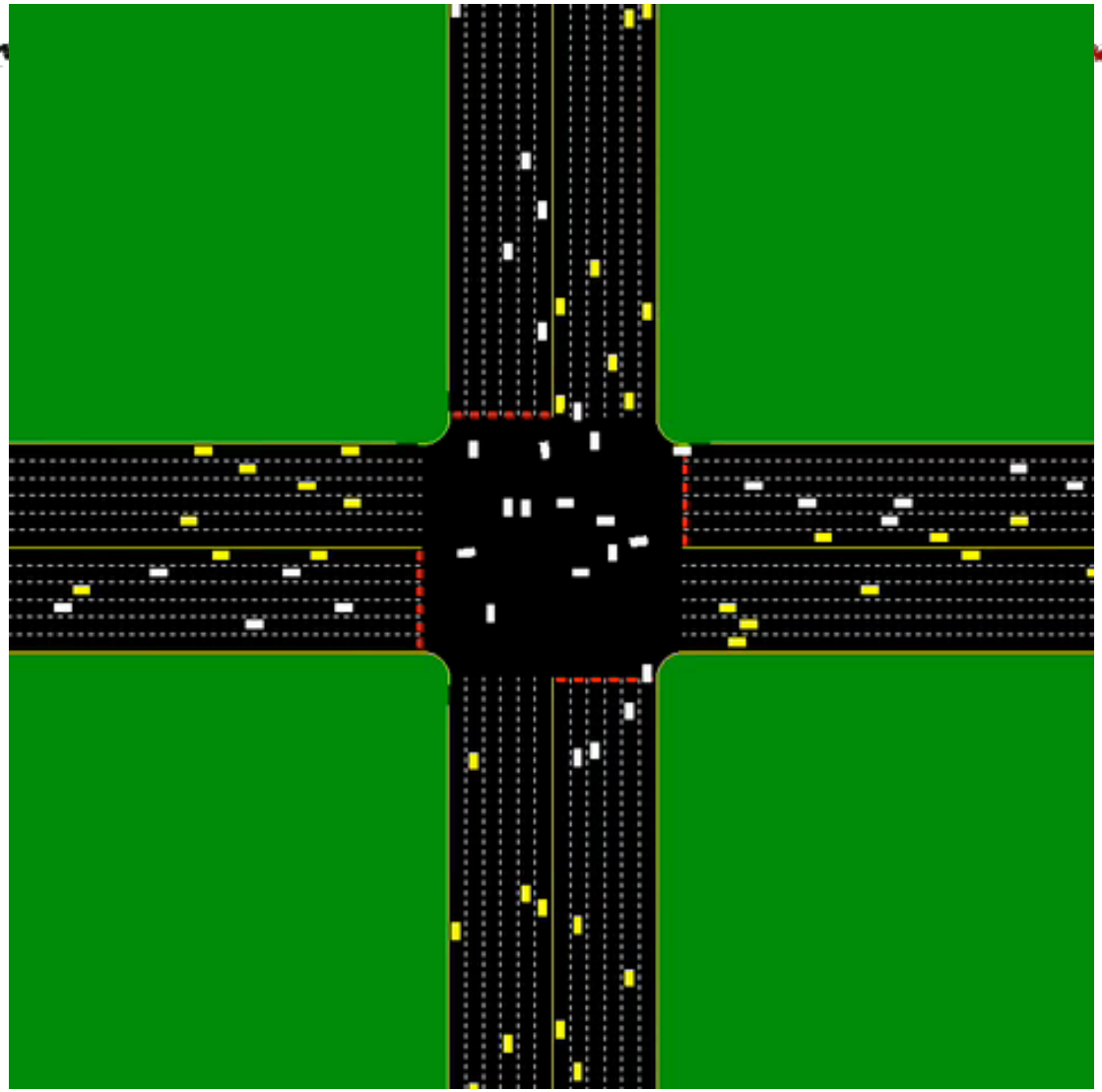
- *Compositional semantics, structural induction*
- *Proof trees, proof by contradiction*
- *Inference rules (e.g., resolution)*
- *Soundness, completeness*
- *Horn clauses*
- *Nonmonotonic logic*

# In FOL

- *Compositional semantics*
  - *objects, functions, predicates*
  - *terms, atoms, literals, sentences*
  - *quantifiers, free/bound variables*
  - *models, interpretations*
- *Generalized de Morgan's law*
- *Skolemization, CNF*

# Project Ideas

# Traffic insanity



# Sensor planning

- *Plan a path for this robot so that it gets a good view of an object as fast as possible*

# Mini-robots

- *Do something cool w/ Lego Mindstorms*
  - *plan footstep placements*
  - *plan how to grip objects*

# Poker

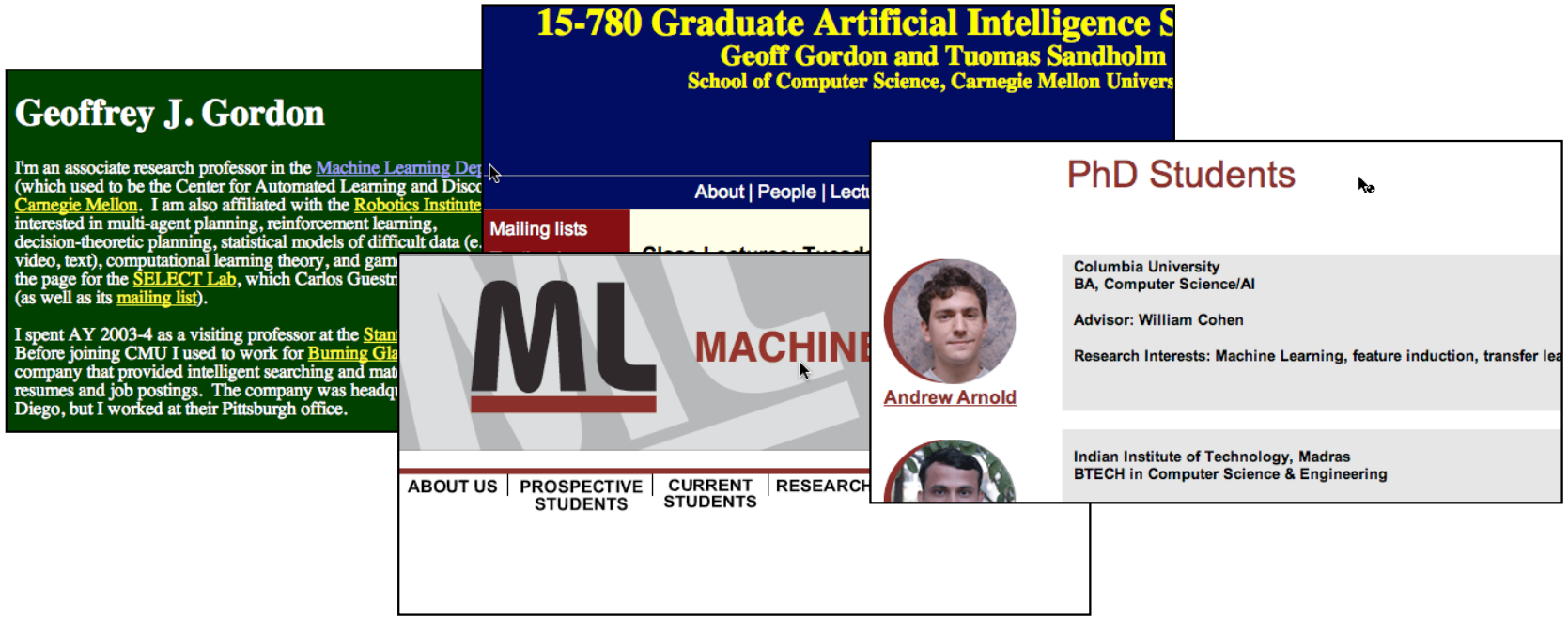
# Poker

- *Minimax strategy for heads-up poker = solving linear program*
- *1-card hands, 13-card deck: 52 vars, instantaneous*
- *RI Hold'Em: ~1,000,000 vars*
  - *2 weeks / 30GB (exact sol, CPLEX)*
  - *40 min / 1.5GB (approx sol)*
- *TX Hold'Em: ??? (up to $10^{17}$ vars or so)*

# Poker

- *Learning by repeated play*

# Understand the web




- *Write a probabilistic knowledge base describing a portion of the web*
- *Learn parameters of the model*

# Proofs in FOL

# FOL is special

- *Despite being much more powerful than propositional logic, there is still a **sound** and **complete** inference procedure for FOL*
- *Almost any significant extension breaks this property*
- *This is why FOL is popular: very powerful language with a sound & complete inference procedure*

# Proofs

- *Proofs by contradiction work as before:*
  - *add ¬S to KB*
  - *put in CNF*
  - *run resolution*
  - *if we get an empty clause, we've proven S by contradiction*
- *But, CNF and resolution have changed*

# Generalizing resolution

- *Propositional:* $(\neg a \vee b) \wedge a \models b$
- *FOL:*

  $(\neg man(x) \vee mortal(x)) \wedge man(Socrates)$

  $\models (\neg man(Socrates) \vee mortal(Socrates))$
  $\wedge man(Socrates)$

  $\models mortal(Socrates)$
- *Difference: had to substitute* $x \rightarrow Socrates$

# Universal instantiation

- *What we just did is UI:*

  $(\neg man(x) \vee mortal(x))$

  $\models (\neg man(Socrates) \vee mortal(Socrates))$

- *Works for $x \rightarrow$ any ground term*

  $(\neg man(uncle(student(Socrates))) \vee mortal(uncle(student(Socrates))))$

- *For proofs, need a good way to find useful instantiations*

# Substitution lists

- *List of variable → value pairs*
- *Values may contain variables (leaving flexibility about final instantiation)*
- *But, no LHS may be contained in any RHS*
  - *i.e., applying substitution twice is the same as doing it once*
- *E.g., x → Socrates, y → LCA(Socrates, z)*

*LCA = last common advisor*

# Unification

- *Two FOL terms **unify** with each other if there is a substitution list that makes them syntactically identical*
- *man(x), man(Socrates) unify using the substitution x → Socrates*
- *Importance: purely syntactic criterion for identifying good substitutions*

# Unification examples

- *loves(x, x), loves(John, y) unify using*
  $x \rightarrow John, y \rightarrow John$
- *loves(x, x), loves(John, Mary) can't unify*
- *loves(uncle(x), y), loves(z, aunt(z)):*

# Unification examples

- *loves(x, x), loves(John, y) unify using*
  $x \rightarrow John, y \rightarrow John$
- *loves(x, x), loves(John, Mary) can't unify*
- *loves(uncle(x), y), loves(z, aunt(z)):*
  - $z \rightarrow uncle(x), y \rightarrow aunt(uncle(x))$
  - *loves(uncle(x), aunt(uncle(x)))*

# Quiz

- *Can we unify*

  *knows(John, x)  knows(x, Mary)*

- *What about*

  *knows(John, x)  knows(y, Mary)*

# Standardizing apart

- *Can we unify*

  *knows(John, x)  knows(x, Mary)*

  *No!*

- *What about*

  *knows(John, x)  knows(y, Mary)*

  $x \rightarrow Mary, y \rightarrow John$

# Standardize apart

- *But knows(x, Mary) is logically equivalent to knows(y, Mary)!*
- *Moral: standardize apart before unifying*

# Most general unifier

- *May be many substitutions that unify two formulas*
- *MGU is unique (up to renaming)*
- *Simple, moderately fast algorithm for finding MGU (see RN); more complex, linear-time algorithm*

Linear unification. MS Paterson, MN Wegman. Proceedings of the eighth annual ACM symposium on Theory of Computing, 1976.

# First-order resolution

- *Given clauses* $(a \vee b \vee c)$, $(\neg c' \vee d \vee e)$, *and a substitution list V unifying c and c'*
- *Conclude* $(a \vee b \vee d \vee e) : V$

# Example

$raining \wedge outside(x) \Rightarrow wet(x)$

$wet(x) \Rightarrow rusty(x) \vee rustproof(x)$

$robot(x) \Rightarrow \neg rustproof(x)$

$raining$

$guidebot(Bobby)$

$guidebot(x) \Rightarrow robot(x) \wedge outside(x)$

# First-order factoring

- *When removing redundant literals, we have the option of unifying them first*
- *Given clause $(a \vee b \vee c)$, substitution $V$*
- *If $a : V$ and $b : V$ are the same*
- *Then we can conclude $(a \vee c) : V$*

# Completeness

- *First-order resolution (together with first-order factoring) is sound and complete for FOL*
- *Famous theorem*

# Completeness

# Proof strategy

- *We'll show FOL completeness by reducing to propositional completeness*
- *To prove S, put KB ∧ ¬S in clause form*
- *Turn FOL KB into propositional KBs*
  - *in general, infinitely many*
- *Check each one in order*
- *If any one is unsatisfiable, we will have our proof*

# Propositionalization

- *Given a FOL KB in clause form*
- *And a set of terms U (for **universe**)*
- *We can **propositionalize** KB under U by substituting elements of U for free variables in all combinations*

# Propositionalization example

- $(\neg man(x) \vee mortal(x))$
- $mortal(Socrates)$
- $favorite\_drink(Socrates) = hemlock$
- $drinks(x, favorite\_drink(x))$

- $U = \{Socrates, hemlock, Fred\}$

# Propositionalization example

- $(\neg man(Socrates) \vee mortal(Socrates))$
  $(\neg man(Fred) \vee mortal(Fred))$
  $(\neg man(hemlock) \vee mortal(hemlock))$
- $drinks(Socrates, favorite\_drink(Socrates))$
  $drinks(hemlock, favorite\_drink(hemlock))$
  $drinks(Fred, favorite\_drink(Fred))$
- $mortal(Socrates) \wedge$
  $favorite\_drink(Socrates) = hemlock$

# Choosing a universe

- *To check a FOL KB, propositionalize it using some universe U*
- *Which universe?*

# Herbrand Universe

*Jacques Herbrand*
*1908–1931*

- ***Herbrand universe** H of formula S:*
  - *start with all objects mentioned in S*
  - *or synthetic object X if none mentioned*
  - *apply all functions mentioned in S to all combinations of objects in H, add to H*
  - *repeat*

# Herbrand Universe

- *E.g., loves(uncle(John), Mary) yields*

  *H = {John, Mary, uncle(John), uncle(Mary), uncle(uncle(John)), uncle(uncle(Mary)), …}*

# Herbrand's theorem

- *If a FOL KB in clause form is unsatisfiable*
- *And H is its Herbrand universe*
- *Then the propositionalized KB is unsatisfiable for some* ***finite*** $U \subseteq H$

# Significance

- *This is one half of the equivalence we want: unsatisfiable FOL KB* $\Rightarrow \exists$ *finite U. unsatisfiable propositional KB*

# Example

- $(\neg man(x) \vee mortal(x)) \wedge man(uncle(Socrates)) \wedge \neg mortal(x)$
- $H = \{S, u(S), u(u(S)), \ldots\}$
- *If* $U = \{u(S)\}$, *PKB* =

  $(\neg man(u(S)) \vee mortal(u(S))) \wedge man(u(S)) \wedge \neg mortal(u(S))$
- *Resolving twice yields F*

# Converse of Herbrand

- *A. J. Robinson proved "lifting lemma"*
- *Write PKB for a propositionalization of KB (under some universe)*
- *Any resolution proof in PKB corresponds to a resolution proof in KB*
- *…and, if PKB is unsatisfiable, there is a proof of F (by prop. completeness); so, lifting it shows KB unsatisfiable*

# Example

- $(\neg man(u(S)) \vee mortal(u(S))) \wedge man(u(S)) \wedge \neg mortal(u(S))$
- *We resolved on man(u(S)) yielding mortal(u(S))*
- *Lifted, resolve ¬man(x) w/ man(u(S)), binding* $x \rightarrow u(S)$

# Proofs w/ Herbrand & Robinson

- *So, FOL KB is unsatisfiable* ***if and only if*** *there is a subset of its Herbrand universe making PKB unsatisfiable*
- *I.e., if we have a way to find proofs in propositional logic, we have a way to find them in FOL*

# Proofs w/ Herbrand & Robinson

- *To prove S, put KB ∧ ¬S in CNF: KB'*
- *Build subsets of Herbrand universe in increasing order of size: $U_1$, $U_2$, …*
- *Propositionalize KB' w/ $U_i$, look for proof*
- *If $U_i$ unsatisfiable, use lifting to get a contradiction in KB'*
- *If $U_i$ satisfiable, move on to $U_{i+1}$*

# How long will this take?

- *If S is not entailed, we will never find a contradiction*
- *In this case, if H infinite, we'll never stop*
- *So, entailment is* ***semidecidable***
  - *equivalently, entailed statements are* ***recursively enumerable***

# Variation

- *Restrict semantics so we only need to check one finite propositional KB*
- ***Unique names**: objects with different names are different (John ≠ Mary)*
- ***Domain closure**: objects without names given in KB don't exist*
- *Restrictions also make entailment, validity feasible*

# Who? What? Where?

# Wh-questions

- *We've shown how to prove a statement like mortal(Socrates)*
- *What if we have a question whose answer is not just yes/no, like "who killed JR?" or "where is my robot?"*
- *Simplest approach: prove* $\exists x$. *killed(x, JR), hope the proof is constructive*


54

# Answer literals

- *Simple approach doesn't always work*
- *Instead of ¬S(x), add (¬S(x) ∨ answer(x))*
- *If there's a contradiction, we can eliminate ¬S(x) by resolution and unification, leaving answer(x)* ***with x bound*** *to a value that causes a contradiction*


55

# Example

$$Kills(Jack, Cat) \vee Kills(Curiosity, Cat)$$

$$\neg Kills(Jack, x)$$

# FOL Extensions

# Equality

- ***Paramodulation*** *is sound and complete for FOL+equality (see RN)*
- *Or, resolution +* ***axiom schema***

# Second order logic

- *SOL adds quantification over predicates*
- *E.g., principle of mathematical induction:*
  - $\forall P.\ P(0) \land (\forall x.\ P(x) \Rightarrow P(S(x))) \Rightarrow \forall x.\ P(x)$
- *There is no sound and complete inference procedure for SOL (Gödel's famous incompleteness theorem)*

# Others

- *Temporal logics ("P(x) will be true at some time in the future")*
- *Modal logics ("John believes P(x)")*
- *Nonmonotonic FOL*
- *First-class functions (lambda operator, application)*
- …

# Using FOL

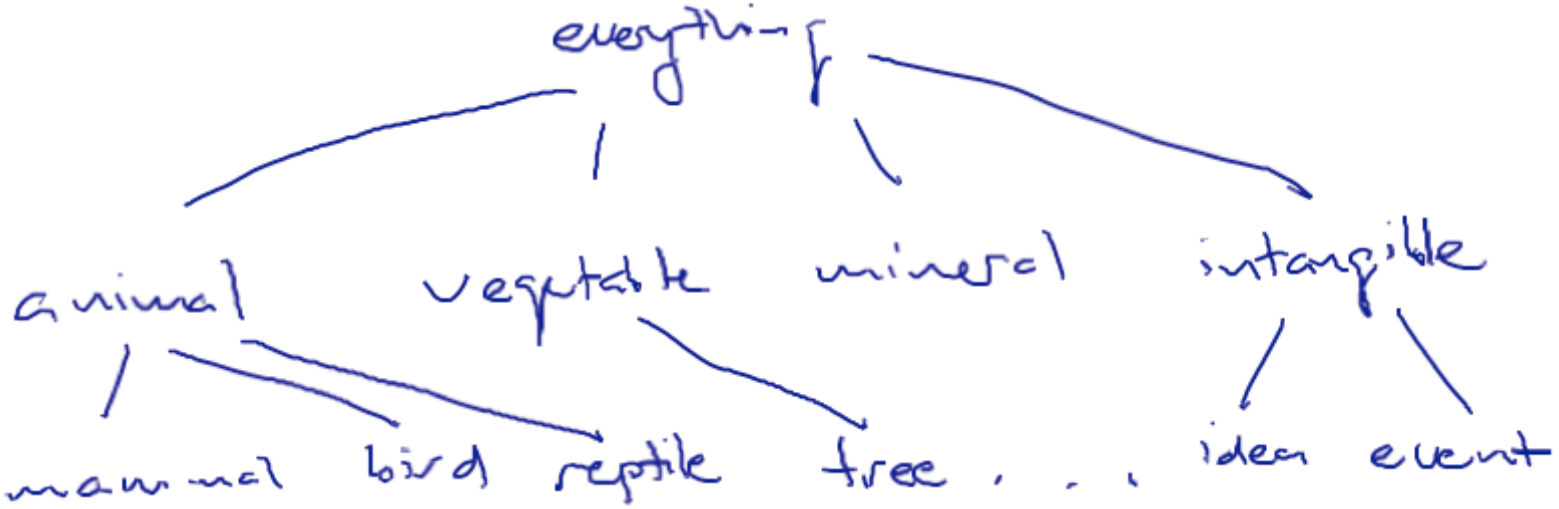
# Knowledge engineering

- *Identify relevant objects, functions, and predicates*
- *Encode general background knowledge about domain (reusable)*
- *Encode specific problem instance*
- *Pose queries (is P(x) true? Find x such that P(x))*

# Common themes

- *RN identifies many common idioms and problems for knowledge engineering*
- *Hierarchies, fluents, knowledge, belief, …*
- *We'll look at a couple*

# Taxonomies




- *isa(Mammal, Animal)*
- *disjoint(Animal, Vegetable)*
- *partition({Animal, Vegetable, Mineral, Intangible}, Everything)*

# Inheritance

- *Transitive: isa(x, y) ∧ isa(y, z) ⇒ isa(x, z)*
- *Attach properties anywhere in hierarchy*
  - *isa(Pigeon, Bird)*
  - *isa(x, Bird) ⇒ flies(x)*
  - *isa(x, Pigeon) ⇒ gray(x)*
- *So, isa(Tweety, Pigeon) tells us Tweety is gray and flies*

# Physical composition

- *partOf(Wean4625, WeanHall)*
- *partOf(water37, water)*
- *Note distinction between **mass** and **count** nouns: any partOf a mass noun is also an example of that same mass noun*

# Fluents

- *Fluent = property that changes over time*
  - *at(Robot, Wean4623, 11AM)*
- *Actions change fluents*
- *Fluents chain together to form possible worlds*
- $at(x, p, t) \wedge adj(p, q) \Rightarrow poss(go(x, p, q)) \wedge at(x, q, result(go(x, p, q), t))$

# Frame problem

- *Suppose we execute an unrelated action (e.g., talk(Professor, FOL))*
- *Robot shouldn't move:*
  - *if at(Robot, Wean4623, t), want at(Robot, Wean4623, result(talk(Professor, FOL)))*
- *But we can't prove it using tools described so far!*

# Frame problem

- *The **frame problem** is that it's a pain to list all of the things that don't change when we execute an action*
- *Naive solution: **frame axioms***
  - *for each fluent, list actions that can't change fluent*
  - *KB size: O(AF) for A actions, F fluents*

# Frame problem

- *Better solution:* ***successor-state*** *axioms*
- *For each fluent, list actions that* ***can*** *change it (typically fewer): if go(x, p, q) is possible,*

  $at(x, q, result(a, t)) \Leftrightarrow$
  $a = go(x, p, q) \vee (at(x, q, t) \wedge a \neq go(y, z))$

- *Size O(AE+F) if each action has E effects*

# Sadly, also necessary…

- *Debug knowledge base*
  - *Severe bug: logical contradictions*
  - *Less severe: undesired conclusions*
  - *Least severe: missing conclusions*
- *First 2: trace back chain of reasoning until reason for failure is revealed*
- *Last: trace desired proof, find what's missing*

# SAT

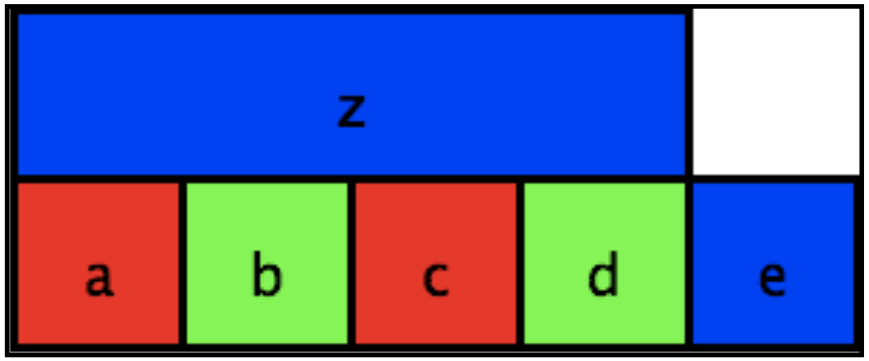
# Satisfiability

- *SAT is the problem of determining whether a given propositional logic sentence is satisfiable*
- *A **decision problem**: given an instance, answer yes or no*
- *A fundamental problem in CS*

# SAT is a general problem

- *Many other useful decision problems* ***reduce*** *to SAT*
- *Informally, if we can solve SAT, we can solve these other problems*
- *So a good SAT solver is a good AI building block*

# Example decision problem




- *k-coloring: can we color a map using only k colors in a way that keeps neighboring regions from being the same color?*

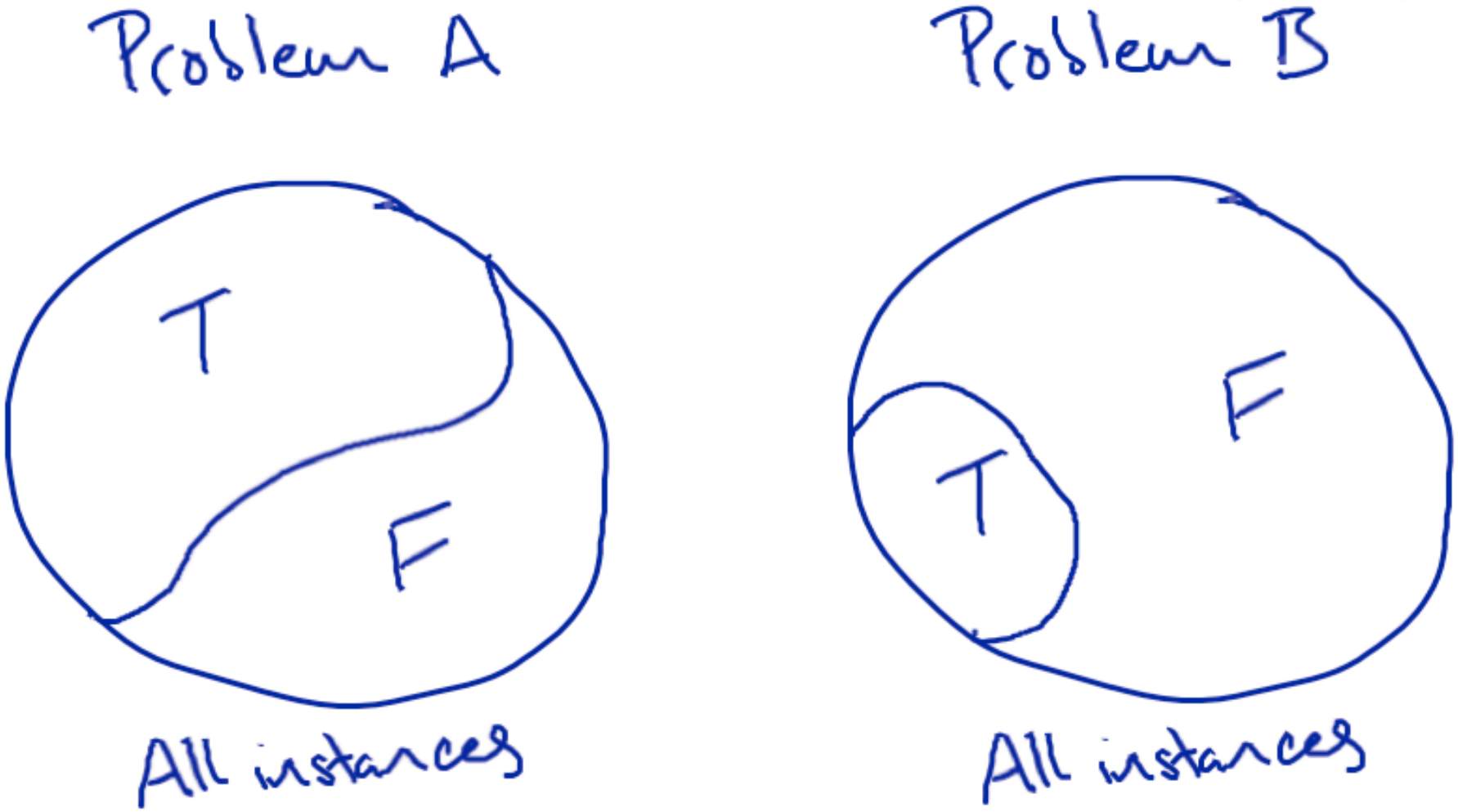
# Example decision problem

- *Propositional planning: given a list of operators (w/ preconditions, effects), can we apply operators in some order to achieve a desired goal?*
- *Have cake & eat it too example*
- *We'll see later how to represent as SAT*

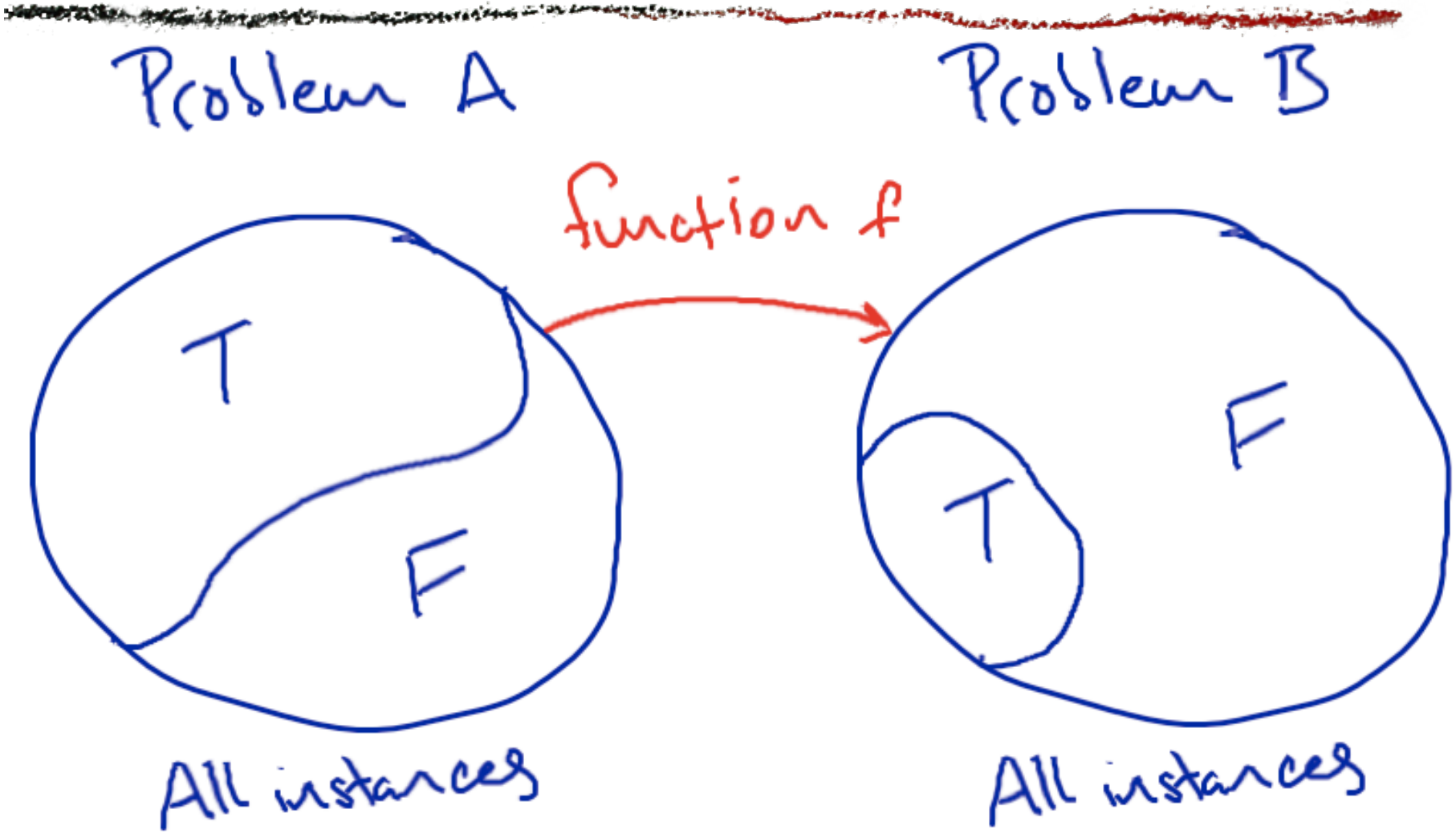
# Reduction

- *Loosely, "A reduces to B" means that if we can solve B then we can solve A*
- *More formally, A, B are decision problems (instances $\mapsto$ truth values)*
- *A reduction is a poly-time function f such that, given an instance a of A*
  - *f(a) is an instance of B, and*
  - *A(a) = B(f(a))*

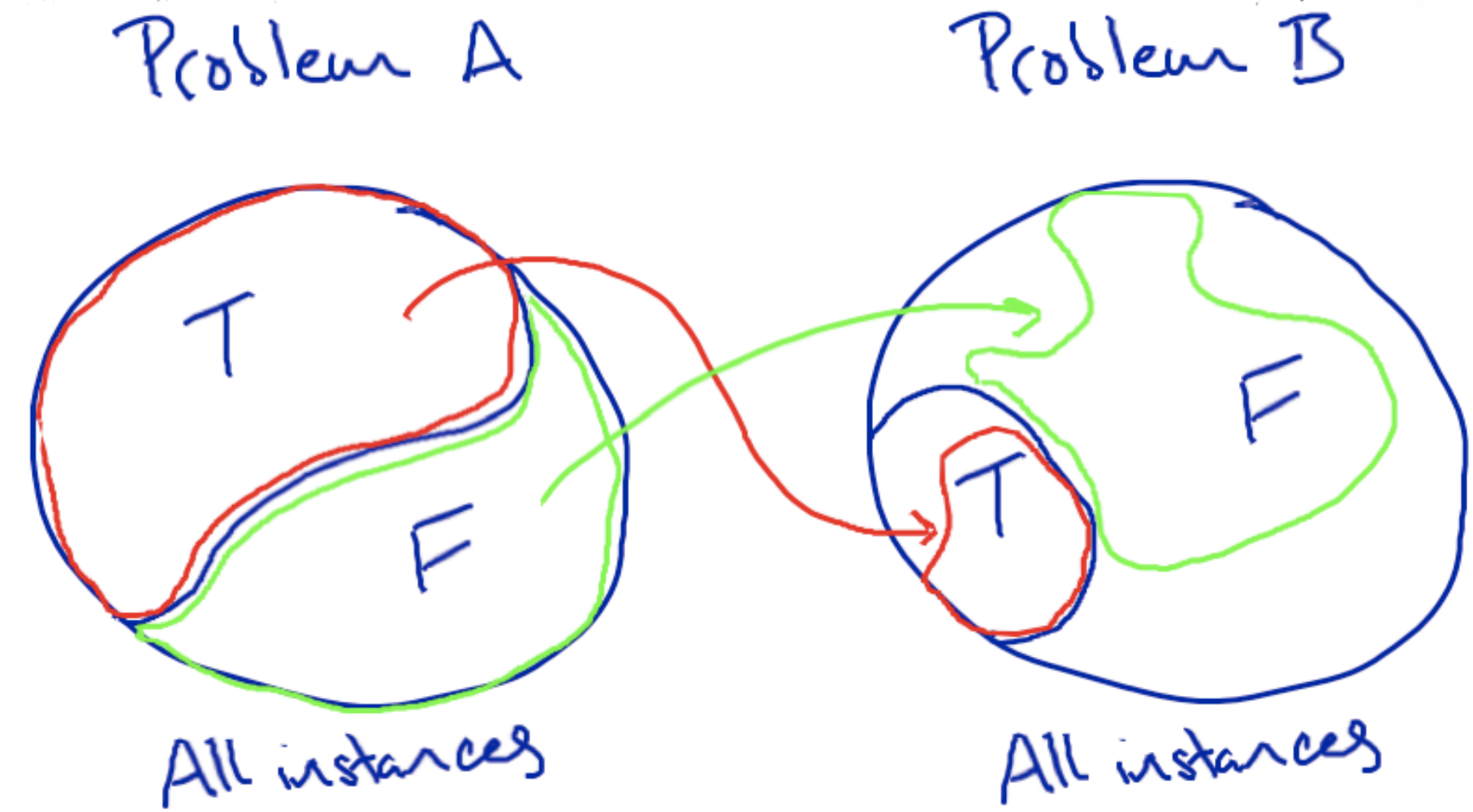
# Reduction picture



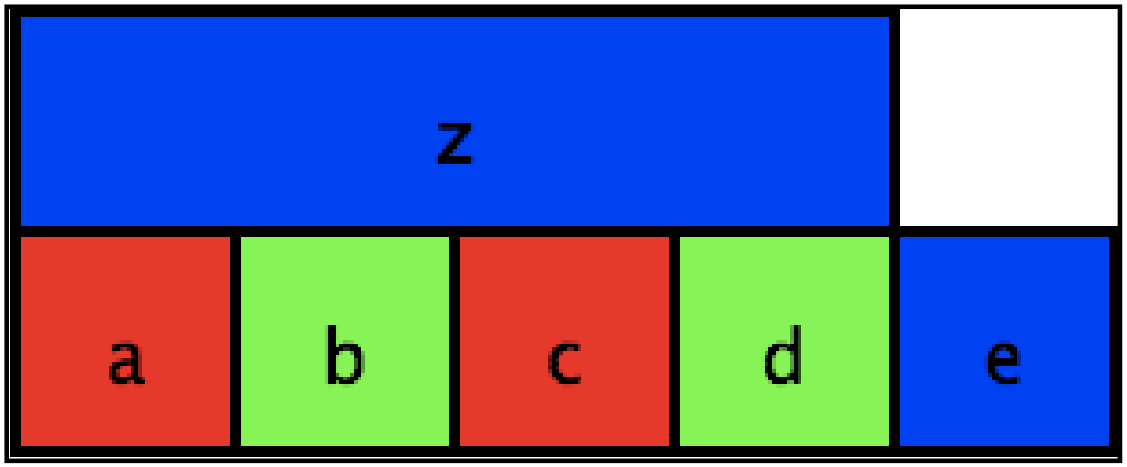
# Reduction picture



# Reduction picture



# Back to example

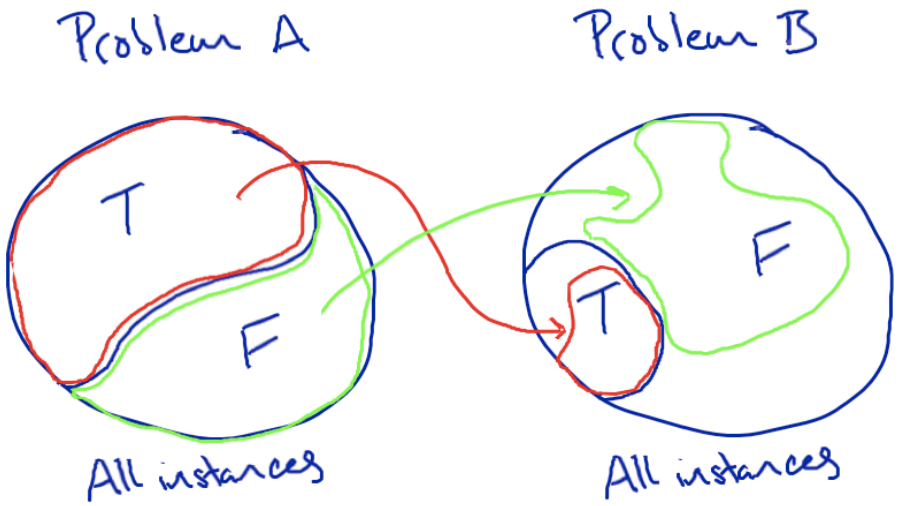



- *Each square must be red, green, or blue*
- *Adjacent squares can't both be red (similarly, green or blue)*

# Back to example

- $(a_r \vee a_g \vee a_b) \wedge (b_r \vee b_g \vee b_b) \wedge (c_r \vee c_g \vee c_b) \wedge (d_r \vee d_g \vee d_b) \wedge (e_r \vee e_g \vee e_b) \wedge (z_r \vee z_g \vee z_b)$
- $(\neg a_r \vee \neg b_r) \wedge (\neg a_g \vee \neg b_g) \wedge (\neg a_b \vee \neg b_b)$
- $(\neg a_r \vee \neg z_r) \wedge (\neg a_g \vee \neg z_g) \wedge (\neg a_b \vee \neg z_b)$
- …

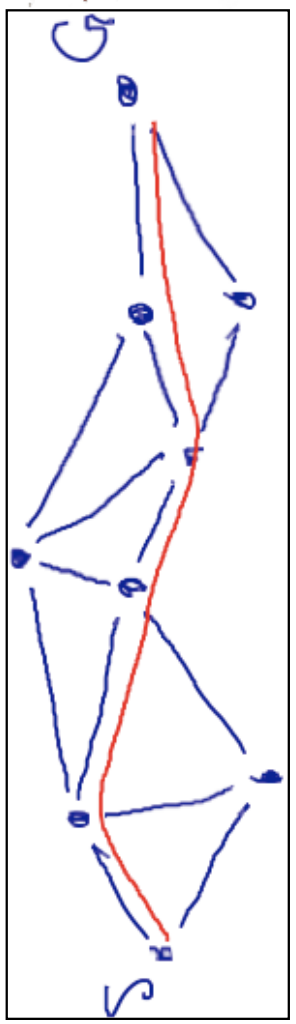
# Direction of reduction




- *If A reduces to B:*
  - *if we can solve B, we can solve A*
  - *so B must be at least as hard as A*
- *Trivially, can take an easy problem and reduce it to a hard one*

# Not-so-useful reduction

- *Path planning reduces to SAT*
- *Variables: is edge e in path?*
- *Constraints:*
  - *exactly 1 path-edge touches start*
  - *exactly 1 path-edge touches goal*
  - *either 0 or 2 touch each other node*



# More useful: reduction to 3SAT

- *We saw that decision problems can be reduced to SAT*
  - *is CNF formula satisfiable?*
- *Can reduce even further, to 3SAT*
  - *is 3CNF formula satisfiable?*
- *Useful if reducing SAT/3SAT to another problem (to show other problem hard)*

# Reduction to 3SAT

- *Must get rid of long clauses*
- *E.g.,* $(a \vee \neg b \vee c \vee d \vee e \vee \neg f)$
- *Replace with*

$$(a \vee \neg b \vee x) \wedge (\neg x \vee c \vee y) \wedge$$
$$(\neg y \vee d \vee z) \wedge (\neg z \vee e \vee \neg f)$$

# NP-completeness

- *S. A. Cook in 1971 proved that many useful decision problems reduce back and forth to SAT*
  - *showed how to simulate poly-size-memory computer w/ (very complicated, but still poly-size) SAT problem*
- *Equivalently, SAT is exactly as hard (in theory at least) as these other problems*

# Cost of reduction

- *Complexity theorists often ignore little things like constant factors (or even polynomial factors!)*
- *So, is it a good idea to reduce your decision problem to SAT?*
- *Answer: sometimes…*

# Cost of reduction

- *SAT is well studied ⇒ fast solvers*
- *So, if there is an efficient reduction, ability to use fast SAT solvers can be a win*
  - *e.g., 3-coloring*
  - *another example later (SATplan)*
- *Other times, cost of reduction is too high*
  - *usu. because instance gets bigger*
  - *will also see example later (MILP)*

# Choosing a reduction

- *May be many reductions from problem A to problem B*
- *May have **wildly** different properties*
  - *e.g., solving transformed instance may take seconds vs. days*