Inference in First-Order Logic

CHAPTER 9

Oliver Schulte

Outline

- Reducing first-order inference to propositional inference
- Unification
- Lifted Resolution

Basic Setup

- We focus on a set of 1st-order clauses.
- All variables are universally quantified.
- Many knowledge bases can be converted to this format.
- Existential quantifiers are eliminated using function symbols
  - Quantifier elimination, Skolemization.
- Example UBC Prolog Demo

Two Basic Ideas for Inference in FOL

1. Grounding:
   I. Treat first-order sentences as a **template**.
   II. Instantiating all variables with all possible constants gives a set of ground propositional clauses.
   III. Apply efficient propositional solvers, e.g. SAT.

2. Lifted Inference:
   1. Generalize propositional methods for 1st-order methods.
   2. Unification: recognize instances of variables where necessary.

Universal instantiation (UI)

- Notation: Subst({v/g}, α) means the result of substituting g for v in sentence α

- 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., $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ yields

$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$, **{x/John}**

$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$, **{x/Richard}**

$King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$, **{x/Father(John)}**

Reduction to propositional form

Suppose the KB contains the following:

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
Father(x)
King(John)
Greedy(John)
Brother(Richard,John)

- Instantiating the universal sentence in all possible ways, we have:
  $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
  $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
  King(John)
  Greedy(John)
  Brother(Richard,John)

- The new KB is propositionalized: propositional symbols are
  - King(John), Greedy(John), Evil(John), King(Richard), etc

Reduction continued

- Every FOL KB can be propositionalized so as to preserve entailment
  - A ground sentence is entailed by new KB iff entailed by original KB

- Idea for doing inference in FOL:
  - propositionalize KB and query
  - apply resolution-based inference
  - return result

- Problem: with function symbols, there are infinitely many ground terms,
  - e.g., *Father*(*Father*(*Father*(*John*))), etc

Reduction continued

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

Example

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
Father(x)
King(John)
Greedy(Richard)
Brother(Richard,John)

Query Evil(X)?

- Depth 0

Father(John)

Father(Richard)

King(John)

Greedy(Richard)

Brother(Richard , John)

$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$

$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$

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

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

- Depth 1

Depth 0 +

Father(Father(John))

Father(Father(John))

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

Issues with Propositionalization

1. Problem: works if α is entailed, loops if α is not entailed

1. Propositionalization generates lots of irrelevant sentences
   - So inference may be very inefficient. E.g., consider KB

$\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
King(John)
$\forall y$ Greedy(y)
Brother(Richard,John)

   - It seems obvious that *Evil*(*John*) is entailed, but propositionalization produces lots of facts such as *Greedy*(*Richard*) that are irrelevant.
   - Approach: Magic Set Rewriting, from deductive databases.

1. With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations.
   - Current Research, Mitchell and Ternovska SFU.

- Alternative: do inference directly with FOL sentences

Unification

- Recall: Subst(θ, p) = result of substituting θ into sentence p

- Unify algorithm: takes 2 sentences p and q and returns a unifier if one exists

  Unify(p,q) = θ where Subst(θ, p) = Subst(θ, q)

- Example:
  p = Knows(John,x)
  q = Knows(John, Jane)

  Unify(p,q) = {x/Jane}

Unification examples

- simple example: query = Knows(John,x), i.e., who does John know?

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}

- Last unification fails: only because x can't take values John and OJ at the same time
- Problem is due to use of same variable x in both sentences
- Simple solution: Standardizing apart eliminates overlap of variables, e.g., Knows(z,OJ)

Unification

- To unify *Knows(John,x)* and *Knows(y,z)*,
- θ = {y/John, x/z } or θ = {y/John, x/John, z/John}
- The first unifier is more general than the second.
-
- **Theorem:** There is a single most general unifier (MGU) that is unique up to renaming of variables.
- MGU = { y/John, x/z }
- General algorithm in Figure 9.1 in the text

Recall our example…

$\forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x)$

King(John)

$\forall y \ Greedy(y)$

Brother(Richard,John)

- We would like to infer Evil(John) without propositionalization.
- Basic Idea: Use Modus Ponens, Resolution when literals **unify.**

Generalized Modus Ponens (GMP)

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

where we can unify p_i' and p_i for all *i*

Example:

$$\frac{King(John), Greedy(John) \quad , \forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)}{Evil(John)}$$

p_1' is $King(John)$ p_1 is $King(x)$

p_2' is $Greedy(John)$ p_2 is $Greedy(x)$

θ is {x/John} q is $Evil(x)$

Subst(θ,q) is $Evil(John)$

Logic programming: Prolog

- Program = set of clauses = `head :- literal`$_1$`, … literal`$_n$.

```
criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).
Missile(m1).
Owns(nono,m1).
Sells(west,X,nono):- Missile(X) Owns(nono,X).
weapon(X):- missile(X).
hostile(X) :- enemy(X,america).
american(west)
```

`Query : criminal(west)?`

`Query: criminial(X)?`

- membership
  - member(X,[X|_]).
  - member(X,[_|T]):- member(X,T).
    - ?-member(2,[3,4,5,2,1])
    - ?-member(2,[3,4,5,1])

- subset
  - subset([],L).
  - subset([X|T],L):- member(X,L),subset(T,L).
    - ?- subset([a,b],[a,c,d,b]).

- Nth element of list
  - nth(0,[X|_],X).
  - nth(N,[_|T],R):- nth(N-1,T,R).
    - ?nth(2,[3,4,5,2,1],X)

Proof Search in Prolog

- As in the propositional case, can do a depth-first or breadth-first search + unification.
- See UBC definite clause tool for demonstration.

Resolution in FOL

- Full first-order version:

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

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,

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

with $\theta = \{x/Ken\}$
- Apply resolution steps to CNF(KB $\wedge \neg\alpha$); complete for FOL.
- Gödel's completeness theorem.

Knowledge Base in FOL

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.
- Exercise: Formulate this knowledge in FOL.
-

Knowledge Base in FOL

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

… it is a crime for an American to sell weapons to hostile nations:

$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Nono … has some missiles, i.e., $\exists x\ Owns(Nono,x) \wedge Missile(x)$:

$Owns(Nono,M_1)$ *and* $Missile(M_1)$

… all of its missiles were sold to it by Colonel West

$Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$

Missiles are weapons:

$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile":

$Enemy(x,America) \Rightarrow Hostile(x)$

West, who is American …

$American(West)$

The country Nono, an enemy of America …

$Enemy(Nono,America)$

Example Knowledge Base in FOL (Hassan)

... it is a crime for an American to sell weapons to hostile nations:

American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)

Nono … has some missiles, i.e., ∃x Owns(Nono,x) ∧ Missile(x):

Owns(Nono,M_1) and Missile(M_1)

… all of its missiles were sold to it by Colonel West

Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)

Missiles are weapons:

Missile(x) ⇒ Weapon(x)

An enemy of America counts as "hostile":

Enemy(x,America) ⇒ Hostile(x)

West, who is American …

American(West)

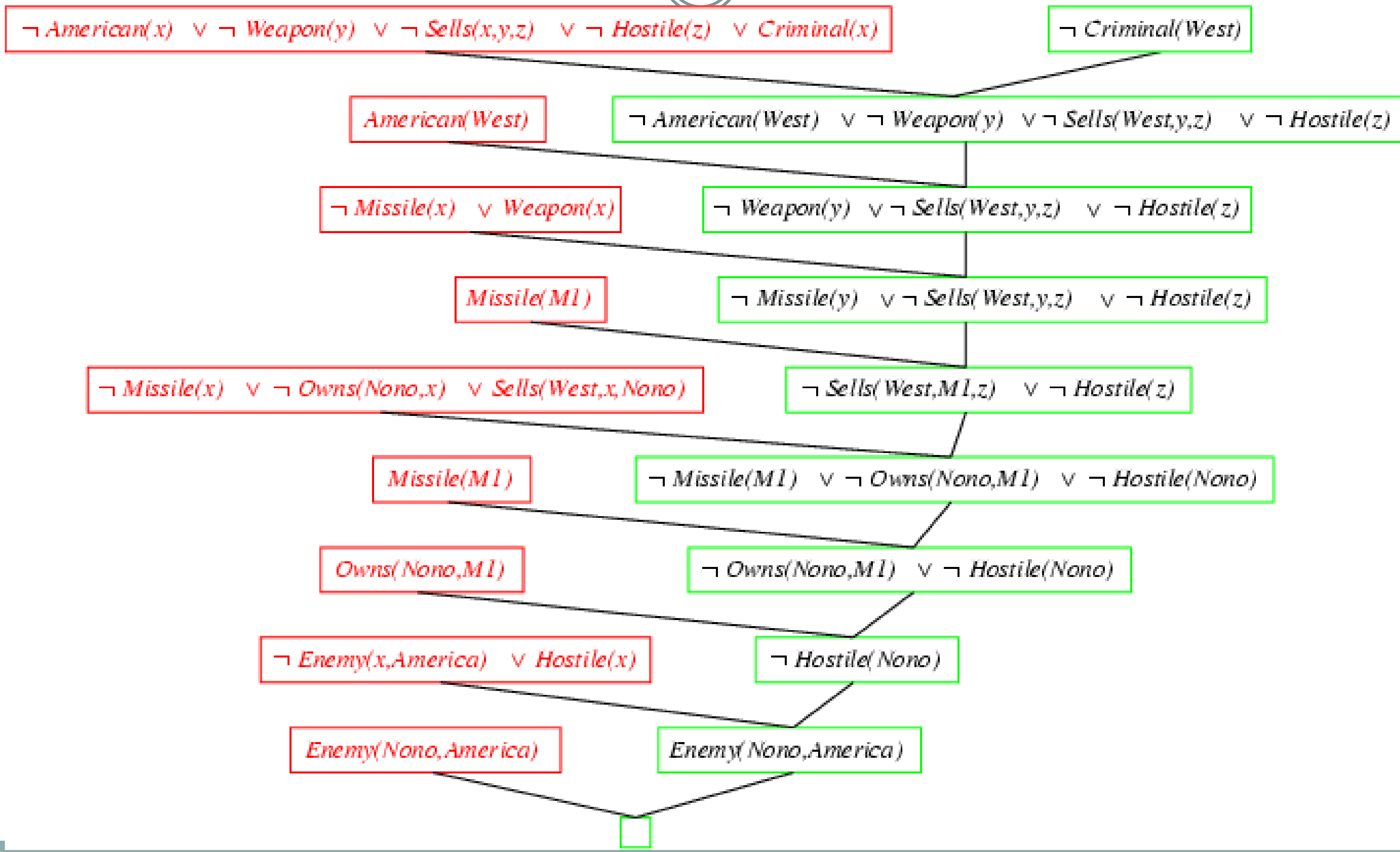
The country Nono, an enemy of America …

Enemy(Nono,America)

Can be converted to CNF

Query: Criminal(West)?

Resolution proof



Skolemization and Quantifier Elimination

- Problem: how can we use Horn clauses and aply unification with existential quantifiers?
- Not allowed by Prolog (try Aispace demo).
- Example.
  - Forall x. thereis y. Loves(y,x).
  - Forall x. forall y. Loves(y,x) => Good(x).
  - This entails (forall x. Good(x)) and Good(jack).
- Replace existential quantifiers by **Skolem functions.**
  - Forall x. Loves(**f(x)**,x).
  - Forall x. forall y. Loves(y,x) => Good(x).
  - This entails (forall x. Good(x)) and Good(jack).

The point of Skolemization

- Sentences with [forall thereis …] structure become [forall …].
  - Can use unification of terms.
- Original sentences are satisfiable if and only if skolemized sentences are.
- See Aispace demo.

Complex Skolemization Example

KB:

- *Everyone who loves all animals is loved by someone.*
- *Anyone who kills animals is loved by no-one.*
- *Jack loves all animals.*
- *Either Curiosity or Jack killed the cat, who is named Tuna.*

Query: *Did Curiosity kill the cat?*

Inference Procedure:

1. Express sentences in FOL.
2. Eliminate existential quantifiers.
3. Convert to CNF form and negated query.

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

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

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

D. $Kills(Jack, Tuna) \lor Kills(Curiosity, Tuna)$

E. $Cat(Tuna)$

F. $\forall x\, Cat(x) \Rightarrow Animal(x)$

$\neg$G. $\neg Kills(Curiosity, Tuna)$

A1. Animal(F(x)) $\vee$ Loves(G(x), x)

A2. $\neg$Loves(x, F(x)) $\vee$ Loves(G(x), x)

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

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

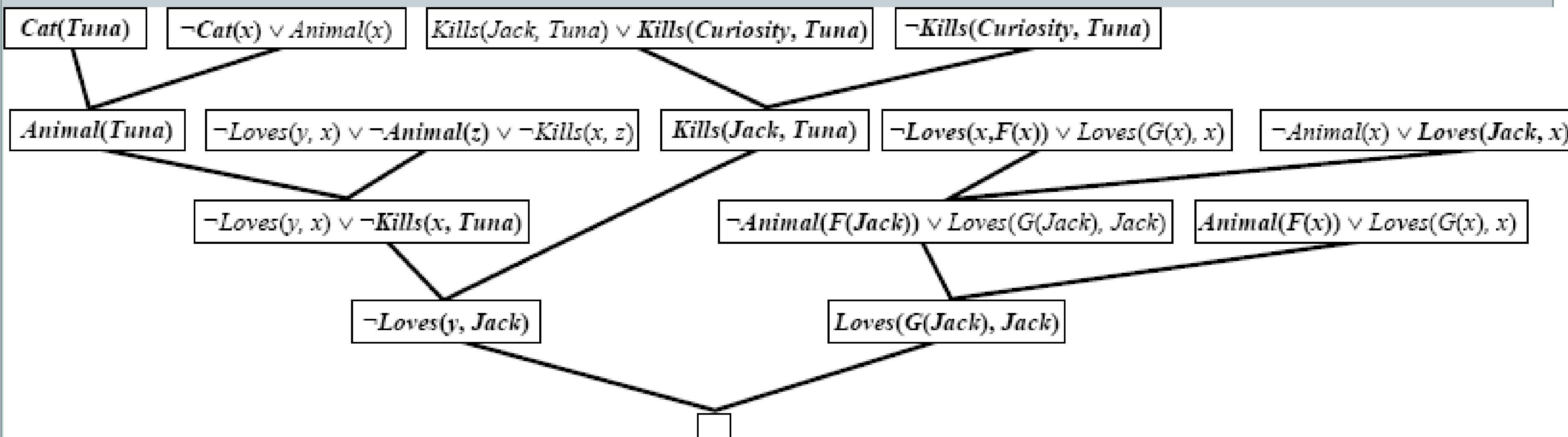
D. Kills(Jack, Tuna) $\vee$ Kills(Curiosity, Tuna)

E. Cat(Tuna)

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

$\neg$G. $\neg$Kills(Curiosity, Tuna)

Resolution-based Inference



Summary

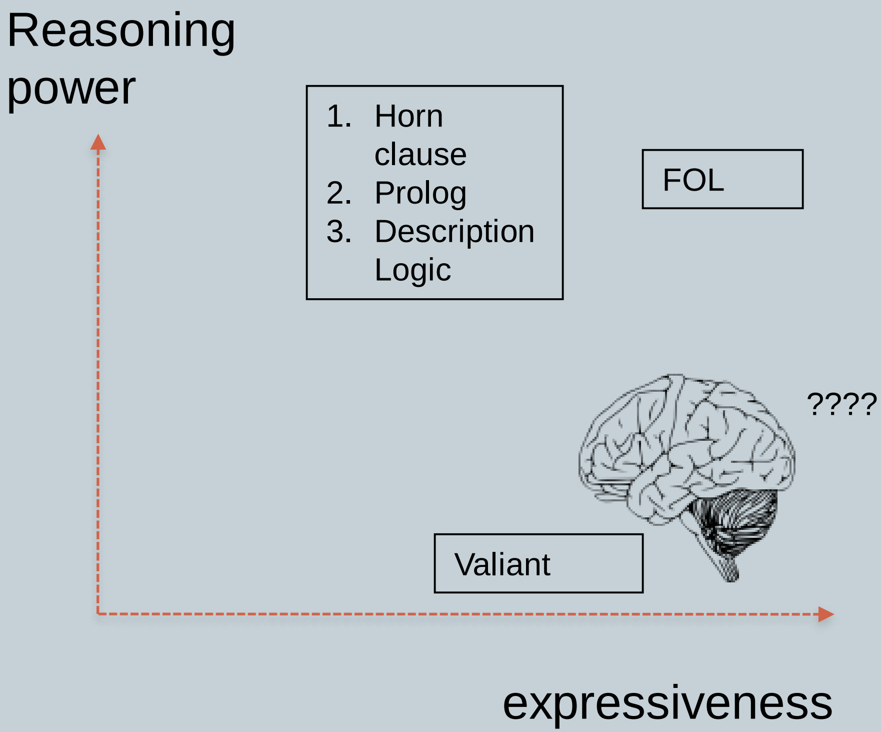
- Basic FOL inference algorithm (satisfiability check).

1. Use Skolemization to eliminate quantifiers
    1. Only universal quantifiers remain.
2. Convert to clausal form.
3. Use resolution + unification.

- This algorithm is **complete** (Gödel 1929).

Expressiveness vs. Tractability

- There is a fundamental trade-off between **expressiveness** and **tractability** in Artificial Intelligence.
- Similar, even more difficult issues with probabilistic reasoning (later).



Summary

- Inference in FOL
  - Grounding approach: reduce all sentences to PL and apply propositional inference techniques.

- FOL/Lifted inference techniques
  - Propositional techniques + Unification.
  - Generalized Modus Ponens
  - Resolution-based inference.

- Many other aspects of FOL inference we did not discuss in class