First-Order Logic
(aka First Order Predicate Calculus)

First-order logic

models the world in terms of

- **Objects,** which are things with individual identities
- **Properties** of objects that distinguish them from other objects
- **Relations** that hold among sets of objects
- **Functions,** which are a subset of relations where there is only one “value” for any given “input”

Examples

- **Constant symbols,** which represent objects/individuals
  - Mary
  - 3
  - Green
- **Function symbols,** which map individuals to individuals
  - father-of(Mary) = John
  - color-of(Sky) = Blue
- **Predicate symbols,** which map individuals to truth values
  - greater(5,3)
  - green(Grass)
  - color(Grass, Green)

are all provided by the user

First-order logic

- **Variable symbols**
  - E.g., x, y, foo
- **Connectives**
  - as in PL: not ($\neg$), and ($\wedge$), or ($\vee$), implies ($\rightarrow$), if and only if (biconditional $\leftrightarrow$)
- **Quantifiers**
  - Universal $\forall$**x** or **(Ax)**
  - Existential $\exists$**x** or **(Ex)**

Quantifiers

- **Universal quantification**
  - $(\forall x)P(x)$ means that P holds for **all** values of x in the domain associated with that variable
  - E.g., $(\forall x)$ dolphin(x) $\rightarrow$ mammal(x)
- **Existential quantification**
  - $(\exists\ x)P(x)$ means that P holds for **some** value of x in the domain associated with that variable
  - E.g., $(\exists\ x)$ mammal(x) $\wedge$ lays-eggs(x)
  - allows to make a statement about some object without naming it

Sentences

are built from terms and atoms

- a **term** (denoting a individual) is a constant symbol, a variable symbol, or an n-place function of n terms
  - e.g., Mary, x, $f(x_1, ..., x_n)$ where each x_i is a term
  - a term with no variables is a **ground term**
- an **atomic sentence** (which has value true or false) is an n-place predicate of n terms
- a **complex sentence** is formed from atomic sentences connected by the logical connectives
  - $\neg P$, $P \vee Q$, $P \wedge Q$, $P \rightarrow Q$, $P \leftrightarrow Q$ where P and Q are sentences
- a **quantified sentence** has quantifiers $\forall$ and $\exists$
- a **well-formed formula** (**wff**) is a sentence containing no *free* variables,
  - i.e., all variables are *bound* by universal or existential quantifiers
  - $(\forall x)P(x,y)$ has x bound as a universally quantified variable, but y is free

A BNF for FOL

```
S := <Sentence> ;
<Sentence> := <AtomicSentence> |
          <Sentence> <Connective> <Sentence> |
          <Quantifier> <Variable>,... <Sentence> |
          "NOT" <Sentence> |
          "(" <Sentence> ")";
<AtomicSentence> := <Predicate> "(" <Term>, ... ")" |
                    <Term> "=" <Term>;
<Term> := <Function> "(" <Term>, ... ")" |
          <Constant> |
          <Variable>;
<Connective> := "AND" | "OR" | "IMPLIES" | "EQUIVALENT";
<Quantifier> := "EXISTS" | "FORALL" ;
<Constant> := "A" | "X1" | "John" | ... ;
<Variable> := "a" | "x" | "s" | ... ;
<Predicate> := "Before" | "HasColor" | "Raining" | ... ;
<Function> := "Mother" | "LeftLegOf" | ... ;
```

Precedence in FOL
(Saving Parentheses)

- just like in PL (or in arithmetic), proper use of parentheses can be tedious
- PL precedence extended now for FOL
- with quantifiers having the least priority

i.e.,

not > and > or > implies > if and only if > forall = exists

e.g.,

$(((\forall x\ Px) \wedge T) \rightarrow (U \vee (V \wedge T)\ )) = \forall x\ Px \wedge T \rightarrow U \vee V \wedge T$

Notes on Quantifiers

- Universal quantifiers are often used with “implies” to form “rules”:

  $(\forall x)$ student(x) $\rightarrow$ smart(x) means “All students are smart”
- Universal quantification *rarely* used for statements about every individual:

  $(\forall x)$student(x)$\wedge$smart(x) means “Everyone in the world is a student and is smart”

- Existential quantifiers usually with “and” to specify a list of properties:

  $(\exists x)$ student(x) $\wedge$ smart(x) means “There is a student who is smart”
- Common mistake to represent above statement in FOL as:

  $(\exists x)$ student(x) $\rightarrow$ smart(x)

  But what happens when there is a person who is *not* a student?

Examples of FOL

- **Every gardener likes the sun.**

  $\forall x\ gardener(x) \rightarrow likes(x,Sun)$
- **You can fool some of the people all of the time.**

  $\exists x\ \forall t\ \ person(x) \wedge time(t) \rightarrow can\text{-}fool(x,t)$
- **You can fool all of the people some of the time.** (two ways)

  $\forall x\ \exists t\ (person(x) \rightarrow time(t) \wedge can\text{-}fool(x,t))$

  $\forall x\ (person(x) \rightarrow \exists t\ (time(t) \wedge can\text{-}fool(x,t))$
- **All purple mushrooms are poisonous**.

  $\forall x\ (mushroom(x) \wedge purple(x)) \rightarrow poisonous(x)$

Quantifier Scope

- FOL sentences have structure, like programs
- especially, variables in a sentence have a scope
- e.g.,“everyone who is alive loves someone”

$$(\forall x)\ \text{alive}(x) \rightarrow (\exists y)\ \text{loves}(x,y)$$

Scope of x

Scope of y

Quantifier Scope

Switching universal / existential quantifiers

- *does not* change the meaning
- $(\forall x)(\forall y)P(x,y) \leftrightarrow (\forall y)(\forall x)\ P(x,y)$ - “Dogs hate cats”.
- $(\exists x)(\exists y)P(x,y) \leftrightarrow (\exists y)(\exists x)\ P(x,y)$ - “A cat killed a dog”

Switching universals and existentials

- *does* change meaning
- Everyone likes someone: $(\forall x)(\exists y)$ likes(x,y)
- Someone is liked by everyone: $(\exists y)(\forall x)$ likes(x,y)

Connections between All and Exists

using De Morgan's laws:

1. $(\forall x)\ \neg P(x) \leftrightarrow \neg(\exists x)\ P(x)$
2. $\neg(\forall x)\ P \leftrightarrow (\exists x)\ \neg P(x)$
3. $(\forall x)\ P(x) \leftrightarrow \neg\ (\exists x)\ \neg P(x)$
4. $(\exists x)\ P(x) \leftrightarrow \neg(\forall x)\ \neg P(x)$

Connections between All and Exists

examples

1. All dogs don’t like cats $\leftrightarrow$ No dog likes cats
2. Not all dogs dance $\leftrightarrow$ There is a dog that doesn’t dance
3. All dogs sleep $\leftrightarrow$ There is no dog that doesn’t sleep
4. There is a dog that talks $\leftrightarrow$ Not all dogs can’t talk

Expressing Uniqueness

- express that there is a single, unique object that satisfies a certain condition
- notational shortcut $\exists!$

$\exists!\ x\ P(x)$

$$= \exists x\ P(x) \wedge \forall y\ (P(y) \rightarrow x=y)$$

$$= \exists x\ P(x) \wedge \neg\exists y\ (P(y) \wedge x \neq y)$$

Quantified inference rules

- Universal instantiation
  - $\forall x\ P(x) \therefore P(A)$ ← **therefore symbol $\therefore$**
- Universal generalization
  - $P(A) \wedge P(B) \ldots \therefore \forall x\ P(x)$
- Existential instantiation
  - $\exists x\ P(x) \therefore P(F)$ ← **skolem constant F**
- Existential generalization
  - $P(A) \therefore \exists x\ P(x)$

Universal instantiation (a.k.a. universal elimination)

If $(\forall x)\ P(x)$ is true, then $P(C)$ is true, where C is *any* constant in the domain of x

- e.g., $(\forall x)$ eats(Ziggy, x) $\Rightarrow$ eats(Ziggy, IceCream)
- the variable symbol can be replaced
  - by any ground term,
  - i.e., any constant symbol or function symbol applied to ground terms only

Existential instantiation (a.k.a. existential elimination)

From $(\exists x)\ P(x)$ infer $P(c)$

- e.g.: $(\exists x)$ eats(Ziggy, x) $\rightarrow$ eats(Ziggy, Stuff)
- the variable is replaced by a **new constant** (i.e., that is not in any other sentence in the KB)
- aka **skolemization**; constant is a skolem constant
- Convenient
  - to use this to reason about the unknown object,
  - rather than constantly manipulating the existential quantifier

Existential generalization (a.k.a. existential introduction)

If $P(c)$ is true, then $(\exists x)\ P(x)$ is inferred.

- e.g.: eats(Ziggy, IceCream) $\Rightarrow$ $(\exists x)$ eats(Ziggy, x)
- All instances of the constant symbol are replaced by the new variable symbol (i.e., the variable symbol cannot already exist)

(Toy) Example for Use of FOL

- Genealogy Knowledge Base
  - contains facts of immediate family relations (spouses, parents, etc.)
  - contains definitions of more complex relations (ancestors, relatives)
  - is able to answer queries about relationships between people
- Predicates
  - parent(x, y), child(x, y), father(x, y), daughter(x, y), etc.
  - spouse(x, y), husband(x, y), wife(x,y)
  - ancestor(x, y), descendant(x, y)
  - male(x), female(y)
  - relative(x, y)
- Facts
  - husband(Joe, Mary), son(Fred, Joe)
  - spouse(John, Nancy), male(John), son(Mark, Nancy)
  - father(Jack, Nancy), daughter(Linda, Jack)
  - daughter(Liz, Linda)
  - etc.

Rules for Genealogical Relations

$(\forall x,y)$ parent(x, y) $\leftrightarrow$ child (y, x)
$(\forall x,y)$ father(x, y) $\leftrightarrow$ parent(x, y) $\wedge$ male(x) //similarly mother(x, y)
$(\forall x,y)$ daughter(x, y) $\leftrightarrow$ child(x, y) $\wedge$ female(x) //similarly son(x, y)
$(\forall x,y)$ husband(x, y) $\leftrightarrow$ spouse(x, y) $\wedge$ male(x) //similarly wife(x, y)
$(\forall x,y)$ spouse(x, y) $\leftrightarrow$ spouse(y, x) //symmetric
$(\forall x,y)$ parent(x, y) $\rightarrow$ ancestor(x, y)
$(\forall x,y)(\exists z)$ parent(x, z) $\wedge$ ancestor(z, y) $\rightarrow$ ancestor(x, y)
$(\forall x,y)$ descendant(x, y) $\leftrightarrow$ ancestor(y, x)
$(\forall x,y)(\exists z)$ ancestor(z, x) $\wedge$ ancestor(z, y) $\rightarrow$ relative(x, y)
$(\forall x,y)$ spouse(x, y) $\rightarrow$ relative(x, y)
$(\forall x,y)(\exists z)$ relative(z, x) $\wedge$ relative(z, y) $\rightarrow$ relative(x, y) //transitive
$(\forall x,y)$ relative(x, y) $\leftrightarrow$ relative(y, x) //symmetric

Example Queries

- ancestor(Jack, Fred) // the answer is yes
- relative(Liz, Joe) // the answer is yes
- relative(Nancy, Matthew)
  // no answer in general,
  // no if under **closed world assumption**

- ($\exists$z) ancestor(z, Fred) $\wedge$ ancestor(z, Liz) ?

Semantics of FOL

- **Domain M:** the set of all objects in the world (of interest)
- **Interpretation I:** includes
  - Assign each constant to an object in M
  - Define each function of n arguments as a mapping M^n => M
  - Define each predicate of n arguments as a mapping M^n => {T, F}
  - Therefore, every ground predicate with any instantiation will have a truth value
  - In general there is an infinite number of interpretations because |M| is infinite
- **Define logical connectives: ~, ^, v, =>, <=>** as in PL
- **Define semantics of (∀x) and (∃x)**
  - (∀x) P(x) is true iff P(x) is true under all interpretations
  - (∃x) P(x) is true iff P(x) is true under some interpretation

Semantics of FOL

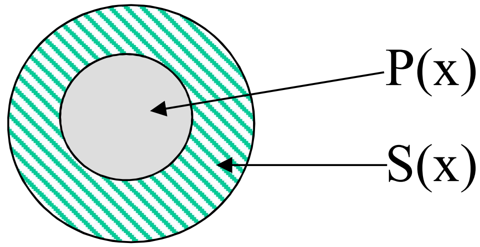
- **Model**
  - an interpretation of a set of sentences
  - such that every sentence is *True*
- **A sentence is**
  - **satisfiable** if it is true under some interpretation
  - **valid** if it is true under all possible interpretations
  - **inconsistent** if there does not exist any interpretation under which the sentence is true
- **Logical consequence**
  - S |= X if all models of S are also models of X

Axioms, Definitions and Theorems

- **axioms**
  - facts and rules that attempt to capture all of the (foundational) facts and concepts about a domain
- axioms can be used to prove **theorems**
  - Mathematicians (and other clever, i.e. “lazy” people ☺) do not want any unnecessary axioms
  - i.e., no axioms that can be derived from other axioms
  - but dependent axioms can make reasoning faster
  - a good set of axioms for a domain is a kind of design problem
- **definition** of a predicate is of the form “$p(X) \leftrightarrow \ldots$”
  - can be decomposed into two parts
  - **Necessary** description: “$p(x) \rightarrow \ldots$”
  - **Sufficient** description “$p(x) \leftarrow \ldots$”

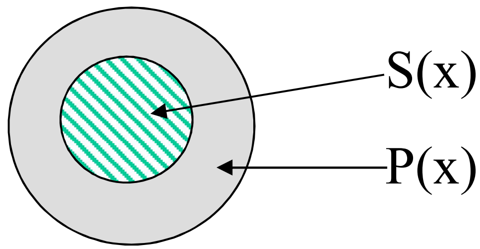
Necessary & Sufficient

S(x) is a necessary condition of P(x)



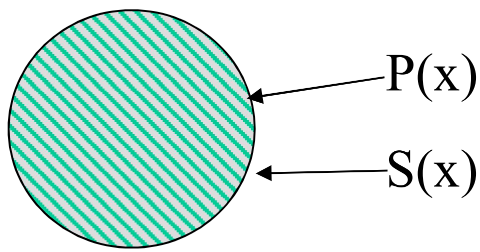
$(\forall x)\ P(x) \Rightarrow S(x)$

S(x) is a sufficient condition of P(x)



$(\forall x)\ P(x) \Leftarrow S(x)$

S(x) is a necessary and sufficient condition of P(x)



$(\forall x)\ P(x) \Leftrightarrow S(x)$

Example: Axioms for Set Theory

1. The only sets are the empty set and those made by adjoining something to a set:

   $\forall$s set(s) <=> (s=EmptySet) $\vee$ ($\exists$x,r Set(r) $\wedge$ s=Adjoin(s,r))

2. The empty set has no elements adjoined to it:

   ~ $\exists$x,s Adjoin(x,s)=EmptySet

3. Adjoining an element already in the set has no effect:

   $\forall$x,s Member(x,s) <=> s=Adjoin(x,s)

4. The only members of a set are the elements that were adjoined into it:

   $\forall$x,s Member(x,s) <=> $\exists$y,r (s=Adjoin(y,r) $\wedge$ (x=y $\vee$ Member(x,r)))

5. A set is a subset of another iff all of the 1st set's members are members of the 2^{nd}:

   $\forall$s,r Subset(s,r) <=> ($\forall$x Member(x,s) => Member(x,r))

6. Two sets are equal iff each is a subset of the other:

   $\forall$s,r (s=r) <=> (subset(s,r) $\wedge$ subset(r,s))

7. Intersection

   $\forall$x,s1,s2 member(X,intersection(S1,S2)) <=> member(X,s1) $\wedge$ member(X,s2)

8. Union

   $\exists$x,s1,s2 member(X,union(s1,s2)) <=> member(X,s1) $\vee$ member(X,s2)

Higher-order logic

FOL

- only allows to quantify over variables,
- and variables can only range over objects

HOL allows us to quantify over relations

- example: (quantify over functions)

  "two functions are equal iff they produce the same value for all arguments"

  $\forall f\ \forall g\ (f = g) \leftrightarrow (\forall x\ f(x) = g(x))$
- example: (quantify over predicates)

  $\forall r\ \text{transitive}(r) \rightarrow (\forall xyz)\ r(x,y) \wedge r(y,z) \rightarrow r(x,z))$
- more expressive, but undecidable

FOL Inference with Horn Clauses & Generalized Modus Ponens

FOL Inference

- FOL harder than PL
  - variables can take on an *infinite* number of values
  - hence, potentially *infinite* number of ways to apply the Universal Elimination rule
- *Gödel's Completeness Theorem*:
  FOL entailment is only *semidecidable*
  - If a sentence is **true** given a set of axioms, there is a procedure that will determine this
  - If the sentence is **false**, then there is no guarantee that a procedure will ever determine this, i.e., it **may never halt**

Generalized Modus Ponens

- Modus Ponens: P, P=>Q |= Q
- Generalized Modus Ponens (GMP)
  - combines And-Introduction, Universal-Elimination, and Modus Ponens
  - $P(c)$ *and* $Q(c)$ *and* $\forall x\ P(x) \wedge Q(x) \rightarrow R(x) \vdash R(c)$

Horn clauses

a FOL Horn clause is a sentence of the form:

$$P_1(x) \wedge P_2(x) \wedge ... \wedge P_n(x) \rightarrow Q(x)$$

where

- ≥ 0 P_is and 0 or 1 Q
- the P_is and Q are positive (i.e., non-negated) literals

- equivalently: $P_1(x) \vee P_2(x) \ldots \vee P_n(x)$ where the P_i are all atomic and *at most one* is positive
- Horn clauses represent a *subset* of the set of sentences representable in FOL
- programming in Prolog is based on Horn clauses

Horn clauses

- special cases
  - *Typical rule:* $P_1 \wedge P_2 \wedge \ldots P_n \rightarrow Q$
  - *Constraint:* $P_1 \wedge P_2 \wedge \ldots P_n \rightarrow \text{false}$
  - *A fact:* $\text{true} \rightarrow Q$
- not Horn clauses
  - $p(a) \vee q(a)$
  - $(P \wedge Q) \rightarrow (R \vee S)$

Horn clauses

- quantifiers
  - variables in conclusion: universally quantified
  - variables in premises: existentially quantified
- Example: grandparent relation
  - parent(P1, X) $\wedge$ parent(X, P2) $\rightarrow$ grandParent(P1, P2)
  - $\forall$ P1,P2 $\exists$ X parent(P1,X) $\wedge$ parent(X, P2) $\rightarrow$ grandParent(P1, P2)
  - Prolog: grandParent(P1,P2) :- parent(P1,X), parent(X,P2)

Forward chaining (FC)

- proof
  - start with the given axioms/premises in KB
  - deriving new sentences using GMP
  - until the goal/query sentence is derived
- inference using GMP
  - is **sound** and **complete**
  - for KBs containing **only Horn clauses**

Backward chaining (BC)

- Proofs
  - start with the goal query
  - find rules with that conclusion
  - and then prove each of the antecedents in the implication
  - Keep going until you reach premises
    - Avoid loops: check if new subgoal is already on the goal stack
    - Avoid repeated work: check if new subgoal
      - Has already been proved true
      - Has already failed
- Backward-chaining deduction using GMP
  - is also sound and complete
  - for KBs containing only Horn clauses

Forward vs. backward chaining

- FC is data-driven
  - "Automatic" processing
  - E.g., object recognition, routine decisions
  - May do a lot of work that is irrelevant to the goal
  - Efficient when you want to compute all conclusions
- BC is goal-driven
  - tends to be better for problem-solving
  - Where are my keys? How do I get to my next class?
  - Efficient when you want one or a few decisions

Mixed strategy

- option: bi-directional search
- many practical reasoning systems do both forward and backward chaining
- encoding of a rule determines how it is used, e.g.,

```
% this is a forward chaining rule
spouse(X,Y) => spouse(Y,X).
% this is a backward chaining rule
wife(X,Y) <= spouse(X,Y), female(X).
```

Completeness of GMP

GMP (using forward or backward chaining)

- is complete for KBs with only Horn clauses
- is *not* complete for KBs with non-Horn clauses

alternative

- resolution on the CNF form
- linear time for PL
- polynomial on FOL with unification (more next)

FOL Inference with Resolution

Resolution

- resolution
  - **sound** and **complete** inference procedure
  - for unrestricted FOL
- reminder: PL resolution

  $P_1 \vee P_2 \vee \ldots \vee P_n$

  $\neg P_1 \vee Q_2 \vee \ldots \vee Q_m$

  resolvent: $P_2 \vee \ldots \vee P_n \vee Q_2 \vee \ldots \vee Q_m$

need to extend this to quantifiers and variables

Resolution covers many cases

- Modus Ponens
  - from P and $P \rightarrow Q$ derive Q
  - from P and $\neg P \vee Q$ derive Q
- Chaining
  - from $P \rightarrow Q$ and $Q \rightarrow R$ derive $P \rightarrow R$
  - from $(\neg P \vee Q)$ and $(\neg Q \vee R)$ derive $\neg P \vee R$
- Contradiction detection
  - from P and $\neg P$ derive false
  - from P and $\neg P$ derive the empty Horn clause (=false)

Resolution in first-order logic

Given sentences in *conjunctive normal form:*

- $P_1 \vee \ldots \vee P_n$ and $Q_1 \vee \ldots \vee Q_m$
- P_i and Q_i are literals

if P_j and $\neg Q_k$ **unify** with substitution list θ,
then derive the resolvent sentence:

- $\text{subst}(\theta, P_1 \vee \ldots \vee P_{j-1} \vee P_{j+1} \ldots P_n \vee Q_1 \vee \ldots Q_{k-1} \vee Q_{k+1} \vee \ldots \vee Q_m)$

Example

- from clause $\underline{\mathbf{P(x, f(a))}} \vee \mathbf{P(x, f(y))} \vee \mathbf{Q(y)}$
- and clause $\underline{\neg\mathbf{P(z, f(a))}} \vee \neg\mathbf{Q(z)}$
- derive resolvent $\mathbf{P(z, f(y))} \vee \mathbf{Q(y)} \vee \neg\mathbf{Q(z)}$
- Using $\boldsymbol{\theta} = \{\mathbf{x/z}\}$

Converting to CNF

1. Eliminate all <=> connectives

   $(P \leftrightarrow Q) \Rightarrow ((P \rightarrow Q) \wedge (Q \rightarrow P))$

2. Eliminate all $\rightarrow$ connectives

   $(P \rightarrow Q) \Rightarrow (\neg P \vee Q)$

3. Reduce the scope of each negation symbol to single predicate

   $\neg\neg P \Rightarrow P$

   $\neg(P \vee Q) \Rightarrow \neg P \wedge \neg Q$

   $\neg(P \wedge Q) \Rightarrow \neg P \vee \neg Q$

   $\neg(\forall x)P \Rightarrow (\exists x)\neg P$

   $\neg(\exists x)P \Rightarrow (\forall x)\neg P$

4. Standardize variables
   - rename all variables
   - so that each quantifier has its own unique variable name

Converting to CNF

5. Eliminate existential quantification
   by Skolem constants/functions

$(\exists x)P(x) \Rightarrow P(C)$
   C is a Skolem constant (a new constant symbol)

$(\forall x)(\exists y)P(x,y) \Rightarrow (\forall x)P(x, f(x))$
   since $\exists$ is within scope of a universally quantified variable,
   use a **Skolem function f** to construct a new value that **depends on** the universally quantified variable (f must be a new function name)

E.g., $(\forall x)(\exists y)loves(x,y) \Rightarrow (\forall x)loves(x,f(x))$
   i.e., f(x) specifies the person that x loves

Converting to CNF

6. remove universal quantifiers
   - move them all to the left end
   - make the scope of each the entire sentence
   - drop the “prefix” part

   Ex: $(\forall x)P(x) \Rightarrow P(x)$
7. use distributive and associative laws

   $(P \wedge Q) \vee R \Rightarrow (P \vee R) \wedge (Q \vee R)$

   $(P \vee Q) \vee R \Rightarrow (P \vee Q \vee R)$
8. split conjuncts into separate clauses
9. standardize variables: each clause contains only variable names that do not occur in any other clause

Resolution in first-order logic

Unification

- process of finding all legal substitutions
- that make logical expressions look identical

i.e., a pattern matching process
(can be done with efficiently in linear time)

Unification

- **pattern-matching** procedure
  - takes two atomic sentences as input
  - returns “failure” if they do not match and a substitution list θ if they do
- i.e., $unify(p,q) = \theta$ means $subst(\theta, p) = subst(\theta, q)$ for two atomic sentences, *p* and *q*
- **θ** is called the **most general unifier** (mgu)

Unification Examples

Knows(John,x), Knows(John, Jane)

–> {x/Jane)

Knows(John,x), Knows(y, Bill)
–> {x/Bill, y/John)

Knows(John,x), Knows(x, Bill)
–> Fail

Unification algorithm

```
procedure unify(p, q) =  θ
   Scan p and q left-to-right and find the first corresponding
     terms where p and q "disagree" (i.e., p and q not equal)
   If there is no disagreement, return θ  (success!)
   Let r and s be the terms in p and q, respectively,
     where disagreement first occurs
   If variable(r) then {
     Let θ = union(θ, {r/s})
     Return unify(subst(θ, p), subst(θ, q), θ)
   } else if variable(s) then {
     Let θ = union(θ, {s/r})
     Return unify(subst(θ, p), subst(θ, q), θ)
   } else return "Failure"
  end
```

Unification: Remarks

Unify

- is a linear-time algorithm
- returns the most general unifier
- i.e., the shortest-length substitution list
  that makes the two literals match

important constraint:

- a variable can never be replaced by a term containing that variable
- example: x/f(x) is illegal
- should be checked when making the recursive calls

Why Unification

Given:

$\neg$ knows(John, x) $\vee$ likes(John, x)

knows (John, Jenny)

unification: {x/Jenny}

$\neg$ knows(John, Jenny) $\vee$ likes(John, Jenny)

knows (John, Jenny)

-------------- *resolution* --------------

likes(John, Jenny)

Resolution Refutation

Given:

- consistent set of axioms KB
- and goal sentence Q

show that KB |= Q

Proof by contradiction:
add $\neg$Q to KB and try to prove false,
i.e. (KB |- Q) $\leftrightarrow$ (KB $\wedge$ $\neg$Q |- False)

aka ***Davis-Putnam*** Algorithm for FOL

Resolution refutation

Resolution is **refutation complete**

- can show that a given sentence Q is entailed by KB
- but it can not (in general) generate all logical consequences of a set of sentences
- also, it cannot be used to prove that Q is not entailed by KB
- so, resolution will not always give an answer (since entailment is only semi-decidable)

Note: Why not just run two proofs in parallel?
One tries to prove Q and the other try to prove $\neg$Q ... ?!?!
(KB might not entail either one)

Resolution example

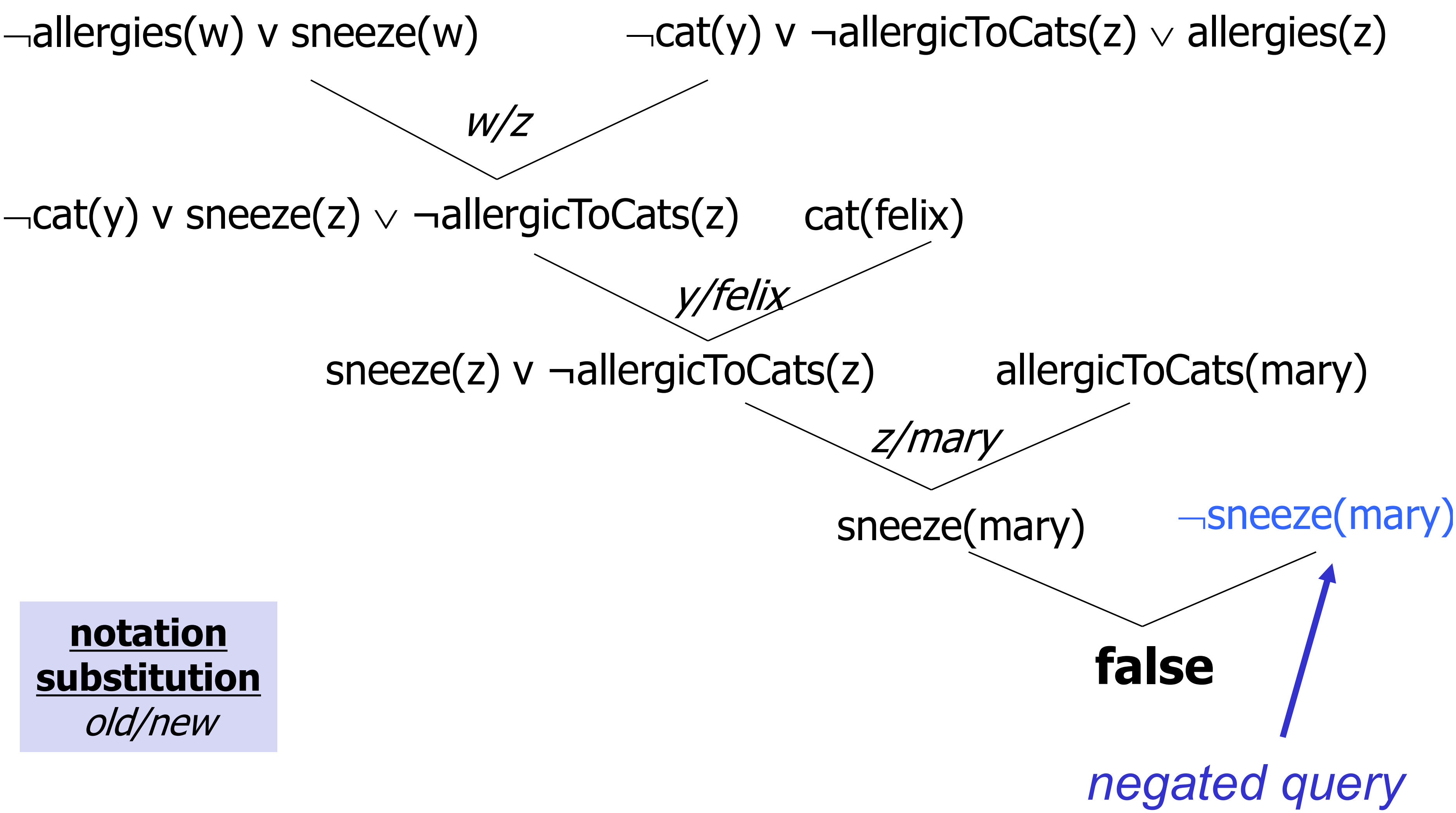
KB:

- allergies(X) $\rightarrow$ sneeze(X)
- cat(Y) $\wedge$ allergicToCats(X) $\rightarrow$ allergies(X)
- cat(felix)
- allergicToCats(mary)

Goal:

- sneeze(mary)

Refutation Resolution as Proof Tree



Refutation Resolution
(in “classical” list form)

1. $\neg$allergies(w) v sneeze(w)
2. $\neg$cat(y) v $\neg$allergicToCats(z) v allergies(z)
3. cat(felix)
4. allergicToCats(mary)
5. $\neg$ sneeze(mary)
6. $\neg$ cat(y) v sneeze(w) v $\neg$allergicToCats(z) [1.&2., w/z]
7. sneeze(w) v $\neg$allergicToCats(z) [3.&6., y/felix]
8. sneeze(mary) [4.&7., z/mary]
9. **false** [5.&8.]

(Simple) Summary of Logic

Automated Theorem Proving

Boolean aka Propositional Logic

- **CNF**: (a+b+c+...), (d+e+...)
- **resolution**:
  a+c1+..., -a+d1+... |- c1+...+d1+...
- proof by **refutation**: add negated query q to KB
  (-q must also be in CNF)

sound & complete

Automated Theorem Proving

<u>First Order Logic aka 1st Order Predicate Calculus</u>
(Davis-Putnam; like PL but...)

- CNF including **skolemization**
- **unification** when using resolution
- proof by refutation

sound & refutation complete

Automated Theorem Proving

<u>First Order Logic aka 1st Order Predicate Calculus</u>

Davis-Putnam:

CNF, resolution with unification, refutation

sound & **refutation complete**

Gödel: FOL is only *semidecidable*

- if a sentence is **true** (given a set of axioms) this can be algorithmically determined (e.g., Davis-Putnam)
- if the sentence is **false**, then there is no guarantee that any procedure will ever determine this, i.e., it **may never halt**

Automated Theorem Proving

<u>special case in FOL: Horn Clauses</u>

(useful subset of FOL)

$$P_1(x) \wedge P_2(x) \wedge \ldots \wedge P_n(x) \rightarrow Q(x)$$

- ≥ 0 P_is and 0 or 1 Q
- the P_is and Q are positive (i.e., non-negated) literals
- variables in conclusion: universally quantified
- variables (only) in premises: existentially quantified

Automated Theorem Proving

Horn Clauses

$$P_1(x) \wedge P_2(x) \wedge \ldots \wedge P_n(x) \rightarrow Q(x)$$

- *typical rule:* $P_1 \wedge P_2 \wedge \ldots P_n \rightarrow Q$
- *constraint:* $P_1 \wedge P_2 \wedge \ldots P_n \rightarrow \text{false}$
- *fact:* $\text{true} \rightarrow Q$

Automated Theorem Proving

Horn Clauses

$$P_1(x) \wedge P_2(x) \wedge \ldots \wedge P_n(x) \rightarrow Q(x)$$

inference with **Generalized Modus Ponens (GMP)**

$P(c)$ *and* $Q(c)$ *and* $\forall x\, P(x) \wedge Q(x) \rightarrow R(x) \vdash R(c)$

sound & complete