# Overview of Inference in First-Order Logic

Chapter 9

## Outline

◆□ ▶ < 圖 ▶ < 圖 ▶ < 圖 ▶ < 圖 • 의 Q @</p>

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

# Two Approaches for Inference in FOL

▲ロト ▲帰ト ▲ヨト ▲ヨト 三日 - の々ぐ

#### Propositionalisation:

- Treat a first-order sentences as a template.
- Instantiating all variables with all possible constants gives a set of ground propositional clauses.
- Apply efficient propositional solver, e.g. SAT.

# Two Approaches for Inference in FOL

#### Propositionalisation:

- Treat a first-order sentences as a template.
- Instantiating all variables with all possible constants gives a set of ground propositional clauses.
- Apply efficient propositional solver, e.g. SAT.

#### Lifted Inference:

• Generalize propositional methods for 1st-order methods.

- Issue: dealing with variables and quantifiers
- Primary approach: resolution
- Unification: instantiate variables where necessary.

### Propositionalisation

(ロ)、(型)、(E)、(E)、 E) の(の)

- Easy case: A finite world in which all individuals have names
  - E.g. the wumpus world, but also many planning, scheduling, etc. problems

## Propositionalisation

- Easy case: A finite world in which all individuals have names
  - E.g. the wumpus world, but also many planning, scheduling, etc. problems
- Idea:
  - Replace a universally-quantified sentence with all of its instances
  - Replace an existentially-quantified sentence with a disjunction of its instances

## Propositionalisation

- Easy case: A finite world in which all individuals have names
  - E.g. the wumpus world, but also many planning, scheduling, etc. problems
- Idea:
  - Replace a universally-quantified sentence with all of its instances
  - Replace an existentially-quantified sentence with a disjunction of its instances
- A formula (KB, etc.) with no variables is called ground
- *Inference procedure:* Ground the KB and the query, and run an inference procedure for propositional logic.

#### Universals

▲ロト ▲帰ト ▲ヨト ▲ヨト 三日 - の々ぐ

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

. . .

 $King(John) \land Greedy(John) \Rightarrow Evil(John)$  $King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$  $King(car_{54}) \land Greedy(car_{54}) \Rightarrow Evil(car_{54})$ 

#### Existentials

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

• E.g.,  $\exists x \ Likes(John, x)$ 

yields

 $Likes(John, John) \lor Likes(John, Richard) \lor \cdots \lor Likes(John, car_{54}) \lor \cdots$ 

#### Existentials

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

• E.g.,  $\exists x \ Likes(John, x)$ 

yields

 $Likes(John, John) \lor Likes(John, Richard) \lor \cdots \lor Likes(John, car_{54}) \lor \cdots$ 

Q: What does "Everyone likes someone" look like?

#### Reduction to propositional inference

- Suppose the KB contains just the following: ∀x King(x) ∧ Greedy(x) ⇒ Evil(x) King(John), Greedy(John), Brother(Richard, John)
- Instantiating the universal sentence in all possible ways, we get King(John) ∧ Greedy(John) ⇒ Evil(John) King(Richard) ∧ Greedy(Richard) ⇒ Evil(Richard) King(John), Greedy(John), Brother(Richard, John)
- The new KB is propositionalized.
- Proposition symbols are King(John), Greedy(John), Evil(John), King(Richard), etc.

#### Problems with propositionalization

- Usually generates lots of irrelevant sentences.
- E.g., consider: ∀x King(x) ∧ Greedy(x) ⇒ Evil(x), ∀y Greedy(y), King(John), Brother(Richard, John)
  - For query *Evil(John)*, propositionalization produces lots of facts (like *Greedy(Richard*)) that are irrelevant

• k-ary predicate and n constants  $\Rightarrow$   $n^k$  instances

#### Problems with propositionalization

- Usually generates lots of irrelevant sentences.
- E.g., consider: ∀x King(x) ∧ Greedy(x) ⇒ Evil(x), ∀y Greedy(y), King(John), Brother(Richard, John)
  - For query *Evil(John)*, propositionalization produces lots of facts (like *Greedy(Richard*)) that are irrelevant
- k-ary predicate and n constants  $\Rightarrow$   $n^k$  instances
- However, many recent AI applications use propositionalization for FO KBs over a finite domain.
  - Has led to work in *intelligent grounding*.
- Can make propositionalization work for arbitrary FO theories
  - See text for more

#### General FOL: Dealing with Variables

Consider the KB:

- $\{ \forall x (Grad(x) \Rightarrow Student(x)), \forall y (Student(y) \Rightarrow Happy(y)), Grad(ZeNian), UGrad(Andrei) \}$
- Intuitively *Happy*(*ZeNian*) is inferrable.
- For such a deduction Andrei is irrelevant.

Idea: Try to limit instantiation of variables to useful instances.

#### Unification

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

- If two formulas can be made the same by substitutions of variables, they are said to be *unified*
- Unification is the process of making 2 formulas (terms, etc) the same by finding an appropriate substitution for variables.

### Unification

- If two formulas can be made the same by substitutions of variables, they are said to be *unified*
- Unification is the process of making 2 formulas (terms, etc) the same by finding an appropriate substitution for variables.
- Consider:

 $\forall x (Grad(x) \Rightarrow Student(x)), Grad(ZeNian)$ 

### Unification

- If two formulas can be made the same by substitutions of variables, they are said to be *unified*
- Unification is the process of making 2 formulas (terms, etc) the same by finding an appropriate substitution for variables.
- Consider:

 $\forall x (Grad(x) \Rightarrow Student(x)), \qquad Grad(ZeNian)$ 

- To obtain *Student*(*ZeNian*) we have the following steps:
  - Figure out how to make *Grad*(*x*) and *Grad*(*ZeNian*) the same.
    - This is easy: Bind x to ZeNian.
  - Substituting, we get the rule instance: *Grad*(*ZeNian*) ⇒ *Student*(*ZeNian*).
  - Can now derive *Student*(*ZeNian*).

Look for substitution  $\theta$  such that  $\alpha\theta=\beta\theta$ 

$\alpha$	$\beta$	$\mid  heta$
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)	

Look for substitution  $\theta$  such that  $\alpha\theta=\beta\theta$ 

$\alpha$	$\beta$	$\theta$
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)	

◆□ ▶ < 圖 ▶ < 圖 ▶ < 圖 ▶ < 圖 • 의 Q @</p>

Look for substitution  $\theta$  such that  $\alpha \theta = \beta \theta$ 

$\alpha$	$\beta$	$\theta$
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)	

Look for substitution  $\theta$  such that  $\alpha\theta=\beta\theta$ 

$\alpha$	$\beta$	$\theta$
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)	

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

Look for substitution  $\theta$  such that  $\alpha \theta = \beta \theta$ 

$\alpha$	$\beta$	$\theta$
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

*Problem:* Can't substitute both *John* and *OJ* for x at the same time.

Solution: Standardize variables apart:

• Replace *Knows*(*x*, *OJ*) with *Knows*(*y*, *OJ*)

## Reasoning and Unification

- Unification lets us work with both universally quantified variables and arbitrary terms.
- We can use unification in rules such as: *Parent*(x, y) ∧ *Parent*(y, z) ⇒ *GrandParent*(x, z) where the variables are taken as being universally quantified.
- Then forward chaining and backward chaining with unification can be defined for such rules.
- For backward chaining, following one line of development, one ends up with the programming language Prolog.

### Resolution: Brief summary

- Resolution can be used in the first-order case (where it forms the basis for much of theorem proving)
- Full first-order version:

$$\frac{\ell_1 \vee C_1, \quad \ell_2 \vee C_2}{(C_1 \vee C_2)\theta} \quad \text{where } \ell_1 \theta = \neg \ell_2 \theta.$$

• For example,

$$\frac{\neg Rich(x) \lor Unhappy(x)}{\frac{Rich(Ken)}{Unhappy(Ken)}} \quad \text{with } \theta = \{x/Ken\}$$

• For details see the text or CMPT 411.

# Inference in FOL

For *KB* and query  $\alpha$ :

- Convert  $KB \land \neg \alpha$  to CNF.
  - This is trickier than in propositional logic, since we have to deal with variables and quantifiers.
- Apply resolution steps to  ${\it CNF}({\it KB} \wedge \neg lpha)$ 
  - No longer guaranteed to terminate if satisfiable
- Complete for FOL

## Summary

- Propositionalization
  - 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 not discussed in class