15-780: Graduate AI

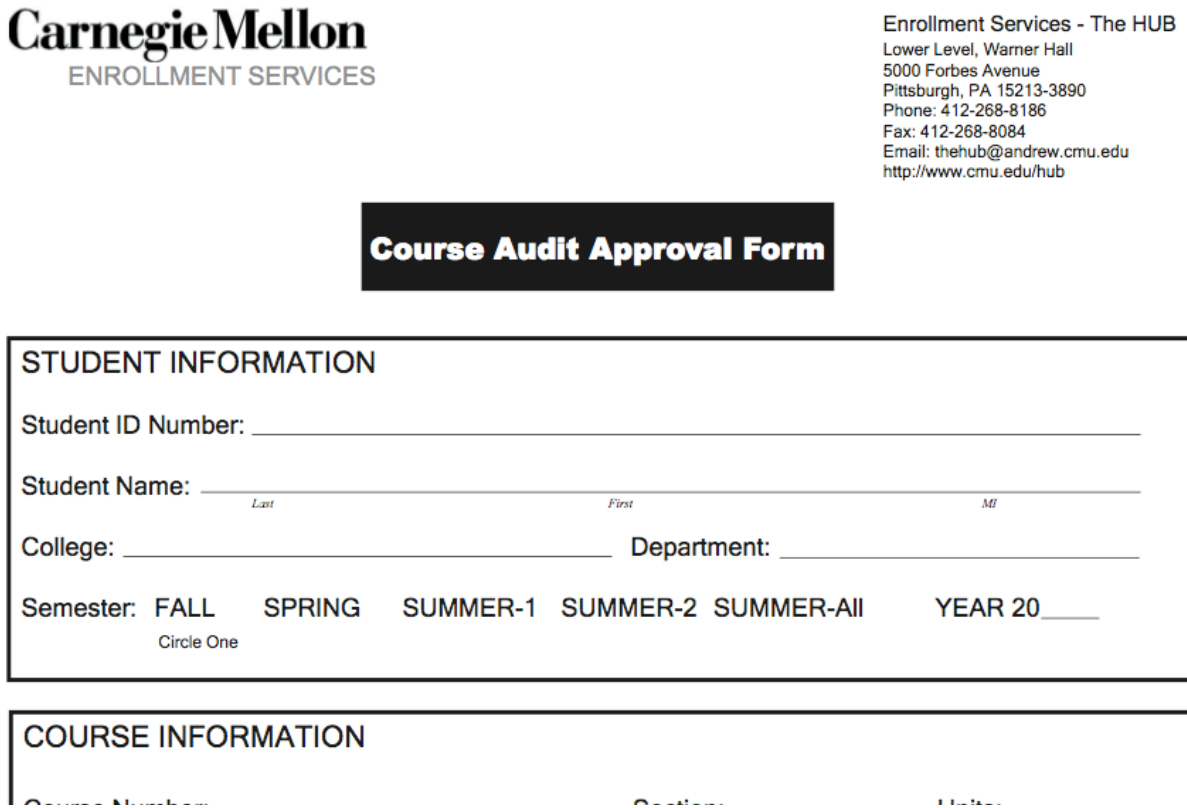
Lecture 2. Proofs & FOL

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

Admin

Audits

- *http://www.cmu.edu/hub/forms/ESG-AUDIT.pdf*



Carnegie Mellon
ENROLLMENT SERVICES

Enrollment Services - The HUB
Lower Level, Warner Hall
5000 Forbes Avenue
Pittsburgh, PA 15213-3890
Phone: 412-268-8186
Fax: 412-268-8084
Email: thehub@andrew.cmu.edu
http://www.cmu.edu/hub

Course Audit Approval Form

STUDENT INFORMATION

Student ID Number: _____

Student Name: _____
Last *First* *MI*

College: _____ Department: _____

Semester: FALL SPRING SUMMER-1 SUMMER-2 SUMMER-All YEAR 20____
Circle One

COURSE INFORMATION

Review

What is AI?

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

Propositional logic

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

Propositional logic

- *Manipulating formulas (e.g., de Morgan)*
- *Normal forms (e.g., CNF)*
- *Tseitin transformation to CNF*
- *How to translate informally-specified problems into logic (e.g., 3-coloring)*

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* $\models$ *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

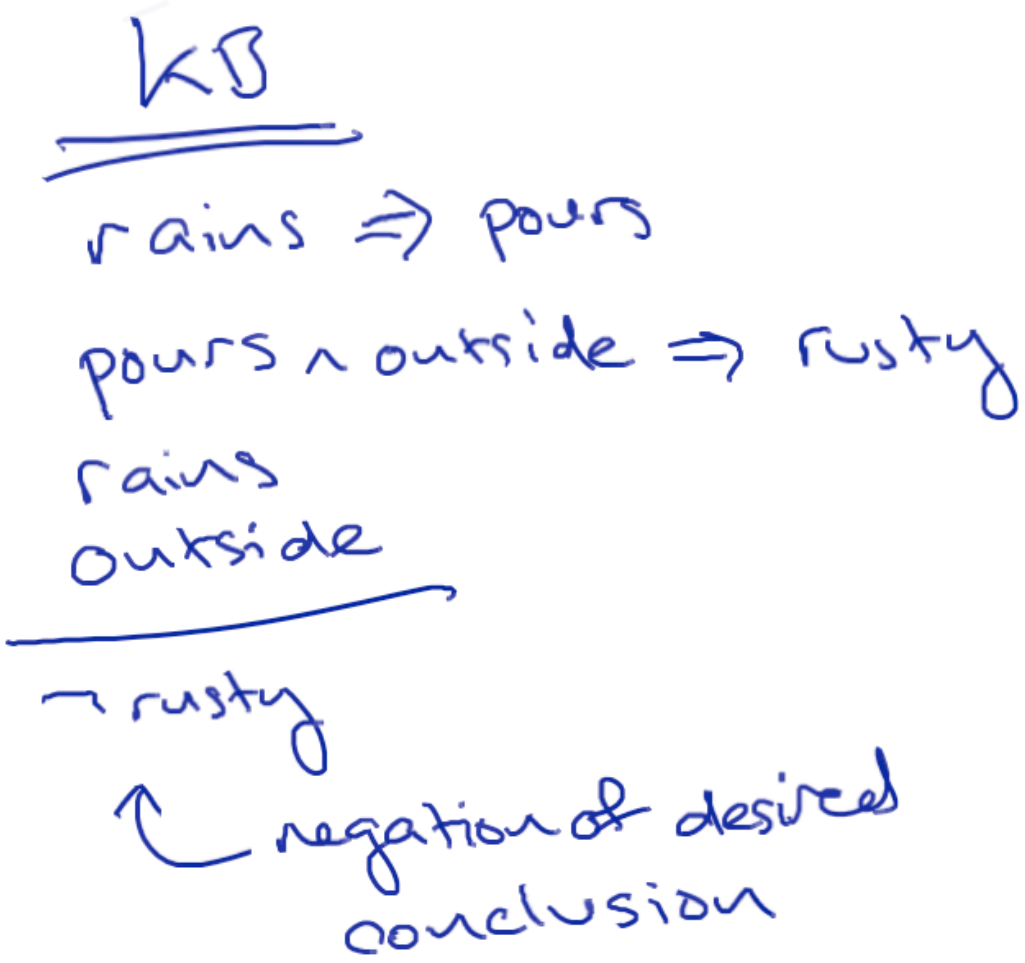
rains

outside

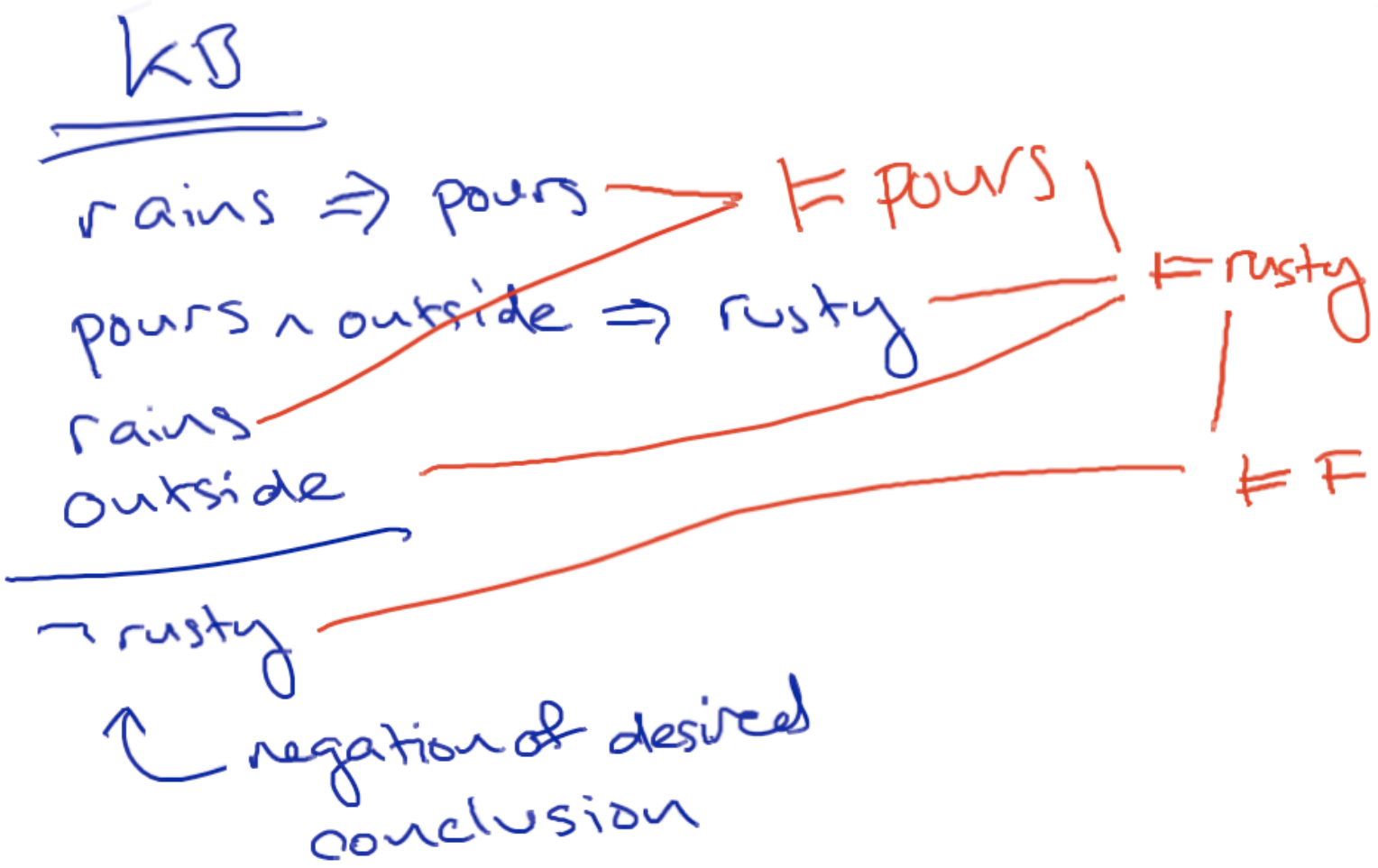
Proof by contradiction

- *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



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…

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

- ***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:*

  $(\neg raining \vee grass\text{-}wet) \wedge \neg grass\text{-}wet \models \neg raining$

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

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

- *Case* $c = True$

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

  $= (T) \wedge (d \vee e)$

  $= (d \vee e)$

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

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

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

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

  as desired

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

- *Modus ponens by itself is **incomplete***
- *Resolution is **complete** for propositional logic*
  - *not obvious—famous theorem due to Robinson*
  - *caveat: also need **factoring**, removal of redundant literals* $(a \vee b \vee a) = (a \vee b)$

Variations

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

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

- *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*

Handling uncertainty

- *Fuzzy logic / certainty factors*
  - *simple, but don't scale*
- *Nonmonotonic logic*
  - *also doesn't scale*
- *Probabilities*
  - *may or may not scale—more later*
  - *Dempster-Shafer (interval probability)*

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

- *Famously difficult to debug a KB with certainty factors, because…*
- *it's hard to separate a large KB into mostly-independent chunks that interact only through a well-defined interface, because…*
- *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

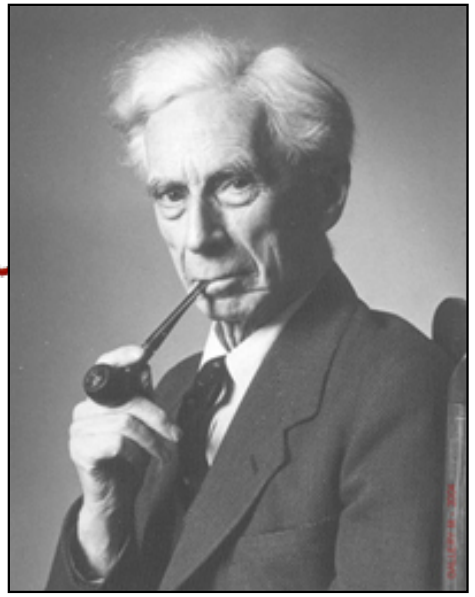
- *Now set as few abnormality predicates as possible*
- *Can prove flies(Polly) or flies(Tux) with no ab(x) assumptions*
- *If we assert ¬flies(Tux), must now assume ab(Tux) to maintain consistency*
- *Can't prove flies(Tux) any more, but can still prove flies(Polly)*

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 …*

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 ≠ 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 and Boolean*
  - *done(slides(15-780)) ⇒ happy(professor(15-780))*
- *Functions convert objects to objects; predicates convert objects to Booleans; connectives convert 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*** *= literals joined by connectives like* $\wedge \vee \neg \Rightarrow$
  - *raining*
  - *done(slides(780))* $\Rightarrow$ *happy(professor)*

Semantics

- *Models are now much more complicated*
  - *List of objects (finite or countable)*
  - *Table of function values for each function mentioned in formula*
  - *Table of predicate values for each predicate mentioned in formula*
- *Meaning of sentence: model $\mapsto$ {T, F}*
- *Meaning of term: model $\mapsto$ object*

For example

KB describing example

- $alive(cat)$
- $ear\text{-}of(cat) = ear$
- $in(cat, box) \wedge in(ear, box)$
- $\neg in(box, cat) \wedge \neg in(cat, nil) \ldots$
- $ear\text{-}of(box) = ear\text{-}of(ear) = ear\text{-}of(nil) = nil$
- $cat \neq box \wedge cat \neq ear \wedge cat \neq nil \ldots$

Aside: avoiding verbosity

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

Aside: typed variables

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

Model of example

- *Objects: C, B, E, N*
- *Assignments:*
  - *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), ¬in(C, C), ¬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

- *Want "all men are mortal," or closure*
- *Add **quantifiers** and **object variables***
  - $\forall x.\ man(x) \Rightarrow mortal(x)$
  - $\neg \exists x.\ lunch(x) \wedge free(x)$
- $\forall$*: no matter how we fill in object variables, formula is still true*
- $\exists$*: there is some way to fill in object variables to make formula true*

Variables

- *Build atoms from variables x, y, … as well as constants John, Fred, …*
  - *man(x), loves(John, z), mortal(brother(y))*
- *Build formulas from these atoms*
  - *man(x) ⇒ mortal(brother(x))*
- *New syntactic construct: term or formula w/* ***free variables***

New syntax ⇒ new semantics

- *New part of model:* ***interpretation***
- *Maps variables to model objects*
  - $x: C, y: N$
- *Meaning of a term or formula: look up its free variables in the interpretation, then continue as before*
- $alive(ear(x)) \mapsto alive(ear(C)) \mapsto alive(E) \mapsto T$

Working with interpretations

- *Write (M / x: obj) for the model which is just like M except that variable x is interpreted as the object obj*
- *M / x: obj is a **refinement** of M*

Binding

- *Adding quantifier for x is called **binding** x*
  - *In (∀x. likes(x, y)), x is bound, y is free*
- *Can add quantifiers and apply logical operations like ∧∨¬ in any order*
- *But must wind up with **ground** formula (no free variables)*

Semantics of ∀

- *A sentence (∀x. S) is true in M if S is true in (M / 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*
  - *since happy(A) is satisfied in M/x:A, happy(B) in M/x:B, happy(C) in M:x/C*

Semantics of ∃

- *A sentence (∃x. S) is true in M if there is some object obj in M such that S is true in model (M / x: obj)*

Example

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

Scoping rules

- *Portion of formula where quantifier applies = **scope***
- *Variable is bound by **innermost** enclosing scope with matching name*
- *Two variables in different scopes can have same name—they are still different vars*

Scoping examples

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

Quantifier nesting

- *English sentence "everybody loves somebody" is ambiguous*
- *Translates to logical sentences*
  - $\forall x.\, \exists y.\, loves(x, y)$
  - $\exists y.\, \forall x.\, loves(x, y)$

Reasoning in FOL

Entailment, etc.

- *As before, entailment, unsatisfiability, validity, equivalence, etc. refer to all possible models*
- *But now, can't determine by enumerating models*
  - *since there could be infinitely many*

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 ∃?

- $\exists x.\ happy(x)$
- $happy(happy\_person())$

- $\forall y.\ \exists x.\ loves(y, x)$
- $\forall y.\ loves(y, loved\_one(y))$

Skolemization

Thoraf Albert Skolem
1887–1963

- *Called **Skolemization** (after Thoraf Albert Skolem)*
- *Eliminate ∃ using function of arguments of all enclosing ∀ quantifiers*

Do we really need ∀?

Getting rid of quantifiers

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

For example

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

Exercise

- $(\forall x.\ honest(x)) \Rightarrow happy(Diogenes)$

Proofs

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:*

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

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

Unification

- *Two FOL sentences **unify** with each other if there is a way to set their variables so that they are identical*
- *man(x), man(Socrates) unify using the substitution x = Socrates*

Unification examples

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

Most general unifier

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

First-order resolution

- *Given clauses* $(a \vee b \vee c)$, $(\neg c' \vee d \vee e)$
- *And a variable substitution V*
- *If c : V and c' : V are the same*
- *Then we can conclude*
- $(a \vee b \vee d \vee e) : V$

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*

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

- ***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)*
- *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 ⇒ unsatisfiable propositional KB*

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*

Proofs w/ Herbrand & Robinson

- *So, FOL KB is unsatisfiable **if and only if** there is a subset of Herbrand universe making PKB unsatisfiable*
- *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}*

Making it faster

- *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*

Why FOL

What's so special about FOL

- *Existence of a sound, complete inference procedure*
- *Additions to FOL break this procedure*
  - *although sometimes there's a replacement*

Equality

- ***Paramodulation*** *is sound and complete for FOL+equals (see RN)*

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 knows P(x)")*
- *First-class functions (lambda operator, application)*
- …