LOD – The logic of Descriptions

Theories

LOD - The logic of Descriptions

- TBoxes and terminologies
- LOD theories
- Unfolding

LOD - TBox (definition)

Definition (TBox) A TBox is a finite set of **concept inclusions**, i.e. a finite set of expressions of the type

$$C \sqsubseteq D$$

where *C* is a concept (an etype) and *D* is an atomic formula.

Observation 1: “T” in “”TBox stands for “Term”

Observation 2: TBox is a theory, according to our earlier terminology

Observation 3: A LOD theory is a set of constraints on the domain structure

Observation 4: In general *C* is allowed to be an atomic formula. Our constraint is motivated by our interest in modeling language (definitions and descriptions).

LOD – acyclic TBox (definition)

Definition (Acyclic TBox). A **TBox** is **acyclic** if it satisfies the following properties:

1. any concept can appear at most once on the left side of a definition
2. it is acyclic (there are no definition cycles)

Observation 1: the simplest case of cycle is the subsumption

$$C \sqsubseteq C$$

Observation 2: Acyclicity is crucial in the definition of concepts and in the description of their properties (as etypes).

LOD - Terminology (definition)

Definition (Definitional TBox, Terminology) A **TBox** is **definitional** (it is a **terminology**) if it satisfies the following properties:

1. It is acyclic
2. It contains only concept equivalences

Observation: A concept inclusion $A \sqsubseteq C$ can always be transformed in a definition $A \equiv C \sqcap A_C$ with a suitable A_C

LOD 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

LOD - The logic of Descriptions

- TBoxes and terminologies
- LOD theories
- Unfolding

LOD theories

- Lexicons
- Lexical teleontologies
- Knowledge teleontologies
- Teleologies
- Example

Natural language Lexicon

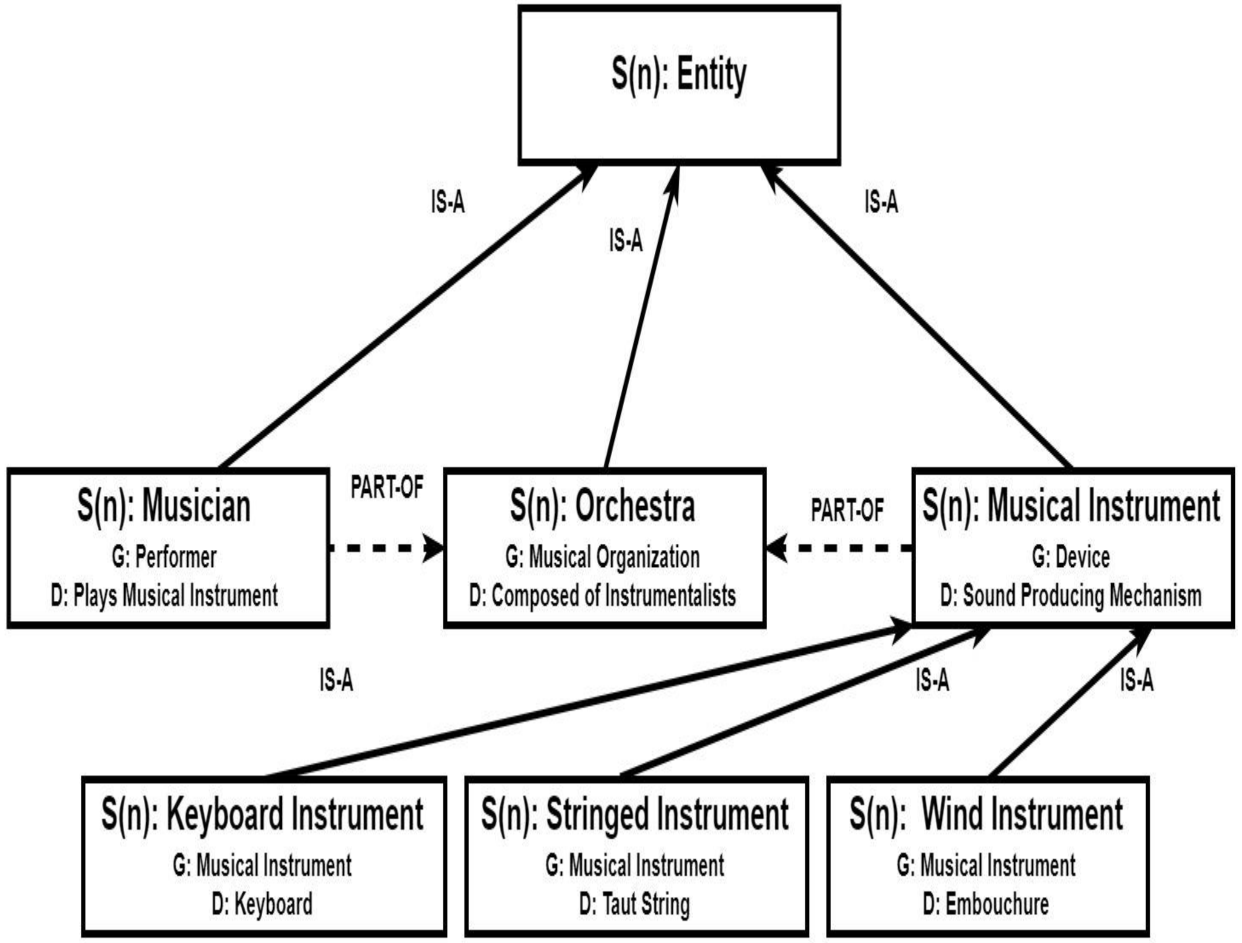
Lexicons are *informal hierarchies* which encode natural language(s) via Genus-Differentia: an ISA hierarchy of synsets.

Sense? One of the many concepts denoted by a polysemous word (e.g., *car* stands for *automobile* and *railway car*).

Synsets? sets of synonyms, e.g., words having same or similar meaning for the same sense (e.g., *car*, *automobile*)

Genus? set of properties which define the scope of a sense, e.g., *musical instrument (G)* for *stringed instrument.*

Differentia? Set of properties which qualify / differentiate senses with the same genus, e.g., *Taut String (D)* for *Stringed Instrument.*



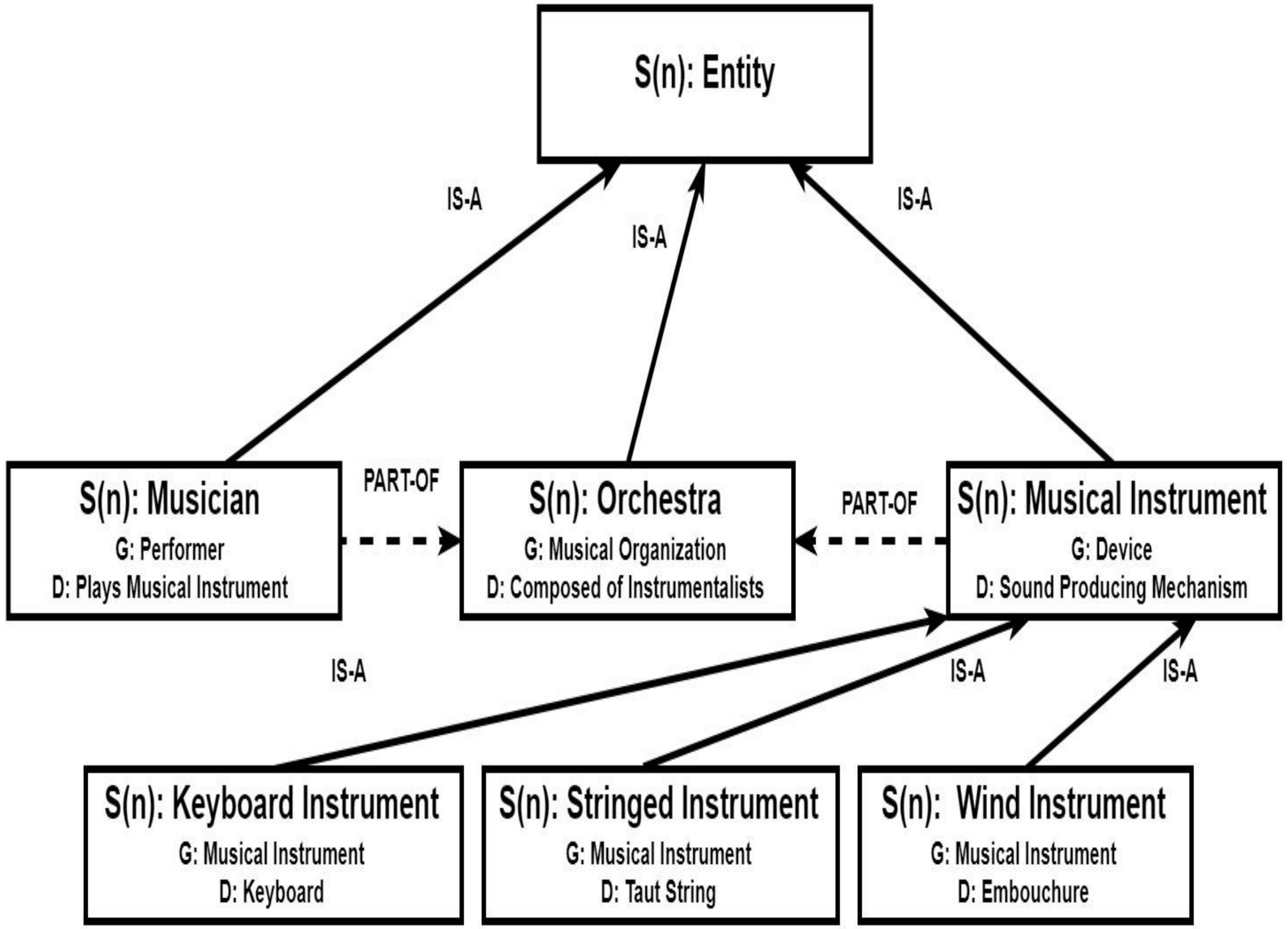
Natural language Lexicon (continued)

The IS-A hierarchy semantically models superclass - subclass relations between senses based on genus-differentia

Each sense is identified via a unique identifier named *GID*, e.g., *588967* for *Musician* (not visible to user)

Each word is (implicitly) defined via a universal quantification over its differentia, e.g., Stringed Instrument's differentia with respect to Musical instrument is ∀*D.TautStrings*.

Lexicons are built by adhering to quality principles for modelling, e.g., how to differentiate a node into children nodes, etc.



Natural language Lexicon (continued)

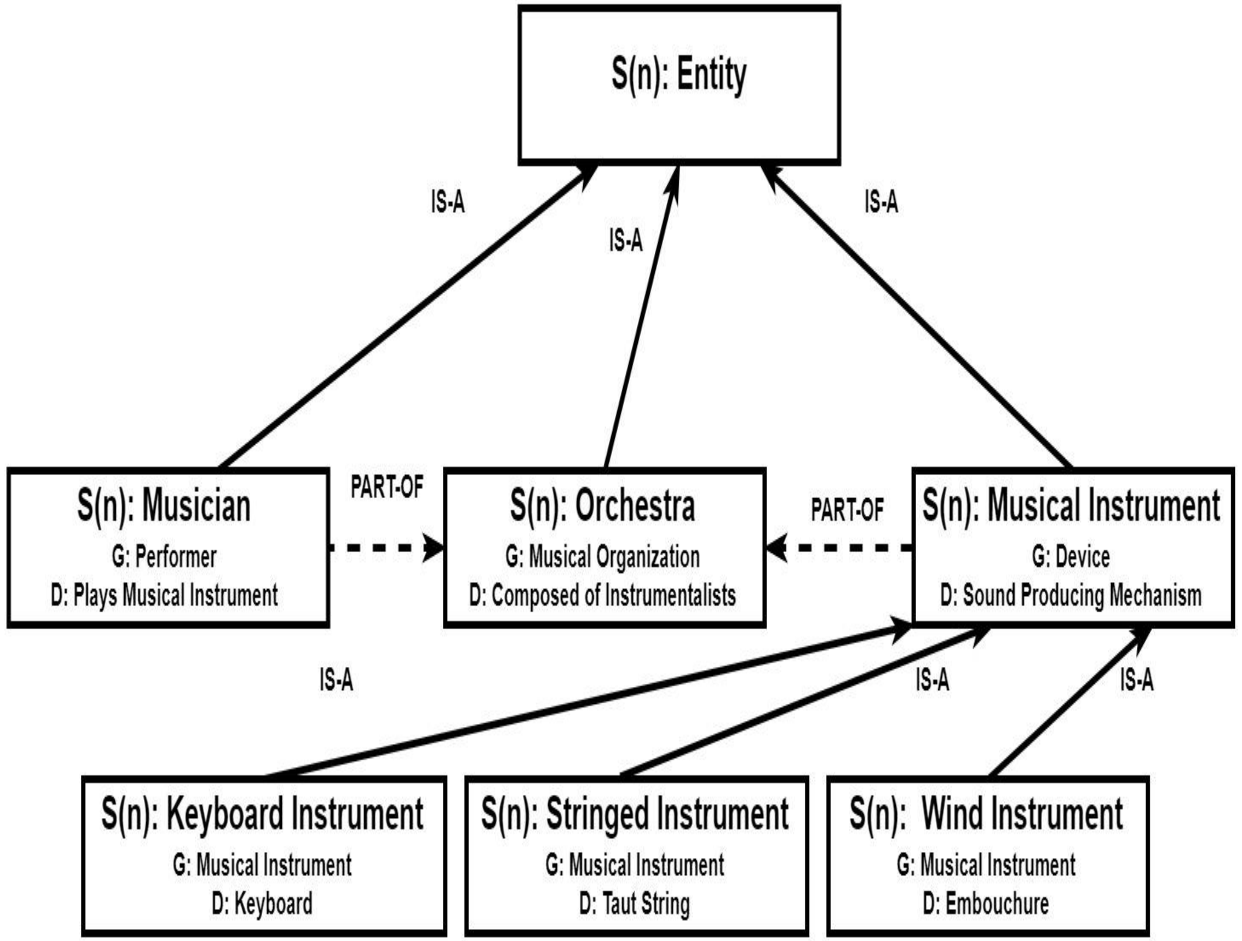
In addition to the IS-A hierarchy, lexicons are also organized according to a PART-OF hierarchy.

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.

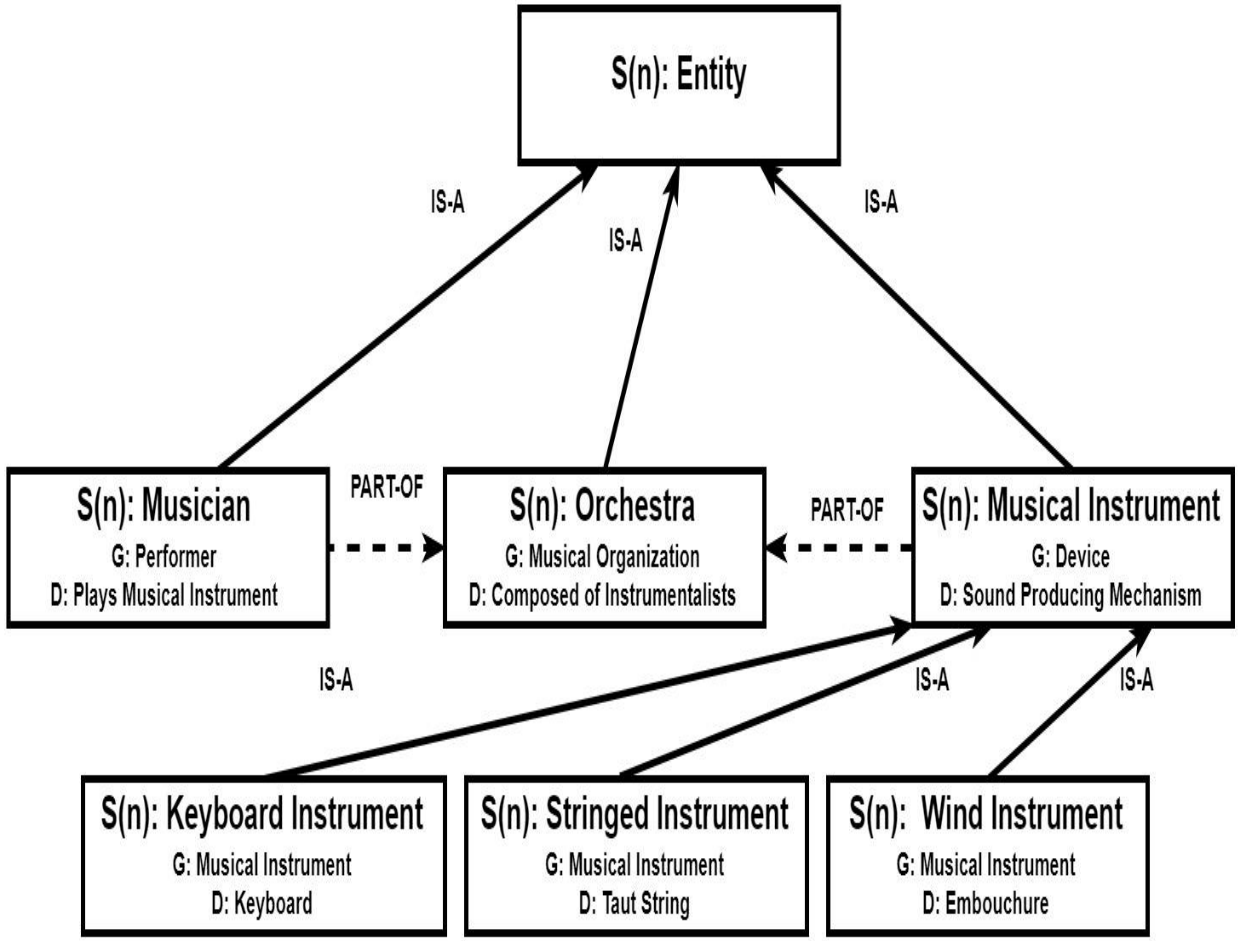
The IS-A hierarchy models the behavior of entities, that is how objects specialize in their properties (i.e., their functions and actions).

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

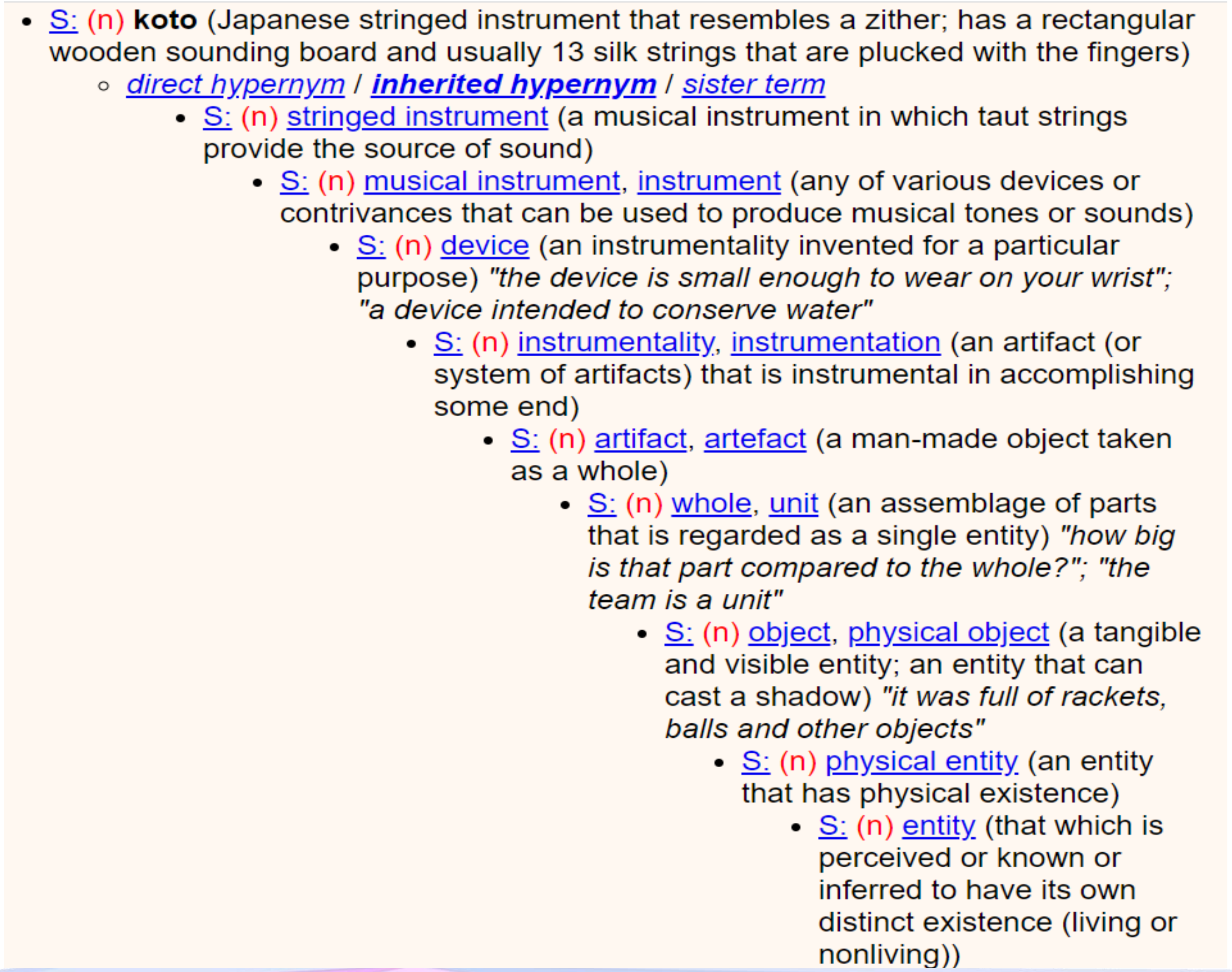
Everything, the top concept of the PART-OF hierarchy contains all parts and therefore has all properties.

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

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



Natural language Lexicon – example (WordNet - Koto)



- 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 indicate subclass and superclass IS-A relationship

Meronym indicates the part-of relation.

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

Domain Lexicon

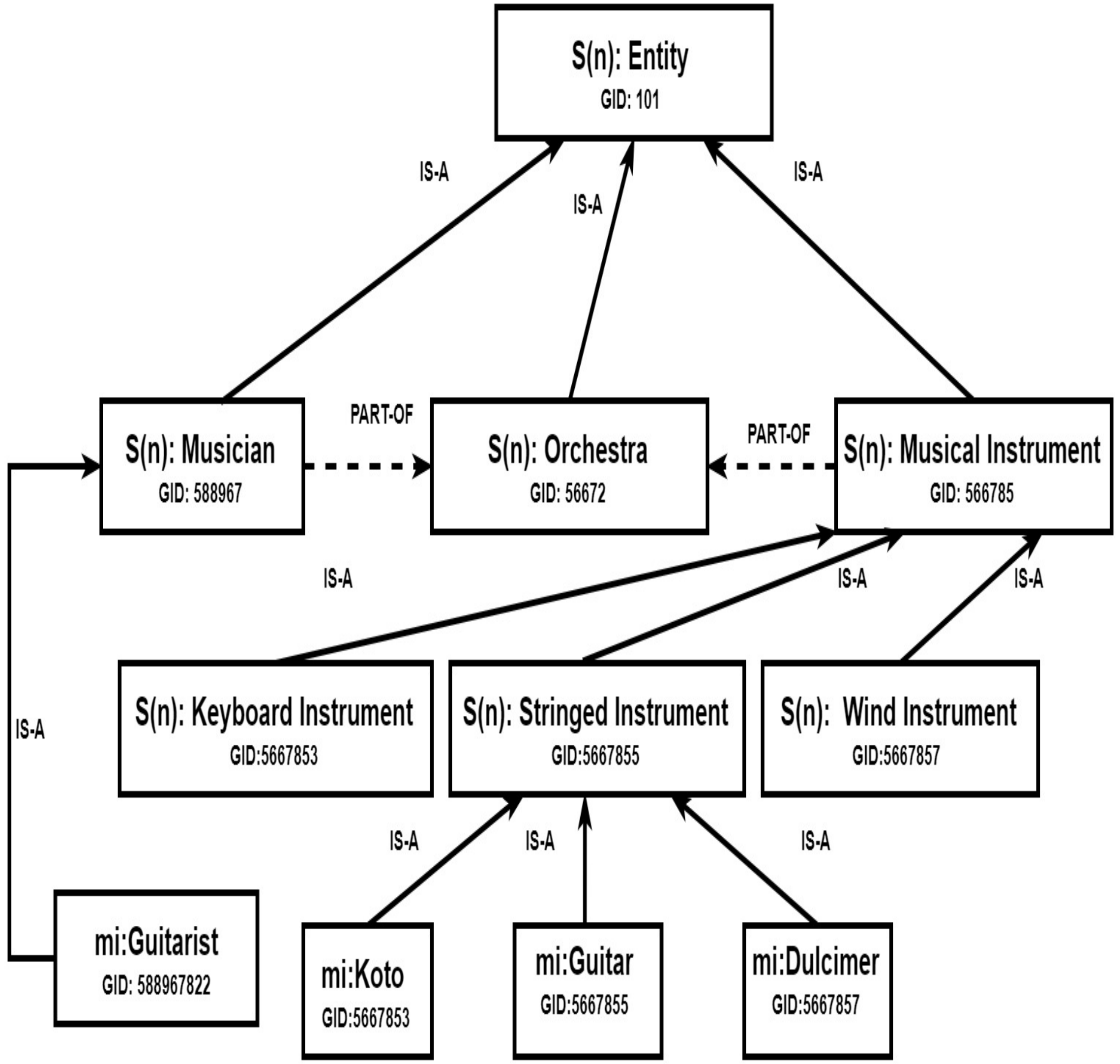
Domain Lexicons (XML name spaces) are informal hierarchies which encode domain language(s)

They extend natural language lexicons with domain-specific terminology. They are therefore language aware.

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 unique identifier named *GID*, e.g., *5667853* for *mi:Koto,* denoting the single sense of that word.



LOD – Lexicon formalization

IS-A hierarchy, a from the example above on musical instruments:

- Label ≡ Genus ⊓ Differentia
- KeyboardInstr ≡ MusicalInstr ⊓
  ∃meansOfSoundProduction.Keyboard ⊓ ∀meansOfSoundProduction.Keyboard …
- KeyboardInstr ⊑ MusicalInstr
- KeyboadInstr ⊓ StringedInstr ⊑ Ø
- KeyboadInstr ⊓ WindInstr ⊑ Ø

Observation 1: only data properties

Observation 2: exists and forall quantifiers (possibly more than one pair)

Observation 3: as many pairwise disjointness constraints as there are siblings

Observation 4: a terminology with only conjuncts and additional disjointness constraints

PART-OF hierarchy, a from the example above on musical instruments:

- Part-of(label1, label2), HasPart(label2,label1) [or HasWhole (label1,label2) Whole-of(label2,label1)]

Observation 5: only object properties with part/whole relation and inverse relation

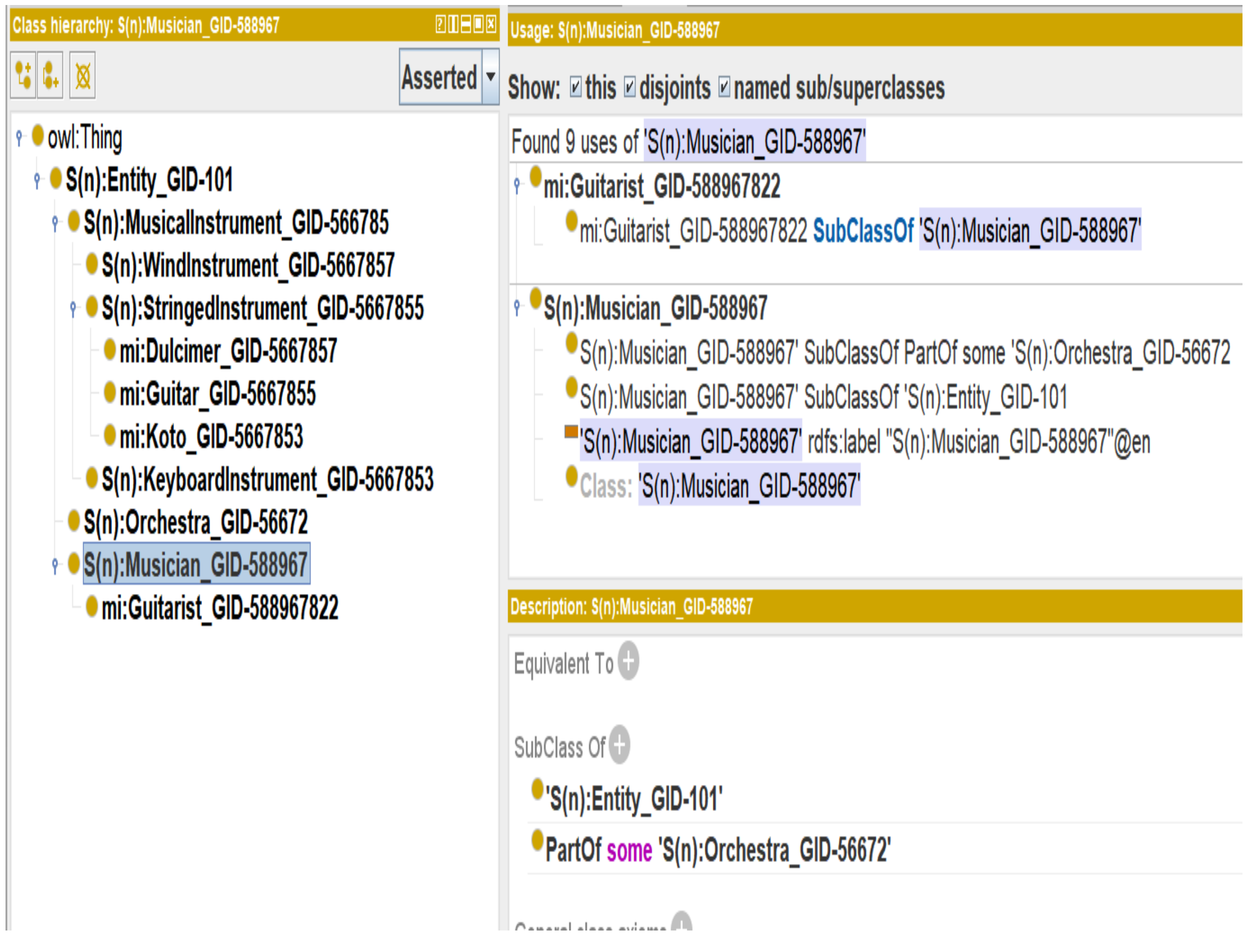
LOD - Protégé (General Example)

Left: concept IS-A hierarchy

- owl:Thing (predefined root)

Right: concept property specification

- Class name (Class)
- IS-A hierarchy dependency (SubClassOf)
- Data property with its codomain (Domain)
- How and in which language a concept is named (rdfs:label)
- some ~ keyword for existential quantification.



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'
```

LOD – Lexicon example (Protégé)

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

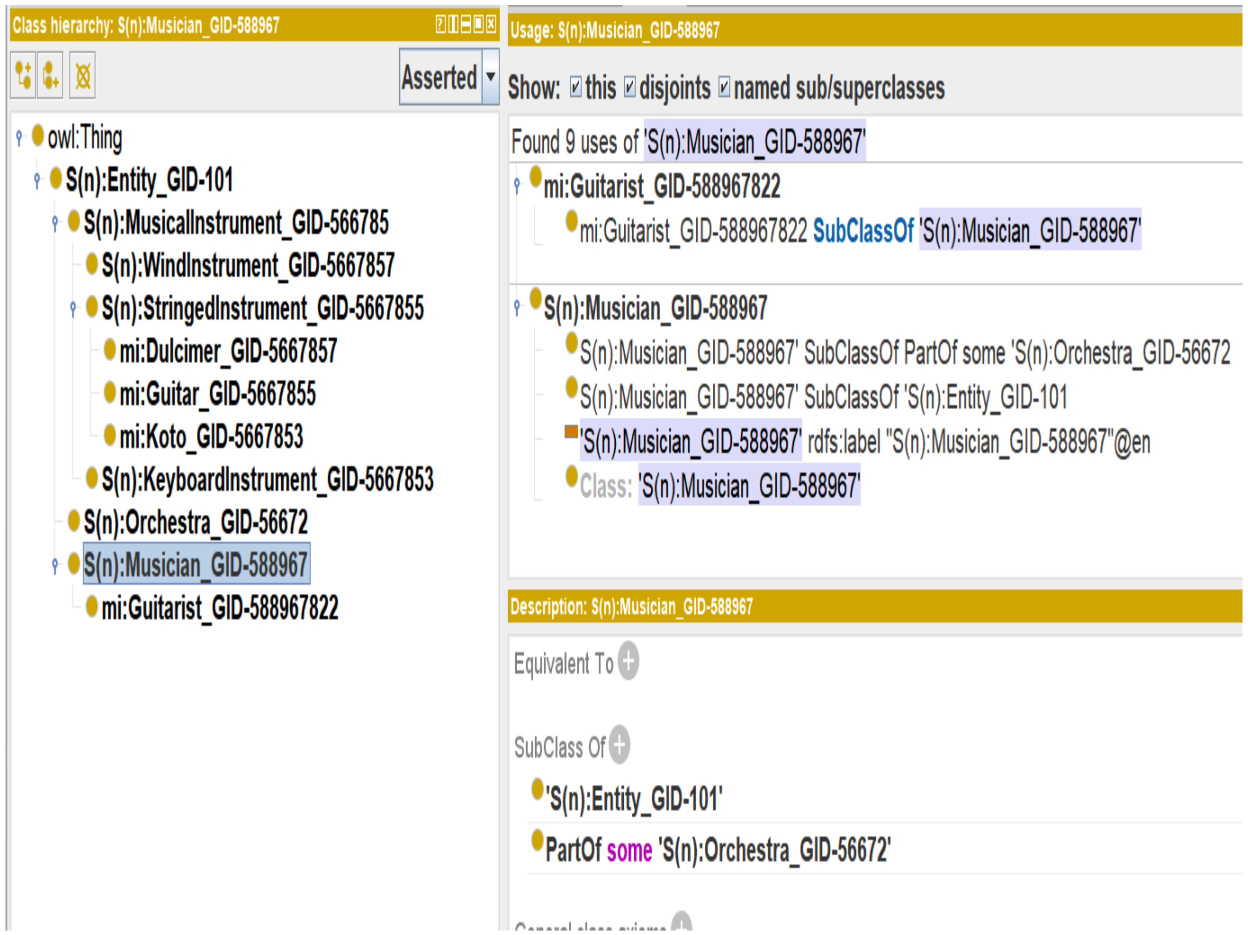
You can see the entire class hierarchy starting from Entity downwards depicting the concepts with their unique GIDs.

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

Observation: Formalization language: OWL/ RDF



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'

LOD theories

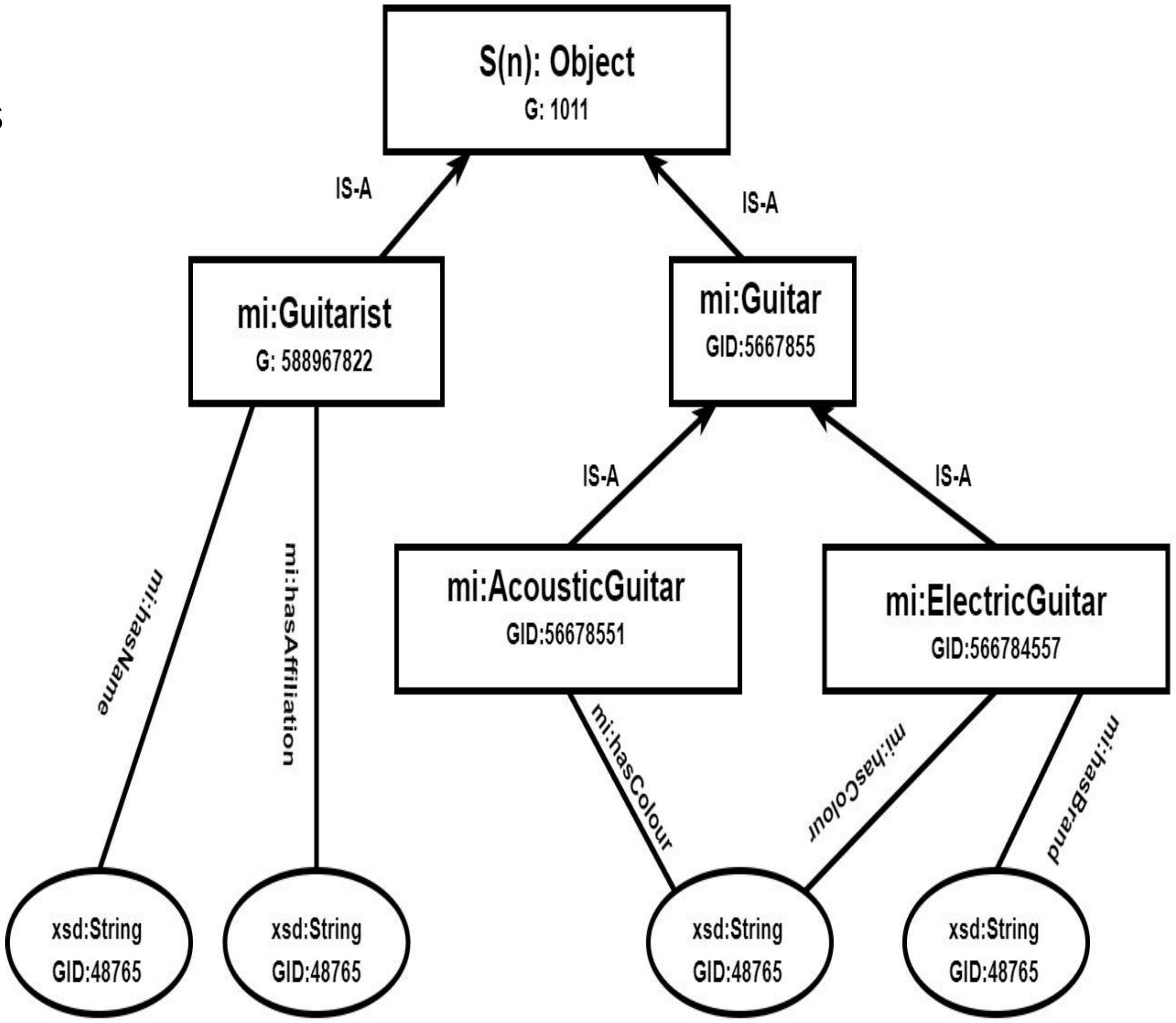
- Lexicons
- Lexical teleontologies
- Knowledge teleontologies
- Teleologies
- Example

Lexical Teleontology

Lexical Teleontologies are specialized lexicons which focus on a purpose-specific (application dependent) part of a lexicon.

Lexical teleontologies are obtained from lexicons by:

- Identifying the root concept, as from the purpose. This is a whole defining the reference space or time containment
- Keeping the relevant concepts among those which are part-of the root in the PART-OF hierarchy
- Keeping the relevant concepts which specialize, via the IS-A and PART-OF hierarchies, the concepts from previous step
- Keeping the part-of relations as needed
- Dropping irrelevant concepts (below/ above the whole)
- Substituting the root concept with most specific concept in the IS-A hierarchy which subsumes all the parts (e.g., from Orchestra to Object



LOD – Lexical teleontology formalization

A lexical teleontology is obtained from a formalized lexicon according to the process described in the previous slide, that is:

- Eliminate the irrelevant elements
- Define the root concept
- Add in the metadata information about the whole being formalized.

LOD - Lexical teleontology Example (Protégé)

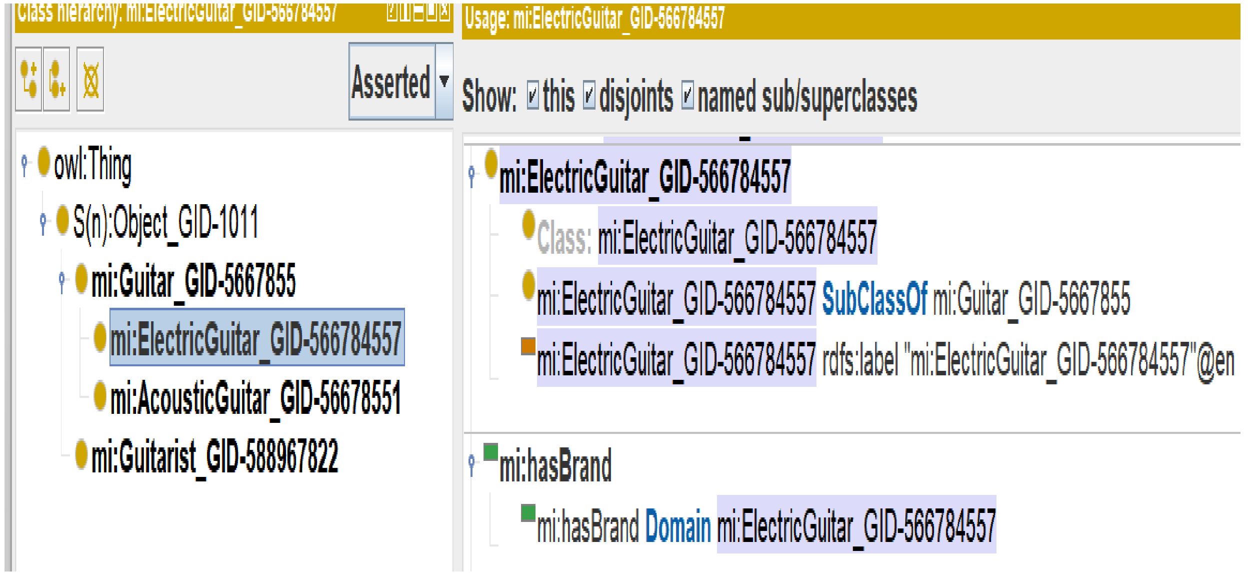
The snippet on the right side shows the lexical teleontology formalized via the Protégé ontology editor.

You can see the entire class hierarchy starting from Object downwards depicting the lexical teleontlogy concepts, with their unique GIDs. Irrelevant concepts such as Koto, Dulcimer, etc. have been eliminated.

You can see (partially) data properties, e.g., ElectricGuitar - mi:hasBrand - String

You can also see (partially) visualization of LOD formalization of lexical teleontology,

e.g., Electric Guitar is: SubClassOf Guitar.



Class hierarchy: mi:ElectricGuitar_GID-566784557

Asserted

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

Usage: mi:ElectricGuitar_GID-566784557

Show: ☑ this ☑ disjoints ☑ named sub/superclasses

- mi:ElectricGuitar_GID-566784557
  - Class: mi:ElectricGuitar_GID-566784557
  - mi:ElectricGuitar_GID-566784557 SubClassOf mi:Guitar_GID-5667855
  - mi:ElectricGuitar_GID-566784557 rdfs:label "mi:ElectricGuitar_GID-566784557"@en
- mi:hasBrand
  - mi:hasBrand Domain mi:ElectricGuitar_GID-566784557

LOD theories

- Lexicons
- Lexical teleontologies
- Knowledge teleontologies
- Teleologies
- Example

Knowledge teleontology

Knowledge Teleontologies extend lexical teleontologies by

- Transforming lexical teleontology concepts (of type string) into etypes by describing them with additional data and object properties
- Transforming data properties with codomain a concept transformed into an etype into an object propert with codomain an etypes
- Adding additional object and data properties which describe the specific aspects which are relevant to the selected context (the whole and parts selected in the lexical teleontology)

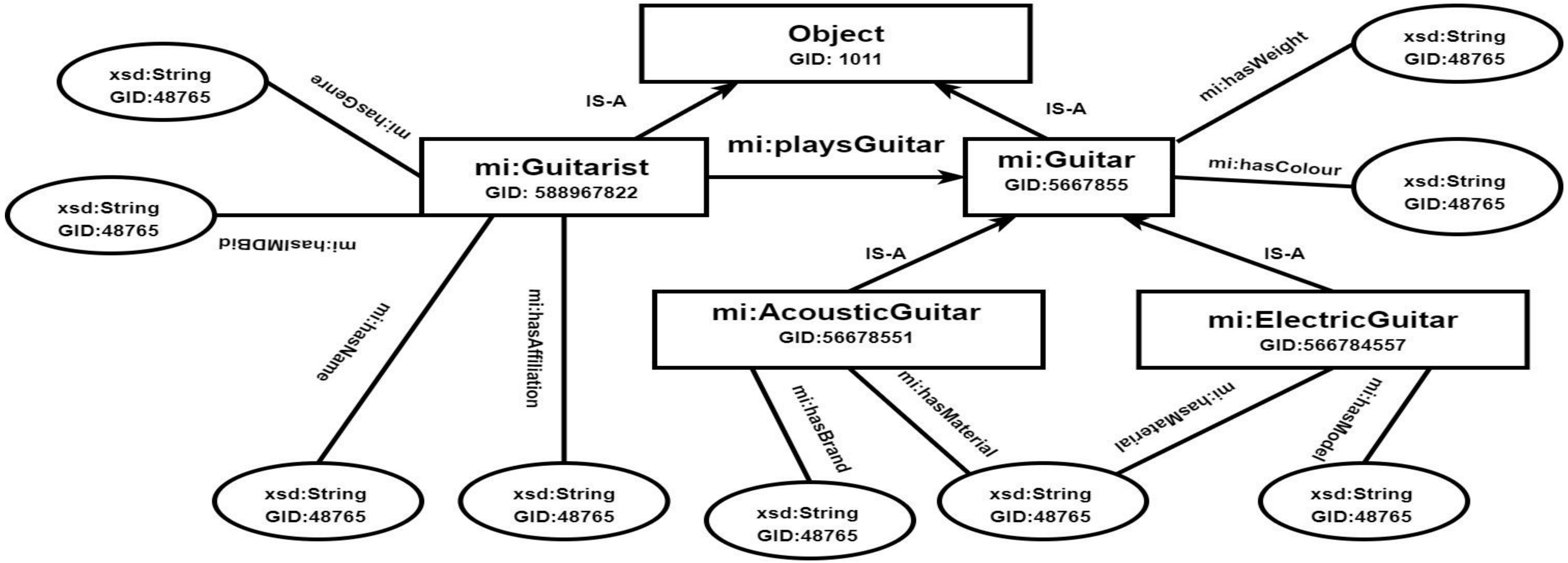
Observation 1: lexical teleontologies/ lexicons define concepts modeling the elements of the world.

Observation 2: Knowledge teleontologies describe concepts, transformed in etypes, by providing relevant local description properties

Observation 3: Knowledge teleontologies "are" formalized EER models

Terminology: We drop the attribute lexical/ knowledge when the context makes clear the meaning

Knowledge teleontology (example)



LOD –teleontology formalization

ISA-Hierarchy: *From the above example of musical instruments lexical teleontology:*

ElectricGuitar $\equiv$ Guitar $\sqcap \forall$hasSoundAmplification.withInputJack *[definition]*

ElectricGuitar $\sqcap$AcousticGuitar $\sqsubseteq \emptyset$

…

Teleontology generation:

ElectricGuitar#1 $\equiv$ ElectricGuitar $\sqcap\exists$hasColour.String $\sqcap\exists$hasBrand.String *[description]*

ElectricGuitar#1 $\sqsubseteq$ ElectricGuitar

Observation 1: A description is a definition enriched with (data and object) properties
Observation 2: A description contains only conjuncts
Observation 3: when building a description no constraints (e.g., no disjointness constraints) on the selected properties
Observation 4: no changes to the PART-OF hierarchy

Teleontology - Example

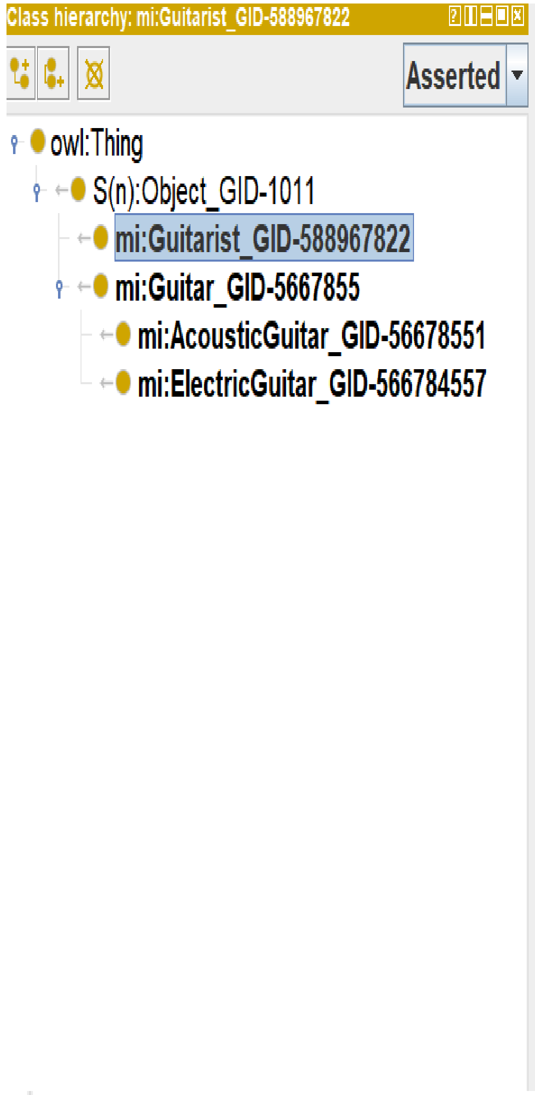
The snippets on the right hand side shows the knowledge teleontology formalized via the Protégé ontology editor.

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.

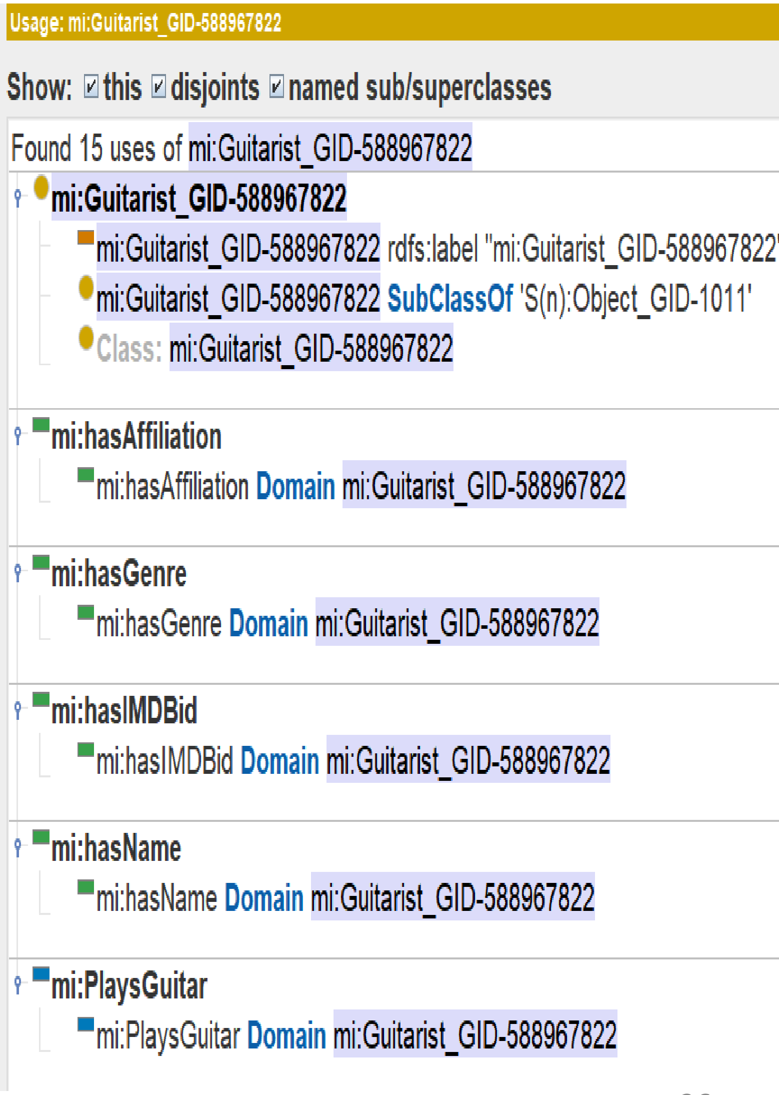
mi:Guitarist - mi:hasIMDBid - xsd:String is a data property-based assertion which indicates that a guitarist has an IMDB id encoded as a string.



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

LOD theories

- Lexicons
- Lexical teleontologies
- Knowledge teleontologies
- Teleologies
- Example

Teleology / ETG

Teleoogies are flattened teleontologies.

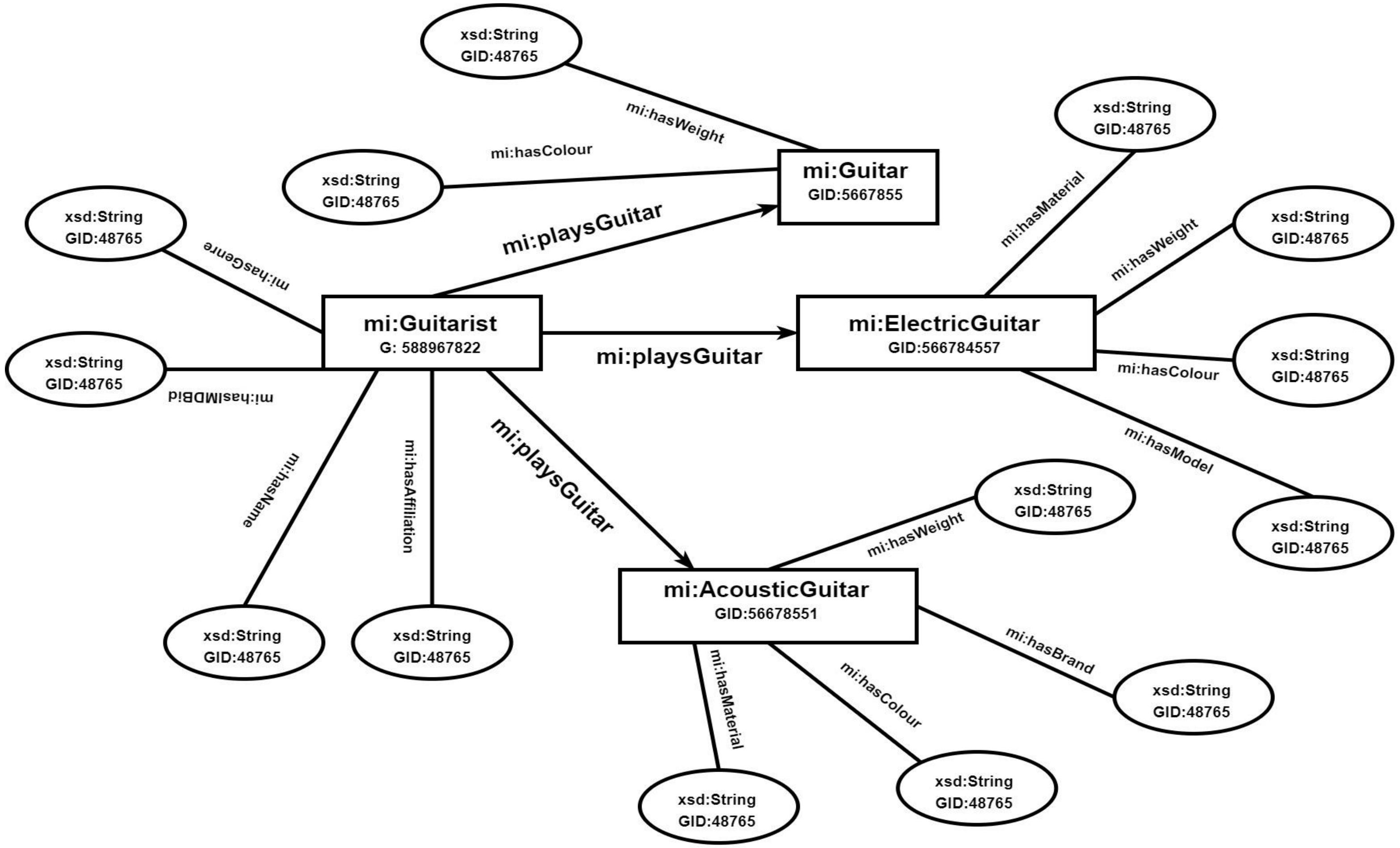
Flattening Process:

- Starting from the root, each concept is defined in terms of the concept one level above in the hierarchy
- Remove the dependence from the more general concept by expanding the definition
- Iterate the process till the leaf nodes

Observation 1: Teleologies are formalizations of ER models

Observation 2: Teleologies are formalisations of ETGs

Teleology


xsd:String GID:48765
mi:hasWeight
mi:Guitar GID:5667855
mi:hasColour
xsd:String GID:48765
mi:playsGuitar
xsd:String GID:48765
mi:hasGenre
mi:Guitarist G: 588967822
xsd:String GID:48765
mi:hasIMDBid
mi:hasName
mi:hasAffiliation
xsd:String GID:48765
xsd:String GID:48765
mi:playsGuitar
mi:ElectricGuitar GID:566784557
mi:playsGuitar
mi:hasMaterial
xsd:String GID:48765
mi:hasWeight
xsd:String GID:48765
mi:hasColour
xsd:String GID:48765
mi:hasModel
xsd:String GID:48765
mi:AcousticGuitar GID:56678551
mi:hasWeight
xsd:String GID:48765
mi:hasBrand
xsd:String GID:48765
mi:hasMaterial
xsd:String GID:48765
mi:hasColour
xsd:String GID:48765

LOD – Teleology formalization

ISA-Hierarchy: From the above example of musical instruments teleontology :

ElectricGuitar#1 $\sqsubseteq$ ElectricGuitar $\sqcap \exists$hasColour.String $\sqcap \exists$hasBrand.String

-- [description = definition enriched with data properties]

AcousticGuitar#1 $\sqsubseteq$ AcousticGuitar $\sqcap \exists$hasMaterial.String $\sqcap \exists$hasModel.String

-- [description = definition enriched with data properties]

Guitarist (leaf in teleology) ≡

Musician (implicit in teleology) $\sqcap \exists$hasName.String $\sqcap \exists$hasAffiliation.String

PART-OF-Hierarchy: From the above example of musical instruments teleontology

Teleology - Example

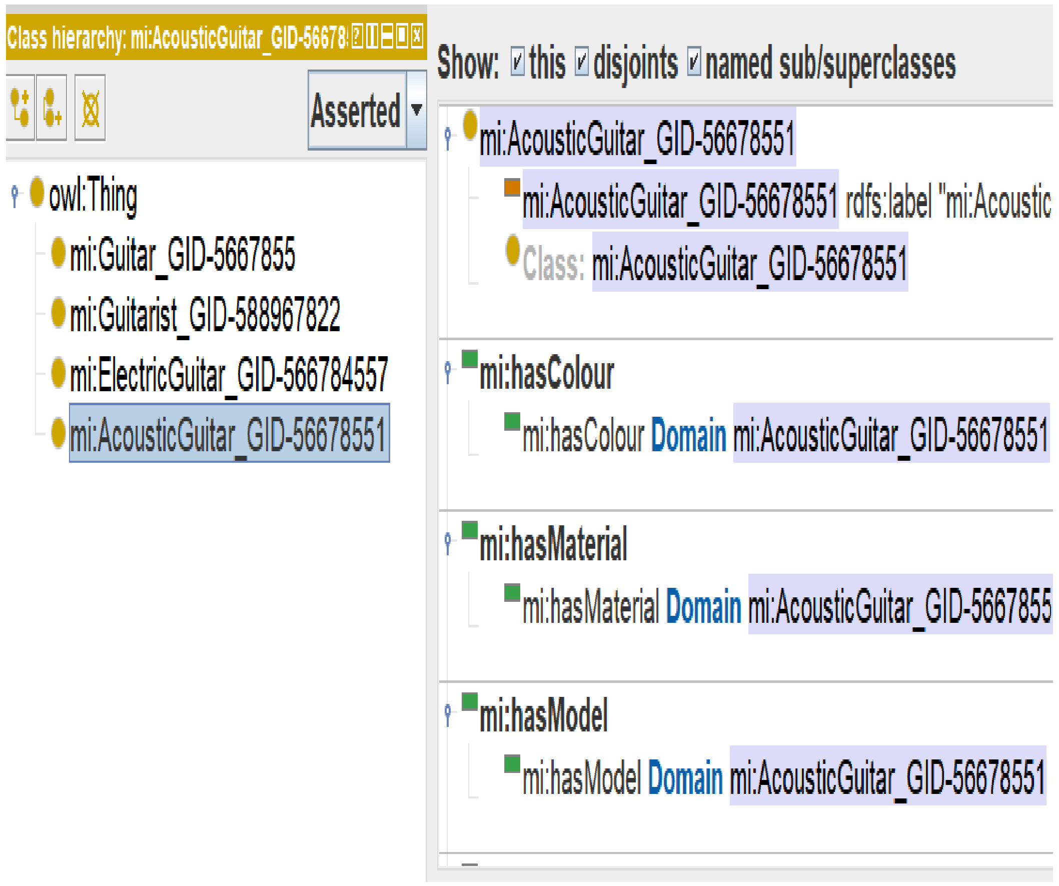
The snippet on the right hand side shows (partially) the teleology formalized via the Protégé ontology editor.

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

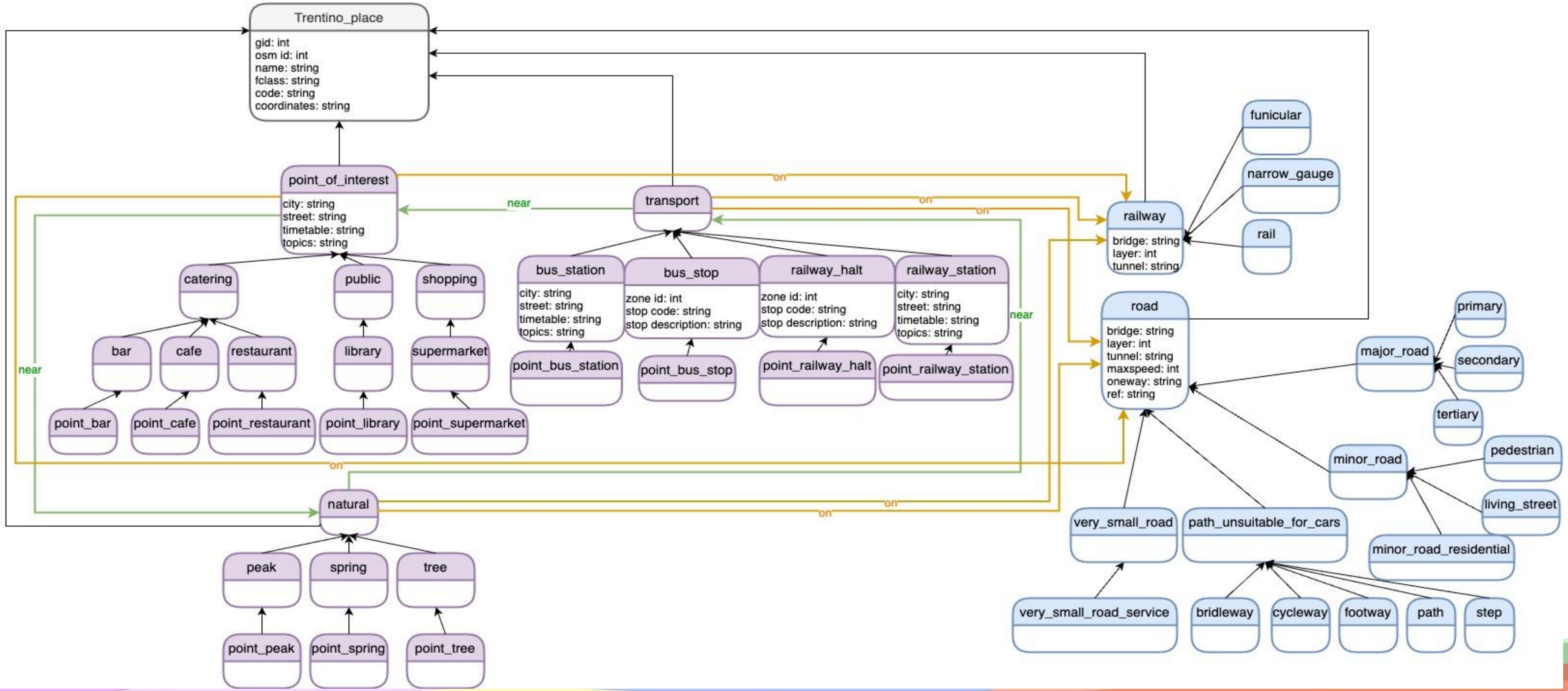
Show: ☑ this ☑ disjoints ☑ named sub/superclasses

- 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

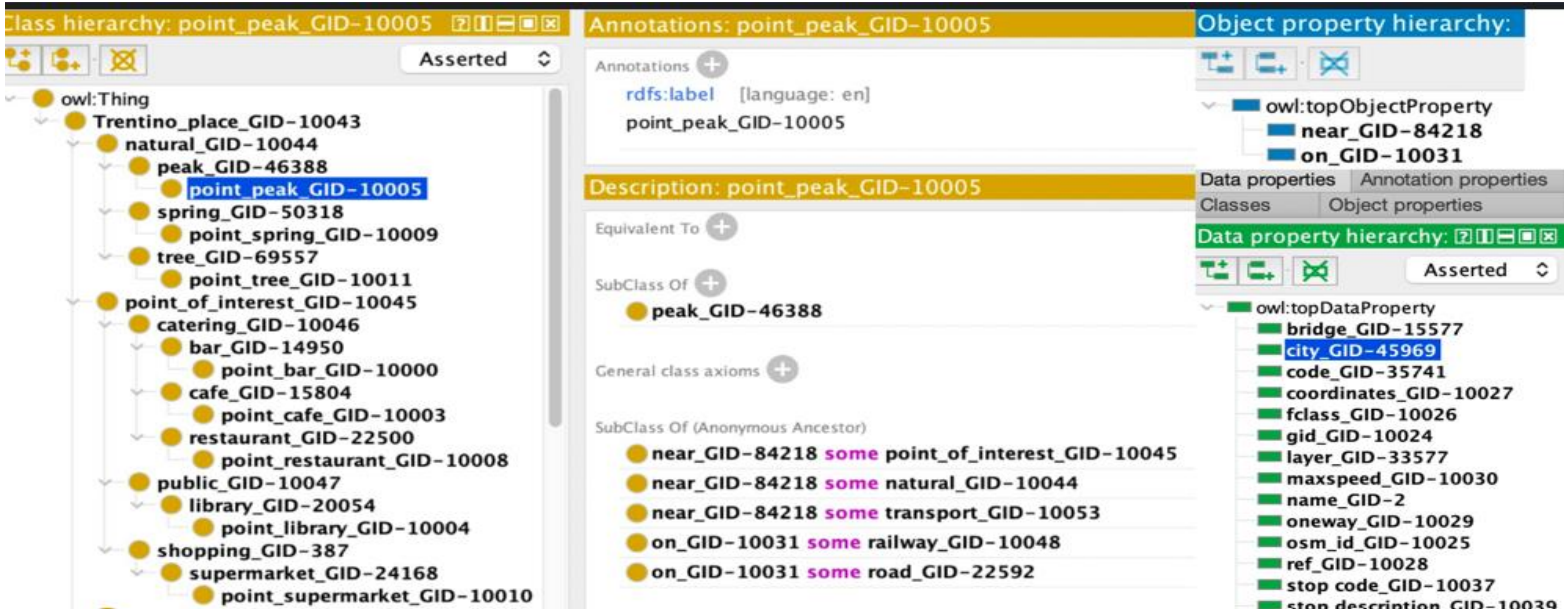
LOD theories

- Lexicons
- Lexical teleontologies
- Knowledge teleontologies
- Teleologies
- Example

EER diagram of Open Street Maps data



Teleontology of Open Street Maps data



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

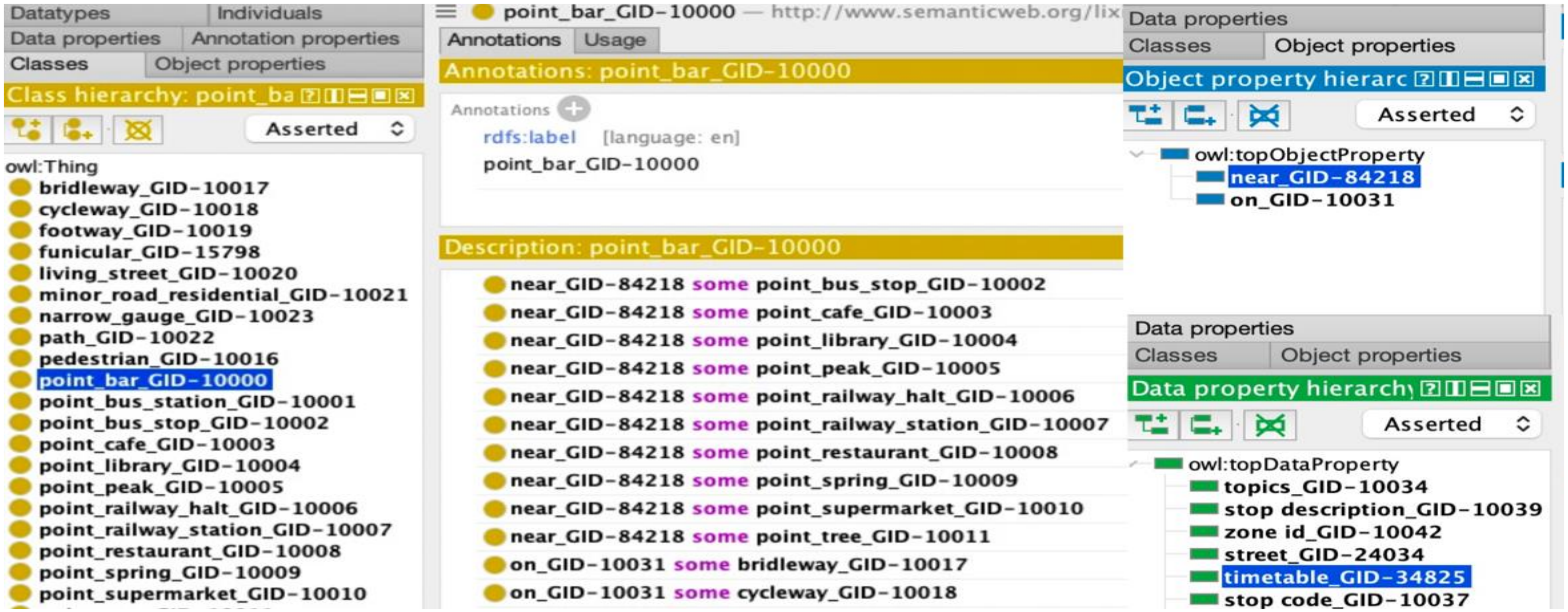
Classes | Object properties

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

LOD - The logic of Descriptions

- TBoxes and terminologies
- LOD theories
- Unfolding

Unfolding a Concept (notion)

Definition (Concept unfolding) A defined concept is unfolded if all the defined concepts occurring in its definiendum are substituted with their definition

Example. From:

ElectricGuitar $\equiv$ Guitar $\sqcap$ $\forall$hasSoundAmplification.withInputJack

ElectricGuitar#1 = ElectricGuitar $\sqcap$ $\exists$hasColour.String $\sqcap$ $\exists$hasBrand.String

To:

ElectricGuitar#1=Guitar $\sqcap$ $\forall$hasSoundAmplification.withInputJack $\sqcap$ $\exists$hasColour.String $\sqcap$ $\exists$hasBrand.String

Remark: In an acyclic terminology the process of concept unfolding can be applied recursively up to any level, with the possibility to primitive concepts (etypes).

Unfolding a TBox (notion)

Definition (TBox unfolding). A definitional TBox T can be unfolded into a Tbox T′ by (recursively) unfolding all its defined concepts.

Observation 1: Let T be an acyclic 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’.

Observation 2: All the reasoning problems for a TBox can be solved in a TBox with only primitive concepts

Observation 3: Teleontologies are nested subsumption hierarchies. Teleologies / ETGs are constructed from teleontologies via unfolding

Unfolding a TBox (example)

$$\mathcal{T} = \begin{cases} Person \equiv \exists hasname.String \sqcap \forall HasJob.Organization \\ Woman \equiv Person \sqcap Female \\ Man \equiv Person \sqcap \neg Woman \\ Mother \equiv Woman \sqcap \exists hasChild.Person \\ Father \equiv Man \sqcap \exists hasChild.Person \\ Parent \equiv Father \sqcup Mother \end{cases}$$

NOTE: Partial (lazy) or full unfolding, depending on the need. Above a partial unfolding

Unfolding a TBox (example)

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

Complexity of unfolding

TBox definitions are like macros that can be expanded into primitive concepts

The size of the unfolded TBox grows polinomially with the depth of the TBox induced subsumption hierarchy. For instance, from

$$A0 \equiv \exists A1 \sqcap \forall A1$$
$$A1 \equiv \exists A2 \sqcap \forall A2$$
$$A2 \equiv \exists A3 \sqcap \forall A3$$

We obtain

$A0 \equiv \exists(\exists(\exists A3 \sqcap \forall A3) \sqcap \forall(\exists A3 \sqcap \forall A3)) \sqcap \forall(\exists(\exists A3 \sqcap \forall A3) \sqcap \forall(\exists A3 \sqcap \forall A3))$ *(times 3)*

$A1 \equiv \exists(\exists A3 \sqcap \forall A3) \sqcap \forall(\exists A3 \sqcap \forall A3)$ *(times 2)*

LOD – Modeling

Some key applications