# Ontology Languages
## Description Logics

ANNA QUERALT, OSCAR ROMERO

(FACULTAT D’INFORMÀTICA DE BARCELONA)

Examples in this section are based on:

- D. Calvanese and D. Lembo (tutorial on DL @ISCW’07)
- F. Baader et al. The Description Logic Handbook

# DESCRIPTION LOGICS

## HOW TO MODEL KNOWLEDGE AND ASSERT INSTANCES

# Logics-Based Knowledge Representation

First-Order Logic (FOL)

- Suitable for knowledge representation
  - Classes as unary predicates
  - Properties / relationships as binary predicates
  - Constraints as logical formulas using those predicates
- Undecidability
  - In the general case, there is no algorithm that determines if a FOL formula implies another

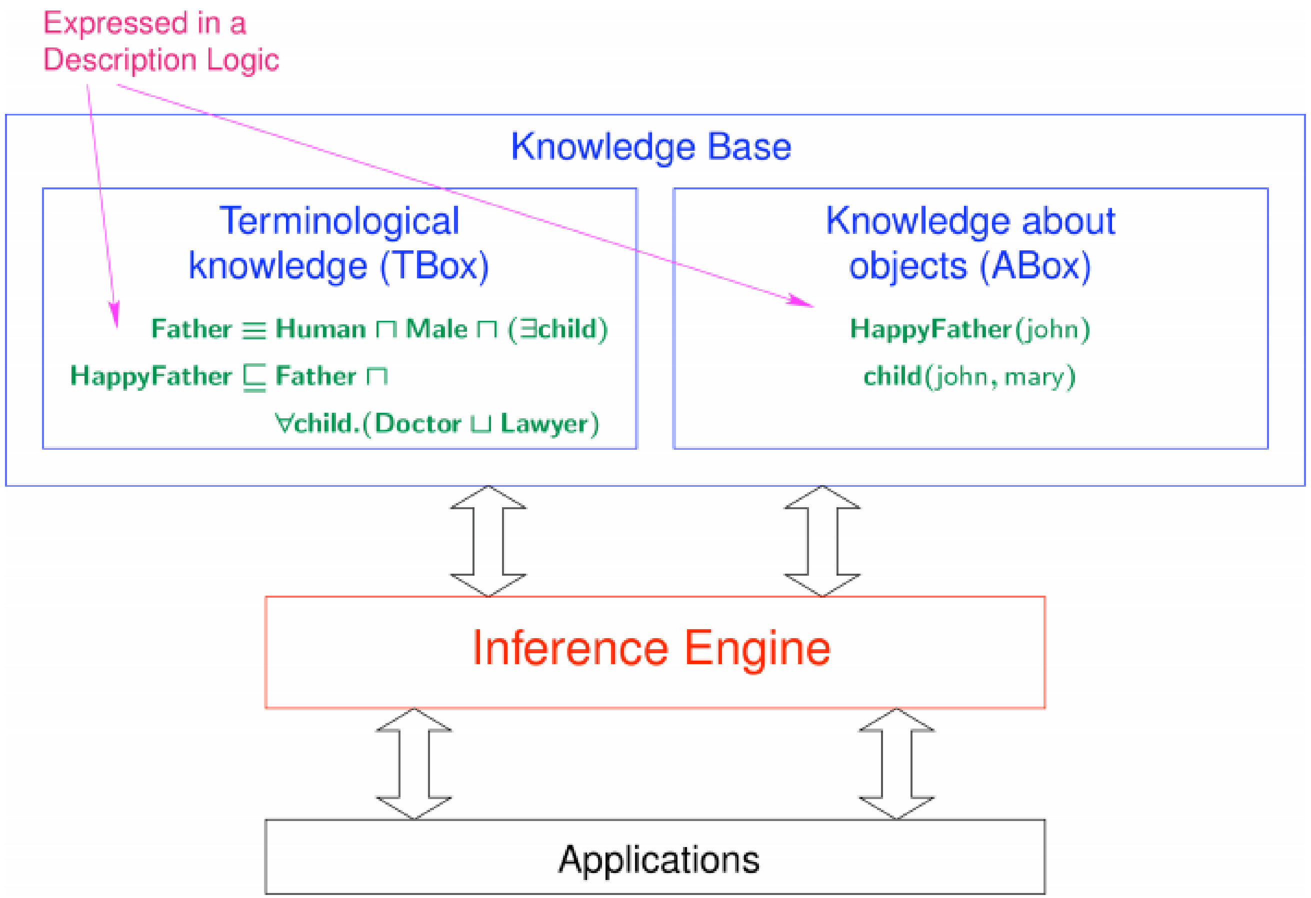
Decidable Fragments of FOL

- Description Logics (binary predicates bounded number of variables)
- Datalog (Horn-clauses)

# Decidable Subsets of FOL

| | Datalog | Description Logics |
|---|---|---|
| Focus | Instances | Knowledge |
| Approach | Centralized | Decentralized |
| Reasoning | Closed-world assumption | Open-world assumption |
| Unique name | Unique name assumption | Non-unique name assumption |

# Description Logic (DL) Knowledge Base



# Description Logics and Ontologies

Description Logics are used to assert **knowledge** and **instances**

- The knowledge is asserted in the TBOX (DL terminology)
- The instances are asserted in the ABOX (DL assertions)

A DL TBOX and ABOX is a decidable subset of FOL. DL defines accordingly reasoning services for DL KBs

We say a *knowledge base* is an ontology if:

- It defines the ontology terminology (TBOX)
- The asserted instances (ABOX) are complaint with the terminology (i.e., TBOX)
- It provides **sound** reasoning services

Thus:

- Any Description Logic KB is always an ontology
- A RDFS KB is an ontology if:
    - You define a TBOX
    - The RDFS ABOX is compliant with the TBOX
    - You use sound inference rules (i.e., those defined by the SPARQL community)
- Strictly speaking, although many people say the opposite, a RDF knowledge base is not an ontology if we follow the definition above

# Description Logic: TBOX

A DL TBOX is characterized by a set of constructs for building **complex concepts and roles** from **atomic concepts and roles**:

- Concepts correspond to classes
- Roles correspond to relationships

Atomic concepts / roles:

- Must be explicitly defined by the user (e.g., the person concept or the lectures role)

Complex concepts / roles:

- They are derived from atomic concepts or roles (e.g., a lecturer is a person who lectures)
- They must be derived using the pre-defined **concept and role constructs** provided by the description logic

It is called TBOX because it defines the **terminology** (of the domain)

- It is equivalent to the metadata / schema layer we have used for RDFS

# Description Logic: TBOX

A DL TBOX is characterized by a set of constructs for building **complex concepts and roles** from **atomic concepts and roles**:

- Concepts correspond to classes
- Roles correspond to relationships

**A DL TBOX formal semantics** are given in terms of interpretations:

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of:

- a nonempty set $\Delta^{\mathcal{I}}$, the domain of $\mathcal{I}$
- an interpretation function $\cdot^{\mathcal{I}}$, which maps
  - each individual $a$ to an element $a^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$
  - each atomic concept $A$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$
  - each atomic role $P$ to a subset $P^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

# Concept Constructs

Atomic concepts and roles are defined explicitly by the user!

| Construct | Syntax | Example | Semantics |
|---|---|---|---|
| atomic concept | $A$ | Doctor | $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ |
| atomic role | $P$ | hasChild | $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ |
| atomic negation | $\neg A$ | ¬Doctor | $\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$ |
| conjunction | $C \sqcap D$ | Hum ⊓ Male | $C^{\mathcal{I}} \cap D^{\mathcal{I}}$ |
| (unqual.) exist. res. | $\exists R$ | ∃hasChild | $\{ a \mid \exists b.\, (a,b) \in R^{\mathcal{I}} \}$ |
| value restriction | $\forall R.C$ | ∀hasChild.Male | $\{a \mid \forall b.\, (a,b) \in R^{\mathcal{I}} \to b \in C^{\mathcal{I}}\}$ |
| bottom | $\bot$ | | $\emptyset$ |

($C$, $D$ denote arbitrary concepts and $R$ an arbitrary role)

The above constructs form the basic language $\mathcal{AL}$ of the family of $\mathcal{AL}$ languages.

# Additional Concept and Role Constructs

| Construct | $\mathcal{AL}\cdot$ | Syntax | Semantics |
|---|---|---|---|
| disjunction | $\mathcal{U}$ | $C \sqcup D$ | $C^{\mathcal{I}} \cup D^{\mathcal{I}}$ |
| top | | $\top$ | $\Delta^{\mathcal{I}}$ |
| qual. exist. res. | $\mathcal{E}$ | $\exists R.C$ | $\{ a \mid \exists b.\, (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}} \}$ |
| (full) negation | $\mathcal{C}$ | $\neg C$ | $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$ |
| number restrictions | $\mathcal{N}$ | $(\geq k\, R)$ | $\{ a \mid \#\{b \mid (a,b) \in R^{\mathcal{I}}\} \geq k \}$ |
| | | $(\leq k\, R)$ | $\{ a \mid \#\{b \mid (a,b) \in R^{\mathcal{I}}\} \leq k \}$ |
| qual. number restrictions | $\mathcal{Q}$ | $(\geq k\, R.C)$ | $\{ a \mid \#\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \geq k \}$ |
| | | $(\leq k\, R.C)$ | $\{ a \mid \#\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \leq k \}$ |
| inverse role | $\mathcal{I}$ | $R^{-}$ | $\{ (a,b) \mid (b,a) \in R^{\mathcal{I}} \}$ |
| role closure | $_{reg}$ | $\mathcal{R}^{*}$ | $(R^{\mathcal{I}})^{*}$ |

# Understanding DL Axioms

What is the meaning of these axioms? Write the **interpretation** corresponding to each axiom

$\forall \mathsf{hasChild}.(\mathsf{Doctor} \sqcup \mathsf{Lawyer})$

$\exists \mathsf{hasChild}.\mathsf{Doctor}$

$\neg(\mathsf{Doctor} \sqcup \mathsf{Lawyer})$

$(\geq 2\, \mathsf{hasChild}) \sqcap (\leq 1\, \mathsf{sibling})$

$(\geq 2\, \mathsf{hasChild}.\, \mathsf{Doctor})$

$\forall \mathsf{hasChild}^{-}.\mathsf{Doctor}$

$\exists \mathsf{hasChild}^{*}.\mathsf{Doctor}$

# TBOX Definition

A DL TBOX only includes terminological axioms of the following form

- Inclusion (*subsumption*) $C_1 \sqsubseteq C_2$ is satisfied by $\mathcal{I}$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
  $R_1 \sqsubseteq R_2$ is satisfied by $\mathcal{I}$ if $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$

  Example: $\mathrm{PhDStudent} \sqsubseteq \mathrm{Student} \sqcap \mathrm{Researcher}$

- Equivalence $C_1 \sqsubseteq C_2,\ C_2 \sqsubseteq C_1$

  Example: $\mathrm{PhDStudent} \equiv \mathrm{Student} \sqcap \mathrm{Researcher}$

# Description Logics: ABOX

Defines instances in terms of the terminological axioms defined in the TBOX

- Concept assertions
  - Student(Pere)
- Role assertions
  - Teaches(Oscar, Pere)

We **cannot** assert instances for a concept not defined previously in the TBOX

We can assert instances of both atomic and complex concepts / roles

It is called ABOX because it defines **assertions** on the TBOX concepts and roles

- It is equivalent to the instance layer we have used for RDFS

# Example of DL Knowledge Base

TBox assertions:

- Inclusion assertions on concepts:

$$
\begin{aligned}
\mathsf{Father} &\equiv \mathsf{Human} \sqcap \mathsf{Male} \sqcap \exists \mathsf{hasChild} \\
\mathsf{HappyFather} &\sqsubseteq \mathsf{Father} \sqcap \forall \mathsf{hasChild}.(\mathsf{Doctor} \sqcup \mathsf{Lawyer} \sqcup \mathsf{HappyPerson}) \\
\mathsf{HappyAnc} &\sqsubseteq \forall \mathsf{descendant}.\mathsf{HappyFather} \\
\mathsf{Teacher} &\sqsubseteq \neg \mathsf{Doctor} \sqcap \neg \mathsf{Lawyer}
\end{aligned}
$$

- Inclusion assertions on roles:

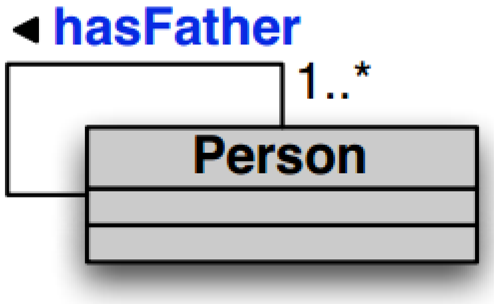
$$\mathsf{hasChild} \sqsubseteq \mathsf{descendant} \qquad\qquad \mathsf{hasFather} \sqsubseteq \mathsf{hasChild}^{-}$$

ABox membership assertions:

- $\mathsf{Teacher}(\mathtt{mary}),\quad \mathsf{hasFather}(\mathtt{mary}, \mathtt{john}),\quad \mathsf{HappyAnc}(\mathtt{john})$

# Exercise

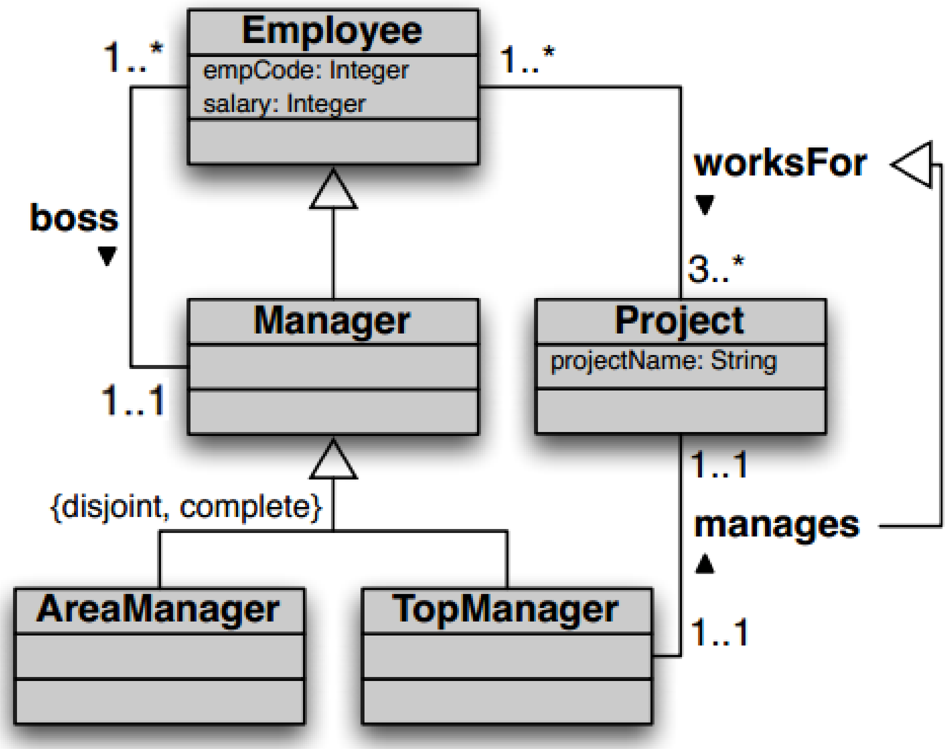
Represent as concept expressions the following UML diagram




$$
\begin{array}{lrcl}
\text{TBox } \mathcal{T}\text{:} & \exists \mathsf{hasFather} & \sqsubseteq & \mathsf{Person} \\
& \exists \mathsf{hasFather}^{-} & \sqsubseteq & \mathsf{Person} \\
& \mathsf{Person} & \sqsubseteq & \exists \mathsf{hasFather}
\end{array}
$$

# Exercise II

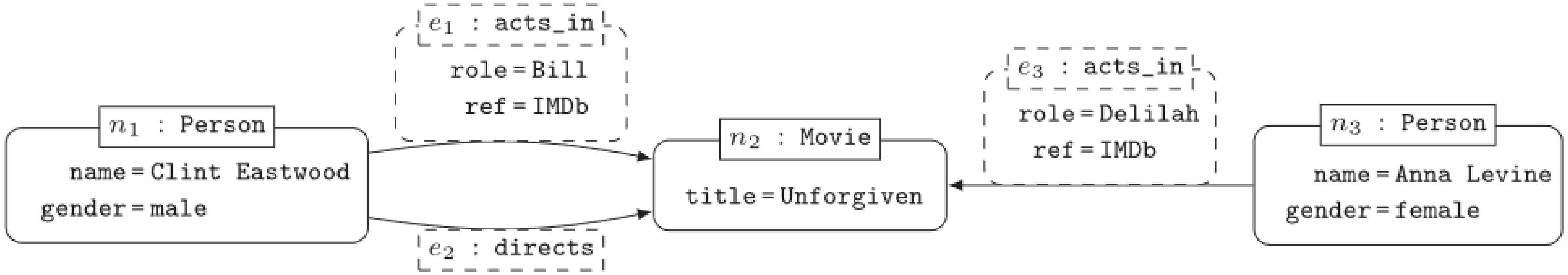
Represent as concept expressions the following UML diagram




$$
\begin{array}{rcl}
\mathsf{Manager} & \sqsubseteq & \mathsf{Employee} \\
\mathsf{AreaManager} & \sqsubseteq & \mathsf{Manager} \\
\mathsf{TopManager} & \sqsubseteq & \mathsf{Manager} \\
\mathsf{Manager} & \sqsubseteq & \mathsf{AreaManager} \sqcup \\
 & & \mathsf{TopManager} \\
\mathsf{AreaManager} & \sqsubseteq & \neg\mathsf{TopManager} \\
\mathsf{Employee} & \sqsubseteq & \exists \mathsf{salary} \\
\exists \mathsf{salary}^- & \sqsubseteq & \mathsf{Integer} \\
\exists \mathsf{worksFor} & \sqsubseteq & \mathsf{Employee} \\
\exists \mathsf{worksFor}^- & \sqsubseteq & \mathsf{Project} \\
\mathsf{Project} & \sqsubseteq & \exists \mathsf{worksFor}^- \\
\mathsf{Employee} & \sqsubseteq & \geq 3\ \mathsf{worksFor} \\
 & \ldots &
\end{array}
$$

# Exercise III

Create a DL KB capturing as much constraints as possible from the following graph:



Examples in this section are based on:

- D. Calvanese and D. Lembo (tutorial on DL @ISCW’07)
- F. Baader et al. The Description Logic Handbook

# DESCRIPTION LOGICS

## REASONING

# Model of a DL Ontology

**Model of a DL knowledge base**

An interpretation $\mathcal{I}$ is a model of $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ if it satisfies all assertions in $\mathcal{T}$ and all assertions in $\mathcal{A}$.

$\mathcal{O}$ is said to be satisfiable if it admits a model.

The fundamental reasoning service from which all other ones can be easily derived is . . .

**Logical implication**

$\mathcal{O}$ logically implies and assertion $\alpha$, written $\mathcal{O} \models \alpha$, if $\alpha$ is satisfied by all models of $\mathcal{O}$.

# TBOX Reasoning

- Concept Satisfiability: $C$ is satisfiable wrt $\mathcal{T}$, if there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is not empty, i.e., $\mathcal{T} \not\models C \equiv \bot$.
- Subsumption: $C_1$ is subsumed by $C_2$ wrt $\mathcal{T}$, if for every model $\mathcal{I}$ of $\mathcal{T}$ we have $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, i.e., $\mathcal{T} \models C_1 \sqsubseteq C_2$.
- Equivalence: $C_1$ and $C_2$ are equivalent wrt $\mathcal{T}$ if for every model $\mathcal{I}$ of $\mathcal{T}$ we have $C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$, i.e., $\mathcal{T} \models C_1 \equiv C_2$.
- Disjointness: $C_1$ and $C_2$ are disjoint wrt $\mathcal{T}$ if for every model $\mathcal{I}$ of $\mathcal{T}$ we have $C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} = \emptyset$, i.e., $\mathcal{T} \models C_1 \sqcap C_2 \equiv \bot$.
- Functionality implication: A functionality assertion $(\textbf{funct}\ R)$ is logically implied by $\mathcal{T}$ if for every model $\mathcal{I}$ of $\mathcal{T}$, we have that $(o, o_1) \in R^{\mathcal{I}}$ and $(o, o_2) \in R^{\mathcal{I}}$ implies $o_1 = o_2$, i.e., $\mathcal{T} \models (\textbf{funct}\ R)$

# Reasoning Complexity

**Complexity of concept satisfiability: [DLNN97]**

| | |
|---|---|
| $\mathcal{AL}$, $\mathcal{ALN}$ | PTIME |
| $\mathcal{ALU}$, $\mathcal{ALUN}$ | NP-complete |
| $\mathcal{ALE}$ | coNP-complete |
| $\mathcal{ALC}$, $\mathcal{ALCN}$, $\mathcal{ALCI}$, $\mathcal{ALCQI}$ | PSPACE-complete |

*Observations:*

- Two sources of complexity:
  - union ($\mathcal{U}$) of type NP,
  - existential quantification ($\mathcal{E}$) of type coNP.

  When they are combined, the complexity jumps to PSPACE.
- Number restrictions ($\mathcal{N}$) do not add to the complexity.

# Ontology Reasoning

- Ontology Satisfiability: Verify whether an ontology $\mathcal{O}$ is satisfiable, i.e., whether $\mathcal{O}$ admits at least one model.
- Concept Instance Checking: Verify whether an individual $c$ is an instance of a concept $C$ in $\mathcal{O}$, i.e., whether $\mathcal{O} \models C(c)$.
- Role Instance Checking: Verify whether a pair $(c_1, c_2)$ of individuals is an instance of a role $R$ in $\mathcal{O}$, i.e., whether $\mathcal{O} \models R(c_1, c_2)$.
- Query Answering:

The certain answers to $q(\vec{x})$ over $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, denoted $cert(q, \mathcal{O})$, ...

are the tuples $\vec{c}$ of constants of $\mathcal{A}$ such that $\vec{c} \in q^{\mathcal{I}}$, for every model $\mathcal{I}$ of $\mathcal{O}$.

# Example

TBOX:

$$Researcher \sqsubseteq \neg Professor$$
$$Researcher \sqsubseteq \neg Lecturer$$
$$\exists TeachesTo^{-} \sqsubseteq Student$$

$$Student \sqcap \neg Undergrad \sqsubseteq GraduateStudent$$
$$\exists TeachesTo.Undergrad \sqsubseteq Professor \sqcup Lecturer$$

TBOX Inferences: $Researcher \sqsubseteq \forall TeachesTo.GraduateStudent$ ⇦ concept subsumption

ABOX:

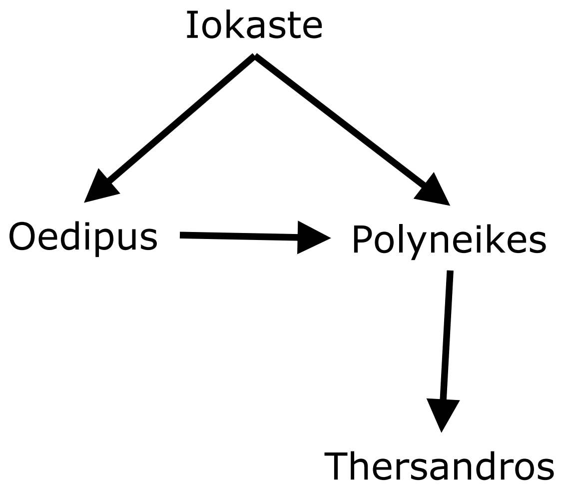
$$TeachesTo(dupond, pierre)$$
$$\neg GraduateStudent(pierre)$$
$$\neg Professor(dupond)$$

Ontology Inferences: $Lecturer(dupond)$ ⇦ concept instance checking

# Open-World Assumption

Something evaluates false **only if** it contradicts other information in the ontology

hasSon(Iokaste,Oedipus)
hasSon(Iokaste,Polyneikes)
hasSon(Oedipus,Polyneikes)
hasSon(Polyneikes,Thersandros)
patricide(Oedipus)
¬patricide(Thersandros)




$\text{Query} \equiv \exists \text{hasSon}.(\text{patricide} \sqcap \exists \text{hasSon}.\neg \text{patricide})$
$\text{ABox} \models \text{Query}(\text{Iokaste})?$

# Modeling with Description Logics

It is hard to build good ontologies with DL

- The names of the classes are irrelevant
- Classes are overlapping by default
- Domain and range definitions are axioms, not constraints
- Open world assumption
  - Anything might be true unless explicit asserted knowledge contradicts it (negation)
- Non-unique name assumption
  - Although families such as the DL-Lite family assume the unique name assumption

In this course, we aim at modeling usual data models and we will solely focus on modeling UML-like TBOXes (like the examples we have seen during this lecture)

# Summary

Description Logics

- TBOX
  - Constructs
  - Formal Semantics
- ABOX
- Reasoning
  - Open-World Assumption