First Order Logic
(Predicate Calculus)

CPS 270
Ronald Parr

Limitations of Propositional Logic

• Suppose you want to say: All humans are mortal
• For $\sim 6B$ people, you would need $\sim 6B$ propositions
• Suppose you want to stay that (at least) one person has perfect pitch
• You would need a disjunction of $\sim 6B$ propositions

• There has to be a better way...
First Order Logic

- Propositional logic is very restrictive
  - Can’t make global statements about objects in the world
  - Workarounds tends to have very large KBs
- First order logic is more expressive
  - Relations, quantification, functions
  - but... inference is trickier

First Order Syntax

- Sentences
- Atomic sentence predicate(term)
- Terms – functions, constants, variables
- Connectives
- Quantifiers
- Constants
- Variables
Relations

• Assert relationships between objects

• Examples
  – Loves(Harry, Sally)
  – Between(Canada, US, Mexico)

• Semantics
  – Object and predicate names are mnemonic only
  – Interpretation is imposed from outside
  – Often we imply the “expected” interpretation of predicates and objects with suggestive names

Functions

• Functions are special cases of relations

• Suppose R(x_1,x_2,...,x_n,y) is such that for every value of x_1,x_2,...,x_n there is a unique y

• Then R(x_1,x_2,...,x_n) can be used as a shorthand for y
  – Crossed(Right_leg_of(Ron), Left_leg_of(Ron))

• Remember that the object identified by a function depends upon the interpretation
Quantification

• For all objects in the world...

\[ \forall x \text{happy}(x) \]

• For at least one object in the world...

\[ \exists x \text{happy}(x) \]

Examples

• Everybody loves somebody

\[ \forall x \exists y \text{Loves}(x, y) \]

• Everybody loves everybody

\[ \forall x \forall y \text{Loves}(x, y) \]

• Everybody loves Raymond

\[ \forall x \text{Loves}(x, \text{Raymond}) \]

• Raymond loves everybody

\[ \forall x \text{Loves}(\text{Raymond}, x) \]
Equality

- Equality states that two objects are the same
  - Son_of(Barbara) = Ron
- Equality is a special relation that holds whenever two objects are the same
- We can imagine that every interpretation comes with its own identity relation
  - Identical(object27, object58)

What’s Missing?

- There are many extensions to first order logic
- Higher order logics permit quantification over predicates:
  \[ \forall x, y (x = y) \iff (\forall p (p(x) \iff p(y))) \]
- Uniqueness
- Extensions typically replace a potentially long series of conjuncts with a single expression
Inference

• All rules of inference for propositional logic apply to first order logic
• We need extra rules to handle substitution for quantified variables

\[
\text{SUBST(} \{x/\text{Harry}, y/\text{Sally}, \text{Loves}(x,y)\} = \text{Loves(Harry, Sally)}
\]

Inference Rules

• Universal Elimination

\[
\forall v : \alpha(v) \\
\frac{}{\text{SUBST(}\{v/g\}, \alpha(v))}
\]

• How to read this:
  – We have a universally quantified variable v in \(\alpha\)
  – Can substitute any g for v and \(\alpha\) will still be true
Inference Rules

• Existential Elimination

\[ \exists v : \alpha(v) \quad \frac{}{\text{SUBST}(\{v/k\}, \alpha(v))} \]

• How to read this:
  – We have a universally quantified variable \( v \) in a
  – Can substitute any \( k \) for \( v \) and \( \alpha \) will still be true
  – IMPORTANT: \( k \) must be a previously unused constant (skolem constant). Why is this OK?

Skolemization within Quantifiers

• Skolemizing w/in universal quantifier is tricky
• Everybody loves somebody

\[ \forall x \exists y : \text{loves}(x, y) \]

• With Skolem constants, becomes:

\[ \forall x : \text{loves}(x, \text{object34752}) \]

• Why is this wrong?
• Need to use skolem functions:

\[ \forall x : \text{loves}(x, \text{personlovedby}(x)) \]
Inference Rules

• Existential Introduction

\[ \alpha(g) \]

\[ \text{SUBST}(\{v/g\}, \exists v : \alpha(v)) \]

• How to read this:
  – We know that the sentence \( \alpha \) is true
  – Can substitute variable \( v \) for any constant \( g \) in \( \alpha \) and (w/existential quantifier) and \( \alpha \) will still be true
  – Why is this OK?

Generalized Modus Ponens Example

• If has_US_birth_certificate(X) then natural_US_citizen(X)

• has_US_birth_certificate(Obama)

• Conclude \( \text{SUBST}([Obama/X], \text{natural_US_citizen(X)}) \)

• i.e., natural_US_citizen(Obama)
Generalized Modus Ponens

\[ \text{SUBST}(\theta,p_i') = \text{SUBST}(\theta,p_i) \forall i \]

\[ p_1', p_2', \ldots, p_n', (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \]

\[ \text{SUBST}(\theta,q) \]

• How to read this:
  – We have an implication which implies q
  – Any consistent substitution of variables on the LHS must yield a valid conclusion on the RHS

Unification

• Substitution is a non-trivial matter
• We need an algorithm unify:

\[ \text{Unify}(p,q) = \theta : \text{Subst}(\theta,p) = \text{Subst}(\theta,q) \]

• Important: Unification replaces variables:

  \[ \exists x \text{Loves}(John,x) \]
  \[ \exists x \text{Hates}(John,x) \]

• Are these the same x?
Unification Example

∀xKnows(John,x) ⇒ Loves(John,x)
Knows(John,Jane)
∀yKnows(y,Leonid)
∀yKnows(y,Mother(y))
∀xKnows(x,Elizabeth)

Note: All unquantified variables are assumed universal from here on.

Unify(Knows(John,x),Knows(John,Jane)) = \{x / Jane\}
Unify(Knows(John,x),Knows(y,Leonid)) = \{x / Leonid,y / John\}
Unify(Knows(John,x),Knows(y,Mother(y))) = \{y / John,x / Mother(John)\}
Unify(Knows(John,x),Knows(x,Elizabeth)) = \{x_1 / Elizabeth,x_2 / John\}

Most General Unifier

• Unify(Knows(John,x),Knows(y,z))
  – \{y/John,x/z\}
  – \{y/John,x/z,w/Freda\}
  – \{y/John,x/John,z/John\}

• When in doubt, we should always return the most general unifier (MGU)
  – MGU makes least commitment about binding variables to constants
Proof Procedures

- Suppose we have a knowledge base: KB
- We want to prove q
- Forward Chaining
  - Like search: Keep proving new things and adding them to the KB until we are able to prove q
- Backward Chaining
  - Find $p_1...p_n$ s.t. knowing $p_1...p_n$ would prove q
  - Recursively try to prove $p_1...p_n$

Forward Chaining Example

\[ \forall x \text{Knows}(John,x) \Rightarrow \text{Loves}(John,x) \]
\[ \text{Knows}(John, Jane) \]
\[ \forall y \text{Knows}(y, Leonid) \]
\[ \forall y \text{Knows}(y, \text{Mother}(y)) \]
\[ \forall x \text{Knows}(x, Elizabeth) \]

- Loves(John, Jane)
- Knows(John, Leonid)
- Loves(John, Leonid)
- Knows(John, Mother(John))
- Loves(John, Mother(John))
- Knows(John, Elizabeth)
- Loves(John, Elizabeth)
Forward Chaining

Procedure Forward_Chain(KB,p)
If p is in KB then return
Add p to KB
For each (p_1 \land \ldots \land p_n \Rightarrow q) in KB such that for some i,
Unify(p_i,p)=q succeeds do
Find_And_Infer(KB,[p_1,\ldots,p_{i-1},p_{i+1},\ldots,p_n],q,q)
end

Procedure Find_and_Infer(KB,premises,conclusion,q)
If premises=[] then
Forward_Chain(KB,Subst(q,conclusion))
Else for each p’ in KB such that
Unify(p’,Subst(q,Head(premises)))=q_2 do
Find_And_Infer(KB,Tail(premises),conclusion,[q,q_2])
end

A Note About Forward Chaining

• As presented, forward chaining seems undirected
• Can view forward chaining as a search problem
• Can apply heuristics to guide this search
• If you’re trying to prove that Barack Obama is a natural born citizen, should you should start by proving that square127 is also a rectangle???
• Interesting AI history: AM/Eurisko controversy
  – Doug Lenat introduced what was essentially a forward chaining system for coming up with interesting math concepts
  – Claimed to (re)discover many interesting concepts using only some simple heuristics
  – Methodology sharply criticized due to opacity (see Ritchie and Hanna 1984 and response from Lenat and Brown 1984)
Backward Chaining Example

∀x Knows(John, x) ⇒ Loves(John, x)
Knows(John, Jane)
∀y Knows(y, Leonid)
∀y Knows(y, Mother(y))
∀x Knows(x, Elizabeth)

- Goal: Loves(John, Jane)?
- Subgoal: Knows(John, Jane)

Backward Chaining

Function Back_Chain(KB, q)
    Back_Chain_List(KB, [q], {})

Function Back_Chain_List(KB, qlist, q)
If qlist=[] then return q
q<-head(qlist)
For each q_i in KB such that q_i<-Unify(q,q_i') succeeds do
    Answers <- Answers + [q, q_i]
For each (p_1^...^p_n=>q_i) in KB: q_i<-Unify(q,q_i') succeeds do
    Answers<- Answers +
        Back_Chain_List(KB, Subst(q_i, [p_1,...,p_n]), [q, q_i])
return union of Back_Chain_List(KB, Tail(qlist), q) for each q in answers
Completeness

\[ \forall x P(X) \Rightarrow Q(x) \]
\[ \forall x \neg P(X) \Rightarrow R(x) \]
\[ \forall x Q(x) \Rightarrow S(x) \]
\[ \forall x R(x) \Rightarrow S(x) \]
\[ S(A) \]

• Problem: Generalized Modus Ponens not complete
• Forward/Backward chaining rely upon generalized MP
• Goal: A sound and complete inference procedure for first order logic

Generalized Resolution

\[ \theta = \text{Unify}(p_j, \neg q_k) \]

\[ (p_1 \lor \ldots \lor p_j \ldots \lor p_m) \lor (q_1 \lor \ldots \lor q_k \ldots \lor q_n) \]

\[ \text{SUBST}(\theta, (p_1 \lor \ldots \lor p_{j-1} \lor p_{j+1} \ldots \lor p_m \lor q_1 \lor \ldots \lor q_{k-1} \lor q_{k+1} \ldots \lor q_n)) \]

• If the same term appears in both positive and negative form in two disjunctions, they cancel out when disjunctions are combined
Generalized Resolution Example

$$(\neg P(x) \lor Q(x))$$
$$(P(x) \lor R(x))$$
$$(\neg Q(x) \lor S(x))$$
$$(\neg R(x) \lor S(x))$$
$$S(A)$$

Example on board/tablet...

Resolution Properties

- Proof by refutation (asserting negation and resolving to nil) is sound and complete
  (NB: We did not do this in the previous example)
- Resolution is not complete in a generative sense, only in a testing sense
- This is only part of the job
- To use resolution, we must convert everything to a canonical form, i.e., all sentences must be disjunctions with only implicit universal quantification and existential quantification replaced with skolemization
Canonical Form

- Eliminate Implications
- Move negation inwards
- Standardize (apart) variables
- Move quantifiers Left
- Skolemize
- Drop universal quantifiers
- Distribute AND over OR
- Flatten nested conjunctions and disjunctions

Computational Properties

- We can enumerate the set of all proofs
- We can check if a proof is valid
- First order logic is complete (Gödel)

- What if no valid proof exists?
- Inference in first order logic is semi-decidable
- Compare with halting problem (halting problem is semi-decidable)

- As with propositional logic, horn clauses are an important special case. More about this when we discuss prolog in a future lecture.
Gödel’s Incompleteness Result

• Gödel’s incompleteness result is, perhaps, better known
• Incompleteness applies to logical/mathematical systems rich enough to contain numbers and math
  – Need a way of enumerating all valid proofs within the system
  – Need a way of referring to proofs by number
• Construct a Gödel sentence:
  – S: For all i, i is not the number of a proof of the sentence j
  – (Equivalent to saying, there does not exist a proof of sentence j)
  – Suppose sentence S is sentence j
    • If S is false, then we have a contradiction
    • If S is true, then we can’t have a proof of it

Diagonalization

• Incompleteness can be seen as an instance of diagonalization:
  – Define a set
    (Rationals, TMs that halt, theorems that are provable)
  – Use rules of the system to create an impossible object

• Example: Proof that reals are not enumerable
  (i.e., not countable and therefore larger than the rationals)
Countability of Rationals

\[ X = \frac{n_0 \times 2^0 + n_1 \times 2^1 + n_2 \times 2^2 \ldots}{d_0 \times 2^0 + d_1 \times 2^1 + d_2 \times 2^2 \ldots} \]

<table>
<thead>
<tr>
<th>Label</th>
<th>( n_0 )</th>
<th>( d_0 )</th>
<th>( n_1 )</th>
<th>( d_1 )</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>\ldots</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>\ldots</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>\ldots</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>\ldots</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
</tbody>
</table>

Uncountability of Reals

- **Given:**

<table>
<thead>
<tr>
<th>Label</th>
<th>( n_0 )</th>
<th>( d_0 )</th>
<th>( n_1 )</th>
<th>( d_1 )</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>\ldots</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>\ldots</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>\ldots</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>\ldots</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
</tbody>
</table>

- **Construct:**

<table>
<thead>
<tr>
<th>Label</th>
<th>( n_0 )</th>
<th>( d_0 )</th>
<th>( n_1 )</th>
<th>( d_1 )</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>\ldots</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>\ldots</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>\ldots</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>\ldots</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
</tbody>
</table>
Implications of all this

• Sophomoric interpretation: AI is impossible/implausible because there will always be true things that cannot be discovered by logic

• A bit of reality:
  – Incompleteness talks about a system’s ability to prove things about itself
  – For any given system, it may be possible to prove things by talking about the system in a more expressive language
  – Relationship of the unprovable to intelligence is murky at best: Are the things you can’t justify the things that make you intelligent?
  – Not clear that anything interesting is unprovable in a practical sense (though plenty of interesting things remain unproven)

First Order Logic Conclusions

• First order logic adds relations and quantification to predicate logic
• Inference in first order logic is, essentially, a generalization of inference in predicate logic
  – Resolution is sound and complete
  – Use of resolution requires:
    • Conversion to canonical form
    • Proof by refutation
• In general, inference is first order logic is semi-decidable
• FOL + basic math is no longer complete