First-order logic

- First-order logic (FOL) 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”

Ex.: Objects: Students, lectures, companies, cars ...

- Relations: Brother-of, bigger-than, outside, part-of, has-color, occurs-after, owns, visits, precedes, ...
- Properties: blue, oval, even, large, ...
- Functions: father-of, best-friend, second-half, one-more-than ...

Atomic Sentences

- Propositions are represented by a **predicate applied to a** tuple of terms. A predicate represents a property of or relation between terms that can be true or false:
- Brother(John, Fred), Left-of(Square1, Square2), GreaterThan(plus(1,1), plus(0,1))
- **Sentences in logic state facts** that are true or false.
- In FOL properties and n-ary relations do express that:

LargerThan(2,3) is false. Brother(Mary,Pete) is false.

- Note: **Functions do not state facts** and form no sentence: Brother(Pete) refers to the object John (his brother) and is neither true nor false.
- Brother(Pete,Brother(Pete)) is True.

Binary relation ↑ Brother

Function ↑ Brother(Pete)

Syntax of First-order logic

Sentence → *Atomicsentence*
| (*Sentence Connective Sentence*)
| *Quantifier Variable,. . . Sentence*
| >*Sentence*
AtomicSentence → *Predicate(Term, . . .)*
| (*Term* = *Term*
Term -→ *Function(Term, . . .)*
I *Constant*
| *Variable*
Connective → ¬, ∧, ∨, ⇒
Quantifier → ∀, ∃
Constant → *A* (XI (*John* 1 . . .
Variable → *a* | *x* | s | . . .
Predicate → *Before*…
Function → *Mother* | …

Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- Model contains objects (domain elements) and relations among them
- Interpretation specifies referents for

constant symbols → objects
predicate symbols → relations
function symbols → functional relations

- An atomic sentence *predicate*($term_1$,...,$term_n$) is true iff the objects referred to by $term_1$,...,$term_n$ are in the relation referred to by *predicate*

Entailment

- Entailment means that one thing follows from another:

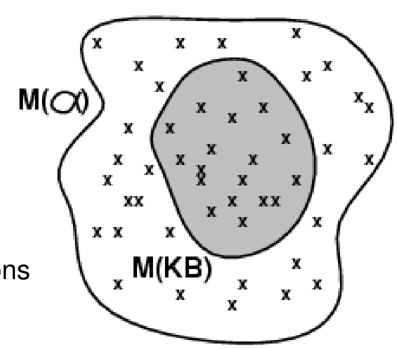
$$KB \models \alpha$$

Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true

  - E.g., the KB containing "the Greens won" and "the Reds won" entails "Either the Greens or the reds won"
  - E.g., x+y = 4 entails 4 = x+y
  - Entailment is a relationship between sentences (i.e., syntax) that is based on semantics
  - entailment: necessary truth of one sentence given another

Models

- Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated
- We say *m* is a model of a sentence α if α is true in *m*
- *M(α)* is the set of all models of α
- Then $KB \models \alpha$ iff $M(KB) \subseteq M(\alpha)$
  - E.g. *KB* = Greens won and Reds won α = Greens won



- Think of KB and α as collections of constraints and of models m as possible states. M(KB) are the solutions to KB and M(α) the solutions to α. Then, $KB \models \alpha$ when all solutions to KB are also solutions to α.

Inference in FOL chapter 9 in Russel

- $KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i

i.e. deriving sentences from other sentences

- Soundness: i is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

i.e. derivations produce only entailed sentences *(no wrong inferences, but maybe not all inferences)*

- Completeness: i is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

i.e. derivations can produce all entailed sentences (*all inferences can be made, but maybe some wrong extra ones as well)*

Validity and satisfiability

- A sentence is **valid** if it is true in **all models**,
- e.g., *True*, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the following:
$KB \models \alpha$ if and only if ($KB \Rightarrow \alpha$) is valid

A sentence is **satisfiable** if it is true in **some model**
e.g., $A \vee B$, C

A sentence is **unsatisfiable** if it is true in **no models**
e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:
$KB \models \alpha$ if and only if ($KB \wedge \neg \alpha$) is unsatisfiable
(there is no model for which KB=true and is false)

Proof Methods in FOL

Major Families:

- GMP
- Reduction
- Resolution
- Forward chaining
- Backward chaining

Some Other inference tools:

Entailment/ Unification/

Proof Methods in FOL

- GMP: Using the generalized form of Modus Ponense
- Reduction: Reduce all FOL sentences to propositional Calculus then use inference in propositional calculus
- Resolution – Refutation
  - Negate goal
  - Convert all pieces of knowledge into clausal form (disjunction of literals)
  - See if contradiction indicated by null clause $\square$ can be derived
- Forward chaining
  - Given P, $P \rightarrow Q$, to infer Q
  - P, match *L.H.S* of
  - Assert Q from *R.H.S*
- Backward chaining
  - Q, Match *R.H.S* of $P \rightarrow Q$
  - assert P
  - Check if P exists

Universal instantiation (UI)

- Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall v\ \alpha}{\text{Subst}(\{v/g\}, \alpha)}$$

for any variable *v* and ground term *g*

- E.g., ∀x *King*(*x*) ∧ *Greedy*(*x*) ⇒ *Evil*(*x*) yields:

King(*John*) ∧ *Greedy*(*John*) ⇒ *Evil*(*John*)
King(*Richard*) ∧ *Greedy*(*Richard*) ⇒ *Evil*(*Richard*)
King(*Father*(*John*)) ∧ *Greedy*(*Father*(*John*)) ⇒ *Evil*(*Father*(*John*))

Existential instantiation (EI)

- For any sentence α, variable *v*, and constant symbol *k* that does not appear elsewhere in the knowledge base:

$$\frac{\exists v\ \alpha}{\text{Subst}(\{v/k\}, \alpha)}$$

- E.g., ∃*x Crown*(*x*) ∧ *OnHead*(*x*,*John*) yields:

$Crown(C_1) \wedge OnHead(C_1, John)$

provided C_1 is a new constant symbol, called a Skolem constant

Unification

- $\forall x$ *King*(*x*) $\wedge$ *Greedy*(*x*) $\Rightarrow$ *Evil*(*x*)
- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

- Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

- Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

- Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}}
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

- Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x,OJ)	{fail}

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- To unify *Knows(John,x)* and *Knows(y,z)*,

  θ = {y/John, x/z } or θ = {y/John, x/John, z/John} or others…
- There are many possible unifiers for some atomic sentences. The first unifier is more general than the second.
- The UNIFY algorithm returns the most general unifier (MGU) that is unique up to renaming of variables. MGU makes the least commitment to variable values.

The Unification Algorithm

•In order to match sentences in the KB, we need a routine.
•UNIFY(p,q) takes two atomic sentences and returns a substitution that makes them equivalent.
UNIFY(p,q)= θ where SUBST(θ,p)=SUBST(θ,q) θ is called a unifier.

```
function UNIFY(x, y, θ) returns a substitution to make x and y identical
   inputs: x, a variable, constant, list, or compound
           y, a variable, constant, list, or compound
           θ, the substitution built up so far

   if θ = failure then return failure
   else if x = y then return θ
   else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
   else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
   else if COMPOUND?(x) and COMPOUND?(y) then
       return UNIFY(ARGS[x], ARGS[y], UNIFY(OP[x], OP[y], θ))
   else if LIST?(x) and LIST?(y) then
       return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], θ))
   else return failure
```

The Unification Algorithm

```
function UNIFY-VAR(var, x, θ) returns a substitution
   inputs: var, a variable
           x, any expression
           θ, the substitution built up so far

   if {var/val} ∈ θ then return UNIFY(val, x, θ)
   else if {x/val} ∈ θ then return UNIFY(var, val, θ)
   else if OCCUR-CHECK?(var, x) then return failure
   else return add {var/x} to θ
```

Inference Rules for Quantifiers

- **Universal Elimination:** "∀*v* α |- SUBST({v/g}, α)"

  for any sentence, α, variable, *v, and ground term, g*
- ∀ x Study(x, AI) |- Study(Mary, AI)

- **Existential Elimination:** "∃v α |- SUBST({v/k},a)"

  for any sentence, α, variable, *v, and constant symbol, k,* that doesn't occur elsewhere in the KB (Skolem constant)
- ∃x (Owns(Mary,x) ∧ Cat(x)) |- Owns(Mary,Jusy)∧ Cat(Jusy)

- **Existential Introduction:** "α |- ∃ v SUBST({*g/v*}, α*)"*

  for any sentence, α, variable, *v, that does not occur in* α*,* and ground term, *g, that does occur in* α
- Study(Mary, AI) |- ∃ x Study(x, AI)

Proof Example

1) $\forall$ x,y(Parent(x,y) $\wedge$ Male(x) $\Rightarrow$ Father(x,y))

2) Parent(Tom,John)

3) Male(Tom) Using Universal Elimination from 1)

4) $\forall$ y(Parent(Tom,y) $\wedge$ Male(Tom) $\Rightarrow$ Father(Tom,y))

Using Universal Elimination from 4)

5) Parent(Tom,John) $\wedge$ Male(Tom) $\Rightarrow$ Father(Tom,John)

Using And Introduction from 2) and 3)

6) Parent(Tom,John) $\wedge$ Male(Tom)

Using Modes Ponens from 5) and 6)

7) Father(Tom,John)

Generalized Modus Ponens (GMP)

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta}$$

where θ is a substitution such that for all *i* SUBST(θ, p_i')=SUBST(θ, p_i)

Ex.:

1) $\forall$ x,y(Parent(x,y) $\wedge$ Male(x) $\Rightarrow$ Father(x,y))

2) Parent(Tom,John)

3) Male(Tom)

q={x/Tom, y/John)

4) Father(Tom,John)

Generalized Modus Ponens (GMP)

- In order to Apply generalized Modus Ponens, all sentences in the KB must be in the form of **Horn Clauses:**
- where a clause is a disjunction of literals, because they can be rewritten as disjunctions with at most one non-negated literal.

$\forall v_1, v_2, \ldots, v_n \ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)$ can be expressed as

$\forall \ v_1, v_2, \ldots, v_n \ \neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_n \vee q$

- If we have exactly one definite clause, the sentence is called a **definite clause**
- Quantifiers can be dropped since all variables can beassumed to be universally quantified by default.
- Many sentences can be transformed into Horn clauses, but not all (e.g. $P(x) \vee Q(x)$, and $\neg P(x)$)

Resolution

- **Propositional version.**

$\{a \vee b, \neg b \vee g\} \mid\!- \ a \vee g$ **OR** $\{\neg a \Rightarrow b, b \Rightarrow g\} \mid\!- \ \neg a \Rightarrow g$

- Reasoning by cases **OR** transitivity of implication
- **First-order form**
- For two literals p_k and q_l in two clauses

$p_1 \vee p_2 \vee \ldots \vee p_n$

$q_1 \vee q_2 \vee \ldots \vee q_n$

such that θ =UNIFY(p_k, $\neg q_l$), derives

SUBST(θ, $p_1 \vee p_2 \vee \ldots p_{k-1} \vee p_{k+1} \vee \ldots \vee p_n \vee q_1 \vee q_2 \vee \ldots q_{l-1} \vee q_{l+1} \vee \ldots \vee q_n$)

- For resolution to apply, all sentences must be in **conjunctive normal form,**

Conjunction Normal Form (CNF)

We like to prove:

$KB \models \alpha$

equivalent to : $KB \wedge \neg\alpha$ *unsatifiable*

We first rewrite $KB \wedge \neg\alpha$ into conjunctive normal form (CNF).

A "conjunction of disjunctions"

$(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Clause   Clause   literals

In theory
• Any KB can be converted into CNF.
• In fact, any KB can be converted into CNF-3, i.e. using clauses with at most 3 literals.

Example: Conversion to CNF (PC)

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta)\wedge(\beta \Rightarrow \alpha)$.

   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and double-negation: $\neg(\alpha \vee \beta) = \neg\alpha \wedge \neg\beta$

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

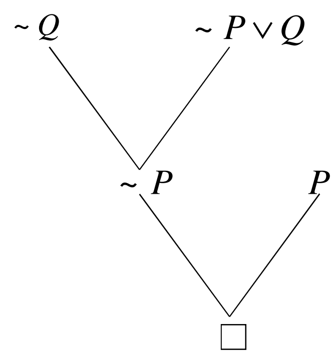
4. Apply distributive law ($\wedge$ over $\vee$) and flatten:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Resolution (PC)

1. P
2. $P \rightarrow Q$ converted to $\sim P \vee Q$
3. $\sim Q$

Draw the resolution tree (actually an inverted tree). Every node is a clausal form and branches are intermediate inference steps.



Resolution Algorithm (PC)

- The resolution algorithm tries to prove: $KB \models \alpha$ *equivalent to* $KB \wedge \neg\alpha$ *unsatisfiable*

- Generate all new sentences from KB and the query.
- One of two things can happen:

1. We find $P \wedge \neg P$ which is unsatisfiable. i.e. we can entail the query.

2. We find no contradiction: there is a model that satisfies the sentence $KB \wedge \neg\alpha$ (non-trivial) and hence we cannot entail the query.

Resolution Algorithm in FOPC

1) Convert sentences in the KB to CNF (clausal form)
2) Take the negation of the proposed query, convert it to CNF, and add it to the KB.
3) Repeatedly apply the resolution rule to derive new clauses.
4) If the empty clause (False) is eventually derived, stop and conclude that the proposed theorem is true.

<u>Procedure</u>:

✓Eliminate implications and biconditionals
✓Move ¬ inward
✓Standardize variables
✓Move quantifiers left
✓Skolemize: replace each existentially quantified variable with a Skolem constant or Skolem function
✓Distribute ∧ over ∨ to convert to conjunctions of clauses
✓Convert clauses to implications if desired for readability
(¬ a ∨ ¬ b ∨ c ∨ d) To   a ∨ b => c ∨ d

Conversion to CNF

- Everyone who loves all animals is loved by someone:

$\forall x([\forall y\ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\ Loves(y,x)])$

1. Eliminate biconditionals and implications

$\forall x([\neg\forall y\ (\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\ Loves(y,x)])$

2. Move ¬ inwards:"$\neg\forall x\ p \equiv \exists x\ \neg p$, $\neg \exists x\ p \equiv \forall x\ \neg p$"

$\forall x\ ([\exists y\ (\neg(\neg Animal(y) \vee Loves(x,y)))] \vee [\exists y\ Loves(y,x)]\ )$

$\forall x\ ([\exists y\ (\neg\neg Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y\ Loves(y,x)]\ )$

$\forall x([\exists y\ (Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y\ Loves(y,x)]\ )$

Conversion to CNF contd.

3. Standardize variables: each quantifier should use a different one

   $\forall x([\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z\ Loves(z,x)])$

4. Skolemize: a more general form of existential instantiation.

   Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

   $\forall x([Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x))$

5. Drop universal quantifiers:

   $[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$

6. Distribute $\vee$ over $\wedge$:

   $[Animal(F(x)) \vee Loves(G(x),x)] \wedge [\neg Loves(x,F(x)) \vee Loves(G(x),x)]$

Resolution in PC

Conjunctive Normal Form (CNF)

conjunction of disjunctions of literals

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

- Resolution inference rule (for CNF):

$$\frac{\ell_i \vee \ldots \vee \ell_k, \qquad m_1 \vee \ldots \vee m_n}{\ell_i \vee \ldots \vee \ell_{s-1} \vee \ell_{s+1} \vee \ldots \vee \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where ℓ_s and m_j are complementary literals.

E.g., $$\frac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$$

Resolution is sound and complete for propositional logic

Resolution in FOL

- Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

  where `Unify`$(\ell_i, \neg m_j) = \theta$.

The two clauses are assumed to be standardized apart so that they share no variables.

- For example,

$$\frac{\neg Rich(x) \vee Unhappy(x) \qquad Rich(Ken)}{Unhappy(Ken)}$$

  with $\theta = \{x/Ken\}$

A More Concise Version

$$\frac{\bigvee_{i \in A} L_i \qquad \bigvee_{i \in B} L_i}{\bigvee_{i \in C} Subst(\theta, L_i)} \qquad \begin{array}{l} Unify(L_j, \neg L_k) \\ j \in A, k \in B \\ C = (A \cup B) \backslash \{j, k\} \end{array}$$

E.g. for A = {1, 2, 7} first clause is $L_1 \vee L_2 \vee L_7$

Empty Clause means False

- Resolution theorem proving ends
  - When the resolved clause has no literals (empty)
- This can only be because:
  - Two **unit clauses** were resolved
    - One was the negation of the other (after substitution)
  - Example: q(X) and ¬q(X) or: p(X) and ¬p(bob)
- Hence if we see the empty clause
  - This was because there was an inconsistency
  - Hence the proof by refutation

Resolution as Search

- **Initial State**: Knowledge base (KB) of axioms and negated theorem in CNF
- **Operators**: Resolution rule picks 2 clauses and adds new clause
- **Goal Test**: Does KB contain the empty clause?

- Search space of KB states