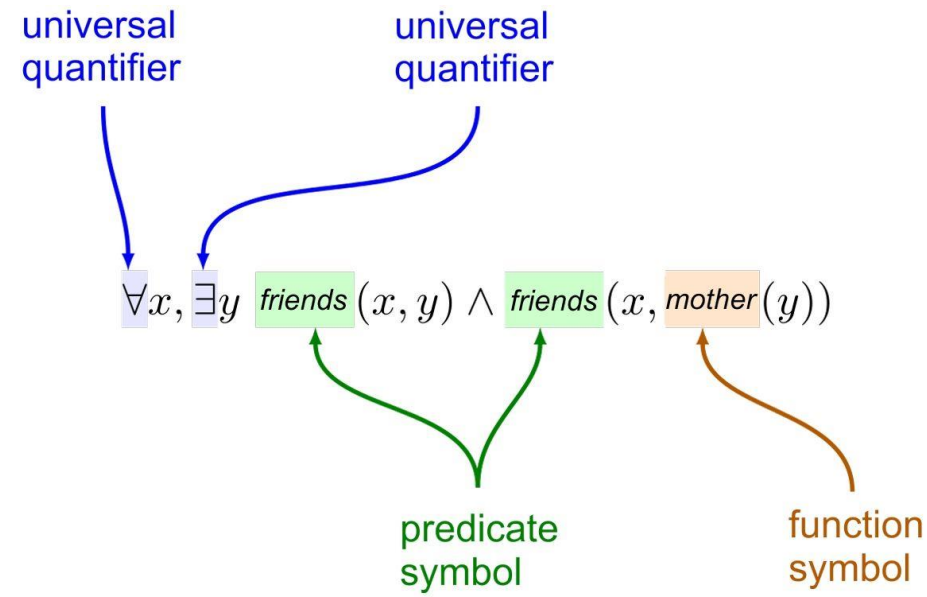


# 3. Predicate logic

Lara Quijano Sánchez



# Lesson's Scheme

## □ 3. Predicate logic

### □ 3.1 Elements of predicate logic

- 3.1.1. Common elements with propositional logic
- 3.1.2. Variables and quantifiers
- 3.1.3. Predicates
- 3.1.4. Features
- 3.1.5. Atoms, terms, literals and clauses
- 3.1.6. Normal shapes

### □ 3.2 Substitution and unification

### □ 3.3 Inference in predicate logic

- 3.3.1 Generalized rules of inference
  - 3.3.1.1 Modus ponens
  - 3.3.1.2 Resolution
- 3.3.2 Extraction of responses using Green's trick.

### □ 3.4 The equality predicate

### □ 3.5 Forward and backward chaining

# Readings

- ❑ CHAPTERS 8,9 of Russell & Norvig

- ❑ CHAPTERS 15,16 of 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)

# Knowledge representation and inference

- ❑ Knowledge representation:
  - ❑ Something broader and more flexible than a heuristic
  - ❑ We have seen heuristics to guide search algorithms
  - ❑ Heuristics are a too simple type of knowledge representation:
    - ❑ Insufficient to represent the complexity of real systems
  - ❑ Problem solving in complex domains requires more general and flexible representations
- ❑ Knowledge can be represented in multiple ways
- ❑ Choose a formalism that allows us to adequately represent certain facts
  - ❑ Knowledge representation formalism
    - ❑ What information do I have?
  - ❑ Associated reasoning or inference mechanisms
    - ❑ What can I do (deduce) with the information I have?

# Types of Representation Techniques

- ❑ Various formalisms to build knowledge bases
  - ❑ Relationship-based representations
    - ❑ Logic
    - ❑ Semantic networks
  - ❑ Object-based representations
    - ❑ Frames
    - ❑ Object-Attribute-Value
  - ❑ Action-based representations
    - ❑ Production systems (or Rules)
    - ❑ Scripts
  - ❑ Combinations and modifications of the above

# Representation and reasoning techniques

## ❑ Inference mechanisms

- ❑ Obtaining new knowledge from current knowledge

### ❑ Deduction

- ❑ From general laws we obtain particular knowledge

- ❑ If the premises are true, the conclusion is true.

### ❑ Induction

- ❑ It is the generalization of the information extracted from particular cases.

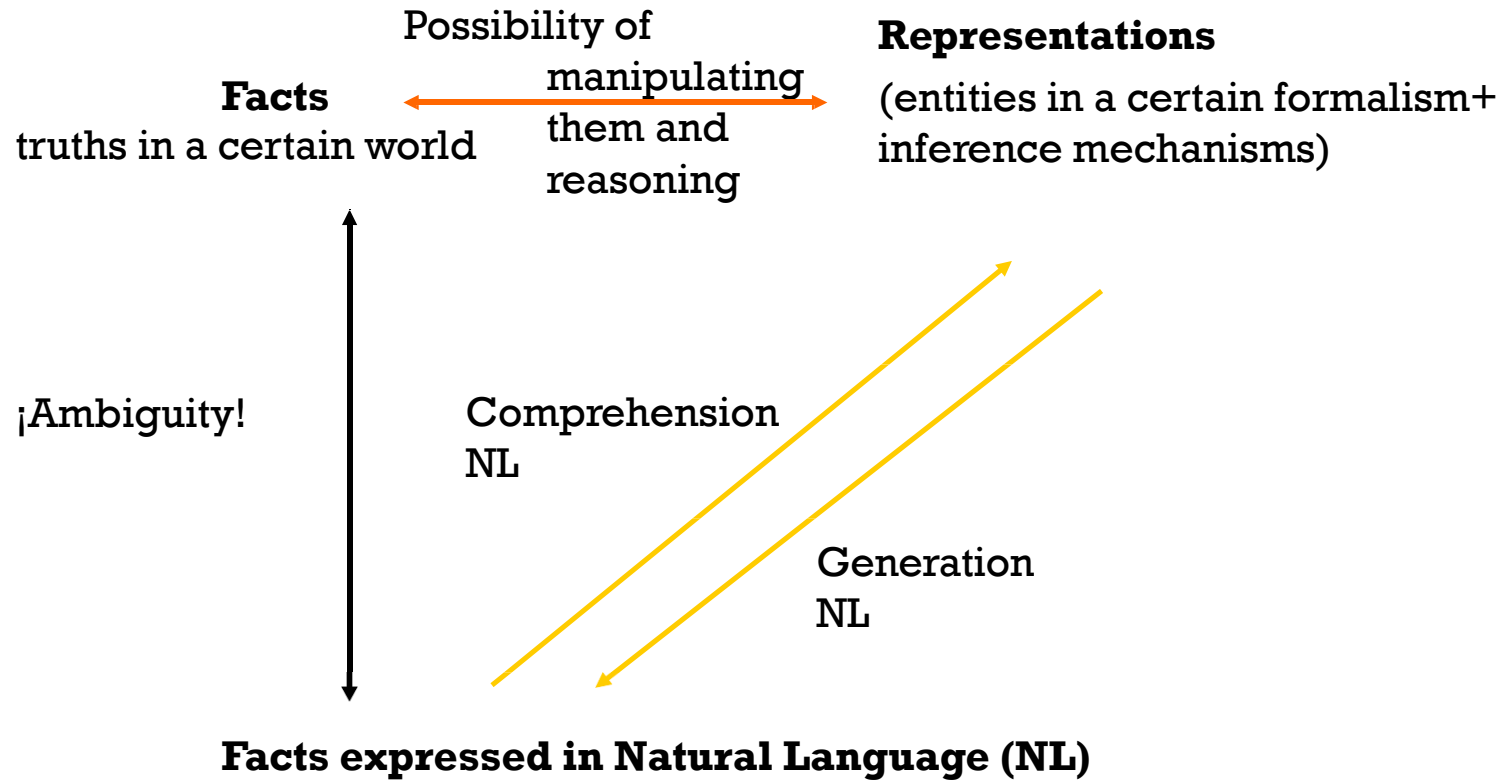
- ❑ Its validity cannot be guaranteed.

- ❑ It is the basis of learning

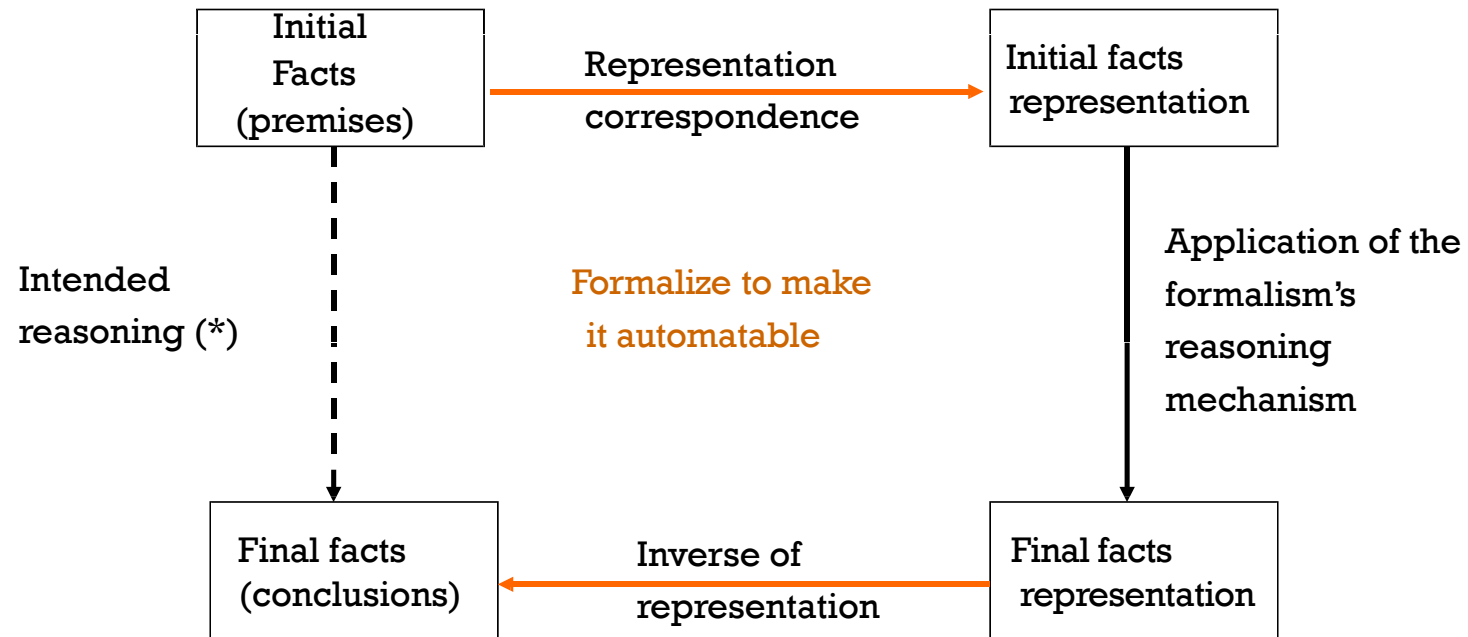
### ❑ Abduction

- ❑ It is the ability to generate plausible explanations for a certain event that has occurred

# Representation Role: Example



# Participating elements





# Properties of a representation

- ❑ For a certain domain you have to value (+ or -)
  - ❑ **Representative suitability**: ability to represent all types of knowledge required in that domain
  - ❑ **Inferential suitability**: ability to manipulate the symbols of the representational formalism and infer new (desired) knowledge
  - ❑ **Inferential efficiency**: ability to incorporate meta-knowledge that allows to improve reasoning processes
  - ❑ **Purchasing efficiency**: the ability to easily acquire new knowledge from the outside, ideally under the control of the system itself (or simply by adding it by one person) while keeping consistency with the existing knowledge
- ❑ No representation technique optimizes all of these properties for all domains and types of knowledge
  - ❑ A multitude of techniques. Many systems based on more than one

# Types of knowledge

## Knowledge bases (KB)

The entire body of knowledge usable by the system, represented in some given formalism, together with the management mechanisms of that knowledge (incorporation, deletion, modification, exact query, approximate query, inference, consistency control, etc.)

## □ Types of knowledge

### □ Factual or declarative (representation of facts)

- **Explicit**: is entered directly

- **Implicit**: inferred from explicit knowledge

### □ Procedural

- Indicates how to act through steps in various situations

### □ Meta-knowledge or control knowledge

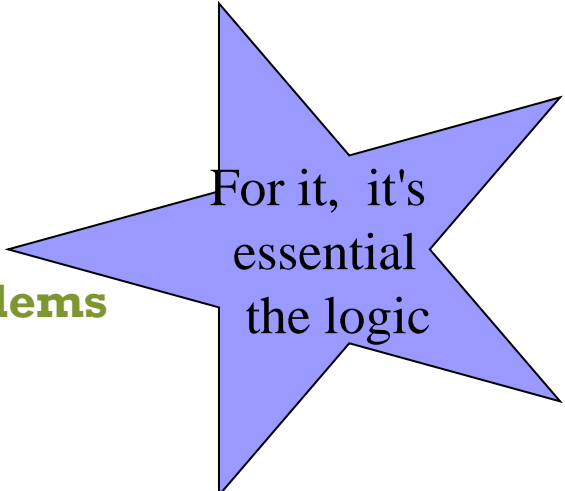
- Knowledge at a higher level: knowledge about one's own knowledge, which allows it to be managed more efficiently

# Goal: the knowledge appears explicitly and conclusions are reached from the stated knowledge

- ❑ Learn to design agents that:
  - ❑ Build interpretations of the world
  - ❑ Derive new representations of the world by inference
  - ❑ Use those new representations to know what to do
- ❑ Knowledge representation:

**The intended role of knowledge representation in AI is to reduce intelligent action problems into mere Search problems**

– *Grinsberg*



For it, it's  
essential  
the logic

## Analogy between programming and AI problems

### Programming

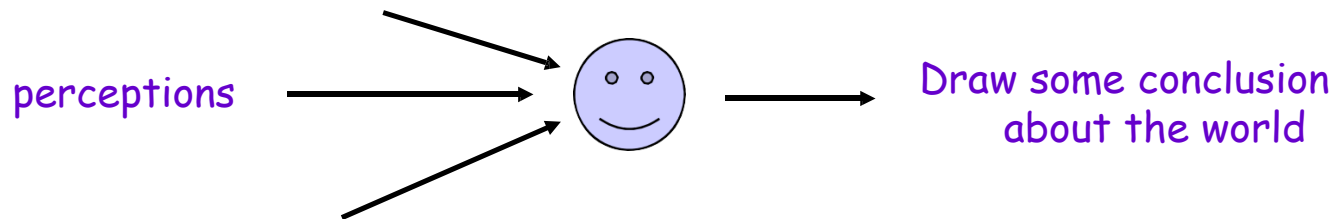
1. create an algorithm to solve the problem
2. Select a programming language to code the task
3. Capture the algorithm as a program
4. Run the program

### Artificial intelligence

1. Identify the knowledge necessary to solve the problem
2. Select the language with which said knowledge can be represented
3. Write knowledge within that language
4. Use the consequences of knowledge to solve the problem

# What is a logic?

- ❑ A logic is a formal language
  - ❑ It has a **syntax** that determines which expressions are legal (the shape)
  - ❑ It also has **semantics** that determine what do legal expressions represent (the content)
  - ❑ And it usually has an **inference system** that allows manipulating syntactic expressions to obtain other syntactic expressions
    - ❑ For what purpose?
      - ❑ Obtain expressions with an "interesting" meaning
      - ❑ Tell us something "new" about the world



# Logic

## □ ontological commitment

- for the agent, what exists in the world

- in the case of propositional logic, for the agent there are facts that will be true or false

## □ epistemological commitment

- for the agent, what is the attitude towards the facts

- in the case of propositional logic, the agent believes that a sentence is true or false, or has not reached any conclusion

# Types of logics and their concerns

Language	Ontology (what exists)	Epistemology (what do you think of the facts)
Propositional Logic	facts	true / false / don't know
First order logic (FOL)	facts, objects, links	true / false / don't know
Temporal logic	facts, objects, links, time	true / false / don't know
Probability logic	facts	degree of certainty
Fuzzy logic	degree of truth	degree of certainty

# 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.  $(\forall x) \text{Human}(x) \Rightarrow \text{Mortal}(x)$

- **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”

$(\forall x, y) \quad (\text{Son}(x, y) \Leftrightarrow \text{Child}(x, y) \wedge \text{Male}(x))$

## □ **General statements:**

“All of Mary’s children are married, but are unhappy”

$(\forall x) \quad (\text{Child}(x, \text{Mary}) \Rightarrow \text{Married}(x) \wedge \neg \text{Happy}(x))$

## □ **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: a term.

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

### □ Relation constants or predicates (usually upper case)

Input arguments: list of terms between parentheses

Evaluates: 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.

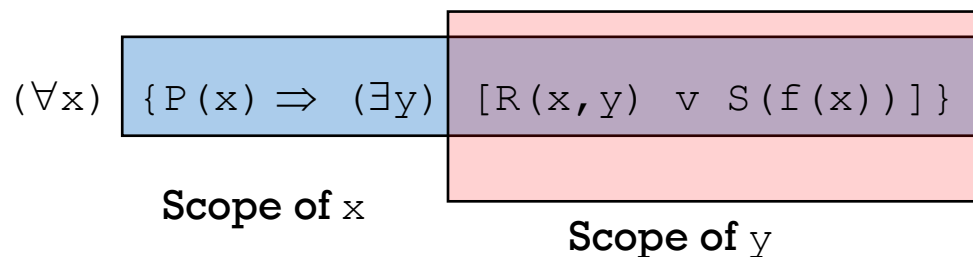
e.g.  $x$  (reals),  $n$  (integer),  $p$  (persons), etc.

E.g.  $x, y, p, q, \dots, p_1, p_2, \dots$

## □ 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**

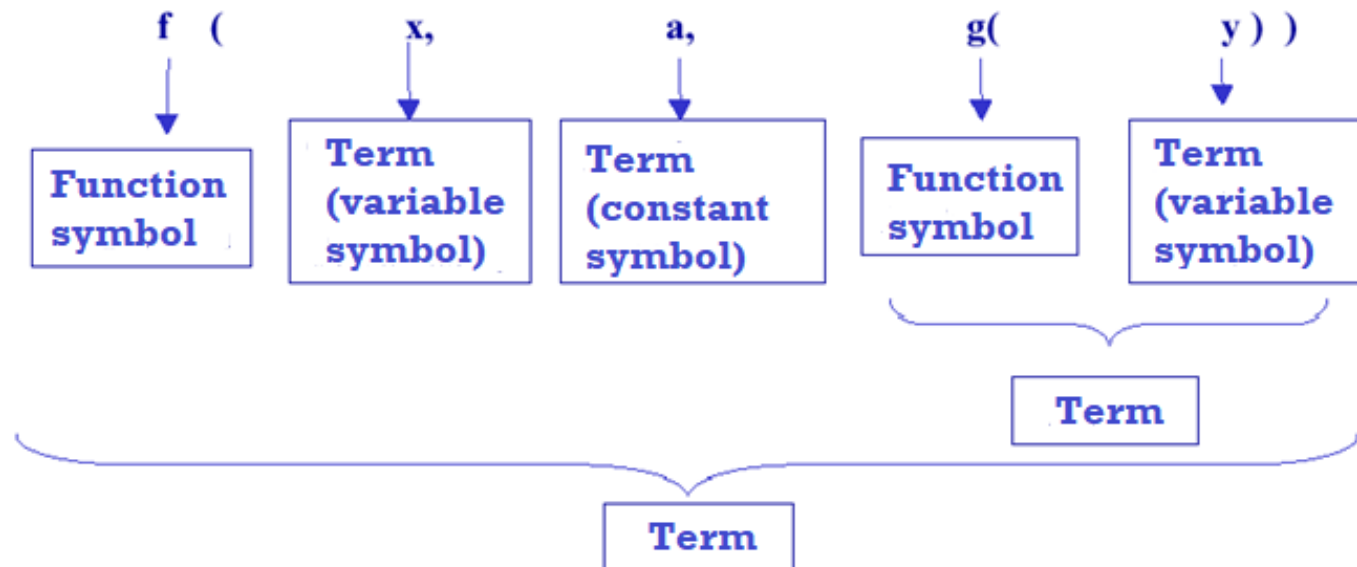
E.g.  $\neg$  Parent(John, fatherOf(Peter))

# Predicate logic: syntax

## □ Terms

- A **constant** symbol is a term  $(a, b, c \dots)$
- A **variable** symbol is a term  $(x, y, z \dots)$
- If  $f$  is a function symbol of arity  $n$ , and  $t_1, t_2, \dots, t_n$  are terms, so  $f(t_1, t_2, \dots, t_n)$  is a **composed** term

## □ Example: $f(x, a, g(y))$



# Predicate logic: syntax

## Formulas

ATOMIC

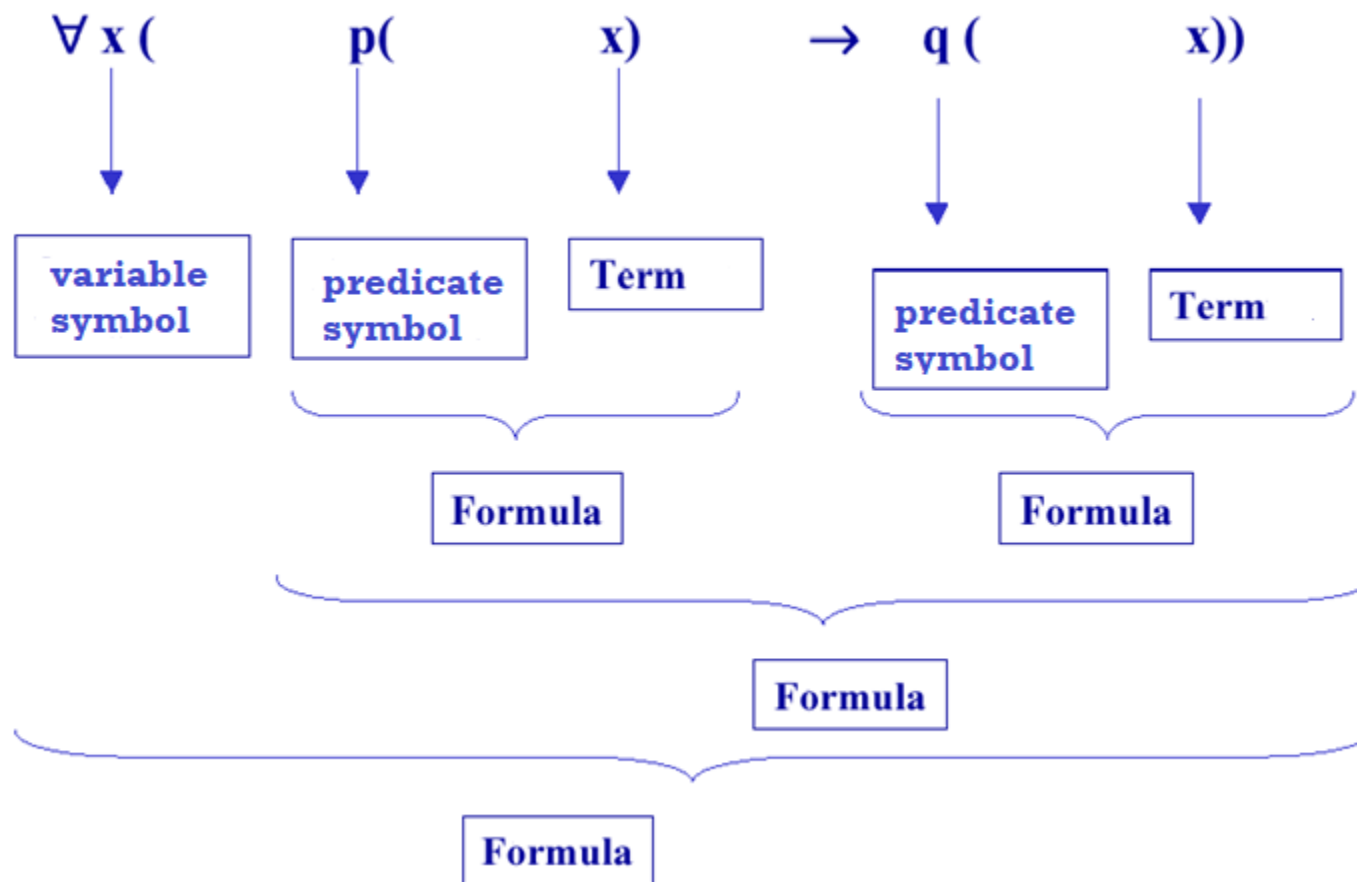
- Truth symbols  $\top$  and false or contradiction symbols  $\perp$  are formulas
- If  $p$  is a predicate symbol of arity  $n$ , and  $t_1, t_2, \dots, t_n$  are terms, then  $p(t_1, t_2, \dots, t_n)$  is a formula
- If  $F$  is a formula, then  $\neg F$  is a formula

COMPOSED

- If  $F$  and  $G$  are formulas, then:
  - $(F \wedge G)$  is a formula
  - $(F \vee G)$  is a formula
  - $(F \rightarrow G)$  is a formula
  - $(F \leftrightarrow G)$  is a formula
- If  $x$  is a variable symbol, and  $F$  is a formula, then:
  - $\forall x F$  is a formula
  - $\exists x F$  is a formula

# Predicate logic: syntax

□ Example formula:  $\forall x (p(f(x)) \rightarrow q(x))$



# Interpretation and semantics

## □ Constants

- **Constants** are **symbolic objects** that correspond to **objects in the real world**
- The **predicates of arity 1** correspond to **properties of the object**
- **Predicates of arity  $n > 1$**  express **relationships between objects**

## □ Variables

- The **domain** of a variable is the set of symbolic objects among which said variable can take value
- **Assignment**: Operation by which a variable that appears in an **wff** (well formed formula) is replaced by a particular value within its domain.

## □ Quantifiers

- **Universal quantifier**:  $(\forall x) w(x)$  has a **True** value if  $w(x)$  has a **True** value for all possible assignments of the variable  $x$
- **Existential quantifier**:  $(\exists x) w(x)$  has **True** value if  $w(x)$  has **True** value for any of the possible assignments of  $x$

# Predicate logic: semantics

- The meanings of terms and formulas are obtained by setting an interpretation  $I$  consisting of
  - A set called  $U$  universe (or domain) of discourse
  - An association of elements of  $U$  to constant symbols
  - An association of  $U$  relations to predicate symbols
  - An association of functions in  $U$  to function symbols



# Predicate logic: semantics

## Constants

- **a**
- **b**
- **c**

First order logic

SETS

Universe of discourse

marker

eraser

pencil

## Predicate symbols

- **h**
- **m**
- **t**

fungible = {marker, eraser, pencil}

material = {marker, eraser, pencil}

boughtBefore = {(marker, eraser), (eraser, pencil)}

## Formulas

$\forall x (h(x) \rightarrow m(x))$

fungible material

$\exists x \exists y ((h(x) \wedge h(y) \wedge t(x,y)))$

boughtBefore  $\neq \emptyset$  in the domain of fungible

$h(a) \wedge h(b) \wedge h(c)$

marker, eraser, pencil  $\in$  fungible

# Free and Bound Variables

- The appearance of a variable  $X$  within a formula  $\varphi$  is said to be **bound** if it is **affected** by a **quantifier**, or more precisely:  $X$  appears bound in  $\varphi$  if it is within a sub-formula of  $\varphi$  of the form  $\forall\psi$  or  $\exists\psi$ .
- The appearance of  $X$  in  $\varphi$  is free if it is not bound, i.e., if it is not affected by any quantifier.

$$\exists X \underbrace{p(X)}_{\text{bound}}, \underbrace{f(Y)}_{\text{free}}) \vee \forall Y \underbrace{q(Y)}_{\text{bound}}, \underbrace{g(X)}_{\text{free}})$$

- A variable  $X$  is free in  $\varphi$  if all occurrences of  $X$  in  $\varphi$  are free.
- A variable  $X$  is bound in  $\varphi$  if any of the occurrences of  $X$  in  $\varphi$  is bound.

# Free and Bound Variables

- ❑ Can the same variable appear bound and free in the same formula?

$$\varphi \equiv \forall X \, r(\underbrace{X}_{\text{bound}}, \underbrace{Y}_{\text{free}}) \wedge \forall Y \, p(\underbrace{X}_{\text{free}}, \underbrace{Y}_{\text{bound}}, Z)$$

- ❑ To avoid confusion, rename:

$$\varphi' \equiv \forall X' \, r(X', Y) \wedge \forall Y' \, p(X, Y', Z)$$

- ❑ Are  $\varphi$  and  $\varphi'$  the same?
- ❑ "Equal" not ... it seems they are equivalent: they have the same meaning, but to see it we would have to give semantics or meaning to the formulas.
- ❑ We will be especially interested in the semantics of sentences: closed formulas, i.e., without free variables.

# Predicate logic: properties

## ❑ Declarative representation

- ❑ The way in which knowledge should be used is not fixed

  - ❑ Easy to incorporate new knowledge, or eliminate it

  - ❑ Simple integration of two or more KBs

## ❑ Correction and completeness

- ❑ It has correct and complete deductive mechanisms

  - ❑ They allow to deduce new knowledge (conclusions) from the starting knowledge (premises)

  - ❑ They can be used to answer questions or solve problems

    - ❑ Automatic theorem proof: one of the initial areas of AI

## ❑ The deduction is semi-decidable in FOL

- ❑ If what we intend to demonstrate is not deduced from the content of the knowledge base, it is not guaranteed that the demonstration process will complete

  - ❑ There is no decision procedure, not even exponential cost

# Difficulties in representation

- ❑ Natural language → predicate logic
  - ❑ Ambiguity, common sense, implications and disjunctions ...
- ❑ Incompleteness of knowledge represented
  - ❑ It is impossible to represent **ALL** knowledge
  - ❑ We represent only the most important
  - ❑ We do not include knowledge of "common sense" → there will be reasonings that cannot be made
- ❑ Infinity of representation alternatives within logic
  - ❑ They influence the reasoning process. Some representations facilitate one type of reasoning and hinder another
  - ❑ The important thing is to use a consistent representation within the same KB
- ❑ Difficulties in the representation of knowledge by default
  - ❑ For example, exceptions to inheritance

# Representation alternatives (i)

## □ Example

### □ Terminological factual knowledge representation

□ Instance ( $\in$ ) and subclass ( $\subseteq$ ) relationships

□ Essential properties

## □ Advantages and disadvantages of different alternatives

□ "Flipper is a dolphin"

*exemplary of a kind*

□ "All dolphins are vertebrates"

*subclass relationship*

□ "Vertebrates have skeletons"

*essential property*

## □ Criteria for choice

### □ What can they do efficiently

□ Represent what we have

□ Get what we are looking for (deductions, inferences)

□ Our own tastes

□ Representation is consistent across the domain

# Representation alternatives (ii)

## ❑ Version 1

- ❑ **Implicit** representation of **instances**: the **class** name is a unary predicate symbol, the argument is the instance

### **Dolphin (flipper)**

- ❑ Representation of the dolphin **subclass** of the vertebrate class

$\forall x (\text{Dolphin}(x) \rightarrow \text{Vertebrate}(x))$

- ❑ Representation of **essential properties**

$\forall x (\text{Vertebrate}(x) \rightarrow \text{Has-Skeleton}(x))$

### ❑ **Advantages**

- ❑ Simplicity of representation

### ❑ **Disadvantages**

- ❑ For any class (Dolphin, Elephant ...)
  - ❑ create specific predicates and rules (subclasses)
- ❑ General reasoning is complicated
- ❑ Implicit knowledge is deduced through implications.
  - ❑ It is better to state things explicitly by means of facts: VERSION 2

# Representation alternatives (iii)

## □ Version 2

□ **Explicit** representation of **instance** membership to **classes**: a binary predicate symbol  $\text{Is\_A}(\text{instance}, \text{Class})$  is used

**$\text{Is\_A}(\text{flipper}, \text{Dolphin})$**

□ Representation of the dolphin **subclass** of the vertebrate class

**$\forall x (\text{Is\_A}(x, \text{Dolphin}) \rightarrow \text{Is\_A}(x, \text{Vertebrate}))$**

□ Representation of **essential properties**

**$\forall x (\text{Is\_A}(x, \text{Vertebrate}) \rightarrow \text{Has\_Skeleton}(x))$**

## □ Advantages

□ Explicit representation of class membership,  $\text{Is\_A}$

## □ Disadvantages

□ The subclass relationship remains implicit



# Representation alternatives (iv)

## □ Version 3

□ Explicit representation of instances and subclasses using pred Is\_A

Is\_A(flipper, Dolphin)

Is\_A(Dolphin, Vertebrate)

$\forall x (\text{Is\_A}(x, \text{Vertebrate}) \rightarrow \text{Has\_Skeleton}(x))$

## □ Advantages

□ Explicit representation of class membership

□ Explicit representation of the subclass relationship

## □ Disadvantages

□ The system cannot differentiate if flipper is individual or class

□ It cannot be deduced that Flipper has a skeleton

□ It remains to specify the transitivity of the relation Is\_A. We have to add it

$\forall x \forall y \forall z (\text{Is\_A}(x, y) \wedge \text{Is\_A}(y, z) \rightarrow \text{Is\_A}(x, z))$

# Representation alternatives (v)

## □ Version 4

- Explicit representation of instances and subclasses using two different binary predicate symbols

**Specimen (flipper, Dolphin)**

**Subclass (Dolphin, Vertebrate)**

$\forall x (\text{Exemplary}(x, \text{Vertebrate}) \rightarrow \text{Has\_Skeleton}(x))$

- We add the transitivity (between subclasses and instances)

$\forall x \forall y \forall z (\text{Exemplary}(x, y) \wedge \text{Subclass}(y, z) \rightarrow \text{Exemplary}(x, z))$

$\forall x \forall y \forall z (\text{Subclass}(x, y) \wedge \text{Subclass}(y, z) \rightarrow \text{Subclass}(x, z))$

# Representation alternatives (vi)

- ❑ The four versions use predicate logic or first-order logic (FOL) as a representation technique
  - ❑ There are different possibilities of representation within logic (in general, this happens with any formalism)
  - ❑ Some representations facilitate one type of reasoning and hinder another

# Conclusion: Predicate logic allows us to express knowledge (i)

□ We have seen how predicate logic allows expressing “sentences” of natural language:

□ Eg: "Some mammals read Quevedo"

□ We define

$PS = \{Ismammal^1, ReadQuevedo^1\}, FS = \emptyset;$

with the meaning "intuitive":

$Ismammal(X) ::= X$  is a mammal

$ReadQuevedo(X) ::= X$  reads Quevedo

□ And we formalize:

$\exists X, (Ismammal(X) \wedge ReadQuevedo(X))$

□ In another way:

□ We define

$PS = \{Ismammal^1, Read^2\}, FS = \{Q\};$

with the meaning "intuitive":

$Ismammal(X) ::= X$  is a mammal

$Read(X, Y) ::= X$  read  $Y$

$Q ::= Quevedo$

□ And we formalize:

$\exists X, (Ismammal(X) \wedge Read(X, Q))$

# Conclusion: Predicate logic allows us to express knowledge (ii)

□ Another example: "The product of any number  $n$  by 1 is  $n$ " (This is a "mathematical" sentence, but it is expressed in natural language).

□ We define

$$SP = \{=^2\}, SF = \{1^0, *^2\}$$

with the intuitive meaning

$= ::=$  binary predicate of equality (to be equal)

$1 ::=$  the "one" of math

$* ::=$  the "product" in math

□ And we formalize

$$\forall X, X * 1 = X$$

□ This example shows that mathematics is "steeped" in logic (logic can express mathematical properties ... it can even express things about logic itself!).

# Normal forms in predicate logic

- ❑ To manipulate predicates automatically (on a computer) we want them to have a uniform format.
- ❑ It is also interesting that this format is as simple as possible
- ❑ The underlying justification is that we need to have them in **normal form** in order to be able to use the resolution mechanism (which we will see later).
  - ❑ In particular, the clause form only uses the connectives  $\neg$  and  $\vee$  (negation and disjunction), and we only need a single deduction rule
- ❑ With this restriction, generality is not lost because there is an algorithm for conversion to conjunctive normal form that allows to pose any logical problem in the form of clauses

# Well-formed formulas

- A **wff** is constructed using atoms and connectives in the same way as in propositional logic.
- 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**

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

$\text{Father}(x, y)$  “ $x$  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”

# Equivalence and inference rules

## □ Equivalence rules

### □ Equivalence rules of propositional logic.

$$\square \neg (\forall x) w(x) \equiv (\exists x) \neg w(x)$$

$$\square \neg (\exists x) w(x) \equiv (\forall x) \neg w(x)$$

### □ Renaming quantized variables (dummy variables). The new name must be different from that of the other variables in the wff.

$$\square (\forall x) w(x) \equiv (\forall y) w(y)$$

$$\square (\exists x) w(x) \equiv (\exists y) w(y)$$

## □ Inference rules

### □ Inference rules of propositional logic.

#### □ Universal instantiation (UI)

[Sound/correct]

$(\forall x) w(x) \vdash_{UI} w(A)$ , where  $A$  is any symbolic object constant in the domain of  $x$ .

#### □ Existential generalization (EG)

[Sound/correct]

$w(A) \vdash_{EG} (\exists x) w(x)$ , where  $x$  is variable symbol whose domain is the domain to which the object constant  $A$  belongs



# Lets remember Equivalence rules

□ Two **different WFFs**  $w_1, w_2$  are **equivalent** ( $w_1 \equiv w_2$ ) when they have the **same truth table**

□ **Neutral element** :  $(w_1 \wedge V) \equiv w_1; (w_1 \vee F) \equiv w_1$

□ **Absorption Laws**:  $(w_1 \vee (w_1 \wedge w_2)) \equiv w_1$

$(w_1 \wedge (w_1 \vee w_2)) \equiv w_1$

□ **Contradiction laws / law of the excluded middle** :

$(w_1 \wedge \neg w_1) \equiv F; (w_1 \vee \neg w_1) \equiv V$

□ **Domination Laws**:  $(w_1 \wedge F) \equiv F; (w_1 \vee V) \equiv V$

□ **Idempotency** :  $(w_1 \wedge w_1) \equiv w_1; (w_1 \vee w_1) \equiv w_1$

□ **Elimination of double negation** :  $\neg \neg w_1 \equiv w_1$

□ **De Morgan Laws**:  $\neg(w_1 \vee w_2) \equiv \neg w_1 \wedge \neg w_2; \neg(w_1 \wedge w_2) \equiv \neg w_1 \vee \neg w_2$

□ **Commutativity** :  $w_1 \vee w_2 \equiv w_2 \vee w_1; w_1 \wedge w_2 \equiv w_2 \wedge w_1$

□ **Associative laws** :

$(w_1 \wedge w_2) \wedge w_3 \equiv w_1 \wedge (w_2 \wedge w_3) \equiv w_1 \wedge w_2 \wedge w_3$  [conjunction]

$(w_1 \vee w_2) \vee w_3 \equiv w_1 \vee (w_2 \vee w_3) \equiv w_1 \vee w_2 \vee w_3$  [disjunction]

□ **Distributive laws**:

$w_1 \wedge (w_2 \vee w_3) \equiv (w_1 \wedge w_2) \vee (w_1 \wedge w_3)$

$w_1 \vee (w_2 \wedge w_3) \equiv (w_1 \vee w_2) \wedge (w_1 \vee w_3).$

□ **Definition of conditional** :  $w_1 \Rightarrow w_2 \equiv \neg w_1 \vee w_2$

□ **Opposed** :  $w_1 \Rightarrow w_2 \equiv \neg w_2 \Rightarrow \neg w_1$

□ **Definition of biconditional**:  $w_1 \Leftrightarrow w_2 \equiv (w_1 \Rightarrow w_2) \wedge (w_2 \Rightarrow w_1) \equiv (w_1 \wedge w_2) \vee (\neg w_1 \wedge \neg w_2)$

# 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 whose scope includes the scope of the existential quantifier that is being eliminated.

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

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

$Sk$  is an object whose identity we do not know, but which we know exists

- 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.

Eg. "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 their 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 if their Skolem form is satisfiable.
- A set of wffs is unsatisfiable if their Skolem form is unsatisfiable.

# Lets remember: models and satisfiability

- Given an interpretation  $I = (A, v)$ , if  $[[\varphi]]^A_v = \text{true}$  we say:
  - $A$  **satisfies**  $\varphi$  in state  $v$  or that  $I$  satisfies  $\varphi$ , or
  - $I$  is a **model** of  $\varphi$  and it we note it like:  $A \models \varphi$(In the case that  $[[\varphi]]^A_v = \text{false}$  it is said that  $I$  does not satisfy  $\varphi$  and is denoted as  $A \not\models \varphi$ )
- A formula is satisfactory if it admits one model, and unsatisfactory otherwise.
- Two formulas  $\varphi$  and  $\psi$  are logically equivalent (and it is written  $\varphi \approx \psi$ ) if  $[[\varphi]]^A_v = [[\psi]]^A_v$  for all interpretation  $I = (A, v)$ ,
- $\varphi$  is **logically valid** if  $I \models \varphi$  for all interpretation  $I$

# Lets remember: logical consequence

□ Given the formulas  $\varphi_1, \dots, \varphi_n$ ,  $\psi \in L_\Sigma$  is said that  $\psi$  is a **logical consequence** of  $\{\varphi_1, \dots, \varphi_n\}$  and we write it :

$$\underbrace{\{\varphi_1, \dots, \varphi_n\}}_{\text{Premises / hypotheses}} \models \underbrace{\psi}_{\text{conclusion}}$$

if for every interpretation  $I = (A, v)$  we have:

$$I \models \varphi_1, \dots \quad I \models \varphi_n \Rightarrow I \models \psi$$

# CNF in first order logic

**1. Eliminate implications**  $\Leftrightarrow, \Rightarrow$ :  $w_1 \Rightarrow w_2 \equiv \neg w_1 \vee w_2$  ;  $w_1 \Leftrightarrow w_2 \equiv (\neg w_1 \vee w_2) \wedge (\neg w_2 \vee w_1)$

**2. Reduce the scope of  $\neg$**

□ de Morgan's laws

$$\neg(w_1 \vee w_2) \equiv \neg w_1 \wedge \neg w_2$$

$$\neg(w_1 \wedge w_2) \equiv \neg w_1 \vee \neg w_2$$

□ 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

$$\begin{aligned} & [(\forall x) [P(x) \Rightarrow R(x)]] \vee [(\exists x) P(x)] \\ & \equiv [(\forall x) [P(x) \Rightarrow R(x)]] \vee [(\exists y) P(y)] \end{aligned}$$

**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.

$$\begin{aligned} \text{wff in prenex form} = & \text{Prefix (string of quantifiers)} \\ & + \text{Matrix (quantifier-free formula)} \end{aligned}$$

**6. Drop universal quantifiers.**

**7. Use distributive laws and equivalence rules** of propositional logic to transform the **matrix to CNF** (turn it into a conjunction of disjunctions (**conjunctive normal form**)).

# 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.

# 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 (Apply Distributive law)

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



# Lets remember: Inference Rules

- **Inference rules:** Typographic rules that only manipulate symbols (and, therefore, do not use the meaning of these symbols, nor the truth tables) and that allow us to generate new WFFs from a given set of WFFs.
- **Correct inference rules:** Inference rules in which the models of the initial set of WFFs are also models of the generated WFFs.
- **Equivalence rules are correct inference rules.** (Note: not every inference rule is of equivalence)
- **Inference rule set:**

Let  $w_1, w_2$  be two WFFs

- (1) **Modus ponens:**  $\{w_1, w_1 \Rightarrow w_2\} \vdash_{M.P.} w_2$
- (2) **Modus tollens:**  $\{\neg w_2, w_1 \Rightarrow w_2\} \vdash_{M.T.} \neg w_1$
- (3) **Introduction of  $\wedge$ :**  $\{w_1, w_2\} \vdash_{\wedge INTRO} w_1 \wedge w_2$
- (4) **Commutativity of  $\wedge$ :**  $\{w_1 \wedge w_2\} \vdash_{\wedge CONMUTA} w_2 \wedge w_1$
- (5) **Elimination of  $\wedge$ :**  $\{w_1 \wedge w_2\} \vdash_{\wedge ELIMIN} w_1$
- (6) **Introduction of  $\vee$ :**  $\{w_1\} \vdash_{\vee INTRO} w_1 \vee w_2$   
 $\{w_2\} \vdash_{\vee INTRO} w_1 \vee w_2$
- (7) **Elimination of  $\neg\neg$ :**  $\{\neg\neg w_1\} \vdash_{\neg\neg ELIMIN} w_1$

□ This set of inference rules is correct, but not complete.

# Inference mechanisms

- Deduction: obtaining new knowledge (implicit)
- It is about knowing if a formula  $Q$  is true knowing:
  - The axioms that are logically valid whatever the meaning of the symbols (tautologies)  
 $\neg F \vee F$
  - Axioms that are valid only assuming certain meanings of symbols (explicit knowledge)
  - The rules of inference
    - For example:

*modus ponens*

$$\begin{array}{c} P \rightarrow Q \\ P \\ \hline Q \end{array}$$

*modus tollens*

$$\begin{array}{c} P \rightarrow Q \\ \neg Q \\ \hline \neg P \end{array}$$

# Formal deduction example

- Given a set of hypotheses or premises

$\text{dog (snowy)}$   
 $\forall x (\text{dog (x)} \rightarrow \text{animal (x)})$   
 $\forall y (\text{animal (y)} \rightarrow \text{mortal (y)})$

- Prove a conclusion

$\text{mortal (snowy)}$

- Steps applied

1. Apply universal instantiation with  $x = \text{snowy}$

$\text{dog (snowy)} \rightarrow \text{animal (snowy)}$

2. Apply modus ponens

$\text{animal (snowy)}$

3. Apply universal instantiation with  $y = \text{snowy}$

$\text{animal (snowy)} \rightarrow \text{mortal (snowy)}$

4. Apply modus ponens

$\text{mortal (snowy)}$

# Deduction by rebuttal

□ Knowledge base:  $T \leftarrow Q$ ,  $S \leftarrow T$ ,  $P \leftarrow S$ , **P**

□ Question: **Q?**

Step 1.

$$\begin{array}{c} Q \leftarrow T \\ \neg Q \\ \hline \neg T \end{array}$$

Step 2.

$$\begin{array}{c} T \leftarrow S \\ \neg T \\ \hline \neg S \end{array}$$

Step 3.

$$\begin{array}{c} S \leftarrow P \\ \neg S \\ \hline \neg P \end{array}$$

Step 4.

$$\begin{array}{c} \textbf{P} \\ \neg P \\ \hline \bullet \end{array}$$

□ Step 1. We deny what we intended to demonstrate

□ Steps 1, 2 and 3. Repeated application of modus tollens

□ Step 4. **P** and  $\neg P$  ~ a contradiction is reached

Refutation: reasoning by reduction to the absurd

$$\Phi \models \psi \Leftrightarrow \text{Insat } \Phi \cup \{\neg\psi\}$$

# Deduction by resolution

**Example:**  $\neg a$ ?

$$b \wedge c \rightarrow a$$

$$d \rightarrow c$$

$b$

$d$

**Clausal Form**

$$a \vee \neg b \vee \neg c$$

$$c \vee \neg d$$

$b$

$d$

Add the negation of what we want to prove:  $\neg a$  and **prove that we arrive to a contradiction** (unsatisfiable set)

*Resolution*

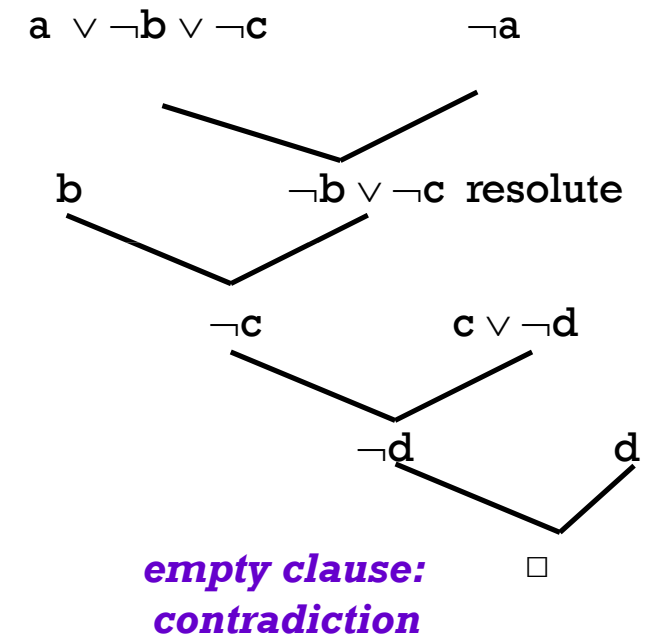
$\varphi \vee Q$  *a positive literal in a clause*

$\neg Q \vee \psi$  *and the same negative in another*

-----  
 $\varphi \vee \psi$  *resolute*

Propositions  $\rightarrow$   
Predicates

(Resolution with  
unification)



# Resolution with unification

$$\begin{array}{c} a(3) \vee b(y) \vee \neg c(z, y) \quad \neg a(w) \vee b(\text{'john'}) \vee c(w, \text{'mikel'}) \\ \swarrow \quad \searrow \\ w / 3 \\ b(y) \vee \neg (c(z, y)) \vee c(3, \text{'mikel'}) \end{array}$$

M.g.u.: The **most general possible unifier** of candidate literals Applies to parent clauses before calculating the resolvent . Variants of the clauses (fresh variables) must be considered

$$\begin{array}{c} a(3) \vee b(y) \vee \neg c(z, y) \quad \equiv \neg a(w) \vee b(\text{'john'}) \vee c(w, \text{'mikel'}) \\ \swarrow \quad \searrow \\ y / \text{'mikel'} \\ z / w \\ a(3) \vee b(\text{'mikel'}) \vee \neg a(w) \vee b(\text{'john'}) \end{array}$$

A single pair of complementary literals is selected

# Resolution strategies

- ❑ It is not easy to determine in each step which clauses to resolve and with which literals (very high degree of non-determinism)
  - ❑ If the choice is made systematically, the resolution will come to a contradiction if the set of clauses is unsatisfactory
  - ❑ So, it is a complete process, but it can take a long time ..
- ❑ Various strategies are used to speed up the process
  - ❑ Index the clauses by the predicates they contain, indicating whether or not they are negated, to facilitate their location
  - ❑ Eliminate tautologies ( $\varphi \vee \neg \varphi$ ) and clauses that are subsumed by others (are implied by others:  $\varphi \vee \psi$  is subsumed by  $\varphi$ )
  - ❑ Try to solve with one of the clauses of the formula that we are trying to refute, or with a clause that has been obtained by resolution from one of them (intuition: the contradiction has to come from there)
  - ❑ Solve with clauses that have only one literal (idea: decrease the size of the generated clauses)

# Answer questions

❑ Does flipper has a skeleton?

❑ There are two options to know if something is deducted or not from the KB

1. Prove  $\text{Has\_skeleton}(\text{flipper}) \rightarrow \text{add } \neg \text{Has\_skeleton}(\text{flipper})$
2. Prove  $\neg \text{Has\_skeleton}(\text{flipper}) \rightarrow \text{add } \text{Has\_skeleton}(\text{flipper})$

❑ The correct choice depends on how the predicates of the KB are

$\text{Specimen}(\text{flipper}, \text{Dolphin})$

$\text{Subclass}(\text{Dolphin}, \text{Vertebrate})$

$\forall x (\text{Exemplary}(x, \text{Vertebrate}) \rightarrow \text{Has\_Skeleton}(x))$

$\forall x \forall y \forall z (\text{Exemplary}(x, y) \wedge \text{Subclass}(y, z) \rightarrow \text{Exemplary}(x, z))$

$\forall x \forall y \forall z (\text{Subclass}(x, y) \wedge \text{Subclass}(y, z) \rightarrow \text{Subclass}(x, z))$



# Answer questions

□ Are there any dolphins?

□ Prove  $\exists x \text{ Specimen}(x, \text{Dolphin})$

→ add  $\neg \exists x \text{ Specimen}(x, \text{Dolphin})$  in clausal form

→  $\sim \forall x \neg \text{Example}(x, \text{Dolphin})$  in clausal form

→ add  $\neg \text{Example}(t, \text{Dolphin})$  variant

□ The correct choice depends on how the predicates of the KB are

$\text{Specimen}(\text{flipper}, \text{Dolphin})$

$\text{Subclass}(\text{Dolphin}, \text{Vertebrate})$

$\forall x (\text{Exemplary}(x, \text{Vertebrate}) \rightarrow \text{Has\_Skeleton}(x))$

$\forall x \forall y \forall z (\text{Exemplary}(x, y) \wedge \text{Subclass}(y, z) \rightarrow \text{Exemplary}(x, z))$

$\forall x \forall y \forall z (\text{Subclass}(x, y) \wedge \text{Subclass}(y, z) \rightarrow \text{Subclass}(x, z))$

□ If unification is required, the resolution will return the values that make the question true:  $\{t = \text{flipper}\}$

# From logic to logic programming

- Unification

- Resolution

- We know:

- How to represent knowledge through argumentation and formalize these arguments in the language of predicate logic.

- How to transform logical formulas into clauses.

- To demonstrate the validity of an argumentation is to refute (demonstrate the unsatisfactoriness) of a set of formulas:

- premises + denial of conclusion (... maybe we know something about truth tables?)

- We want to know:

- A method of constructing rebuttals using the clausal form.

- A way to translate this method into a specific algorithm. Many more things ..... but, let's start with the unification

# 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  $w$ .

$$w = w(v_1, v_2, v_3, \dots, v_n)$$
$$\sigma = \{v_1 := t_1, v_2 := t_2, \dots, v_n := t_n\}$$

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

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

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

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

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

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

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

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

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

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

# 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:  $w = P(x, f(y), B)$

□  $\sigma_1 = \{x := z, y := w\}$

$$w\sigma_1 = P(z, f(w), B) \quad [\text{Alphabetic variant}]$$

□  $\sigma_2 = \{y := A\}$

$$w\sigma_2 = P(x, f(A), B)$$

□  $\sigma_3 = \{x := g(z), y := A\}$

$$w\sigma_3 = P(g(z), f(A), B)$$

□  $\sigma_4 = \{x := C, y := A\}$

$$w\sigma_4 = P(C, f(A), B) \quad [\text{Ground instance} \rightarrow \text{Instance without variables}]$$

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  $w_1\sigma = w_2\sigma = \dots = w_m\sigma$ .

$\Gamma = \{w_1, w_2, \dots, w_m\}$  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.

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

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

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  $\sigma = \mu \cdot \sigma'$

- The **disagreement set** of a non-empty set of wffs  $\Gamma$  is the first (leftmost) subexpression where the wffs in  $\Gamma$  disagree

$\Gamma = \{w_1, w_2, \dots, w_m\}$  is the first (leftmost)

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

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

# Unification algorithm

□ **Unification algorithm** (Chang & Lee 1973)

**in:**  $\Gamma$  ;    **out:**  $\text{mgu}(\Gamma)$  [most general unifier of  $\Gamma$ ]

1.  $k \leftarrow 0$ ;     $\Gamma_k \leftarrow \Gamma$ ;     $\sigma_k \leftarrow \varepsilon$  (empty substitution) (we put in  $\Gamma$  all the formulas that we want to unify)
2. If  $\Gamma$  is a singleton, **return**  $\sigma_k$  else continue
3.  $D_k \leftarrow$  disagreement set of  $\Gamma_k$  (we put in  $D$  the element  $k$  that we want to unify)
4. If  $D_k$  contains    (i) a term  $t_k$   
                             (ii) a variable symbol  $v_k$  not in  $t_k$   
    then
  - $\sigma_{k+1} \leftarrow \sigma_k \cdot \{v_k := t_k\}$  (we substitute the variable for the term, i.e. we keep the most complex)
  - $\Gamma_{k+1} \leftarrow \Gamma_k \{v_k := t_k\}$  (eliminate repeated wffs) (we update the formulas with the substitutions)
- else **exit with failure** [ $\Gamma$  is not unifiable]
5.  $k \leftarrow k+1$  and go to step 2.

# Unification algorithm

## Example 1.

$$\Gamma = \{P[f(x, g(A, y)), g(A, y)], P[f(x, z), z]\}$$

$$1. D_0 \leftarrow \{z, g(A, y)\};$$

$$\sigma_1 \leftarrow \{z := g(A, y)\}, \quad \Gamma_1 \leftarrow \{P[f(x, g(A, y)), g(A, y)]\};$$

## Example 2.

$$\Gamma = \{A(x, z, g(x, y, f(z))), A(y, f(x), w)\}$$

$$1. D_0 \leftarrow \{x, y\};$$

$$\sigma_1 \leftarrow \{x := y\}, \Gamma_1 \leftarrow \{A(y, z, g(y, y, f(z))), A(y, f(y), w)\}$$

$$2. D_1 \leftarrow \{z, f(y)\};$$

$$\sigma_2 \leftarrow \sigma_1 \cdot \{z := f(y)\};$$

$$\Gamma_2 \leftarrow \{A(y, f(y), g(y, y, f(f(y))))\}, A(y, f(y), w)\};$$

$$3. D_2 \leftarrow \{g(y, y, f(f(y))), w\};$$

$$\sigma_3 \leftarrow \sigma_2 \cdot \{w := g(y, y, f(f(y)))\};$$

$$\Gamma_3 \leftarrow \{A(y, f(y), g(y, y, f(f(y))))\};$$

# Resolution in FOL

□ **A clause** is a disjunction of literals

(it can be represented as a set of literals with implicit disjunctions among the literals in the set).

□ A wff in **CNF** is a conjunction of clauses

(it can be represented as a set of clauses with implicit conjunctions among the clauses in the set).

□ **Binary Resolution between clauses**

Assume that the literals  $\{\phi, \psi\}$  have a most general unifier  $\mu$ .

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

Eg.  $\Gamma = \{P(x) \vee Q(f(x)), R(g(y)) \vee \neg Q(f(A))\}$

$\phi = Q(f(x));$

$\psi = Q(f(A)); \neg\psi = \neg Q(f(A));$

$\mu = \text{mgu}(\{\psi, \phi\}) = \{x := A\}$

infer by resolution  $\{P(A) \vee R(g(y))\}$



# Generalized resolution

❑ Binary resolution is **sound/correct** but **not complete for refutation**

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

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

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  $K_1$  can be unified with the negated literals in clause  $K_2$  by means of a most general unifier  $\mu$ .

Then, from the set of clauses  $\{K_1 \vee \Sigma_1, K_2 \vee \Sigma_2\}$ , it is possible to infer  $W = (\Sigma_1 \vee \Sigma_2) \mu$

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

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

infer by resolution  $\{\square\}$

# Generalized subsumption

□ Clause  $K_1$  **subsumes clause**  $K_2$  if there exists a **substitution**  $\sigma$  such that

$$K_1\sigma \subset K_2$$

□ For the purpose of **inference** the **subsumed clause**,  $K_2$  can be **eliminated** from the knowledge base

□ Example:

$K_1$ : Enjoys(Ringo, z)

$K_2$ :  $\neg$ Pleasant(Reading)  $\vee$  Enjoys(Ringo, Reading)

$\sigma$ : {z := Reading}

[Intuition: Clause  $K_1$  is more “restrictive” than clause  $K_2$ ]

# 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)**

# 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:  $(\forall x) [\neg P(x) \vee \text{Answer}(x)]$

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

# Answer extraction: example

Example:

**KB:**        **All humans are mortal.**  $(\forall x) [ \text{Human}(x) \Rightarrow \text{Mortal}(x) ]$

**Socrates is a human.**     $\text{Human}(\text{Socrates})$

**Is there a mortal man?**  $(\exists x) [ \text{Human}(x) \wedge \text{Mortal}(x) ]$

Add to KB:  $(\forall x) [ \neg \text{Human}(x) \vee \neg \text{Mortal}(x) \vee \text{Ans}(x) ]$

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

# 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 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  $\Delta_{\text{FOL}} \models_{\text{W}_{\text{FOL}}} \text{in FOL by propositionalization:}$

□ Construct  $\alpha_{\text{FOL}} = \Delta_{\text{FOL}} \wedge \neg \text{W}_{\text{FOL}}$

□ 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 if 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 by refutation in FOL

## Generalized resolution by 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  $(\alpha \text{ is UNSAT})$  then  $\Delta \models w$

(ii) If resolution does not produce the empty clause  $(\alpha \text{ is SAT})$  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 when formalizing

Usually,

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

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

E.g. “Everybody at the UAM is smart”

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

“There are people at the UAM who are smart”

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

## COMMON MISTAKES

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

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

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

“Everybody is at the UAM and is smart”

BUT: “The greatest divisor of an integer is the integer itself”

$$(\forall n) \quad [Divisor(n, n) \wedge \neg (\exists m) \quad [Divisor(m, n) \wedge Greater(m, n)]]$$

# Did curiosity kill the cat? (i)

- ☐ 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? (ii)

- ❑ 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

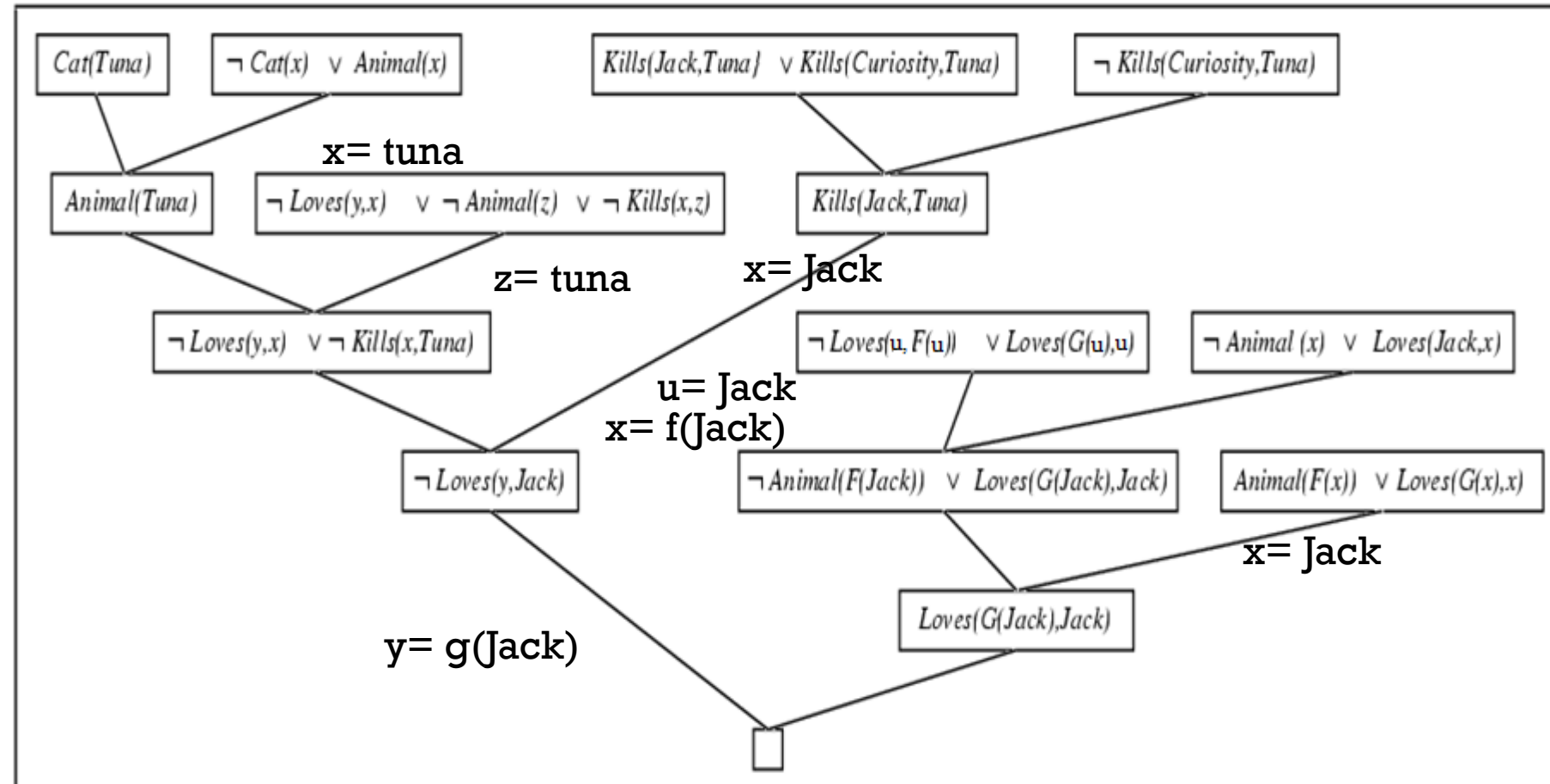
$$(\forall x) \text{Cat}(x) \Rightarrow \text{Animal}(x)$$

- ❑ Did Curiosity kill the cat?

$$\text{Kills}(\text{Curiosity}, \text{Tuna}) \quad ???$$

# Did curiosity kill the cat? (iii)

- 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)

# Example: Ontology for sets

## □ **Ontology for set theory** (vocabulary of object constants, predicates, functions)

### □ Object constants:

□ Empty set:  $\{\}$

□ Set elements:  $A, B, C, \dots$

□ Predicates:  $\text{Set}^1, \text{Member}^2(\in), \text{Subset}^2(\subset), \text{Equal}^2(=)$

□ Functions:  $\text{adjoin}^2(\{\mid\}), \text{union}^2(\cup), \text{intersection}^2(\cap)$

## □ **Axioms of set theory:**

$$1. \forall s [\text{Set}(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s') [\text{Set}(s') \wedge (s = \{x \mid s'\})]]$$

$$2. \neg(\exists x, s) [\{x \mid s\} = \{\}]$$

$$3. (\forall x, s) [x \in s \Leftrightarrow s = \{x \mid s\}]$$

$$4. (\forall x, s) [x \in s \Leftrightarrow (\exists y, s') [s = \{y \mid s'\} \wedge (x = y \vee x \in s')]]$$

$$5. (\forall s_1, s_2) [s_1 \subset s_2 \Leftrightarrow (\forall x) [x \in s_1 \Rightarrow x \in s_2]]$$

$$6. (\forall s_1, s_2) [s_1 = s_2 \Leftrightarrow (s_1 \subset s_2 \wedge s_2 \subset s_1)]$$

$$7. (\forall x, s_1, s_2) [x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)]$$

$$8. (\forall x, s_1, s_2) [x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)]$$

# Example: Lists

- **Ontology for lists**

- **Object constants:** `Null, A, B, C...` (atoms)

- **Predicates:** `Find2, Atom1, Listp1`

- **Functions:** `cons2, append2, first1, rest1`

- **Axioms?** [homework: encode the axioms to characterize the properties of lists ]

# Example: Natural numbers

## □ Ontology for natural numbers

- Object constant: 0

- Predicates

  - Natural number test:  $\text{NatNum}^1$

  - Equality predicate:  $\text{Equal}^2$  (=)

- Functions

  - Successor function:  $\text{succ}^1$

  - Addition function:  $\text{sum}^2$  (+)

## □ Peano axioms

$\text{NatNum}(0)$

$(\forall n) [\text{NatNum}(n) \Rightarrow \text{NatNum}(\text{succ}(n))]$

$(\forall n) [0 \neq \text{succ}(n)]$

$(\forall m, n) [m \neq n \Rightarrow \text{succ}(m) \neq \text{succ}(n)]$

$(\forall n) [\text{NatNum}(n) \Rightarrow \text{sum}(n, 0) = n]$

$(\forall m, n) [\text{NatNum}(m) \wedge \text{NatNum}(n) \Rightarrow \text{sum}(\text{succ}(m), n) = \text{succ}(\text{sum}(m, n))]$

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.

□ `Equal(fatherOf(John), Peter)`  
[also: `fatherOf(John)=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.

$(\forall x, y) [x=y \Rightarrow f_i^{(1)}(x) = f_i^{(1)}(y)] ; i=1, 2, \dots$

$(\forall x, y) [x=y \Rightarrow P_i^{(1)}(x) = P_i^{(1)}(y)] ; i=1, 2, \dots$

$(\forall x, y, z, t) [(x=z) \wedge (y=t) \Rightarrow f_i^{(2)}(x, y) = f_i^{(2)}(z, t)] ; i=1, 2, \dots$

$(\forall x, y, z, t) [(x=z) \wedge (y=t) \Rightarrow P_i^{(2)}(x, y) = P_i^{(2)}(z, t)] ; i=1, 2, \dots$

.....



# 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.