Propositional logic is a weak language

- Hard to identify "individuals." E.g., Mary, 3
- Can’t directly talk about properties of individuals or relations between individuals. E.g., “Bill is tall”
- Generalizations, patterns, regularities can’t easily be represented. E.g., all triangles have 3 sides
- First-Order Logic (abbreviated FOL or FOPC) is expressive enough to concisely represent this kind of situation.
  - FOL adds relations, variables, and quantifiers, e.g.,
    - "Every elephant is gray": ∀x (elephant(x) → gray(x))
    - "There is a white alligator": ∃x (alligator(x) ∧ white(x))

Example

Consider the problem of representing the following information:
- Every person is mortal.
- Confucius is a person.
- Confucius is mortal.
How can these sentences be represented so that we can infer the third sentence from the first two?

Example cont.

- In PL we have to create propositional symbols to stand for all or part of each sentence. For example, we might do:
  - P = "person"; Q = "mortal"; R = "Confucius"
- so the above 3 sentences are represented as:
  - P ⇒ Q; R ⇒ P; R ⇒ Q
- Although the third sentence is entailed by the first two, we needed an explicit symbol, R, to represent an individual, Confucius, who is a member of the classes “person” and “mortal.”
- To represent other individuals we must introduce separate symbols for each one, with means for representing the fact that all individuals who are “people” are also “mortal.”

Problems with the propositional Wumpus hunter

- Lack of variables prevents stating more general rules.
  - E.g., we need a set of similar rules for each cell
- Change of the KB over time is difficult to represent
  - Standard technique is to index facts with the time when they’re true
  - This means we have a separate KB for every time point.

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”
- Examples:
  - 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 …
A BNF for FOL

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

Domain of Discourse

- Constant symbols, which represent individuals in the world
  - 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)

FOL Syntax

- Variable symbols
  - E.g., x, y, foo
- Connectives
  - Same as in PL: not (~), and (^), or (v), implies (=>), if and only if (<=)
- Quantifiers
  - Universal ∀x or (Ax)
  - Existential ∃x or (Ex)

Sentences and WFFs

- A term (denoting a real-world individual) is a constant symbol, a variable symbol, or an n-place function of n terms.
- x and f(x1, ..., xn) are terms, where each xi is a term.
- A term with no variables is a ground term
- An atomic sentence (which has value true or false) is either
  - an n-place predicate of n terms, or, term = term
- A sentence is
  - an atomic sentence
  - ¬P(x) or P(x) => Q(y), P(x) ^ Q(y), P(x) <=> Q(y)
  - If P(x) is a sentence and x is a variable, then (∀x)P(x) and (∃x)P(x) are sentences
- A well-formed formula (wff) is a sentence containing no "free" variables, i.e., all variables are "bound" by universal or existential quantifiers.
  - (∀x)P(x,y) has x bound as a universally quantified variable, but y is free.

Quantifiers

- Universal quantification
  - (∀x)P(x) means that P holds for all values of x in the domain associated with that variable
  - E.g., (∀x) dolphin(x) => mammal(x)
- Existential quantification
  - (∃x)P(x) means that P holds for some value of x in the domain associated with that variable
  - E.g., (∃x) mammal(x) ^ lays-eggs(x)
  - Permits one to make a statement about some object without naming it

What's the problem?
Quantifier Scope

Switching the order of universal quantifiers does not change the meaning:

\((\forall x)(\forall y)P(x,y) \iff (\forall y)(\forall x)P(x,y)\)

Similarly, you can switch the order of existential quantifiers:

\((\exists x)(\exists y)P(x,y) \iff (\exists y)(\exists x)P(x,y)\)

Switching the order of universals and existential 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

We can relate sentences involving \(\forall\) and \(\exists\) using De Morgan’s laws:

\(\neg(\forall x)P(x) \iff (\exists x)\neg P(x)\)
\(\neg(\exists x)P(x) \iff (\forall x)\neg P(x)\)

Translating English to FOL

Every gardener likes the sun.

\((\forall x)\) gardener\((x) \implies \) likes\((x,\)Sun\()\)

You can fool some of the people all of the time.

\((\exists x)\) (person\((x) \land (\forall t)\) can-fool\((x,t)\)\)

You can fool all of the people some of the time.

\((\forall x)\) (person\((x) \implies (\exists t)\) time\((t) \land \) can-fool\((x,t)\)\)

Notational differences

Different symbols for and, or, not, implies, ...

- \(\land\) \(\lor\) \(\neg\) \(\implies\)
- \(p \implies (q \land r)\)
- \(p \land (q \lor r)\)
- etc

Prolog

\(\text{cat}(X) :- \text{furry}(X), \text{meows}(X), \text{has}(X,\text{claws})\)

Lispy notations

(forall ?x (implies (and (furry ?x) (has ?x claws)))
(meows ?x)
(has ?x claws))
(cat ?x)))

Inference in first-order logic

- Inference rules
- Forward chaining
- Backward chaining
- Resolution
  - Unification
  - Proofs
  - Clausal form
  - Resolution as search

Tarski’s World

http://www-csli.stanford.edu/hp/Tarski1.html
Inference rules for FOL

- Inference rules for propositional logic apply to FOL as well
  - Modus Ponens, etc.
- New (sound) inference rules for use with quantifiers:
  - Universal elimination
  - Existential introduction
  - Existential elimination
  - Generalized Modus Ponens (GMP)

Universal elimination

- If (∀x) P(x) is true, then P(c) is true, where c is any constant in the domain of x
- Example:
  - (∀x) eats(Ziggy, x)
  - 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 introduction

- If P(c) is true, then (∃x) P(x) is inferred.
- Example
  - eats(Ziggy, IceCream)
  - (∃x) eats(Ziggy, x)
- All instances of the given constant symbol are replaced by the new variable symbol
- Note that the variable symbol cannot already exist anywhere in the expression

Existential elimination

- From (∃x) P(x) infer P(c)
- Example:
  - (∃x) eats(Ziggy, x)
  - eats(Ziggy, Stuff)
- Note that the variable is replaced by a brand-new constant not occurring in this or any other sentence in the KB
- Also known as skolemization; constant is a skolem constant
- In other words, we don't want to accidentally draw other inferences about it by introducing the constant
- Convenient to use this to reason about the unknown object, rather than constantly manipulating the existential quantifier

Generalized Modus Ponens (GMP)

- Apply modus ponens reasoning to generalized rules
- Combines And-Introduction, Universal-Elimination, and Modus Ponens
- E.g., from P(c) and Q(c) and (∀x)(P(x) ^ Q(x)) => R(x) derive R(c)
- General case:
  - Given
    - atomic sentences P₁, P₂, ..., Pₙ
    - implication sentence Q₀ ^ Q₁ ^ ... ^ Qₙ => R
  - Derive new sentence: subst(θ, R)
- Substitutions
  - subst(θ, α) denotes the result of applying a set of substitutions defined by θ to the sentence α
  - A substitution list θ = {v₁/t₁, v₂/t₂, ..., vᵣ/ᵣ} means to replace all occurrences of variable symbol vᵢ by term tᵢ
  - Substitutions are made in left-to-right order in the list
  - subst({x/IceCream, y/Ziggy}, eats(y,x)) = eats(Ziggy, IceCream)

Automated inference for FOL

- Automated inference using FOL is harder than PL
  - Variables can potentially take on an infinite number of possible values from their domains
  - Hence there are potentially an infinite number of ways to apply the Universal-Elimination rule of inference
  - Gödel's Completeness Theorem says that 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
Completeness of some inference techniques

- **Truth Tabling**
  - is not complete for FOL because truth table size may be infinite

- **Generalized Modus Ponens**
  - is not complete for FOL
  - Generalized Modus Ponens is complete for KBs containing only Horn clauses

- **Resolution Refutation**
  - is complete for FOL

Horn clauses (again)

- A Horn clause is a sentence of the form:
  \( (\forall x) P_1(x) \land P_2(x) \land \ldots \land P_n(x) \Rightarrow Q(x) \)
  where
  - there are 0 or more \( P_i \)s and 0 or 1 \( Q \)
  - the \( P_i \)s and \( Q \) are positive (i.e., non-negated) literals
- Equivalently: \( P_1(x) \lor P_2(x) \ldots \lor P_n(x) \) where the \( P_i \)s are all atomic and at most one of them is positive
- Prolog is based on Horn clauses
- Horn clauses represent a subset of the set of sentences representable in FOL

Horn clauses II

- **Special cases**
  - \( P_1 \land P_2 \land \ldots \land P_n \Rightarrow Q \)
  - \( P_1 \land P_2 \land \ldots \land P_n \Rightarrow \text{false} \)
  - true \( \Rightarrow Q \)
- **These are not Horn clauses:**
  - \( p(a) \lor q(a) \)
  - \( P \land Q \Rightarrow R \lor S \)

Unification

- **Unification** is a "pattern-matching" procedure
  - Takes two atomic sentences as input
  - Returns "Failure" if they do not match and a substitution list, \( \theta \), if they do
  - That is, \( \text{unify}(p,q) = \theta \) means \( \text{subst}(\theta, p) = \text{subst}(\theta, q) \) for two atomic sentences, \( p \) and \( q \)
- \( \theta \) is called the **most general unifier** (mgu)
- All variables in the given two literals are implicitly universally quantified
- To make literals match, replace (universally quantified) variables by terms

Unification algorithm

procedure unify(p, q, \( \theta \))
  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 \( \theta \) (success!)
  Let \( r \) and \( s \) be the terms in \( p \) and \( q \), respectively.
  where disagreement first occurs
  If variable(r) then {
    Let \( \theta = \text{union}(\theta, \{ r/s \}) \)
    Recurse and return unify(subst(\( \theta \), \( p \)), subst(\( \theta \), \( q \)), \( \theta \))
  } else if variable(s) then {
    Let \( \theta = \text{union}(\theta, \{ s/r \}) \)
    Recurse and return unify(subst(\( \theta \), \( p \)), subst(\( \theta \), \( q \)), \( \theta \))
  } else return "Failure"
end

Unification: Remarks

- **Unify** is a linear-time algorithm that returns the most general unifier (mgu), i.e., the shortest-length substitution list that makes the two literals match.
- In general, there is not a unique minimum-length substitution list, but unify returns one of minimum length.
- A variable can never be replaced by a term containing that variable
- Example: \( x(f(x)) \) is illegal.
- This "occurs check" should be done in the above pseudo-code before making the recursive calls.
Unification examples

Example:
- parents(x, father(x), mother(Bill))
- parents(Bill, father(Bill), y)
- {x/Bill, y/mother(Bill)}

Example:
- parents(x, father(x), mother(Bill))
- parents(Bill, father(y), z)
- {x/Bill, y/Bill, z/mother(Bill)}

Example:
- parents(x, father(x), mother(Jane))
- parents(Bill, father(y), mother(y))
- Failure

Forward chaining in FOL

Proofs start with the given axioms/premises in KB, deriving new sentences using GMP until the goal/query sentence is derived

This defines a forward-chaining inference procedure because it moves “forward” from the KB to the goal

Inference using GMP is complete for KBs containing only Horn clauses

Forward Chaining Example

KB:
1. If allergies(X) then sneeze(X)
2. If cat(Y) and allergic-to-cats(X) then allergies(X)
3. cat(Felix)
4. allergic-to-cats(Lise)

Conclude:
- sneeze(Lise)

Backward Chaining Example

KB:
1. If allergies(X) then sneeze(X)
2. If cat(Y) and allergic-to-cats(X) then allergies(X)
3. cat(Felix)
4. allergic-to-cats(Lise)

Goal:
- sneeze(Lise)

Backward chaining in FOL

Backward-chaining deduction using GMP is complete for KBs containing only Horn clauses

Proofs start with the goal query, find implications that would allow you to prove it, and then prove each of the antecedents in the implication, continuing to work “backwards” until you arrive at the axioms, which we know are true

Completeness of GMP for HC

GMP (using forward or backward chaining) is complete for KBs containing only Horn clauses

It is not complete for simple KBs that contain non-Horn clauses

The following entail that S(A) is true:

- (∀x) P(x) => Q(x)
- (∀x) ~P(x) => R(x)
- (∀x) Q(x) => S(x)
- (∀x) R(x) => S(x)

If we want to conclude S(A), with GMP we cannot, since the second one is not a Horn form

It is equivalent to P(x) ∨ R(x)
Resolution
Resolution is a **sound** and **complete** inference procedure for FOL

**Resolution Rule for PL:**
- \( P_1 \lor P_2 \lor \ldots \lor P_n \)
- \( \neg P_1 \lor Q_2 \lor \ldots \lor Q_m \)
- Resolvent: \( P_2 \lor \ldots \lor P_n \lor Q_2 \lor \ldots \lor Q_m \)

**Examples**
- \( P \) and \( \neg P \), derive \( Q \) (Modus Ponens)
- \( (\neg P \lor Q) \) and \( (\neg Q \lor R) \), derive \( \neg P \lor R \)
- \( P \) and \( \neg P \), derive False [contradiction!]
- \( (P \lor Q) \) and \( (\neg P \lor \neg Q) \), derive True

FOL resolution

**Given sentences**
- \( P_1 \lor \ldots \lor P_n \)
- \( Q_1 \lor \ldots \lor Q_m \)

where each \( P_i \) and \( Q_i \) is a literal, i.e., a positive or negated predicate symbol with its terms, if \( P_j \) and \( \neg Q_k \) unify with substitution list \( \theta \), then derive the resolvent sentence:
  \[ \text{subst}(\theta, P_1 \lor \ldots \lor P_{j-1} \lor P_{j+1} \ldots P_n \lor Q_1 \lor \ldots Q_{k-1} \lor Q_{k+1} \lor \ldots \lor Q_m) \]

**Examples**
- From clause \( P(x, f(a)) \lor P(x, f(y)) \lor Q(y) \)
- and clause \( \neg P(z, f(a)) \lor \neg Q(z) \),
  derive resolvent clause \( P(z, f(y)) \lor Q(y) \lor \neg Q(z) \)
  using \( \theta = \{x/z\} \)

Resolution refutation proofs

**Given a consistent set of axioms KB and goal sentence \( Q \), show that \( KB \models Q \)

**Proof by contradiction:** Add \( \neg Q \) to KB and try to prove false,
  i.e., \( (KB \models Q) \leftrightarrow (KB \lor \neg Q \models \text{False}) \)

Resolution can establish that a given sentence \( Q \) is entailed by KB, but can’t (in general) be used to generate all logical consequences of a set sentences
- Also, it cannot be used to prove that \( Q \) is **not entailed** by KB.
- Resolution won’t always give an answer since entailment is only semidecidable
  - And you can’t just run two proofs in parallel, one trying to prove \( Q \) and the other trying to prove \( \neg Q \) since KB might not entail either one

Resolution – issues
Resolution is only applicable to sentences in clausal form, e.g.
- \( P_1 \lor P_2 \lor \ldots \lor P_n \)
  where \( P_i \)s are negated or non-negated atomic predicates

**Issues:**
- Can we convert every FOL sentence into this form?
  - Yes - as we will see shortly
- How to pick which pair of sentences to resolve?
  - Determines the “search” strategy of the prover
  - How to pick which pair of literals, one from each sentence, to unify?
  - Again, part of the search strategy

Converting FOL sentences to clausal form

1. Eliminate all \( <\leftrightarrow \) connectives
   \( (P \leftrightarrow Q) \Rightarrow (P \Rightarrow Q) \land (Q \Rightarrow P) \)
2. Eliminate all \( \Rightarrow \) connectives
   \( (P \Rightarrow Q) \Rightarrow (\neg P \lor Q) \)
3. Reduce the scope of each negation symbol to a single predicate
   \( \neg \neg P \Rightarrow P \)
   \( (\neg P \lor Q) \Rightarrow (\neg P \lor Q) \)
   \( (\neg P \lor Q) \Rightarrow \neg P \lor Q \)
   \( (\neg P \lor Q) \Rightarrow \neg P \lor Q \)
4. Standardize variables: rename all variables so that each quantifier has its own unique variable name
Converting sentences to clausal form

5. Eliminate existential quantification by introducing Skolem constants/functions

\[(\exists x)P(x) \iff P(c)\]

c is a Skolem constant (a brand-new constant symbol that is not used in any other sentence)

\[(\forall x)((\exists y)P(x,y)) \iff (\forall x)(P(x, f(x)))\]

since \(\exists y\) is within the 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 brand-new function name not occurring in any other sentence in the KB.

E.g., \((\forall x)(\exists y)\text{loves}(x,y) \iff (\forall x)\text{loves}(x,f(x))\)

In this case, \(f(x)\) specifies the person that \(x\) loves

Converting FOL sentences to clausal form

6. Remove universal quantifiers by (1) moving them all to the left end; (2) making the scope of each the entire sentence; and (3) dropping the “prefix” part

Ex: \((\forall x)P(x) \iff P(x)\)

7. Distribute \(\lor\) over \(\land\)

\[(P \land Q) \lor R \iff (P \lor R) \land (Q \lor R)\]

8. Split conjuncts into a separate clauses

9. Standardize variables so each clause contains only variable names that do not occur in any other clause

Example

7. Convert to conjunction of disjunctions

\[(-P(x) \lor -P(y) \lor P(f(x,y))) \land (-P(x) \lor Q(x,g(x))) \land (-P(w) \lor -P(g(x)))\]

8. Create separate clauses

\[\neg P(x) \lor \neg P(y) \lor P(f(x,y))\]

\[\neg P(x) \lor Q(x,g(x))\]

\[\neg P(w) \lor \neg P(g(x))\]

9. Standardize variables

\[\neg P(x) \lor \neg P(y) \lor P(f(x,y))\]

\[\neg P(z) \lor Q(z,g(z))\]

\[\neg P(w) \lor \neg P(g(x))\]

Example proof

Did Curiosity kill the cat?

1. Dog(spike)
2. Owns(Jack,spike)
3. \neg\text{Dog}(y) \lor \neg\text{Owns}(x,y) \lor \text{AnimalLover}(x)
4. \neg\text{AnimalLover}(x) \lor \neg\text{Animal}(y) \lor \neg\text{Kills}(x,y)
5. \text{Kill}(Jack,Tuna) \lor \text{Kill}(Curiosity,Tuna)
6. \text{Cat}(Tuna)
7. \neg\text{Cat}(x) \lor \text{Animal}(x)
Example: Did Curiosity kill the cat?

1. Dog(spike)
2. Owns(Jack, spike)
3. ~Dog(y) v ~Owns(x, y) v AnimalLover(x)
4. ~AnimalLover(x1) v ~Animal(y1) v ~Kills(x1, y1)
5. Kills(Jack, Tuna) v Kills(Curiosity, Tuna)
6. Cat(Tuna)
7. ~Cat(x2) v Animal(x2)
8. ~Kills(Curiosity, Tuna) negated goal
9. Kills(Jack, Tuna) 5,8
10. ~AnimalLover(Jack) v ~Animal(Tuna) 9,4 x1/Jack, y1/Tuna
11. ~Dog(y) v ~Owns(x, y) v ~Animal(Tuna) 10,3 x/Jack
12. ~Owns(Jack, spike) v ~Animal(Tuna) 11,1
13. ~Animal(Tuna) 12,2
14. ~Cat(Tuna) 12,7 x2/Tuna
15. False 14,6

FOL in the Realworld

- Simon's prediction 40 years ago: In the next 10 years, a computer will prove a major mathematical theorem.
- Achieved last year
- Using extended Resolution Theorem prover, scientists at Argonne National Labs recently proved the first major open theorem by a computer
- TP used general heuristics such as preference for proving simple statements and using resolution steps that worked in other cases.
- After 8 days running on workstation, were able to find proof
- Computers had been used in the past to solve theorems, but not ones that people had been unable to solve. Exception 4-coloring problem. However, in that case computer enumerated all possibilities.

FOL Summary

- Syntax - terms, WFF, quantifiers
- New Inference rules for quantifiers
- Unification
- Horn clauses - FC, BC
- Resolution Refutation
- Converting to clausal form