Intelligenza Artificiale

Knowledge representation and reasoning

First-Order Logic

Ivan Heibi
Dipartimento di Filologia Classica e Italianistica (FICLIT)
Ivan.heibi2@unibo.it

Recap, so far

Propositional logic is a simple **language** consisting of **proposition symbols** and **logical connectives**. It can handle propositions that are known to be **true,** known to be **false**, or completely **unknown**

Can we express in Propositional logic the following sentences?

- *Every man is mortal*
- *Socrate is a man*
- *Socrate is mortal*

Recap, so far

Propositional logic is a simple **language** consisting of **proposition symbols** and **logical connectives**. It can handle propositions that are known to be **true,** known to be **false**, or completely **unknown**

Can we express in Propositional logic the following sentences?

- Every man is mortal = ?
- Socrate is a man = M
- Socrate is mortal = P

$$M \Rightarrow P$$

How can I say “every man”?

Formal languages – summary overview

Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0, 1]$
Fuzzy logic	facts with degree of truth $\in [0, 1]$	known interval value

Figure 8.1 Formal languages and their ontological and epistemological commitments.

First-Order Logic

Every man is mortal = $\forall x(man(x) \implies mortal(x))$

Socrate is a man = man(x)

Socrate is mortal = mortal(x)

Components of First-order logic

Differently from propositional logic which assumes world contains sentences, First-Order Logic assumes the world contains:

- **Objects/Individuals:** *people*, *houses*, *numbers*, *theories*, *Barack Obama*, *Mattarella ...*
- **Relations:** these can be unary relations or properties such as *red*, *round.. N-ary relations brother-of, has colour, occurred after, owns, comes between ..*
- **Functions (relations in which there is only one "value" for a given input)**: *father of, best friend, beginning of*

Domains

A **model** in Propositional logic is an assignment of the propositional symbols under consideration.

FOL models are more articulated

FOL models are defined with respect to a **domain (domain of a model)**: the set of objects (also called individuals or domain elements) it contains.

The domain is required to be **nonempty**; every possible world must contain at least one object.

Domains – example

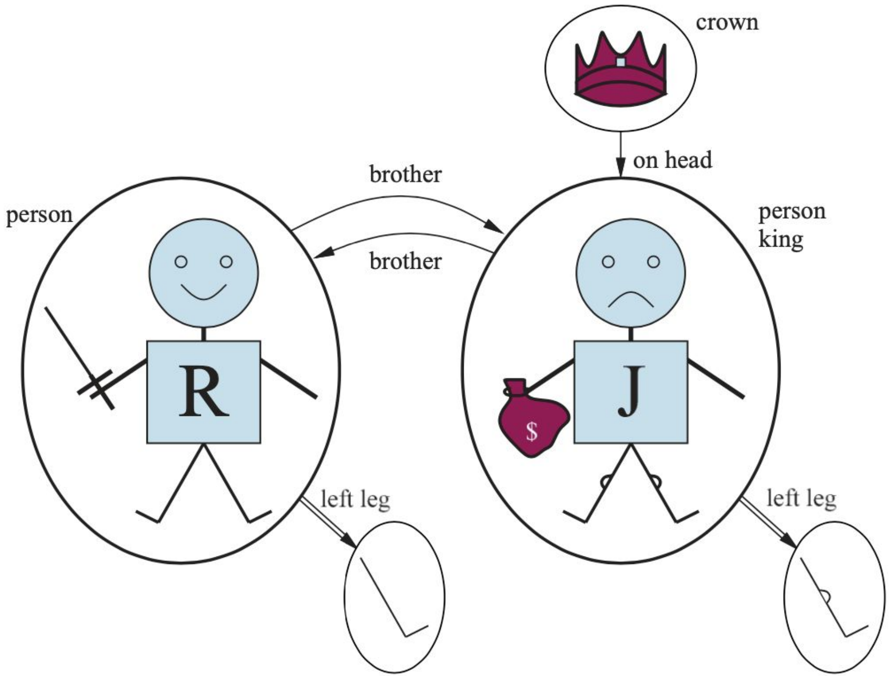
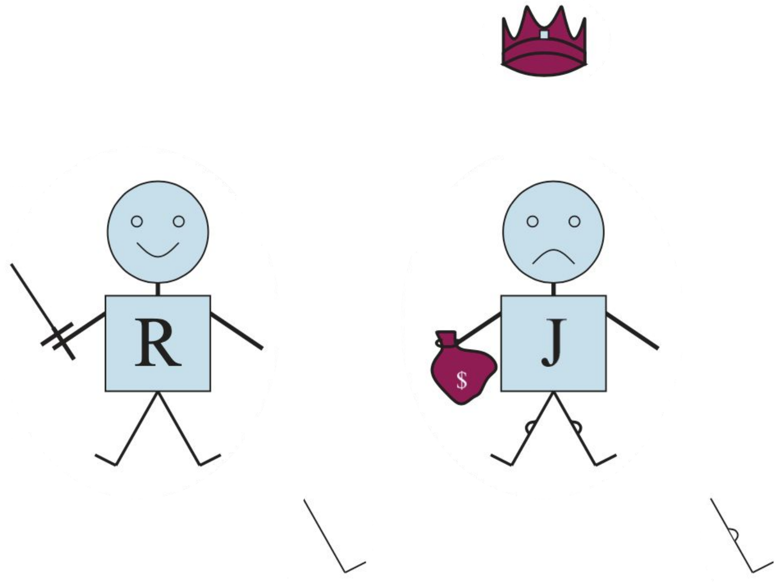


Figure 8.2 A model containing five objects, two binary relations (brother and on-head), three unary relations (person, king, and crown), and one unary function (left-leg).

Objects

The model contains 5 different objects.



Relations – unary

Formally, a **relation is a set of tuples of objects that are related**
– a tuple is a selection of objects arranged in a fixed order

Relation 1: ***person***

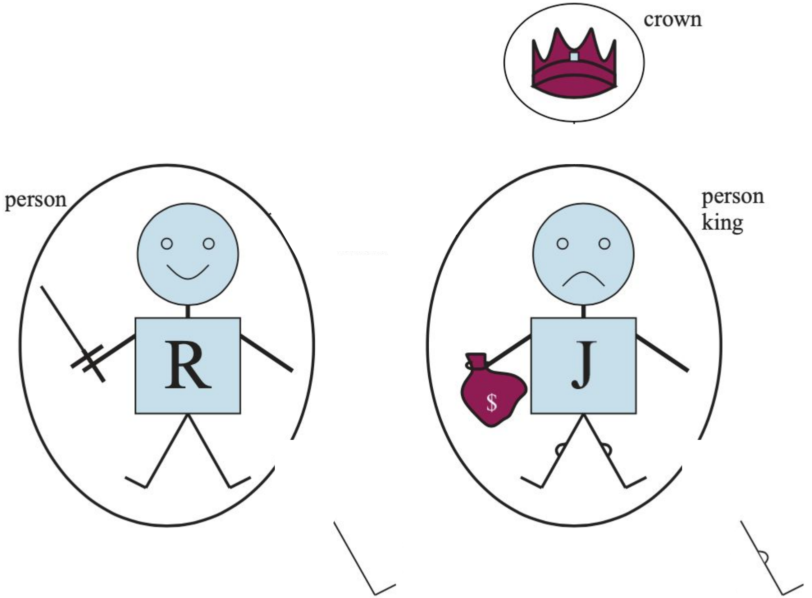
{ < R > , < J > }

Relation 2: ***crown***

{ <  > }

Relation 3: ***king***

{ < J > }



Relations – binary

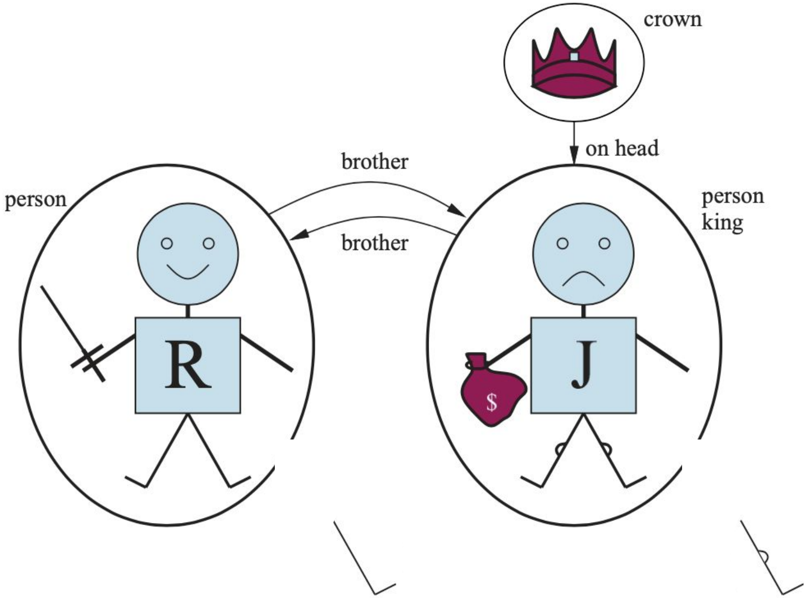
Relation 4: ***on head***

{< , J >}

Relation 5: ***brother***

{< R , J > , < J , R > }

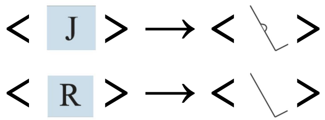
Both are **binary relations** since they relate pairs of objects.

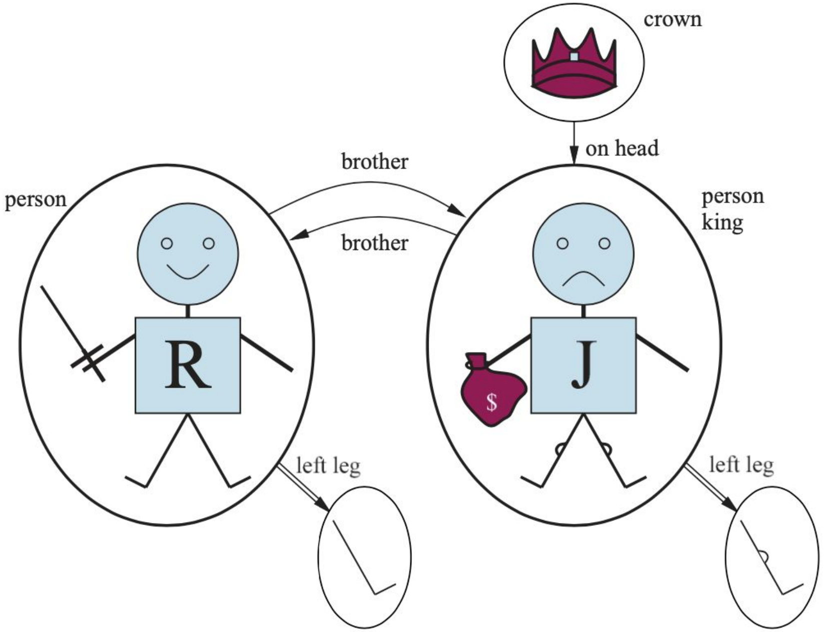


Functions

Some relationships are best considered as <u>functions</u>, in that a given object must be related to **<u>one exact object</u>**

Function 1: ***left leg:***





Symbols

The basic syntactic elements of first-order logic are the **symbols** that stand for **objects, relations and functions**.

The symbols come in three kinds:

- **Constants** symbols, which stand for objects; (e.g., *Richard*, *John*)
- **Predicate** symbols, which stand for relations; (e.g., *Brother*, *OnHead*, *Person*, *King*, *Crown*)
- **Function** symbols, which stand for functions (e.g., *LeftLeg*).

Conventionally symbols will begin with uppercase letters

Interpretations

As in propositional logic, every model must provide the information required to determine if any given sentence is true or false.

In FOL, in addition to objects, relations and functions, <u>each model includes an **interpretation** that specifies exactly which objects, relations and functions are referred to by the constant, predicate and function symbols.</u>

In our example a possible interpretation would be:

- ***Richard*** refers to Richard the Lionheart and ***John*** refers to the evil King John.
- ***LeftLeg*** refers to the left leg function seen before
- ***Brother*** refers to the brotherhood relation seen before
- ***OnHead*** refers to the relation that holds between the crown and King John

There are many other possible interpretations (e.g., *Richard* to the crown)

FOL – so far, summary

FOL consists of a **set of objects** and an **interpretation that maps constant symbols to objects**, **function symbols to functions on those objects**, and **predicate symbols to relations**.

Just as with propositional logic, entailment, validity, and so on are defined in terms of <u>all possible models.</u>

Models vary in how many objects they contain—from one to infinity—and in the way the constant symbols map to objects.
> checking entailment by enumerating all possible models is very difficult!
(e.g., in our previous example the number of possible combinations can be very large)

Domains – interpretation

Objects

R,
J,
crown,
r_left_leg
j_left_leg

Model		FOL		
Unary relations	person	**Predicates**	Person	Person(R) Person(J)
	king		King	King(J)
	crown		Crown	Crown(crown)
Binary relations	brother		Brother	Brother(R,J) Brother(J,R)
	on head		OnHead	OnHead(crown,J)
Functions	left leg	**Functions**	LeftLeg	LeftLeg(R) → r_left_leg LeftLeg(J) → j_left_leg

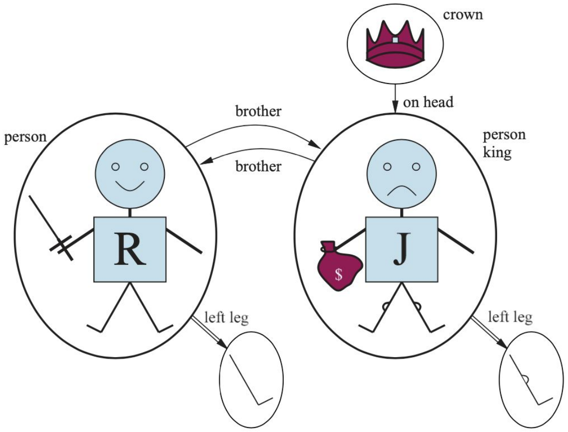


Figure 8.2 A model containing five objects, two binary relations (brother and on-head), three unary relations (person, king, and crown), and one unary function (left-leg).

Terms

A **term** is a logical expression that refers to an **object**.

Terms include: **constants, variables and “function symbols”.**

Is not always convenient to have a distinct symbol to name every object!

> we refer to an object without giving it a name “King’s John left leg” (i.e., *LeftLeg(John)*).

This what function symbols are for: instead of naming explicitly we use a function to refer to the object *LeftLeg(John)*

Function may also have **multiple arguments**, but they always **identify only one element** for each possible combination of the arguments.

Atomic sentences

Atomic sentences state facts by using/combining terms and predicates.

An **atomic sentence** (or atom) is a predicate symbol followed by a parenthesized list of terms
Brother(Richard, John)

Atomic sentences can have complex terms as arguments
Married(Father(Richard) , Mother(John))

An atomic sentence is true in a given model if the relation referred by the predicate symbol holds among the objects referred by the arguments.

Complex sentences

We can use **logical connectives to construct more complex sentences**, with the same syntax and semantics as in propositional calculus.

$\neg$*Brother(LeftLeg(Richard) , John)*
Brother(Richard, John) $\land$ *Brother(John, Richard)*
King(Richard) $\lor$ *King(John)*
$\neg$*King(Richard)* $\Rightarrow$ *King(John)*

Quantifiers

Quantifiers let us express properties regarding the **quantity of objects**, instead of enumerating the objects by name.

FOL contains two standard quantifiers, called **universal** and **existential**.

- **Universal quantification ($\forall$)**
- **Existential quantification ($\exists$)**

Universal quantification – variables

Universal quantifiers make statements about **every object** in the domain

"For all x, if x is a king, then x is a person."

$\forall x\ King(x) \Rightarrow Person(x)$

***x* is called variable** (written in lowercase by convention)

A variable is a term by itself, therefore it could be used in a function as an argument

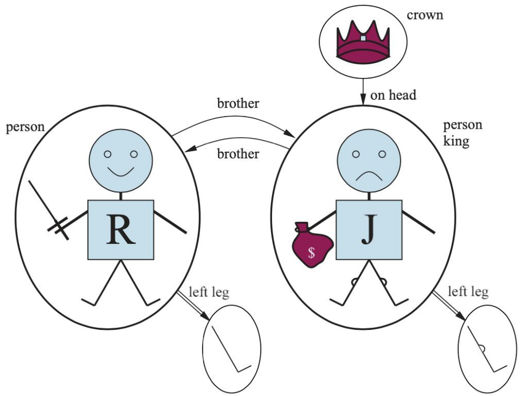
$LeftLeg(x)$

Universal quantification – interpretation

The sentence ∀*x P*, (where P is any logical sentence) is true if *P* is true for every object *x* within the domain.

Consider the Richard & John example:

x ⟶ *Richard*
x ⟶ *John,*
x ⟶ *Richard's left leg,*
x ⟶ *John's left leg,*
x ⟶ *the crown.*



∀x King(x) ⇒ Person(x)

Richard is a king ⇒ Richard is a person. **TRUE**
John is a king ⇒ John is a person. **TRUE**
Richard's left leg is a king ⇒ Richard's left leg is a person. **TRUE**
John's left leg is a king ⇒ John's left leg is a person. **TRUE**
The crown is a king ⇒ the crown is a person. **TRUE**

P	Q	P ⇒ Q
T	T	T
T	F	F
F	T	T
F	F	T

Even if sounds absurd! The truth table for ⇒ says it's true whenever its premise is false—regardless of the truth of the conclusion. (true in the model, but make no claim whatsoever about the personhood qualifications)

Exential quantification

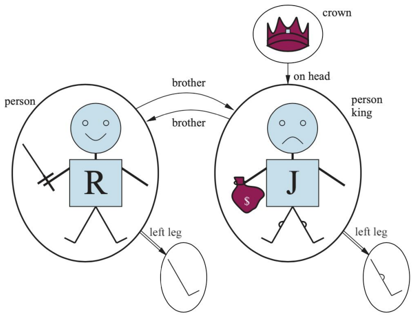
Existential quantification makes statements about **some objects** of the domain

$$\exists x Crown(x) \land OnHead(x, John)$$

Pronounced *"There exists an x such that ..."* or *"For some x..."*

The sentence states that it's true for at least one object x in the domain:

Richard is a crown ∧ Richard is on John's head; **FALSE**
John is a crown ∧ John is on John's head; **FALSE**
Richard's left leg is a crown ∧ Richard's left leg is on John's head; **FALSE**
John's left leg is a crown ∧ John's left leg is on John's head; **FALSE**
The crown is a crown ∧ the crown is on John's head. **TRUE**



Nested quantifiers

We can express more complex sentences using multiple quantifiers.

Quantifiers are of the **same** type:
"Brothers are siblings"

$$\forall x \forall y Brother(x, y) \Rightarrow Sibling(x, y)$$

"Everybody loves somebody"

$$\forall x \exists y Loves(x, y)$$

"There is someone who is loved by everyone":

$$\exists y \forall x Loves(x, y)$$

Connections between quantifiers

The two quantifiers are connected with each other through negation:

Asserting that everyone dislikes spiders is the same as asserting there does not exist someone who likes them:

$\forall x \neg Likes(x, Spiders)$ **is equivalent to** $\neg \exists x Likes(x, Spiders)$

"Everyone likes ice cream" means that there is no one who does not like ice cream:

$\forall x Likes(x, IceCream)$ **is equivalent to** $\neg \exists x \neg Likes(x, IceCream)$

Equality

We can use the equality symbol to signify that two terms refer to the same object

For example,

Father(John) = Henry

Means that the object referred to by *Father(John)* and the object referred to by *Henry* are the same.

syntax

$$
\begin{array}{rcl}
\textit{Sentence} & \rightarrow & \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} & \rightarrow & \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{ComplexSentence} & \rightarrow & (\ \textit{Sentence}\ ) \\
& \mid & \neg\ \textit{Sentence} \\
& \mid & \textit{Sentence} \land \textit{Sentence} \\
& \mid & \textit{Sentence} \lor \textit{Sentence} \\
& \mid & \textit{Sentence} \ \Rightarrow\ \textit{Sentence} \\
& \mid & \textit{Sentence} \ \Leftrightarrow\ \textit{Sentence} \\
& \mid & \textit{Quantifier}\ \textit{Variable}, \ldots\ \textit{Sentence} \\
\\
\textit{Term} & \rightarrow & \textit{Function}(\textit{Term}, \ldots) \\
& \mid & \textit{Constant} \\
& \mid & \textit{Variable} \\
\\
\textit{Quantifier} & \rightarrow & \forall \mid \exists \\
\textit{Constant} & \rightarrow & A \mid X_1 \mid \textit{John} \mid \cdots \\
\textit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\textit{Predicate} & \rightarrow & \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \cdots \\
\textit{Function} & \rightarrow & \textit{Mother} \mid \textit{LeftLeg} \mid \cdots \\
\text{OPERATOR PRECEDENCE} & : & \neg, =, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

Example

FOL – Example

We want to represent information about employees and their relationships with each other using first-order logic.

Mario, Alice, and Sara are all employees;
Alice is the manager of Mario and Sara.
Alice and Mario are friends.

Let's define some predicates …

FOL – Example

We want to represent information about employees and their relationships with each other using first-order logic.

Mario, Alice, and Sara are all employees;
Alice is the manager of Mario and Sara.
Alice and Mario are friends.

Let’s define some predicates:

- Employee(x)
- Manager(x,y)
- Friend(x,y)

FOL – Example

We want to represent information about employees and their relationships with each other using first-order logic.

Mario, Alice, and Sara are all employees;
Alice is the manager of Mario and Sara.
Alice and Mario are friends.

Let's define some predicates:

- Employee(x)
- Manager(x,y)
- Friend(x,y)

Employee(Mario)
Employee(Alice)
Employee(Sara)
Manager(Alice,Mario)
Manager(Alice,Sara)
Friend(Alice,Mario)

FOL – Example

We want to represent information about employees and their relationships with each other using first-order logic.

Mario, Alice, and Sara are all employees;
Alice is the manager of Mario and Sara.
Alice and Mario are friends.

Let's express the fact that there is at least one employee that is managed by Alice and has at least one friend.

- Employee(x)
- Manager(x,y)
- Friend(x,y)

FOL – Example

We want to represent information about employees and their relationships with each other using first-order logic.

Mario, Alice, and Sara are all employees;
Alice is the manager of Mario and Sara.
Only, Alice and Mario are colleagues at same company; Sara works into another company.

Let's express the fact that there is at least one employee that is managed by Alice and has at least one friend.

$$\exists x(Employee(x) \wedge Manager(Alice, x) \wedge \exists y(Friend(y, x)))$$

FOL – Example

We want to represent information about employees and their relationships with each other using first-order logic.

Mario, Alice, and Sara are all employees;
Alice is the manager of Mario and Sara.
Only, Alice and Mario are colleagues at same company; Sara works into another company.

Let's express the fact that not all the employees managed by Alice are also friends with her.

- Employee(x)
- Manager(x,y)
- Friend(x,y)

FOL – Example

We want to represent information about employees and their relationships with each other using first-order logic.

Mario, Alice, and Sara are all employees;
Alice is the manager of Mario and Sara.
Only, Alice and Mario are colleagues at same company; Sara works into another company.

Let's express the fact that not all the employees managed by Alice are also her friends.

$$\neg\forall x(Employee(x) \land Manager(Alice, x) \land Friend(Alice, x))$$