LOD - The logic of Descriptions

Sept 13,2023

LOD - The logic of Descriptions

- Introduction
- Domain
- Language
- Interpretation function
- Entailment
- Reasoning problems
- Entailment properties

LOD - The logic of Descriptions

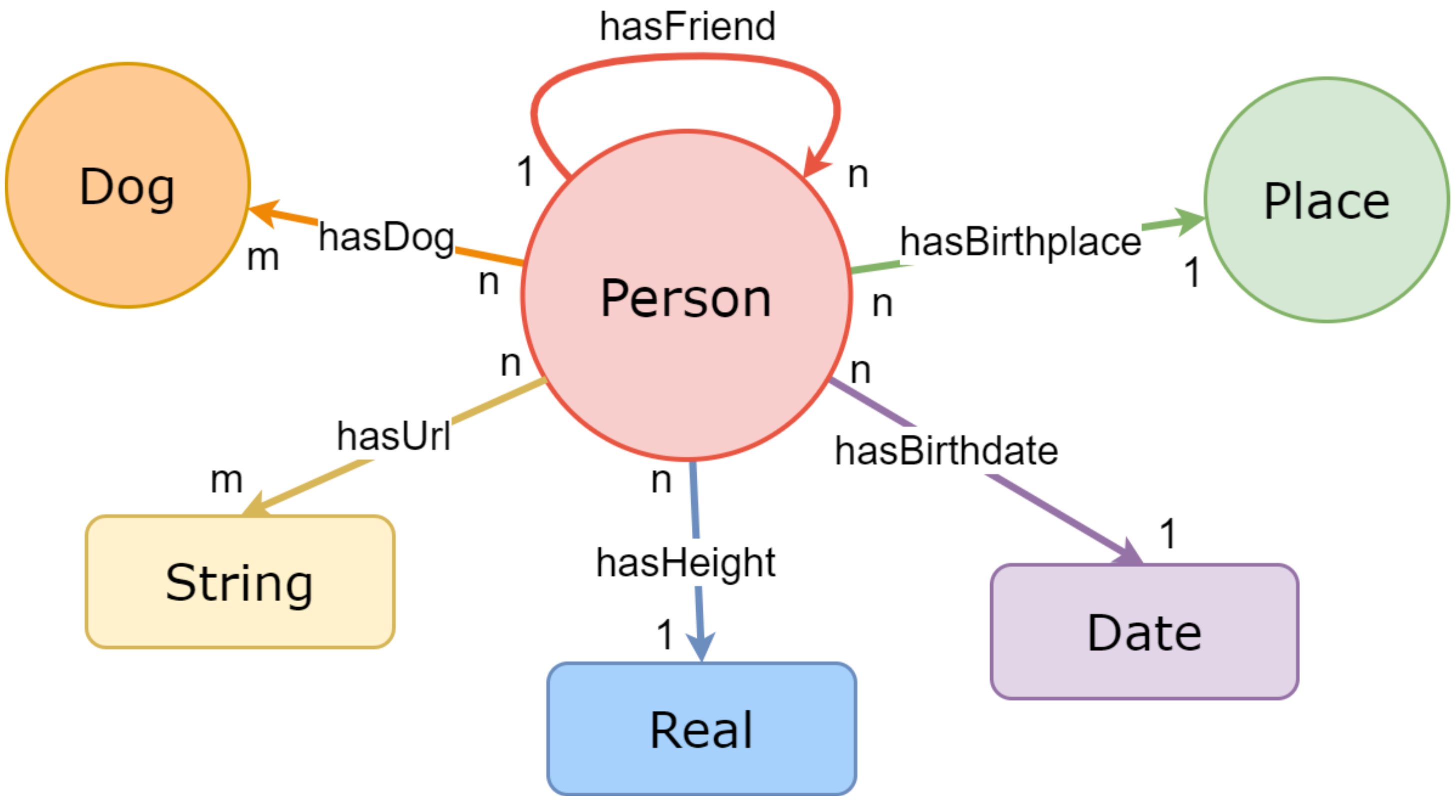
- The Logic of Descriptions (LOD) allows us to reason about the concepts and roles that describe entities in the world.
- Thus, we do not represent and reason about specific entities, but, in a more abstract way, about the classes associated to their properties.
- LOD allows to reason about ETG's.
- Any LOE EG is built with reference to a LOD ETG.
- LOD is conceptually similar to the Logics of Description (DL)

LOD - The logic of Descriptions

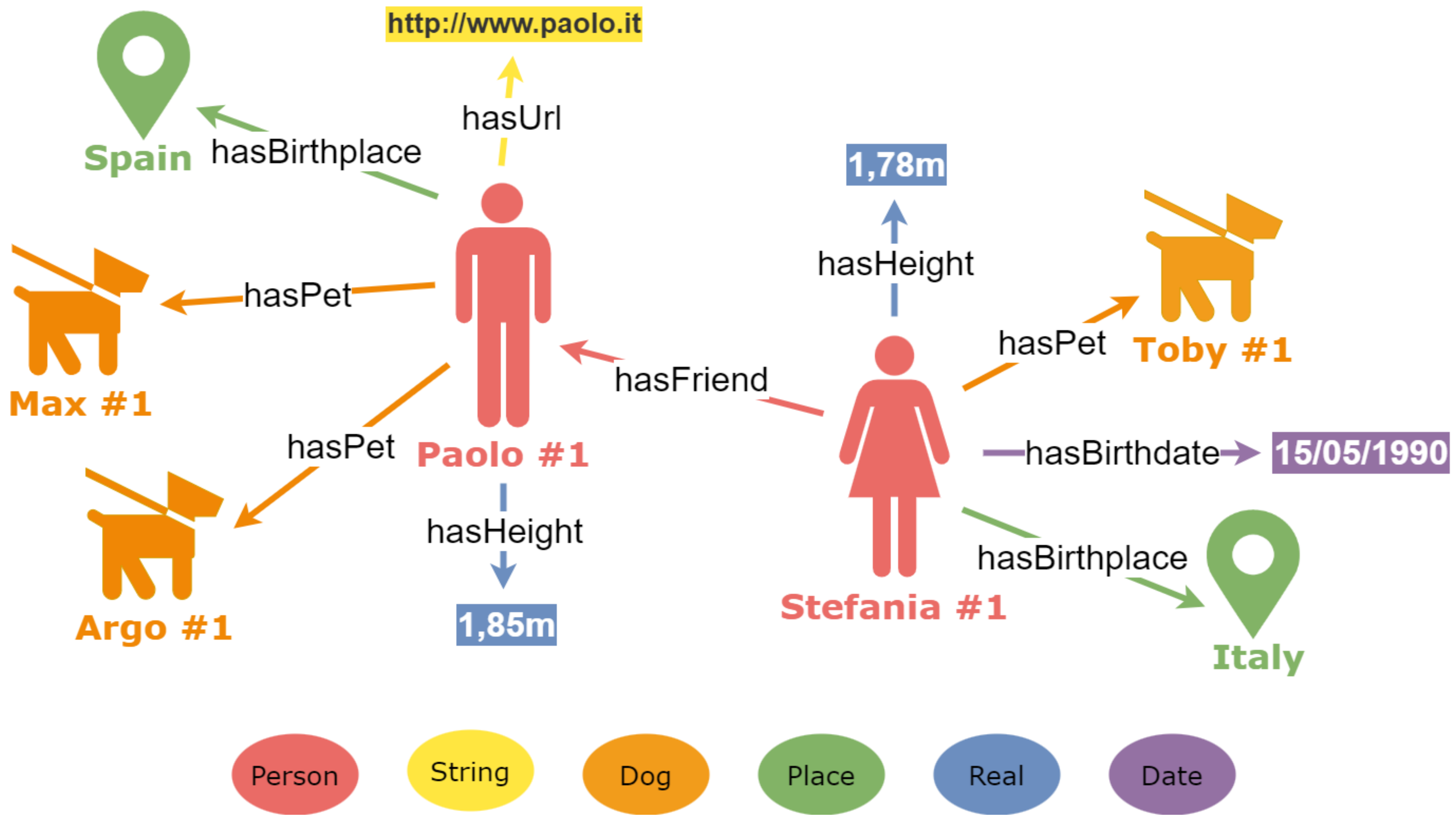
In LOD we have the following ETG fact elements:

- An entity type (etype) is a class of entities (corresponding to the concept to which an entity belongs in a LOE EG);
- A datatype (dtype) is a class of (data) values (corresponding to the dtype to which a value belongs in a LOE EG);
- An Object Property describes a relation between two etypes (not beween two entities, as in LOE)
- A Data Property, also called Attribute, describes a characteristic of an etype (not of an entity as in LOE);

An example of ETG



An example of EG for the previous ETG



LoD – The Logic of Descriptions - definition

We formally define LOD as follows

$$\mathbf{LOD} = \langle \mathbf{ETG}, \models_{\mathbf{LOD}} \rangle$$

with

$$\mathbf{ETG} = \langle \mathbf{L}_{\mathbf{LOD}}, \mathbf{D}_{\mathbf{LOD}}, \mathbf{I}_{\mathbf{LOD}} \rangle$$

Below, any time no confusion arises, we drop the subscripts.

LOD - The logic of Descriptions

- Introduction
- Domain
- Language
- Interpretation function
- Entailment
- Types of theories
- Reasoning problems
- Entailment properties

LoD – Domain/facts

Definition (Domain, intensional definition)

$$Di = < E, \{C\}, \{R\} >$$

where:

$$E = \{e\} \cup \{v\}$$
$$\{C\} = ET \cup DT$$
$$\{R\} = \{OR\} \cup \{DR\}$$

where E is a set of **entities** and **values**, ET = $\{E_T\}$, $E_T = \{e\}$ and DT = $\{D_T\}$, $D_T = \{v\}$ are **sets of entity types (etypes)** and **data types** (**dtypes**), respectively, and OR, DR are **(binary) object** and **data relations**.

Observation. LOD allows for the following facts:

- Every etype ET or dtype DT is a fact, that is ET ⊆ E, DT ⊆ E.
- Every relation R populated by its two arguments is a fact, that is, OR ⊆ ET1 x ET2, DR ⊆ ET x DT.

Facts only have one of the four possible forms above

An example of domain of ETG (continued)

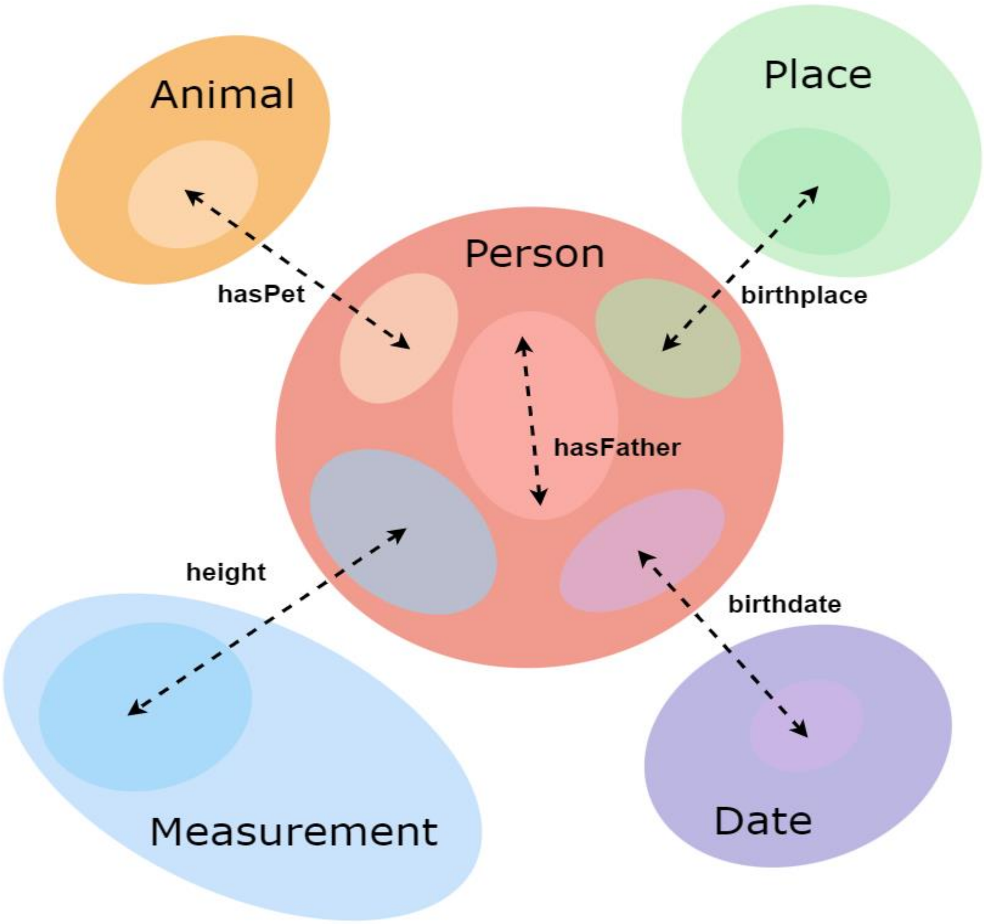
ET = {P, D, L, entity, …}
DT = {Real,String, dtype, …}
{R} = {hF, hD, hH, hB, hL, hU, …}

from which we construct the following facts in the domain:

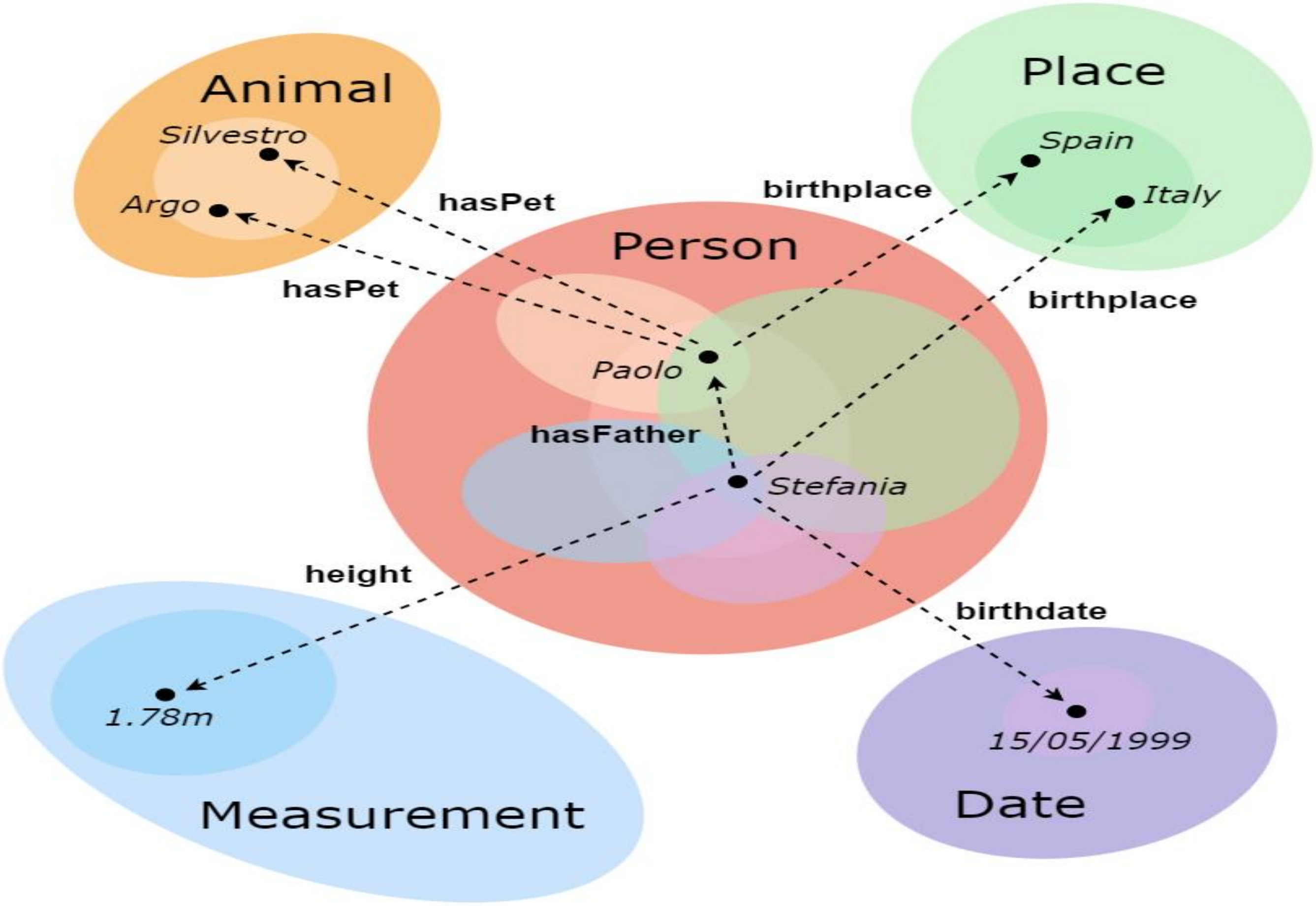
D = {P ⊆ entity, Real ⊆ dtype, hF(P, P), D ⊆ entity, hD(P, D), hH(P, Real), …}

with, e.g., hF(P, P) standing for hF ⊆ P x P

An example of ETG – Venn diagram (continued)



An EG for the example ETG– Venn diagram



LOD - The logic of Descriptions

- Introduction
- Domain
- Language
- Interpretation function
- Entailment
- Reasoning problems
- Entailment properties

LoD – Language/wffs

Definition 11.4 (The language L)

$$L = L\mathit{a} \cup L\mathit{c}$$

with

$$L\mathit{a} = LA \cup LA\mathit{c}$$

Observation 11.2 (LOE ***versus*** LOD**)** LOE allows for atomic assertions. LOD allows for (different) **atomic assertions** (L*A*), for **complex assertions** (L*Ac*) and also **complex formulas** (L*c*).

LoD – Assertions

Definition (The language of atomic assertions LA**)**

$$LA =< A\alpha, WA >$$

where Aα is the alphabet and WA is the set of formation rules for generating cmplex assertions.

Definition 11.6 (Alphabet Aα**)** The alphabet of the atomic formula language contains the etype and dtype names and the names of the object and data properties:

$$A\alpha =< \emptyset, ET \cup DT , \{OP\} \cup \{DP\} >$$

Assertions – BNF production rules

<assertion> ::= <etype> | <dtype>

⊤ | ⊥ |

∃<objProp>(<etype>, <etype>) |

∃<dataProp>(<etype>, <dtype>) |

∀<objProp>(<etype>, <etype>) |

∀<dataProp>(<etype>, <dtype>)

<etype> ::= ET1 | . . . | ETn

<dtype> ::= DT1 | . . . | DTn

<objProp> ::= OP1 | . . . | OPn

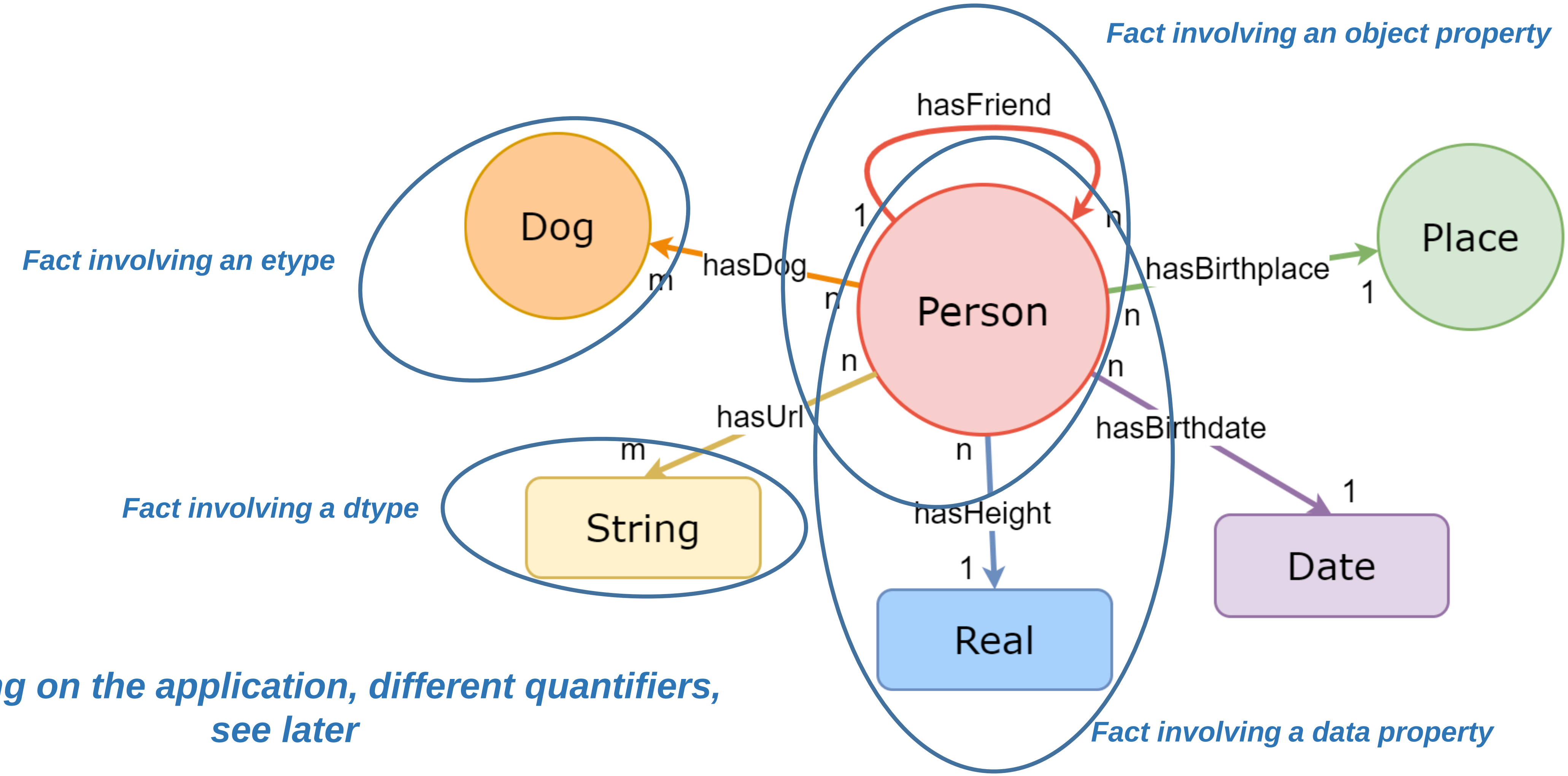
<dataProp> ::= DP1 | . . . | DPn

Compare with LOE

Assertions – Example

- Person *(**Intuition:** the set of entities – in the domain of interpretation – which are called called persons)*
- ∃hasFriend.Person *(**Intuition:** the set of entities which have – at least – one friend who is a person)*
- Real *(**Intuition:** the set of reals)*
- ∃hasHeight.Real *(**Intuition:** the set of entities which have their height - at least one – which measured as a real number)*
- ∀hasFriend.Person *(**Intuition:** the set of entities whose friends are only persons)*

Example – how assertions represent ETG facts



Depending on the application, different quantifiers, see later

LoD – Atomic wffs

Definition (The language of atomic formulas La)

$$\mathrm{L}a =< \mathrm{LA}, \mathrm{W}a >$$

where LA is the language of (atomic) assertions and Wa is a set of formation rules for generating complex assertions.

Definition 11.6 (Alphabet) The alphabet consists of all the formulas in LA

Complex assertions – BNF production rules

<awff> ::= <assertion>

<awff> ::= <awff> ⊓ <awff> |

<awff> ⊔ <awff> |

¬ <awff>

Complex assertions – Example

- Person ⊓ ∃hasFriend.Person *(**Intuition:** the set of entities which are persons and have a friend which is a person)*

- Person ⊔ Dog *(**Intuition:** the set of entities which are a person or a dog)*

- Person ⊓ ¬(∃hasFriend.Person) *(**Intuition:** the set of entities which are persons and which do not have a friend which is a person)*

Complex assertions – Example concept names

Consider the following concept names:

Vehicle, Boat, Bicycle, Car, Device, Wheel, Engine, Axle, Rotation,
Water, Human, Driver, Adult, Child

Formalize the following natural language statements:

- Nothing (empty set): $\bot$
- Everything (All the interpretation domain): $\top$
- Humans and vehicles: Human $\sqcap$ Vehicle
- Vehicles and not boats: Vehicle $\sqcap \neg$ Boat
- Wheels or engines and humans: (Wheel $\sqcup$ Engine) $\sqcap$ Human
- Adults or children: Adult $\sqcup$ Child

Complex assertions – Example roles

Consider the previous concept names plus the following role names:

hasPart, poweredBy, capableOf, travelsOn, controls

Formalize in DL the following natural language statements:

1. Those vehicles that have wheels and are powered by an engine
2. Those vehicles that have wheels and are powered by a human
3. Those vehicles that travel on water
4. Those objects which have no wheels
5. Those objects which do not travel on water
6. Those devices that have an axle and are capable of rotation
7. Those humans who control a vehicle
8. The drivers of cars

Complex assertions – Example roles

1. Vehicle $\sqcap$ $\exists$hasPart.Wheel $\sqcap$ $\exists$poweredBy.Engine
2. Vehicle $\sqcap$ $\exists$hasPart.Wheel $\sqcap$ $\exists$poweredBy.Human
3. Vehicle $\sqcap$ $\exists$travelsOn.Water
4. $\forall$hasPart.$\neg$Wheel
5. $\forall$travelsOn.$\neg$Water
6. Device $\sqcap$ $\exists$hasPart.Axle $\sqcap$ $\exists$capableOf.Rotation
7. Human $\sqcap$ $\exists$controls.Vehicle
8. Driver $\sqcap$ $\exists$controls.Car

LoD – complex wffs (the full language)

Definition (The language of complex formulas Lc)

$$\mathrm{L}c = < \mathrm{L}a, \mathrm{W}c >$$

where La is the language of complex assertions from above and Wc is a set of formula constructors

Definition (Alphabet) The alphabet is all the atomic formulas (atomic and complex assertions) in La

Complex formulas – BNF production rules

<cwff> ::= <concept> ⊑ <awff> |
<concept> ≡ <awff>

where:

- <concept> : we restrict <concept> to be an etype
- ⊑ : subsumption relation
- ≡ : equivalence relation

NOTE: In most common logics in the literature we have <awff> instead of <concept>.

Complex formulas

- <concept> ⊑ <awff>
  - A **concept inclusion** (formula)
  - *To be read* <concept> *is subsumed by* <awff>
- <concept> ≡ <awff>
  - A **concept definition** (formula)
  - *To be read* <concept> *is equivalent to* <awff>

Complex formulas – concept inclusion examples

1. Boats have no wheels
2. Cars and bicycles do not travel on water
3. Drivers of cars are adults
4. Humans are not vehicles
5. Wheels or engines are not humans
6. Humans are either adults or children
7. Adults are not children

Complex formulas – concept inclusion examples

1. $\text{Boat} \sqsubseteq \forall \text{hasPart}.\neg \text{Wheel}$
2. $\text{Car} \sqcup \text{Bicycle} \sqsubseteq \forall \text{travelsOn}.\neg \text{Water}$
3. $\text{Driver} \sqcap \exists \text{controls}.\text{Car} \sqsubseteq \text{Adult}$
4. $\text{Human} \sqsubseteq \neg\ \text{Vehicle}$
5. $\text{Wheel} \sqcup \text{Engine} \sqsubseteq \neg\ \text{Human}$
6. $\text{Human} \sqsubseteq \text{Adult} \sqcup \text{Child}$
7. $\text{Adult} \sqsubseteq \neg \text{Child}$

Complex formulas – definition examples

1. Cars are exactly those vehicles that have wheels and are powered by an engine
2. Bicycles are exactly those vehicles that have wheels and are powered by a human
3. Boats are exactly those vehicles that travel on water
4. Wheels are exactly those devices that have an axle and are capable of rotation
5. Drivers are exactly those humans who control a vehicle

Complex formulas – definition examples

1. Car $\equiv$ Vehicle $\sqcap$ $\exists$hasPart.Wheel $\sqcap$ $\exists$poweredBy.Engine
2. Bicyle $\equiv$ Vehicle $\sqcap$ $\exists$hasPart.Wheel $\sqcap$ $\exists$poweredBy.Human
3. Boat $\equiv$ Vehicle $\sqcap$ $\exists$travelsOn.Water
4. Wheel $\equiv$ Device $\sqcap$ $\exists$hasPart.Axle $\sqcap$ $\exists$capableOf.Rotation
5. Driver $\equiv$ Human $\sqcap$ $\exists$controls.Vehicle

LOD - The logic of Descriptions

- Introduction
- Domain
- Language
- Interpretation function
- Entailment
- Reasoning problems
- Entailment properties

Interpretation of atomic formulas

$I(\top) = \mathbb{D}$

$I(\bot) = \emptyset$

$I(A \sqcap B) = I(A) \cap I(B)$

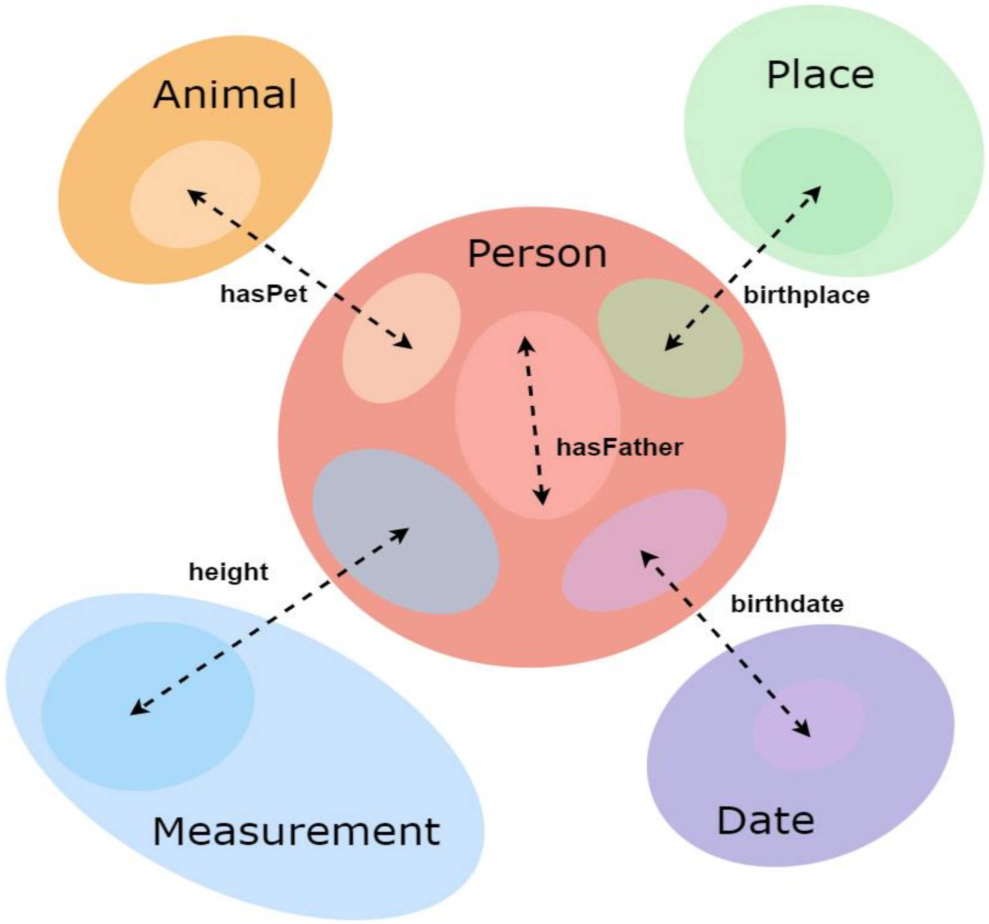
$I(A \sqcup B) = I(A) \cup I(B)$

$I(\neg A) = \mathbb{D} \setminus I(A)$

$I(\exists R.A) = \{d \in \mathbb{D} \mid$ there is an $e \in \mathbb{D}$ with $(d, e) \in I(R)$ and $e \in I(A)\}$

$I(\forall R.A) = \{d \in \mathbb{D} \mid$ for all $e \in \mathbb{D}$ if $(d, e) \in I(R)$ then $e \in I(A)\}$

Interpretation function (Venn diagram) – example above



Most often we assume both universal and existential quantifier

The first does not imply the second (when premise of the first is never satisfied)

LOD - The logic of Descriptions

- Introduction
- Domain
- Language
- Interpretation function
- Entailment
- Types of theories
- Reasoning problems
- Entailment properties

Entailment relation

Definition (Entailment |=)

- M $|= w1 \sqsubseteq w2$ iff $I(w1) \subseteq I(w2)$
- M $|= w1 \equiv w2$ iff $I(w1) = I(w2)$
  iff $I(w1) \subseteq I(w2)$ and $I(w2) \subseteq I(w1)$

With $w1, w2 \in L$;

NOTE: $w1$ is not necessarily an assertion

Entailment relation (extended)

Definition (Entailment |=)

1. M $\models w1 \sqsubseteq w2$ iff $I(w1) \subseteq I(w2)$
2. M $\models w1 \equiv w2$ iff $I(w1) = I(w2)$
   iff $I(w1) \subseteq I(w2)$ and $I(w2) \subseteq I(w1)$
3. M $\models w1 \sqsupseteq w2$ iff $I(w2) \subseteq I(w1)$
4. M $\models w1 \perp w2$ iff $I(w1) \cap I(w2) \subseteq \emptyset$

with

- $w1, w2 \in L$;
- $w1 \sqsupseteq w2$ a notational variant of $w2 \sqsubseteq w1$;
- $w1 \perp w2$ a notation for $w1 \sqcap w2 \sqsubseteq \perp$ (a special case of $w1 \sqsubseteq w2$)

LOD - The logic of Descriptions

- Introduction
- Domain
- Language
- Interpretation function
- Entailment
- Reasoning problems
- Entailment properties

Reasoning problems (definition)

- Model checking
- Satisfiability with respect to T
- Subsumption with respect to T
- Equivalence with respect to T
- Disjointness with respect to T

Model checking

Model checking. Given I, M is a model of a theory T, where T is a set of complex formulas, if the following two conditions hold.

- If $C \sqsubseteq D \in T$, then $I(C) \subseteq I(D)$
- If $C \equiv D \in T$, then $I(C) = I(D)$

NOTE: A model checking problem

Satisfiability

Satisfiability with respect to T. A complex assertion C is satisfiable with respect to T if there exists an interpretation function I of T such that I(C) is nonempty.

In this case we say also that I is a model of C, with respect to T.

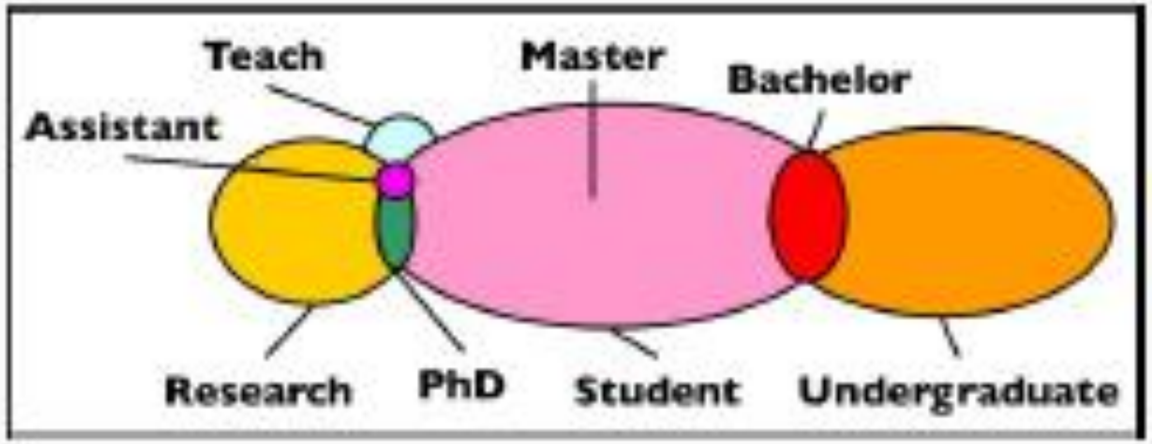
NOTE 1: T can also be empty (as with all the next reasoning problems)

NOTE 2: A satisfiability problem (I builds the model)

Satisfiability

Consider the Tbox

$$\mathcal{T} = \begin{cases} Undergraduate \sqsubseteq \neg Teach \\ Bachelor \equiv Student \sqcap Undergraduate \\ Master \equiv Student \sqcap \neg Undergraduate \\ PhD \equiv Master \sqcap Research \\ Assistant \equiv PhD \sqcap Teach \end{cases}$$



Satisfiability

Example 1

Is Bachelor ⊓ PhD satisfied by $\mathcal{T}$? **(No)**

The problem can be formalized as:

$$\mathcal{T} \models Bachelor \sqcap PhD$$

Proof:

Bachelor ⊓ PhD

≡ (Student ⊓ Undergraduate) ⊓ (Master ⊓ Research)

≡ (Student ⊓ Undergraduate) ⊓ ((Student ⊓ ¬ Undergraduate) ⊓ Research)

≡ Student ⊓ **Undergraduate** ⊓ ¬ **Undergraduate** ⊓ Research

≡ Student ⊓ ⊥ ⊓ Research

Subsumption

Subsumption with respect to T A complex assertion C is subsumed by a complex assertion D with respect to T if

$$I(C) \subseteq I(D)$$

for every I (used to build models for T, with T possibly empty) .

In this case we write

$$C \sqsubseteq_T D$$

or

$$T \models C \sqsubseteq D$$

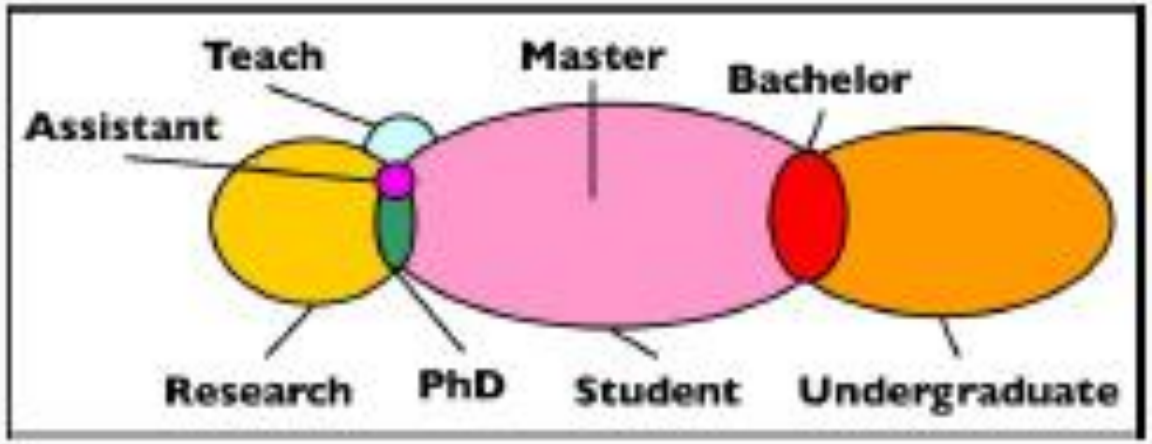
NOTE: A validity problem

Subsumption

Consider the Tbox

$$\mathcal{T} = \begin{cases} Undergraduate \sqsubseteq \neg Teach \\ Bachelor \equiv Student \sqcap Undergraduate \\ Master \equiv Student \sqcap \neg Undergraduate \\ PhD \equiv Master \sqcap Research \\ Assistant \equiv PhD \sqcap Teach \end{cases}$$

It should be checked for all models of T



Subsumption

Example 2

Is PhD $\sqsubseteq$ Student satisfiable? (Yes)

The problem can be formalized as:

$$\mathcal{T} \models PhD \sqsubseteq Student$$

Proof:

PhD

$\equiv$ Master $\sqcap$ Research

$\equiv$ (Student $\sqcap \neg$ Undergraduate) $\sqcap$ Research

$\sqsubseteq$ Student

Equivalence

Equivalence with respect to T. Two complex assertions C and D are equivalent with respect to T if

$$I(C) = I(D)$$

for every model I (used to build models for T, with T possibly empty) .

In this case we write

$$C \equiv_T D$$

or

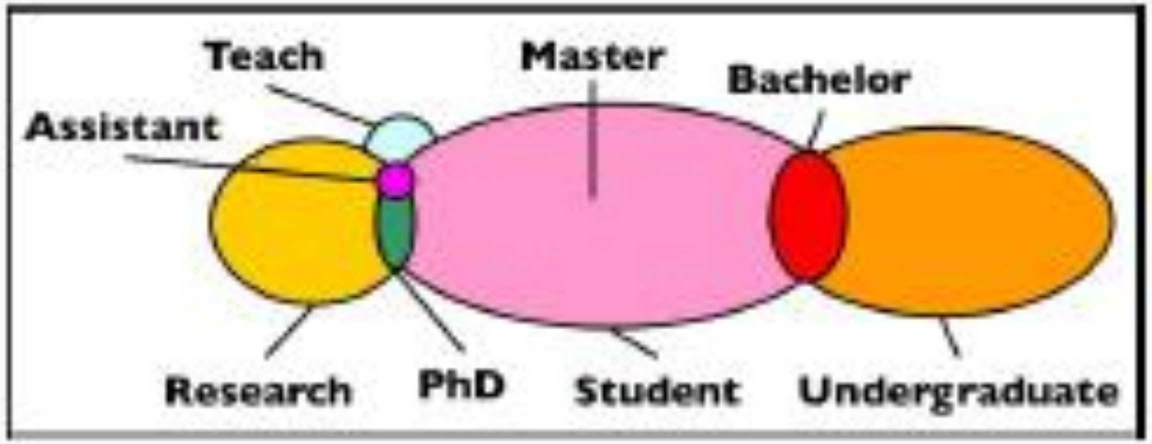
$$T \models C \equiv D$$

NOTE: A validity problem

Equivalence

Consider the Tbox

$$\mathcal{T} = \begin{cases} Undergraduate \sqsubseteq \neg Teach \\ Bachelor \equiv Student \sqcap Undergraduate \\ Master \equiv Student \sqcap \neg Undergraduate \\ PhD \equiv Master \sqcap Research \\ Assistant \equiv PhD \sqcap Teach \end{cases}$$



It should be checked for all models of T

Equivalence

Example 3

$$\text{Is Student} \equiv \text{Bachelor} \sqcup \text{Master consistent with } \mathcal{T}\text{? (Yes)}$$

The problem can be formalized as:

$$\mathcal{T} \models \mathit{Student} \equiv \mathit{Bachelor} \sqcup \mathit{Master}$$

Proof:

$$\begin{aligned}
&\text{Bachelor} \sqcup \text{Master} \\
&\equiv (\text{Student} \sqcap \text{Undergraduate}) \sqcup (\text{Student} \sqcap \neg \text{Undergraduate}) \\
&\equiv \text{Student} \sqcup (\text{Undergraduate} \sqcap \neg \text{Undergraduate}) \\
&\equiv \text{Student} \sqcup \top \\
&\equiv \text{Student}
\end{aligned}$$

Disjointness

Disjointness with respect to T. Two complex assertions C and D are disjoint with respect to T if

$$I(C) \cap I(D) = \emptyset$$

for every I (used to build models for T, with T possibly empty) .

In this case we write

$$C \perp_T D$$

or

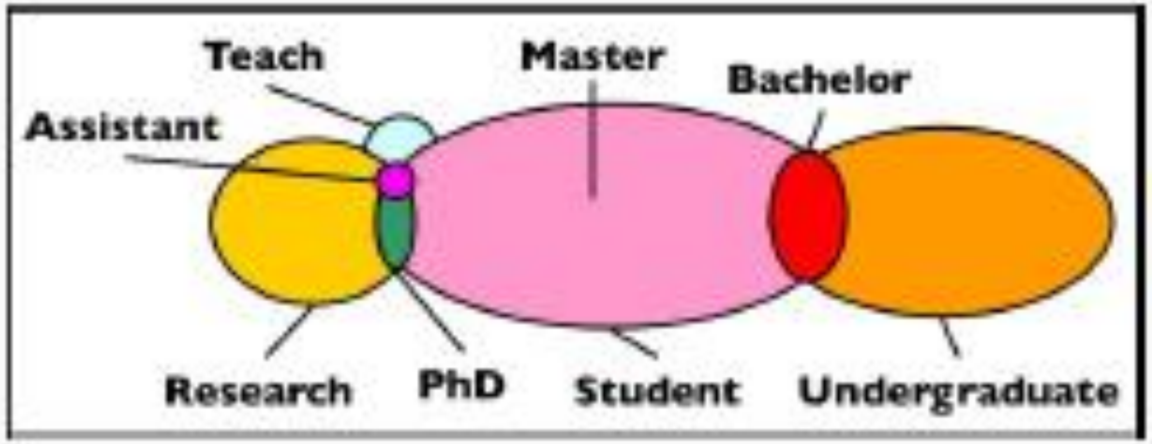
$$T \models C \perp D$$

NOTE: A validity problem

Disjointness

Consider the Tbox

$$\mathcal{T} = \begin{cases} Undergraduate \sqsubseteq \neg Teach \\ Bachelor \equiv Student \sqcap Undergraduate \\ Master \equiv Student \sqcap \neg Undergraduate \\ PhD \equiv Master \sqcap Research \\ Assistant \equiv PhD \sqcap Teach \end{cases}$$



It should be checked for all models of T

Disjointness

Example 4

Is Undergraduate $\sqcap$ Assistant $\sqsubseteq \bot$ consistent with $\mathcal{T}$? (Yes)

The problem can be formalized as:

$$\mathcal{T} \models \mathit{Undergraduate} \sqcap \mathit{Assistant} \sqsubseteq \bot$$

Proof:

$\text{Undergraduate} \sqcap \text{Assistant}$

$\sqsubseteq \neg\,\text{Teach} \sqcap \text{Assistant}$

$\equiv \neg\,\text{Teach} \sqcap (\text{PhD} \sqcap \text{Teach})$

$\equiv \bot \sqcap \text{PhD}$

$\equiv \bot$

Reasoning problems (Reduction)

- **Model checking.** Core Q/A functionality (see LOE)
- **Equivalence**. C $\equiv_T$ D iff C $\sqsubseteq_T$ D and D $\sqsubseteq_T$ C
- **Subsumption**. C $\sqsubseteq_T$ D iff C $\sqcap \neg$D is *unsatisfiable* with respect to T
- **Disjointness.** C $\perp$T D iff C $\sqcap$ D is *unsatisfiable*

Observation

- LOD reasoning can be implemented as *LOD satisfiability* (see above)
- LOD satisfiability can be implemented as Truth Table satisfiability (see later)

LOD - The logic of Descriptions

- Introduction
- Domain
- Language
- Interpretation function
- Entailment
- Reasoning problems
- Entailment properties

Observations (Logical entailment – properties)

Intuition (Reflexivity): $w \mid= w$ **YES!**

Intuition (Cut): If $\Gamma \mid= w1$ and $\Sigma \cup \{w1\} \mid= w2$ then $\Gamma \cup \Sigma \mid= w2$ **YES!**

Intuition (Compactness)

If $\Gamma \mid= w$ then there is a finite subset $\Gamma 0 \subseteq \Gamma$ such that $\Gamma 0 \mid= w$. **YES!**

Intuition (Monotonicity): If $\Gamma \mid= w$ then $\Gamma \cup \Sigma \mid= w$ **YES!**

Intuition (NonMonotonicity) $\Gamma \mid= w$ and $\Gamma \cup \Sigma$ not$\mid= w$ **NO!**

LOD - The logic of Descriptions

Sept 13,2023