

CCES – College of Computer Engineering and Sciences

Prince Sattam Bin Abdulaziz University

**CS600: Advanced AI**

---

# First-Order Logic

Chapter 5 — Predicates, Quantifiers, Unification, and Resolution

---

Dr. Khalil Chebil

Artificial Intelligence Course

Why FOL & Syntax  
Quantifiers & Semantics  
Unification & Inference

Resolution & Summary

## Learning Objectives

- ▶ Explain what **objects**, **predicates** and **quantifiers** add to propositional logic
- ▶ Translate English into FOL without the classic  $\forall/\exists$  mistakes, and evaluate formulas in a model
- ▶ Compute **most general unifiers** and apply **Skolemization** correctly
- ▶ Prove queries with **forward chaining**, **backward chaining** and **resolution**

# 1

## Section 1

### Why FOL & Syntax

# Why First-Order Logic? | Propositional logic cannot generalise

## The propositional way

One symbol per fact, one rule per case:

*SocratesIsMan*  $\rightarrow$  *SocratesIsMortal*

*PlatolsMan*  $\rightarrow$  *PlatolsMortal*

... one rule for every person.

Wumpus on  $4 \times 4$ : 16 separate “breeze” rules.

## The first-order way

$\forall x \text{ Man}(x) \rightarrow \text{Mortal}(x)$

$\forall s \text{ Breezy}(s) \leftrightarrow \exists r \text{ Adjacent}(r, s) \wedge \text{Pit}(r)$

One sentence covers every person, every square.

## What FOL adds

**Objects:** John, Mary, square [1, 2]

**Relations:** *Parent*(John, Mary)

**Functions:** *Mother*(Ann) names an object

**Quantifiers:** “for all”  $\forall$ , “there exists”  $\exists$

## Scale

Family KB with 1,000 people: the grandparent rule alone has  $1000^3 = 10^9$  ground instances in propositional form. In FOL it is one line.

# Syntax: Terms, Predicates, Atoms | The building blocks

| Element   | Denotes                         | Examples                    |
|-----------|---------------------------------|-----------------------------|
| Constant  | a specific object               | <i>John, Beef, 2</i>        |
| Variable  | any object                      | <i>x, y, z</i>              |
| Function  | an object built from others     | <i>Mother(x), x + 1</i>     |
| Predicate | true/false property or relation | <i>Cat(x), Parent(x, y)</i> |

**Term:** constant, variable, or function applied to terms (an *object*).

**Atom:** predicate applied to terms (a *statement*).

## Term or atom?

*Mother(Ann):*  
*Loves(Ann, Mother(Ann)):*

*Mother(Mother(x)):*  
*Cat(Cat(x)):*

## Complex sentences

Combine atoms with  $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$  and quantifiers, as in propositional logic.

# Syntax: Terms, Predicates, Atoms | The building blocks

| Element   | Denotes                         | Examples                             |
|-----------|---------------------------------|--------------------------------------|
| Constant  | a specific object               | <i>John, Beef, 2</i>                 |
| Variable  | any object                      | <i>x, y, z</i>                       |
| Function  | an object built from others     | <i>Mother(x), <math>x + 1</math></i> |
| Predicate | true/false property or relation | <i>Cat(x), Parent(x, y)</i>          |

**Term:** constant, variable, or function applied to terms (an *object*).

**Atom:** predicate applied to terms (a *statement*).

## Term or atom?

*Mother(Ann)*: term (a person)

*Loves(Ann, Mother(Ann))*: atom (true or false)

*Mother(Mother(x))*: term (a grandmother)

*Cat(Cat(x))*: ✗ ill-formed: a predicate is not an object

## Complex sentences

Combine atoms with  $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$  and quantifiers, as in propositional logic.

## Section 2

# Quantifiers & Semantics

# Quantifiers | Statements about whole domains

## Universal $\forall$ : “for every”

$$\forall x \text{ Student}(x) \rightarrow \text{Smart}(x)$$

Over the domain  $\{Ann, Bob\}$  it is a conjunction:

$$\begin{aligned} & (\text{Student}(Ann) \rightarrow \text{Smart}(Ann)) \\ & \wedge (\text{Student}(Bob) \rightarrow \text{Smart}(Bob)) \end{aligned}$$

## Existential $\exists$ : “for some”

$$\exists x \text{ Student}(x) \wedge \text{Smart}(x)$$

Over the same domain it is a disjunction:

$$\begin{aligned} & (\text{Student}(Ann) \wedge \text{Smart}(Ann)) \\ & \vee (\text{Student}(Bob) \wedge \text{Smart}(Bob)) \end{aligned}$$

## Rule of thumb

$\forall$  goes with  $\rightarrow$ .  $\exists$  goes with  $\wedge$ .

## Mistake 1: $\forall x \text{ Student}(x) \wedge \text{Smart}(x)$

Says *everything* is a student and smart, including chairs.

## Mistake 2: $\exists x \text{ Student}(x) \rightarrow \text{Smart}(x)$

True as soon as one object is *not* a student (the implication is vacuously true), so it says almost nothing.

## Translating English into FOL | Think first, then reveal

| English                           | FOL |
|-----------------------------------|-----|
| 1 All cats are mammals.           |     |
| 2 Some student passed every exam. |     |
| 3 No student failed.              |     |
| 4 Everyone has a mother.          |     |
| 5 Siblings share a parent.        |     |
| 6 Only students take exams.       |     |

## Translating English into FOL | Think first, then reveal

| English                           | FOL                                                                                                                                      |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| 1 All cats are mammals.           | $\forall x \text{ Cat}(x) \rightarrow \text{Mammal}(x)$                                                                                  |
| 2 Some student passed every exam. | $\exists s \text{ Student}(s) \wedge \forall e (\text{Exam}(e) \rightarrow \text{Passed}(s, e))$                                         |
| 3 No student failed.              | $\neg \exists x (\text{Student}(x) \wedge \text{Failed}(x))$<br>$\equiv \forall x (\text{Student}(x) \rightarrow \neg \text{Failed}(x))$ |
| 4 Everyone has a mother.          | $\forall x \exists y \text{ Mother}(y, x)$                                                                                               |
| 5 Siblings share a parent.        | $\forall x, y \text{ Sibling}(x, y) \rightarrow$<br>$\exists p (\text{Parent}(p, x) \wedge \text{Parent}(p, y))$                         |
| 6 Only students take exams.       | $\forall x, e \text{ Takes}(x, e) \rightarrow \text{Student}(x)$                                                                         |

# Quantifier Order & Negation | Order changes meaning

## Same quantifiers commute

$$\forall x \forall y \equiv \forall y \forall x \quad \exists x \exists y \equiv \exists y \exists x$$

## Mixed quantifiers do not

$\forall x \exists y \text{ Loves}(x, y)$ : everyone loves *somebody* (maybe different people).

$\exists y \forall x \text{ Loves}(x, y)$ : there is *one* person whom everyone loves.

The second implies the first, not the other way round.

## Negation flips the quantifier

$$\neg \forall x P(x) \equiv \exists x \neg P(x)$$

$$\neg \exists x P(x) \equiv \forall x \neg P(x)$$

## Worked

$$\neg \forall x \exists y \text{ Loves}(x, y)$$

$$\equiv \exists x \neg \exists y \text{ Loves}(x, y)$$

$$\equiv \exists x \forall y \neg \text{Loves}(x, y)$$

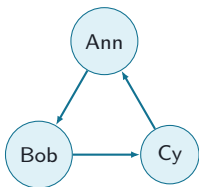
“Not everyone loves somebody” = “someone loves nobody”.

## Semantics: Evaluating in a Model | Fix a domain and the meaning of every symbol

### A small model

Domain:  $\{Ann, Bob, Cy\}$

$Loves = \{(Ann, Bob), (Bob, Cy), (Cy, Ann)\}$



arrow  $x \rightarrow y$ :  $x$  loves  $y$

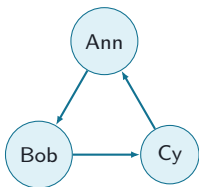
| Sentence                                                                                                                                                      | True here? |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|
| $\forall x \exists y Loves(x, y)$                                                                                                                             |            |
| $\exists y \forall x Loves(x, y)$                                                                                                                             |            |
| $\exists x Loves(x, x)$                                                                                                                                       |            |
| $\forall x, y Loves(x, y) \rightarrow Loves(y, x)$                                                                                                            |            |
| <b>Valid:</b> true in every model. <b>Entailment</b> $KB \models \alpha$ : $\alpha$ true in every model of $KB$ , which now may have infinitely many objects. |            |

## Semantics: Evaluating in a Model | Fix a domain and the meaning of every symbol

### A small model

Domain:  $\{Ann, Bob, Cy\}$

$Loves = \{(Ann, Bob), (Bob, Cy), (Cy, Ann)\}$



arrow  $x \rightarrow y$ :  $x$  loves  $y$

| Sentence                                           | True here?                          |
|----------------------------------------------------|-------------------------------------|
| $\forall x \exists y Loves(x, y)$                  | ✓ each person has an outgoing arrow |
| $\exists y \forall x Loves(x, y)$                  | ✗ nobody receives three arrows      |
| $\exists x Loves(x, x)$                            | ✗ no self-loops                     |
| $\forall x, y Loves(x, y) \rightarrow Loves(y, x)$ | ✗ Ann loves Bob, not back           |

**Valid:** true in every model. **Entailment**  $KB \models \alpha$ :  $\alpha$  true in every model of  $KB$ , which now may have infinitely many objects.

## Section 3

# Unification & Inference

# Substitution & Unification | Making two expressions identical

## Substitution $\theta = \{x/t, \dots\}$

Replace each variable  $x$  by the term  $t$ :  
 $P(x, f(y))\{x/a, y/b\} = P(a, f(b))$

## Unifier

$\theta$  **unifies**  $E_1$  and  $E_2$  if  $E_1\theta = E_2\theta$ .  
The **most general unifier** (MGU)  
commits to as little as possible.

## Two ways to fail

Clash:  $P(x, x)$  vs.  $P(a, b)$  needs  $x = a$   
and  $x = b$ .

**Occur check:**  $x$  vs.  $f(x)$  would need  
an infinite term.

Find the MGU:

| $E_1$            | $E_2$                 | MGU |
|------------------|-----------------------|-----|
| $Knows(John, x)$ | $Knows(John, Jane)$   |     |
| $Knows(John, x)$ | $Knows(y, Bill)$      |     |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ |     |
| $Knows(John, x)$ | $Knows(x, Eliza)$     |     |
| $P(x, f(y))$     | $P(y, f(a))$          |     |

# Substitution & Unification | Making two expressions identical

## Substitution $\theta = \{x/t, \dots\}$

Replace each variable  $x$  by the term  $t$ :  
 $P(x, f(y))\{x/a, y/b\} = P(a, f(b))$

## Unifier

$\theta$  **unifies**  $E_1$  and  $E_2$  if  $E_1\theta = E_2\theta$ .  
The **most general unifier** (MGU)  
commits to as little as possible.

## Two ways to fail

Clash:  $P(x, x)$  vs.  $P(a, b)$  needs  $x = a$   
and  $x = b$ .

**Occur check:**  $x$  vs.  $f(x)$  would need  
an infinite term.

Find the MGU:

| $E_1$            | $E_2$                 | MGU                             |
|------------------|-----------------------|---------------------------------|
| $Knows(John, x)$ | $Knows(John, Jane)$   | $\{x/Jane\}$                    |
| $Knows(John, x)$ | $Knows(y, Bill)$      | $\{x/Bill, y/John\}$            |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | $\{y/John, x/Mother(John)\}$    |
| $Knows(John, x)$ | $Knows(x, Eliza)$     | fail ( $x = John$ and $Eliza$ ) |
| $P(x, f(y))$     | $P(y, f(a))$          | $\{x/a, y/a\}$                  |

## Standardising Apart & the MGU | Two details that trip up implementations

### Standardising apart

The failure  $\text{Knows}(\text{John}, x)$  vs.  $\text{Knows}(x, \text{Eliza})$  is an accident: the two sentences are separate rules, and each  $x$  is local to its own sentence.

Rename one:  $\text{Knows}(z, \text{Eliza})$ . Now  
 $\theta = \{z/\text{John}, x/\text{Eliza}\}$

### Inside one formula

Different quantifiers should use different names:  
 $\forall x P(x) \wedge \exists x Q(x)$  becomes  $\forall x P(x) \wedge \exists y Q(y)$ .

### Why “most general”?

$\text{Knows}(\text{John}, x)$  vs.  $\text{Knows}(y, z)$ :

$\theta_1 = \{y/\text{John}, x/z\}$  gives  $\text{Knows}(\text{John}, z)$

$\theta_2 = \{y/\text{John}, x/\text{John}, z/\text{John}\}$  gives

$\text{Knows}(\text{John}, \text{John})$

$\theta_1$  is the MGU:  $\theta_2$  is  $\theta_1$  followed by  $\{z/\text{John}\}$ .

Choosing  $\theta_2$  too early would miss answers like  $\text{Knows}(\text{John}, \text{Jane})$ .

The MGU is unique up to renaming of variables.

## Universal instantiation

$$\frac{\forall x \alpha}{\alpha\{x/t\}} \quad \text{for any ground term } t$$

$\forall x \text{ Mortal}(x)$  gives  $\text{Mortal}(\text{Socrates})$ .

## Existential instantiation

$$\frac{\exists x \alpha}{\alpha\{x/C\}} \quad C \text{ a **new** constant}$$

$\exists x \text{ King}(x)$  gives  $\text{King}(C_1)$ , where  $C_1$  names “some king”.

Not  $\text{King}(\text{John})$ : we do not know *which* king it is.

## Generalised Modus Ponens

$$\frac{p'_1, \dots, p'_n, (p_1 \wedge \dots \wedge p_n \rightarrow q)}{q\theta} \quad p'_i\theta = p_i\theta$$

Rule:  $\text{Eats}(x, y) \wedge \text{Meat}(y) \rightarrow \text{NonVeg}(x)$

Facts:  $\text{Eats}(\text{John}, \text{Beef})$ ,  $\text{Meat}(\text{Beef})$

1. Unify  $\text{Eats}(x, y)$  with the fact:  
 $\{x/\text{John}, y/\text{Beef}\}$
2.  $\text{Meat}(y)\theta = \text{Meat}(\text{Beef})$  is a fact ✓
3. Conclude  $\text{NonVeg}(\text{John})$

GMP is one step that does instantiation and modus ponens together.

# Skolemization | Remove $\exists$ by naming the witness

## Skolem constant ( $\exists$ not inside any $\forall$ )

$$\exists x P(x) \Rightarrow P(C_1)$$

## Skolem function ( $\exists$ inside $\forall$ )

“Everyone has a mother”:

$$\forall x \exists y \text{Mother}(y, x) \Rightarrow \forall x \text{Mother}(M(x), x)$$

The witness depends on  $x$ , so it becomes a function  $M(x)$  of  $x$ .

## Why not a constant?

$\forall x \text{Mother}(C, x)$  would say one woman is everyone's mother.

## Worked: nested quantifiers

$$\forall x \exists y \forall z \exists w P(x, y, z, w)$$

$y$  is inside  $\forall x$  only:  $y \mapsto f(x)$

$w$  is inside  $\forall x$  and  $\forall z$ :  $w \mapsto g(x, z)$

$$\forall x \forall z P(x, f(x), z, g(x, z))$$

## Try it

$$\exists y \forall x \text{Loves}(x, y) \Rightarrow$$

$$\forall x \exists y \text{Loves}(x, y) \Rightarrow$$

The Skolemized formula is *satisfiable iff* the original is, which is all resolution needs.

# Skolemization | Remove $\exists$ by naming the witness

## Skolem constant ( $\exists$ not inside any $\forall$ )

$$\exists x P(x) \Rightarrow P(C_1)$$

## Skolem function ( $\exists$ inside $\forall$ )

“Everyone has a mother”:

$$\forall x \exists y \text{Mother}(y, x) \Rightarrow \forall x \text{Mother}(M(x), x)$$

The witness depends on  $x$ , so it becomes a function  $M(x)$  of  $x$ .

## Why not a constant?

$\forall x \text{Mother}(C, x)$  would say one woman is everyone's mother.

## Worked: nested quantifiers

$$\forall x \exists y \forall z \exists w P(x, y, z, w)$$

$y$  is inside  $\forall x$  only:  $y \mapsto f(x)$

$w$  is inside  $\forall x$  and  $\forall z$ :  $w \mapsto g(x, z)$

$$\forall x \forall z P(x, f(x), z, g(x, z))$$

## Try it

$$\exists y \forall x \text{Loves}(x, y) \Rightarrow \forall x \text{Loves}(x, C_2)$$

$$\forall x \exists y \text{Loves}(x, y) \Rightarrow \forall x \text{Loves}(x, L(x))$$

The Skolemized formula is *satisfiable iff* the original is, which is all resolution needs.

# Forward Chaining in FOL | Fire rules for every substitution that matches the facts

## KB

Facts:  $Parent(John, Bob)$ ,  
 $Parent(Bob, Mary)$

R1:  $Parent(x, y) \rightarrow Anc(x, y)$

R2:

$Parent(x, z) \wedge Anc(z, y) \rightarrow Anc(x, y)$

Query:  $Anc(John, Mary)$ ?

Definite clauses: all variables  
implicitly  $\forall$ ; one positive conclusion.

| Round | Rule | $\theta$                    | New fact            |
|-------|------|-----------------------------|---------------------|
| 1     | R1   | $\{x/John, y/Bob\}$         | $Anc(John, Bob)$    |
| 1     | R1   | $\{x/Bob, y/Mary\}$         | $Anc(Bob, Mary)$    |
| 2     | R2   | $\{x/John, z/Bob, y/Mary\}$ | $Anc(John, Mary)$ ✓ |

## Properties

Sound and complete for definite clauses. Function-free KBs (**Datalog**) always terminate, in polynomial time. With function symbols it may run forever.

## Lab KB: A Missing Condition | Forward chaining derives exactly what the rules say

### Family KB (lab)

*Parent(John, Mary), Parent(John, Bob),  
Parent(Mary, Ann)*  
*Parent(X, Y)  $\wedge$  Parent(Y, Z)  $\rightarrow$   
Grandparent(X, Z)*  
*Parent(Z, X)  $\wedge$  Parent(Z, Y)  $\rightarrow$  Sibling(X, Y)*

Which facts are derived?

## Lab KB: A Missing Condition | Forward chaining derives exactly what the rules say

### Family KB (lab)

*Parent(John, Mary), Parent(John, Bob),  
Parent(Mary, Ann)*  
*Parent(X, Y)  $\wedge$  Parent(Y, Z)  $\rightarrow$   
Grandparent(X, Z)*  
*Parent(Z, X)  $\wedge$  Parent(Z, Y)  $\rightarrow$  Sibling(X, Y)*

Which facts are derived?

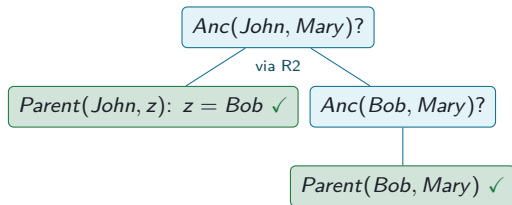
### Derived

*Grandparent(John, Ann)*  
*Sibling(Mary, Bob), Sibling(Bob, Mary)*

### And also...

*Sibling(Mary, Mary), Sibling(Bob, Bob),  
Sibling(Ann, Ann):* X and Y may bind to the same person.  
Fix: add  $X \neq Y$  to the premises.

# Backward Chaining in FOL | Goal-driven, as in Prolog



## Trace

R1 needs  $Parent(John, Mary)$ : not a fact.  
R2 with  $\{x/John, y/Mary\}$ : subgoals  $Parent(John, z)$  and  $Anc(z, Mary)$ .  
First subgoal binds  $z = Bob$ ; second succeeds by R1.

## Depth-first danger

If R2 were written  
 $Anc(z, y) \wedge Parent(x, z) \rightarrow Anc(x, y)$ , a  
Prolog-style prover would recurse forever.  
Order of premises matters.



# Section 4

## Resolution & Summary

## Conversion to CNF | Six steps before resolution

### Steps

1. Eliminate  $\leftrightarrow$ ,  $\rightarrow$
2. Move  $\neg$  inward (also  $\neg\forall \rightarrow \exists\neg$ ,  $\neg\exists \rightarrow \forall\neg$ )
3. Standardise variables apart
4. Skolemize
5. Drop the  $\forall$  quantifiers
6. Distribute  $\vee$  over  $\wedge$

### Worked: “every cat has an owner”

$$\forall x [Cat(x) \rightarrow \exists y Owns(y, x)]$$

$$\text{Step 1: } \forall x [\neg Cat(x) \vee \exists y Owns(y, x)]$$

$$\text{Step 4: } \forall x [\neg Cat(x) \vee Owns(F(x), x)]$$

$$\text{Step 5: } \neg Cat(x) \vee Owns(F(x), x) \quad \text{one clause}$$

$$\text{Try it: } \neg\exists x (Cat(x) \wedge Dog(x))$$

## Conversion to CNF | Six steps before resolution

### Steps

1. Eliminate  $\leftrightarrow$ ,  $\rightarrow$
2. Move  $\neg$  inward (also  $\neg\forall \rightarrow \exists\neg$ ,  $\neg\exists \rightarrow \forall\neg$ )
3. Standardise variables apart
4. Skolemize
5. Drop the  $\forall$  quantifiers
6. Distribute  $\vee$  over  $\wedge$

### Worked: “every cat has an owner”

$$\forall x [Cat(x) \rightarrow \exists y Owns(y, x)]$$

$$\text{Step 1: } \forall x [\neg Cat(x) \vee \exists y Owns(y, x)]$$

$$\text{Step 4: } \forall x [\neg Cat(x) \vee Owns(F(x), x)]$$

$$\text{Step 5: } \neg Cat(x) \vee Owns(F(x), x) \quad \text{one clause}$$

### Try it: $\neg\exists x (Cat(x) \wedge Dog(x))$

Step 2:

$$\forall x \neg(Cat(x) \wedge Dog(x)) \equiv \forall x (\neg Cat(x) \vee \neg Dog(x));$$

$$\text{drop } \forall: \neg Cat(x) \vee \neg Dog(x).$$

# Resolution with Unification | Unify complementary literals, then resolve

## The rule

$$\frac{\ell \vee \alpha \quad \neg m \vee \beta}{(\alpha \vee \beta)\theta}$$

where  $\theta = \text{UNIFY}(\ell, m)$ .

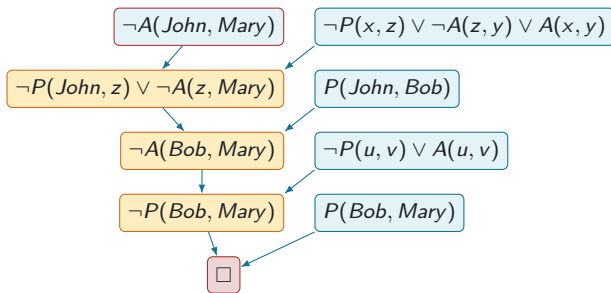
## Simple example

$\neg \text{Cat}(x) \vee \text{Mammal}(x), \text{Cat}(W)$   
 $\theta = \{x/W\} \Rightarrow \text{Mammal}(W)$   
 $W = \text{Whiskers}$

## Properties

Sound and **refutation-complete**: if  $KB \models \alpha$ ,  $\square$  is eventually derived.

Prove  $\text{Anc}(\text{John}, \text{Mary})$  by refutation:



$P = \text{Parent}, A = \text{Anc}$ .  $\theta$ s:  $\{x/\text{John}, y/\text{Mary}\}, \{z/\text{Bob}\}, \{u/\text{Bob}, v/\text{Mary}\}$ . R1 was renamed  $u, v$  (standardised apart).

## Application: Rule-Based Medical Diagnosis | From the chapter's Streamlit project

### Rules with confidence

Flu: at least 3 of {fever, body\_aches, cough, fatigue}; base confidence 0.8.

The “3 of 4” test is a disjunction of the four 3-symptom conjunctions.

Score when  $k$  of  $n$  conditions hold:

$$\text{confidence} = \text{base} \times \frac{k}{n}$$

### Worked

Patient reports fever, cough, fatigue (no body aches):

$$0.8 \times 3/4 =$$

### Is this really first-order?

Each symptom is a yes/no variable about *one* patient, so the KB is **propositional**. A first-order version would quantify over patients:

$$\forall p \text{ Fever}(p) \wedge \text{Cough}(p) \wedge \dots \rightarrow \text{Flu}(p)$$

The partial-match score is a heuristic for uncertainty; principled treatments (probabilistic graphical models) come in Chapter 11.

## Application: Rule-Based Medical Diagnosis | From the chapter's Streamlit project

### Rules with confidence

Flu: at least 3 of {fever, body\_aches, cough, fatigue}; base confidence 0.8.

The “3 of 4” test is a disjunction of the four 3-symptom conjunctions.

Score when  $k$  of  $n$  conditions hold:

$$\text{confidence} = \text{base} \times \frac{k}{n}$$

### Worked

Patient reports fever, cough, fatigue (no body aches):

$$0.8 \times 3/4 = \mathbf{0.60}, \text{ i.e. } 60\% \text{ for Flu.}$$

### Is this really first-order?

Each symptom is a yes/no variable about *one* patient, so the KB is **propositional**. A first-order version would quantify over patients:

$$\forall p \text{ Fever}(p) \wedge \text{Cough}(p) \wedge \dots \rightarrow \text{Flu}(p)$$

The partial-match score is a heuristic for uncertainty; principled treatments (probabilistic graphical models) come in Chapter 11.

# Propositional vs. First-Order Logic | More power, less decidability

| Aspect                | Propositional              | First-order                                                                                 |
|-----------------------|----------------------------|---------------------------------------------------------------------------------------------|
| <b>Talks about</b>    | facts (true/false)         | objects, relations, functions                                                               |
| <b>Generalisation</b> | one rule per case          | quantifiers: one rule for all                                                               |
| <b>Matching</b>       | identical symbols          | unification                                                                                 |
| <b>Entailment</b>     | decidable (co-NP-complete) | <b>semi-decidable</b> : a proof is found if one exists; otherwise the search may never stop |
| <b>Horn inference</b> | linear time                | Datalog polynomial; with functions may not terminate                                        |

## Practical consequence

Theorem provers and Prolog use time limits, depth bounds or restricted languages (Datalog, description logics) to stay practical.

## 1. $\forall$ with $\wedge$ , $\exists$ with $\rightarrow$

Usually wrong; use  $\forall \dots \rightarrow$  and  $\exists \dots \wedge$ .

## 2. Swapping quantifier order

$\forall x \exists y$  and  $\exists y \forall x$  mean different things.

## 3. Reusing a known name for $\exists$

Existential instantiation needs a *new* constant.

## 4. Skolem constant inside $\forall$

Use a function of the enclosing universal variables.

## 5. Forgetting to standardise apart

Variables in different clauses are unrelated; rename them.

## 6. Missing conditions in rules

$Sibling(x, x)$  appears unless the rule says  $x \neq y$ .

## Practice Problems | Answers on the next click

| Problem                                                               | Answer |
|-----------------------------------------------------------------------|--------|
| 1 Translate: "Every course has some student who likes it."            |        |
| 2 Negate and simplify $\forall x (P(x) \rightarrow Q(x))$ .           |        |
| 3 MGU of $P(f(x), y)$ and $P(z, g(z))$ ?                              |        |
| 4 MGU of $Q(x, x)$ and $Q(y, f(y))$ ?                                 |        |
| 5 Skolemize $\forall x \exists y \forall z \exists w R(x, y, z, w)$ . |        |
| 6 Why can FOL resolution run forever?                                 |        |

## Practice Problems | Answers on the next click

| Problem                                                               | Answer                                                                                            |
|-----------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| 1 Translate: "Every course has some student who likes it."            | $\forall c \text{ Course}(c) \rightarrow \exists s (\text{Student}(s) \wedge \text{Likes}(s, c))$ |
| 2 Negate and simplify $\forall x (P(x) \rightarrow Q(x))$ .           | $\exists x (P(x) \wedge \neg Q(x))$                                                               |
| 3 MGU of $P(f(x), y)$ and $P(z, g(z))$ ?                              | $\{z/f(x), y/g(f(x))\}$                                                                           |
| 4 MGU of $Q(x, x)$ and $Q(y, f(y))$ ?                                 | fail: $x/y$ , then $y$ vs. $f(y)$ fails the occur check                                           |
| 5 Skolemize $\forall x \exists y \forall z \exists w R(x, y, z, w)$ . | $\forall x \forall z R(x, f(x), z, g(x, z))$                                                      |
| 6 Why can FOL resolution run forever?                                 | Entailment is semi-decidable; if $\alpha$ is not entailed, new resolvents may never run out       |

# Chapter Summary | Key takeaways

## Represent

Objects, predicates, functions and quantifiers let one sentence cover a whole domain. Watch the  $\forall/\rightarrow$  and  $\exists/\wedge$  pairing and the order of quantifiers.

## Reason

Unification (MGU) matches patterns. GMP drives forward and backward chaining on definite clauses; CNF + Skolemization + resolution handle full FOL.

## Limits

Entailment is semi-decidable; knowledge bases are only as good as their rules (missing  $x \neq y$ ); pure logic has no notion of uncertainty.

## Looking ahead

Ch. 6: **machine learning**, learning rules and models from data instead of writing them by hand.

# Thank you!

Questions & Discussion

Next: Chapter 6 — Machine Learning