| **Propositional Logic** | **Predicate logic** | **EL** | **EL⊥** | **ALC** |
| --- | --- | --- | --- | --- |
|  |  | C | T | C ∏ D | ∃ r.C | C | T | ⊥ | C ∏ D | ∃ r.C | C | T | ⊥ | C ∏ D | ¬C | ∃ r.C ( C ⊔ D | ∀r.C ) |
| Deal with one object | Can relate objects | There is no way to express negations | Limit the expressivity of predicate rules. Allows for anonymous objects. Belongs to the family of DLs | Can represent complex classes |
| *Ana is a professor*  we are saying something about a specific person, this knowledge doesn't say anything about bob | *Ana supervises Bob*  Two individuals in a relationship, something that we can’t do in propositional logic. | Every EL concept can be represented by a labelled tree | We can “kinda” express negation. In the form A ∏ B ⊑ ⊥  But there can still be something that is not A and not B at the same time | Is EL extended with the negation |
| From proposition  *Mamma* | To predicates  *Mammal(dumbo[[1]](#footnote-0))* can be true or false |  |  |  |
|  | Have constants, unary and binary predicates and variables |  |  |  |
|  | Predicates represent property of objects (unary) or relationships between objects (binary) |  |  |  |

# 

# 00 - Opening

Logic: language without the nuances (sfumature) used to express and manipulate knowledge

# 01 - What is Knowledge Representation?

Sound: true premises derive true conclusions

Complete: whatever follow can be derived (partially true)

**Propositional logic:** simple logic, speaks about prepositions (properties) it is too inexpressive

**Predicate logic:** is too complex, no sound and complete derivation system

DLs: Description Logic is a family of KR languages

# 02 - Boolean Algebra and Propositional Logic

**Propositional logic:** formalism for combining propositions (sentences, statements) to make conclusion about their truth value

Atomic propositions: state one specific fact or property

* *mammas are vertebrate*
* *Vertebrates are vegetarians*
* *Cars have umbrellas*
* *X* (there is no implicit meaning)

Syntax: tell which expressions are allowed

Semantics: specify what do the expression mean

# 03 - Propositional Rules

Clause: disjunction of literals (variables or negated variables)

Horn Clause: clause with exactly one positive literal

Knowledge base: finite set of rules

K ⊨ ϕ K entails ϕ, ϕ is a consequence of K K → ϕ is a tautology

K is a knowledge base

φ is a consequence that I am interested in

Redux: knowledge base without facts in the body

This language is very inexpressive, need to use predicate logic

# 04 - Predicate Logic and Rules

Domain: all object that we are interested about

Interpretation functions: tells how to read symbols

Predicate rule: predica clause P(t) ← Q(t1), … , Q(tn) such that all variables in P(t) appear in the body

Predicate knowledge base: finite set of predicate rules

Canonical model: minimal (in terms of constraints, biggest possible) of all possible models. If we conclude something is true for this model it is true for all possible models

# 05 - Extensions of Rules and Introduction to DLs

EL⊥ is part of DLs

DLs are characterised by

* clear syntax
* unambiguous semantics

ALC

ALC can represent complex classes

* ¬ mortal class of immortal (**non** mortal) objects
* Green ∏ fragile object that are green **and** fragile (conjunction)
* ∃hasChild.Female people with female children. It is a relationship between classes (maybe one female and one boy)
* ∀ hasChild.Female people with **only** female children

# 06 - A Light-weight Description Logic (EL)

EL⊥

* **concept** (names) are unary predicates sets. They are interpreted as sets
* **Role** (names) are binary predicates pairs. Binary relationship
* T is a tautology always true. We will interpret it as a problems that holds for all the individuals
* ⊥is a contradiction always false. Is the empty set, no object can satisfy this property
* ∏ is the conjunction In logical terms it is ^. The conjunction of two property is satisfied iff both properties are satisfied
* ∃ is about role successor. Is a binary predicate, property about the successor of a role. It is a sequence of arrows

Any EL⊥ with ⊥ as a subconcept is equivalent to ⊥

TBox: terminological box, imposed restrictions on the potential interpretations of concepts

GCI in normal form if in has one of these shapes

* A ⊑ B
* A1 ∏ A2 ⊑ B
* A ⊑ ∃r.B
* ∃r.A ⊑ B

Every TBox can be transformed in normal form by applying the normalisation rules

# 07 - ALC

In EL there is no way to express negations, but in EL⊥ we can partially achieve it

ALC is EL extended with the negation operator

A ALC concept is in negation normal form (NNF) iff negations apply to concept names only.

Saturated: no rule is applicable anymore

Open: without contradiction

# 08 - Extension of ALC

Nominal O: special concept that have one individual

Cardinality Constraints Q, N: (N unquantified number restriction) (Q qualified number restrictions). It limits the number of successors of an object. Limit how many successor an object can have

Inverse Role I:

1. dumbo is an object, by itself it has no value [↑](#footnote-ref-0)