Foundations of Semantic Knowledge Graphs

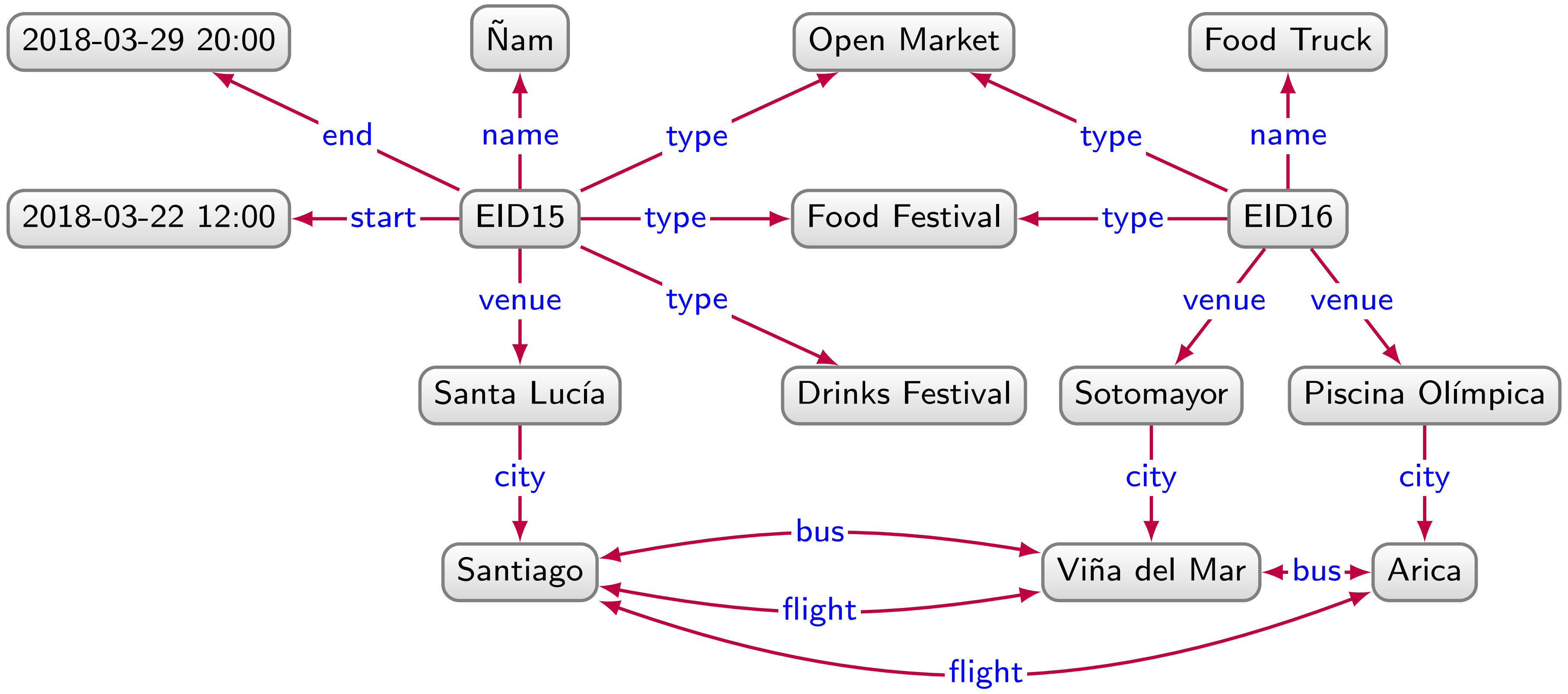
Prof. Dr. Stefan Linus Zander

Deductive Knowledge and Ontologies

Outline

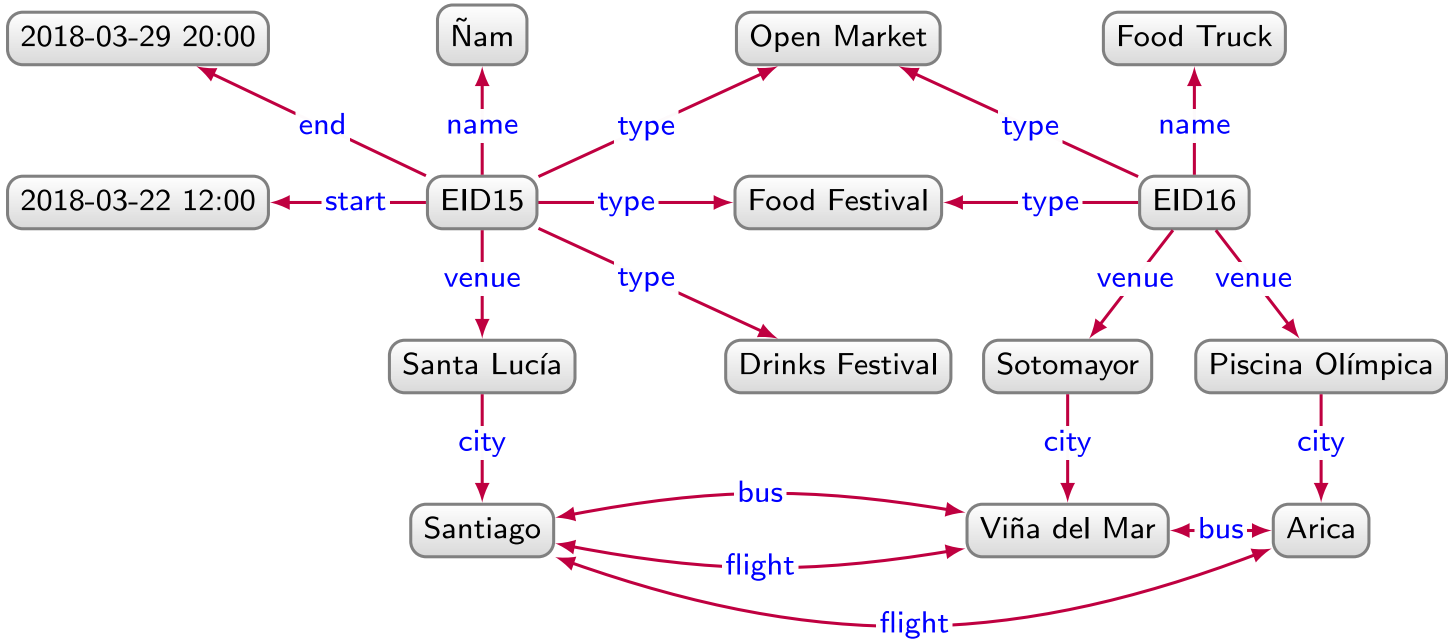
- Knowledge Types for Deduction
- Ontologies
- Interpretations and Models
- Ontology Language Features
- Basic Reasoning Types

What additional information can we deduce from the following graph ?



Source: Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, Antoine Zimmermann (2021) Knowledge Graphs, Synthesis Lectures on Data, Semantics, and Knowledge, No. 22, 1–237, DOI: 10.2200/S01125ED1V01Y202109DSK022, Springer.

We can deduce more from a graph than what edges explicitly indicate



Examples:

- Ñam festival (`EID15`) will be located in Santiago, even though the graph does not contain an edge `EID15-location→Santiago`
- The cities connected by flights must have some airport nearby, even though the graph does not contain nodes referring to these airports.

Given **data** as premise + some **general rules** we know a priori → we can use a **deductive process** to **derive new data**.

Humans derive new knowledge based on commonsense and domain knowledge

Commonsense Knowledge

- general premise and rules
- known a priori
- shared by many people
- knowledge that is universally justifed

Domain Knowledge

- knowledge shared by few experts in an area
  - *e.g. an expert in biology may know that* hemocyanin *is a protein containing copper that carries oxygen in the blood of some species of* Mollusca *and* Arthropoda
- encoded in very specific domain models

⚠ Machines, in contrast, do not have a priori access to such deductive faculties

- Machines need **formal instructions** in terms of premises and entailment regimes to produce similar deductions w.r.t. humans
- In this way, we will be making more of the **meaning** (i.e., semantics) of the graph **explicit** in a **machine-readable format**.
- Entailment regimes formalise the conclusions that logically follow as a consequence of a given set of premises.

Entailment regimes formalise the conclusions that logically follow as a consequence of a given set of premises

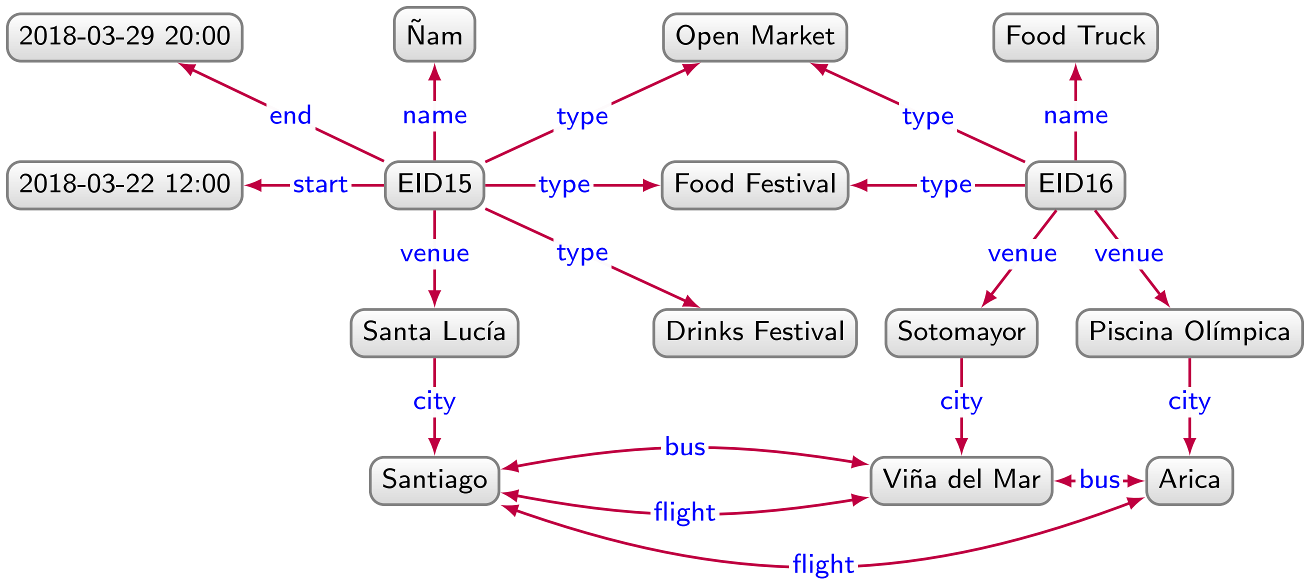
These **deductions** may serve a range of applications, such as

- improving query answering
- (deductive) classification
- finding inconsistencies
- deducing class memberships
- etc.

Once instructed, machines can (often) apply deductions with a precision, efficiency, and scale beyond human performance.

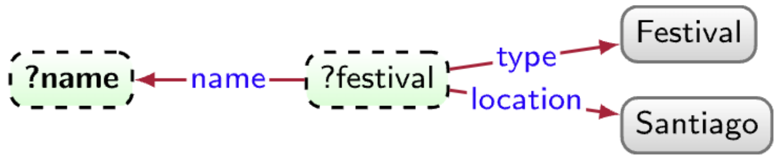
Example: Using entailment regimes for query answering

Premise: Example data graph with explicit statements



Source: *https://kgbook.org/#chap-deductive*

Query: "find festivals located in Santiago"



Explanations

- Without entailment regimes, the graph pattern would return no results
  - no node of type `festival`
  - nothing has directly the location `Santigao`
- `Ñam` could be automatically entailed if we stated that
  - x being a `Food Festival` entails that x is a `Festival`, or
  - x having venue y in city z entails that x has location z

How Ontologies can help us to automatically compute entailments based on formal semantics

We will learn about ways in which **more complex entailments** can be expressed and automated.

A number of **logical frameworks** could be leverages for these purposes
⇝ e.g. First-Order Logic, Datalog, Prolog, Answer Set Programming etc.

We focus on ontologies, which...

- ...constitute a formal representation of knowledge
- ...can be represented as a graph

We discuss...

- ...how ontologies can be formally defined,
- ...how they can be created using ontology languages, and
- ...how new facts can be deduced via reasoning based on entailment regimes

What is an Ontology ?

To enable entailment, we must be precise about what the terms we use mean.

Definition An ontology is a concrete, formal representation of what terms mean within the scope in which they are used (e.g., a given domain), formulated using an ontology language.

Source: based on https://kgbook.org/#ssec-ontologies

- The term stems from the philosophical study of ontology, concerning the kinds of entities that exist, the nature of their existence, what kinds of properties they have, and how they may be identified and categorised.
- Ontologies can guide how graph data are modelled.
- Given that ontologies are formal representations, they can be used to automate entailment.
- The **usefulness** of an ontology depends on
  - the level of agreement on what that ontology defines,
  - how detailed it is, and
  - how broadly and consistently it is adopted $\rightsquigarrow$ thus enhancing interoperability.

A widely accepted definition of ontologies in CS

Definition An ontology is a formal, explicit specification of a shared conceptualization.

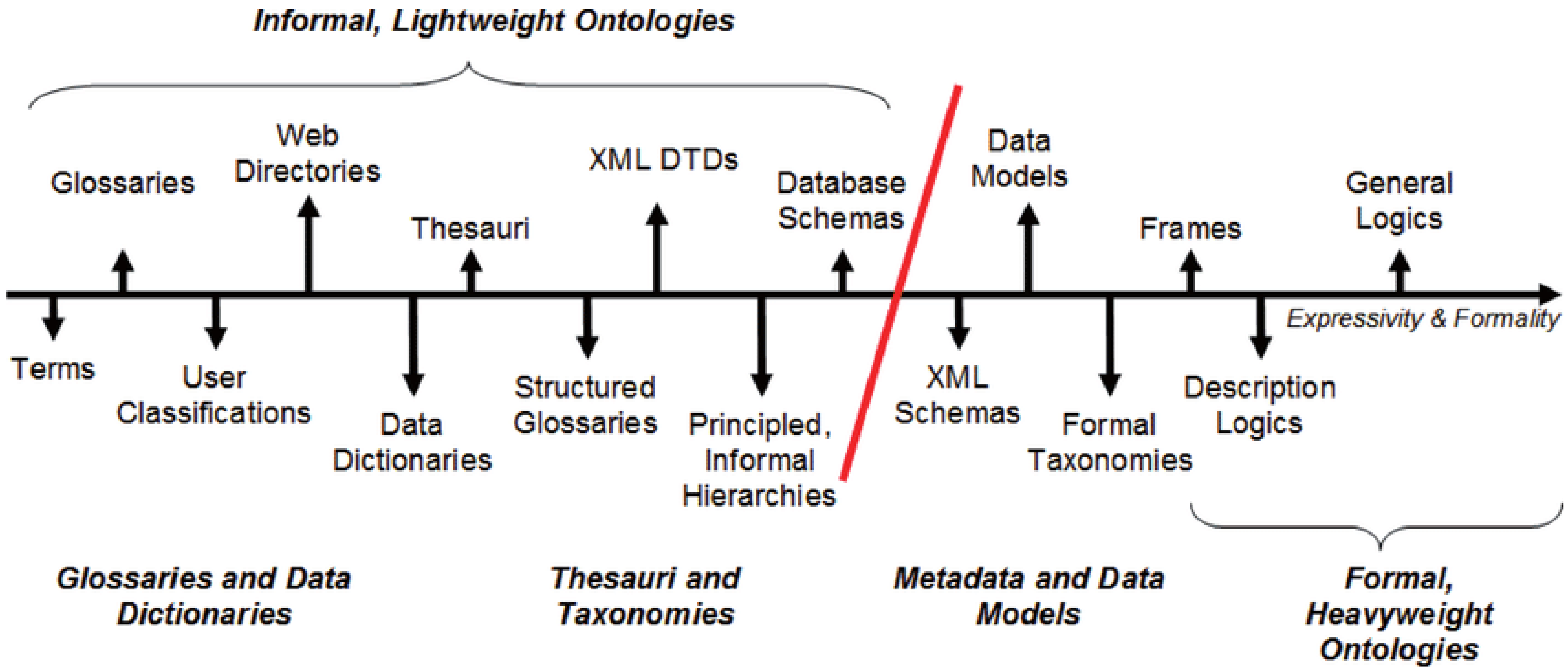
Source: Studer, Benjamins, Fensel. "Knowledge Engineering: Principles und Methods." Data und Knowledge Engineering. 25 (1998) 161-197 based on Tom Gruber 1993.

- **formal**
  - based on mathematics and logics
  - interpretable by machines
  - specific algorithms are able to compute the correctness (ie satisfiability) of an ontology
- **explicit specification**
  - described in axiomatic forms, ie., semantics of the terms are expressed in form of logical axioms
  - An ontology language is used for expressing elements of a domain
- **shared conceptualisation**
  - common understanding about the elements and constituents of a domain
  - created by a group of experts
  - reflect consensual knowlegde plus a shared committment

Additional popular ontology definitions

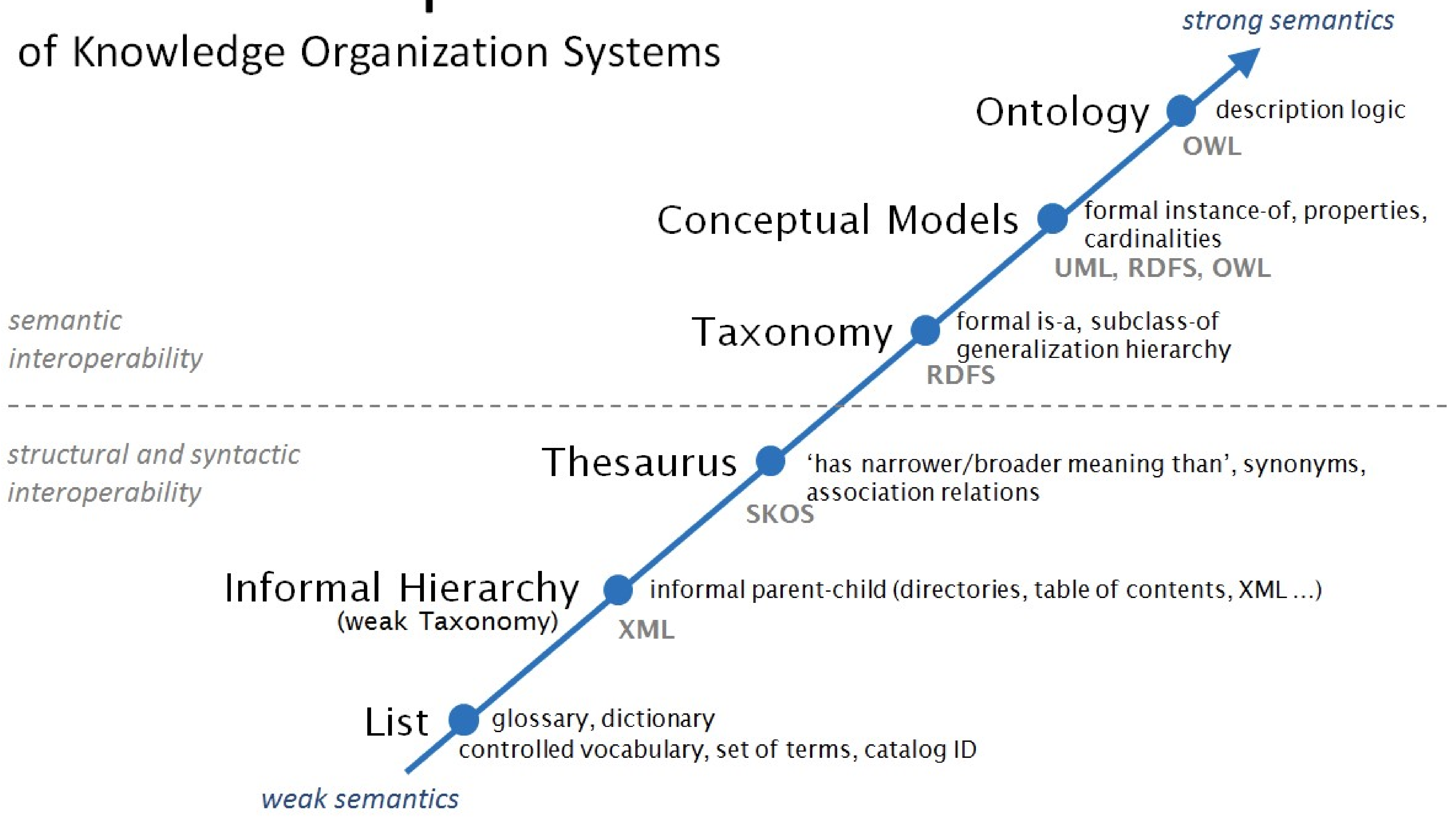
1. *"An ontology defines the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary."*
   Neches, R.; Fikes, R.; Finin, T.; Gruber, T.; Patil, R.; Senator, T.; Swartout, W.R. Enabling Technology for Knowledge Sharing. AI Magazine. Winter 1991. 36-56
2. *"An ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base."*
   B. Swartout; R. Patil; k. Knight; T. Russ. Toward Distributed Use of Large-Scale Ontologies. Ontological Engineering. AAAI-97 Spring Symposium Series. 1997. 138-148
3. *"An ontology provides the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base."*
   A. Bernaras; I. Laresgoiti; J. Correra. Building und Reusing Ontologies for Electrical Network Applications. ECAI96. 12th European Conference on Artificial Intelligence. Ed. John Wiley & Sons, Ltd. 298-302

The Ontology Spectrum



Semantic Spectrum

of Knowledge Organization Systems


strong semantics
Ontology
description logic
OWL
Conceptual Models
formal instance-of, properties, cardinalities
UML, RDFS, OWL
semantic interoperability
Taxonomy
formal is-a, subclass-of generalization hierarchy
RDFS
structural and syntactic interoperability
Thesaurus
'has narrower/broader meaning than', synonyms, association relations
SKOS
Informal Hierarchy (weak Taxonomy)
informal parent-child (directories, table of contents, XML ...)
XML
List
glossary, dictionary
controlled vocabulary, set of terms, catalog ID
weak semantics

Ontologies Summary

- An ontology is a set of axioms
- Axioms describe the formal semantics of the concepts used in and defined by an ontology
- Concepts are organized in a taxonomical classification system and identified via their symbols
- An ontology language defines the set of ontology lanugage elements that can be used for the formulation of axioms
- The expressivity of an ontology language is determined by description logic upon which it is built (cf. ALC, EL++, SHOIN(D), SROIQ(D) ...)

Excursus: Interpretations and Models

Interpretations and Models

In order to understand the semantics of a data graph, we map elements of it to entities in the real world $\longrightarrow$ this process is called interpreation.

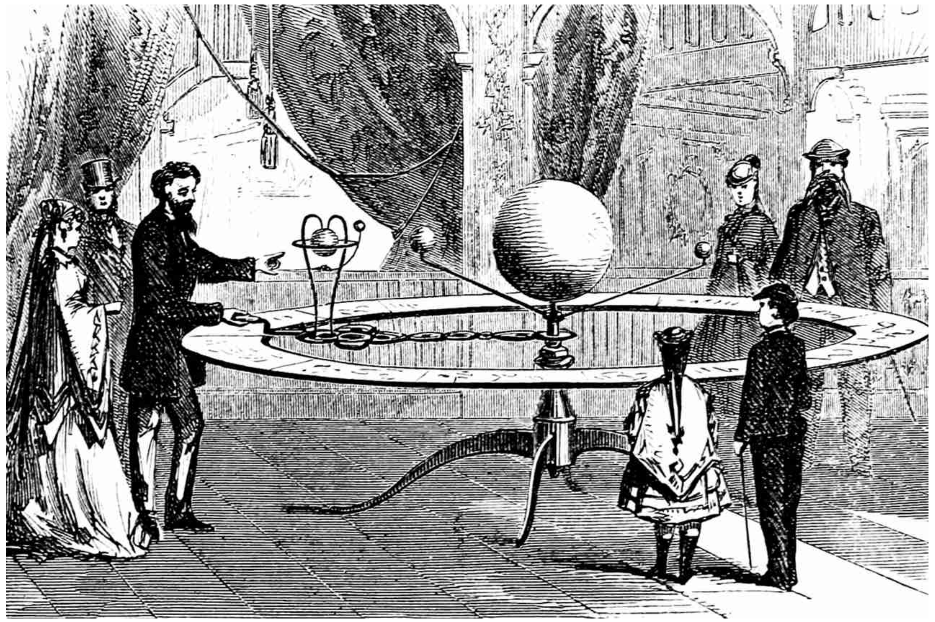
By interpreting a data graph, we create a domain graph that connects real-world entities with real-world relations and reflects the structure and elements of the data graph.

The process of interpretation involves mapping the nodes and edges in the data graph to nodes (ie entities) and edges (ie relations) of the domain graph.

Along these lines, we can abstractly define an interpretation of a data graph as being composed of two elements:

1. a **domain graph**
2. and a **mapping** from the terms (nodes and edges-labels) of the data graph to those of the domain graph.

The domain graph follows the same model as the data graph.



Why Interpretations are useful

The distinction between nodes/edges and entities/relations becomes important, when we define the meaning of ontology features and entailment.

- **Example**
  e.g. if we ask whether there is an edge labelled `flight` between `Arica` and `Viña del Mar` for the data graph, the answer is no. However, if we ask if the entities `Arica` and `Viña del Mar` are connected by the relation `flight`, then the answer depends on what assumptions we make when interpreting the graph.
  - Under the Closed World Assumption (CWA), if we do not have additional knowledge, then the answer is 'no' – since what is not known is assumed to be false.
  - Under the Open World Assumption (OWA), we cannot be certain that this relation does not exist as this could be part of some knowledge not (yet) described by the graph
  - Under the Unique Name Assumption (UNA), the data graph describes at least two flights to `Santiago`
  - Under No Unique Name Assumption (NUNA), we can only say that there is at least one such flight since `Viña del Mar` and `Arica` may be the same entity with two "names".

Why Interpretations are useful

These assumptions (or lack thereof) define which interpretations are valid, and which interpretations satisfy which data graphs.

We call an interpretation that satisfies a data graph a model of that data graph.

The UNA forbids interpretations that map two data terms to the same domain term.
The NUNA allows such interpretations.

Under the CWA, an interpretation that contains an edge $x \rightarrow p \rightarrow y$ in its domain graph can only satisfy a data graph from which we can entail x–p➛y.
Under the OWA, an interpretation containing the edge $x \rightarrow p \rightarrow y$ can satisfy a data graph not entailing x–p➛y so long it does not explicitly contradict that edge.

OWL adopts the **NUNA** and **OWA**, which is the most general case:

- multiple nodes/edge-labels in the graph may refer to the same entity/relation-type (per the NUNA), and
- anything not entailed by the data graph is not assumed to be false as a consequence (per the OWA).

Interpretations

The OWA and NUNA allow an infinite number of possible interpretations (models) for a data graph.

However, the addition of axioms to a model limits the number of possible interpretations.

E.g. by declaring a property as irreflexiv, the following assertion is no longer a possible model of the graph:

```
:x      :y          :z .  // assertion in the data graph
:a      :a          :a .  // is a model of the graph (the OWA and NUNA allows to map multiple nodes to one entity in the model)

as long as the following assertion holds

x = y = z = a

By adding addtional axioms with semantic conditions to the graph, we restrict models for the graph:

:x      :y          :z .
:y      rdf:type    owl:IrreflexiveObjectProperty .

With these axioms, the above interpretation is no longer a model of the graph as it breaks the irreflexive axiom condition.
```

Ontology Features

- ...for Individuals
- ...for Properties
- ...for Classes

Ontology Features: Introduction

Premise

- We can associate certain patterns in the data graph with semantic conditions that define which interpretations satisfy it.

Example

- E.g. we can add a semantic condition to enforce that if our data graph contains the edge `:p rdfs:subPropertyOf :q`, then any edge `:x :p :y` in the domain graph of the interpretation must also have a corresponding edge `:x :q :y` to satisfy the data graph.

```
Data graph:                 :p     rdfs:subPropertyOf    :q .    (a)
                            :x         :q                :y .    (b)

Semantic Condition:         (c)    →    (a)    ∧    (b) .

Domain graph:               :x     :p     :y .                   (c)
```

These **semantic conditions** then form the features of an ontology language.
We will discuss such features by means of the ontology language OWL in the upcomming chapter.

Ontology Features for Individuals

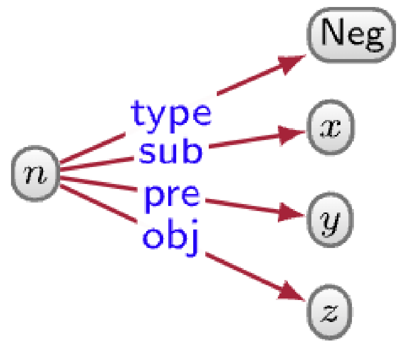
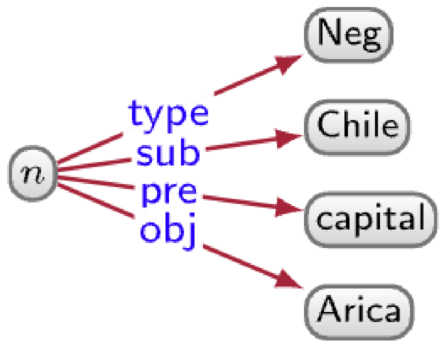
Ontology Features for Individuals

OWL provides the following features for describing the formal, model-theoretic semantics of individuals.

- **Assertions**: we can assert (binary) relations between individuals using edges
  - e.g. `:Santa Lucía  :hasCity  :Santiago .`
  - ⇝ i.e. we refer to the condition that the relation is given in the domain graph of the interpretation; if so, the interpretation satisfies the axiom.
- **Same entity**: Based on the OWA, we can state that two terms refer to the same entity
  - e.g. `:CarbonDioxide   owl:sameAs   :ARX012345`
- **Different from**: In OWL everything might be potentially identical if we don't explicitly state the difference
  - e.g. `:Valparaíso  owl:differentFrom  :Región de Valparaíso` distinguishes the city from the region of the same name.
- **Negation**: We can state that a relation does not hold using negation, which can be serialised as a graph using a form of **reification**.
  - e.g. `NegativeObjectPropertyAssertion( :hasWife :Bill :Mary )`

Examples are taken from: Harald Sack's Lecture on Knowledge Graphs, https://kgbook.org, and Section 4.4 of the OWL2-Primer (https://www.w3.org/TR/owl2-primer/)

Overview of Ontology Features for Individuals

Feature	Axiom	Condition	Example
ASSERTION	$x \xrightarrow{y} z$	$x \xrightarrow{y} z$	Chile $\xrightarrow{\text{capital}}$ Santiago
NEGATION	 Neg type sub pre obj n x y z	not $x \xrightarrow{y} z$	 Neg type sub pre obj n Chile capital Arica
SAME AS	$x_1 \xrightarrow{\text{same as}} x_2$	$x_1 = x_2$	Región V $\xrightarrow{\text{same as}}$ Región de Valparaíso
DIFFERENT FROM	$x_1 \xrightarrow{\text{diff. from}} x_2$	$x_1 \neq x_2$	Valparaíso $\xrightarrow{\text{diff. from}}$ Región de Valparaíso

Ontology Features for Property Axioms

Ontology Features for Property Axioms

For a **pair of properties**, additional semantics can be defined using OWL (⇝ more infos on the OWL slides)

- specialization and generalizations (see RDFS slides)
- equivalence
- inverse
- disjontness
- transitivity
- symmetric
- asymmetric
- reflexive
- irreflexive

Multiplicity: we can also define the multiplicity of the relation denoted by properties based on being

- functional (ie., many-to-one)
- inverse-functional (ie. one-to-many)

Key: We may further define a key for a class, denoting the set of properties whose values uniquely identify the entities of that class.

Chain: A path expression only allowing concatenation of properties such that pairs of entities related by the chain are also related by the given property.

Overview of ontology features for property axioms (1/3)

Feature	**Axiom**	**Condition** (for all x_*, y_*, z_*)	**Example**
SUB-PROPERTY	$p \xrightarrow{\text{subp. of}} q$	$x \xrightarrow{p} y$ implies $x \xrightarrow{q} y$	venue $\xrightarrow{\text{subp. of}}$ location
DOMAIN	$p \xrightarrow{\text{domain}} c$	$x \xrightarrow{p} y$ implies $x \xrightarrow{\text{type}} c$	venue $\xrightarrow{\text{domain}}$ Event
RANGE	$p \xrightarrow{\text{range}} c$	$x \xrightarrow{p} y$ implies $y \xrightarrow{\text{type}} c$	venue $\xrightarrow{\text{range}}$ Venue
EQUIVALENCE	$p \xrightarrow{\text{equiv. p.}} q$	$x \xrightarrow{p} y$ iff $x \xrightarrow{q} y$	start $\xrightarrow{\text{equiv. p.}}$ begins
INVERSE	$p \xrightarrow{\text{inv. of}} q$	$x \xrightarrow{p} y$ iff $y \xrightarrow{q} x$	venue $\xrightarrow{\text{inv. of}}$ hosts
DISJOINT	$p \xrightarrow{\text{disj. p.}} q$	not $x \underset{p}{\overset{q}{\rightrightarrows}} y$	venue $\xrightarrow{\text{disj. p.}}$ hosts

Overview of ontology features for property axioms (2/3)

TRANSITIVE	$p \xrightarrow{\text{type}} \text{Transitive}$	$x \xrightarrow{p} y \xrightarrow{p} z$ implies $x \xrightarrow{p} z$	$\text{part of} \xrightarrow{\text{type}} \text{Transitive}$
SYMMETRIC	$p \xrightarrow{\text{type}} \text{Symmetric}$	$x \xrightarrow{p} y$ iff $y \xrightarrow{p} x$	$\text{nearby} \xrightarrow{\text{type}} \text{Symmetric}$
ASYMMETRIC	$p \xrightarrow{\text{type}} \text{Asymmetric}$	not $\begin{tikzcd} x \arrow[r, bend left, "p"] & y \arrow[l, bend left, "p"] \end{tikzcd}$	$\text{capital} \xrightarrow{\text{type}} \text{Asymmetric}$
REFLEXIVE	$p \xrightarrow{\text{type}} \text{Reflexive}$	$\begin{tikzcd} x \arrow[loop right, "p"] \end{tikzcd}$	$\text{part of} \xrightarrow{\text{type}} \text{Reflexive}$
IRREFLEXIVE	$p \xrightarrow{\text{type}} \text{Irreflexive}$	not $\begin{tikzcd} x \arrow[loop right, "p"] \end{tikzcd}$	$\text{flight} \xrightarrow{\text{type}} \text{Irreflexive}$

Overview of ontology features for property axioms (3/3)

FUNCTIONAL	$p \xrightarrow{\text{type}} \text{Functional}$	$y_1 \xleftarrow{p} x \xrightarrow{p} y_2$ implies $y_1 = y_2$	population $\xrightarrow{\text{type}}$ Functional
INV. FUNCTIONAL	$p \xrightarrow{\text{type}} \text{Inv. Functional}$	$x_1 \xrightarrow{p} y \xleftarrow{p} x_2$ implies $x_1 = x_2$	capital $\xrightarrow{\text{type}}$ Inv. Functional
KEY	$c \xrightarrow{\text{key}} \boxed{\begin{matrix} p_1 \\ \vdots \\ p_n \end{matrix}}$	$$\begin{tikzcd} & c & \\ & y_1 & \\ x_1 \arrow[uur, "\text{type}"] \arrow[ur, "p_1"] \arrow[r, "\cdots"] \arrow[dr, "p_n"'] & \cdots & x_2 \arrow[uul, "\text{type}"'] \arrow[ul, "p_1"'] \arrow[l, "\cdots"'] \arrow[dl, "p_n"] \\ & y_n & \end{tikzcd}$$ implies $x_1 = x_2$	City $\xrightarrow{\text{key}}$ $\boxed{\begin{matrix}\text{lat} \\ \text{long}\end{matrix}}$
CHAIN	$p \xrightarrow{\text{chain}} \boxed{\begin{matrix} q_1 \\ \vdots \\ q_n \end{matrix}}$	$x \xrightarrow{q_1} y_1 \cdots \rightarrow y_{n-1} \xrightarrow{q_n} z$ implies $x \xrightarrow{p} z$	location $\xrightarrow{\text{chain}}$ $\boxed{\begin{matrix}\text{location} \\ \text{part of}\end{matrix}}$

Ontology Features for Class Axioms

Ontology Features for Class Axioms

A **pair of classes** can be defined as

- being in a super-/subclass-relation
- equivalent
- disjont

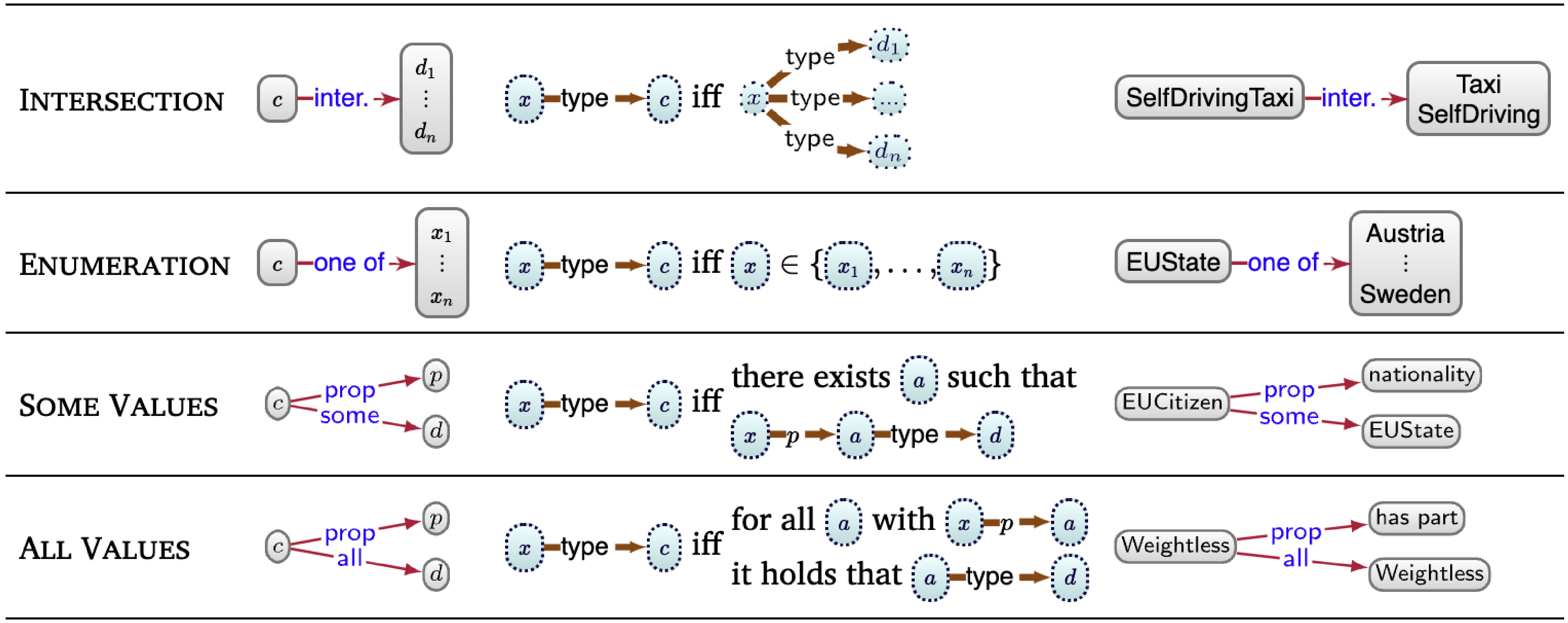
Novel classes can be defined by...

- **I.) ...applying set operators to other classes**
  - one can define a novel class as the complement of another class
  - the union or intersection of a list (of arbitrary length) of other classes
  - an enumeration of all of its instances
- **II.) ...based on conditions / restrictions that the properties of its instances satisfy**
  - by placing restrictions on a particular property p, one can define classes whose instances are all of the entities that have:
    - some value from a given class on p
    - all values from a given class on p
    - have a specific individual as a value on p (has value)
    - have themselves as a reflexive value on p (has self)
    - have at least / at most / or exactly some number of values on p (cardinality)
    - have at least / at most / or exactly some number of values on p from a given class (qualified cardinality)

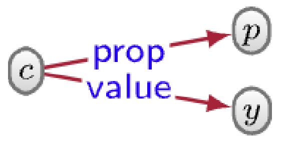
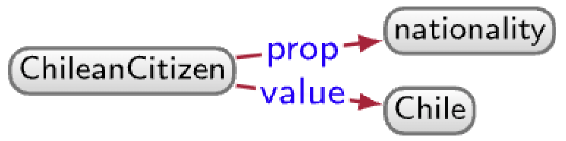
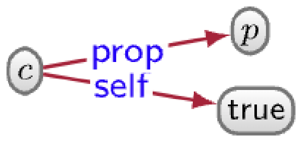
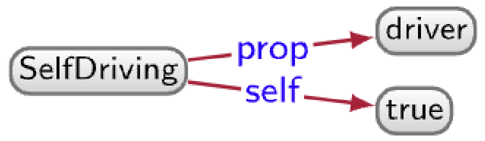
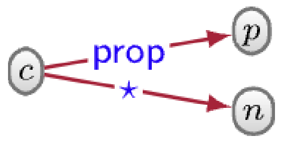
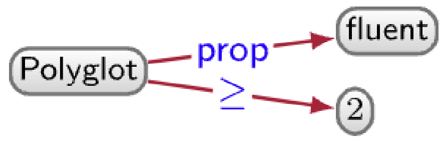
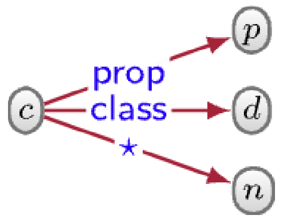
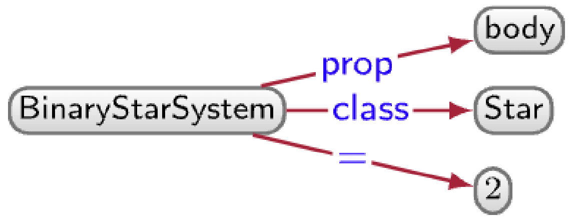
Overview of ontology features for class axioms (1/3)

Feature	**Axiom**	**Condition** (for all x_*, y_*, z_*)	**Example**
SUB-CLASS	$c \xrightarrow{\text{subc. of}} d$	$x \xrightarrow{\text{type}} c$ implies $x \xrightarrow{\text{type}} d$	City $\xrightarrow{\text{subc. of}}$ Place
EQUIVALENCE	$c \xrightarrow{\text{equiv. c.}} d$	$x \xrightarrow{\text{type}} c$ iff $x \xrightarrow{\text{type}} d$	Human $\xrightarrow{\text{equiv. of}}$ Person
DISJOINT	$c \xrightarrow{\text{disj. c.}} d$	not $c \xleftarrow{\text{type}} x \xrightarrow{\text{type}} d$	City $\xrightarrow{\text{disj. c.}}$ Region
COMPLEMENT	$c \xrightarrow{\text{comp.}} d$	$x \xrightarrow{\text{type}} c$ iff not $x \xrightarrow{\text{type}} d$	Dead $\xrightarrow{\text{comp.}}$ Alive
UNION	$c \xrightarrow{\text{union}} \boxed{d_1 \; \vdots \; d_n}$	$x \xrightarrow{\text{type}} c$ iff $x \xrightarrow{\text{type}} d_1$ or $x \xrightarrow{\text{type}} \ldots$ or $x \xrightarrow{\text{type}} d_n$	Flight $\xrightarrow{\text{union}}$ DomesticFlight InternationalFlight

Overview of ontology features for class axioms (2/3)


INTERSECTION
c inter. d1 ⋮ dn
x type c iff x type d1, type …, type dn
SelfDrivingTaxi inter. Taxi SelfDriving
ENUMERATION
c one of x1 ⋮ xn
x type c iff x ∈ {x1, . . . , xn}
EUState one of Austria ⋮ Sweden
SOME VALUES
c prop p some d
x type c iff there exists a such that x p a type d
EUCitizen prop nationality some EUState
ALL VALUES
c prop p all d
x type c iff for all a with x p a it holds that a type d
Weightless prop has part all Weightless

Overview of ontology features for class axioms (3/3)

HAS VALUE		$x \xrightarrow{\text{type}} c$ iff $x \xrightarrow{p} y$	
HAS SELF		$x \xrightarrow{\text{type}} c$ iff $x \xrightarrow{p} x$	
CARDINALITY $\star \in \{=, \leq, \geq\}$		$x \xrightarrow{\text{type}} c$ iff $\#\{a \mid x \xrightarrow{p} a\} \star n$	
QUALIFIED CARDINALITY $\star \in \{=, \leq, \geq\}$		$x \xrightarrow{\text{type}} c$ iff $\#\{a \mid x \xrightarrow{p} a \xrightarrow{\text{type}} d\} \star n$	

In the next lecture, we will talk about how to create ontologies using OWL and Protégé