Inference in First Order Logic

CS 540-2
Louis Oliphant
oliphant@cs.wisc.edu

Remembering Inference

- Entailment – Given some Knowledge what must logically follow

$$\alpha \models \beta$$

- Inference is the mechanism by which you can show what is entailed

$$\alpha \vdash_i \beta$$

i – the inference mechanism

- We would like Inference mechanisms that are:
  - Sound
  - Complete

Methods of Inference in FOL

- There are two main methods of inference in FOL:
  - convert FOL sentences to propositional sentences
  - Resolution in FOL
- Additional methods exist for subsets of FOL
  - First Order Definite Clauses and Datalog
  - Forward Chaining and Backward Chaining

Reducing to Propositional Logic

$$\forall x \; King(x) \wedge Greedy(x) \Rightarrow Evil(x)$$

- Universal Instantiation – substitute in all possible constants from the knowledge base for the variable

$$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$$
$$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$$
$$King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$$
...

- Then treat each predicate with its arguments as a unique symbol in propositional logic.

Reducing to Propositional Logic

$$\exists x\ \text{Crown}(x) \wedge \text{OnHead}(x,\text{John})$$

- Existential Instantiation – substitute in a new constant not in the knowledge base for the variable

$$\text{Crown}(C_1) \wedge \text{OnHead}(C_1,\text{John})$$

- Again, treat each predicate with its arguments as a unique symbol in propositional logic.

Reducing to Propositional Logic

- Universal Instantiation – substitute in all possible constants from the knowledge base for the variable

$$\frac{\forall v\ \alpha}{\text{subst}(\{v/g\}, \alpha)^*}$$

*for any variable v and ground term g.

- Existential Instantiation – substitute in a new constant not in the knowledge base for the variable

$$\frac{\exists v\ \alpha}{\text{subst}(\{v/k\}, \alpha\}^*}$$

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

Reducing to Propositional Logic

- Using Universal Instantiation and Existential Instantiation you can convert all FOL sentences into a set of Propositional sentences
- Then use standard Propositional reasoning methods to decide if a query is true or false.

Reducing to Propositional Logic

- Let's Try it. Here are the sentences in the Knowledge base:

$$\forall x\ \text{Man}(x) \Rightarrow \text{Mortal}(x)$$
$$\text{Man}(\text{Socrates})$$

- And Here is my Query:

$$\text{Mortal}(\text{Socrates})$$

- Oh, and one other point. The Knowledge base contains the function Father(x). What is the set of Propositional sentences?

Problems with converting to Propositional Logic

- Universal instantiation creates a huge number of sentences in propositional logic.
  - Each term in the knowledge base needs to be substituted into sentences in each possible way
    How many ways to substitute 3 constants into this predicate?

    $\forall x,y\ Pred(x,y)$

- Universal instantiation may create an infinite number of sentences in propositional logic if there are any functions in the knowledge base
  Father(John), Father(Father(John)), ...

Reducing to Propositional Logic

Theorem: Herbrand (1930). If a sentence α is entailed by a FOL KB, it is entailed by a finite subset of the propositionalized KB

Idea: For $n = 0$ to ∞ do
create a propositional KB by instantiating with depth-n terms
see if α is entailed by this KB

Problem: works if α is entailed, infinitely loops if α is not entailed

Theorem: Turing (1936), Church (1936) Entailment for FOL is semidecidable (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.)

So let's try reasoning in FOL directly.

Unification

Standardize Apart: Change the name of variables so sentences don't have any in common

- To reason in FOL directly we need to be able to decide if there is some setting for the variables that would make two sentences identical:

$\forall x$ Knows(John,x) -- Knows(John,Jane) – {x/Jane}
$\forall x$ Knows(John,x) -- $\forall y$ Knows(y,Father(Jane)) – {x/Father(Jane), y/John}
$\forall x$ Knows(John,x) -- $\forall x$ Knows(x,Father(Jane)) – ????
$\forall x$ Knows(John,x) -- $\forall y$ Knows(John,y) –

- Unification is the process of generating a substitution (if one exists) that would make two sentences look identical.
- We want the Most General Unifier
  (MGU is unique up to renaming of variables for any two sentences)

Unification

- To reason in FOL directly we need to be able to decide if there is some setting for the variables that would make two sentences identical:

$\forall x$ Knows(John,x) -- Knows(John,Jane) – {x/Jane}
$\forall x$ Knows(John,x) -- $\forall y$ Knows(y,Father(Jane)) – {x/Father(Jane), y/John}
$\forall x$ Knows(John,x) -- $\forall y$ Knows(y,Father(Jane)) – {x/Father(Jane), y/John}
$\forall x$ Knows(John,x) -- $\forall y$ Knows(John,y) – {x/Jane, y/Jane} or {y/x} or {x/Father(Jane), y/Father(Jane)}

- Unification is the process of generating a substitution (if one exists) that would make two sentences look identical.
- We want the Most General Unifier
  (MGU is unique up to renaming of variables for any two sentences)

Unification Algorithm

```
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 COMP
      return U
  else if LIST?
      return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], θ))
  else return failure
```

What if you have to unify(a,F(a))? -- fail
This makes algorithm $O(n^2)$

```
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
```

Resolution: brief summary

- 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 $\texttt{Unify}(\ell_i, \neg m_j) = \theta$.

- 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 θ = {x/Ken}

- Use resolution on CNF(KB ∧ ¬α); complete for FOL
- So First convert to Conjunctive Normal Form

Conversion to CNF in FOL

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

4. Skolemize: 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 ∨ over ∧ :
   $[Animal(F(x)) \vee Loves(G(x),x)] \wedge [\neg Loves(x,F(x)) \vee Loves(G(x),x)]$

Conversion to CNF contd.

You try – "Anyone who kills all the animals is loved by no one."

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

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

1. Eliminate biconditionals and implications
   $a \Leftrightarrow b \equiv a \Rightarrow b \wedge b \Rightarrow a$ $\quad a \Rightarrow b \equiv \neg a \vee b$
2. Move $\neg$ inwards:
   $\neg \forall x\ p \equiv \exists x\ \neg p$ $\quad \neg\ \exists x\ p \equiv \forall x\ \neg p$
   $\neg(a \wedge b) \equiv \neg a \vee \neg b$ $\quad \neg(a \vee b) \equiv \neg a \wedge \neg b$
3. Standardize variables: each quantifier should use different one
4. Skolemize: general form of existential instantiation.
   Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables
5. Drop universal quantifiers
6. Distribute $\vee$ over $\wedge$
   $(a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$

Example Knowledge Base

Everyone who loves all animals is loved by someone.

Anyone who kills an animal is loved by no one.

Jack loves all animals.

Either Jack or Curiosity killed the cat, who is named Tuna.

Did Curiosity kill the cat?

- Convert to FOL sentences
- Convert to CNF
- Use Resolution by Refutation to prove $(KB \wedge \neg\alpha)$ is false

Convert to FOL sentences

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

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

$\forall x\ Animal(x) \Rightarrow Loves(Jack,x)$

$Kills(Jack,Tuna) \vee Kills(Curiosity,Tuna)$

$Cat(Tuna)$

$\forall x\ Cat(x) \Rightarrow Animal(x)$

$\neg Kills(Curiosity,Tuna)$

Convert FOL sentences to CNF

Remember to write every clause on a seperate line

$Animal(F(x)) \vee Loves(G(x),x)$

$\neg Loves(x,F(x)) \vee Loves(G(x),x)$

$\neg Animal(y) \vee \neg Kills(x,y) \vee \neg Loves(z,x)$

$\neg Animal(x) \vee Loves(Jack,x)$

$Kills(Jack,Tuna) \vee Kills(Curiosity,Tuna)$

$Cat(Tuna)$

$\neg Cat(x) \vee Animal(x)$

$\neg Kills(Curiosity,Tuna)$

Resolution by Refutation

Cat(Tuna) | ¬Cat(x) ∨ Animal(x) | Kills(Jack,Tuna) ∨ Kills(Curiosity,Tuna) | ¬Kills(Curiosity,Tuna)

¬Animal(y) ∨ ¬Kills(x,y) ∨ ¬Loves(z,x)

¬Loves(x,F(x)) ∨ Loves(G(x),x) | ¬Animal(x) ∨ Loves(Jack,x)

Animal(F(x)) ∨ Loves(G(x),x)

Resolution by Refutation

Cat(Tuna) | ¬Cat(x) ∨ Animal(x) | Kills(Jack,Tuna) ∨ Kills(Curiosity,Tuna) | ¬Kills(Curiosity,Tuna)

¬Animal(y) ∨ ¬Kills(x,y) ∨ ¬Loves(z,x)

¬Loves(x,F(x)) ∨ Loves(G(x),x) | ¬Animal(x) ∨ Loves(Jack,x)

Animal(F(x)) ∨ Loves(G(x),x)

Resolution by Refutation

Cat(Tuna) | ¬Cat(x) ∨ Animal(x) | Kills(Jack,Tuna) ∨ Kills(Curiosity,Tuna) | ¬Kills(Curiosity,Tuna)

Animal(Tuna) | ¬Animal(y) ∨ ¬Kills(x,y) ∨ ¬Loves(z,x)

¬Loves(x,F(x)) ∨ Loves(G(x),x) | ¬Animal(x) ∨ Loves(Jack,x)

Animal(F(x)) ∨ Loves(G(x),x)

Resolution by Refutation

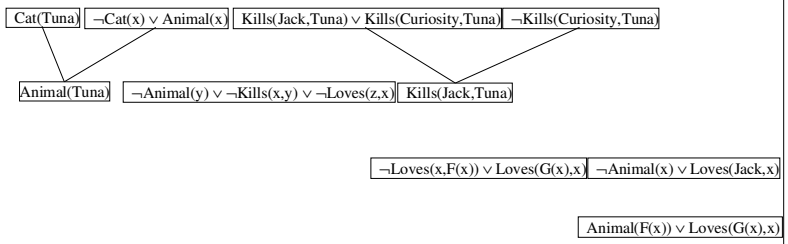
Cat(Tuna) | ¬Cat(x) ∨ Animal(x) | Kills(Jack,Tuna) ∨ Kills(Curiosity,Tuna) | ¬Kills(Curiosity,Tuna)

Animal(Tuna) | ¬Animal(y) ∨ ¬Kills(x,y) ∨ ¬Loves(z,x)

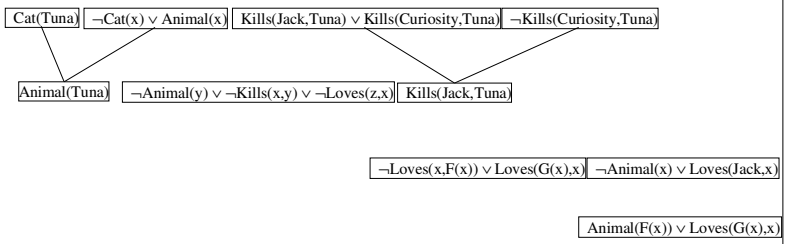
¬Loves(x,F(x)) ∨ Loves(G(x),x) | ¬Animal(x) ∨ Loves(Jack,x)

Animal(F(x)) ∨ Loves(G(x),x)

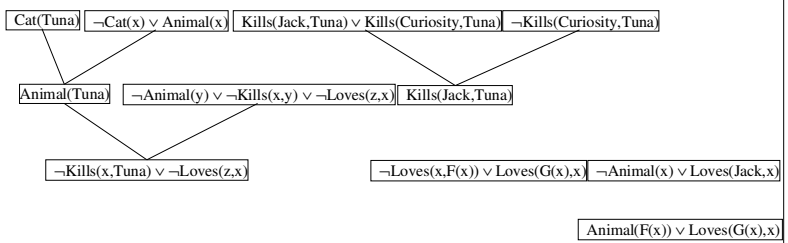
Resolution by Refutation



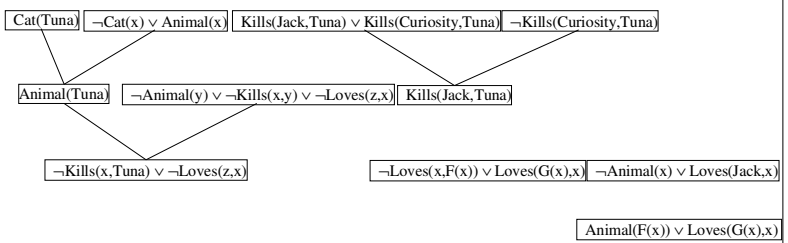
Resolution by Refutation



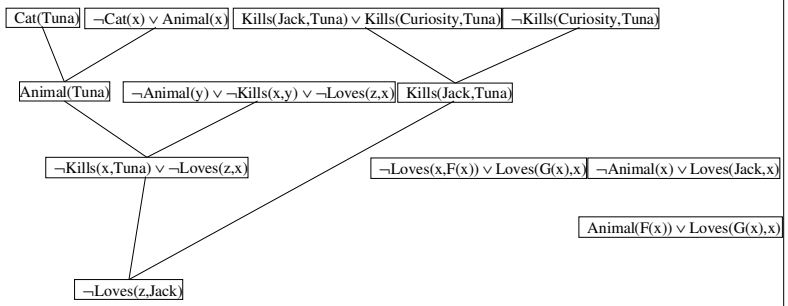
Resolution by Refutation



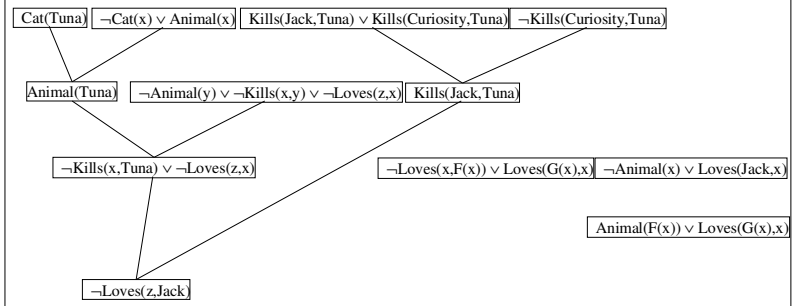
Resolution by Refutation



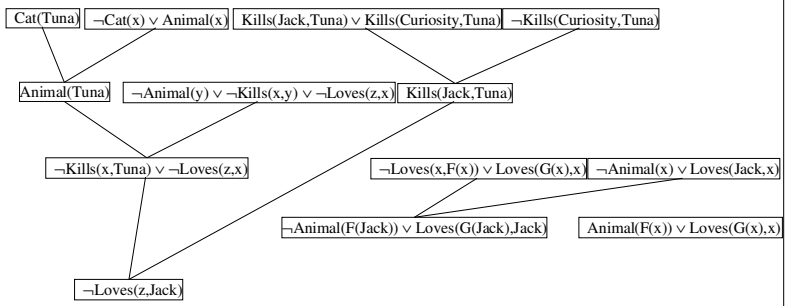
Resolution by Refutation



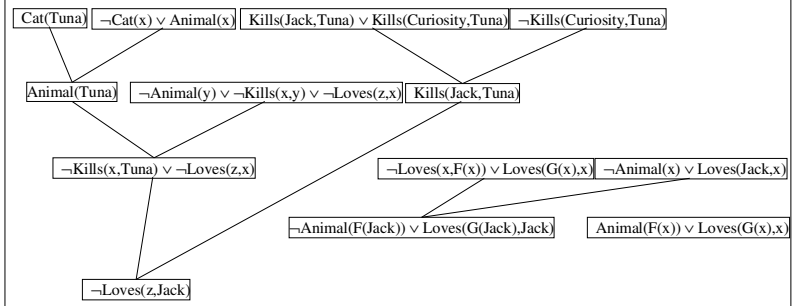
Resolution by Refutation



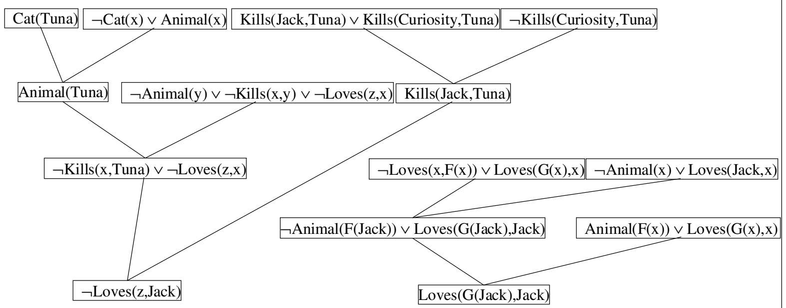
Resolution by Refutation



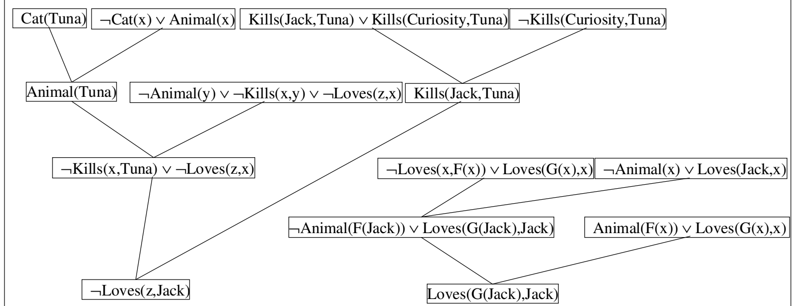
Resolution by Refutation



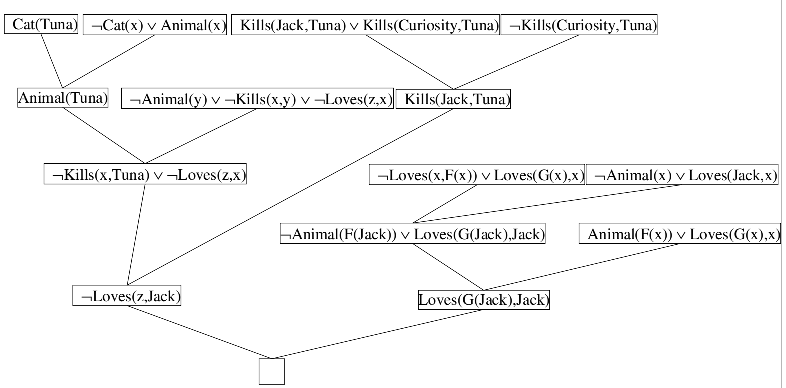
Resolution by Refutation



Resolution by Refutation



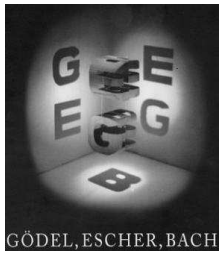
Resolution by Refutation



Resolution by Refutation

- Resolution is Refutation Complete
  - If $\alpha \models \beta$ then resolution will find a proof in a finite number of steps
  - Gödel's completeness theorem (1930)
  - If $\alpha \not\models \beta$ then resolution may never terminate – entailment for FOL is semidecidable
  - Similar to Turing's Halting Problem
- FOL with mathematical induction is incomplete
  - There are sentences that can not be proven even though $\alpha \models \beta$
  - Gödel's incompleteness theorem (1931)

Gödel, Escher, Bach
an Eternal Golden Braid



"I realized that to me, Gödel and Escher and Bach were only shadows cast in different directions by some central solid essence. I tried to reconstruct the central object, and came up with this book." -- Douglas Hofstadter

Induction

Induction (mathematics) – A two-part method of proving a theorem involving an integral parameter. First the theorem is verified for the smallest admissible value of the integer. Then it is proven that if the theorem is true for any value of the integer, it is true for the next greater value. The final proof contains the two parts.

Induction (logic) – The process of deriving general principles from particular facts or instances.

Conclusion

- How to Convert FOL to Propositional Logic
  - Universal Instantiation
  - Existential Instantiation
  - Drawbacks of doing this
- Reasoning in FOL with resolution
  - converting to CNF and skolemization
  - Most General Unifier
  - substitutions
- Next Time – Definite clauses, Back and Forward Chaining