

15-780: Graduate AI

Lecture 2. Proofs & FOL

Geoff Gordon (this lecture)

Tuomas Sandholm

TAs Erik Zawadzki, Abe Othman

Admin

- *Recitations: Fri. 3PM here (GHC 4307)*
- *Vote: useful to have one tomorrow?* No
 - *would cover propositional & FO logic*
- *Draft schedule of due dates up on web*
 - *subject to change with notice*

Course email list



- *15780students AT cs.cmu.edu*
- *Everyone's official email should be in the list—we've sent a test message, so if you didn't get it, let us know*



Review

What is AI?

- *Lots of examples: poker, driving robots, flying birds, RoboCup*
- *Things that are easy for humans/animals to do, but no obvious algorithm*
- *Search / optimization / summation*
- *Handling uncertainty*
- *Sequential decisions*

Propositional logic

- *Syntax*
 - *variables, constants, operators*
 - *literals, clauses, sentences*
- *Semantics (model $\mapsto \{T, F\}$)*
- *Truth tables, how to evaluate formulas*
- *Satisfiable, valid, contradiction*
- *Relationship to CSPs*

Propositional logic

- *Manipulating formulas (e.g., de Morgan)*
- *Normal forms (e.g., CNF)*
- *Tseitin transformation to CNF*
- *Handling uncertainty (independent Nature choices + logical consequences)*
- *Compositional semantics*
- *How to translate informally-specified problems into logic (e.g., 3-coloring)*

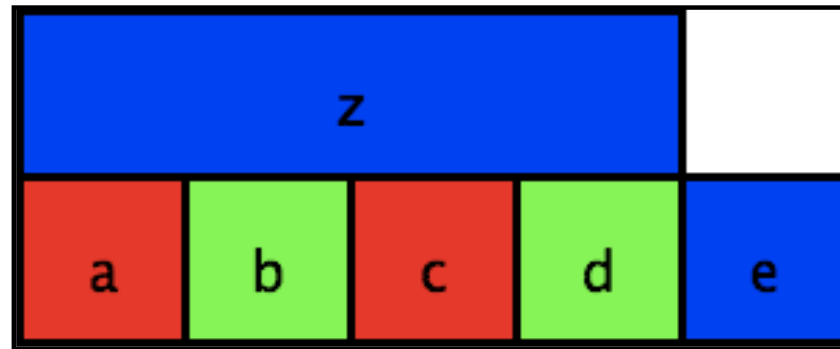


NP

Satisfiability

- *SAT: determine whether a propositional logic sentence has a satisfying model*
- *A **decision problem**: instance \rightarrow yes or no*
- *Fundamental problem in CS*
 - *many decision problems **reduce** to SAT*
 - *informally, if we can solve SAT, we can solve these other problems*
- *A SAT solver is a good AI building block*

Example decision problem



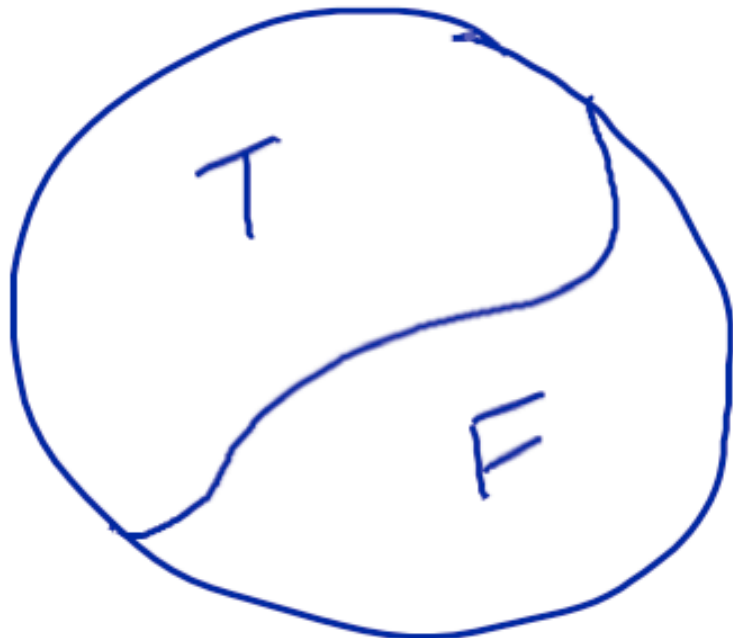
- *k-coloring: can we color a map using only k colors in a way that keeps neighboring regions from being the same color?*

Reduction

- *Loosely, “A reduces to B” means that if we can solve B then we can solve A*
- *Formally, let A, B be decision problems (instances \rightarrow Y or N)*
- *A reduction is a poly-time function f such that, given an instance a of A*
 - *$f(a)$ is an instance of B, and*
 - *$A(a) = B(f(a))$*

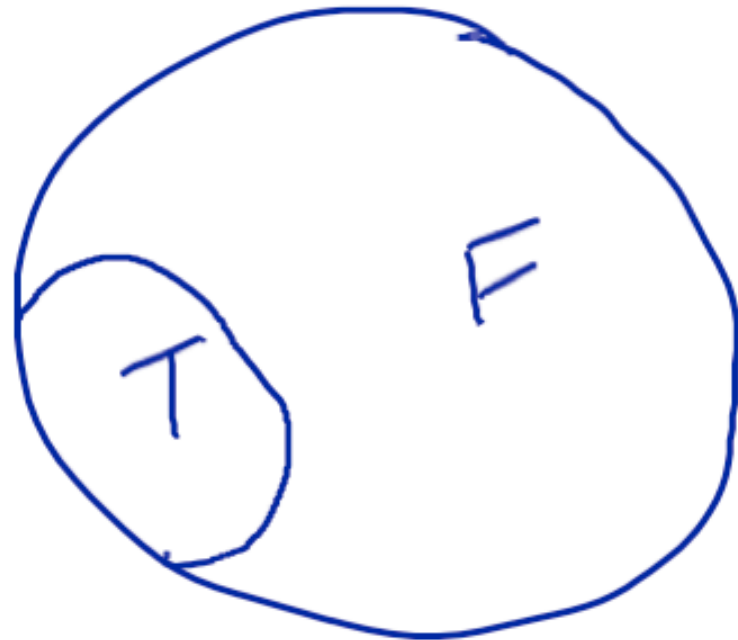
Reduction picture

Problem A



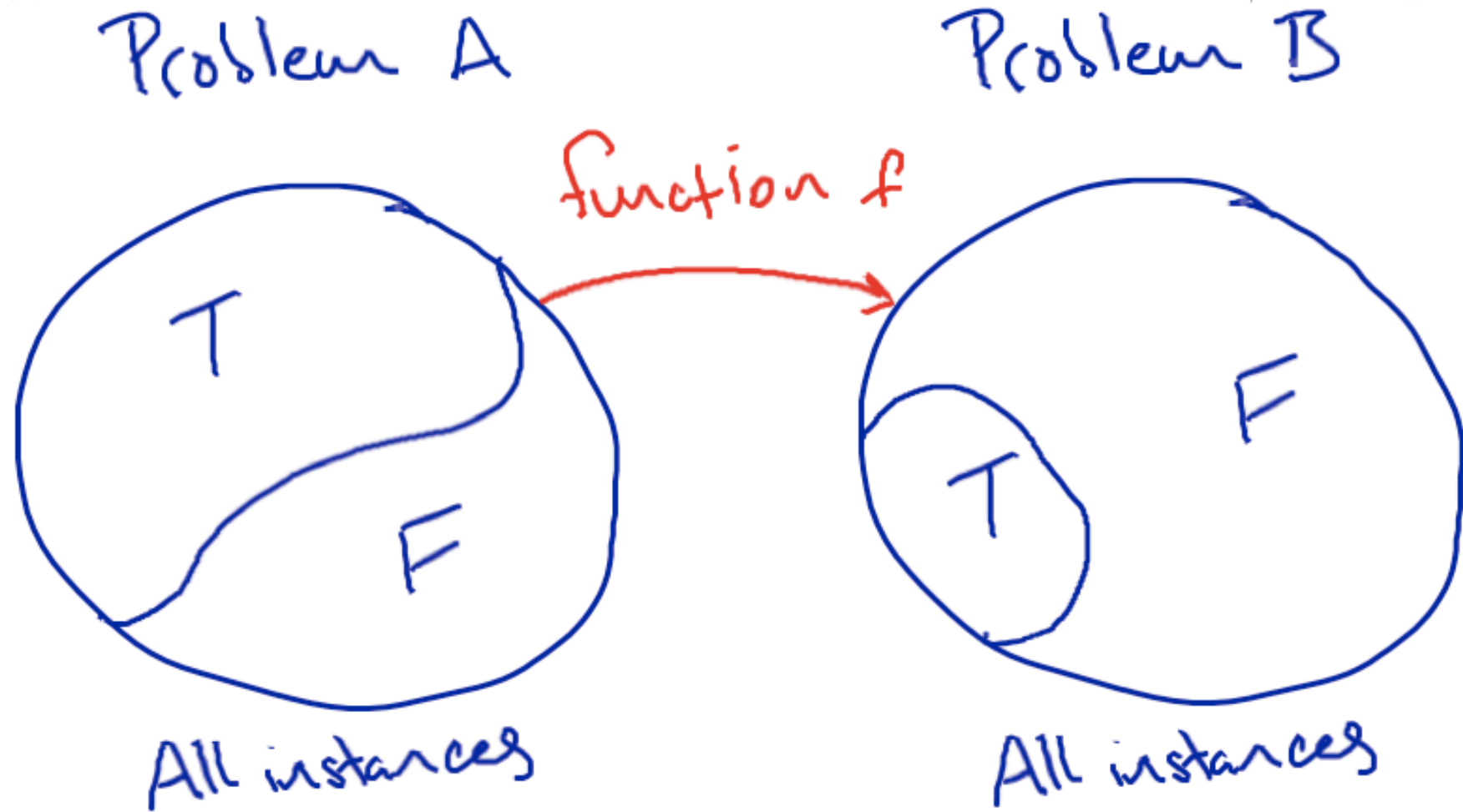
All instances

Problem B

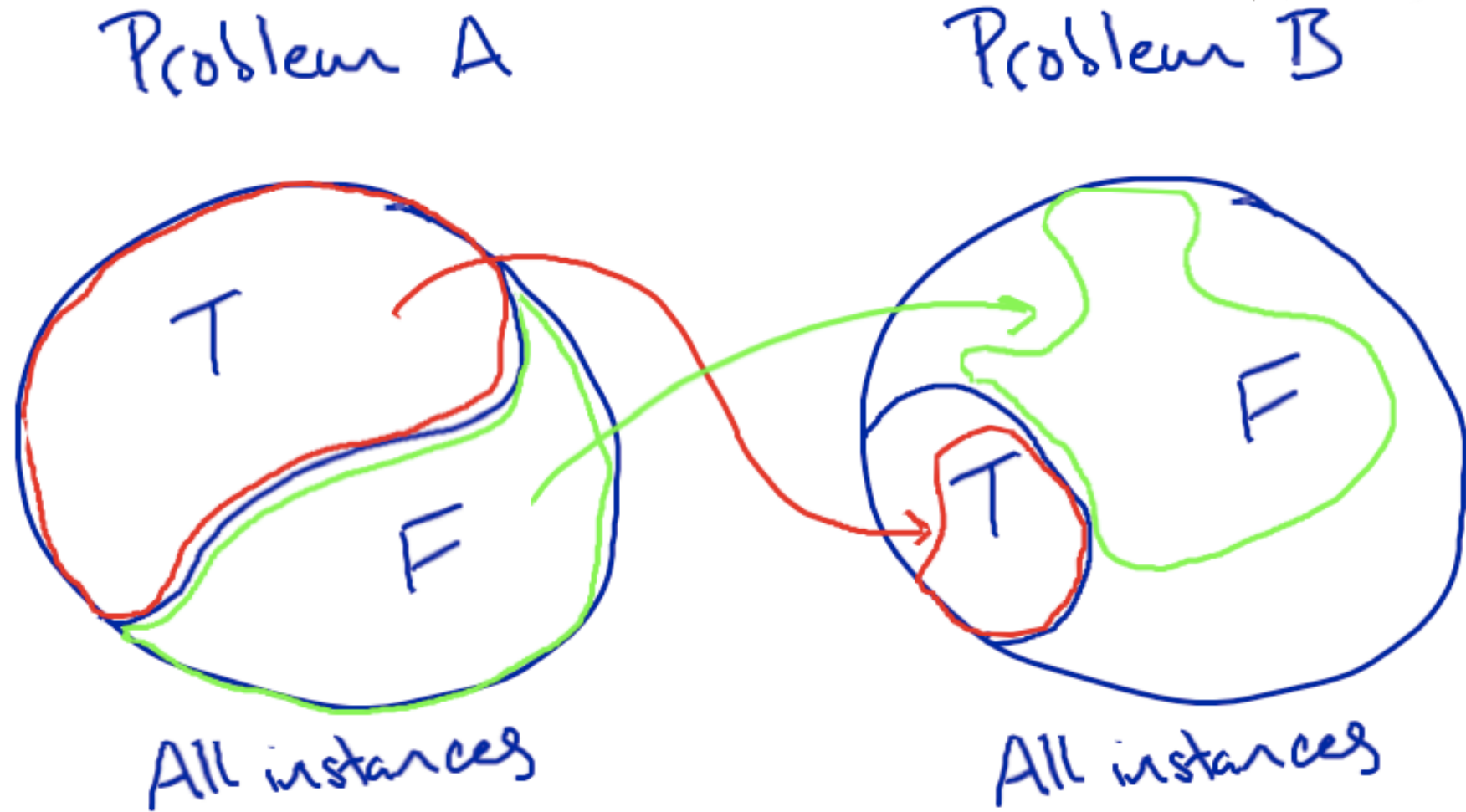


All instances

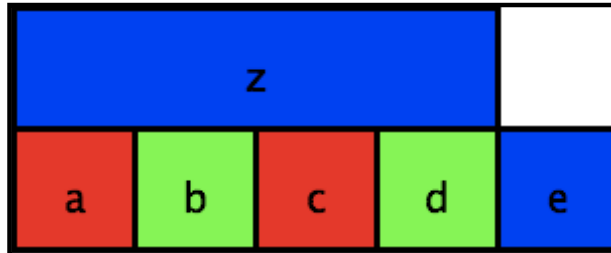
Reduction picture



Reduction picture



Reducing k-coloring \rightarrow SAT



$$(a_r \vee a_g \vee a_b) \wedge (b_r \vee b_g \vee b_b) \wedge (c_r \vee c_g \vee c_b) \wedge$$

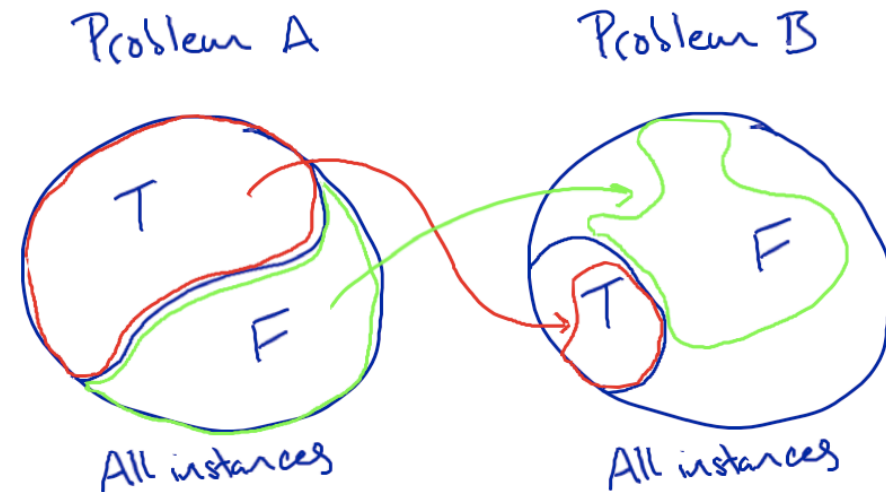
$$(d_r \vee d_g \vee d_b) \wedge (e_r \vee e_g \vee e_b) \wedge (z_r \vee z_g \vee z_b) \wedge$$

$$(\neg a_r \vee \neg b_r) \wedge (\neg a_g \vee \neg b_g) \wedge (\neg a_b \vee \neg b_b) \wedge$$

$$(\neg a_r \vee \neg z_r) \wedge (\neg a_g \vee \neg z_g) \wedge (\neg a_b \vee \neg z_b) \wedge$$

...

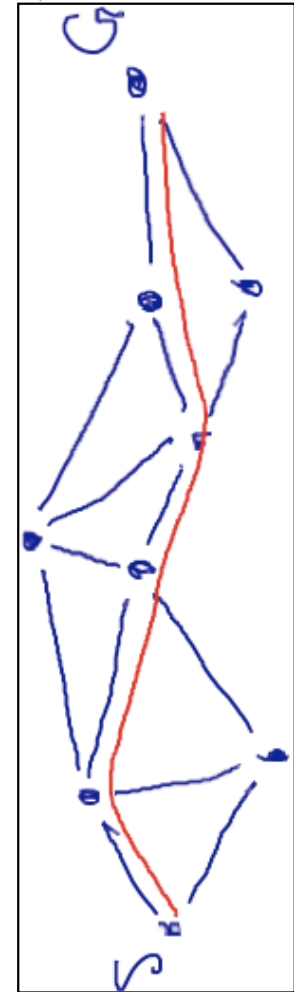
Direction of reduction



- *When 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: SAT \rightarrow CNF-SAT

- *Given any propositional formula, Tseitin transformation produces (in poly time) an equivalent CNF formula*
- *So, given a CNF-SAT solver, we can solve SAT with general formulas*

More useful: CNF-SAT \rightarrow 3SAT

- *Can reduce even further, to 3SAT*
 - *is 3CNF formula satisfiable?*
 - *3CNF: at most 3 literals per clause*
- *Useful if reducing SAT/3SAT to another problem (to show other problem hard)*

CNF-SAT \rightarrow 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

- *A decision problem is in NP if it reduces to SAT*
- *E.g., TSP, k-coloring, propositional planning, integer programming (decision versions)*
- *E.g., path planning, solving linear equations*

NP-complete

- *Many decision problems reduce back and forth to SAT: they are **NP-complete***
 - *Cook showed how to simulate any poly-time nondeterministic computation w/ (very complicated, but still poly-size) SAT problem*
- *Equivalently, SAT is exactly as hard (in theory at least) as these other problems*

S. A. Cook. The complexity of theorem-proving procedures, Proceedings of ACM STOC'71, pp. 151-158, 1971.

Open question: $P = NP$

- $P =$ there is a poly-time algorithm to solve
- $NP =$ reduces to SAT
- We know of no poly-time algorithm for SAT, but we also can't prove that SAT requires more than about linear time!

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 \Rightarrow 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*



Proofs

Entailment

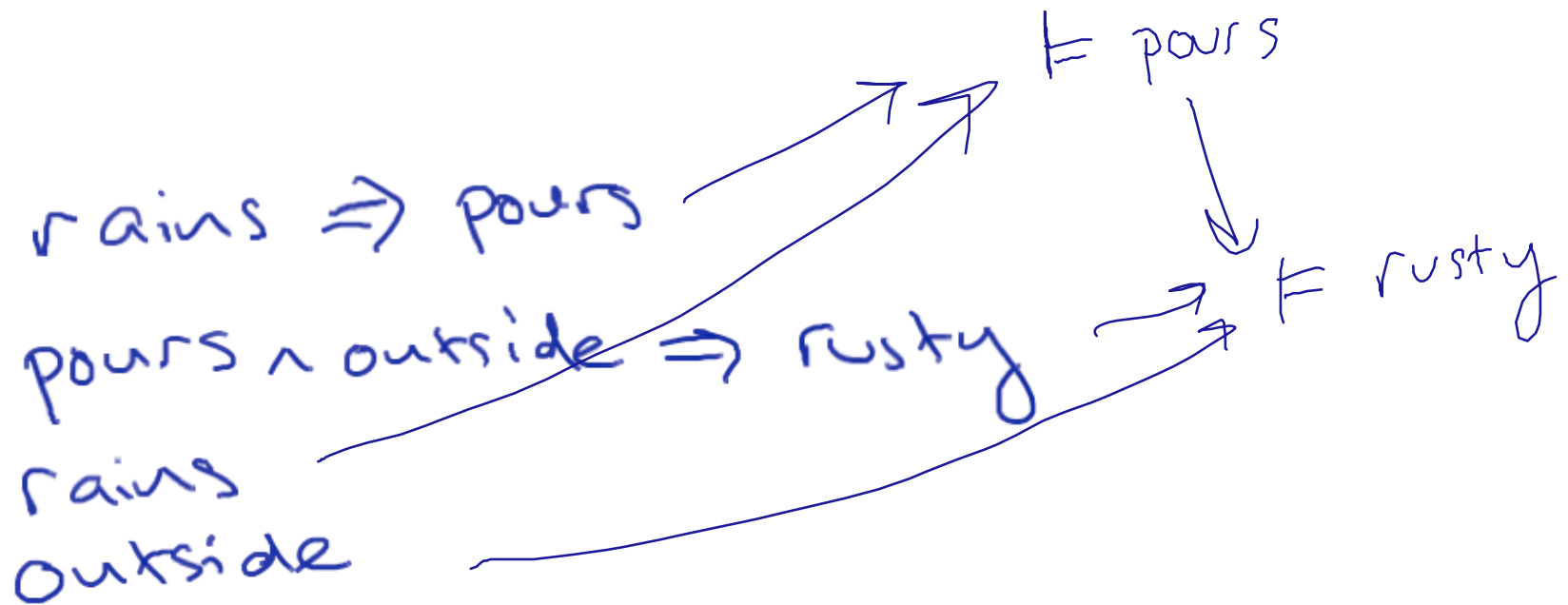
	A	B
TT	F	F
TF	T	T
FT	F	T
FF	T	T

- 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 \models parent*
- *Leaves: assumptions or premises*
- *Root: consequence*
- *If we believe assumptions, we should also believe consequence*

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 $(KB \wedge \neg S)$*
- *Build a proof tree with*
 - *assumptions drawn from clauses of KB'*
 - *conclusion = F*
 - *so, $(KB \wedge \neg S) \models F$ (contradiction)*

Proof by contradiction

KB

$\text{rains} \Rightarrow \text{pours}$

$\text{pours} \wedge \text{outside} \Rightarrow \text{rusty}$

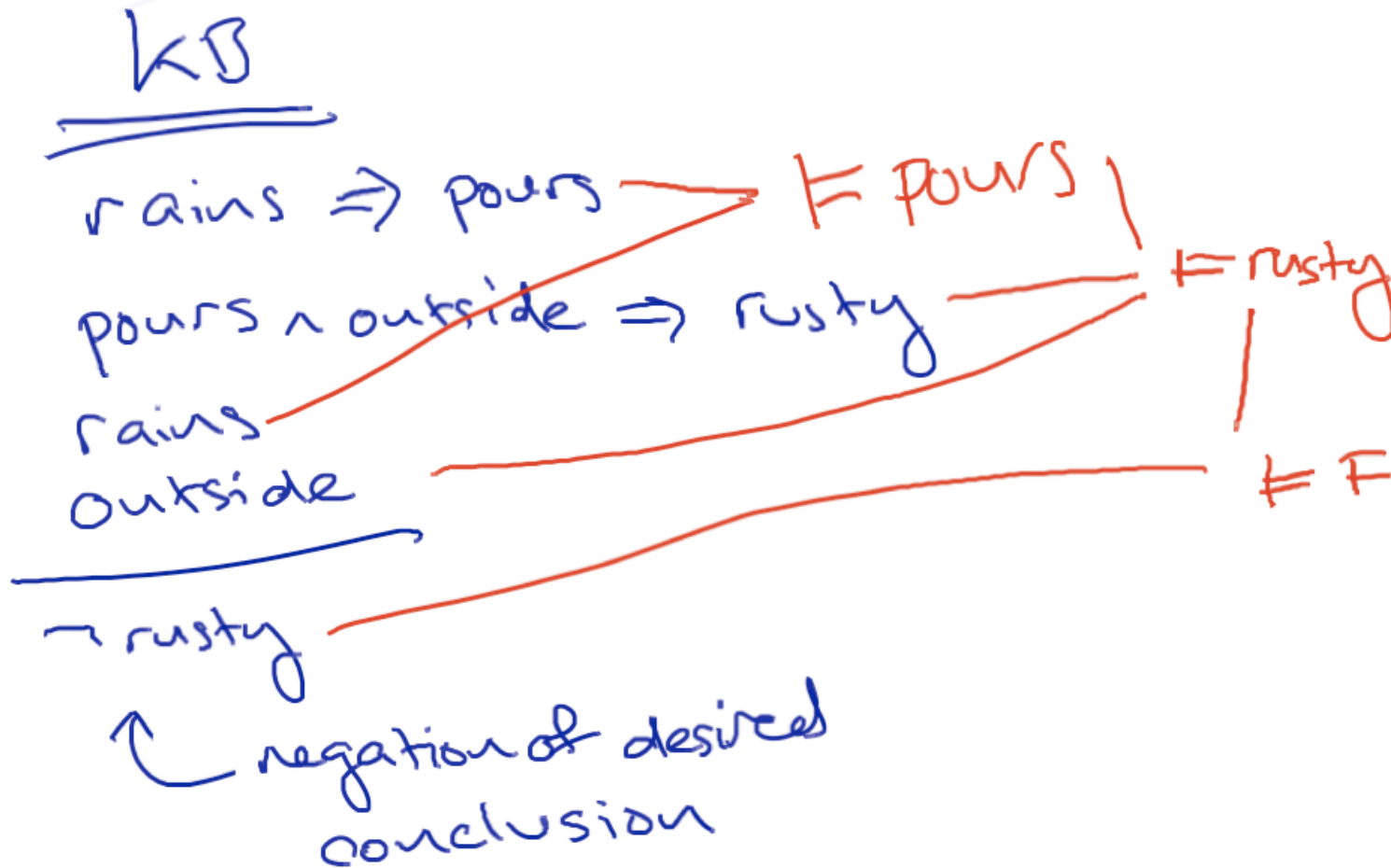
rains

outside

$\neg \text{rusty}$

↖ negation of desired
conclusion

Proof by contradiction





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...

$$\frac{(\alpha \vee c) \quad (\neg c \vee \beta)}{\alpha \vee \beta}$$

- ***Resolution***
 - *α, β are arbitrary subformulas*
 - *Combines two formulas 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:*
 $(\neg \textit{raining} \vee \textit{grass-wet}) \wedge \neg \textit{grass-wet} \models \neg \textit{raining}$

Resolution example

- $rains \Rightarrow pours$
- $pours \wedge outside \Rightarrow rusty$
- *Can we conclude $rains \wedge outside \Rightarrow rusty$?*

Resolution example

- $rains \Rightarrow pours$
- $pours \wedge outside \Rightarrow rusty$
- *Can we conclude $rains \wedge outside \Rightarrow rusty$?*

$\neg rains \vee pours$

$\neg pours \vee \neg outside \vee rusty$

Resolution example

- $rains \Rightarrow pours$
- $pours \wedge outside \Rightarrow rusty$
- *Can we conclude $rains \wedge outside \Rightarrow rusty$?*

$$\frac{\neg rains \vee pours \quad \neg pours \vee \neg outside \vee rusty}{\neg rains \vee \neg outside \vee rusty}$$

Resolution

$$\frac{(\alpha \vee c) \quad (\neg c \vee \beta)}{\alpha \vee \beta}$$

- *Simple proof by case analysis*
- *Consider separately cases where we assign $c = \text{True}$ and $c = \text{False}$*

Resolution case analysis

$$(\alpha \vee c) \wedge (\neg c \vee \beta)$$

$$c = T:$$

$$T \wedge \beta = \beta$$

$$c = F:$$

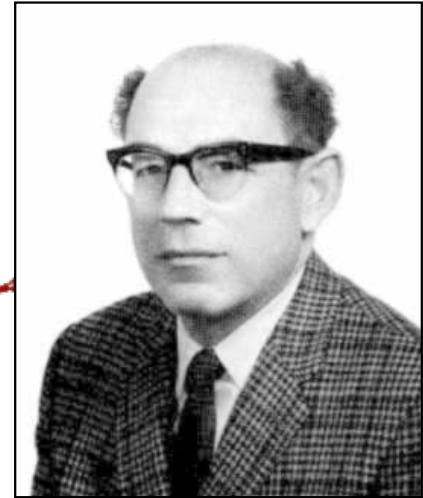
$$\alpha \wedge T = \alpha$$

$$\alpha \vee \beta$$

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 procedure is **complete** if it can conclude everything entailed by KB*

Completeness



J. A. Robinson
1918–1974

- *Modus ponens by itself is incomplete*
- *Resolution + proof by contradiction is complete for propositional formulas represented as sets of clauses*
 - *famous theorem due to Robinson*
 - *if $KB \models F$, we'll derive empty clause*
- *Caveat: also need factoring, removal of redundant literals $(a \vee b \vee a) \models (a \vee b)$*

Algorithms

- *We now have our first* algorithm for SAT*
 - *remove redundant literals (factor) wherever possible*
 - *pick an application of resolution according to some fair rule*
 - *add its consequence to KB*
 - *repeat*
- *Not a great algorithm, but works*

Variations



- *Horn clause inference*
- *MAXSAT*
- *Nonmonotonic logic*

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)*

$$\text{happy}(\text{John}) \wedge \text{happy}(\text{Mary}) \Rightarrow \text{happy}(\text{Sue})$$

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

Why are Horn clauses important

- *Modus ponens alone is complete*
- *So is modus tollens alone*
- *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 (modus ponens)*
- *Repeat*
- *See RN for more details*

MAXSAT

- *Given a CNF formula $C_1 \wedge C_2 \wedge \dots \wedge C_n$*
- *Clause weights w_1, w_2, \dots, w_n (weighted version) or $w_i = 1$ (unweighted)*
- *Find model which satisfies clauses of maximum total weight*
 - *decision version: max weight $\geq w$?*
- *More generally, weights on variables (bonus for setting to T): MAXVARSAT*

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)

...

Nonmonotonic logic

- *Fails if there are penguins in the KB*

- *Fix: instead, add*

$$\text{bird}(\text{Polly}) \wedge \neg \text{ab}(\text{Polly}) \Rightarrow \text{flies}(\text{Polly})$$

$$\text{bird}(\text{Tux}) \wedge \neg \text{ab}(\text{Tux}) \Rightarrow \text{flies}(\text{Tux})$$

...

- *$\text{ab}(\text{Tux})$ is an “abnormality predicate”*
- *Need separate $\text{ab}_i(x)$ for each type of rule*

Nonmonotonic logic

- *Now set as few abnormality predicates as possible (a MAXVARSAT problem)*
- *Can prove flies(Polly) or flies(Tux) with no $ab(x)$ assumptions*
- *If we assert $\neg \text{flies(Tux)}$, must now assume $ab(\text{Tux})$ to maintain consistency*
- *Can't prove flies(Tux) any more, but can still prove flies(Polly)*

Nonmonotonic logic

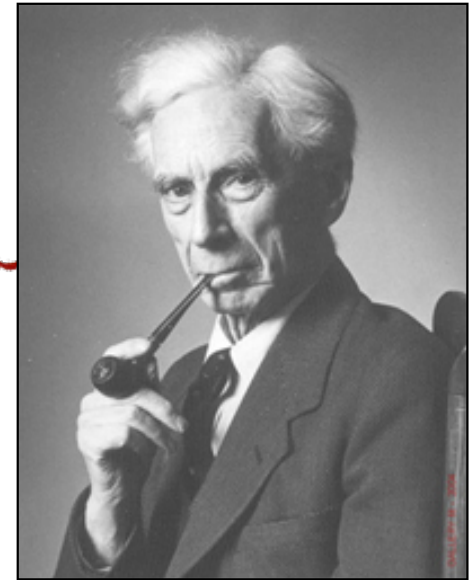
- *Works well as long as we don't have to choose between big sets of abnormalities*
 - *is it better to have 3 flightless birds or 5 professors that don't wear jackets with elbow-patches?*
 - *even worse with nested abnormalities: birds fly, but penguins don't, but superhero penguins do, but ...*



First-order logic

First-order logic

Bertrand Russell
1872-1970



- *So far we've been using opaque vars like **rains** or **happy(John)***
- *Limits us to statements like “it's raining” or “if John is happy then Mary is happy”*
- *Can't say “all men are mortal” or “if John is happy then someone else is happy too”*

Predicates and objects

- *Interpret happy(John) or likes(Joe, pizza) as a **predicate** applied to some **objects***
- *Object = an object in the world*
- *Predicate = boolean-valued function of objects*
- *Zero-argument predicate $x()$ plays same role that Boolean variable x did before*

Distinguished predicates

- *We will assume three distinguished predicates with fixed meanings:*
 - *True / T, False / F*
 - *Equal(x, y)*
- *We will also write $(x = y)$ and $(x \neq y)$*

Equality satisfies usual axioms

- *Reflexive, transitive, symmetric*
- *Substituting equal objects doesn't change value of expression*

$(John = Jonathan) \wedge loves(Mary, John)$
 $\Rightarrow loves(Mary, Jonathan)$

Functions



- *Functions map zero or more objects to another object*
 - *e.g., professor(15-780), last-common-ancestor(John, Mary)*
- *Zero-argument function is the same as an object—John v. John()*

The **nil** object



- *Functions are untyped: must have a value for **any** set of arguments*
- *Typically add a **nil** object to use as value when other answers don't make sense*

Types of values

- *Expressions in propositional logic could only have Boolean (T/F) values*
- *Now we have two types of expressions: object-valued and Boolean-valued*
 - *$done(slides(15-780)) \Rightarrow happy(professor(15-780))$*
- *Functions map objects to objects; predicates map objects to Booleans; connectives map Booleans to Booleans*

Definitions

- *Term = expression referring to an object*
 - *John*
 - *left-leg-of(father-of(president-of(USA)))*
- *Atom = predicate applied to objects*
 - *happy(John)*
 - *raining*
 - *at(robot, Wean-5409, 11AM-Wed)*

Definitions

- *Literal* = possibly-negated atom
 - *happy(John), \neg happy(John)*
- *Sentence or formula* = literals joined by connectives like $\wedge \vee \neg \Rightarrow$
 - *raining*
 - *done(slides(780)) \Rightarrow happy(professor)*
- *Expression* = term or formula

Semantics

- *Models are now much more complicated*
 - *List of objects (nonempty, may be infinite)*
 - *Lookup table for each function mentioned*
 - *Lookup table for each predicate mentioned*
- *Meaning of sentence: model \rightarrow $\{T, F\}$*
- *Meaning of term: model \rightarrow object*

For example



KB describing example

- $alive(cat)$
- $ear-of(cat) = ear$
- $in(cat, box) \wedge in(ear, box)$
- $\neg in(box, cat) \wedge \neg in(cat, nil) \dots$
- $ear-of(box) = ear-of(ear) = ear-of(nil) = nil$
- $cat \neq box \wedge cat \neq ear \wedge cat \neq nil \dots$

Aside: avoiding verbosity

- *Closed-world assumption: literals not assigned a value in KB are false*
 - *avoid stating $\neg in(box, cat)$, etc.*
- *Unique names assumption: objects with separate names are separate*
 - *avoid $box \neq cat$, $cat \neq ear$, ...*

Aside: typed variables

- *KB also illustrates need for **data types***
- *Don't want to have to specify $ear-of(box)$ or $\neg in(cat, nil)$*
- *Could design a type system*
 - *argument of $happy()$ is of type **animate***
- *Include rules saying function instances which disobey type rules have value **nil***

Model of example

- *Objects: C, B, E, N*
- *Function values:*
 - *cat: C, box: B, ear: E, nil: N*
 - *ear-of(C): E, ear-of(B): N, ear-of(E): N, ear-of(N): N*
- *Predicate values:*
 - *in(C, B), \neg in(C, C), \neg in(C, N), ...*

Failed model



- *Objects: C, E, N*
- *Fails because there's no way to satisfy inequality constraints with only 3 objects*

Another possible model

- *Objects: C, B, E, N, X*
- *Extra object X could have arbitrary properties since it's not mentioned in KB*
- *E.g., X could be its own ear*

An embarrassment of models



- *In general, can be infinitely many models*
 - *unless KB limits number somehow*
- *Job of KB is to rule out models that don't match our idea of the world*
- *Saw how to rule out CEN model*
- *Can we rule out CBENX model?*

Getting rid of extra objects

- *Can use quantifiers to rule out CBENX model:*

$$\forall x. x = cat \vee x = box \vee x = ear \vee x = nil$$

- *Called a domain closure assumption*

Quantifiers, informally

- *Add quantifiers and object variables*
 - $\forall x. \text{man}(x) \Rightarrow \text{mortal}(x)$
 - $\neg \exists x. \text{lunch}(x) \wedge \text{free}(x)$
- \forall : *no matter how we replace object variables with objects, formula is still true*
- \exists : *there is some way to fill in object variables to make formula true*

New syntax

- *Object variables are terms*
- *Build atoms from variables x, y, \dots as well as constants John, Fred, ...*
 - *$man(x), loves(John, z), mortal(brother(y))$*
- *Build formulas from these atoms*
 - *$man(x) \Rightarrow mortal(brother(x))$*
- *New syntactic construct: term or formula w/ free variables*

New syntax \Rightarrow new semantics

- *Variable assignment for a model M maps syntactic variables to model objects*
 - $x: C, y: N$
- *Meaning of expression w/ free vars: look up in assignment, then continue as before*
 - $term: (model, var\ asst) \rightarrow object$
 - $formula: (model, var\ asst) \rightarrow truth\ value$

Example

- *Model: CEBN model from above*
- *Assignment: $(x: C, y: N)$*
- *$alive(ear(x)) \mapsto alive(ear(C)) \mapsto alive(E) \mapsto T$*

Working with assignments

- Write ε for an arbitrary assignment (e.g., all variables map to nil)
- Write $(V / x: \text{obj})$ for the assignment which is just like V except that variable x maps to object obj

More new syntax: Quantifiers, binding

- *For any variable x and formula F , $(\forall x. F)$ and $(\exists x. F)$ are formulas*
- *Adding quantifier for x is called **binding** x*
 - *In $(\forall x. \text{likes}(x, y))$, x is bound, y is free*
- *Can add quantifiers and apply logical operations like $\wedge \vee \neg$ in any order*
- *But must eventually wind up with **ground** formula (no free variables)*

Semantics of \forall

- *Sentence $(\forall x. S)$ is T in (M, V) if S is T in $(M, V / x: obj)$ for all objects obj in M*

Example

- *M* has objects (A, B, C) and predicate $happy(x)$ which is true for A, B, C
- Sentence $\forall x. happy(x)$ is satisfied in (M, ε)
 - since $happy(A), happy(B), happy(C)$ are all satisfied in M
 - more precisely, $happy(x)$ is satisfied in $(M, \varepsilon/x:A), (M, \varepsilon/x:B), (M, \varepsilon/x:C)$

Semantics of \exists

- *Sentence $(\exists x. S)$ is true in (M, V) if there is some object obj in M such that S is true in $(M, V / x: obj)$*

Example

- *M has objects (A, B, C) and predicate*
 - *happy(A) = happy(B) = True*
 - *happy(C) = False*
- *Sentence $\exists x. \text{happy}(x)$ is satisfied in (M, ε)*
- *Since $\text{happy}(x)$ is satisfied in $(M, \varepsilon/x:B)$*

Scoping rules (so we don't have to write a gazillion parens)

- *In $(\forall x. F)$ and $(\exists x. F)$, $F = \text{scope} = \text{part of formula where quantifier applies}$*
- *Variable x is bound by **innermost** possible quantifier (matching name, in scope)*
- *Two variables in different scopes can have same name—they are still different vars*
- *Quantification has lowest precedence*

Scoping examples

- $(\forall x. \text{happy}(x)) \vee (\exists x. \neg \text{happy}(x))$
 - *Either everyone's happy, or someone's unhappy*
- $\forall x. (\text{raining} \wedge \text{outside}(x) \Rightarrow (\exists x. \text{wet}(x)))$
 - *The x who is outside may not be the one who is wet*

Scoping examples

- *English sentence “everybody loves somebody” is ambiguous*
- *Translates to logical sentences*
 - $\forall x. \exists y. \text{loves}(x, y)$
 - $\exists y. \forall x. \text{loves}(x, y)$



Equivalence in FOL

Entailment, etc.

- *As before, entailment, satisfiability, validity, equivalence, etc. refer to all possible models*
 - *these words only apply to ground sentences, so variable assignment doesn't matter*
- *But now, can't determine by enumerating models, since there could be infinitely many*
- *So, must do reasoning via equivalences or entailments*

Equivalences

- *All transformation rules for propositional logic still hold*
- *In addition, there is a “De Morgan’s Law” for moving negations through quantifiers*

$$\neg \forall x. S \equiv \exists x. \neg S$$

$$\neg \exists x. S \equiv \forall x. \neg S$$

- *And, rules for getting rid of quantifiers*

Generalizing CNF

- *Eliminate \Rightarrow , move \neg in w/ De Morgan*
 - *but \neg moves through quantifiers too*
- *Get rid of quantifiers (see below)*
- *Distribute $\wedge \vee$, or use Tseitin*

Do we really need \exists ?

- $\exists x. \textit{happy}(x)$
- $\textit{happy}(\textit{happy_person}())$

- $\forall y. \exists x. \textit{loves}(y, x)$
- $\forall y. \textit{loves}(y, \textit{loved_one}(y))$

Skolemization



Thoralf Albert Skolem
1887–1963

- *Called Skolemization (after Thoralf Albert Skolem)*
- *Eliminate \exists by substituting a function of arguments of all enclosing \forall quantifiers*
- *Make sure to use a new name!*

Do we really need \forall ?

- *Positions of quantifiers irrelevant (as long as variable names are distinct)*
 - $\forall x. \text{happy}(x) \wedge \forall y. \text{takes}(y, \text{CS780})$
 - $\forall x. \forall y. \text{happy}(x) \wedge \text{takes}(y, \text{CS780})$
- *So, might as well drop them*
 - $\text{happy}(x) \wedge \text{takes}(y, \text{CS780})$

Getting rid of quantifiers

- *Standardize apart (avoid name collisions)*
- *Skolemize*
- *Drop \forall (free variables implicitly universally quantified)*
- *Terminology: still called “free” even though quantification is implicit*

For example



- $\forall x. \text{man}(x) \Rightarrow \text{mortal}(x)$
 - $\neg \text{man}(x) \vee \text{mortal}(x)$
- $\forall y. \exists x. \text{loves}(y, x)$
 - $\text{loves}(y, f(y))$
- $\forall x. \text{honest}(x) \Rightarrow \text{happy}(\text{Diogenes})$
 - $\neg \text{honest}(x) \vee \text{happy}(\text{Diogenes})$
- $(\forall x. \text{honest}(x)) \Rightarrow \text{happy}(\text{Diogenes})$


Exercise

◦ $(\forall x. \text{honest}(x)) \Rightarrow \text{happy}(\text{Diogenes})$

$\neg (\forall x. \text{honest}(x)) \vee \text{happy}(D)$

$(\exists x. \neg \text{honest}(x)) \vee \text{happy}(D)$

$\neg \text{honest}(\text{foo}()) \vee \text{happy}(D)$



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 w/ equality*
- *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 $\neg S$ 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*:

$(\neg \text{man}(x) \vee \text{mortal}(x)) \wedge \text{man}(\text{Socrates})$

$\models (\neg \text{man}(\text{Socrates}) \vee \text{mortal}(\text{Socrates}))$

$\wedge \text{man}(\text{Socrates})$

$\models \text{mortal}(\text{Socrates})$

◦ *Difference*: had to substitute $x \rightarrow \text{Socrates}$

Universal instantiation

- *What we just did is UI:*

$$(\neg \text{man}(x) \vee \text{mortal}(x))$$

$$\models (\neg \text{man}(\text{Socrates}) \vee \text{mortal}(\text{Socrates}))$$

- *Works for $x \rightarrow$ any term not containing x*

$$\dots \models (\neg \text{man}(\text{uncle}(y)) \vee \text{mortal}(\text{uncle}(y)))$$

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

Substitution lists

- *List of variable \rightarrow term 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., $L = (x \rightarrow \text{Socrates}, y \rightarrow \text{uncle}(z))$*

Substitution lists

- *Apply a substitution to an expression:
syntactically substitute vars \rightarrow terms*
- *E.g., $L = (x \rightarrow Socrates, y \rightarrow uncle(z))$*
 - *$mortal(x) \wedge man(y): L \rightarrow$
 $mortal(Socrates) \wedge man(uncle(z))$*
- *Substitution list \neq variable assignment*

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 $x \rightarrow Socrates$*
- *Importance: purely syntactic criterion for identifying useful substitutions*

Unification examples

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

Quiz

- *Can we unify*

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

- *What about*

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

Quiz

- *Can we unify*

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

No!

- *What about*

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

$x \rightarrow \text{Mary}, y \rightarrow \text{John}$

Standardize apart

- *But $\text{knows}(x, \text{Mary})$ is logically equivalent to $\text{knows}(y, \text{Mary})$!*
- *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.