

---

---

# Predicate logic

---

---

Readings :

- CHAPTERS 8,9 from Russell + Norvig
- CHAPTER 15,16 from Nilsson

BIBLIOGRAPHY:

- E. Paniagua Arís, J. L. Sánchez González, F. Martín Rubio, “Lógica computacional”, Thomson
- Melvin Fitting “First-order logic and automated theorem proving”, Springer-Verlag (New-York 1990)

# Higher-order logics

- Propositional logic lacks expressive power
  - » Concise description of environments with many objects.
  - » General statements about all objects in a given domain, the relations they have, the existence of objects with certain properties, etc.
- Other forms of logic
  - » **Zero order logic** or **Propositional logic**  
Object constants, logical connectives.
  - » **First order logic:**  
++ functions, predicates, quantifiers whose arguments range over objects.  
e.g.  $(\forall x) \text{Man}(x) \Rightarrow \text{Mortal}(x)$
  - » **Second order logic:**  
++ functions, predicates, quantifiers whose arguments range over predicates.  
e.g.  $(\exists P) [P(A) \wedge P(B)]$   
 $(\forall x)(\forall y) [(x=y) \Leftrightarrow (\forall P)(P(x)=P(y))]$   
[ *Leibniz' principle*, 1666]
  - » **Higher order logics:**  
++ functions, predicates, quantifiers whose arguments range over predicates on predicates ...
  - » **Type theory.**

# First order logic

## Example: Family + friends

- **Ontology**
  - » **Connectives, Variables, Quantifiers**
  - » **Constants**
    - Object constants:** Peter, Paul, Mary
    - Functions:** parentOf<sup>1</sup>, bestFriendOf<sup>1</sup>
    - Predicates:** Male<sup>1</sup>, Female<sup>1</sup>, Child<sup>2</sup>, Son<sup>2</sup>, Daughter<sup>2</sup>, Married<sup>1</sup>, Happy<sup>1</sup>,...
- **Definitions:** “Someone’s son is someone’s male child”  
 $(\forall x, y) (\text{Son}(x, y) \Rightarrow \text{Child}(x, y) \wedge \text{Male}(x))$
- **General statements:**  
“All of Mary’s children are married, but are unhappy”  
 $(\forall x) (\text{Child}(x, \text{Mary}) \Rightarrow \text{Married}(x) \wedge \neg \text{Happy}(x))$
- **Existential statements:** “Some of the sons of Peter’s best friend have children”  
 $(\exists x) (\exists y) (\text{Son}(x, \text{bestFriendOf}(\text{Peter})) \wedge \text{Child}(y, x))$

# Language, I

- **Constants:**

- » **Object constants** (usually upper case)

E.g. T, F  
P, Q, John, EiffelTower (symbolic)

- » **Function constants** (usually lower case)

Input arguments: list of terms between parentheses.

Evaluates to: a term.

E.g. fatherOf<sup>1</sup>, distanceBetween<sup>2</sup>

- » **Relation constants or predicates** (us. upper case)

Input arguments: list of terms between parentheses

Evaluates to: a truth value

E.g. Parent<sup>2</sup>, White<sup>1</sup>

**Unary relation constants are properties**

**Notes:**

- » The superindex denotes the arity of the function or of the predicate (i.e. the number of arguments it takes)
- » Symbolic object constants can be considered as either functions or relations of zero arity

# Language, II

- **Punctuation signs**

- » **Comma:** ,
- » **Parentheses:** ( ) [ ] { }

- **Propositional connectives:**

$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$  (in order of precedence)

- **Variable symbols** (usually in lowercase)

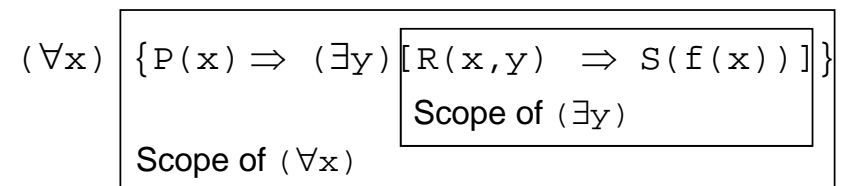
Variables that can take on values in a domain.  
e.g. x (reals), n (integer), P (persons), etc.

E.g. x, y, p, q, ..., p1, p2, ...

- **Quantifiers**

- » **Universal quantifier** ( $\forall$ )
- » **Existential quantifier** ( $\exists$ )

Used in conjunction with variables. The variable is said to be bound to the quantifier.



## Term, atom, literal

- A **term** is either
  - » An **object constant**  
E.g. T, Peter, P, P1, ...
  - » A **variable symbol**  
E.g. x, y
  - » A **function constant** of arity n **followed by n terms** separated by commas and between parentheses.  
E.g. fatherOf(x), distanceBetween(A, B)A **ground term** is a term that does not contain any variables.
- An **atom** is either:
  - » A **term**
  - » An **atomic formula**: A **relation constant** of arity n **followed by n terms** separated by commas and between parentheses.  
E.g. Parent(x, Peter)
- A **literal** is either:
  - » A **positive literal**: An atom  
E.g. Sibling(x, Peter)
  - » A **negative literal**: The negation of an atom  
E.g.  $\neg$  Parent(John, fatherOf(Peter))

## Well-formed formulas

- A **wff** is constructed using atoms and connectives in the same way as in propositional calculus.
- If w is a wff and x is a variable symbol, the following expressions are also wffs
  - »  $(\forall x) w$
  - »  $(\exists x) w$Usually, but not necessarily x will be embedded in w. If x is embedded in w, we usually write  $w(x)$
- **Closed wff** (closed sentence): If all variables in w are quantified over, then w is a closed wff.
  - »  $(\forall x) [P(x) \Rightarrow R(x)]$
  - »  $(\exists x) [P(x) \Rightarrow (\exists y) (R(x, y) \Rightarrow S(f(x)))]$

Note: The **order** in which  $\forall, \exists$  appear is **important**

$x, y \in \{\text{People}\}$

Father(x, y) "x is the father of y" (binary relation)

$(\forall x)(\forall y) \text{Father}(x, y)$  "Everyone is everyone else's father"

$(\exists x)(\exists y) \text{Father}(x, y)$  "There is someone who has a father"

$(\forall x)(\exists y) \text{Father}(x, y)$  "Everyone is the father of someone"

$(\exists y)(\forall x) \text{Father}(x, y)$  "There is someone who has everyone as a father"

$(\exists x)(\forall y) \text{Father}(x, y)$  "There is someone who is the father of everyone"

$(\forall y)(\exists x) \text{Father}(x, y)$  "Everyone has a father"

# Interpretation and semantics

- **Constants**
  - » **Symbolic object constants** correspond to **real world objects**.
  - » **Predicates of arity 1** refer to **object properties**.
  - » **Predicates of arity  $n > 1$**  express **relations between objects**.
- **Variables**
  - » The **domain** of a variable is the range of values of symbolic object constants that the variable can take.
  - » **Assignment**: Operation where a variable that appears in a wff is replaced for one of the symbolic object constants in its domain.
- **Quantifiers**
  - » **Universal quantifier**:  $(\forall x) w(x)$  has value **True** whenever  $w(x)$  has value **True** for all possible assignments of the variable symbol  $x$ .
  - » **Existential quantifier**:  $(\exists x) w(x)$  has value **True** whenever  $w(x)$  has value **True** for some assignment of the variable symbol  $x$ .

# Equivalence and inference rules

- **Equivalence rules**
  - » **Equivalence rules of propositional logic**.
  - » **Renaming quantized variables** (dummy variables). The new name must be different from that of the other variables in the wff.
    - $(\forall x) w(x) \equiv (\forall y) w(y)$
    - $(\exists x) w(x) \equiv (\exists y) w(y)$
  - »  $\neg(\forall x) w(x) \equiv (\exists x) \neg w(x)$
  - »  $\neg(\exists x) w(x) \equiv (\forall x) \neg w(x)$
- **Inference rules**
  - » **Inference rules of propositional logic**.
  - » **Universal instantiation (UI)** [Sound]  
 $(\forall x) w(x) \vdash_{UI} w(A)$ , where  $A$  is any symbolic object constant in the domain of  $x$ .
  - » **Existential generalization (EG)** [Sound]  
 $w(A) \vdash_{EG} (\exists x) w(x)$ , where  $x$  is variable symbol whose domain is the domain to which the object constant  $A$  belongs.

# Skolemization

An **existential quantifier** can be **removed** from a wff by **replacing** each occurrence of the **existentially quantized variable** by either

- » a **Skolem object constant** if there are no universally quantized variables within the scope of the existential quantifier that is being eliminated.

Ex.  $(\exists x) w(x)$

**Skolem form:**  $w(Sk)$

- » a **Skolem function** whose arguments are those universally quantified variables that are bound to universal quantifiers whose scope includes the scope of the existential quantifier that is being eliminated.

Ex. "Every person has a height"

$(\forall p) [ (\exists h) \text{Height}(p, h) ]$

domain of p: Persons.

domain of h: Positive reals.

**Skolem form:**  $(\forall p) \text{Height}(p, h(p))$

$h(p)$  is a Skolem function that takes as an argument a person and returns her/his height.

# Metatheorems for Skolem forms

- Metatheorem SK1: The Skolem form of a wff is NOT equivalent to the original wff

$$w(Sk) \models (\exists x) w(x) \text{ BUT } (\exists x) w(x) \not\models w(Sk)$$

E.g.

$$P(A) \vee P(B) \models (\exists x) P(x)$$

$$\text{BUT } P(A) \vee P(B) \not\models P(Sk)$$

- Metatheorem SK2 (Loveland, 1978):  
The Skolem form of a set of wffs is **inferentially equivalent** to the original set of wffs.

That is, the Skolem form of a set of wffs is satisfiable exactly in those cases where the original knowledge base is satisfiable.

- » A set of wffs is satisfiable iff their Skolem form is satisfiable.
- » A set of wffs is unsatisfiable iff their Skolem form is unsatisfiable.

# CNF in first order logic

1. **Eliminate implications**  $\Leftrightarrow, \Rightarrow$
2. **Reduce the scope of  $\neg$** 
  - » de Morgan's laws
$$\neg(w1 \vee w2) \equiv \neg w1 \wedge \neg w2$$
$$\neg(w1 \wedge w2) \equiv \neg w1 \vee \neg w2$$
  - » Elimination of repeated negations ( $\neg\neg w \equiv w$ )
  - » Combination of  $\neg$  with quantifiers.
$$\neg(\forall x) w(x) \equiv (\exists x) \neg w(x)$$
$$\neg(\exists x) w(x) \equiv (\forall x) \neg w(x)$$
3. **Standardize variables:** Rename variables so that each different variable in the set of wffs has a different symbol
$$[(\forall x) [P(x) \Rightarrow R(x)]] \vee [(\exists x) P(x)]$$
$$\equiv [(\forall x) [P(x) \Rightarrow R(x)]] \vee [(\exists y) P(y)]$$
4. **Skolemization:** Eliminate existential quantifiers and replace existentially quantized variables by Skolem constants or Skolem functions as appropriate.
5. **Convert to prenex form** by moving all universal quantifiers to the beginning of the wff.

wff in **prenex form** = **Prefix** (string of quantifiers)  
+ **Matrix** (quantifier-free formula)
6. **Drop universal quantifiers.**
7. **Use distributive laws and equivalence rules** of propositional logic to transform the **matrix to CNF.**

13

# Example 1: Conversion to CNF

$$[(\forall x) Q(x)] \Rightarrow$$
$$(\forall x)(\forall y) [(\exists z) [P(x,y,z) \Rightarrow (\forall u) R(x,y,u,z)]]$$

1. **Eliminate implications**  $\Leftrightarrow, \Rightarrow$ 
$$\neg[(\forall x) Q(x)] \vee$$
$$(\forall x)(\forall y) [(\exists z) [\neg P(x,y,z) \vee (\forall u) R(x,y,u,z)]]$$
2. **Reduce scope of negations:**
$$[(\exists x) \neg Q(x)] \vee$$
$$(\forall x)(\forall y) [(\exists z) [\neg P(x,y,z) \vee (\forall u) R(x,y,u,z)]]$$
3. **Standardize variables**
$$[(\exists w) \neg Q(w)] \vee$$
$$(\forall x)(\forall y) [(\exists z) [\neg P(x,y,z) \vee (\forall u) R(x,y,u,z)]]$$
4. **Skolemization:**
$$\neg Q(A) \vee (\forall x,y) [\neg P(x,y,f(x,y)) \vee (\forall u) R(x,y,u,f(x,y))]$$
5. **Prenex form:**
$$(\forall x,y,u) [\neg Q(A) \vee \neg P(x,y,f(x,y)) \vee R(x,y,u,f(x,y))]$$
6. **Drop universal quantifiers**
$$\neg Q(A) \vee \neg P(x,y,f(x,y)) \vee R(x,y,u,f(x,y))$$
7. **Convert to CNF:** wff already in CNF.

14

## Example 2: Conversion to CNF

“Everybody who loves all animals is loved by someone”

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

### 1. Eliminate implications $\Leftrightarrow, \Rightarrow$

$$(\forall x)[\neg(\forall y)\{\neg\text{Animal}(y) \vee \text{Loves}(x,y)\} \vee (\exists y) \text{Loves}(y,x)]$$

### 2. Reduce scope of negations:

$$(\forall x)[(\exists y)\{\text{Animal}(y) \wedge \neg\text{Loves}(x,y)\} \vee (\exists y) \text{Loves}(y,x)]$$

### 3. Standardize variables

$$(\forall x)[(\exists y)\{\text{Animal}(y) \wedge \neg\text{Loves}(x,y)\} \vee (\exists z) \text{Loves}(z,x)]$$

### 4. Skolemization:

$$(\forall x)[\{\text{Animal}(f(x)) \wedge \neg\text{Loves}(x,f(x))\} \vee \text{Loves}(g(x),x)]$$

### 5. Prenex form: wff already in prenex form

### 6. Drop universal quantifiers

$$\{\text{Animal}(f(x)) \wedge \neg\text{Loves}(x,f(x))\} \vee \text{Loves}(g(x),x)$$

### 7. Convert to CNF using distributive laws and equiv. rules

$$\{\text{Animal}(f(x)) \vee \text{Loves}(g(x),x)\} \wedge \{\neg\text{Loves}(x,f(x))\} \vee \text{Loves}(g(x),x)$$

15

## Substitution

- Consider an expression  $w$  containing the variables  $v_1, v_2, v_3, \dots, v_n$ . A substitution  $\sigma$  is a **simultaneous replacement** of terms for variables in  $w$ .

$$w = w(v_1, v_2, v_3, \dots, v_n)$$

$$\downarrow \sigma = \{v_1 := t_1, v_2 := t_2, \dots, v_n := t_n\}$$

$$w\sigma = w(t_1, t_2, t_3, \dots, t_n)$$

The term  $t_i$  cannot contain the variable  $v_i$ .

The term  $t_i$  is an instantiation of the variable  $v_i$ .

$w\sigma$  is an **instance** of  $w$ .

- Consider the substitutions

$$\sigma_1 = \{x_1 := t_1, x_2 := t_2, \dots, x_n := t_n\}$$

$$\sigma_2 = \{y_1 := s_1, y_2 := s_2, \dots, y_m := s_m\}$$

The composition of these substitutions is

$$\sigma_1 \cdot \sigma_2 = \{x_1 := t_1 \sigma_2, x_2 := t_2 \sigma_2, \dots, x_n := t_n \sigma_2, y_1 := s_1, y_2 := s_2, \dots, y_m := s_m\}$$

$$\gg w(\sigma_2 \cdot \sigma_1) = (w\sigma_2)\sigma_1$$

$$\gg (\sigma_1 \cdot \sigma_2) \cdot \sigma_3 = \sigma_1 \cdot (\sigma_2 \cdot \sigma_3)$$

$$\gg \sigma_1 \cdot \sigma_2 \neq \sigma_2 \cdot \sigma_1 \text{ [In general]}$$

$$w = P(x, y), \quad \sigma_1 = \{x := f(y)\}, \quad \sigma_2 = \{y := A\}$$

$$\sigma_1 \cdot \sigma_2 = \{x := f(A), y := A\}, \quad \sigma_2 \cdot \sigma_1 = \{y := A, x := f(y)\}$$

$$w\sigma_1 = P(f(y), y) \quad w(\sigma_1 \cdot \sigma_2) = (w\sigma_1)\sigma_2 = P(f(A), A)$$

$$w\sigma_2 = P(x, A) \quad w(\sigma_2 \cdot \sigma_1) = (w\sigma_2)\sigma_1 = P(f(y), A) \quad 16$$

## Substitution: examples

- Example 1:  $w = A(x, y, f(z))$   
 $\sigma = \{x := D, y := g(x), z := C\}$   
 $w\sigma = A(D, g(x), f(C))$
- Example 2:  $w = B(x) \vee C(x, f(y))$   
 $\sigma = \{x := y, y := g(A)\}$   
 $w\sigma = B(y) \vee C(y, f(g(A)))$
- Example 3:  $w = P(x, f(y), B)$ 
  - »  $\sigma_1 = \{x := z, y := w\}$   
 $w\sigma_1 = P(z, f(w), B)$  [Alphabetic variant]
  - »  $\sigma_2 = \{y := A\}$   
 $w\sigma_2 = P(x, f(A), B)$
  - »  $\sigma_3 = \{x := g(z), y := A\}$   
 $w\sigma_3 = P(g(z), f(A), B)$
  - »  $\sigma_4 = \{x := C, y := A\}$   
 $w\sigma_4 = P(C, f(A), B)$  [Ground instance]

A **ground instance** of a wff is a wff obtained by substitution from the original wff that does not contain any variables.

## Unification

- The **substitution**  $\sigma$  is the **unifier** of a set of wffs  $\Gamma = \{w_1, w_2, \dots, w_m\}$  when  $w_1\sigma = w_2\sigma = \dots = w_m\sigma$ .
  - » The process is called **unification**.
  - » The common instance produced by unification is **unique** except for possible alphabetic variants.

Ex.  $\{P(x, f(y), B), P(z, f(B), B)\}$

$\sigma = \{x := A, y := B, z := A\}$

Unification:  $\{P(A, f(B), B)\}$

- **Unification is sound:** Since all variables are universally quantized, unification is a particular form of universal instantiation, which is a sound inference rule.

- **Most general unifier (MGU)**

The substitution  $\mu$  is the most general unifier of a set of wffs  $\Gamma = \{w_1, w_2, \dots, w_m\}$ ,  $\mu = \text{mgu}(\Gamma)$ , if any unifier of the same set of wffs  $\sigma$  can be expressed as  $\sigma = \sigma' \cdot \mu$

- The **disagreement set** of a non-empty set of wffs  $\Gamma = \{w_1, w_2, \dots, w_m\}$  is the first (leftmost) subexpression where the wffs in  $\Gamma$  disagree

Ex.  $\Gamma = \{P(x, f(A), B), P(x, g(A), B)\}$

$D(\Gamma) = \{f(A), g(A)\}$



## Generalized (lifted) resolution

- Binary resolution is **sound** but **not refutation complete**

Ex.  $\Gamma = \{\neg P(x) \vee \neg P(y), P(z) \vee P(r)\}$

$\mu_1 = \{z := x\}$

infer by resolution  $\{\neg P(y) \vee P(r)\}$

From  $\{\neg P(x) \vee \neg P(y), \neg P(y) \vee P(r)\}$

using the mgu  $\mu_2 = \{r := y\}$

infer by resolution  $\{\neg P(x) \vee \neg P(y)\}$

All binary resolutions produce only alphabetic variants of these clauses.

- Assume that all the literals in clause  $\kappa_1$  can be unified with the negated literals in clause  $\kappa_2$  by means of a most general unifier  $\mu$ .

Then, from the set of clauses  $\{\kappa_1 \cup \Sigma_1, \kappa_2 \cup \Sigma_2\}$ , it is possible to infer  $w = (\Sigma_1 \cup \Sigma_2)\mu$

Ex.  $\Gamma = \{\neg P(x) \vee \neg P(y), P(z) \vee P(r)\}$

$\mu = \{y := x, z := x, r := x\}$

infer by resolution  $\{\square\}$

## Generalized subsumption

- Clause  $\kappa_1$  **subsumes** clause  $\kappa_2$  if there exists a **substitution**  $\sigma$  such that

$$\kappa_1 \sigma \subset \kappa_2$$

- For the purpose of **inference** the **subsumed clause**,  $\kappa_2$  can be **eliminated** from the knowledge base

- Example:

$\kappa_1$ : Enjoys(Ringo, z)

$\kappa_2$ :  $\neg$ Pleasant(Reading)  $\vee$  Enjoys(Ringo, Reading)

$\sigma$ :  $\{z := \text{Reading}\}$

[Intuition: Clause  $\kappa_1$  is more “restrictive” than clause  $\kappa_2$ ]

## Generalized *modus ponens*

Assuming that the wff's  $w_1$  and  $w_2$  have a most general unifier  $\mu$ ,  $w_3\mu$  can be inferred from  $w_1$  and  $w_2 \Rightarrow w_3$

- **Example1:** Is John evil?

Rules:  $(\forall x) [King(x) \wedge Greedy(x) \Rightarrow Evil(x)]$

Facts:  $King(John), Greedy(John)$

one can infer ( $\mu = \{x := John\}$ )

$Evil(John)$

- **Example2:** Does Ringo like Peanuts?

Rules:  $(\forall x) [Food(x) \Rightarrow Likes(Ringo, x)]$

$(\forall p, x) [\neg Dead(p) \wedge Eats(p, x) \Rightarrow Food(x)]$

$(\forall x) [Eats(John, x) \Rightarrow Eats(George, x)]$

Facts:  $Eats(John, Peanuts), \neg Dead(John),$   
 $Food(Apple), Food(Chicken)$

From:  $Eats(John, Peanuts), \neg Dead(John),$

$[\neg Dead(p) \wedge Eats(p, x) \Rightarrow Food(x)]$

(use  $\{p := John, x := Peanuts\}$ )

Infer:  $Food(Peanuts)$

From:  $[Food(x) \Rightarrow Likes(Ringo, x)], Food(Peanuts)$

( $\mu = \{x := Peanuts\}$ )

Infer: **Likes(Ringo, Peanuts)**

23

## Answer extraction

Consider the Knowledge base  $\Delta$ .

Assume we need a proof of  $w = (\exists x)P(x)$

and we want to know for which instances of  $x$  this expression is entailed.

**Green's trick (1999):**

Use resolution refutation and include an answer literal in  $\neg w$  to keep track of the substitutions made.

Add to KB:  $(\forall x) [\neg P(x) \vee Answer(x)]$

**In general:** Add a literal  $Answer(x_1, x_2, \dots, x_m)$  to each clause generated from the negation of the theorem to be proved.

Example:

KB: All men are mortal.  $(\forall x) [Man(x) \Rightarrow Mortal(x)]$

Socrates is a man.  $Man(Socrates)$

Is there a mortal man?  $(\exists x) [Man(x) \wedge Mortal(x)]$

Add to KB:  $(\forall x) [\neg Man(x) \vee \neg Mortal(x) \vee Ans(x)]$

$\alpha = \{ \neg Man(x) \vee \neg Mortal(x), Man(Socrates),$   
 $\neg Man(y) \vee \neg Mortal(y) \vee Ans(y) \}$

$Mortal(Socrates)$

$\neg Mortal(Socrates) \vee Ans(Socrates)$

$Ans(Socrates)$

24

## Entailment in FOL

- **Entailment:** The KB  $\Delta$  entails the wff  $w$  when every model of  $\Delta$  is also a model of  $w$ .

$$\Delta \models w$$

- **Inference procedures:**

- » **Propositionalization**

- Convert FOL wffs to propositional wffs by elimination of all universal and existential quantifications.
- Apply inference for propositional logic on the propositionalized KB.

- » Use **generalized (lifted) inference rules**.

In particular, use proof by refutation with a generalized resolution inference rule.

- Metatheorem [Turing (1936), Church (1936)]

**“Entailment for FOL is semi-decidable”.**

Algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence.

## Propositionalization

- Solving  $\Delta_{\text{FOL}} \models_{\text{FOL}} w$  in FOL by **propositionalization**:
  - » Construct  $\alpha_{\text{FOL}} = \Delta_{\text{FOL}} \wedge \neg w_{\text{FOL}}$
  - » Transform  $\alpha_{\text{FOL}}$  to CNF ( $\exists$  eliminated by Skolemization)
  - » Transform every clause in the CNF form of  $\alpha_{\text{FOL}}$  to propositional clauses by making all possible assignments to the variables in  $\alpha_{\text{FOL}}$ , so that only ground terms appear.

**Problem:** If we consider functions symbols, there are **infinitely many** ground terms,

e.g., `sonOf(sonOf(...sonOf(sonOf(Paul))...))`

- Metatheorem [Herbrand (1930)]

The set of FOL wffs in CNF  $\alpha_{\text{FOL}}$  is unsatisfiable iff there is a finite set of ground clauses of  $\alpha_{\text{FOL}}$  that is unsatisfiable in propositional calculus.

- Algorithm

$n \leftarrow 0; \alpha_{\text{FOL}}$

1.  $n \leftarrow n+1$

2. Create the propositional CNF  $\alpha_n$  by instantiating terms in  $\alpha_{\text{FOL}}$  with a maximum depth in function calls  $n$ .

Until (resolution on  $\alpha_n$  produces the empty clause)

**Problem:** The algorithm works if  $\alpha_{\text{FOL}}$  is UNSAT, but it never halts if  $\alpha_{\text{FOL}}$  is SAT.

# Resolution refutation in FOL

## Generalized resolution refutation

Consider the Knowledge base  $\Delta$  and the wff  $w$  in FOL.

Is  $\Delta \models w$  ?

1. Include the negated wff in KB  $\alpha = \{\Delta \wedge \neg w\}$
2. Convert to CNF
3. Apply generalized (lifted) resolution
  - (i) If resolution produces the empty clause ( $\alpha$  is UNSAT) then  $\Delta \models w$
  - (ii) If resolution does not produce the empty clause ( $\alpha$  is SAT) then  $w$  is not entailed by  $\Delta$ .

- » If  $\Delta \models w$ , algorithm stops and answers yes.
- » If  $\Delta \not\models w$ , algorithm may never stop.

**Resolution is sound, refutation complete but semidecidable.**

# Tricks

Usually,

$$(\forall x) [w1(x) \Rightarrow w2(x)]$$

$$(\exists x) [w1(x) \wedge w2(x)]$$

E.g. “Everybody at the UAM is smart”

$$(\forall x) [At(x, UAM) \Rightarrow Smart(x)]$$

“There are people at the UAM who are smart”

$$(\exists x) [At(x, UAM) \wedge Smart(x)]$$

## COMMON MISTAKES

$$(\exists x) [At(x, UAM) \Rightarrow Smart(x)] \quad \text{[TOO WEAK]}$$

“There is someone who is either smart or not at the UAM”

$$(\forall x) [At(x, UAM) \wedge Smart(x)] \quad \text{[TOO STRONG]}$$

“Everybody is at the UAM and is smart”

## Did curiosity kill the cat?

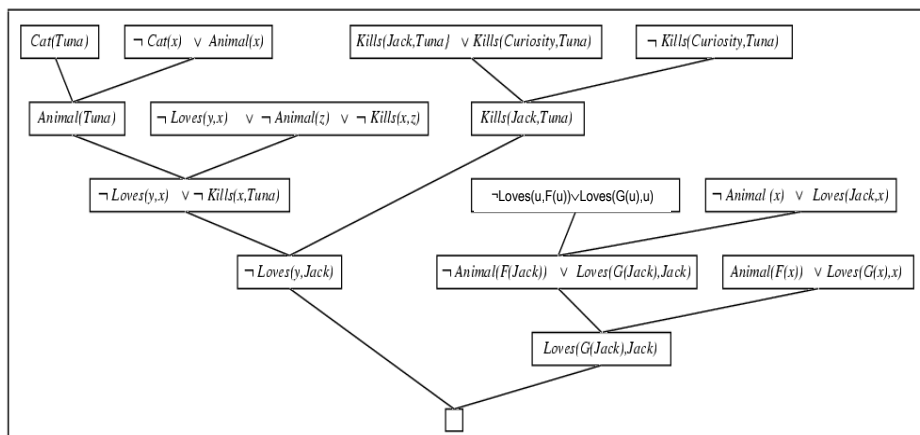
- Everyone who loves all animals is loved by someone.
- Anyone who kills an animal is loved by no one.
- Jack loves all animals.
- Either Jack or Curiosity killed the cat, who is named Tuna.
- Did Curiosity kill the cat?

## Did curiosity kill the cat?

- Everyone who loves all animals is loved by someone.  
 $(\forall x) [ (\forall y) [ \text{Animal}(y) \Rightarrow \text{Loves}(x,y) ] \Rightarrow (\exists z) \text{Loves}(z,x) ]$
- Anyone who kills an animal is loved by no one.  
 $(\forall x) [ (\exists y) ( \text{Animal}(y) \wedge \text{Kills}(x,y) ) \Rightarrow (\forall z) \neg \text{Loves}(z,x) ]$
- Jack loves all animals.  
 $(\forall x) [ \text{Animal}(x) \Rightarrow \text{Loves}(\text{Jack},x) ]$
- Either Jack or Curiosity killed the cat, who is named Tuna.  
 $\text{Kills}(\text{Jack},\text{Tuna}) \vee \text{Kills}(\text{Curiosity},\text{Tuna})$   
 $\text{Cat}(\text{Tuna})$
- A cat is an animal  
 $(\forall x) \text{Cat}(x) \Rightarrow \text{Animal}(x)$
- Did Curiosity kill the cat?  
 $\text{Kills}(\text{Curiosity},\text{Tuna}) \text{ ???}$

## Did curiosity kill the cat?

- A.  $\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)$
- B.  $\neg \text{Loves}(u, F(u)) \vee \text{Loves}(G(u), u)$
- C.  $\neg \text{Animal}(y) \vee \neg \text{Kills}(x, y) \vee \neg \text{Loves}(z, x)$
- D.  $\neg \text{Animal}(x) \vee \text{Loves}(\text{Jack}, x)$
- E.  $\text{Kills}(\text{Jack}, \text{Tuna}) \vee \text{Kills}(\text{Curiosity}, \text{Tuna})$
- F.  $\text{Cat}(\text{Tuna})$
- G.  $\neg \text{Cat}(x) \vee \text{Animal}(x)$
- H.  $\neg \text{Kills}(\text{Curiosity}, \text{Tuna})$



**Note:** Variables in different clauses are different (even if we use the same name for them)

## Ontology for sets

- **Ontology for set theory** (vocabulary of object constants, predicates, functions)
  - » Object constants:
    - Empty set:  $\{\}$
    - Set elements:  $A, B, C, \dots$
  - » Predicates:  $\text{Set}^1$ ,  $\text{Member}^2(\epsilon)$ ,  $\text{Subset}^2(\subset)$ ,  $\text{Equal}^2(=)$
  - » Functions:  $\text{adjoin}^2(\{\mid\})$ ,  $\text{union}^2(\cup)$ ,  $\text{intersection}^2(\cap)$

- **Axioms of set theory:**

1.  $\forall s [\text{Set}(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s') [\text{Set}(s') \wedge (s = \{x \mid s'\})]]$
2.  $\neg (\exists x, s) [\{x \mid s\} = \{\}]$
3.  $(\forall x, s) [x \in s \Leftrightarrow s = \{x \mid s\}]$
4.  $(\forall x, s) [x \in s \Leftrightarrow (\exists y, s') [s = \{y \mid s'\} \wedge (x = y \vee x \in s')]]$
5.  $(\forall s_1, s_2) [s_1 \subset s_2 \Leftrightarrow (\forall x) [x \in s_1 \Rightarrow x \in s_2]]$
6.  $(\forall s_1, s_2) [s_1 = s_2 \Leftrightarrow (s_1 \subset s_2 \wedge s_2 \subset s_1)]$
7.  $(\forall x, s_1, s_2) [x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)]$
8.  $(\forall x, s_1, s_2) [x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)]$

# Lists

- Ontology for lists
  - » Object constants: Nil, A, B, C... (atoms)
  - » Predicates: Find<sup>2</sup>, Atom<sup>1</sup>, Listp<sup>1</sup>
  - » Functions: cons<sup>2</sup>, append<sup>2</sup>, first<sup>1</sup>, rest<sup>1</sup>
- Axioms?

# Natural numbers

- Ontology for natural numbers
  - » Object constant: 0
  - » Predicates
    - Natural number test: NatNum<sup>1</sup>
    - Equality predicate: Equal<sup>2</sup> (=)
  - » Functions
    - Successor function: succ<sup>1</sup>
    - Addition function: sum<sup>2</sup> (+)
- Peano axioms
  - NatNum(0)
  - $(\forall n) [\text{NatNum}(n) \Rightarrow \text{NatNum}(\text{succ}(n))]$
  - $(\forall n) [0 \neq \text{succ}(n)]$
  - $(\forall m, n) [m \neq n \Rightarrow \text{succ}(m) \neq \text{succ}(n)]$
  - $(\forall n) [\text{NatNum}(n) \Rightarrow \text{sum}(n, 0) = n]$
  - $(\forall m, n) [\text{NatNum}(m) \wedge \text{NatNum}(n) \Rightarrow \text{sum}(\text{succ}(m), n) = \text{succ}(\text{sum}(m, n))]$

Note: The principle of **induction** can only be formulated in **second order logic**, where first-order logic relations and functions are viewed as objects and therefore, one can make assertions about them (e.g. one can write a second-order predicate that specifies when a first-order relation is transitive).

## Equality predicate

- **Equality:** Special predicate that allows to express that two different terms refer to the same real world object.
  - »  $\text{Equal}(\text{fatherOf}(\text{John}), \text{Peter})$   
[also:  $\text{fatherOf}(\text{John})=\text{Peter}$  ]  
Meaning: “Peter is the father of John”
  - »  $(\exists x, y)[\text{Brother}(x, \text{Richard}) \wedge \text{Brother}(y, \text{Richard}) \wedge \neg \text{Equal}(x, y)]$   
(also:  $(\exists x, y)[\text{Brother}(x, \text{Richard}) \wedge \text{Brother}(y, \text{Richard}) \wedge (x \neq y)]$ )  
Meaning: “Richard has (at least) two brothers”
- **Axiomatization of equality**
  - » **Reflexive:**  $(\forall x)[x=x]$
  - » **Symmetric:**  $(\forall x, y)[x=y \Rightarrow y=x]$
  - » **Transitive:**  $(\forall x, y, z)[[x=y] \wedge [y=z] \Rightarrow x=z]$
  - » **Substitution rules:** Equals can be substituted for equals in relations and functions.  
 $(\forall x, y)[x=y \Rightarrow f_i^{(1)}(x)=f_i^{(1)}(y)]; i=1, 2, \dots$   
 $(\forall x, y)[x=y \Rightarrow P_i^{(1)}(x)=P_i^{(1)}(y)]; i=1, 2, \dots$   
 $(\forall x, y, z, t)[(x=z) \wedge (y=t) \Rightarrow f_i^{(2)}(x, y)=f_i^{(2)}(z, t)]; i=1, 2, \dots$   
 $(\forall x, y, z, t)[(x=z) \wedge (y=t) \Rightarrow P_i^{(2)}(x, y)=P_i^{(2)}(z, t)]; i=1, 2, \dots$   
 .....

## Paramodulation

- In complex domains, it is **impractical** to specify all axioms that correspond to **substitution rules** for equality.
- **Paramodulation: Specific inference rule** in KB's that include the **equality predicate**. Used in combination with **resolution** forms a complete inference set of rules  
Given  $K_1 = \{\lambda(\tau) \cup K_1'\}$ ;  $K_2 = \{(\alpha = \beta) \cup K_2'\}$ , where  $\alpha, \beta, \tau$  are terms;  $K_1, K_2, K_1', K_2'$  are clauses;  $\lambda(\tau)$  is a literal that contains (among others), term  $\tau$ . If  $\alpha, \tau$  have a most general unifier  $\mu$ , then infer the binary **paramodulant**  
 $\lambda\mu(\beta\mu) \cup K_1'\mu \cup K_2'\mu$   
 where  $\lambda\mu(\beta\mu)$  is the result of replacing a single occurrence of  $\tau\mu$  in  $\lambda\mu$  by  $\beta\mu$ .  
 [Symmetric rule with the roles of  $\alpha, \beta$  reversed]

Example 1: Does  $P(A) \wedge (A=B) \models P(B)$ ?  
 Paramodulant[ $P(A), (A=B)$ ] =  $P(B)$

Example 2:  
 Paramodulant[ $P(g(f(x)) \vee Q(x), (f(g(B))=A) \vee R(g(C)))$ ]  
 =  $P(g(A)) \vee Q(g(B)) \vee R(g(C))$

# Evaluation of expressions

---

---

- For some predicates and functions, using **evaluation of expressions** containing **ground terms** instead of inference can be a simple way to make the reasoning process more efficient.
  - » Equality predicate:  
Evaluate  $\neg(2222=4444)$
  - » Boolean comparisons:  
Evaluate  $(2222 \leq 4444)$
  - » Arithmetic functions:  
Evaluate  $(2222+4444)$
- In inference with wff's in CNF
  - » Clauses that evaluate to
    - `False` cause the inference process to terminate (with a contradiction)
    - `True` can be eliminated from the Knowledge Base.
  - » Literals that evaluate to
    - `False` can be eliminated from the clause.
    - `True` allow us to eliminate the clause from the Knowledge Base.