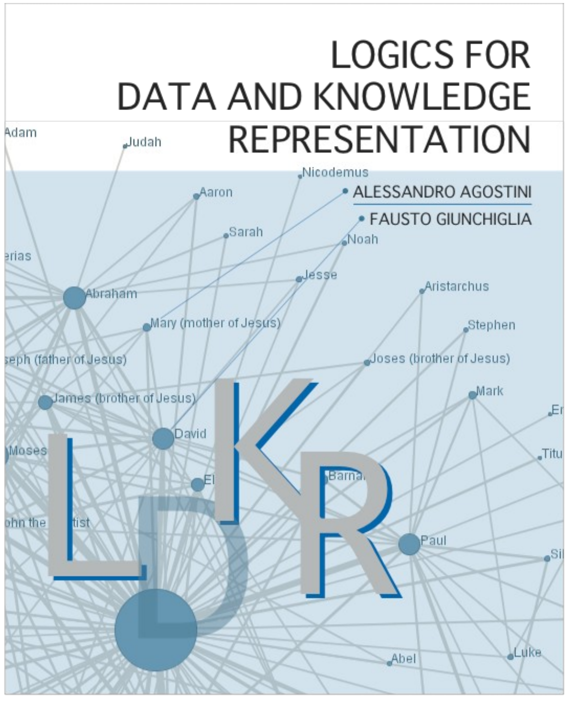


Logics for Data and Knowledge Representation

First Order Logics (FOL)

Originally by Alessandro Agostini and Fausto Giunchiglia
Modified by Fausto Giunchiglia, Rui Zhang and Vincenzo Maltese

Outline

- Introduction
- Syntax
- Semantics
- Reasoning Services

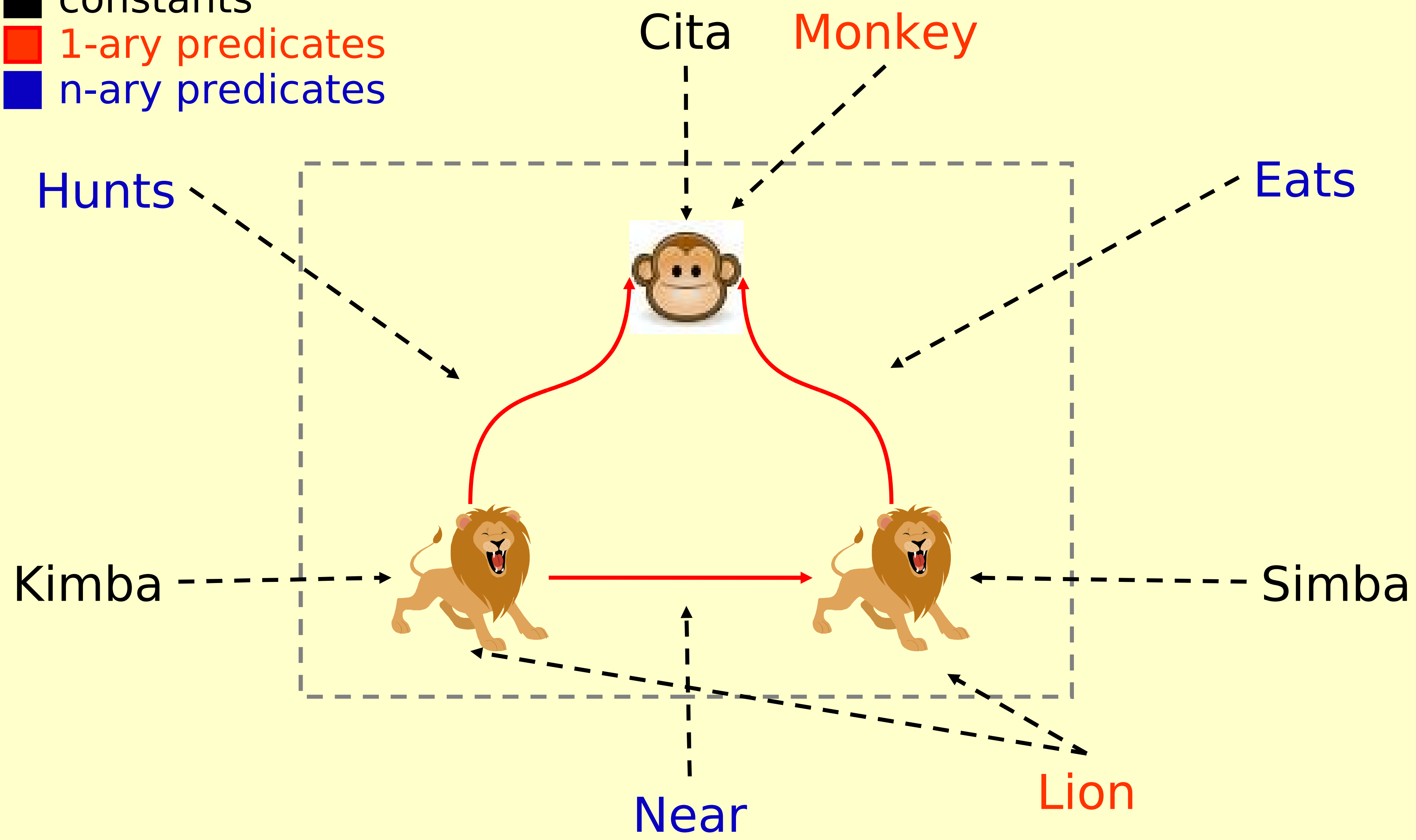
The need for greater expressive power

- We need FOL for a greater expressive power. In FOL we have:
  - constants/individuals (e.g. 2)
  - variables (e.g. x)
  - Unary predicates (e.g. Man)
  - N-ary predicates (eg. Near)
  - functions (e.g. Sum, Exp)
  - quantifiers (∀, ∃)
  - equality symbol = (optional)

- n-ary relations express objects in D^n Near(A,B)
- Functions return a value of the domain, $D^n \rightarrow D$ Multiply(x,y)
- Universal quantification $\forall x\ Man(x) \rightarrow Mortal(x)$
- Existential quantification $\exists x\ (Dog(x) \wedge Black(x))$

Example of what we can express in FOL

- constants
- 1-ary predicates
- n-ary predicates



Alphabet of symbols

- Variables $x_1, x_2, \ldots, y, z$
- Constants $a_1, a_2, \ldots, b, c$
- Predicate symbols $A^1_1, A^1_2, \ldots, A^n_m$
- Function symbols $f^1_1, f^1_2, \ldots, f^n_m$
- Logical symbols $\wedge, \vee, \neg, \rightarrow, \forall, \exists$
- Auxiliary symbols ( )

- Indexes on top are used to denote the number of arguments, called arity, in predicates and functions.
- Indexes on the bottom are used to disambiguate between symbols having the same name.
- Predicates of arity =1 correspond to properties or concepts

Terms and well formed formulas

- Terms can be defined using the following BNF grammar:

  <term> ::= <variable> | <constant> | <function sym> (<term>{,<term>}*)

- A term is a closed term iff it does not contain variables, e.g. Sum(2,3)

- Well formed formulas (wff) can be defined as follows:

  <atomic formula> ::= <predicate sym> (<term>{,<term>}*) |

  <term> = <term>

  <wff> ::= <atomic formula> | ¬<wff> | <wff> ∧ <wff> | <wff> ∨ <wff> |

  <wff> → <wff> | ∀ <variable> <wff> | ∃ <variable> <wff>

NOTE: <term> = <term> is optional. If it is included, we have a FO language with equality.

NOTE: We can also write ∃x.P(x) or ∃x:P(x) as notation (with '.' or ":")

Scope and index of logical operators

Given two wff α and β

- Unary operators

  In $\neg\alpha$, $\forall x\alpha$ and $\exists x\alpha$,

  α is the scope and x is the index of the operator

- Binary operators

  In $\alpha \wedge \beta$, $\alpha \vee \beta$ and $\alpha \rightarrow \beta$,

  α and β are the scope of the operator

NOTE: in the formula $\forall x_1\ A(x_2)$, x_1 is the index but x_1 is not in the scope, therefore the formula can be simplified to $A(x_2)$.

Free and bound variables

- A variable x is bound in a formula γ if it is γ = ∀x α(x) or ∃x α(x) that is x is both in the index and in the scope of the operator.
- A variable is free otherwise.
- A formula with no free variables is said to be a sentence or closed formula.
- A FO theory is any set of FO-sentences.

NOTE: we can substitute the bound variables without changing the meaning of the formula, while it is in general not true for free variables.

Interpretation function

- An interpretation I for a FO language L over a domain D is a function such that:
  - $I(a_i) = a_i$ for each constant a_i
  - $I(A^n) \subseteq D^n$ for each predicate A of arity n
  - $I(f^n)$ is a function f: $D^n \rightarrow D \subseteq D^{n+1}$ for each function f of arity n

Assignment

- An assignment for the variables $\{x_1, \ldots, x_n\}$ of a FO language L over a domain D is a mapping function *a*: $\{x_1, \ldots, x_n\} \rightarrow D$

  $a(x_i) = d_i \in D$

  NOTE: In countable domains (finite and enumerable) the elements of the domain D are given in an ordered sequence $<d_1,\ldots,d_n>$ such that the assignment of the variables x_i follows the sequence.

  NOTE: the assignment *a* can be defined on free variables only.

Interpretation over an assignment a

- An interpretation I_a for a FO language L over an assignment *a* and a domain D is an extended interpretation where:
  - $I_a(x) = a(x)$ for each variable x
  - $I_a(c) = I(c)$ for each constant c
  - $I_a(f^n(t_1,\ldots, t_n)) = I(f^n)(I_a(t_1),\ldots, I_a(t_n))$ for each function f of arity n

NOTE: I_a is defined on terms only

Satisfaction relation

- We are now ready to provide the notion of satisfaction relation:

$$M \models \gamma\ [a]$$

(to be read: M satisfies γ under *a* or γ is true in M under *a*)

where:

- M is an interpretation function I over D
  M is a mathematical structure <D, I>
- *a* is an assignment $\{x_1, \ldots, x_n\} \rightarrow D$
- γ is a FO-formula

NOTE: if γ is a sentence with no free variables, we can simply write: $M \models \gamma$ (without the assignment *a*)

Satisfaction relation for well formed formulas

- γ atomic formula:
  - γ: $t_1 = t_2$ $\quad M \models (t_1 = t_2)[a]$ iff $I_a(t_1) = I_a(t_2)$
  - γ: $A^n(t_1,\ldots, t_n)$ $\quad M \models A^n(t_1,\ldots, t_n)[a]$ iff $(I_a(t_1), \ldots, I_a(t_n)) \in I(A^n)$

Satisfaction relation for well formed formulas

- γ well formed formula:
  - γ: $\neg \alpha$ — $M \models \neg \alpha [a]$ iff $M \not\models \alpha [a]$
  - γ: $\alpha \wedge \beta$ — $M \models \alpha \wedge \beta [a]$ iff $M \models \alpha [a]$ and $M \models \beta [a]$
  - γ: $\alpha \vee \beta$ — $M \models \alpha \vee \beta [a]$ iff $M \models \alpha [a]$ or $M \models \beta [a]$
  - γ: $\alpha \rightarrow \beta$ — $M \models \alpha \rightarrow \beta [a]$ iff $M \not\models \alpha [a]$ or $M \models \beta [a]$

  - γ: $\forall x_i \alpha$ — $M \models \forall x_i \alpha [a]$ iff $M \models \alpha [s]$ for all assignments
    $s = <d_1,\ldots, d'_i,\ldots, d_n>$ where s varies from *a* only
    for the i-th element (s is called an i-th variant of *a*)

  - γ: $\exists x_i \alpha$ — $M \models \exists x_i \alpha [a]$ iff $M \models \alpha [s]$ for some assignment
    $s = <d_1,\ldots, d'_i,\ldots, d_n>$ i-th variant of *a*

Satisfaction relation for a set of formulas

- We say that a formula γ is true (w.r.t. an interpretation I) iff every assignment

  $s = <d_1,\ldots, d_n>$ satisfies γ, i.e. M ⊨ γ [s] for all s.

  NOTE: under this definition, a formula γ might be neither true nor false w.r.t. an interpretation I (it depends on the assignment)

- If γ is true under I we say that I is a model for γ.

- Given a set of formulas Γ, M satisfies Γ iff M ⊨ γ for all γ in Γ

Satisfiability and Validity

- We say that a formula γ is satisfiable iff there is a structure

  M = <D, I> and an assignment *a* such that M ⊨ γ [*a*]

- We say that a set of formulas Γ is satisfiable iff there is a structure M = <D, I> and an assignment *a* such that

  M ⊨ γ [*a*] for all γ in Γ

- We say that a formula γ is valid iff it is true for any structure and assignment, in symbols ⊨ γ

- A set of formulas Γ is valid iff all formulas in Γ are valid.

Entailment

- Let be Γ a set of FO- formulas, γ a FO- formula, we say that

$$\Gamma \models \gamma$$

(to be read Γ entails γ)

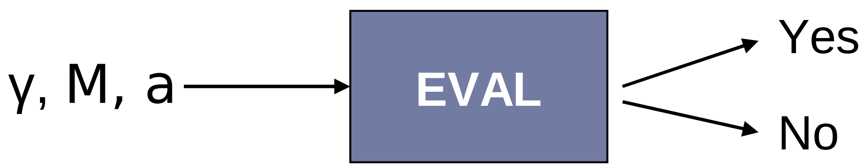
iff for all the interpretations M and assignments a,

if $M \models \Gamma\,[a]$ then $M \models \gamma\,[a]$.

Reasoning Services: EVAL

Model Checking (EVAL)

Is a FO-formula γ true under a structure M = <D, I> and an assignment a? Check M ⊨ γ [a]



Satisfiability (SAT)

Given a FO-formula γ, is there any structure M = <D, I> and an assignment *a* such that M ⊨ γ [a]?

Validity (VAL)

Given a FO-formula γ, is γ true for all the interpretations M and assignments *a*, i.e. ⊨ γ?

NOTE: they are decidable in finite domains

How to reason on finite domains

- $\models \forall x\, P(x)\, [a]$ D = {a, b, c}

  we have only 3 possible assignments *a*(x) = a, *a*(x) = b, *a*(x) = c

  we translate in $\models P(a) \wedge P(b) \wedge P(c)$

- $\models \exists x\, P(x)\, [a]$ D = {a, b, c}

  we have only 3 possible assignments *a*(x) = a, *a*(x) = b, *a*(x) = c

  we translate in $\models P(a) \vee P(b) \vee P(c)$

- $\models \forall x\, \exists y\, R(x,y)\, [a]$ D = {a, b, c}

  we have 9 possible assignments, e.g. *a*(x) = a, *a*(y) = b

  we translate in $\models \exists y\, R(a,y) \wedge \exists y\, R(b,y) \wedge \exists y\, R(c,y)$

  and then in $\models (R(a,a) \vee R(a,b) \vee R(a,c)) \wedge$

  $(R(b,a) \vee R(b,b) \vee R(b,c)) \wedge$

  $(R(c,a) \vee R(c,b) \vee R(c,c))$