# 15-780: Graduate AI
## *Lecture 3. Logic and SAT*

*Geoff Gordon (this lecture)*
*Ziv Bar-Joseph*
*TAs Geoff Hollinger, Henry Lin*

# Admin

- *HW1 out today!*
    - *On course website*
    - *Due Thu 10/4*
- *Reminder: Matlab tutorial today*
    - *NSH 1507, 5PM*

# Working together

- *Working together on HW, looking on web, etc..: great idea!*
    - *but each person must write up and submit his/her own solution, without reference to written/electronic materials from web or other students*
- *Last year's HWs are on course web site*

# Late policy

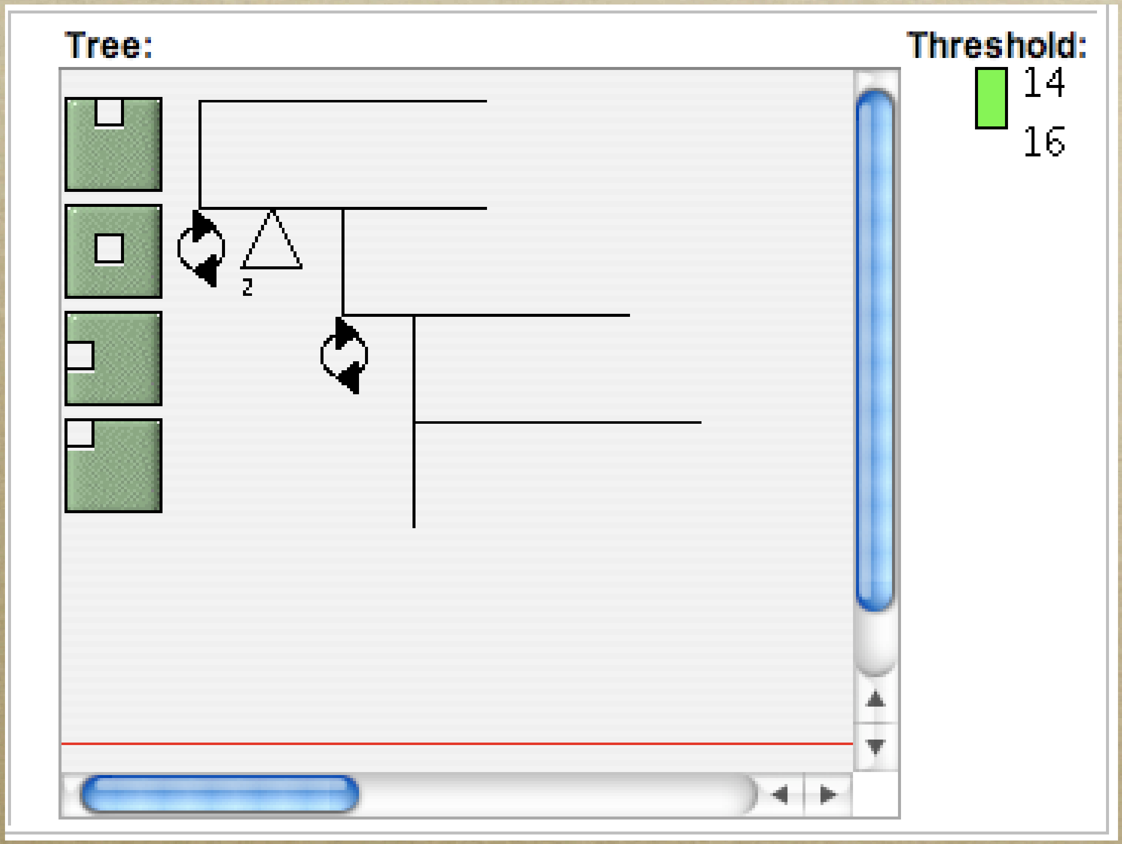
- *If you need to hand a HW in late: contact us* ***before*** *due date*
- *Unless agreed otherwise, HW is worth 75% credit up to 24 hrs late, 50% credit up to 48 hrs late, 0% credit afterwards*
- *Even if for 0% credit, must hand in* ***all*** *assignments to pass*

# Review

# Topics covered

- *C-space*
- *Ways of splitting up C-space*
  - *Visibility graph*
  - *Voronoi*
  - *Exact, approximate cell decomposition*
  - *Adaptive cells (quadtree, parti-game)*
- *RRTs*

# 8/15 puzzle applet




*http://www.cs.ualberta.ca/~aixplore/search/IDA/Applet/SearchApplet.html*

# Project ideas

# Poker

# Poker

- *Minimax strategy for heads-up poker = solving linear program*
- *1-card hands, 13-card deck: 52 vars, <u>instantaneous</u>*
- *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)*

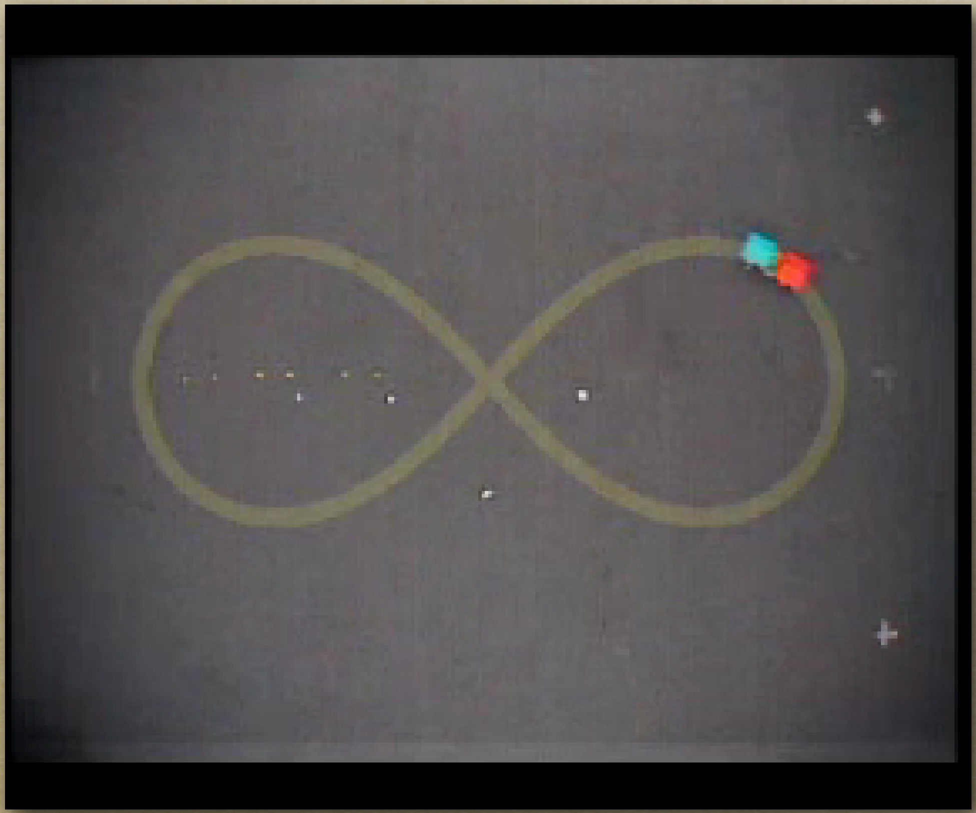
# Scrabble™

- *Can buy a hand-tweaked, very good computer Scrabble player for $30 or so*
- *Can we learn to beat it?*

# Learning models for control

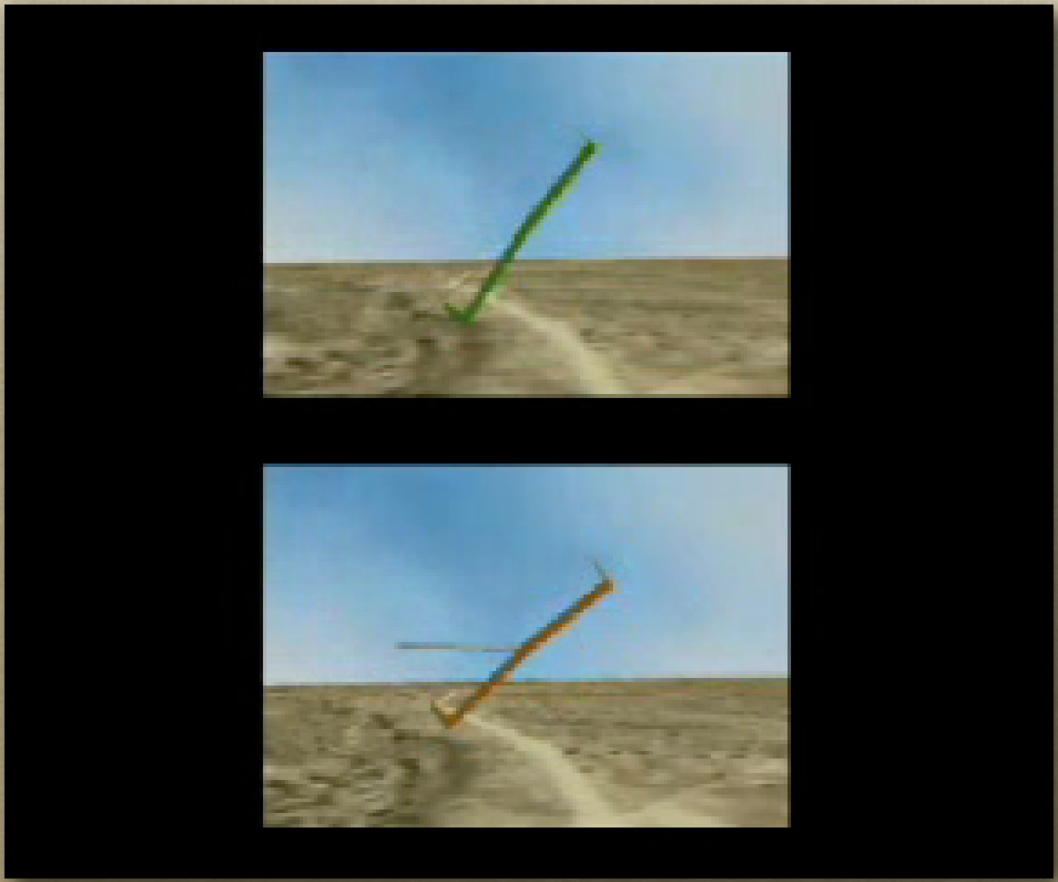
- *Most of this course, we'll assume we have a good model of the world when we're trying to plan*
- *Usually not true in practice—must learn it*
- *Project: learn a model for an interesting system, write a planner for learned model, make planner work on original system*

# Learning models for control



- *R/C car*

# Learning models for control



- *Model airplane*

# Citation

- *"Using Inaccurate Models in Reinforcement Learning." Pieter Abbeel, Morgan Quigley, Andrew Y. Ng*

  *http://www.icml2006.org/icml_documents/camera-ready/001_Using_Inaccurate_Mod.pdf*

# Logic

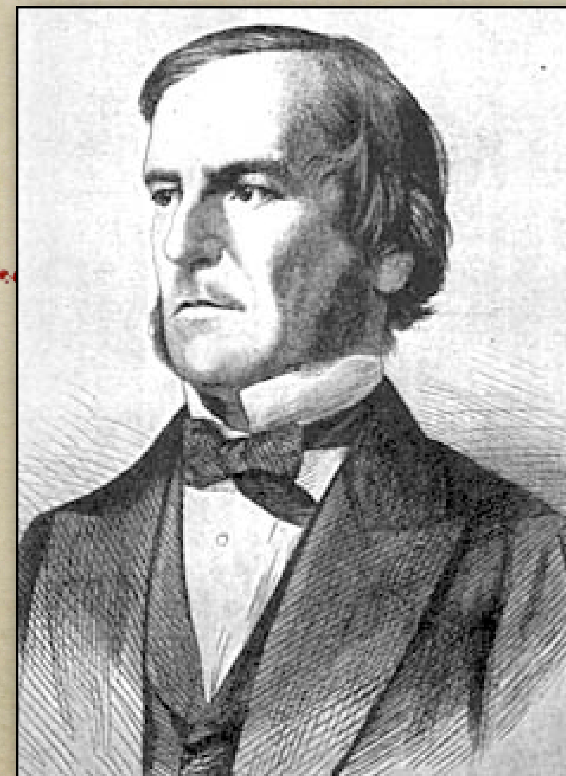
# Why logic?

- *Search: for problems like 8-puzzle, can write compact description of rules*
- *Reasoning: figure out consequences of the knowledge we've given our agent*
- *Foreshadowing: logical inference is a special case of probabilistic inference (Part II)*

# Propositional logic

- *Constants: T or F*
- *Variables: x, y (values T or F)*
- *Connectives:* $\wedge$, $\vee$, $\neg$
  - *Can get by w/ just NAND*
  - *Sometimes also add others:*

    $\oplus$, $\Rightarrow$, $\Leftrightarrow$, …



*George Boole*
*1815–1864*

# Propositional logic

- *Build up expressions like* $\neg x \Rightarrow y$
- *Precedence:* $\neg, \wedge, \vee, \Rightarrow$
- *Terminology: variable or constant with or w/o negation =* ***literal***
- *Whole thing =* ***formula*** *or* ***sentence***

# Expressive variable names

- *Rather than variable names like x, y, may use names like "rains" or "happy(John)"*
- *For now, "happy(John)" is just a string with no internal structure*
    - *there is no "John"*
    - *happy(John)* $\Rightarrow \neg$ *happy(Jack) means the same as* $x \Rightarrow \neg y$

# But what does it mean?

- *A formula defines a mapping*

  *(assignment to variables)* $\mapsto \{T, F\}$

- *Assignment to variables =* ***model***
- *For example, formula* $\neg x$ *yields mapping:*

| $x$ | $\neg x$ |
|---|---|
| $T$ | $F$ |
| $F$ | $T$ |

# More truth tables

| $x$ | $y$ | $x \wedge y$ |
|---|---|---|
| $T$ | $T$ | $T$ |
| $T$ | $F$ | $F$ |
| $F$ | $T$ | $F$ |
| $F$ | $F$ | $F$ |

| $x$ | $y$ | $x \vee y$ |
|---|---|---|
| $T$ | $T$ | $T$ |
| $T$ | $F$ | $T$ |
| $F$ | $T$ | $T$ |
| $F$ | $F$ | $F$ |

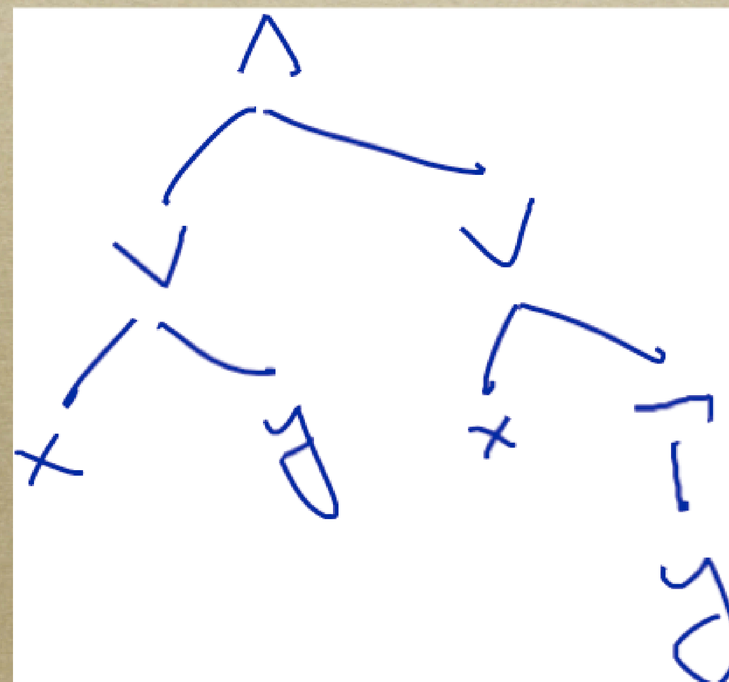
# Truth table for implication

- *$(a \Rightarrow b)$ is logically equivalent to $(\neg a \vee b)$*
- *If a is True, b must be True too*
- *If a False, no requirement on b*
- *E.g., "if I go to the movie I will have popcorn": if no movie, may or may not have popcorn*

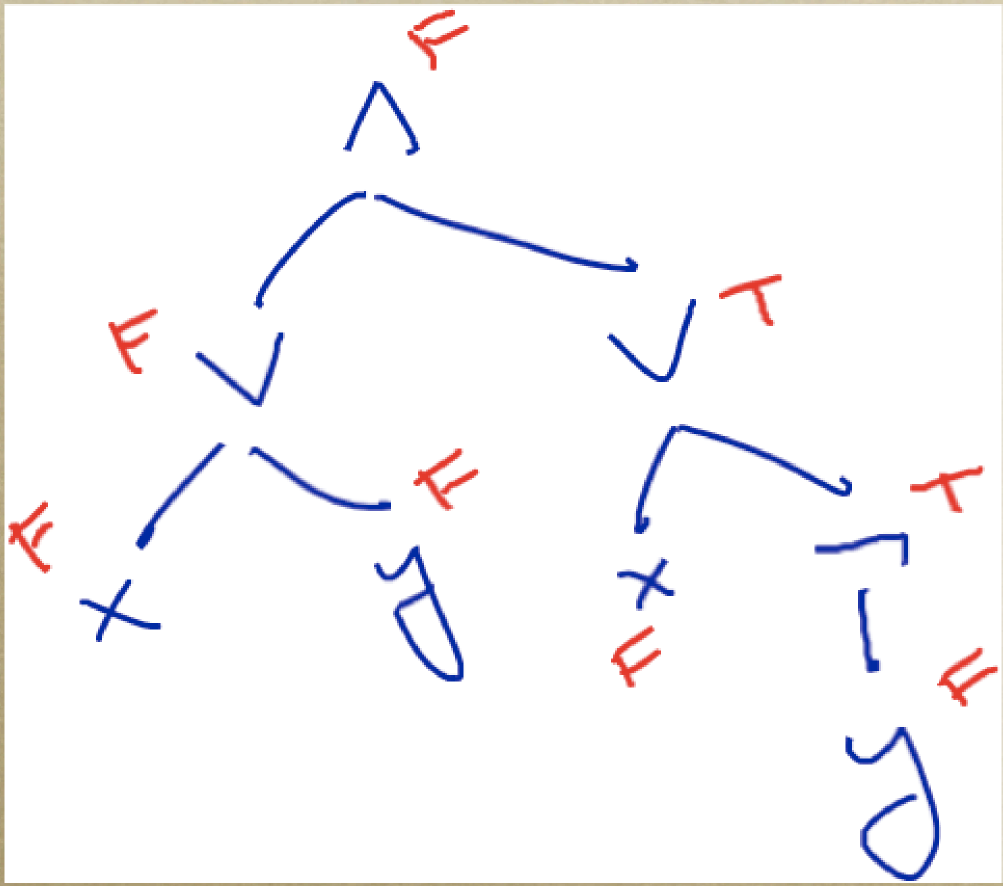
| $a$ | $b$ | $a \Rightarrow b$ |
|---|---|---|
| $T$ | $T$ | $T$ |
| $T$ | $F$ | $F$ |
| $F$ | $T$ | $T$ |
| $F$ | $F$ | $T$ |

# Complex formulas

- *To evaluate a bigger formula*
  - $(x \vee y) \wedge (x \vee \neg y)$ *when* $x = F$, $y = F$
- *Build a parse tree*
- *Fill in variables at leaves using model*
- *Work upwards using truth tables for connectives*



# Example



- $(x \vee y) \wedge (x \vee \neg y)$ *when* $x = F, y = F$

# Bigger truth tables

| $x$ | $y$ | $z$ | $(x \vee y) \Rightarrow z$ |
|---|---|---|---|
| $T$ | $T$ | $T$ | |
| $T$ | $T$ | $F$ | |
| $T$ | $F$ | $T$ | |
| $T$ | $F$ | $F$ | |
| $F$ | $T$ | $T$ | |
| $F$ | $T$ | $F$ | |
| $F$ | $F$ | $T$ | |
| $F$ | $F$ | $F$ | |

# Working with formulas

# Definitions

- *Two sentences are* ***equivalent****,* $A \equiv B$*, if they have same truth value in every model*
  - $(rains \Rightarrow pours) \equiv (\neg rains \lor pours)$
  - *reflexive, transitive, commutative*
- ***Simplifying*** *= transforming a formula into a shorter*, equivalent formula*

# Transformation rules

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$
$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$
$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$
$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$
$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$

$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$
$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

*α, β, γ are arbitrary formulas*

# More rules

$$
\begin{aligned}
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan}
\end{aligned}
$$

*$\alpha$, $\beta$ are arbitrary formulas*

# Still more rules…

- *… can be derived from truth tables*
- *For example:*
  - $(a \lor \neg a) \equiv True$
  - $(True \lor a) \equiv True$
  - $(False \land a) \equiv False$

# Example

$$(a \vee (b \wedge c)) \wedge \neg(b \wedge \neg e)$$

$$(a \vee b) \wedge (a \vee c) \wedge (\neg b \vee e)$$

# Normal Forms

# Normal forms

- *A normal form is a standard way of writing a formula*
- *E.g.,* ***conjunctive normal form*** *(CNF)*
  - *conjunction of disjunctions of literals*
  - $(x \lor y \lor \neg z) \land (x \lor \neg y) \land (z)$
  - *Each disjunct called a* ***clause***
- *Any formula can be transformed into CNF w/o changing meaning*

# CNF cont'd

$$\begin{array}{|l|}\hline
\mathit{happy}(\mathit{John}) \land \\
(\neg \mathit{happy}(\mathit{Bill}) \lor \mathit{happy}(\mathit{Sue})) \land \\
\mathit{man}(\mathit{Socrates}) \land \\
(\neg \mathit{man}(\mathit{Socrates}) \lor \mathit{mortal}(\mathit{Socrates})) \\ \hline
\end{array}$$

- *Often used for storage of knowledge database*
  - *called* ***knowledge base*** *or KB*
- *Can add new clauses as we find them out*
- *Each clause in KB is separately true (if KB is)*

# Another normal form: DNF

- *DNF = disjunctive normal form = disjunction of conjunctions of literals*
- *Doesn't compose the way CNF does: can't just add new conjuncts w/o changing meaning of KB*
- *Example:*

$$\boxed{\begin{array}{c} (rains \lor \neg pours) \land fishing \\ \equiv \\ (rains \land fishing) \lor (\neg pours \land fishing) \end{array}}$$

# Transforming to CNF or DNF

- *Naive algorithm:*
  - *replace all connectives with* $\wedge \vee \neg$
  - *move negations inward using De Morgan's laws and double-negation*
  - *repeatedly distribute over* $\wedge$ *over* $\vee$ *for DNF (*$\vee$ *over* $\wedge$ *for CNF)*

# Example

- *Put the following formula in CNF*

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

# Example

- *Now try DNF*

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

# Discussion

- *Problem with naive algorithm: it's exponential!  (Space, time, size of result.)*
- *Each use of distributivity can almost double the size of a subformula*

# A smarter transformation

- *Can we avoid exponential blowup in CNF?*
- *Yes, if we're willing to introduce new variables*
- *G. Tseitin. On the complexity of derivation in propositional calculus. Studies in Constrained Mathematics and Mathematical Logic, 1968.*

# Example

- *Put the following formula in CNF:*

$$(a \wedge b) \vee (c \wedge d)$$

# Proofs

# Entailment

- *Sentence A **entails** sentence B, $A \models B$, if B is True in every model where A is*
  - *same as saying that $(A \Rightarrow B)$ is valid*

# Proof tree

- *A tree with a formula at each node*
- *At each internal node, children* $\vDash$ *parent*
- *Leaves:* ***assumptions*** *or* ***premises***
- *Root:* ***consequence***
- *If we believe assumptions, we should also believe consequence*

# Proof tree example

rains $\Rightarrow$ pours

pours $\wedge$ outside $\Rightarrow$ rusty

rains

outside

# Proof tree example

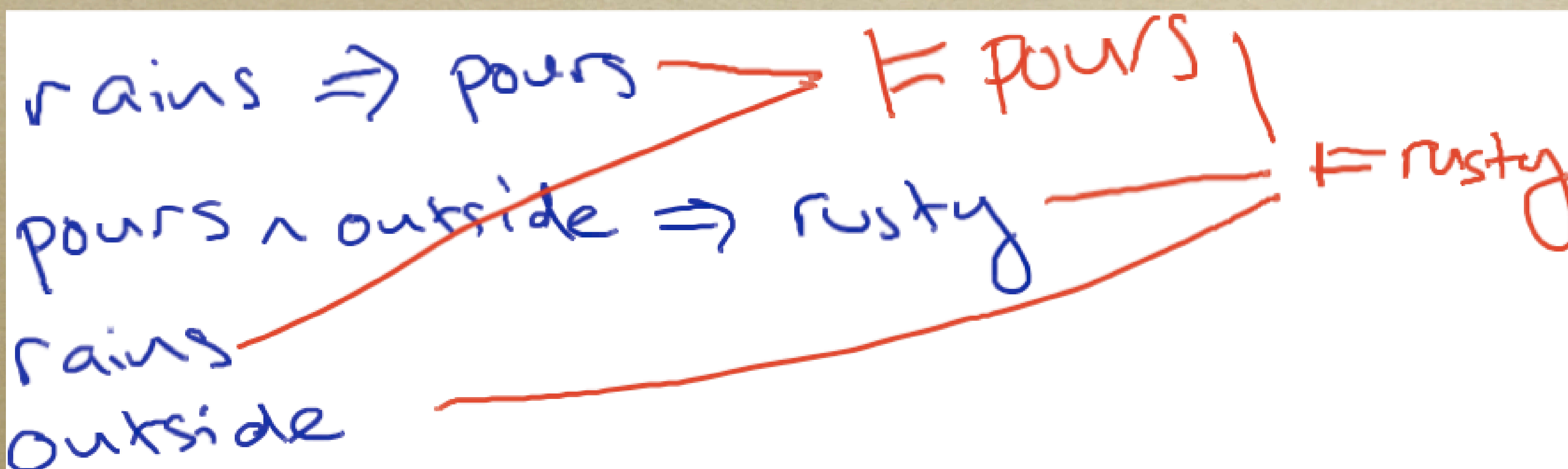
$rains \Rightarrow pours$ $\models pours$

$pours \wedge outside \Rightarrow rusty$

$rains$

$outside$

# Proof tree example



# Proof by contradiction

- *Assume opposite of what we want to prove, show it leads to a contradiction*
- *Suppose we want to show* $KB \models S$
- *Write KB' for* $(KB \land \neg S)$
- *Build a proof tree with*
  - *assumptions drawn from clauses of KB'*
  - *conclusion* $= F$
  - *so,* $(KB \land \neg S) \models F$ *(contradiction)*

# Proof by contradiction

KB

$rains \Rightarrow pours$

$pours \wedge outside \Rightarrow rusty$

$rains$

$outside$

---

$\neg rusty$

negation of desired conclusion

# Proof by contradiction

KB

$rains \Rightarrow pours$

$pours \wedge outside \Rightarrow rusty$

$rains$

$outside$

$\neg rusty$

negation of desired conclusion

$\models pours$

$\models rusty$

$\models F$

# Inference rules

# Inference rule

- *To make a proof tree, we need to be able to figure out new formulas entailed by KB*
- *Method for finding entailed formulas =* ***inference rule***
- *We've implicitly been using one already*

# Modus ponens

$$\frac{(a \wedge b \wedge c \Rightarrow d) \quad a \quad b \quad c}{d}$$

- *Probably most famous inference rule: all men are mortal, Socrates is a man, therefore Socrates is mortal*
- *Quantifier-free version:*

$man(Socrates) \wedge$

$(man(Socrates) \Rightarrow mortal(Socrates))$

# Another inference rule

$$\frac{(a \Rightarrow b) \quad \neg b}{\neg a}$$

- ***Modus tollens***
- *If it's raining the grass is wet; the grass is not wet, so it's not raining*

# One more…

$$\frac{(a \vee b \vee c) \quad (\neg c \vee d \vee e)}{a \vee b \vee d \vee e}$$

- ***Resolution***
- *Combines two sentences that contain a literal and its negation*
- *Not as commonly known as modus ponens / tollens*

# Resolution example

- *Modus ponens / tollens are special cases*
- *Modus tollens:*

  $(\neg \mathit{raining} \lor \mathit{grass\text{-}wet}) \land \neg \mathit{grass\text{-}wet} \models \neg \mathit{raining}$

# Resolution

$$\frac{(a \vee b \vee c) \quad (\neg c \vee d \vee e)}{a \vee b \vee d \vee e}$$

- *Simple proof by case analysis*
- *Consider separately cases where we assign c = True and c = False*

# Resolution

$$(a \lor b \lor c) \land (\neg c \lor d \lor e)$$

- *Case c = True*

$$(a \lor b \lor T) \land (F \lor d \lor e)$$

$$= (T) \land (d \lor e)$$

$$= (d \lor e)$$

# Resolution

$$(a \vee b \vee c) \wedge (\neg c \vee d \vee e)$$

- *Case c = False*

$$(a \vee b \vee F) \wedge (T \vee d \vee e)$$

$$= (a \vee b) \wedge (T)$$

$$= (a \vee b)$$

# Resolution

$$(a \lor b \lor c) \land (\neg c \lor d \lor e)$$

- *Since c must be True or False, conclude*

  $$(d \lor e) \lor (a \lor b)$$

  *as desired*

# Theorem provers

# Theorem prover

- *Theorem prover = mechanical system for finding a proof tree*
- *An application of search techniques from earlier lecture*
  - *Search node = KB (including whatever we've proven so far)*
  - *Neighbor:* $(KB \wedge S)$ *if* $KB \vDash S$

# A basic theorem prover

- *Given KB, want to conclude S*
- *Let KB' = CNF(KB* $\wedge$ $\neg$*S)*
- *Repeat:*
  - *add new clause to KB' using resolution*
- *Until we add empty clause (False) and conclude KB* $\models$ *S*
- *Or run out of new clauses and conclude KB* $\not\models$ *S*

# Soundness and completeness

- *An inference procedure is **sound** if it can only conclude things entailed by KB*
    - *common sense; haven't discussed anything unsound*
- *A set of rules is **complete** if it can conclude everything entailed by KB*

# Completeness

- *Theorem provers based on modus ponens by itself are* ***incomplete***
- *Simple resolution theorem prover from above is* ***complete*** *for propositional logic*

# Variations

- *Horn clause inference (faster)*
- *Ways of handling uncertainty (slower)*
- *CSPs (sometimes more convenient)*
- *Quantifiers / first-order logic*

*Later*

# Horn clauses

- *Horn clause:* $(a \wedge b \wedge c \Rightarrow d)$
- *Equivalently,* $(\neg a \vee \neg b \vee \neg c \vee d)$
- *Disjunction of literals,* ***at most one*** *of which is positive*
- *Positive literal =* ***head****, rest =* ***body***

# Use of Horn clauses

- *People find it easy to write Horn clauses (listing out conditions under which we can conclude head)*

  $$happy(John) \wedge happy(Mary) \Rightarrow happy(Sue)$$

- *No negative literals in above formula; again, easier to think about*

# Why are Horn clauses important

- *Inference in a KB of propositional Horn clauses is linear*
- *E.g., by forward chaining*

# Forward chaining

- *Look for a clause with all body literals satisfied*
- *Add its head to KB*
- *Repeat*
- *See RN for more details*

# Handling uncertainty

- *Fuzzy logic / certainty factors*
  - *simple, but don't scale*
- *Nonmonotonic logic*
  - *also doesn't scale*
- *Probabilities*
  - *may or may not scale—more in Part II*

# Certainty factors

- *KB assigns a **certainty factor** in [0, 1] to each proposition*
- *Interpret as "degree of belief"*
- *When applying an inference rule, certainty factor for consequent is a function of certainty factors for antecedents (e.g., minimum)*

# Problems w/ certainty factors

- *Hard to separate a large KB into mostly-independent chunks that interact only through a well-defined interface*
- *Certainty factors are not probabilities (i.e., do not obey Bayes' Rule)*

# Nonmonotonic logic

- *Suppose we believe all birds can fly*
- *Might add a set of sentences to KB*

$$bird(Polly) \Rightarrow flies(Polly)$$

$$bird(Tweety) \Rightarrow flies(Tweety)$$

$$bird(Tux) \Rightarrow flies(Tux)$$

$$bird(John) \Rightarrow flies(John)$$

$\ldots$

# Nonmonotonic logic

- *Fails if there are penguins in the KB*
- *Fix: instead, add*

  $bird(Polly) \wedge \neg ab(Polly) \Rightarrow flies(Polly)$

  $bird(Tux) \wedge \neg ab(Tux) \Rightarrow flies(Tux)$

  …
- *ab(Tux) is an "abnormality predicate"*
- *Need separate $ab_i(x)$ for each type of rule*

# Nonmonotonic logic

- *Now set as few abnormality predicates as possible*
- *Can prove flies(Polly) or flies(Tux) with no ab(x) assumptions*
- *If we assert ¬flies(Tux), must now assume ab(Tux) to maintain consistency*
- *Can't prove flies(Tux) any more, but can still prove flies(Polly)*

# Nonmonotonic logic

- *Works well as long as we don't have to choose between big sets of abnormalities*
  - *is it better to have 3 flightless birds or 5 professors that don't wear jackets with elbow-patches?*
  - *even worse with nested abnormalities: birds fly, but penguins don't, but superhero penguins do, but …*

# SAT

# Definitions

- *A sentence is* ***satisfiable*** *if it is True in some model*
- *If not satisfiable, it is a* ***contradiction*** *(False in every model)*
- *A sentence is* ***valid*** *if it is True in every model (a valid sentence is a* ***tautology****)*

# 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*

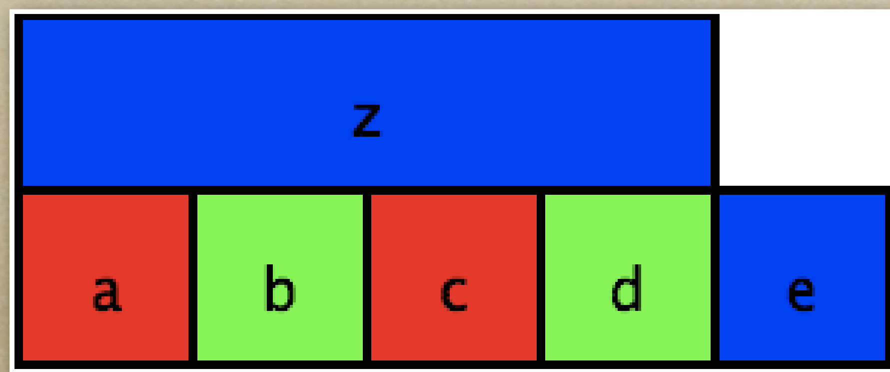
# SAT is a search problem

- *(At least) two ways to write it*
  - *search nodes are (full or partial) models, neighbors differ in assignment for a single variable*
  - *search nodes are formulas, neighbors by entailment*
- *And hybrids (node = model + formula)*

# SAT is a general search problem

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

# Example search problem

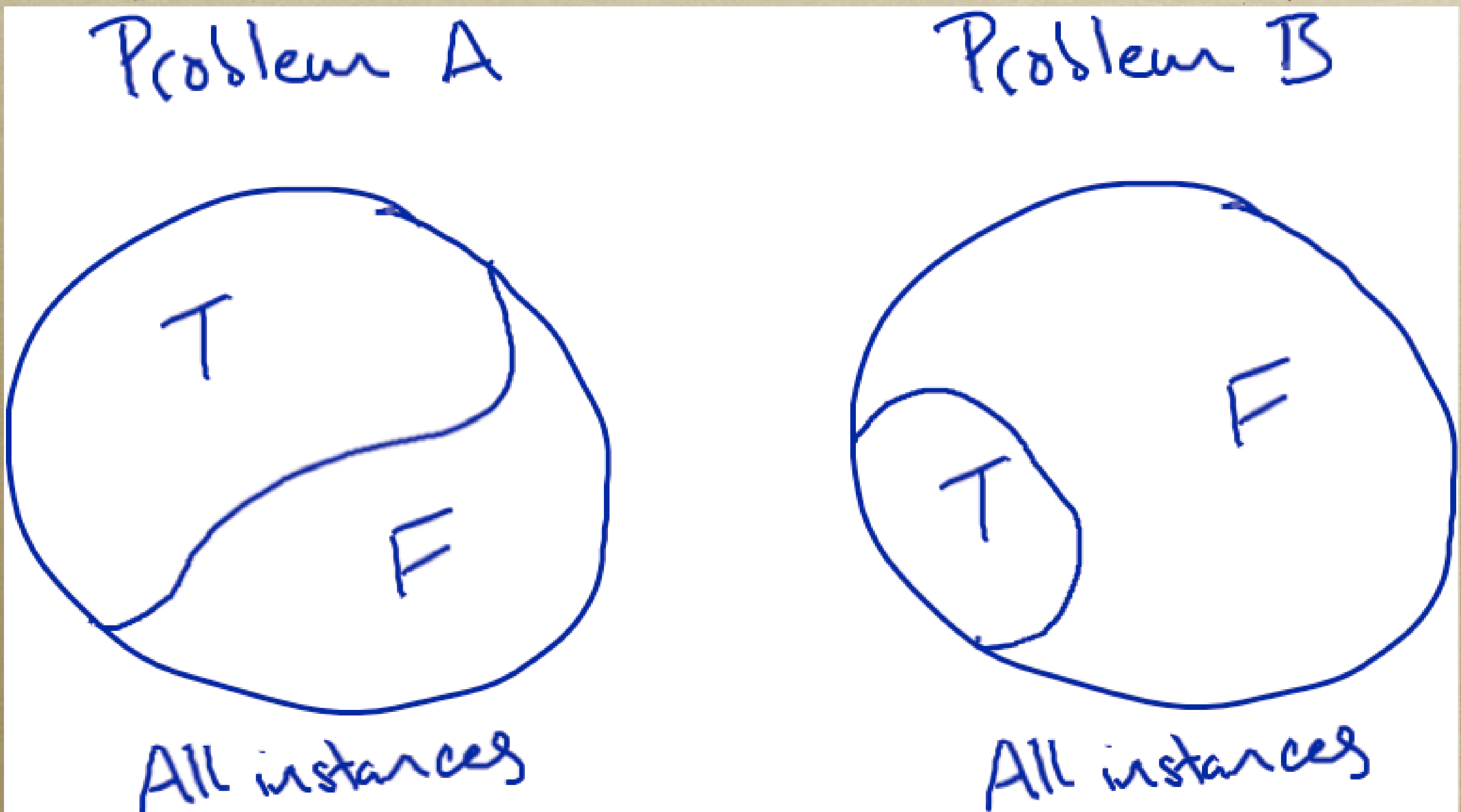



- *3-coloring: can we color a map using only 3 colors in a way that keeps neighboring regions from being the same color?*

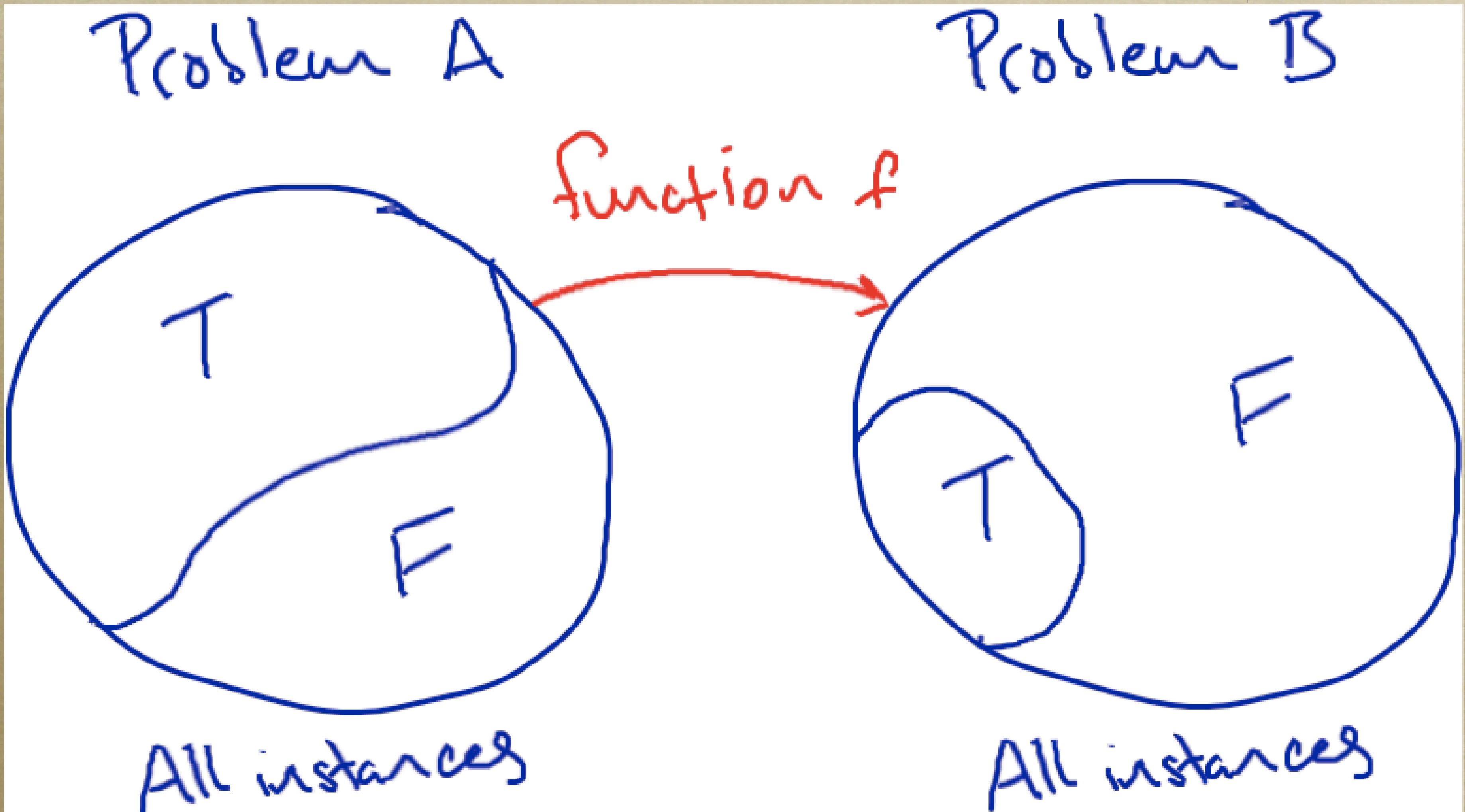
# Reduction

- *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*
  - *$A(a) = B(f(a))$*

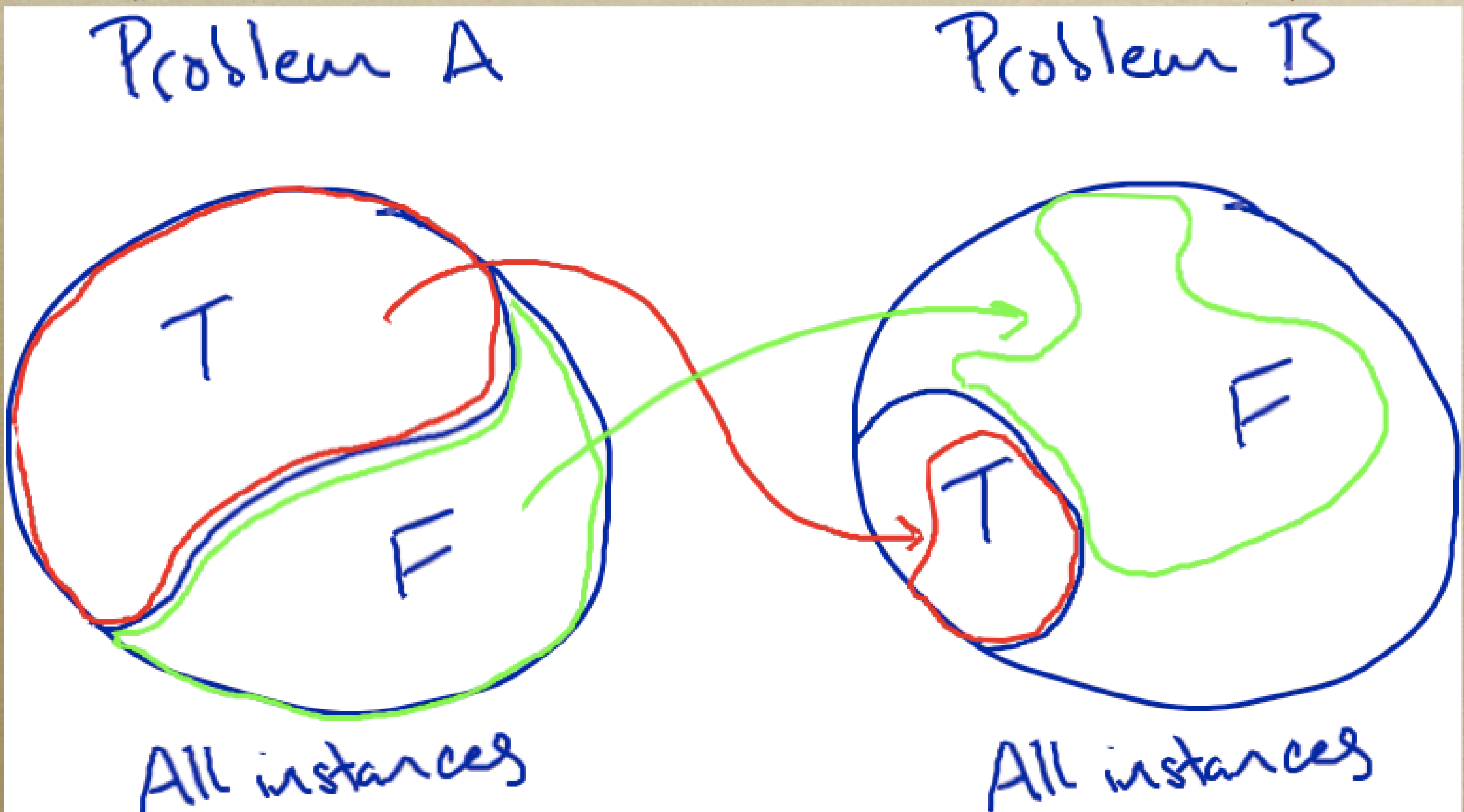
# Reduction picture



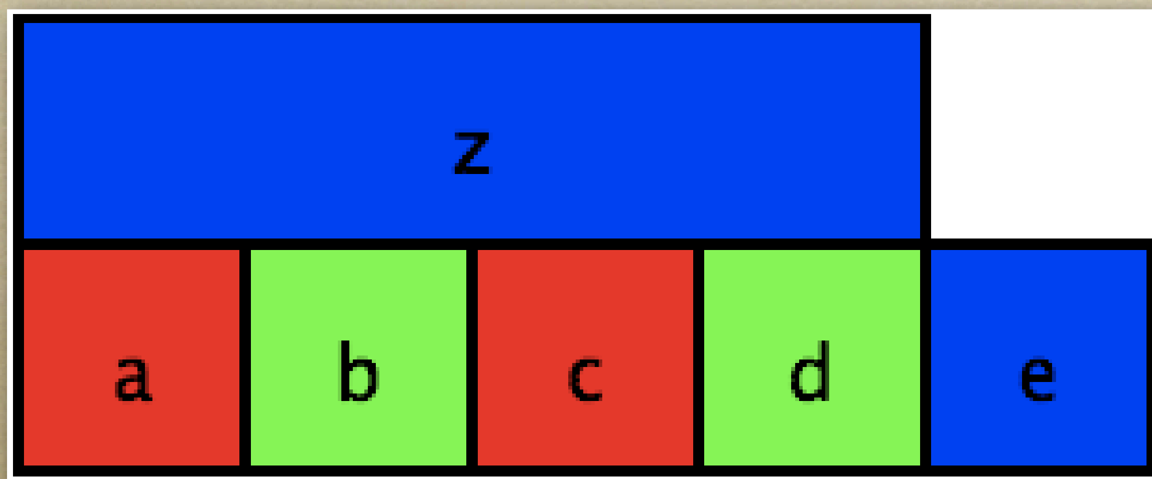
# Reduction picture



# Reduction picture



# Example reduction




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

# Example reduction

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

# Search and reduction

- *S. A. Cook in 1971 proved that many useful search 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 search 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., search on transformed instance may take seconds vs. days*

# Direction of reduction

- *If A reduces to B then*
  - *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*

# Reduction to 3SAT

- *We saw that search 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*

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