KG theories

Reasoning about Knowledge Graphs

(HP2T)

KG theories

- **LoD theories**
- Unfolding LoD theories
- ASK – Reasoning about LoD theories
- KG theories
  - Lexicons – ontologies and language teleontologies
  - EER models – knowledge teleontologies
  - ER models, schemas – teleologies (ETGs)
- Unfolding KG theories
- ASK – Reasoning about KG theories
- Key notions

TBox – A LoD Theory

Definition (TBox). A LoD TBox is an assertional theory consisting of a set of LoD **descriptions**.

Observation (TBox). "T" in "TBox" stands for "Term" (LoD assertions correlate the extension of the terms describing the percepts).

Reminder (LoD Description). A LoD description is a set of constraints on the domain structure. A LoD definition is a LoD description which introduces a new etype as a subset of the domain.

Observation (Using TBoxes). TBoxes are used in DB and KR applications to make explicit the knowledge implicit in DBs and EGs.

(Strongly) definitional TBox

Definition (Definitional TBox) A LoD TBox is a finite set of LoD (**etype) definitions.**

Definition (Strongly definitional TBox) A LoD TBox is a finite set of LoD (**etype) equivalences.**

Observation 4: Our focus is mainly on definitional TBoxes, which are key in AI in the modeling how human language and knowledge structure the world.

Strongly definitional TBOX (example)

Family relations

- Person $\equiv$ $\exists$hasname.String $\sqcap$ $\forall$HasJob.Organization
- Woman $\equiv$ Person $\sqcap$ Female
- Man $\equiv$ Person $\sqcap$ $\neg$ Woman
- Mother $\equiv$ Woman $\sqcap$ $\exists$hasChild.Person
- Father $\equiv$ Man $\sqcap$ $\exists$hasChild.Person
- Parent $\equiv$ Father $\sqcup$ Mother

LoD Terminology

Definition (Uses). Let T be definitional TBox. Let $E \sqsubseteq p$ or $E \equiv p$ be a definition in T. Then we say that E **directly uses** E’, where E’ is an atomic etype, if E’ occurs in *p.* We say that E **uses** E’ if E’ occurs in the right hand side of a definition of an etype mentioned in *p*, and so on recursively.

Observation (Uses). **“**Uses” is defined as the transitive closure of directly uses.

Observation (Acyclic definitional TBox). A definitional TBox is **acyclic** if

- There is no type that uses itself , and
- There are no two definitions of the same etype

Observation 1 (acyclic TBox). The second requirement avoids any type using itself.

Observation 2 (acyclic TBox). An acyclic Tbox avoids the following situation:

$$E_1 \sqsubseteq \ldots E_2 \ldots,\quad E_2 \sqsubseteq \ldots E_3 \ldots,\quad \ldots\quad,\ E_n \sqsubseteq \ldots E_1 \ldots$$

Definition (Terminology). A **Terminology** is an acyclic definitional TBox.

Observation (Terminology). Terminologies are key in the construction of human lexicons and knowledge.

Terminology (negative example)

Family relations

- $Person \equiv Man \sqcup Woman$
- $Woman \equiv Person \sqcap Female$
- $Man \equiv Person \sqcap \neg Woman$
- $Mother \equiv Woman \sqcap \exists hasChild.Person$
- $Father \equiv Man \sqcap \exists hasChild.Person$
- $Parent \equiv Father \sqcup Mother$

Terminology (example)

Family relations

- Person ≡ ∃hasname.String ⊓ ∀HasJob.Organization
- Woman ≡ Person ⊓ Female
- Man ≡ Person ⊓ ¬ Woman
- Mother ≡ Woman ⊓ ∃hasChild.Person
- Father ≡ Man ⊓ ∃hasChild.Person
- Parent ≡ Father ⊔ Mother

Terminology (better – but negative example)

Family relations

- Person ≡ ∃hasname.String ⊓ ∀HasJob.Organization
- Parent ≡ Person ⊓ ∃hasChild.Person
- Mother ≡ Parent ⊓ Female
- Father ≡ Parent ⊓ Male
- Woman ≡ Person ⊓ Female
- Man ≡ Person ⊓ Male
- Mother ⊑ Woman
- Father ⊑ Man

Terminology (even better – still negative example)

Family relations

- Person ≡ $\exists$hasname.String $\sqcap$ $\forall$HasJob.Organization
- Female $\perp$ Male
- Woman ≡ Person $\sqcap$ Female
- Man ≡ Person $\sqcap$ Male
- Parent ≡ Person $\sqcap$ $\exists$hasChild.Person
- Mother $\sqsubseteq$ Parent
- Father $\sqsubseteq$ Parent
- Mother $\sqsubseteq$ Woman
- Father $\sqsubseteq$ Man

KG theories

- LoD theories
- **Unfolding LoD theories**
- ASK – Reasoning about LoD theories
- KG theories
  - Lexicons – ontologies and language teleontologies
  - EER models – knowledge teleontologies
  - ER models, schemas – teleologies (ETGs)
- Unfolding KG theories
- ASK – Reasoning about KG theories
- Key notions

Unfolding an etype

Definition (Definiendum, definiens): The left hand side of a definition A ≡ C (that is, A) is called **definiendum**, the right hand side (that is, C) is called the definiens.

Definition (Defined and primitive etype): Given a TBox, a **defined** etype is an etype which appears on the left of a definition of the TBox. A **primitive** etype is an etype which only appears on the right of the definitions. A primitive type is an **atomic** etype. A defined etype is a **complex** etype.

Definition (etype unfolding) A defined etype is **unfolded** if all the defined etypes occurring in its definiens are recursively substituted with their definition.

Observation (unfolded etype): The definition of an unfolded etype contains only primitive etypes

Example. From:

ElectricGuitar ≡ Guitar ⊓ SoundAmplification

ElectricGuitar#1 ≡ ElectricGuitar ⊓ ∃hasColour.String ⊓ ∃hasBrand.String

… to:

ElectricGuitar#1 ≡ Guitar ⊓ SoundAmplification ⊓ ∃hasColour.String ⊓ ∃hasBrand.String

Remark: In an acyclic terminology the process of etype unfolding is applied recursively up to any level, usually with the goal of reducing to primitive etypes. The process is guaranteed to terminate.

Unfolding a TBox

Observation (Defined and primitive etype): We restrict to the case where a defined etype appears once on the left or a definition. Defined and primitive etypes can appear on the right of definitions any number of times.

Definition (TBox unfolding). A definitional TBox T can be unfolded into a Tbox T’ by (recursively) unfolding all its defined etypes.

Theorem: Let T be a terminology . Let T’ the result of unfolding T. Then M is a model of T if and only if it is a model of T’.

Complexity of TBox unfolding

Observation (TBox unfolding). TBox definitions are like macros that can be unfolded into primitive etypes.

Observation (Complexity of unfolding). The size of the unfolded TBox grows **exponentially** with the depth of the TBox induced subsumption hierarchy. For instance, from

$$A0 \equiv \exists r.A1 \sqcap \forall s A1$$

$$A1 \equiv \exists r.A2 \sqcap \forall s.A2$$

$$A2 \equiv \exists r.A3 \sqcap \forall s.A3$$

... we obtain

$A1 \equiv \exists r.(\exists r.A2 \sqcap \forall r.A2) \sqcap \forall s.(\exists r.A2 \sqcap \forall r.A2)$ (A1 *2 times* A0)

$A2 \equiv \exists r.(\exists r.(\exists r.A3 \sqcap \forall s.A3) \sqcap \forall s.(\exists r.A3 \sqcap \forall s.A3)) \sqcap \forall s .(\exists r.(\exists r.A3 \sqcap \forall s.A3) \sqcap \forall s.(\exists r.A3 \sqcap \forall s.A3))$ (A2 *4 times* A0)

Observation (Complexity of unfolding). Definitions like the above are nested definitions, where the meaning of a defined type depends entirely on the other. Does not apply to our target applications (KG theories).

Unfolding a TBox (example 1 – reprise)

- ~~Person ≡ ∃hasname.String ⊓ ∀HasJob.Organization~~
- Woman ≡ Person ⊓ Female
- Man ≡ Person ⊓ ¬ Woman
- Mother ≡ Woman ⊓ ∃hasChild.Person
- Father ≡ Man ⊓ ∃hasChild.Person
- Parent ≡ Father ⊔ Mother

Observation. The deletion of the first definition makes Person a primitive etype. The choice of the primitive etype is up to the modeler

Observation. The definition of Man is not minimalistic (Woman instead of Female)

Observation. The definition of Parent is redundant (IsParent ≡ HasChild)

Unfolding a TBox (example 1 – continued)

$$\mathcal{T}' = \begin{cases} Woman \equiv Person \sqcap Female \\ Man \equiv Person \sqcap \neg(Person \sqcap Female) \\ Mother \equiv (Person \sqcap Female) \sqcap \exists hasChild.Person \\ Father \equiv (Person \sqcap \neg(Person \sqcap Female)) \sqcap \exists hasChild.Person \\ Parent \equiv (Person \sqcap \neg(Person \sqcap Female)) \sqcap \\ \qquad \exists hasChild.Person \sqcup (Person \sqcap Female) \sqcap \exists hasChild.Person \end{cases}$$

Observation. Unfolding generates disjunctions. In fact (check the Venn Diagram)

$$\neg (A \sqcap B) \equiv \neg A \sqcup \neg B$$

Observation. Disjunction generates decision branches, that is, it increases the complexity of reasoning.

KG theories

- LoD theories
- Unfolding LoD theories
- **ASK – Reasoning about LoD theories**
- KG theories
  - Lexicons – ontologies and language teleontologies
  - EER models – knowledge teleontologies
  - ER models, schemas – teleologies (ETGs)
- Unfolding KG theories
- ASK – Reasoning about KG theories
- Key notions

LoD – Reasoning problems (reminder)

Observation (LoD reasoning problems). The four LoD core reasoning problems are:

- $T \models C$, **Satisfiability with respect to a TBox T**
- $T \models C \sqsubseteq D$, **Subsumption with respect to a TBox T**
- $T \models C \equiv D$, **Equivalence with respect to a TBox T**
- $T \models C \perp D$, **Disjointness with respect to a TBox T**

Lod – Reasoning problems (reminder)

Proposition (Reduction to satisfiability). All the problems reduce to satisfiability. In fact, we have the following equivalences:

- **Equivalence**: $C \equiv_T D$ if and only if $C \sqsubseteq_T D$ and $D \sqsubseteq_T C$;
- **Subsumption**: $C \sqsubseteq_T D$ if and only if $C \sqcap \neg D$ is *unsatisfiable* with respect to T*;*
- **Disjointness:** $C \perp_T D$ if and only if $C \sqcap D$ is *unsatisfiable* with respect to T.

TBox Reasoning by Unfolding

Intuition (LoD satisfiability). LoD Satisfiability can be implemented via unfolding. The algorithm proceeds by unfolding defined etypes one at the time as follows. If:

- a disjunct is found by expanding a definition: split the formula in two and proceed in both branches;
- the formula in a branch is a conjunct of two etypes which are disjoint or, equivalently, a conjunct is ⊥, the formula being built is unsatisfiable. This attempt is given up;
- the formula in a branch is not unfoldable, that is, it contains only primitive etypes: the original formula is satisfiable;
- all the possible branches reduce to an unsatisfiable formula: the original formula is unsatisfiable.

Intuition (Complexity of LoD satisfiability). LoD satisfiability is NP-complete, that is, in the worst case it takes exponential time (there is a need to explore all the paths generated by disjunctions). This is the case independently of whether unfolding does not generate exponentially long formulas (see above).

Intuition (Decision methods for LoD satisfiability). LoD satisfiability can be reduced to LoP (propositional) satisfiability (theorem and algorithm will be provided inside the section on LoP).

Intuition (Tableau method). Tableaux are a complete and elegant method for deciding LoD (and therefore LoP) satisfiability. The state of the art tableaux reasoning algorithms are largely inefficient if compared to the state of the art LoP satisfiability algorithms. The vast majority of the CS and AI reasoning applications use LoP satisfiability.

Intuition (LoD satisfiability, KG theories). If and when needed, our approach is to solve LoD satisfiability as LoP satisfiability. We use LoD satisfiability only for KG theories (see below). In KG theories, LoD satisfiability is solvable in polynomial time, still allowing us to enrich the expressiveness of LoE EGs (by making explicit the meaning of natural language terms/ words).

Unfolding a TBox (example 1 – continued)

$\text{Woman} \equiv \text{Person} \sqcap \text{Female}$

$\text{Man} \equiv \text{Person} \sqcap \neg \text{Woman}$

$\text{Mother} \equiv \text{Woman} \sqcap \exists \text{hasChild.Person}$

$\text{Father} \equiv \text{Man} \sqcap \exists \text{hasChild.Person}$

$\text{Parent} \equiv \text{Father} \sqcup \text{Mother}$

Two disjunctions, which generate two splits

$$\begin{aligned}
\text{Father} &\equiv \text{Person} \sqcap \neg (\text{Person} \sqcap \text{Female}) \sqcap \exists \text{hasChild.Person} \\
&\equiv \text{Person} \sqcap (\neg \text{Person} \sqcup \neg \text{Female}) \sqcap \exists \text{hasChild.Person} \\
&\equiv (\text{Person} \sqcup \neg \text{Person}) \sqcap (\text{Person} \sqcup \neg \text{Female}) \sqcap \exists \text{hasChild.Person} \\
&\equiv (\text{Person} \sqcup \neg \text{Female}) \sqcap \exists \text{hasChild.Person} \\
&\equiv (\text{Person} \sqcap \exists \text{hasChild.Person}) \sqcup (\neg \text{Female} \sqcap \exists \text{hasChild.Person})
\end{aligned}$$

KG theories

- LoD theories
- Unfolding LoD theories
- ASK – Reasoning about LoD theories
- **KG theories**
  - Lexicons – ontologies and language teleontologies
  - EER models – knowledge teleontologies
  - ER models, schemas – teleologies (ETGs)
- Unfolding KG theories
- ASK – Reasoning about KG theories
- Key notions

KG theories

Observation (LoD theories, KG theories). If one checks the literature, one will find lots of different **LoD theories**, used in different fields and, in particular, in the work on DB and KR. In this course we focus on the LoD theories of relevance to AI and in particular on the work of Knowledge Graphs (KGs) and Large Language Models (LLMs). We call them **KG theories**.

Observation (KG theories). We focus on the following modeling problems:

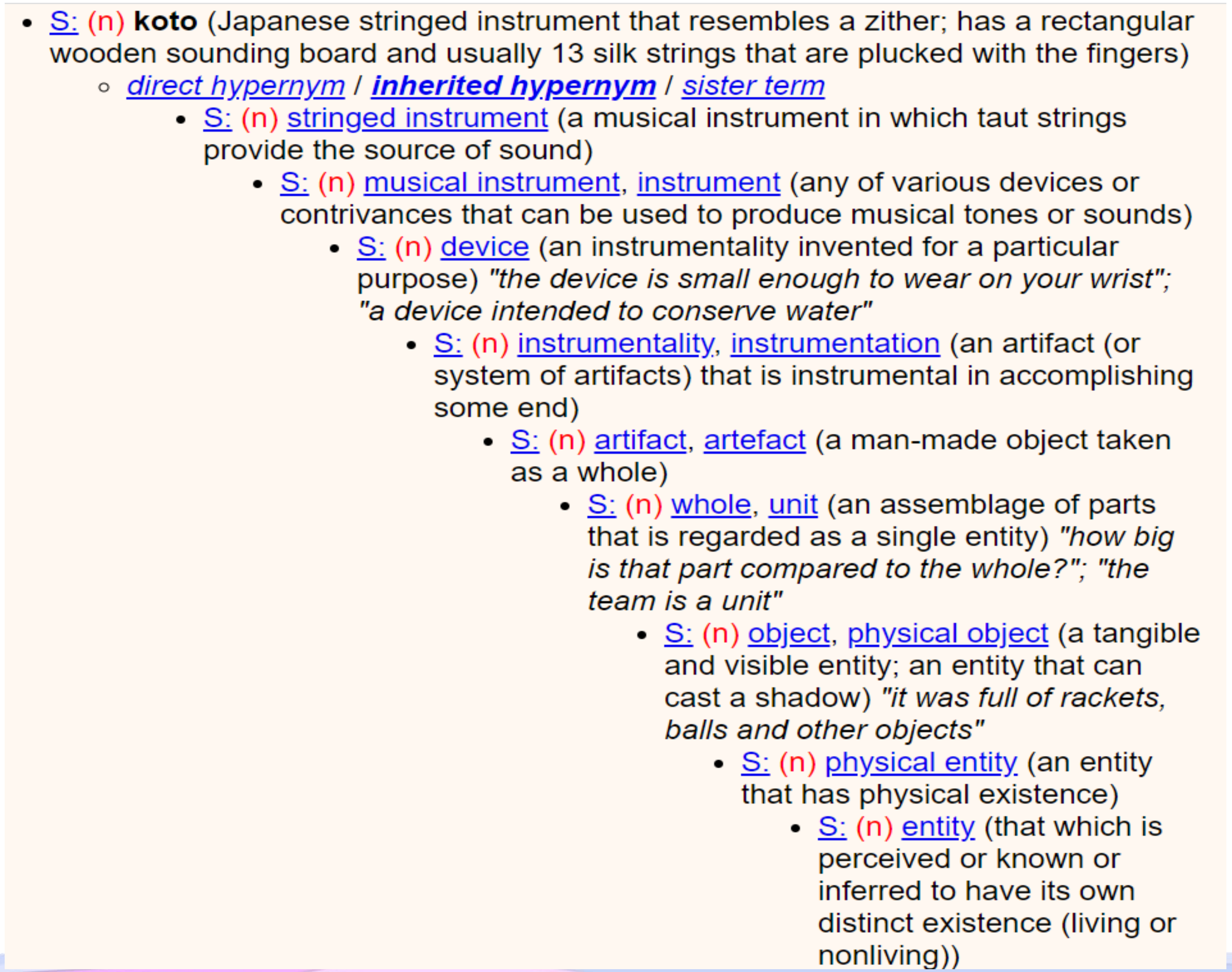
- **Ontologies.** These KG theories formalize digital lexicons (i.e., WordNet, UKC, and similar multi-lingual resources), that is the meaning of words, as used in informal and semi-formal world models.
- **Language teleontologies (LTLO).** These KG theories formalize the so called "domain lexicons/ languages" and also "application lexicons/ languages", that is targeted lexicons which cover the alphabet of specific domains (e.g., Health, Web, Digital libraries) and applications.
- **Knowledge teleontologies (KTLO).** These KG theories formalize Extended ER (EER) models, that is the hierarchical correlations, the inheritance of properties, of etypes.
- **Teleologies (TLO),** also called **etype Graphs (ETGs).** These KG theories formalize KG and DB schemas, and ER models, that is the "knowledge" used to organize data about entities.

Observation (On the relevance of KG theories). The integration of LoE with the information provided by the KG theories described above allows to deal with the low expressiveness of LoE. See the **LoDe** logic.

KG theories

- LoD theories
- Unfolding LoD theories
- ASK – Reasoning about LoD theories
- KG theories
  - **Lexicons – ontologies and language teleontologies**
  - EER models – knowledge teleontologies
  - ER models, schemas – teleologies (ETGs)
- Unfolding KG theories
- ASK – Reasoning with KG theories
- Key notions

Natural Language Lexicon – example



- S: (n) **koto** (Japanese stringed instrument that resembles a zither; has a rectangular wooden sounding board and usually 13 silk strings that are plucked with the fingers)
  - *direct hypernym* / ***inherited hypernym*** / *sister term*
    - S: (n) stringed instrument (a musical instrument in which taut strings provide the source of sound)
      - S: (n) musical instrument, instrument (any of various devices or contrivances that can be used to produce musical tones or sounds)
        - S: (n) device (an instrumentality invented for a particular purpose) *"the device is small enough to wear on your wrist"; "a device intended to conserve water"*
          - S: (n) instrumentality, instrumentation (an artifact (or system of artifacts) that is instrumental in accomplishing some end)
            - S: (n) artifact, artefact (a man-made object taken as a whole)
              - S: (n) whole, unit (an assemblage of parts that is regarded as a single entity) *"how big is that part compared to the whole?"; "the team is a unit"*
                - S: (n) object, physical object (a tangible and visible entity; an entity that can cast a shadow) *"it was full of rackets, balls and other objects"*
                  - S: (n) physical entity (an entity that has physical existence)
                    - S: (n) entity (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))

On the left hand side is the WordNet lexical hierarchy generalizing the concept for Koto. See Princeton WordNet.

S(n) indicates a **synset** associated to a word (here Koto) (one of the possibly many) of synonymous nouns. Here Koto has no synonyms.

Hyponym / **Hypernym** stand for subclass and superclass relationship .

Each synset is described by a **gloss** (a definition of the meaning of a word, most of the time incomplete, provided informally) and an **example** (between quotes in the figure on the left).

Natural language Lexicon

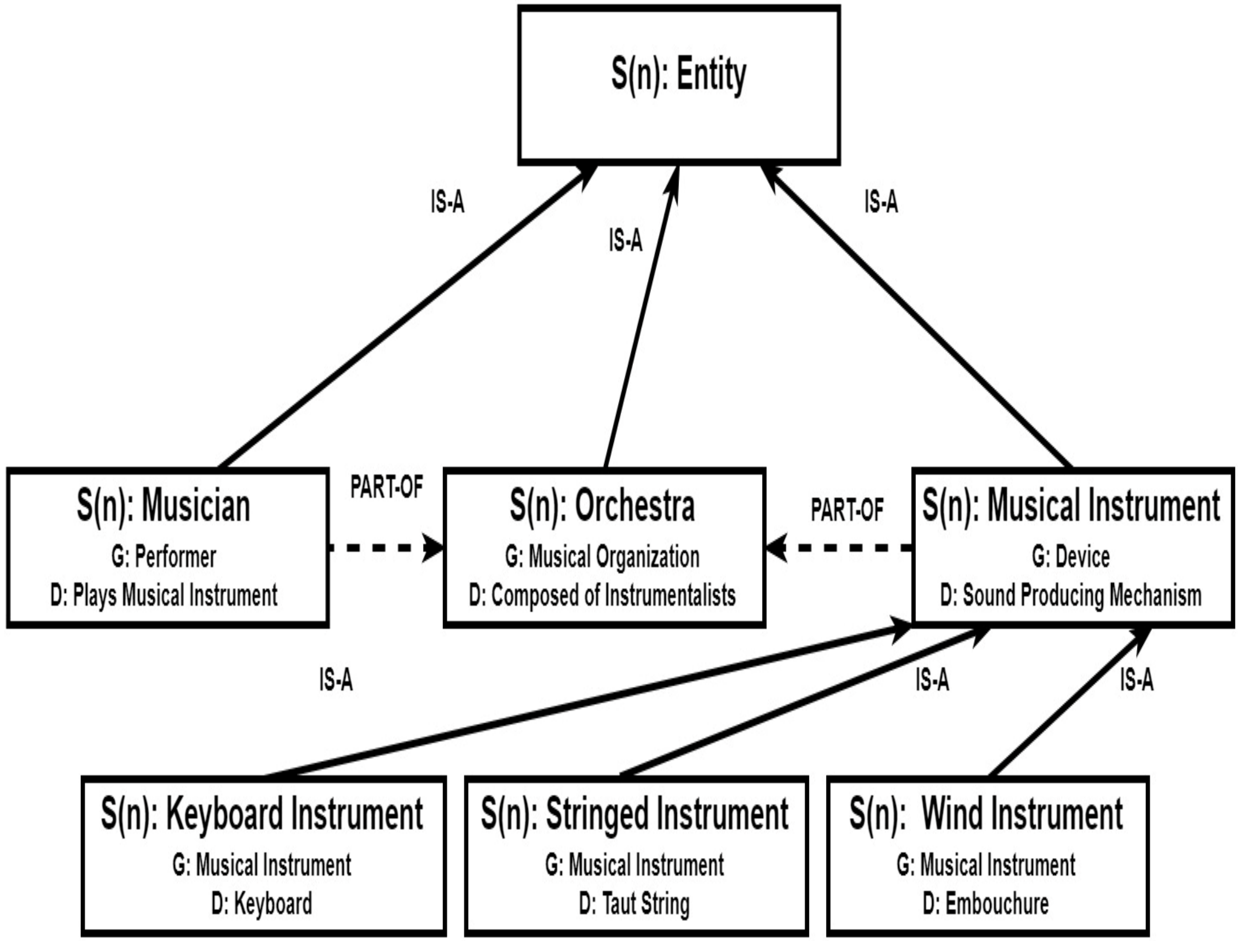
Lexicons. Lexicons are lists of words which encode the meaning of the **words** of one or more natural languages. Lexicons adhere to **high quality principles**. They are usually developed or validated by humans

Words, as they occur in languages, are the main elements of lexicons.

Senses. A sense is one of the multiple **meanings** denoted by a polysemous word (e.g., *car* stands for *automobile* and *railway car*). The word **sense** can be taken as synonymous of the word **concept** (see above).

Synsets. Sets of **synonyms**, that is, synonymous words associated to the same sense.

Unique identifiers (not visible to users) named *GID*, e.g., *588967*, are the names denoting concepts (via the data type concept).



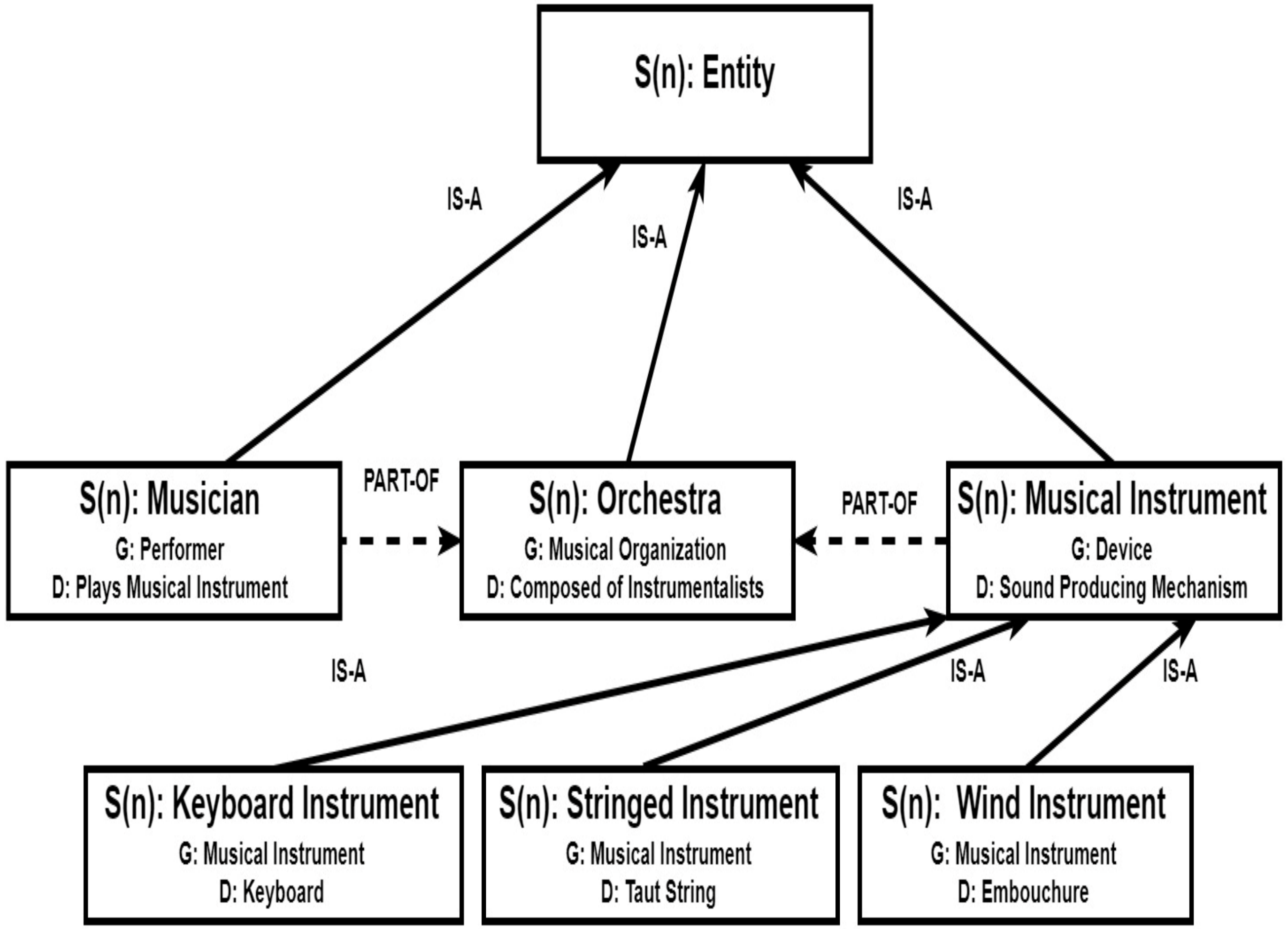
Natural language Lexicon (continued)

A lexicon is a semi-formal **IS-A hierarchy** which semantically models superclass - subclass (hyponym / hypernym) relations between senses.

Genus-Differentia is the main means by which the IS-A hiearchy are built

Genus. The set of properties which define the scope of a sense, e.g., *musical instrument (G)* for *stringed instrument.* Any concept in a hierarchy has, as genus, the concept of the single parent node.

Differentia. The set of properties which qualify / differentiate senses with the same genus, e.g., *Taut String (D)* for *Stringed Instrument.* All the siblings of the same concept, while having the same genus have different differentias. All the differentias are **disjoint**, this guarantees the no ambiguity (of the concepts associated to polysemous) words,



Domain Lexicon

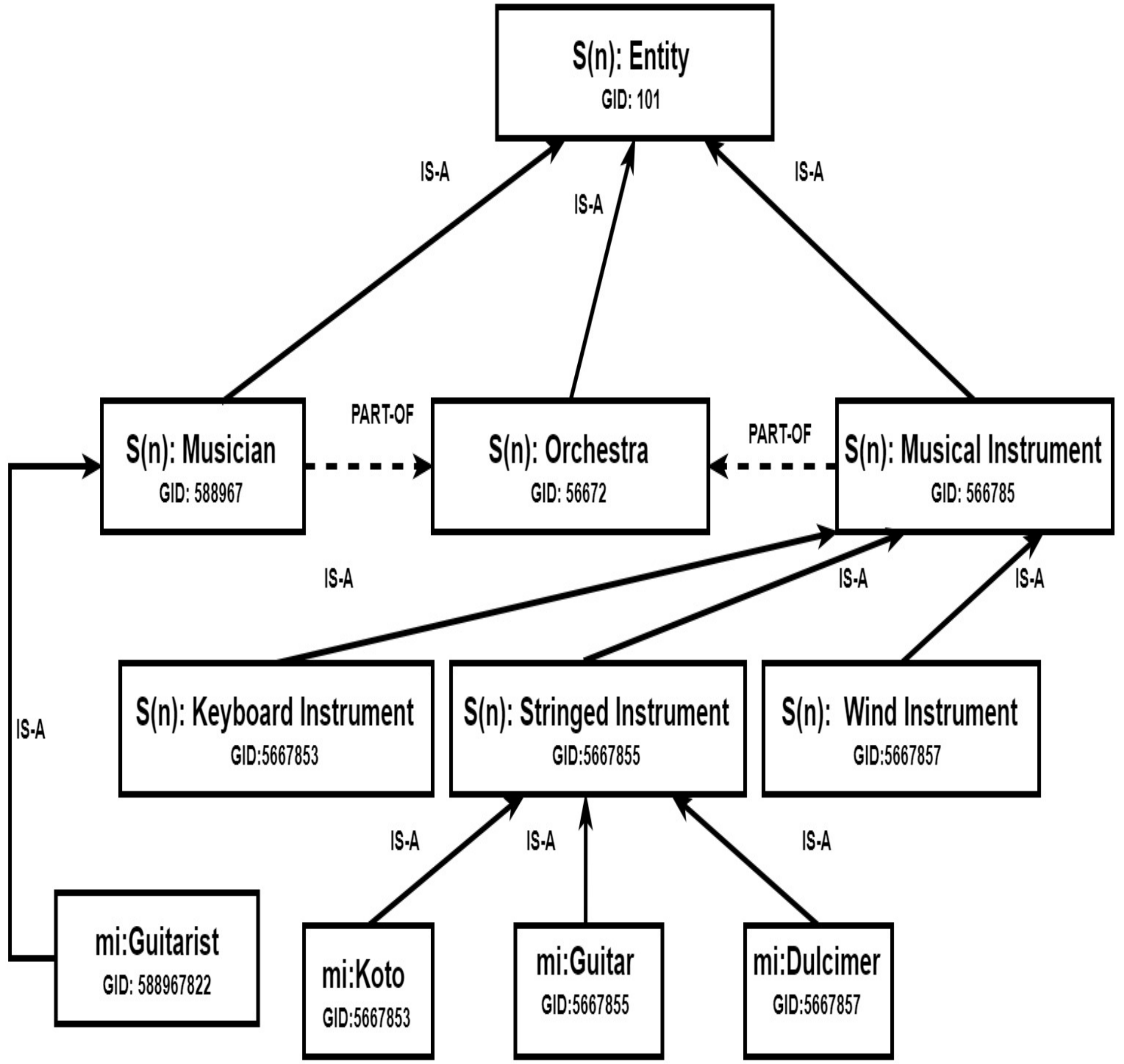
Domain Lexicons (e.g., **Standards**, or **name spaces**) are lexicons which encode the semantics of domain language(s) which focus on specific application domains (e.g., health, tourism).

They usually extend natural language lexicons with **domain-specific words** and **concepts**.

Words in domain languages follow the same rules as those in natural languages with one exception: **they are NOT polysemous**, i.e., they have only one sense.

Each word has a **prefix**, e.g., *mi*: (e.g., for musical instruments) to indicate the domain language/ name space to which the words is associated.

As with lexicons, each sense is identified via a <u>**unique identifier**</u> named *GID*, e.g., *5667853* for *mi:Koto,* denoting the single sense of that word.



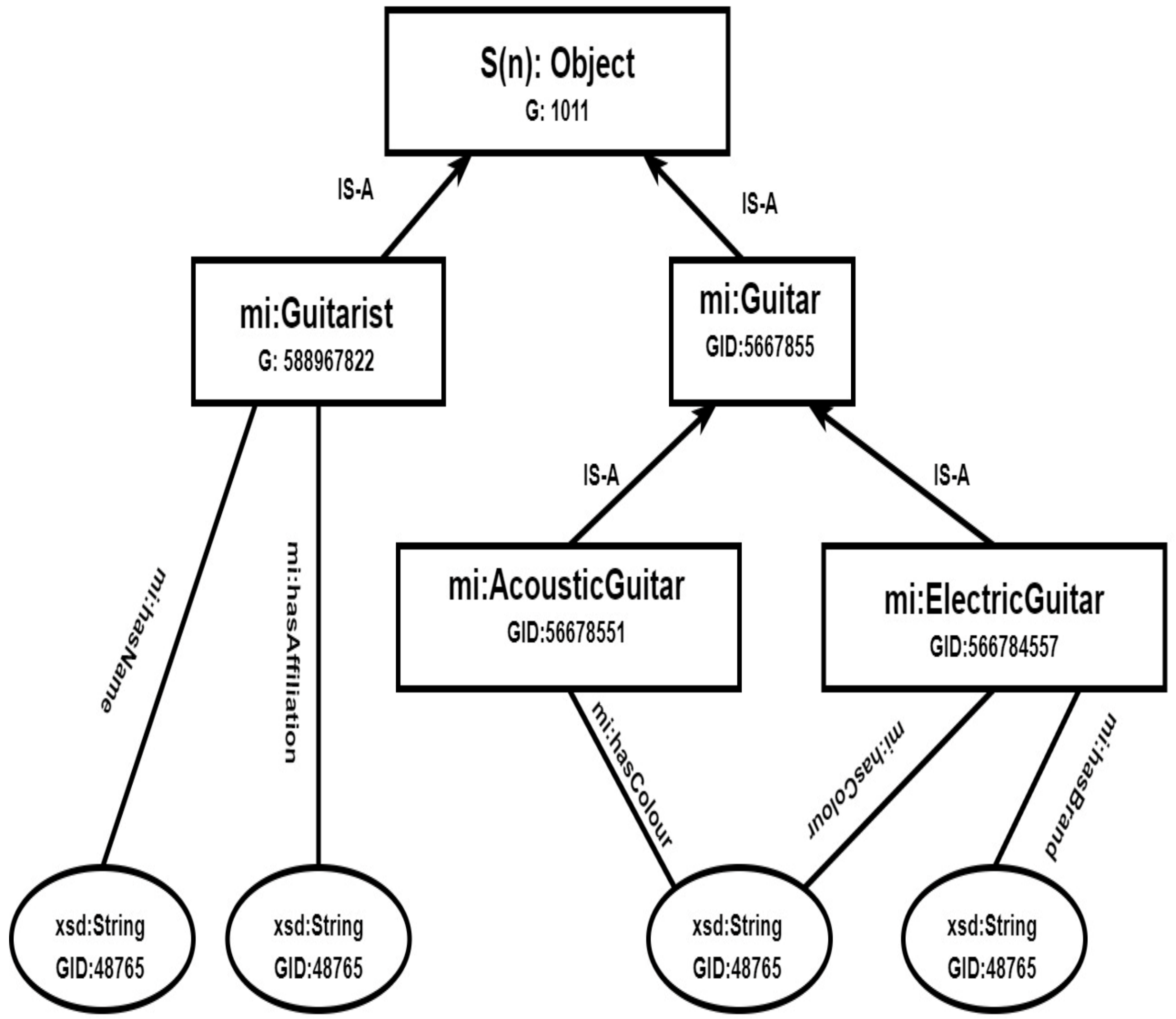
Application Lexicon

Application lexicons are specialized lexicons which focus on a purpose-specific (application dependent) part of a lexicon. They are usually built by selecting concepst from domain (or natural language) lexicons.

Application lexicons are obtained from lexicons by:

- Identifying the root concept, as from the specific need. This is a whole defining the reference space and time containment. In the example above: orchestra
- Name the root concept as «object» (or «thing», or «entity» or anything more specific, e.g., «musical instrument», to define the scope of the concepts which are selected
- Keeping the concepts of all the objects relevant to the application
- Keeping the relevant concepts which specialize, via the IS-A hierarchy, the concepts from previous step
- Dropping irrelevant concepts (below/ above the whole)

Observation (Application lexicons). Application lexicons are the lexicons used in ER/ EER models, and ETGs.



Ontology – a formalized lexicon

Definition (Genus-differentia definition, LoD definition). A genus-differentia definition is as follows

$$\text{Label} \equiv \text{Genus} \sqcap \text{Differentia}$$

where:

- There is a root primitive etype, also called the Root genus
- A genus is an etype (label in the definition above) defined by a genus-differentia definition, starting from the root genus.
- A differentia etype is an primitive etype never occurred before (above) in the hierarchy;
- For all siblings i, j of the same genus,

$$\text{Differentia}_i \perp \text{Differentia}_j$$

A Genus-differentia definition is a **LoD definition.**

Example (Definition of musical instruments). As from the example on musical instruments:

- KeyboardInstr ≡ MusicalInstr ⊓ Keyboard
- KeyboardInstr ⊥ StringedInstr
- KeyboardInstr ⊥ WindInstr
- *… the same for the other siblings*

Observation (Genus-differentia definition). The condition on differentia etypes has the goal of grounding language into analogical representations (assertions into the domain of interpretation).

Ontology – a formalized lexicon

Definition (Ontology, LoD concept). An ontology is a terminology formalizing tree of nodes, each link associated with a genus-differentia definition, where:

- There is only on root genus;
- each node label but the root (genus) is defined with a genus-differentia definition;
- each node label but the root (genus) is defined only once (as from the definition of Tbox).

All labels in an ontology are **(LoD) concepts**.

Definition (Language teleontology). A language teleontology is any terminology which is a subtree of an ontology.

Observation (Ontology, teleontology). Ontologies formalize NL and domain lexicons. Language teleontologies formalize application lexicons.

Observation (Ontology). The notion of ontology used here is restricted in two dimensions:

- (Informal notion) In the literature the notion of ontology has been used in a quite liberal way, and it covers essentially all the categories used here. The terminology introduced here has the goal to identify clearly the different ways in which terminologies can be used to disambiguate KGs.
- (Formal notion) The notion of ontology provided here is quite restricted. More general definitions can be provided which still fit the requirement of a clear mapping to the domain of interpretation.

Observation (Lexical resources). The definitions above (of genus-differentia and ontology) must be understood under the assumption of unique names (e.g., the WordNet concept ids), rather than (ambiguous natural language) words.

Ontology – a formalized lexicon

Observation (Concept). A concept appears:

- once in the left of a definition
- only on the right of all its children (compare with previous example of woman and female)

Observation (Djsjointness relation). This requirement avoids ambiguity between any two LoD concepts. This is enforced by the disjointness of the siblings' differentiae.

Observation (Subsumption relation). The subsumption relation holds between a label and its genus. For instance

$$\text{KeyboardInstr} \sqsubseteq \text{MusicalInstr}$$

All labels are subsumed by the root. That is, the root defines the domain of interpretation.

Observation (LOD Terminology): Any ontology or language teleontology is a terminology.

Ontology – Protégé

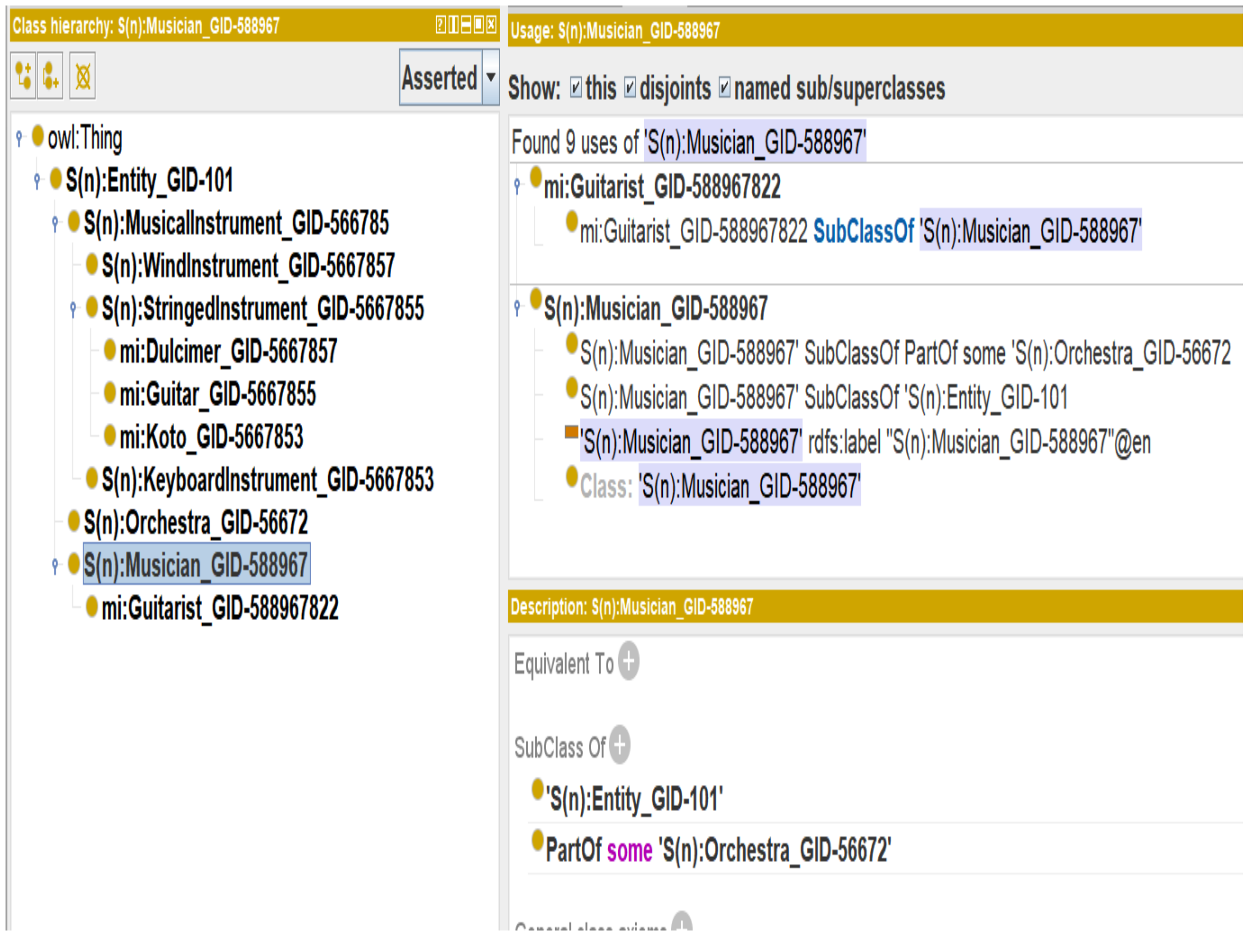
The snippet on the right side shows the domain lexicon formalized via the Protégé ontology editor.

You can see the entire class hierarchy starting from owl:Thing downwards depicting the concepts with their unique GIDs.

Notice mi:Koto, mi:Guitar etc, belong to domain lexicon and not natural language lexicon.

You can also see (partial) visualization of LOD formalization of the example domain lexicon, e.g.,

Musician is - PartOf - (some) Orchestra



Class hierarchy: S(n):Musician_GID-588967

Asserted

- owl:Thing
  - S(n):Entity_GID-101
    - S(n):MusicalInstrument_GID-566785
      - S(n):WindInstrument_GID-5667857
      - S(n):StringedInstrument_GID-5667855
        - mi:Dulcimer_GID-5667857
        - mi:Guitar_GID-5667855
        - mi:Koto_GID-5667853
      - S(n):KeyboardInstrument_GID-5667853
    - S(n):Orchestra_GID-56672
    - S(n):Musician_GID-588967
      - mi:Guitarist_GID-588967822

Usage: S(n):Musician_GID-588967

Show: ☑ this ☑ disjoints ☑ named sub/superclasses

Found 9 uses of 'S(n):Musician_GID-588967'

- mi:Guitarist_GID-588967822
  - mi:Guitarist_GID-588967822 SubClassOf 'S(n):Musician_GID-588967'
- S(n):Musician_GID-588967
  - 'S(n):Musician_GID-588967' SubClassOf PartOf some 'S(n):Orchestra_GID-56672
  - 'S(n):Musician_GID-588967' SubClassOf 'S(n):Entity_GID-101
  - 'S(n):Musician_GID-588967' rdfs:label "S(n):Musician_GID-588967"@en
  - Class: 'S(n):Musician_GID-588967'

Description: S(n):Musician_GID-588967

Equivalent To

SubClass Of

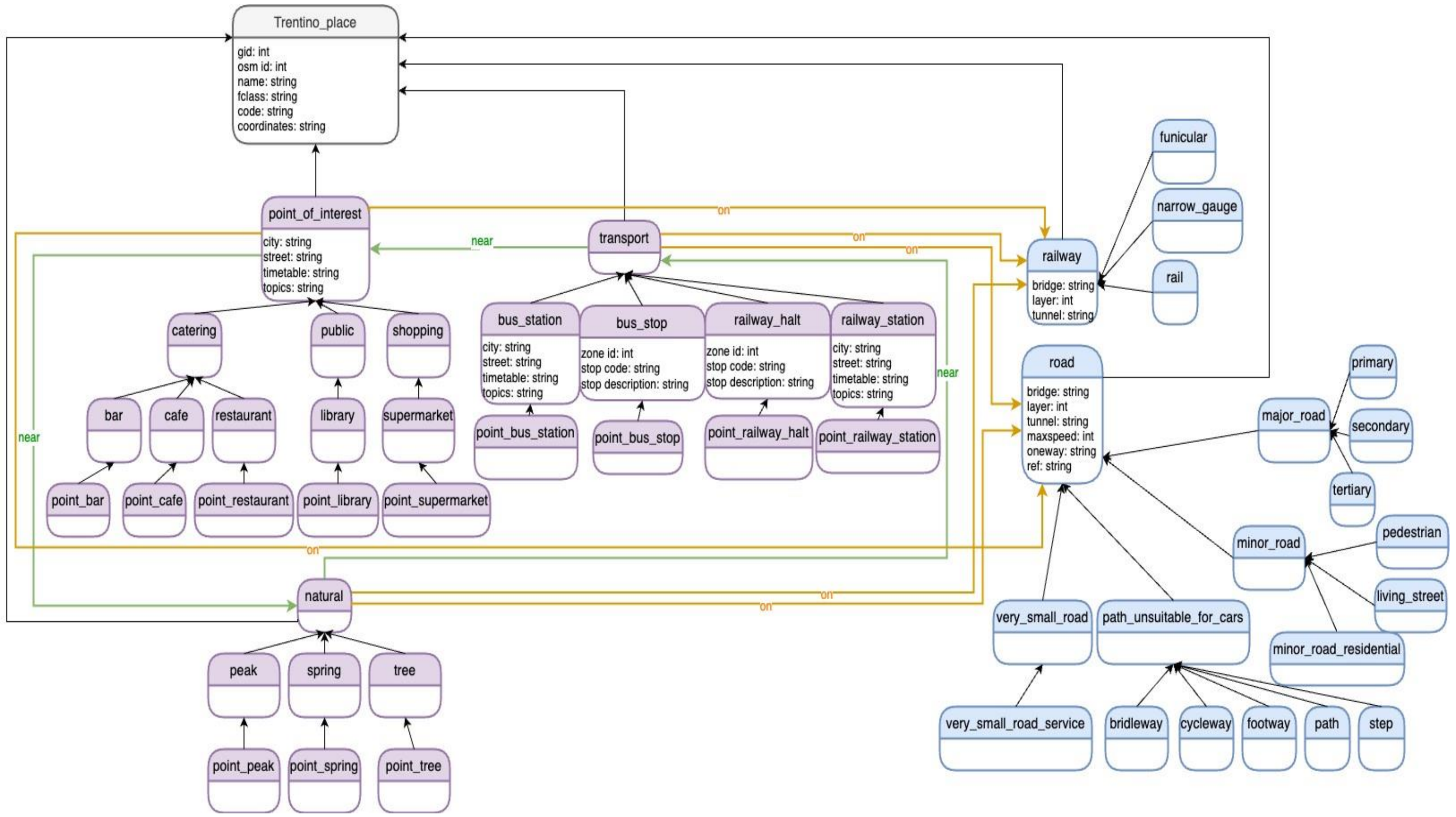
- 'S(n):Entity_GID-101'
- PartOf some 'S(n):Orchestra_GID-56672'

Observation: Formalization language: OWL/ RDF

KG theories

- LoD theories
- Unfolding LoD theories
- ASK – Reasoning about LoD theories
- KG theories
  - Lexicons – ontologies and language teleontologies
  - **EER models – knowledge teleontologies**
  - ER models, schemas – teleologies (ETGs)
- Unfolding KG theories
- ASK – Reasoning with KG theories
- Key notions

EER model – example



On the left hand side the EER model for **Open street map**

Etype: the name at the top in a box

Data property: In the bottom part of a box, associated to their data types.

Object property (horizontal): Links (yellow or green) across etypes, for instance "near", "on", "off".

MoreGeneral/ LessGeneral (MG/LG) Object property (horizontal): Links (black) across etypes. It means that the lower LG etypes inherit the properties of the MG etype

Knowledge teleontology – a formalized EER model

Definition (Teleontology etype description). A teleontology etype description is as follows

$$\text{LabelEtype} \equiv \text{GenusEtype} \sqcap \text{GenusProperty}$$

where:

- There is a root etype, which is a concept
- A GenusEtype is an etype (LabelEtype in the definition above) defined by a LoD description, starting from the root concept.
- A GenusProperty is a conjunction of object and data properties.

We call any definition above a (**LoD**) **Description**. Label is a (**LoD**) **etype**.

Definition (Knowledge Teleontology). A knowledge teleontology is a language teleontology, possibly consisting of a single genus-differentia definition, extended with a set of etype descriptions.

Observation (language vs. knowledge teleontologies). Language telentologies **define** the meaning of the concepts modeling the elements of the world. Knowledge teleontologies **describe** the properties of the language teleontology concepts by adding new etypes and by providing relevant properties.

Observation (etype vs. concept). A LoD etype LabelEtype is NOT a LoD Concept but a description of a LoD concept. It does not have the disjointness properties of LoD concepts. Semantically, a LoD etype is a subset of the concept that it describes.

Knowledge teleontology formalization – example

Example (Language teleontology). A possible definition of the concept denoting electric guitars

$$\text{ElectricGuitar} \equiv \text{Guitar} \sqcap \text{SoundAmplification}$$

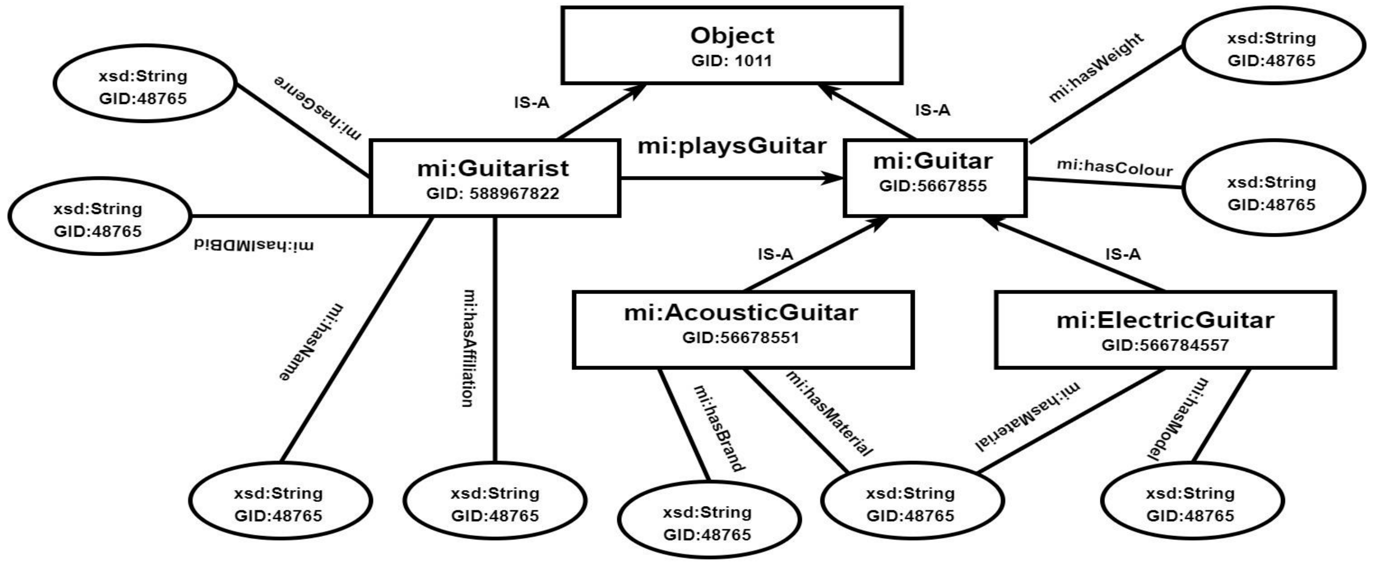
$$\text{ElectricGuitar} \perp \text{AcousticGuitar}$$

… *more disjointness axioms*

Example (Knowledge teleontology). Description of a specific type of electric guitar, that we call ElectricGuitar#1, as a shortcut for "an electric guitar which is colored and has a brand"

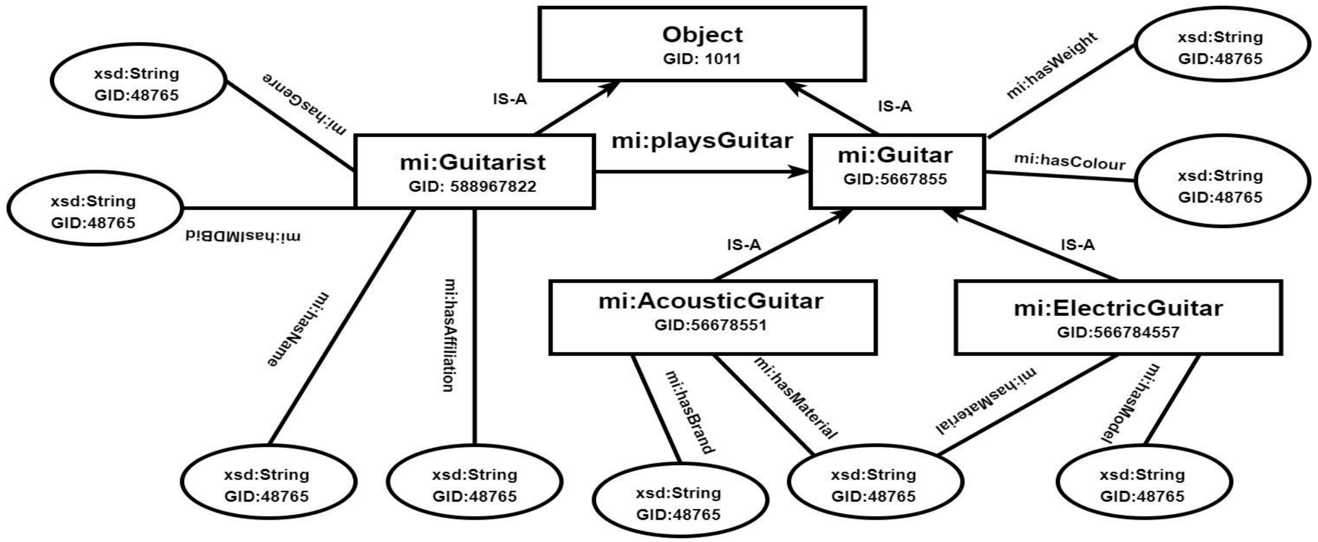
$$\text{ElectricGuitar\#1} \equiv \text{ElectricGuitar} \sqcap \exists\text{hasColour.String} \sqcap \exists\text{hasBrand.String}$$

Knowledge teleontology (example)



Observation (Description & definition). The picture above uses the state of the art (informal) notation used in EER models / KGs: properties are associated to LoD concepts and the Genus-Differentias are left implicit (see also example above of EER models). This graphical notation does not map to the definition of teleontology which requires a new name (id) for an etype associated to a concept. This notation has the key property and advantage of showing how the different LoD etypes /concepts interact in a complex scenario.

Knowledge teleontology (example)

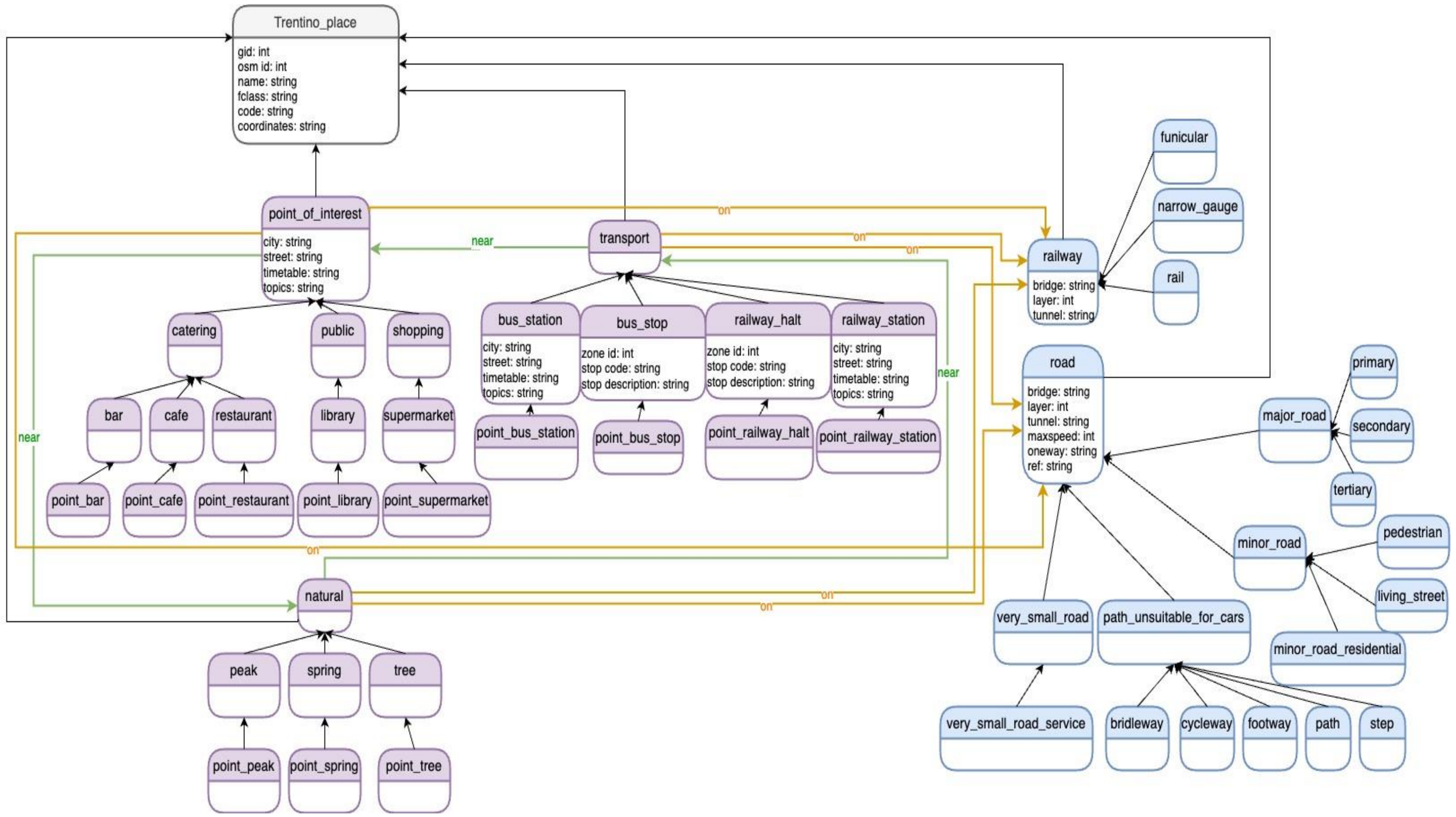


Observation (Description & definition). This notation works ok when there is one etype per concept. With more etypes for it generates concept polysemy: one concept described in two different ways. The graphical solution is to add an etype node for each etype of the same concept, still with the same concept id/name but a different etype ids/ names.

Example (Two etypes for the same concept). Given the concept “person” create an etype node for “professor” and “one” for “student”. Another solution is to make “professor” and “student” new concepts, via suitable genus and differentia.

Question (Possible ambiguity?). What if we have two different LoD descriptions for the same etype, e.g., two instances of the etype “professor” for the concept “person”? And what is we have two different LoD definitions for the same concept?

EER model – The practice - example



On the left hand side the EER model for **Open street map.**

Etype: concepts annotated by properties.

Etype name = concept name.

Genus and Differentia: left implicit.

Result: works well with humans. Creates problems when used to provide information to AI systems (less than NL text). Both when doing automated reasoning and when doing machine learning.

LOD – teleontology formalization (summary)

Observation (Description & definition). A LoD etype is subsumed by its concept

$$\text{ElectricGuitar\#1} \sqsubseteq \text{ElectricGuitar}$$

Observation (Description & definition). A description is a definition enriched with data properties (only conjuncts). The etype defined by a description MUST have a different name / identifier (in the LoD formulation).

Observation (Description & definition). The same definition can be associated multiple diverse descriptions. That is, there can be multiple etypes, with different names for the same concept.

Observation (Description & definition). You can have an etype with genus a concept but not a concept with genus an etype.

Observation (Root of a teleontology). The root of a teleontology is always a concept. A teleontology not necessarily, contains only one concept, the root. See EER model example in the next slide.

Terminology (language vs. knowledge teleontologies). We drop the attribute language/ knowledge when the context makes clear the meaning.

Teleontology – Protégé

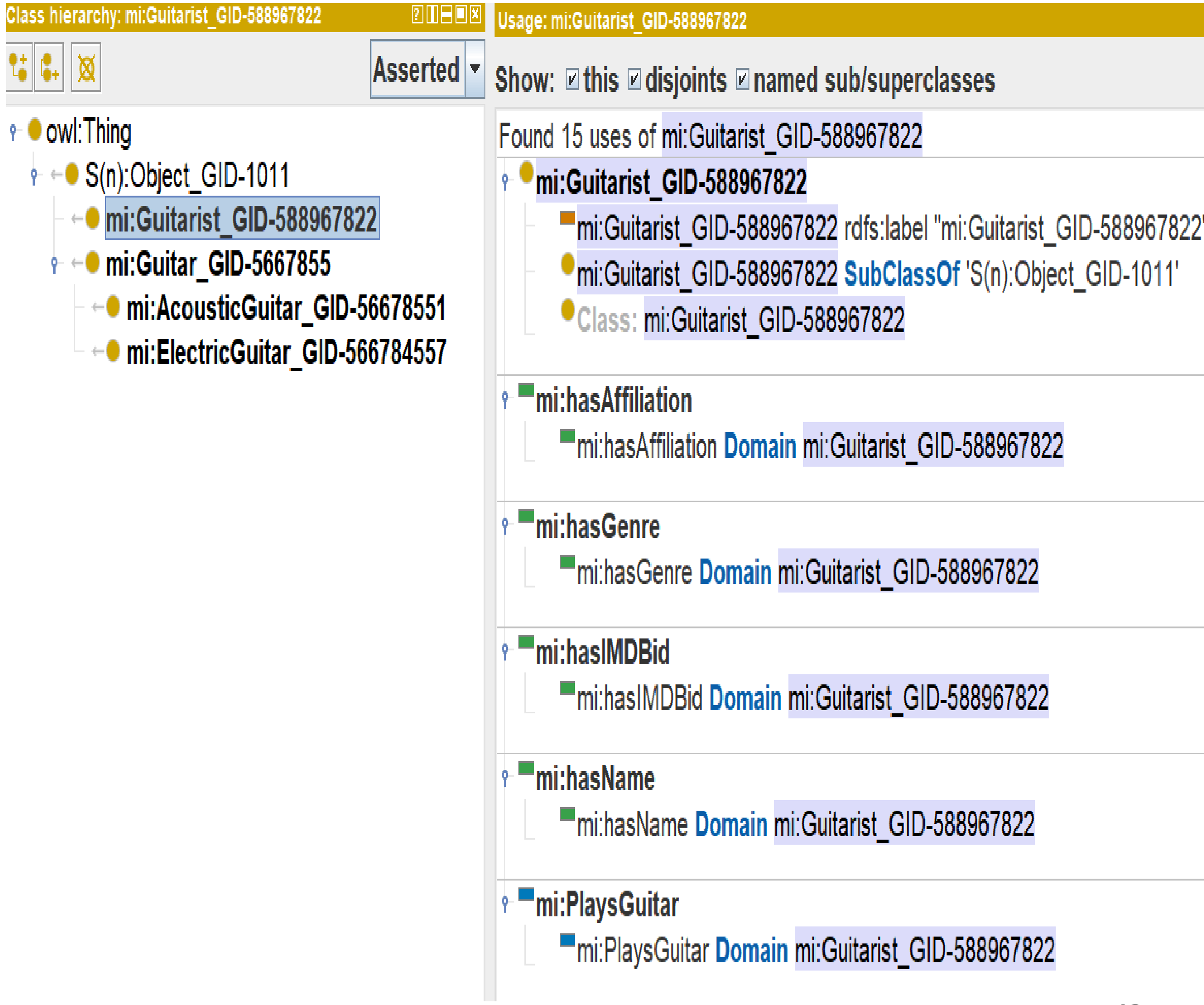
The snippets on the right hand side shows the knowledge teleontology formalized via the Protégé ontology editor.

We add object properties (e.g., playsGuitar) and additional data properties (e.g., hasIMDBid) here.

You can see some (partial) visualization of LOD formalization for, e.g.,

e.g., *mi:Guitarist - mi:PlaysGuitar - mi:Guitar* is an object property-based assertion which indicates that a guitarist plays a guitar.

mi:Guitarist - mi:hasIMDBid - xsd:String is a data property-based assertion which indicates that a guitarist has an IMDB id encoded as a string.



Class hierarchy: mi:Guitarist_GID-588967822

Asserted

- owl:Thing
  - S(n):Object_GID-1011
    - mi:Guitarist_GID-588967822
    - mi:Guitar_GID-5667855
      - mi:AcousticGuitar_GID-56678551
      - mi:ElectricGuitar_GID-566784557

Usage: mi:Guitarist_GID-588967822

Show: ☑ this ☑ disjoints ☑ named sub/superclasses

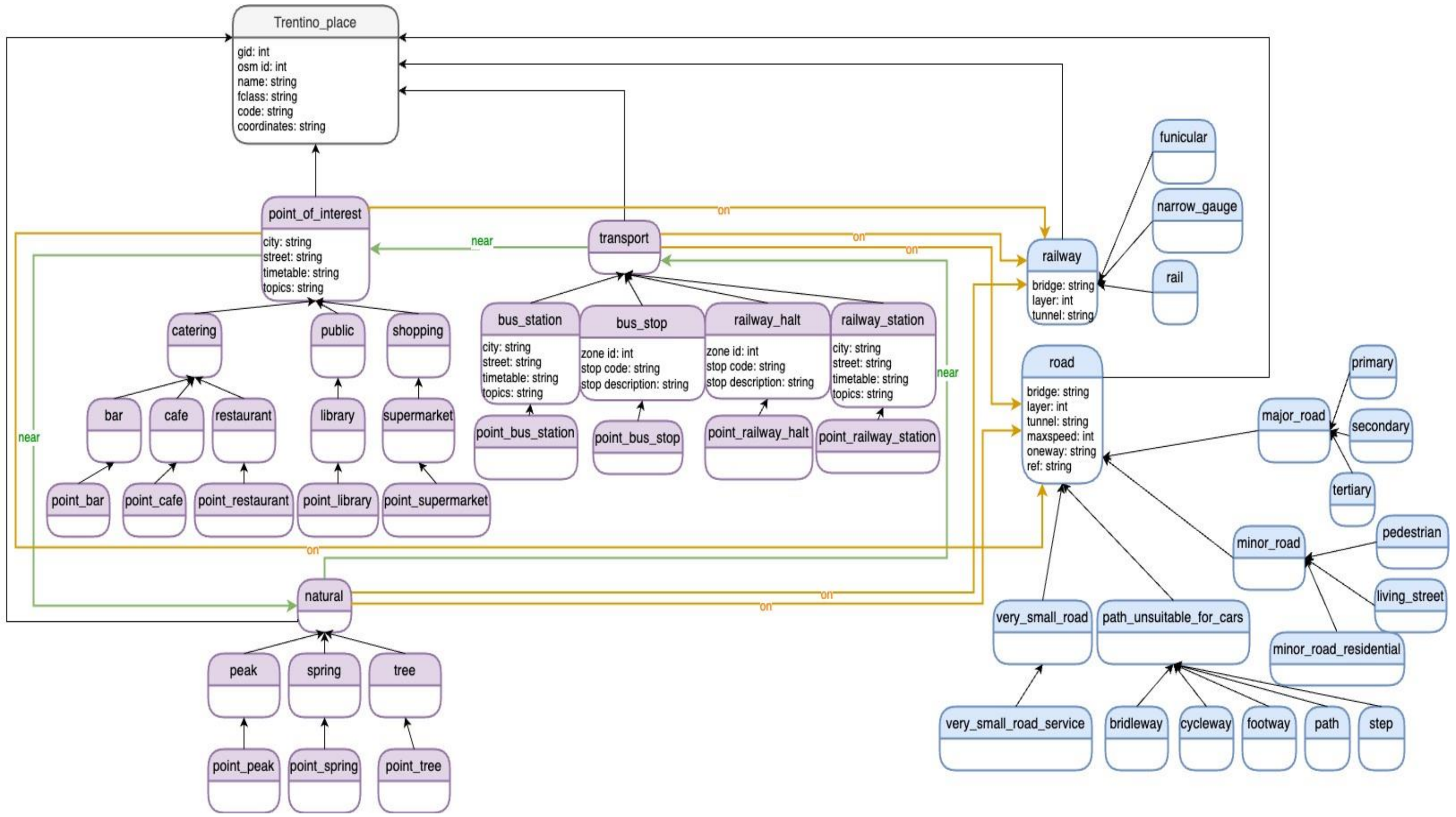
Found 15 uses of mi:Guitarist_GID-588967822

- mi:Guitarist_GID-588967822
  - mi:Guitarist_GID-588967822 rdfs:label "mi:Guitarist_GID-588967822"
  - mi:Guitarist_GID-588967822 SubClassOf 'S(n):Object_GID-1011'
  - Class: mi:Guitarist_GID-588967822
- mi:hasAffiliation
  - mi:hasAffiliation Domain mi:Guitarist_GID-588967822
- mi:hasGenre
  - mi:hasGenre Domain mi:Guitarist_GID-588967822
- mi:hasIMDBid
  - mi:hasIMDBid Domain mi:Guitarist_GID-588967822
- mi:hasName
  - mi:hasName Domain mi:Guitarist_GID-588967822
- mi:PlaysGuitar
  - mi:PlaysGuitar Domain mi:Guitarist_GID-588967822

KG theories

- LoD theories
- Unfolding LoD theories
- ASK – Reasoning about LoD theories
- KG theories
  - Lexicons – ontologies and language teleontologies
  - EER models – knowledge teleontologies
  - **ER models, schemas – teleologies (ETGs)**
- Unfolding KG theories
- ASK – Reasoning with KG theories
- Key notions

ER model – example



On the left hand side the EER model for **Open street map**

Etype: the name at the top in a box

Data property: In the bottom part of a box, associated to their data property.

Object property (horizontal): Links (yellow or green) across etypes, for instance "near", "on", "off".

~~**MoreGeneral/ LessGeneral (MG/LG) Object property (horizontal):** Links (black) across etypes. It means that the lower LG etypes inherit the properties of the MG etype~~

Teleology – a formalized ER model

Definition (Teleologies). Teleologies are teleontologies where all the defined etypes are unfolded into a conjunction of primitive etypes.

Observation: In case of multiple types for the same concept, the unfolding must keep track of which property is attached to which etype definition.

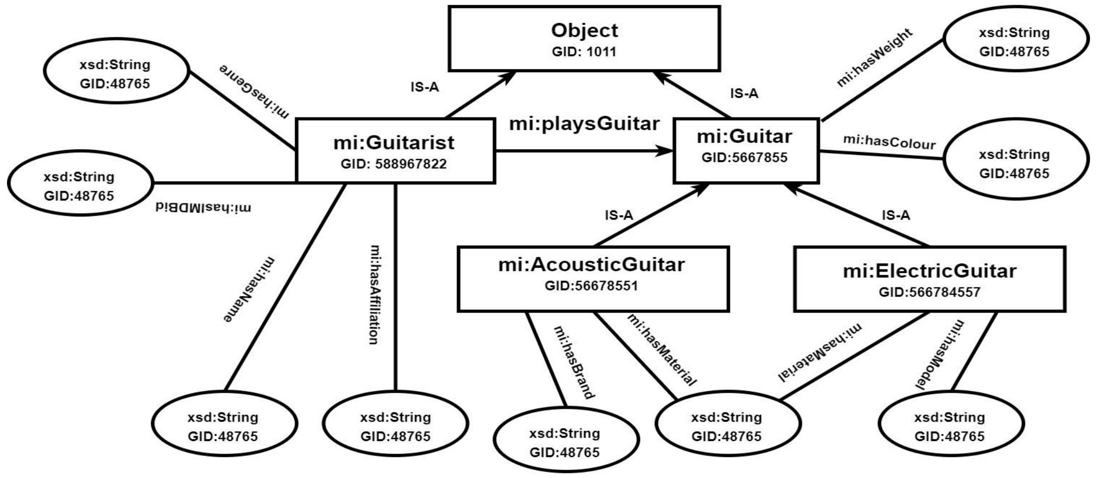
Observation. In mainstream SW engineering, the most common approach is as follows:

- Build the ER model,for instance on the basis of some informal requirements written in natural language.
- Build the EER model as the way to make the specification and follow-up application cleaner, for instance via the use of suitably defined classes.

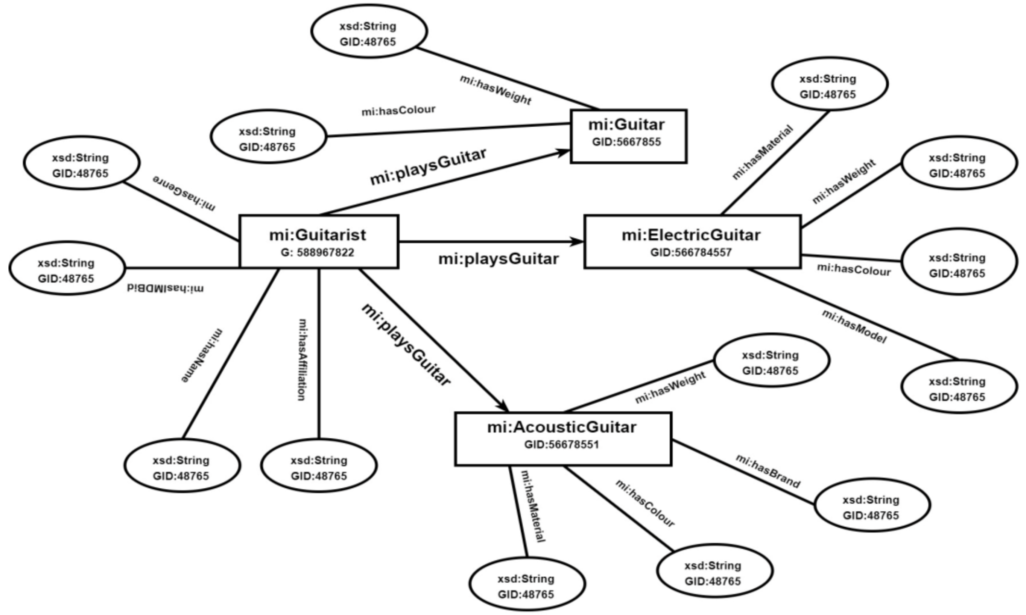
The approach suggested by a Logic-based approach is opposite. It requires a priorly defined ontology. Advatanges: it guarantees much higher quality SW and interoperability. Disadvantages: lots of work to have a general enough ontology.

From teleontologies to teleologies – example

FROM



TO



From teleontologies to teleologies – example

FROM

$Woman \equiv Person \sqcap Female$

$Man \equiv Person \sqcap \neg Female$

$Mother \equiv Woman \sqcap \exists HasChild.Person$

$Father \equiv Man \sqcap \exists HasChild.Person$

TO

$Woman \equiv Person \sqcap Female$

$Man \equiv Person \sqcap \neg Female$

$Mother \equiv Person \sqcap Female \sqcap \exists HasChild.Person$

$Father \equiv Person \sqcap \neg Female \sqcap \exists HasChild.Person$

Teleology - Example

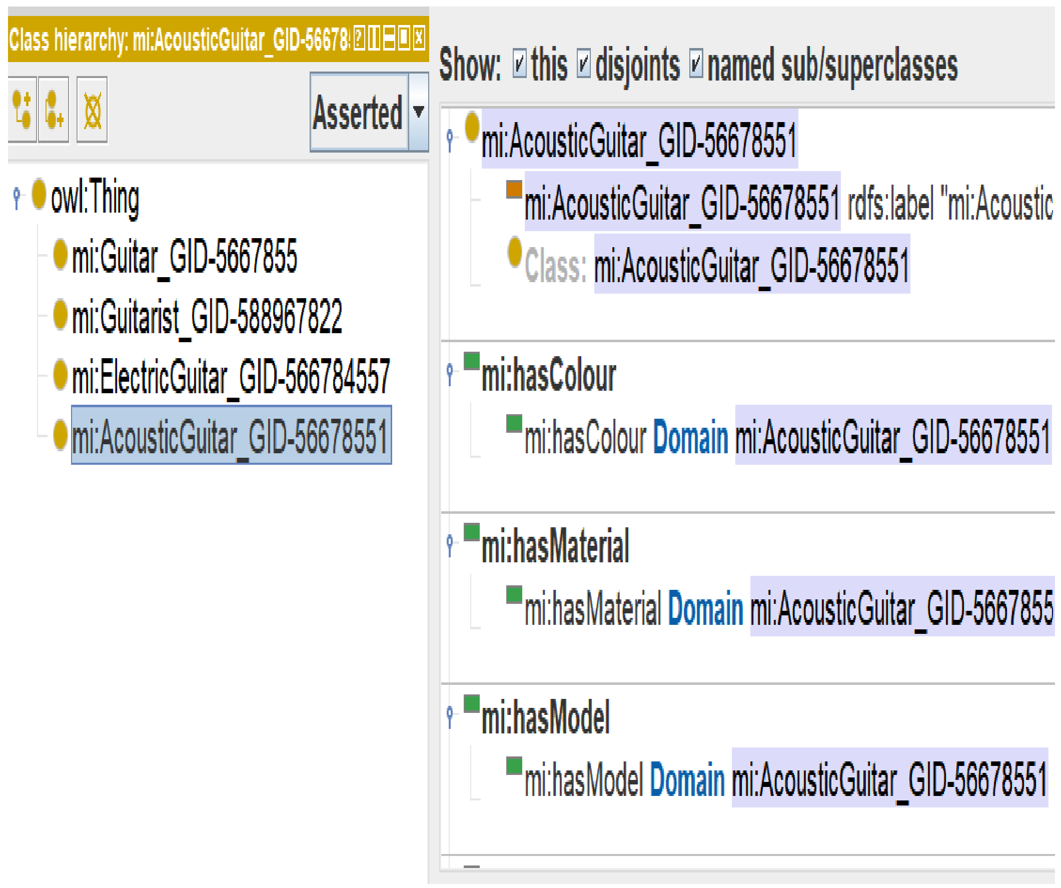
The snippet on the right hand side shows (partially) the teleology formalized via the Protégé ontology editor.

Notice that the class hierarchy is completely flattened, i.e., there are no IS-A links asserting superclass-subclass subsumption relationships.

You can see some (partial) visualization of LOD formalization for, e.g.,

e.g., *mi:AcousticGuitar - mi:hasColour - xsd:String* is a data property-based assertion which indicates that an acoustic guitar has a color which is encoded as a String.

e.g., *mi:AcousticGuitar - mi:hasModel - xsd:String* is a data property-based assertion which indicates that an acoustic guitar is of a specific model spcification encoded as a String.


Class hierarchy: mi:AcousticGuitar_GID-56678

Asserted

owl:Thing
- mi:Guitar_GID-5667855
- mi:Guitarist_GID-588967822
- mi:ElectricGuitar_GID-566784557
- mi:AcousticGuitar_GID-56678551

Show: ☑ this ☑ disjoints ☑ named sub/superclasses

mi:AcousticGuitar_GID-56678551
- mi:AcousticGuitar_GID-56678551 rdfs:label "mi:Acoustic
- Class: mi:AcousticGuitar_GID-56678551

mi:hasColour
- mi:hasColour Domain mi:AcousticGuitar_GID-56678551

mi:hasMaterial
- mi:hasMaterial Domain mi:AcousticGuitar_GID-5667855

mi:hasModel
- mi:hasModel Domain mi:AcousticGuitar_GID-56678551

KG theories

- LoD theories
- Unfolding LoD theories
- ASK – Reasoning about LoD theories
- KG theories
  - Lexicons – ontologies and language teleontologies
  - EER models – knowledge teleontologies
  - ER models, schemas – teleologies (ETGs)
- **Unfolding KG theories**
- ASK – Reasoning with KG theories
- Key notions

Unfolding an etype (reprise)

Definition (Definiendum, definiens): The left hand side of a definition A ≡ C (that is, A) is called **definiendum**, the right hand side (that is, C) is called the definiens.

Definition (Defined and primitive etype): Given a TBox, a **defined** etype is an etype which appears on the left of a definition of the TBox. A **primitive** etype is an etype which only appears on the right of the definitions. A primitive type is an **atomic** etype. A defined etype is a **complex** etype.

Definition (etype unfolding) A defined etype is **unfolded** if all the defined etypes occurring in its definiens are recursively substituted with their definition.

Observation (unfolded etype): The definition of an unfolded etype contains only primitive etypes

Example. From:

ElectricGuitar ≡ Guitar ⊓ SoundAmplification

ElectricGuitar#1 ≡ ElectricGuitar ⊓ ∃hasColour.String ⊓ ∃hasBrand.String

… to:

ElectricGuitar#1 ≡ Guitar ⊓ SoundAmplification ⊓ ∃hasColour.String ⊓ ∃hasBrand.String

Remark: In an acyclic terminology the process of etype unfolding is applied recursively up to any level, usually with the goal of reducing to primitive etypes. The process is guaranteed to terminate.

Complexity of teleontology unfolding

Observation (Teleontology unfolding). Teleontology descriptions are like macros that can be expanded to produce teleologies. Teleologies unfold into themselves.

Observation (Complexity of language teleontology unfolding). The size of the unfolded definition grows **polinomially** with the depth of the teleontology. For instance, from

$$L2 \equiv G2 \sqcap D2$$

$$L1 \equiv L2 \sqcap D1$$

$$L0 \equiv L1 \sqcap D0$$

we obtain

$$L1 \equiv G2 \sqcap D2 \sqcap D1 \qquad \text{(times 2+1 Differentia)}$$

$$L0 \equiv G2 \sqcap D2 \sqcap D1 \sqcap D0 \qquad \text{(times 2 +2 Differentia)}$$

Observation (Complexity of unfolding). The number of paths grows polinomially under the assumption that differentias are primitive etypes

Observation (Complexity of unfolding). The growth is local to the subtree and not to all the teleontology.

Observation (Complexity of knowledge teleontology unfolding). The results for language teleontologies can be replicated only if where we have D0, …, D2 we have types which are NOT defined etypes.

KG theories

- LoD theories
- Unfolding LoD theories
- ASK – Reasoning about LoD theories
- KG theories
  - Lexicons – ontologies and language teleontologies
  - EER models – knowledge teleontologies
  - ER models, schemas – teleologies (ETGs)
- Unfolding KG theories
- **ASK – Reasoning with KG theories**
- Key notions

Entailment with teleontologies

Theorem 1: T |= *p*, that is, *p* satisfiable by T, iff, after unfolding there is no conjunct which occurs both negated and not negated in p_1

Theorem 2: T |= $p_1 \sqsubseteq p_2$ iff, after unfolding, the conjuncts in p_1 are a superset or equal to those of p_2

Theorem 3: T |= $p_1 \equiv p_2$ iff, after unfolding, the conjuncts in p_1 and in p_2 are exactly the same as those of p_2

Theorem 4: T |= $p_1 \perp p_2$ (the same as: T |= $p_1 \sqsubseteq \neg p_2$) iff, after unfolding, one of the conjuncts occurs negated in p_1 and not negated in p_2, or vice versa

Entailment with teleontologies

Observation (reprise): Since we reason with teleontologies we are dealing with TBoxes with only conjuncts

Observation (Entailment with teleontologies): Teleontologies are nested subsumption hierarchies. Teleologies are unfolded teleontologies. That is, entailment in teleontologies is performed by first reducing teleontologies into teleologies.

Observation (Theorem 3): For $p_1 \sqsubseteq p_2$ to hold p_1 must have more (and not less!) conjuncts than p_2. In fact, adding conjuncts makes a set smaller

Observation (ASK). In the general case, the ASK language allows for complex percepts

Satisfiability with teleontologies (reprise)

- Bachelor $\equiv$ Student $\sqcap$ Undergraduate
- Master $\equiv$ Student $\sqcap \neg$ Undergraduate
- PhD $\equiv$ Student $\sqcap \neg$ Undergraduate $\sqcap$ Research
- Assistant $\equiv$ Student $\sqcap \neg$ Undergraduate $\sqcap$ Research $\sqcap$ Teach

Is

$$\text{Bachelor} \sqcap \text{PhD}$$

satisfiable? NO!

Observation. Unfold the query and compare the conjuncts.

Subsumption with teleontologies (reprise)

- Bachelor $\equiv$ Student $\sqcap$ Undergraduate
- Master $\equiv$ Student $\sqcap \neg$ Undergraduate
- PhD $\equiv$ Student $\sqcap \neg$ Undergraduate $\sqcap$ Research
- Assistant $\equiv$ Student $\sqcap \neg$ Undergraduate $\sqcap$ Research $\sqcap$ Teach

Is

$$\text{PhD} \sqsubseteq \text{Student}$$

satisfiable? YES!

Observation. Unfold the query and compare the conjuncts.

Equivalence with teleontologies (reprise)

- Bachelor $\equiv$ Student $\sqcap$ Undergraduate
- Master $\equiv$ Student $\sqcap \neg$ Undergraduate
- PhD $\equiv$ Student $\sqcap \neg$ Undergraduate $\sqcap$ Research
- Assistant $\equiv$ Student $\sqcap \neg$ Undergraduate $\sqcap$ Research $\sqcap$ Teach

Is)

$$\text{Student} \equiv \text{Bachelor} \sqcup \text{Master}$$

satisfiable? YES!

Observation. Unfold the query. Notice that we have extended the query language to allow for disjunction.

Disjointness with teleontologies (reprise)

- Bachelor $\equiv$ Student $\sqcap$ Undergraduate
- Master $\equiv$ Student $\sqcap \neg$ Undergraduate
- PhD $\equiv$ Student $\sqcap \neg$ Undergraduate $\sqcap$ Research
- Assistant $\equiv$ Student $\sqcap \neg$ Undergraduate $\sqcap$ Research $\sqcap$ Teach

Is

$$\text{Assistant} \perp \text{Undergraduate}$$

satisfiable? YES!

Observation. You could also check Assistant $\sqsubseteq \neg$ Undergraduate

KG theories

- LoD theories
- Unfolding LoD theories
- ASK – Reasoning about LoD theories
- KG theories
  - Lexicons – ontologies and language teleontologies
  - EER models – knowledge teleontologies
  - ER models, schemas – teleologies (ETGs)
- Unfolding KG theories
- ASK – Reasoning with KG theories
- **Key notions**

KG theories

- LoD theories
- TBox, definitional TBox, strongly definitional TBox
- Uses, directly uses, acyclic TBox, Terminology
- Unfolding LoD theories
- Complexity of unfolding
- LoD satisfiability, complexity
- LoD satisfiability, general process
- LoD satisfiability vs LoP satisfiability

- Lexicons – ontologies and language teleontologies
- EER models – knowledge teleontologies
- ER models, schemas – teleologies (ETGs)
- Unfolding KG theories
- ASK – satisfiability
- ASK – equivalence
- ASK – subsumption
- ASK – disjointness

KG theories

Reasoning about Knowledge Graphs

(HP2T)

Natural language Lexicon (continued)

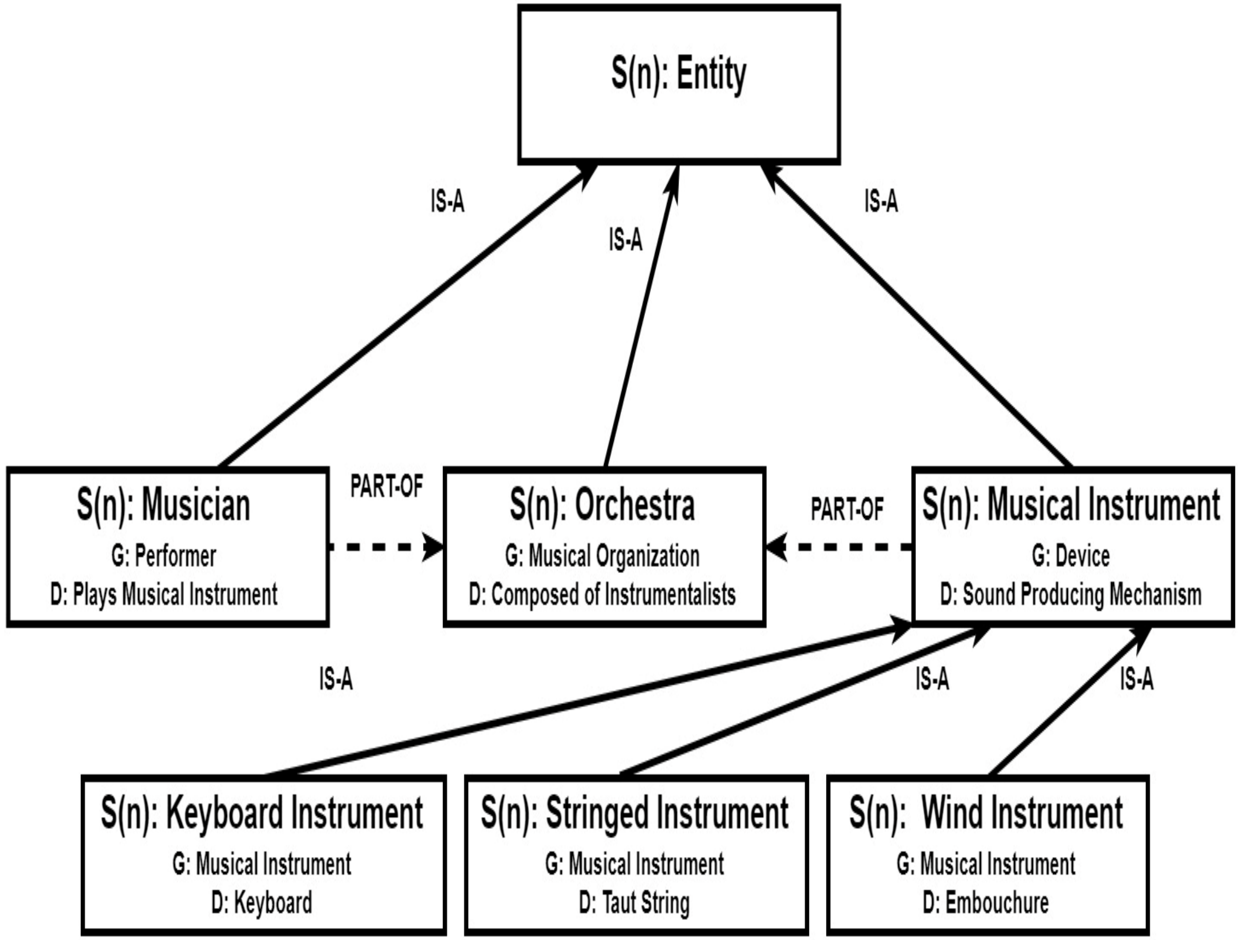
In addition to the IS-A hierarchy, lexicons are also organized according to a **PART-OF** hierarchy.

All concepts have **parts**. For any part there is a "bigger" *whole* which somehow "contains" it. For instance Musicians and Musical Instruments are part-of Orchestras. Musicians have parts, Musicians have parts, …, and so on, down to materials.

Part-of links model the **part-whole** relation which exists between a whole (the **Unity**) and possibly multiple **diverse** parts.

The whole defines the spatial context within whose boundaries the EG is built.

The PART-OF hierarchy defines the relevant component parts of the whole, namely those which will ultimately be considered in an ETG/EG (as, e.g. selected in ER/EER models)



Natural language Lexicon (continued)

The IS-A and PART-OF hierarchies are independent orthogonal hierarchies

The PART-OF hierarchy models containment. Space containment with objects, Time containment with events.

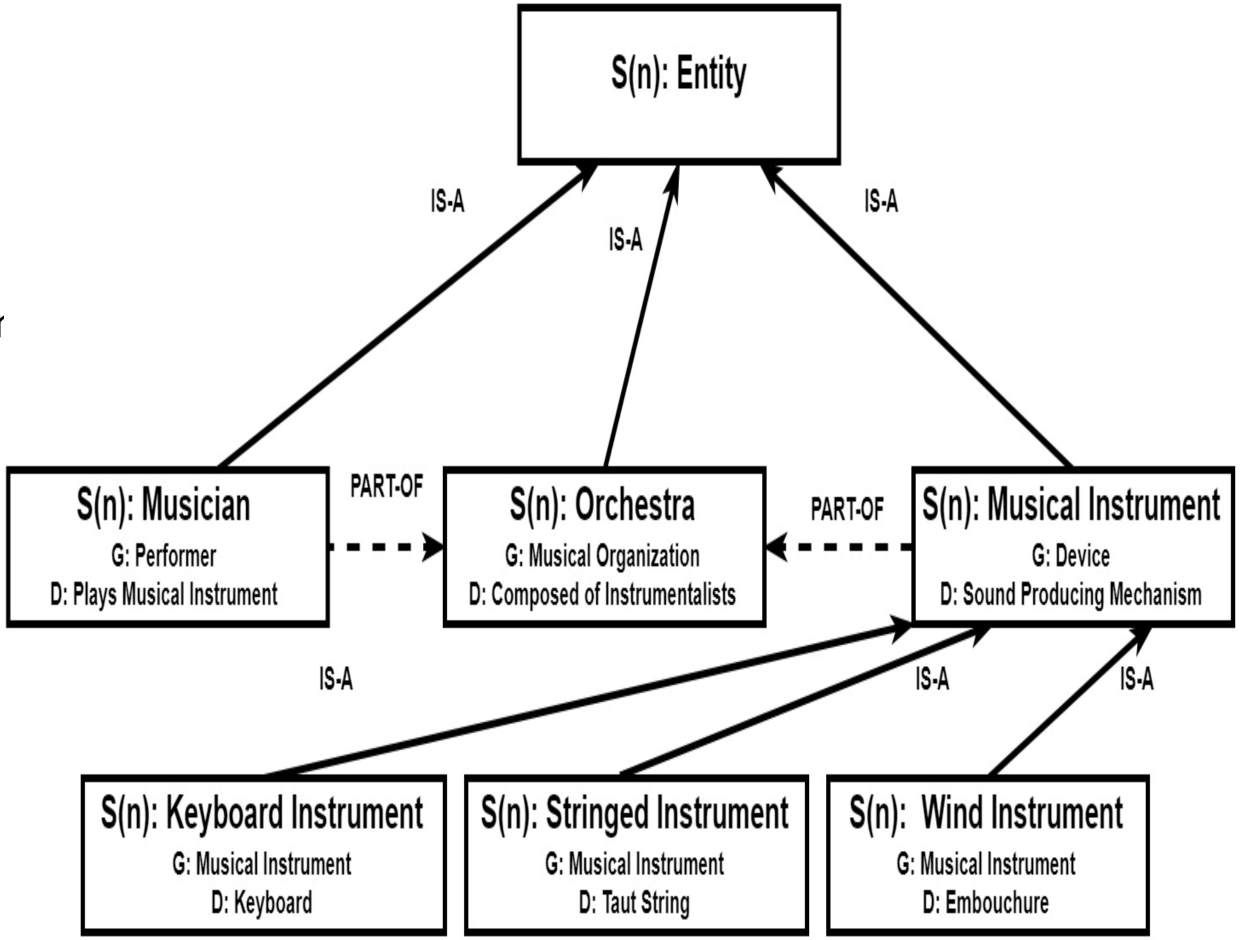
The IS-A hierarchy models the behavior of entities, that is how objects specialize in their properties (i.e., their functior and actions).

Entity (=*anything*), the top concept of the IS-A has no properties and no parts. But it is PART-OF everything.

Everything, the top concept of the PART-OF hierarchy contains all parts and therefore has all properties.

If *PART-OF(part, whole)* **then**
Property(part, P) ≡ Part-Property (whole,P)

The IS-A and PART-OF hierarchies form a *lattice.*



LOD – Lexicon formalization (part-of hierarchy)

Part-of(part, whole)

with inverse relation

Whole-of(label2,label1)

Observation 1: A hierarchy but with no property inheritance

Observation 2: Whole defines the physical (space) boundaries within which the parts are located

Observation 3: Whole provides reference coordinate system, parts provide functionalities

Observation 4: Part-of hierarchy formalized mucs less frequently

Ontology – a formalized lexicon

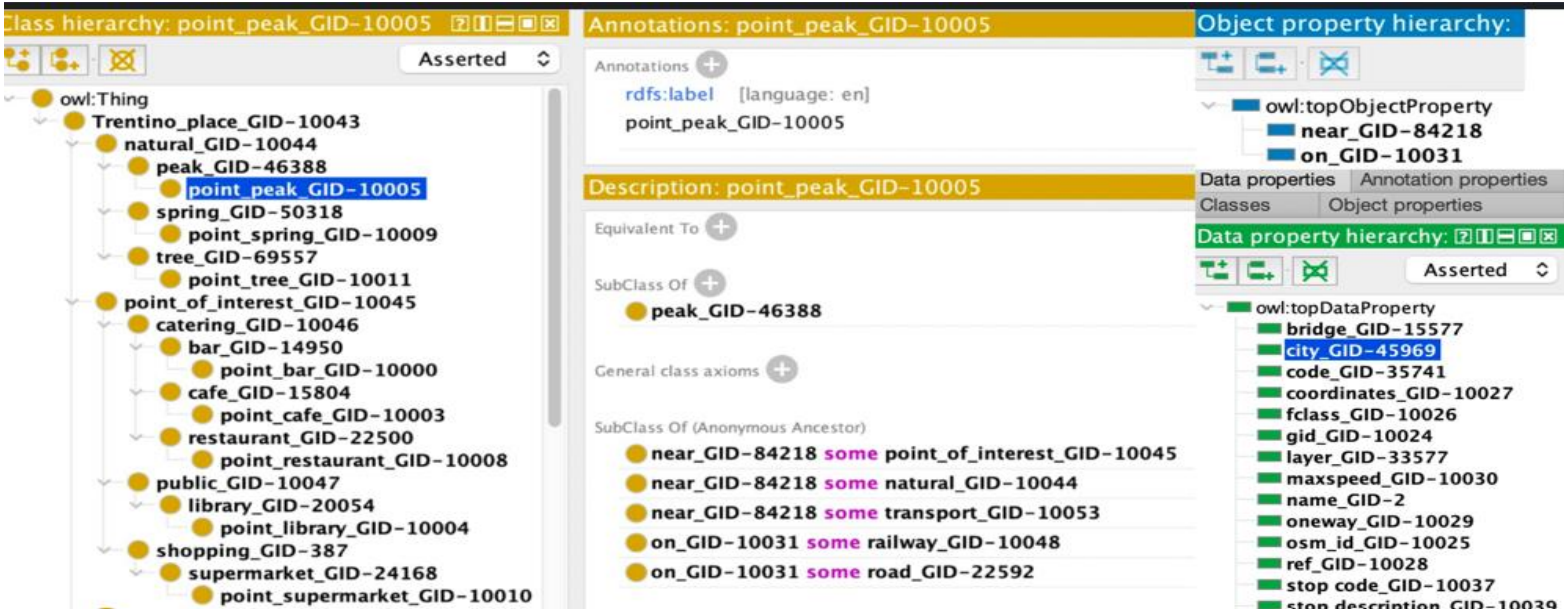
Definition (etype (LOD) concept). We have the following

1. Values: Ids, written word#1, word#2, word#3, …, denoting themselves, that is, word#1, word#2, word#3, …,
2. Data properties: hasSynset, hasGloss, hasExample, …
3. Object properties: sameWord, …
4. Equality: =

Observation (meta-etype). The type concept is a meta-etype as it does not describe the elements of the domain of interpretation but sets of elements of the domain of interpration. Other examples of meta-etypes: etype, dtype, object property, attribute.

Observation (meta-etype). Meta-etypes are metadata. Metadata are used extensively to provide machine readable documentation

Teleontology – Protégé (Open Street Map)



Class hierarchy: point_peak_GID-10005

Asserted

- owl:Thing
  - Trentino_place_GID-10043
    - natural_GID-10044
      - peak_GID-46388
        - point_peak_GID-10005
      - spring_GID-50318
        - point_spring_GID-10009
      - tree_GID-69557
        - point_tree_GID-10011
    - point_of_interest_GID-10045
      - catering_GID-10046
        - bar_GID-14950
          - point_bar_GID-10000
        - cafe_GID-15804
          - point_cafe_GID-10003
        - restaurant_GID-22500
          - point_restaurant_GID-10008
      - public_GID-10047
        - library_GID-20054
          - point_library_GID-10004
      - shopping_GID-387
        - supermarket_GID-24168
          - point_supermarket_GID-10010

Annotations: point_peak_GID-10005

Annotations

rdfs:label [language: en]

point_peak_GID-10005

Description: point_peak_GID-10005

Equivalent To

SubClass Of

peak_GID-46388

General class axioms

SubClass Of (Anonymous Ancestor)

near_GID-84218 some point_of_interest_GID-10045

near_GID-84218 some natural_GID-10044

near_GID-84218 some transport_GID-10053

on_GID-10031 some railway_GID-10048

on_GID-10031 some road_GID-22592

Object property hierarchy:

- owl:topObjectProperty
  - near_GID-84218
  - on_GID-10031

Data properties | Annotation properties

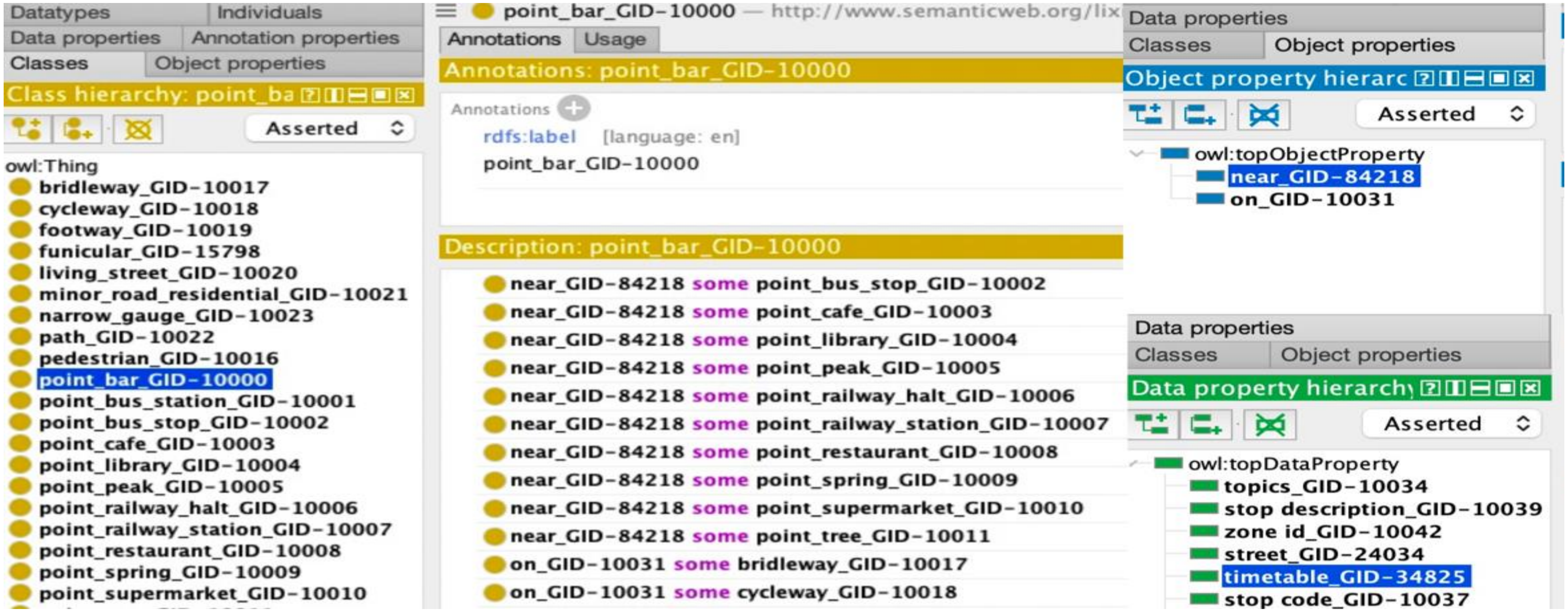
Classes | Object properties

Data property hierarchy:

Asserted

- owl:topDataProperty
  - bridge_GID-15577
  - city_GID-45969
  - code_GID-35741
  - coordinates_GID-10027
  - fclass_GID-10026
  - gid_GID-10024
  - layer_GID-33577
  - maxspeed_GID-10030
  - name_GID-2
  - oneway_GID-10029
  - osm_id_GID-10025
  - ref_GID-10028
  - stop code_GID-10037
  - stop description_GID-10039

Teleology of Open Street Maps data



Datatypes | Individuals
Data properties | Annotation properties
Classes | Object properties
Class hierarchy: point_ba
Asserted

owl:Thing
- bridleway_GID-10017
- cycleway_GID-10018
- footway_GID-10019
- funicular_GID-15798
- living_street_GID-10020
- minor_road_residential_GID-10021
- narrow_gauge_GID-10023
- path_GID-10022
- pedestrian_GID-10016
- point_bar_GID-10000
- point_bus_station_GID-10001
- point_bus_stop_GID-10002
- point_cafe_GID-10003
- point_library_GID-10004
- point_peak_GID-10005
- point_railway_halt_GID-10006
- point_railway_station_GID-10007
- point_restaurant_GID-10008
- point_spring_GID-10009
- point_supermarket_GID-10010

point_bar_GID-10000 — http://www.semanticweb.org/lix

Annotations | Usage

Annotations: point_bar_GID-10000

Annotations
rdfs:label [language: en]
point_bar_GID-10000

Description: point_bar_GID-10000

- near_GID-84218 some point_bus_stop_GID-10002
- near_GID-84218 some point_cafe_GID-10003
- near_GID-84218 some point_library_GID-10004
- near_GID-84218 some point_peak_GID-10005
- near_GID-84218 some point_railway_halt_GID-10006
- near_GID-84218 some point_railway_station_GID-10007
- near_GID-84218 some point_restaurant_GID-10008
- near_GID-84218 some point_spring_GID-10009
- near_GID-84218 some point_supermarket_GID-10010
- near_GID-84218 some point_tree_GID-10011
- on_GID-10031 some bridleway_GID-10017
- on_GID-10031 some cycleway_GID-10018

Data properties
Classes | Object properties
Object property hierarc
Asserted

owl:topObjectProperty
- near_GID-84218
- on_GID-10031

Data properties
Classes | Object properties
Data property hierarchy
Asserted

owl:topDataProperty
- topics_GID-10034
- stop description_GID-10039
- zone id_GID-10042
- street_GID-24034
- timetable_GID-34825
- stop code_GID-10037