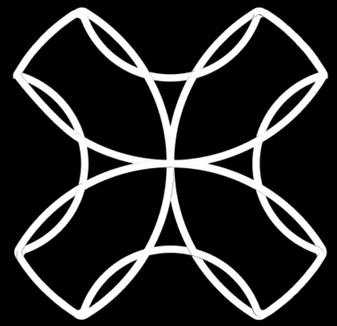


IKNEX

Knowledge Representation and Reasoning

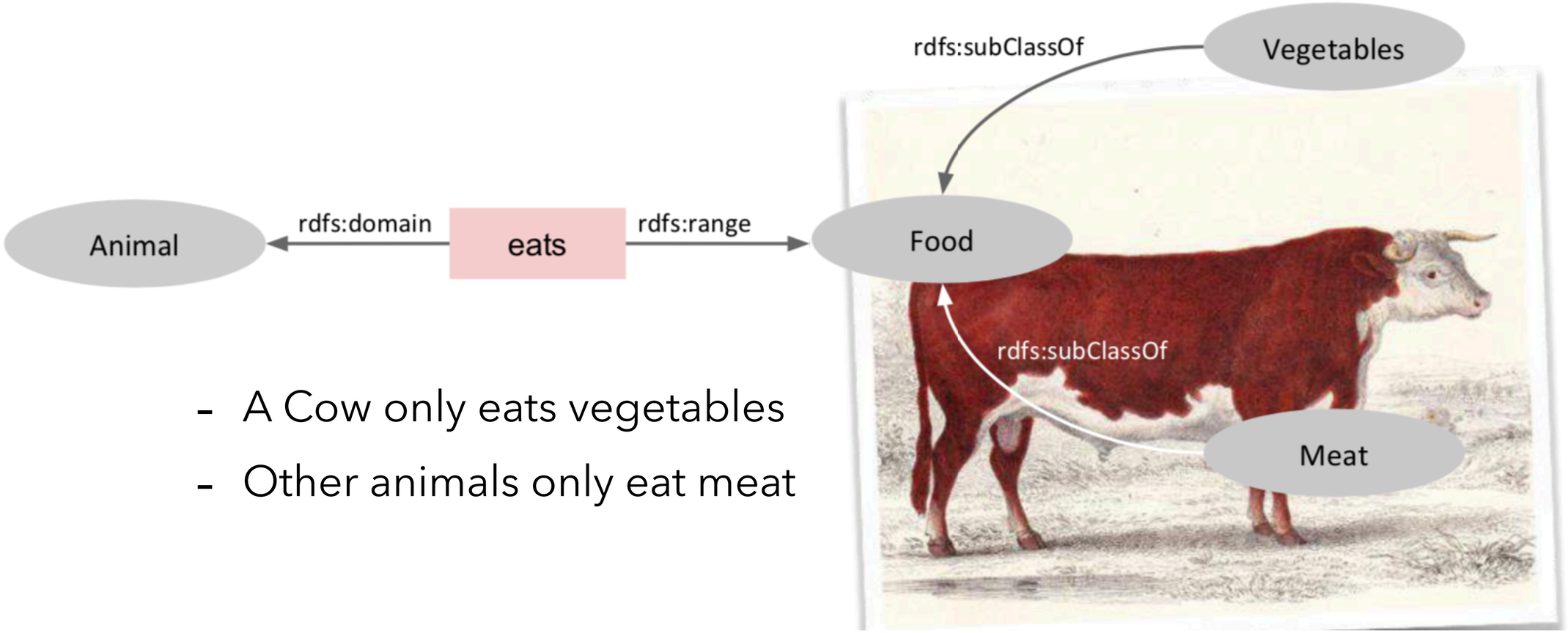
Fall 2023

Dr. Amna Basharat | Amna binte Kamran

Limitations of RDFS

Where RDF(S) Is Not Sufficient…

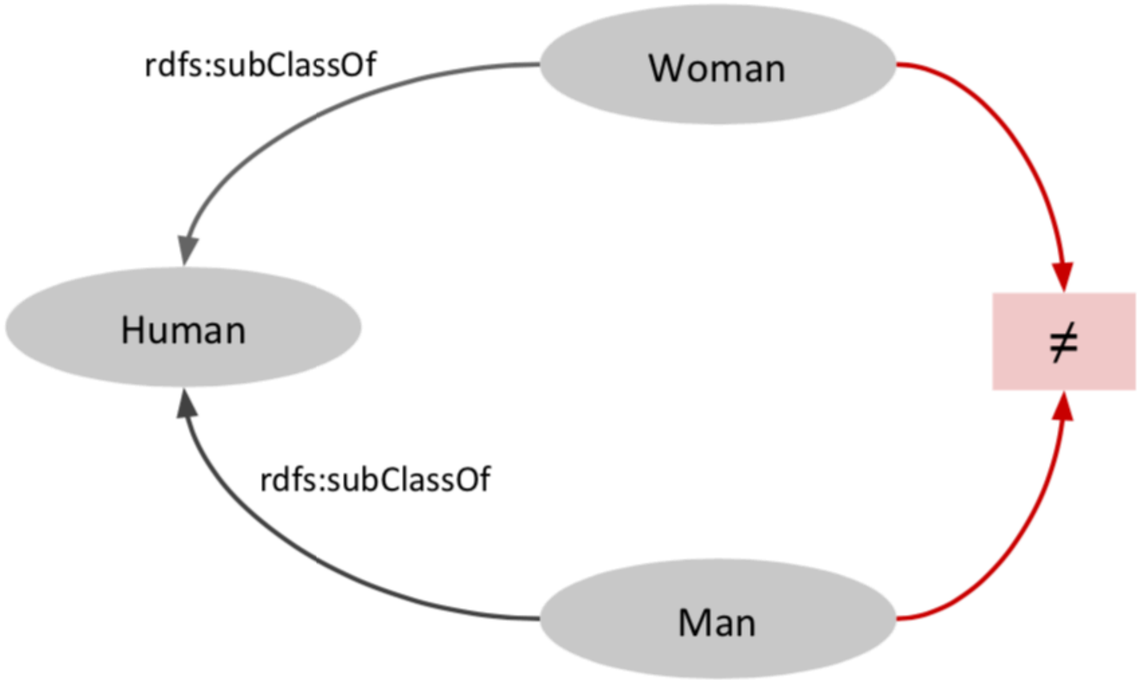
- **Locality of Global Properties**



- A Cow only eats vegetables
- Other animals only eat meat

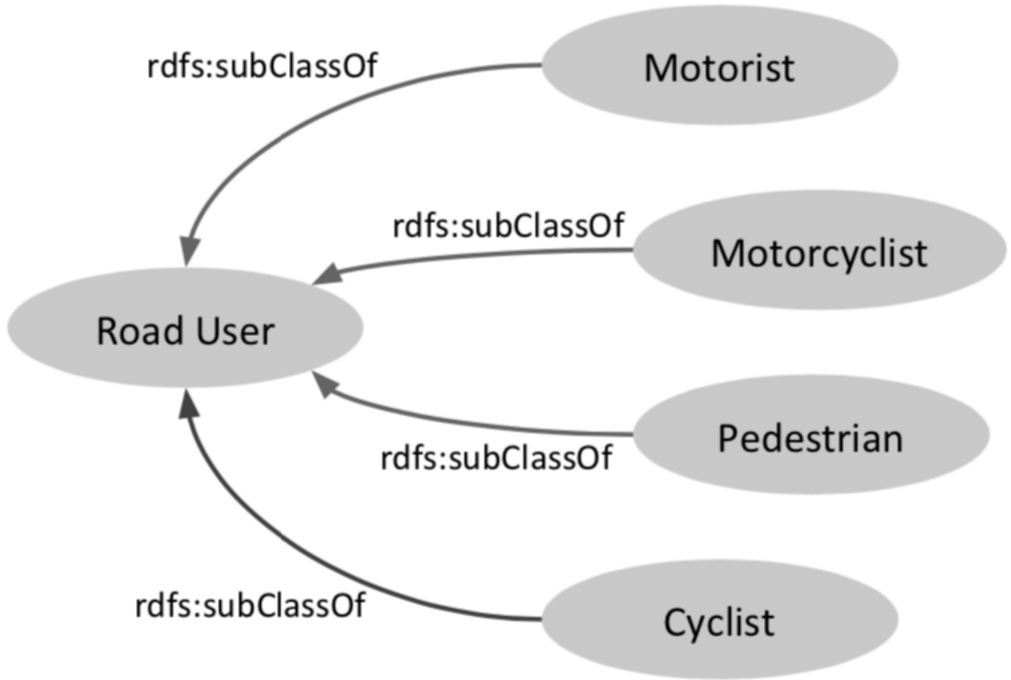
Where RDF(S) Is Not Sufficient…

- **Disjunctive Classes**
  - RDFS Subclass relation cannot express disjunctive class (subclass) membership



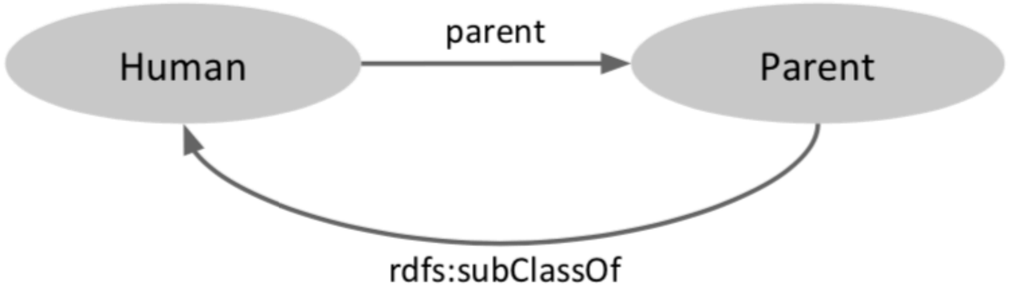
Where RDF(S) Is Not Sufficient…

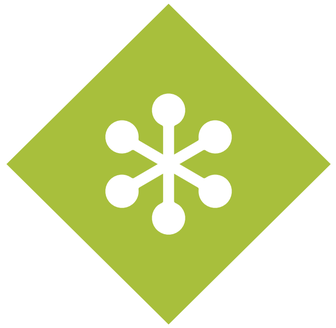
- **Class Combinations**
  - Combination of classes define a new class
  - New class contains only members from given class combinations



Where RDF(S) Is Not Sufficient...

- **Cardinality Constraints**
  - Every human (usually) has two parents

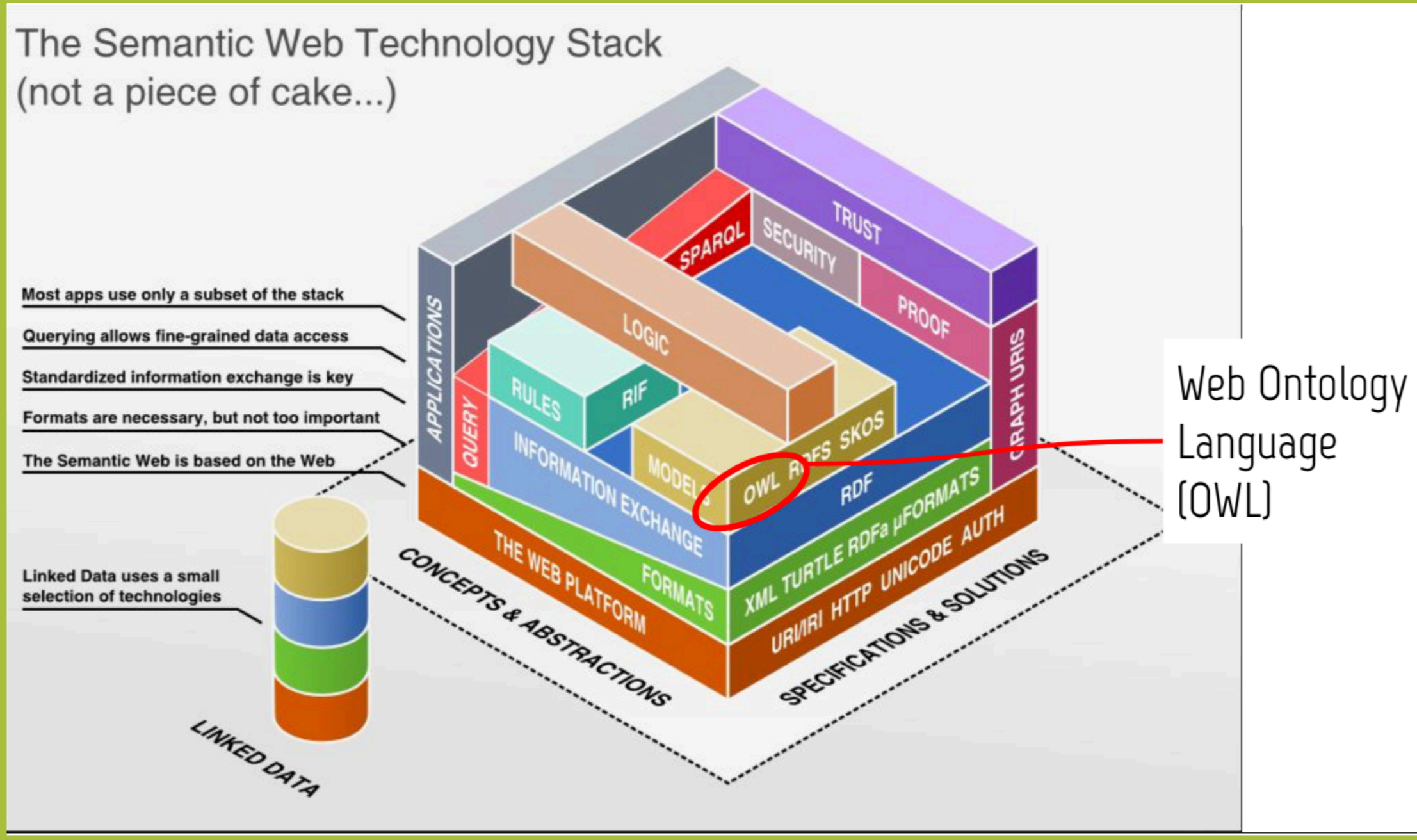




Where RDF(S) Is Not Sufficient…

- **Special Property Constraints**
  - Transitivity (e.g. „is greater than“)
  - Uniqueness (e.g. „is mother of“)
  - Inversiveness (e.g. „is parent of“ and „is child of“)

- **General Problem of RDF(S)**
  - RDF(S) does not have the possibility of **negation**
    - *hpi:harald rdf:type hpi:Vegetarian .*
    - *hpi:harald rdf:type hpi:NonVegetarian .*
    - *…does not automatically generate a contradiction*

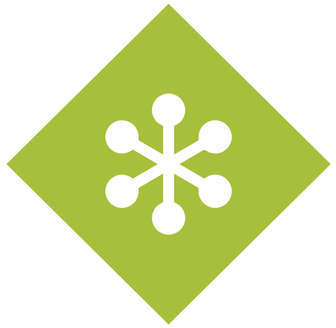


Web Ontology Language (OWL)

What Are Ontologies?

- Definition
- Use
- Components
- Knowledge representation

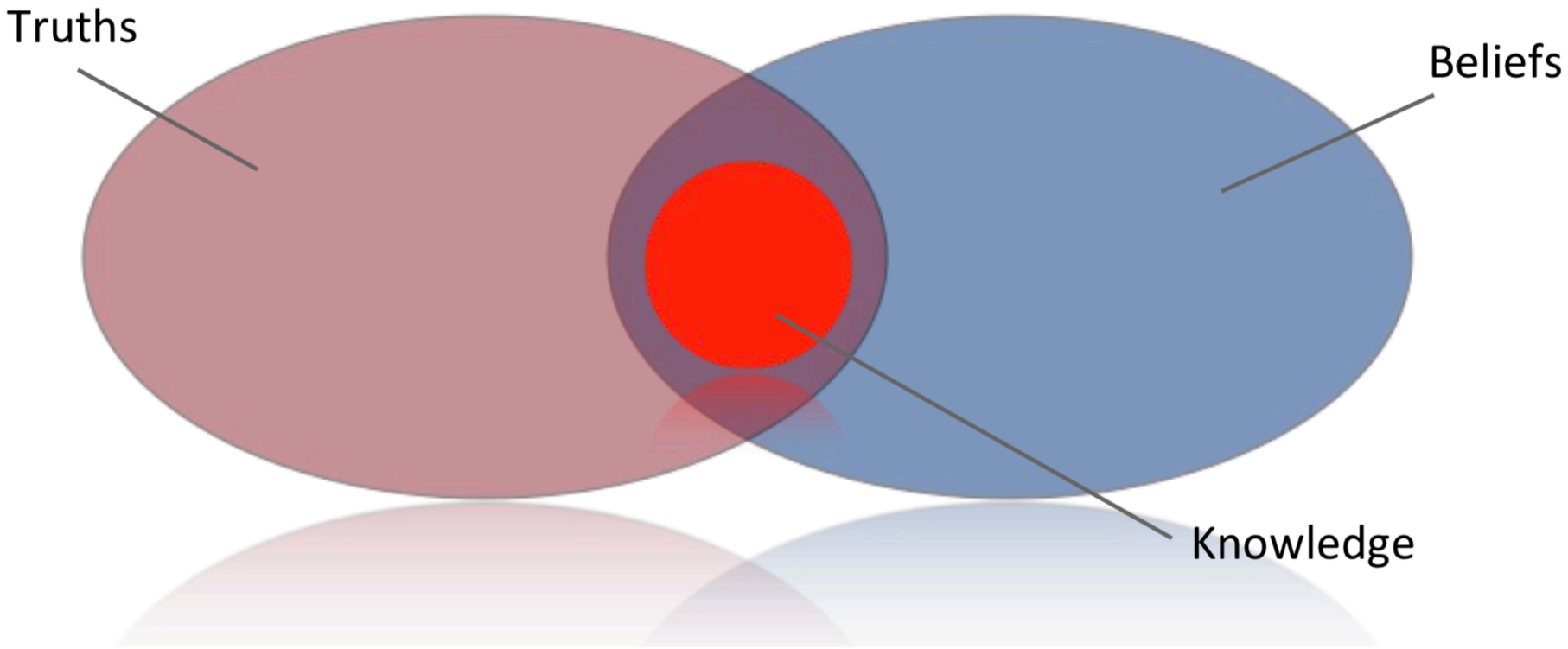

„People can‘t share knowledge if
they don‘t speak a common language“
Thomas Davenport (1997)



...To Speak a Common Language

- common symbols and concepts **(Syntax)**
- agreement about their meaning **(Semantics)**
- classification of concepts **(Taxonomy)**
- associations and relations of concepts **(Thesauri)**
- rules and knowledge about which relations are allowed and make sense **(Ontologies)**

What Is Knowledge?



Traditional Definition: “Knowledge is a subset of all true beliefs”


To represent knowledge,
we need a formal knowledge representation
= ontologies

Ontologies

“Ontologies define 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, Fikes, Finin, Gruber, Senator, Swartout, 1991)

Definitions

- An ontology is an explicit specification of a conceptualization (Gruber)
- 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. (Swartout, Patil, Knight, Russ)
- An ontology provides the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base. (Bernaras, Lasergoiti, Correra)
- An ontology is a formal, explicit specification of a shared conceptualization (Studer, Benjamins, Fensel)

Ontologies on the Semantic Web

- Symbolic models expressed in some formal (logical?) language to allow for automated reasoning

Ontologies Used …

- For communication between people and organizations
- For enabling knowledge reuse and sharing
- As basis for interoperability between systems
- As repository of information
- As query model for information sources
- As vocabularies/schemas for Linked Data

- **Key technology for the Semantic Web**

In Philosophy...

Ontology is the philosophical study of the nature of being, existence, or reality, as well as the basic categories of being and their relations...

Ontologies in Computer Science

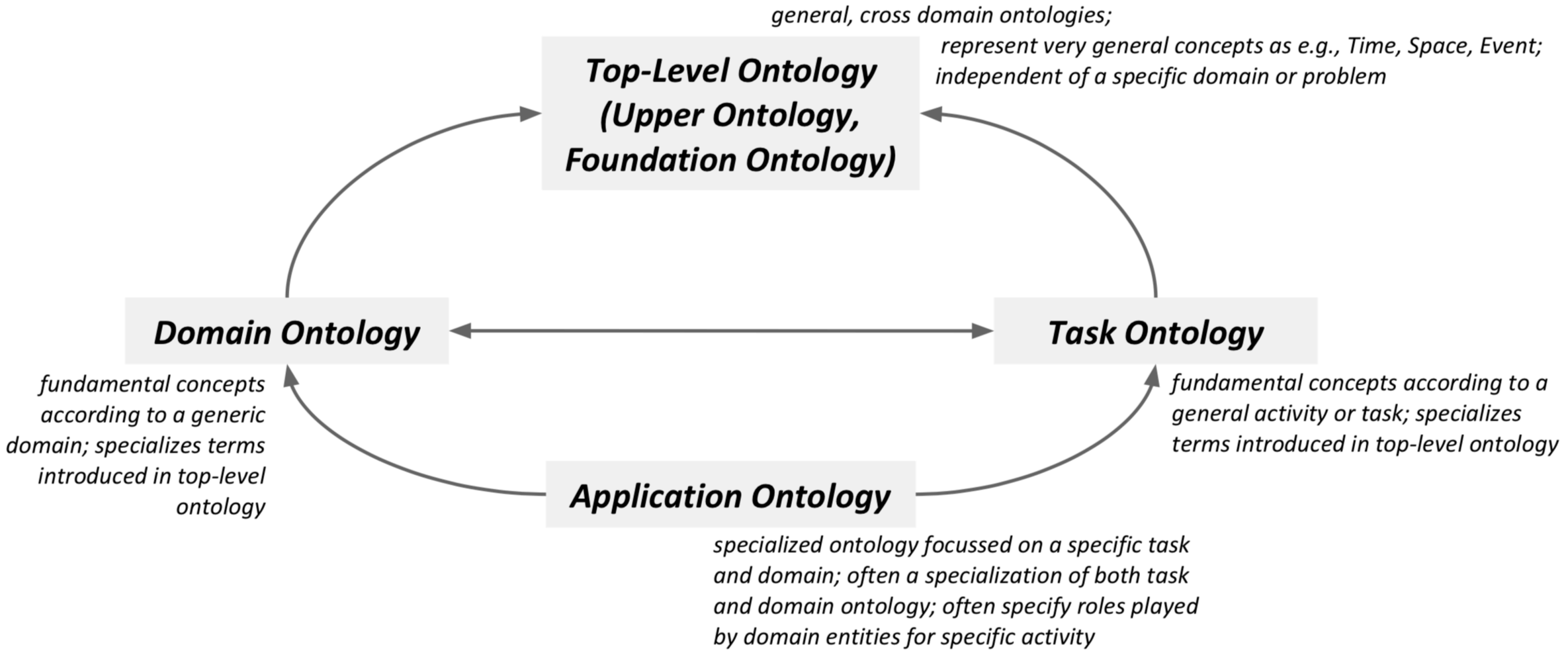
"An ontology is an explicit, formal specification of a shared conceptualization. The term is borrowed from philosophy, where an Ontology is a systematic account of existence. For AI systems, what 'exists' is that which can be represented."

according to Thomas R. Gruber: A Translation Approach to Portable Ontology Specifications.Knowledge Acquisition, 5(2):199-220, 1993.

conceptualization:	abstract model (domain, identified relevant concepts, relations)
explicit:	meaning of all concepts must be defined
formal:	machine understandable
shared:	consensus about ontology

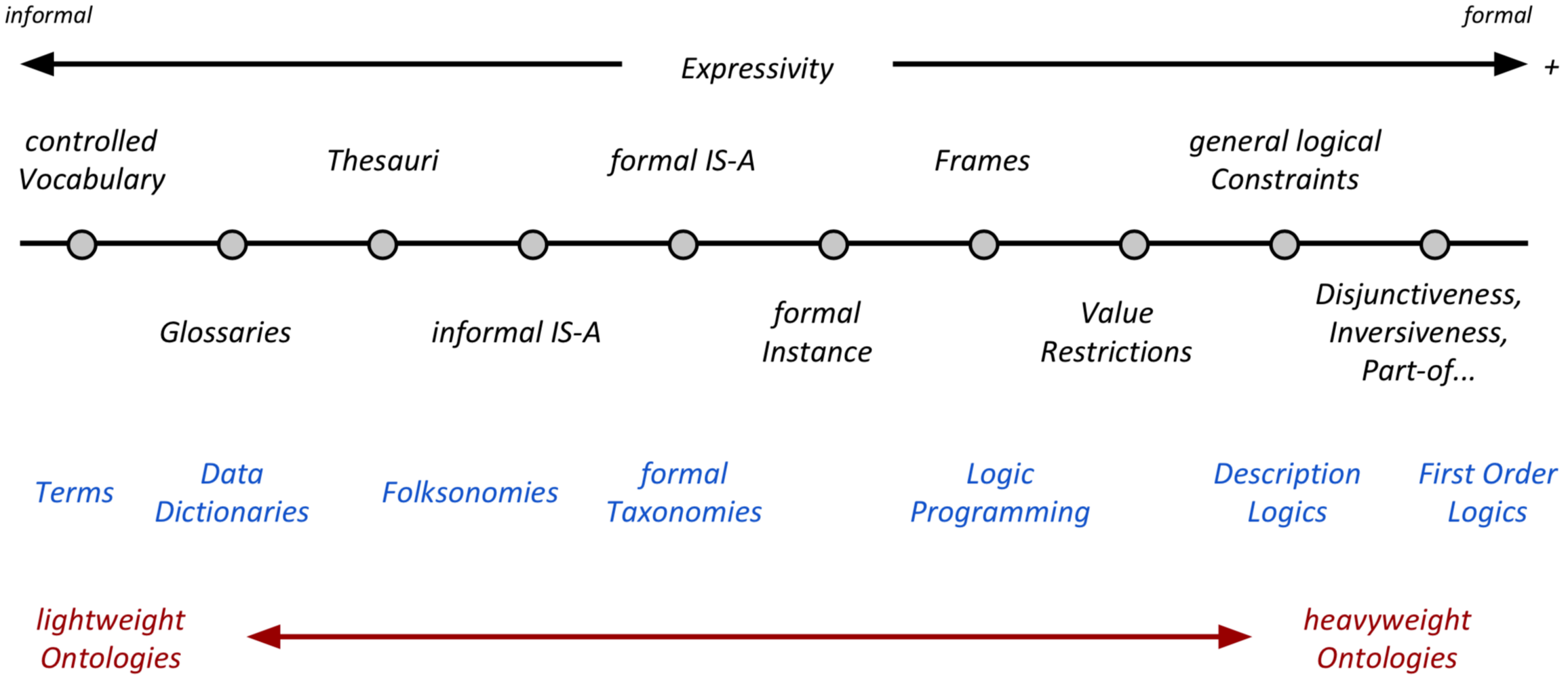
Types of Ontologies

Ontology Types and Categories (According to Their Level of Generality)



(according to Guarino: Formal Ontology in Information Systems, 1998)

Ontology Types and Categories (According to Their Level of Semantic Expressivity)



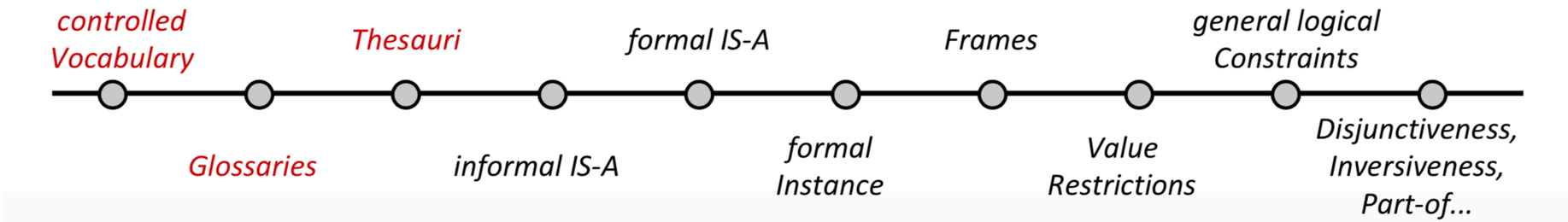
(according to Lassila and McGuiness:The Role of Frame-Based Representation on the Semantic Web, 2001

Ontology Types and Categories (According to Their Level of Semantic Expressivity)

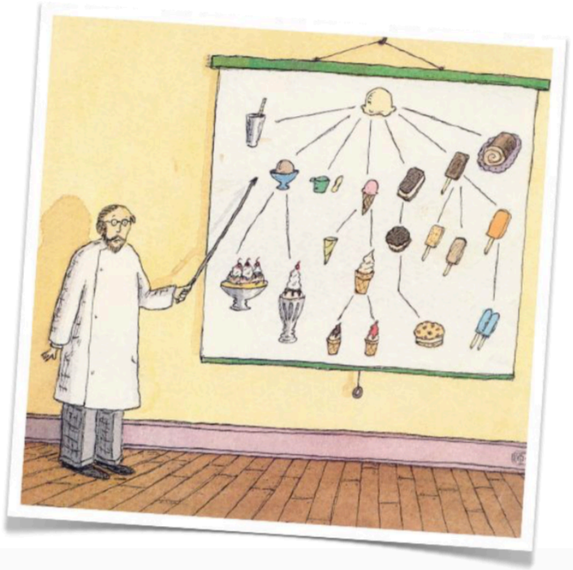
- **Controlled Vocabulary**: finite list of terms (e.g. catalogue)
- **Glossary**: finite list of terms including an informal definition in natural language
- **Thesauri**: [greek. “treasure, treasure house”]

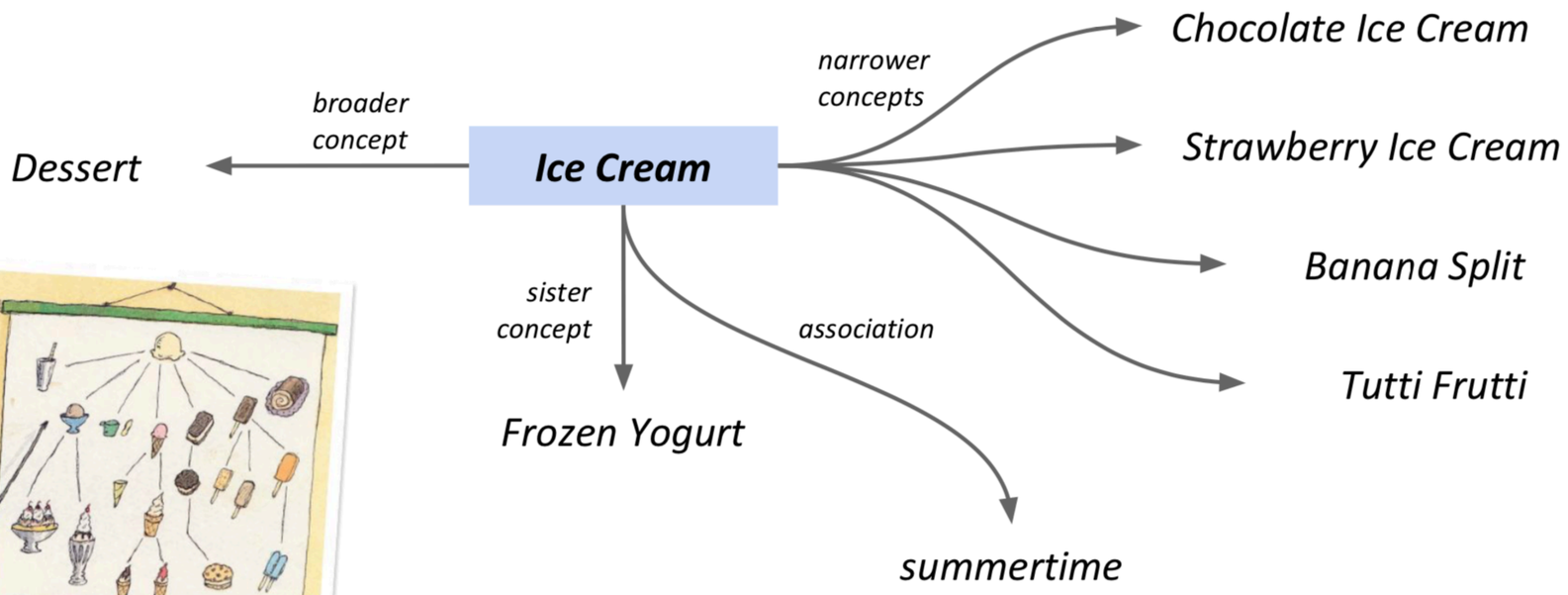
Controlled vocabulary, concepts are connected via relations.

- Equivalency (synonyms)
- Hierarchies (subclasses, superclasses)
- Homographs (Homonyms)
- Associations (similar concepts)

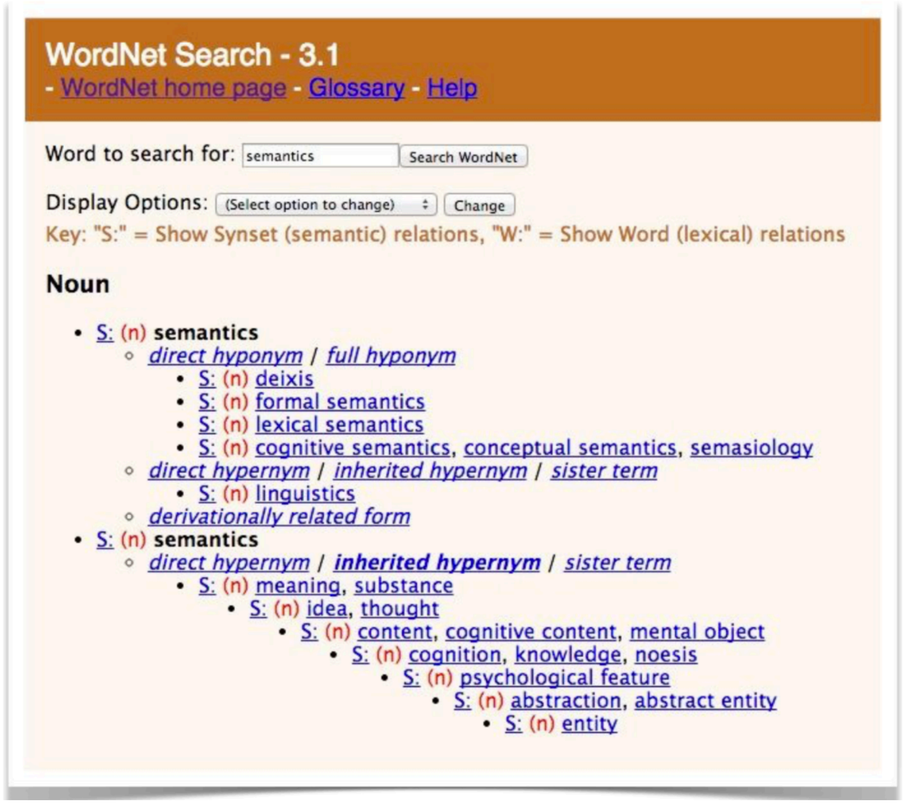


Example of an Thesaurus Entry





WordNet



WordNet

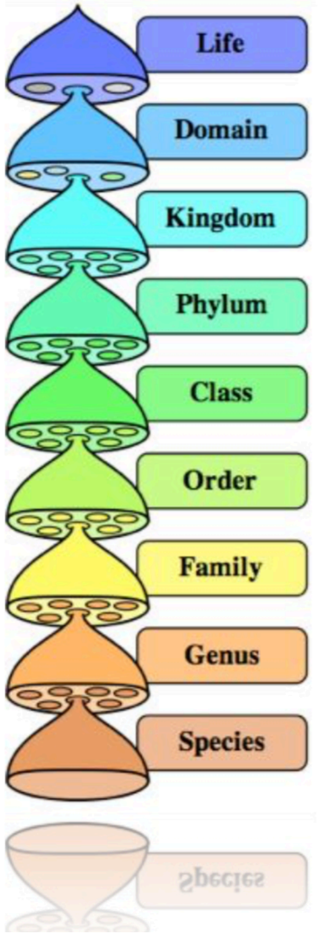
- link-based electronic dictionary with semantic relations
- organized in 117587 „Synsets“,ordered by
  - Nouns (N)
  - Verbs (V)
  - Adjectives (Adj)
  - Adverbs (Av)

- http://wordnetweb.princeton.edu/perl/webwn

Taxonomies

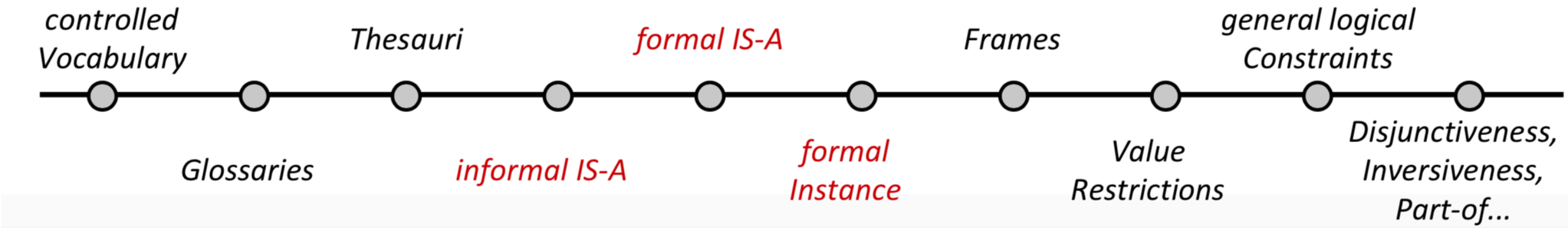
Taxonomy: Definition of a hierarchical system of groups (from [greek] *τάξις* (*taxis)* = order, arrangement and *νόμος* (*nomos)* = law, science) ...

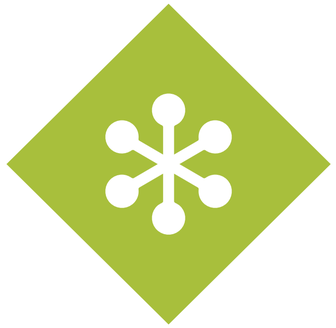
- Also **classification schema**, nomenclature,…
- In science most times classification into (mono-) hierarchical sets (classes, subclasses, …)
- (also) subject of **biology**:
  - The arrangement of organisms into a classification according to simlarities.



Taxonomies

- **Informal IS-A Hierarchy:**
  - Explicit hierarchy of classes, subclass relations are not strict ()e.g. index of a library)
- **Formal IS-A Hierarchy:**
  - Explicit Hierarchy of classes, subclass relations are strict
- **Formal instance:**
  - Explicit Class hierarchy, besides strict subclass relations also instance-of relations are allowed.





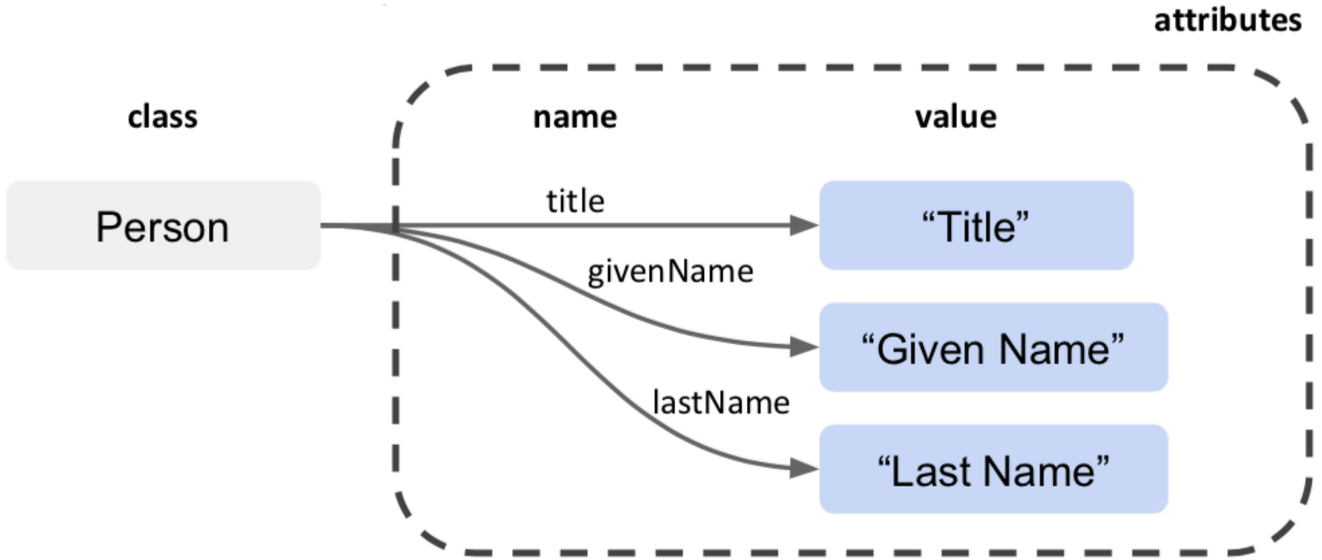
Different Kinds of "Ontologies"

- Controlled vocabularies, Concepts
- Taxonomies Concepts, is-a
- Thesauri
  Concepts, predefined relations
- Data models (e.g. EER, UML) Concepts, relations, axioms
- Logic-based ontologies Concepts, relations, axioms

How To Represent Ontologies

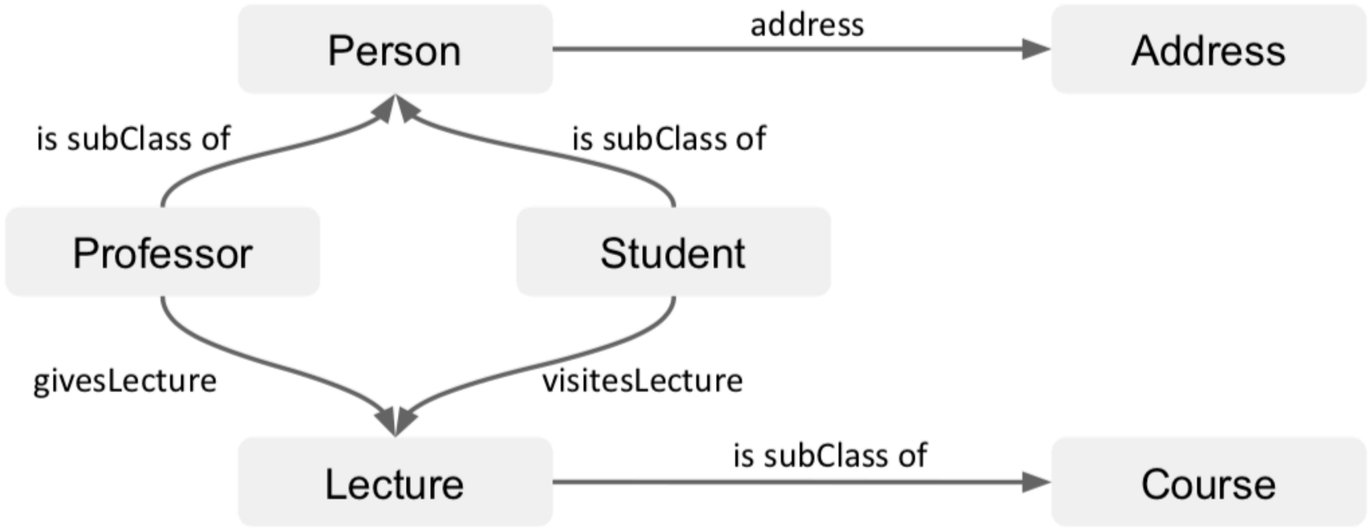
How To Represent Ontologies

- Ontologies can be represented via Classes, Relations and Instances
- Classes are abstract groups, sets, or collections of objects and represent ontology concepts
- Classes are characterised via attributes
- Attributes are name-value pairs



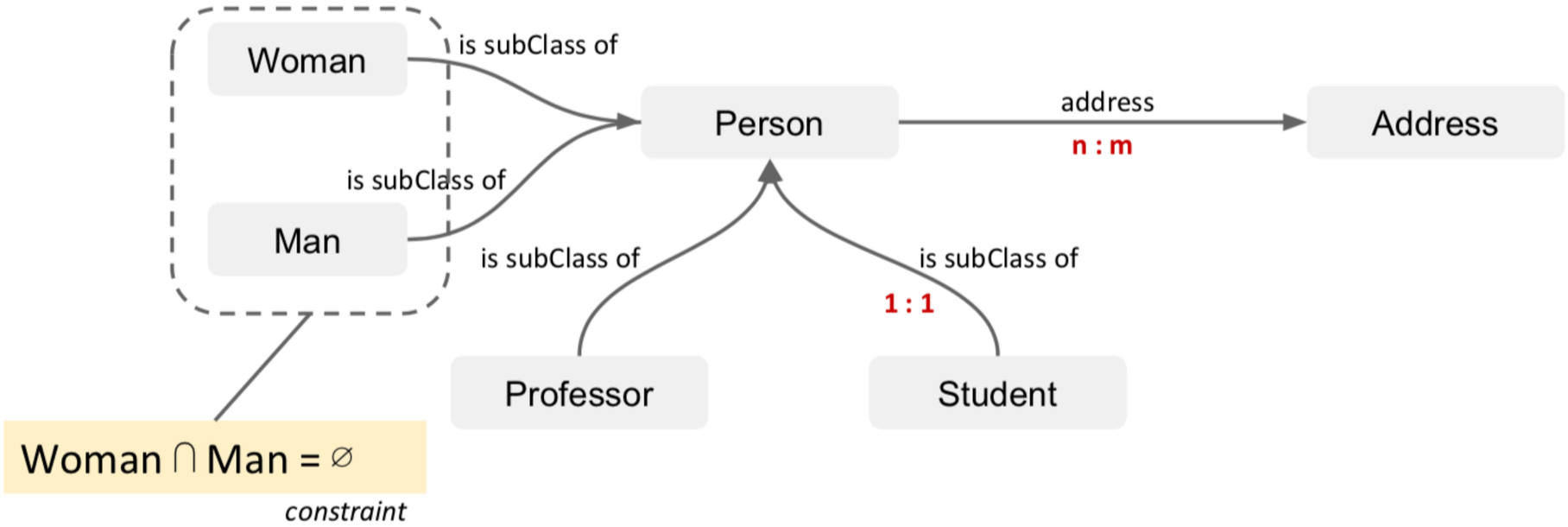
How To Represent Ontologies

- Classes can be related to other classes
  Relations are special attributes, whose values are objects of (other) classes



How To Represent Ontologies

- For Relations and Attributes Rules (Constraints) can be defined that determine allowed/valid values



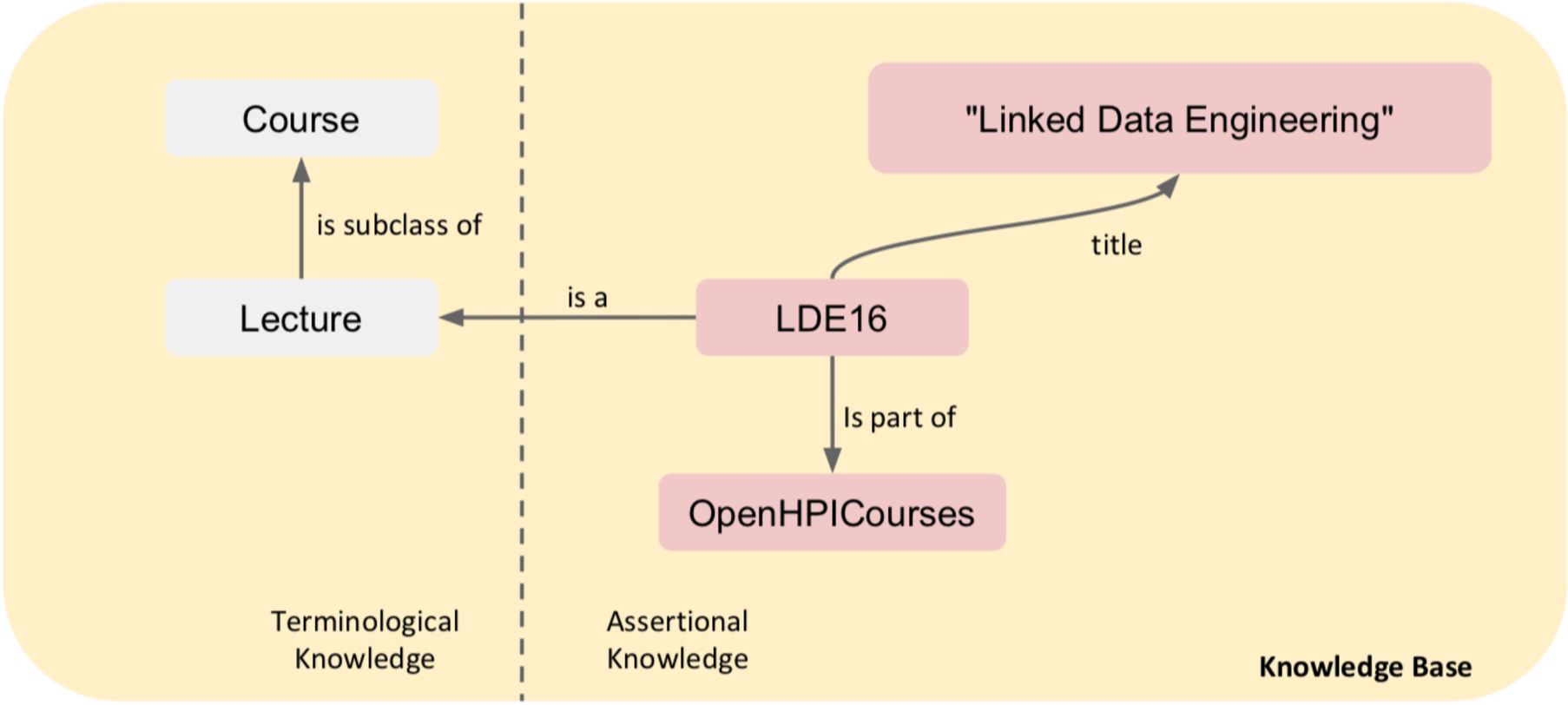
How To Represent Ontologies

- Classes, relations, and constraints can be combined to form (complex) Statements / Assertions
- Special Case: formal Axioms
- Example:
  „it is not possible to attend two courses at the same time“

- Axioms describe knowledge that cannot be expressed simply with the help of other existing components.

How To Represent Ontologies

- Instances describe individuals of an ontology



Understanding Vocabularies

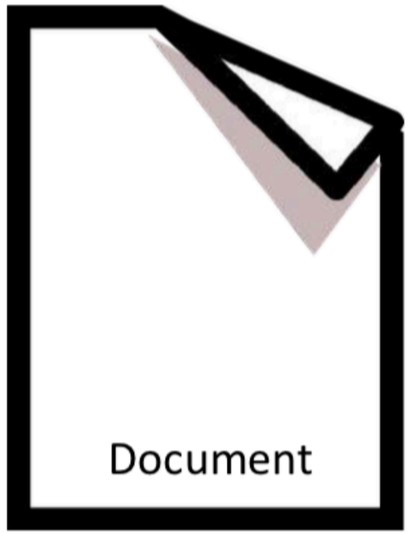
Vocabularies

- A person's vocabulary is the set of words within a language that are familiar to that person. (Wikipedia)
- = “all the words known and used by a particular person” (Cambridge Advanced Learners Dictionary)

- On the Semantic Web, vocabularies define the concepts and relationships used to describe and represent an area of concern. (W3C)
- Vocabularies are used to
  - classify the terms that can be used in a particular application,
  - characterize possible relationships, and
  - define possible constraints on using those terms.

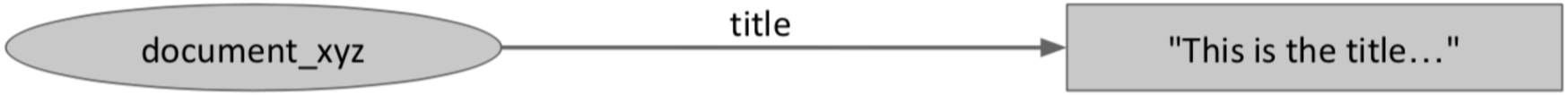
Example

- The Dublin Core Vocabulary (Dublin Core Metadata Element Set)



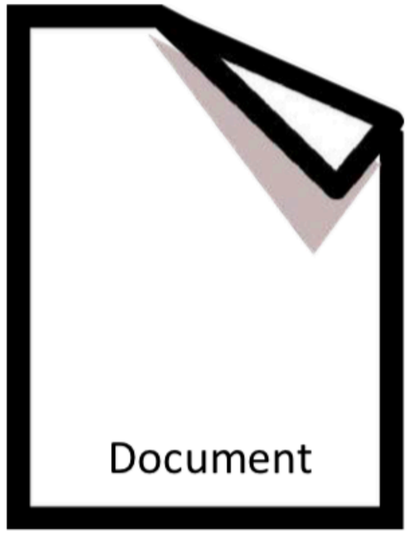
- Title
- Creator
- Subject
- Description
- Publisher
- Contributor
- Date

- Type
- Format
- Identifier
- Source
- Language
- Relation
- Coverage
- Rights



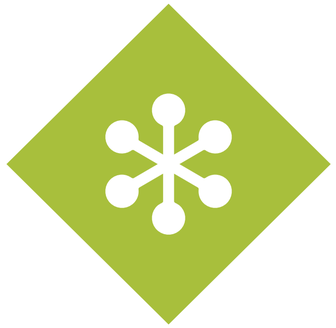
Example

- The Dublin Core Vocabulary (Dublin Core Metadata Element Set)



- Title
- Creator
- Subject
- Description
- Publisher
- Contributor
- Date
- Type
- Format
- Identifier
- Source
- Language
- Relation
- Coverage
- Rights

```
PREFIX dct: <http://purl.org/dc/terms/> .

<http://example.org/document_xyz> dct:title "This is the title…"@en .
```



Example

- Defining personal data for a user/person
  - e.g. given name, last name, address data, web page, email address, etc.
- There are several possibilities:
  - Define a new vocabulary according to your needs
  - Use existing vocabularies and (possibly) adapt your application accordingly
  - Use existing vocabularies and extend vocabularies according to your needs

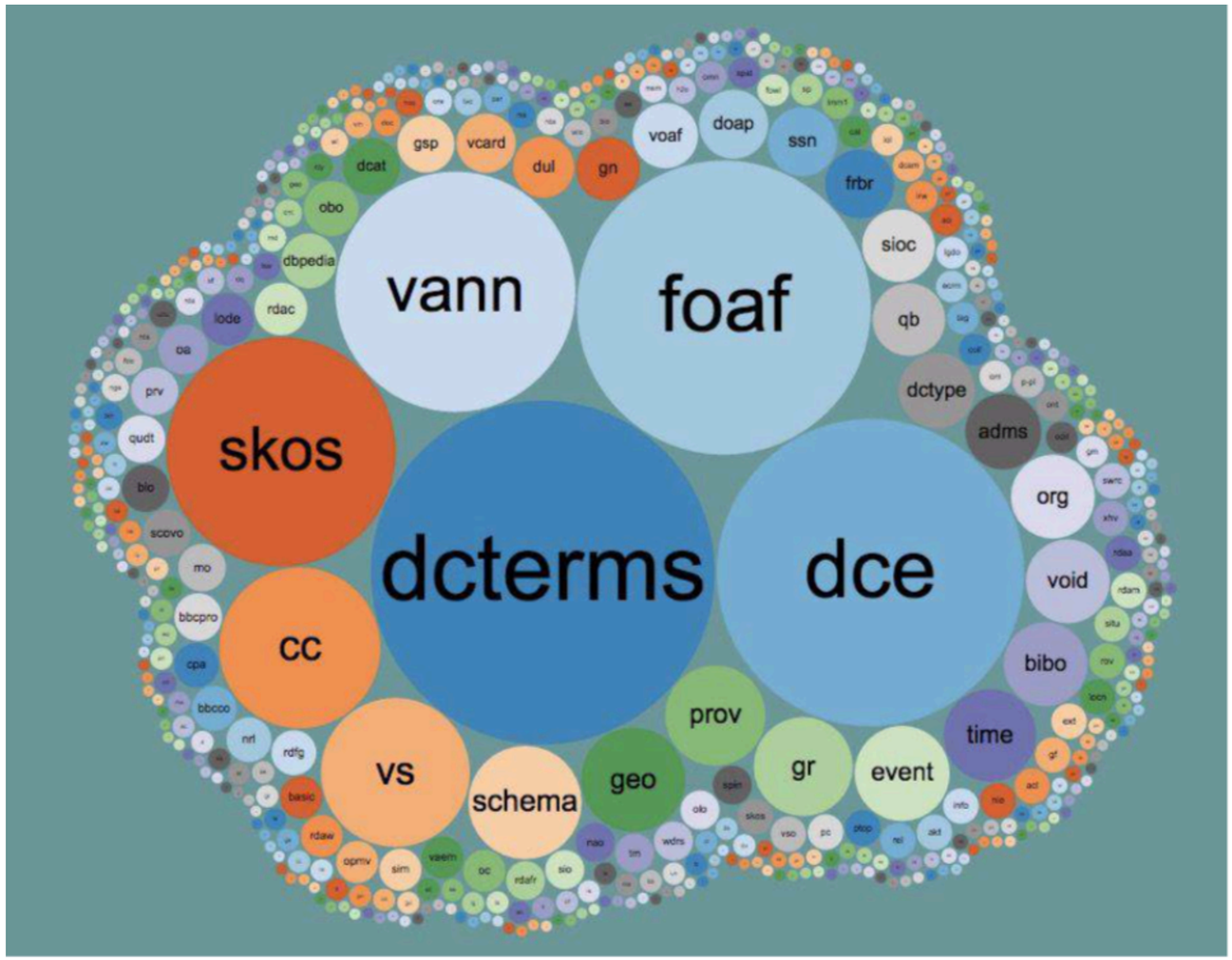
Example of Linked Data Vocabularies

- FOAF - Friend of a Friend
  - describes persons, their activities and their relations to other people and objects

http://www.foaf-project.org

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<#me>
   a foaf:Person ;
   foaf:name "Harald Sack" ;
   foaf:mbox <mailto:harald.sack@hpi.de> ;
   foaf:homepage <https://hpi.de/meinel/lehrstuhl/team-fotos/senior-researcher/sack.html> ;
   foaf:depiction <https://hpi.de/fileadmin/_migrated/pics/harald_min.jpg> ;
   foaf:interest <http://linkeddata.org> ;
   foaf:knows [
      a foaf:Person ;
      foaf:name "Magnus Knuth"
   ] .
```

Where To Find Linked Data Vocabularies



- https://lov.linkeddata.es/dataset/lov/

Web Ontology Language - OWL

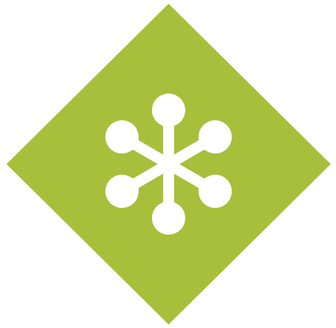
Ontologies and Logic

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

acc. to Thomas R. Gruber: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2):199-220, 1993.

formal (machine readable) semantics

mathematical logic



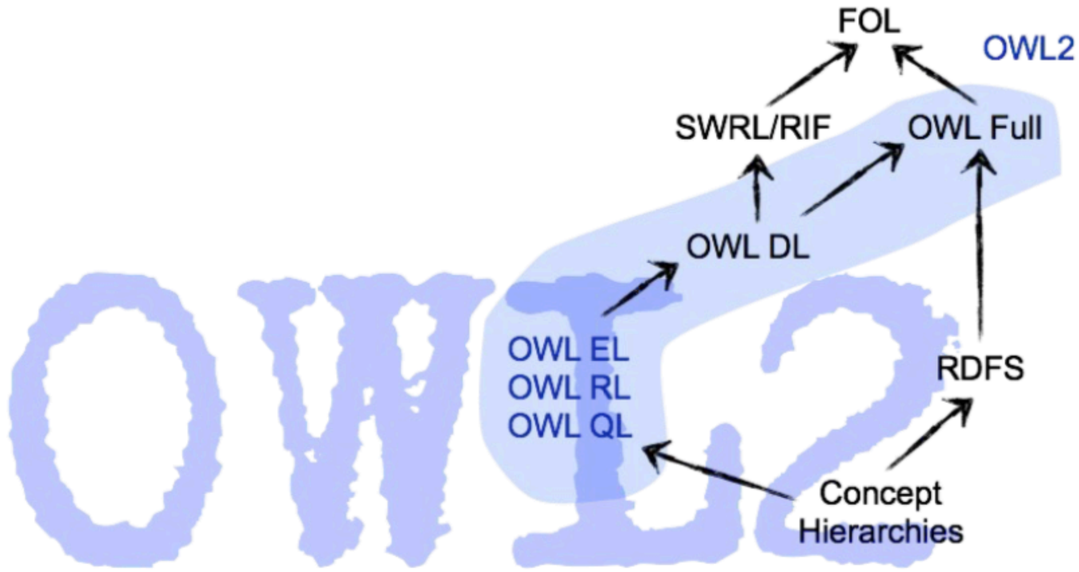
Foundations of Logic

- Definition (for our lecture):
  Logic is the study of how to make formal correct deductions and inferences.

- Why “formal logic”? => to enable automation

Web Ontology Language - OWL

- OWL is a semantic fragment of **First Order Logic (FOL)**
- OWL also exists in different flavors
- OWL EL, OWL RL, OWL QL ⊆ OWL2 DL ⊆ OWL2 Full



OWL2 Is Based on the Description Logic SROIQ(D)

Class Expressions

- Class names A, B
- Conjunction $C \sqcap D$
- Disjunction $C \sqcup D$
- Negation $\neg C$
- Exist. property restriction $\exists R.C$
- Univ. property restriction $\forall R.C$
- Self $\exists S.Self$
- Greater-than $\geq n\ S.C$
- Less-than $\leq n\ S.C$
- Enumerated classes $\{a\}$

Properties

- Property names R, S, T
- Simple properties S, T
- Inverse properties R^-
- Universal property U

Tbox (Class axioms)

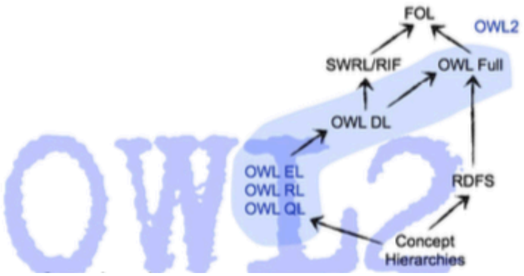
- Inclusion $C \sqsubseteq D$
- Equivalence $C \equiv D$

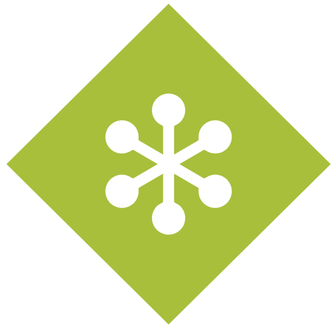
Rbox (Property Axioms)

- Inclusion $R_1 \sqsubseteq R_2$
- General Inclusion $R^{(-)}_1 \circ R^{(-)}_2 \circ \ldots \circ R^{(-)}_n \sqsubseteq R$
- Transitivity
- Symmetry
- Reflexivity
- Irreflexivity
- Disjunctiveness

Abox (Facts)

- Class membership $C(a)$
- Property relation $R(a, b)$
- Negated property relation $\neg S(a, b)$
- Equality $a=b$
- Inequality $a \neq b$

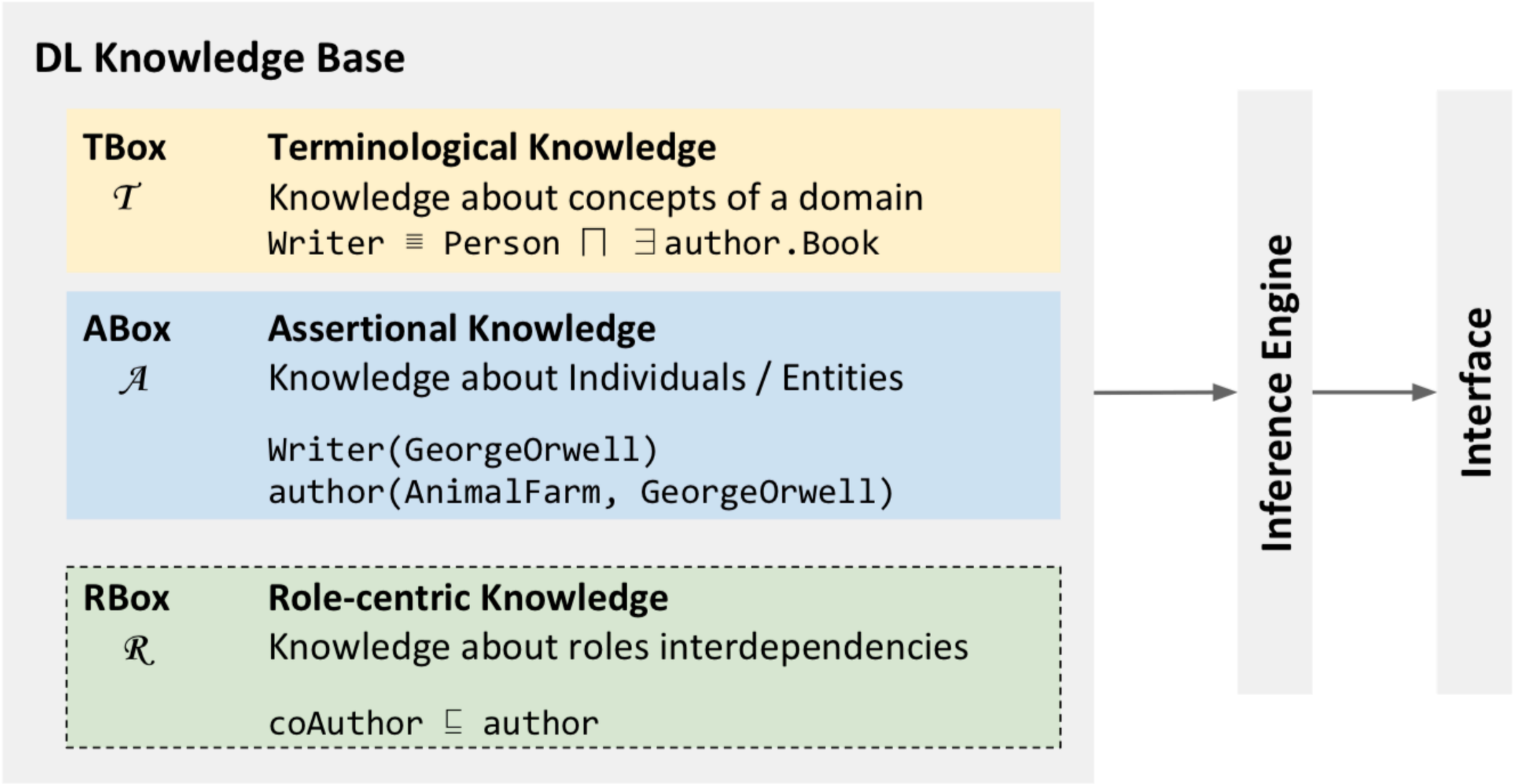




Description Logics (DLs)

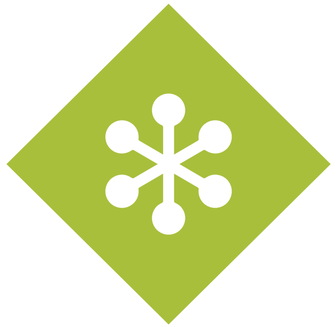
- DLs are fragments of FOL (*compromise of expressivity and saleability*)
- A DL models **concepts, roles** and **individuals**, and their relationships.
- In DL from *simple descriptions more complex descriptions are created* with the help of constructors
- DLs differ in applied constructors (Expressivity)
- DLs have been developed from “semantic Networks”
- DLs are **decidable** (most of the time)
- DLs possess **sufficient expressivity** (most of the time)
- DLs are related to modal logics
- Example for a DL:
  - W3C Standard OWL 2 DL is based on Description Logics *SROIQ(D)*

General DL Architecture



Description Logics (DLs)

- DLs are a family of logic-based formalisms applied for knowledge representations.
- *ALC* (***A****ttribute* ***L****anguage with* ***C****omplement*)
  - Is the smallest deductively complete DL
    - **Conjunction, Disjunction, Negation** are class constructors, denoted as ⊓, ⊔ , ¬
    - Quantifiers restrict domain and range of roles



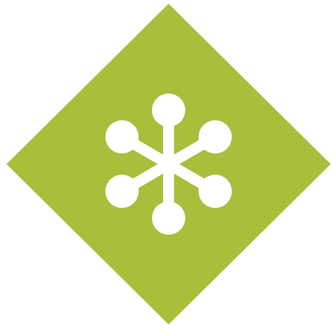
Attributive Language With Complements (-ALC)

- **Basic Building Blocks:**
  - Classes
  - Roles / Properties
  - Individuals
- Writer (GeorgeOrwell)
  - *Individual GeorgeOrwell is of class Writer*
- Book (NineteenEightyfour)
  - *Individual NineteenEightyfour is of class Book*
- author (NineteenEightyfour, GeorgeOrwell)
  - *The book NineteenEightyfour has the author GeorgeOrwell*

Attributive Language With Complements (-ALC)

- $\mathcal{ALC}$ **Atomic Types**
  - Concept names $A, B, \ldots$
  - Special concepts
    - $\top$ - Top (universal concept)
    - $\bot$ - Bottom concept
  - Role names $R, S, \ldots$

- $\mathcal{ALC}$ **Constructors**
  - Negation: $\neg C$
  - Conjunction: $C \sqcap D$
  - Disjunction: $C \sqcup D$
  - Existential quantifier: $\exists R.C$
  - Universal quantifier: $\forall R.C$



Attributive Language With Complements (-ALC)

- **Class Inclusion**
  - $\texttt{Novel} \sqsubseteq \texttt{Book}$
    - *every novel is also a book*
    - equals FOL $(\forall x)(\texttt{Novel}(x) \rightarrow \texttt{Book}(x))$
- **Class Equivalence**
  - $\texttt{Novel} \equiv \texttt{Prose}$
    - *all Prose are exactly Novels*
    - equals FOL $(\forall x)(\texttt{Novel}(x) \leftrightarrow \texttt{Prose}(x))$

Complex Class Relations

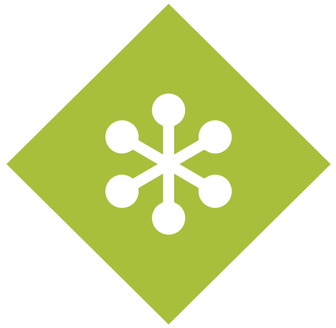
- **Conjunction** $\sqcap$
- **Disjunction** $\sqcup$
- **Negation** $\neg$

$$\text{Novel} \sqsubseteq (\text{Book} \sqcap \text{Fiction}) \sqcup (\text{Paperback} \sqcap \neg\text{Poetry})$$

$\mathcal{ALC}$

$$(\forall x)(\text{Novel}(x) \rightarrow ((\text{Book}(x) \wedge \text{Fiction}(x)) \vee (\text{Paperback}(x) \wedge \neg\text{Poetry}(x))))$$

FOL



ALC - Quantifiers On Roles

- **Strict Binding** of the Range of a Role to a Class
  - $\mathrm{Book} \sqsubseteq \forall \mathrm{author}.\mathrm{Writer}$
  - *A Book must be authored by a Writer*
  - $(\forall x)(\mathrm{Book}(x) \rightarrow (\forall y)(\mathrm{author}(x,y) \rightarrow \mathrm{Writer}(y)))$
- **Open Binding** of the Range of a Role to a Class
  - $\mathrm{Book} \sqsubseteq \exists \mathrm{author}.\mathrm{Person}$
  - *Every Book has at least one author (who is a person)*
  - $(\forall x)(\mathrm{Book}(x) \rightarrow (\exists y)(\mathrm{author}(x,y) \wedge \mathrm{Person}(y)))$

Formal Syntax

- Production rules for creating classes in $\mathcal{ALC}$:
  (A is an atomic class, C and D are complex Classes and R a Role)
  - $C, D ::= A \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C$
- An $\mathcal{ALC}$ **TBox** contains assertions of the form $C \sqsubseteq D$ and $C \equiv D$, where C, D are complex classes.
- An $\mathcal{ALC}$ **ABox** contains assertions of the form $C(a)$ and $R(a, b)$, where C is a complex Class, R a Role and a, b Individuals.
- An $\mathcal{ALC}$-**Knowledge Base** contains an ABox and a TBox.

Description Logics Family

- $\mathcal{ALC}$: Attribute Language with Complement
- $\mathcal{S}$: $\mathcal{ALC}$ + Transitivity of Roles
- $\mathcal{H}$: Role Hierarchies
- $\mathcal{O}$: Nominals
- $\mathcal{I}$: Inverse Roles
- $\mathcal{N}$: Number restrictions $\leq$nR etc.
- $\mathcal{Q}$: Qualified number restrictions $\leq$nR.C etc.
- $(\mathcal{D})$: Datatypes
- $\mathcal{F}$: Functional Roles
- $\mathcal{R}$: Role Constructors

- OWL 2 DL is $\mathcal{SROIQ}(\mathcal{D})$

Open World vs Closed World Assumption

The Open World of Linked Data

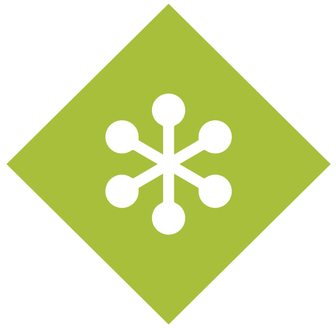
- foaf:**publications** rdf:type rdf:Property ;
  rdf:domain foaf:Person ;
  rdf:range foaf:Document .

  dbpedia:Hasso_Plattner_Institute rdf:type dbo:Organisation ;
  foaf:**publications** <https://hpi.de/technical-reports/> .

  foaf:Person ≠ dbo:Organisation

- According to RDF(S) semantics (foaf:publications rdf:domain foaf:Person .) it can be deduced that

  dbpedia:Hasso_Plattner_Institute rdf:type foaf:Person .

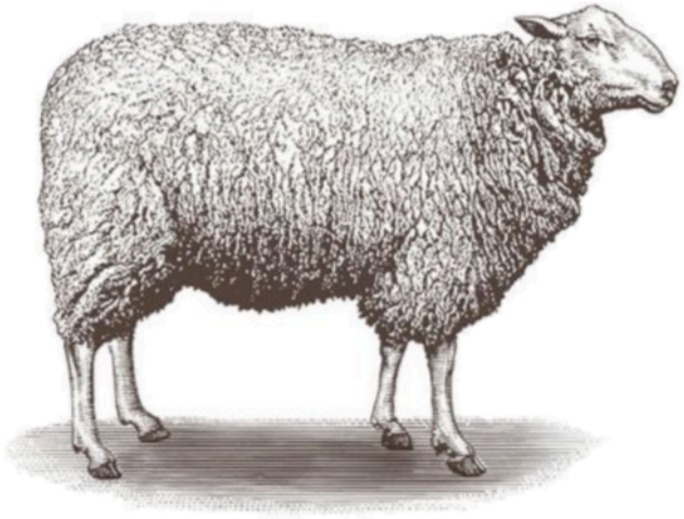


No Unique Name Assumption (UNA)

- In Databases, each individual has a single, unique name
- In DLs, individuals may have more than one name
  - Thus, in DLs it is not necessarily true that two individuals with different names are really two different individuals
- Example:
  - Wizard(HarryPotter) hasFriend(HarryPotter, RonWeasley)
    Wizard(RonWeasley)
  - Wizard(HermioneGranger) hasFriend(HarryPotter, HermioneGranger)
- How many friends does Harry Potter have?
  - DBs, with UNA: 2
  - DLs, no UNA: at least 1
    (since we did not explicitly say that RonWeasley and HermioneGranger are different individuals)

The Open World Assumption - OWA

- If we have an **empty** DL ontology, **everything is possible**
- We then **constrain an ontology iteratively,** making it more restrictive as we go.
- We state what is **not possible**, what is **forbidden** or **excluded**.



$$Sheep \sqsubseteq Animal \sqcap \forall hasLimbs.Leg$$

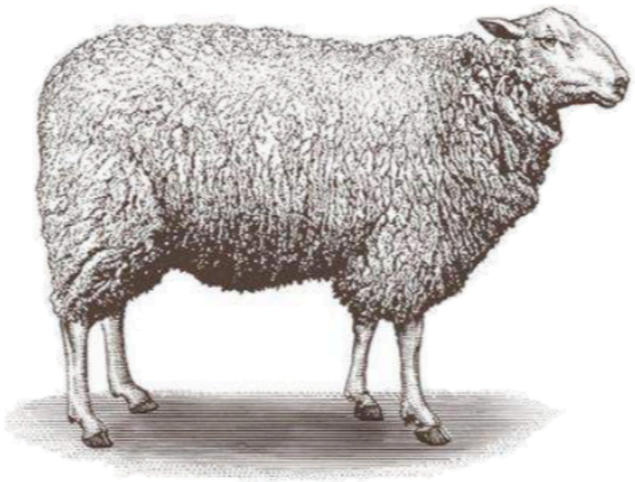
The Open World Assumption - OWA

- Question: Can Sheep fly?

```
A sheep is an animal with 4 legs.
```

- Answer under OWA assumption:
  - *No idea, but probably yes (according to our knowledge base)*
- In the OWA, unless we have a statement (or we can infer) “sheep can/cannot fly” we return “don’t know”

- In the real world, we are used to deal with incomplete information



The Open World Assumption - OWA

- In the Semantic Web / Linked Data we expect people to extend our own models (but we don't worry in advance how)

- The OWA assumes incomplete information by default

- Therefore, we can intentionally underspecify our models and allow others to reuse and extend

```
A sheep is an animal with 4 legs that
(if lifted) can fly
```

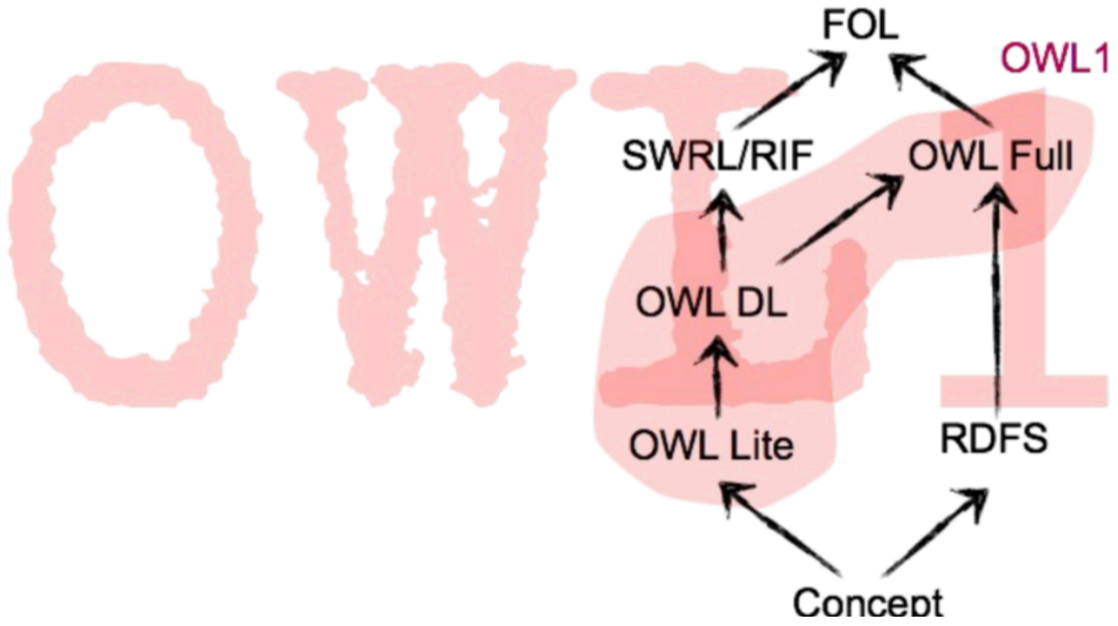
OWL Syntax Variations

Web Ontology Language - OWL

- OWL [$\mathcal{SHOIN}(\mathcal{D})$] W3C Recommendation since 2004
- OWL [$\mathcal{SROIQ}(\mathcal{D})$] W3C Recommendation since 2009
- An OWL Ontology consists of
  - Classes / properties / individuals (instances of classes)
- **Open World Assumption**
  - ‘Absence of information must not be valued as negative information’
  - e.g. : sitsNextTo (PersonA, PersonB)
  - PersonA may also sit next to another person…
- **No Unique Name Assumption**
  - ‘Difference must be expressed explicitly’
  - e.g. : PersonA possibly denotes the same individual as PersonB

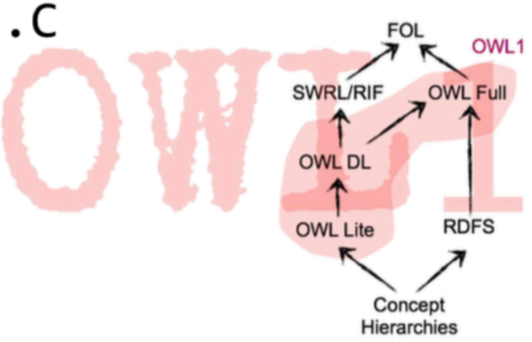
Web Ontology Language - OWL

- OWL is a semantic fragment of FOL
- OWL1 also exists in different flavours
  - OWL Lite $\subseteq$ OWL1 DL $\subseteq$ OWL1 Full



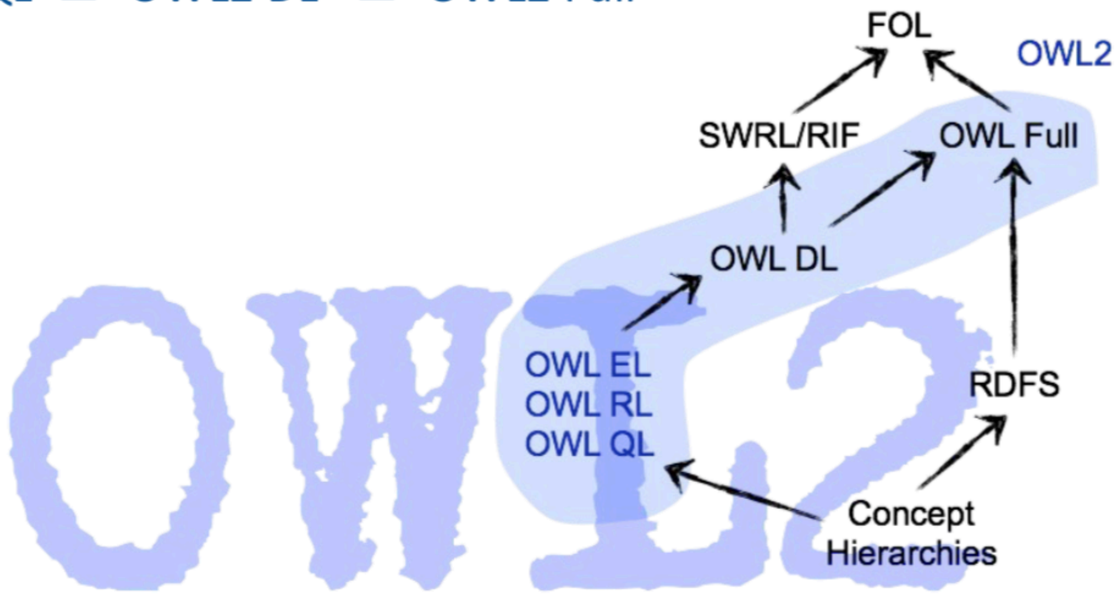
OWL1 Is Based on $\mathcal{SHOIN}(\mathcal{D})$

- **Axioms**
  - **TBox**: subclass relationships $C \sqsubseteq D$ ($\mathcal{H}$)
  - **RBox**: subproperty relationships $R \sqsubseteq S$ ($\mathcal{H}$), inverse properties R^- ($\mathcal{I}$), transitivity $\sqsubseteq^+$ ($\mathcal{S}$)
  - **ABox**: facts for classes $C(a)$, properties $R(a,b)$, equality $a=b$, difference $a \neq b$
- **Class constructors**:
  - conjunction $C \sqcap D$, disjunction $C \sqcup D$, negation $\neg C$ of classes
  - property restrictions: universal $\forall R.C$ and existential $\exists R.C$
  - number restrictions: $\leq n\ R$ and $\geq n\ R$ ($\mathcal{N}$)
  - closed classes (nominals): $\{a\}$ ($\mathcal{O}$)
- Datatypes ($\mathcal{D}$)



Web Ontology Language - OWL

- OWL is a semantic fragment of FOL
- OWL2 also exists in different flavours
  - OWL EL, OWL RL, OWL QL ⊆ OWL2 DL ⊆ OWL2 Full



OWL2 Is Based on $\mathcal{SROIQ}(\mathcal{D})$

Class Expressions

- Class names A, B
- Conjunction $C \sqcap D$
- Disjunction $C \sqcup D$
- Negation $\neg C$
- Exist. property restriction $\exists R.C$
- Univ. property restriction $\forall R.C$
- Self $\exists S.Self$
- Greater-than $\geq n\ S.C$
- Less-than $\leq n\ S.C$
- Enumerated classes $\{a\}$

Properties

- Property names R, S, T
- Simple properties S, T
- Inverse properties R^-
- Universal property U

Tbox (Class axioms)

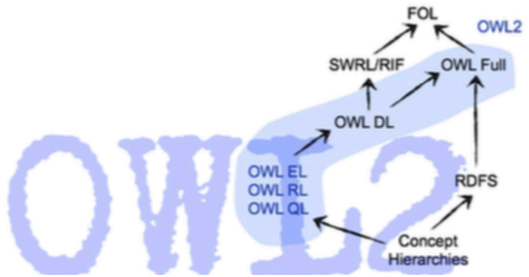
- Inclusion $C \sqsubseteq D$
- Equivalence $C \equiv D$

Rbox (Property Axioms)

- Inclusion $R_1 \sqsubseteq R_2$
- General Inclusion $R^{(-)}_1 \circ R^{(-)}_2 \circ \ldots \circ R^{(-)}_n \sqsubseteq R$
- Transitivity
- Symmetry
- Reflexivity
- Irreflexivity
- Disjunctiveness

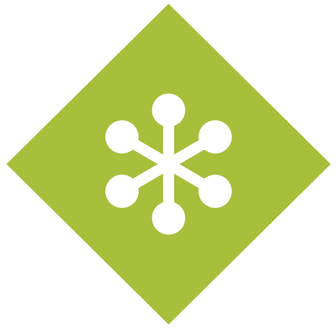
Abox (Facts)

- Class membership $C(a)$
- Property relation $R(a, b)$
- Negated property relation $\neg S(a, b)$
- Equality $a=b$
- Inequality $a \neq b$



OWL Syntax Variants

- OWL2 can be represented in different Syntax variants
  - **Functional Syntax:** substitutes abstract syntax of OWL1
  - **RDF/XML-Syntax:** extension of existing OWL/RDF
  - **OWL/XML-Syntax:** independent XML serialisation
  - **Manchester-Syntax:** machine readable syntax, esp. for ontology editors
  - **Turtle:** concise and easy readable



OWL2 Functional Syntax Example

$$\mathtt{HappyPerson} \equiv \forall \mathtt{hasChild.HappyPerson} \sqcap \exists \mathtt{hasChild.HappyPerson}$$

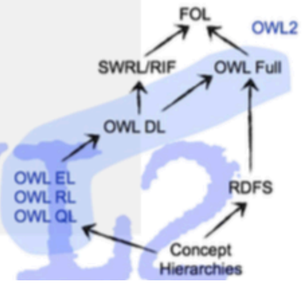
```
Ontology(
  Declaration(Class(:HappyPerson))
  EquivalentClasses(:HappyPerson
    ObjectIntersectionOf(
      ObjectAllValuesFrom(:hasChild :HappyPerson)
      ObjectSomeValuesFrom(:hasChild :HappyPerson)
    )
  )
)
```

- Functional Syntax is easy to define, no RDF restrictions, more compact

OWL2 RDF/XML Syntax Example

$$\text{HappyPerson} \equiv \forall \text{hasChild.HappyPerson} \sqcap \exists \text{hasChild.HappyPerson}$$

```
<?xml version="1.0"?>
<rdf:RDF xmlns=...>
<owl:ObjectProperty rdf:about="http://example.org/turtle#hasChild"/>

<owl:Class rdf:about="http://example.org/turtle#HappyPerson">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty rdf:resource="http://example.org/turtle#hasChild"/>
          <owl:allValuesFrom rdf:resource="http://example.org/turtle#HappyPerson"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="http://example.org/turtle#hasChild"/>
          <owl:someValuesFrom rdf:resource="http://example.org/turtle#HappyPerson"/>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
</rdf:RDF>
```



OWL2 OWL/XML Syntax Example

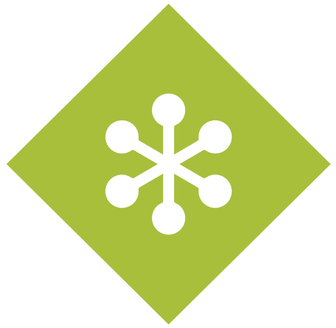
- HappyPerson ≡ ∀hasChild.HappyPerson ⊓ ∃hasChild.HappyPerson

```
<?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"....>

  <Declaration>
    <Class abbreviatedIRI=":HappyPerson"/>
  </Declaration>
  <EquivalentClasses>
    <Class abbreviatedIRI=":HappyPerson"/>
      <ObjectIntersectionOf>
        <ObjectAllValuesFrom>
          <ObjectProperty abbreviatedIRI=":hasChild"/>
          <Class abbreviatedIRI=":HappyPerson"/>
        </ObjectAllValuesFrom>
        <ObjectSomeValuesFrom>
          <ObjectProperty abbreviatedIRI=":hasChild"/>
          <Class abbreviatedIRI=":HappyPerson"/>
        </ObjectSomeValuesFrom>
      </ObjectIntersectionOf>
  </EquivalentClasses>
</Ontology>
```

OWL2 Manchester Syntax Example

$$\text{HappyPerson} \equiv \forall \text{hasChild.HappyPerson} \sqcap \exists \text{hasChild.HappyPerson}$$

```
Ontology:

  ObjectProperty: hasChild

  Class: HappyPerson

  EquivalentTo:
        (hasChild only HappyPerson)
         and (hasChild some HappyPerson)
```



OWL2 Turtle Syntax Example

HappyPerson ≡ ∀hasChild.HappyPerson ⊓ ∃hasChild.HappyPerson

```
:HappyPerson a owl:Class ;
   owl:equivalentClass  [
   a owl:Class ;
   owl:intersectionOf ([ a owl:Restriction ;
                         owl:onProperty :hasChild ;
                         owl:allValuesFrom :HappyPerson  ]
                       [ a owl:Restriction ;
                         owl:onProperty :hasChild ;
                         owl:someValuesFrom :HappyPerson ]
                       )
    ] .
```

OWL Basic Building Blocks

OWL Basic Building Blocks

- OWL namespace:
  @prefix owl: <http://www.w3.org/2002/07/owl#>

- There is a Turtle Syntax for OWL

- **OWL axioms** consist of the following **three building blocks:**
  - **Classes**
    - *comparable with classes in RDFS*
  - **Individuals**
    - *comparable with class instances in RDFS*
  - **Properties**
    - *comparable with properties in RDFS*

OWL Classes

- There exist two predefined classes
  - ***owl:Thing*** *(class that contains all individuals)*
  - ***owl:Nothing*** *(empty class)*
- **Definition** of a class
  - :Book a **owl:Class .**

equivalent expression in description logics

$$\top \equiv C \sqcup \neg C$$

$$\bot \equiv C \sqcap \neg C$$

OWL Individuals

- **Definition of individuals** via class membership
  - :NineteenEightyfour a :Book .
  - Book(NineteenEightyfour)
- Individuals can also be defined without class membership as **named individuals**
  - :HaraldSack a **owl:NamedIndividual .**

OWL Object Properties

- There exist two property variants:
  - *Object properties*
  - *Datatype properties*

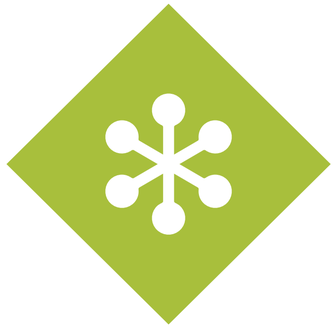
- Object properties have **classes** as range
  - :author a **owl:ObjectProperty .**

- Domain and Range of object properties
  :author a **owl:ObjectProperty ;**
  - *rdfs:domain :Book ;*
  - *rdfs:range :Writer .*

OWL Datatype Properties

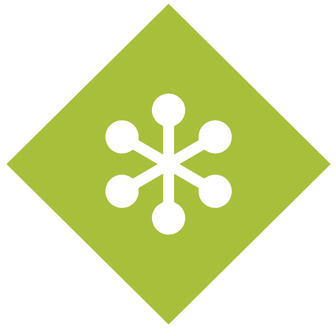
- Datatype properties have **datatypes** as range
  - :publicationYear a **owl:DatatypeProperty .**

- Domain and Range of datatype properties
  - :publicationYear a **owl:DatatypeProperty ;**
    - *rdfs:domain :Book ;*
    - *rdfs:range xsd:integer*



OWL Properties and Individuals

```
:Book a owl:Class .
:Writer a owl:Class .

:author a owl:ObjectProperty ;
        rdfs:domain :Book ;
        rdfs:range  :Writer .

:publicationYear a owl:DatatypeProperty ;
                 rdfs:domain :Book ;
                 rdfs:range xsd:integer .

:GeorgeOrwell a Writer .
:NineteenEightyFour a :Book ;
                    :author :GeorgeOrwell ;
                    :publicationYear 1948 .
```



OWL Class Hierarchies

```
:Novel a owl:Class ;
       rdfs:subClassOf :Book .
:Book a owl:Class ;
      rdfs:subClassOf :Work .
:Work a owl:Class .
```

$$\mathrm{Novel} \sqsubseteq \mathrm{Book}$$
$$\mathrm{Book} \sqsubseteq \mathrm{Work}$$

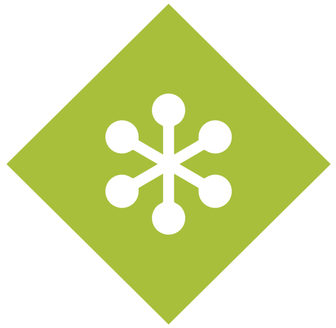
- Via inference it can be entailed that :Novel is also a subclass of :Work

OWL Class Hierarchies and Disjunctiveness

```
:Book a owl:Class .
:Writer a owl:Class .
:Novel a owl:Class ;
       rdfs:subClassOf :Book .
:Poet a owl:Class ;
      rdfs:subClassOf :Writer .

:Book owl:disjointWith :Writer .
```

$$\text{Novel} \sqsubseteq \text{Book}$$
$$\text{Poet} \sqsubseteq \text{Writer}$$
$$\text{Book} \sqcap \text{Writer} \sqsubseteq \bot$$

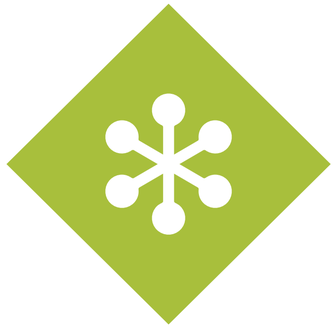
- via inference it can be entailed that :Novel and :Poet are also disjoint classes



OWL Class Hierarchies and Disjunctiveness

- OWL provides a **shortcut** to define several classes to be disjunctive

```
[] a owl:AllDisjointClasses ;
     owl:members
       ( :Book
         :Writer
         :Vegetable
         :Furniture
         :Car ) .
```



OWL Class Hierarchies and Equivalence

```
:Writer a owl:Class .
:Author a owl:Class .
:Poet a owl:Class ;
      rdfs:subClassOf :Writer .

:Writer owl:equivalentClass :Author .
```

$$\texttt{Poet} \sqsubseteq \texttt{Writer}$$
$$\texttt{Writer} \equiv \texttt{Author}$$

- via inference it can be entailed that :Poet is also an :Author

OWL Individuals

```
:NineteenEightyfour a :Novel ;
                    :author :GeorgeOrwell ;
                    :publicationYear 1948 ;
                    owl:sameAs :ARX012345 .

:Novel a owl:Class ;
       rdfs:subClassOf :Book .

:Book a owl:Class.
```

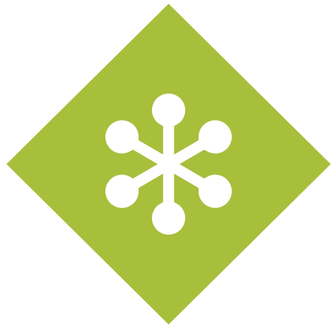
- via inference it can be entailed that :ARX012345 is a :Book
difference of Individuals via **owl:differentFrom**

```
:ARX012345 a :Novel ;
           owl:differentFrom :ARX012346 .
```

OWL Class Hierarchies and Disjunctiveness

- OWL provides a **shortcut** to define several individuals to be different

```
[] a owl:AllDifferent ;
   owl:distinctMembers
     ( :NineteenEightyfour
       :AnimalFarm
       :BraveNewWorld
       :Simulacron5 ) .
```

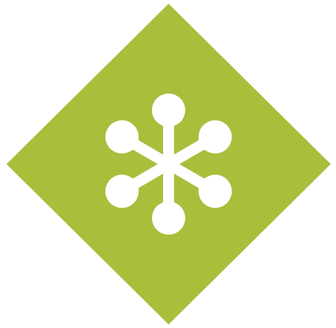


OWL Closed Classes - Nominals

```
:Novel a owl:Class .
:AnimalFarm a :Novel .
:NineteenEightyfour a :Novel .

:NovelsInStore a owl:Class ;
     owl:oneOf
     ( :AnimalFarm
       :NineteenEightyfour ) .
```

NovelsInStore ⊑ {AnimalFarm, NineteenEightyfour}

- There are only two novels available in the store.



OWL Logical Class Constructors

- logical AND (conjunction):
  - **owl:intersectionOf** ⊓
- logical OR (disjunction):
  - **owl:unionOf** ⊔
- logical negation:
  - **owl:complementOf** ¬

- **Logical constructors are applied to create complex class descriptions from atomic classes.**

OWL Logical Class Constructors

```
:Book a owl:Class .

:ThingsInStore a owl:Class .

:BooksInStore a owl:Class ;
      owl:intersectionOf (:ThingsInStore :Book) .
```

BooksInStore ≡ ThingsInStore ⊓ Books

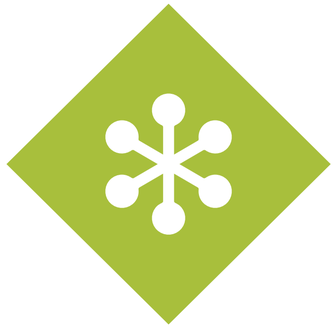
- The class :BooksInStore results from the intersection of all individuals of the classes :ThingsInStore and :Book

OWL Logical Class Constructors

```
:Book a owl:Class ;
      owl:equivalentClass [
         owl:unionOf ( :Novel
                       :Poetry
                       :NonFiction )
] .
```

$$Book \equiv Novel \sqcup Poetry \sqcup NonFiction$$

- Novels, poetry, and non-fiction are also books



OWL Logical Class Constructors

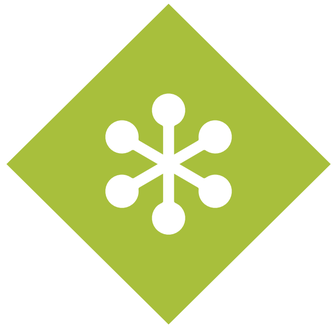
```
:Book a owl:Class ;
      rdfs:subClassOf [
      owl:complementOf :Writer
] .
```

$$Book \sqsubseteq \neg Writer$$

- semantically equivalent assertion:

```
:Book a owl:Class ;
      owl:disjointWith :Writer .
```

OWL Property Restrictions



OWL Property Restrictions

- OWL property restrictions are used to describe complex classes via properties

- Restrictions on values:
  - **owl:hasValue**
  - **owl:allValuesFrom**
  - **owl:someValuesFrom**

- Restrictions on cardinality:
  - **owl:cardinality**
  - **owl:minCardinality**
  - **owl:maxCardinality**

OWL Property Restrictions With Constants

```
:OrwellsBooks a owl:Class ;
              rdfs:subClassOf
                [ a owl:Restriction ;
                  owl:onProperty :author ;
                  owl:hasValue :GeorgeOrwell ] .
```

$$\text{OrwellsBooks} \sqsubseteq \text{author}.(\text{GeorgeOrwell})$$

- Class :OrwellsBooks is described via fixed value assignment (=constant) of the individual :GeorgeOrwell to the property :author

OWL Property Restriction With Strict Binding

```
:Poetry a owl:Class ;
        rdfs:subClassOf
          [ a owl:Restriction ;
            owl:onProperty :author ;
            owl:allValuesFrom :Poet ] .
```

$$Poetry \sqsubseteq \forall author.Poet$$

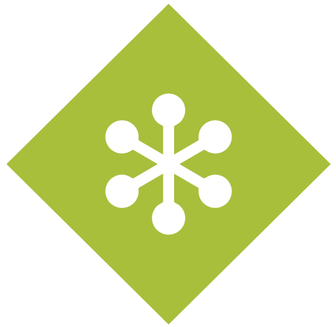
- **owl:allValuesFrom**
  fixes all instances of a specific class C as allowed range for a property p (strict binding) ∀p.C

OWL Property Restriction With Loose Binding

```
:Reader a owl:Class ;
        rdfs:subClassOf
          [ a owl:Restriction ;
            owl:onProperty :reads ;
            owl:someValuesFrom :Book ] .
```

$$\text{Reader} \sqsubseteq \exists \text{reads}.\text{Book}$$

- **owl:someValuesFrom**
  describes that there must exist an individual for p and fixes its range to class C (loose binding) ∃p.C



OWL Cardinality Restrictions

```
:Tetralogy a owl:Class ;
           rdfs:subClassOf
             [ a owl:Restriction ;
               owl:onProperty :hasVolumes ;
               owl:cardinality 4 ] .
```

$$\text{Tetralogy} \sqsubseteq (=4)\text{hasVolumes}$$

- **owl:cardinality** restricts to an exact number
- **owl:minCardinality / owl:maxCardinality** restricts to upper / lower bounds

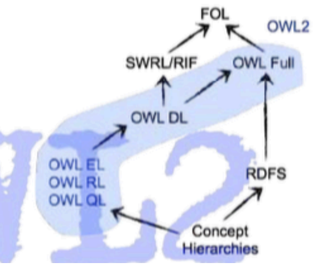
OWL Property Relationships

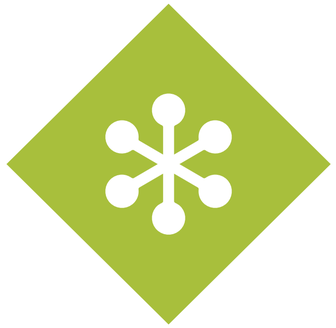
OWL Property Relationships

- **Property hierarchies** can be created via specialisations: rdfs:subPropertyOf
- **Inverse properties** are defined via owl:inverseOf
- **Identical properties** are defined via owl:equivalentProperty

```
:isMadeOf a owl:ObjectProperty ;
       rdfs:subPropertyOf :consistsOf .

:reads a owl:ObjectProperty ;
       owl:inverseOf :isReadBy .

:composedOf a owl:ObjectProperty ;
       owl:equivalentProperty :consistsOf .
```

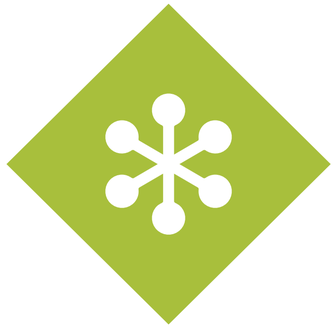
```
isMadeOf   ⊑ consistsOf
reads-     ≡ isReadBy
composedOf ≡ consistsOf
```





OWL Property Relationships

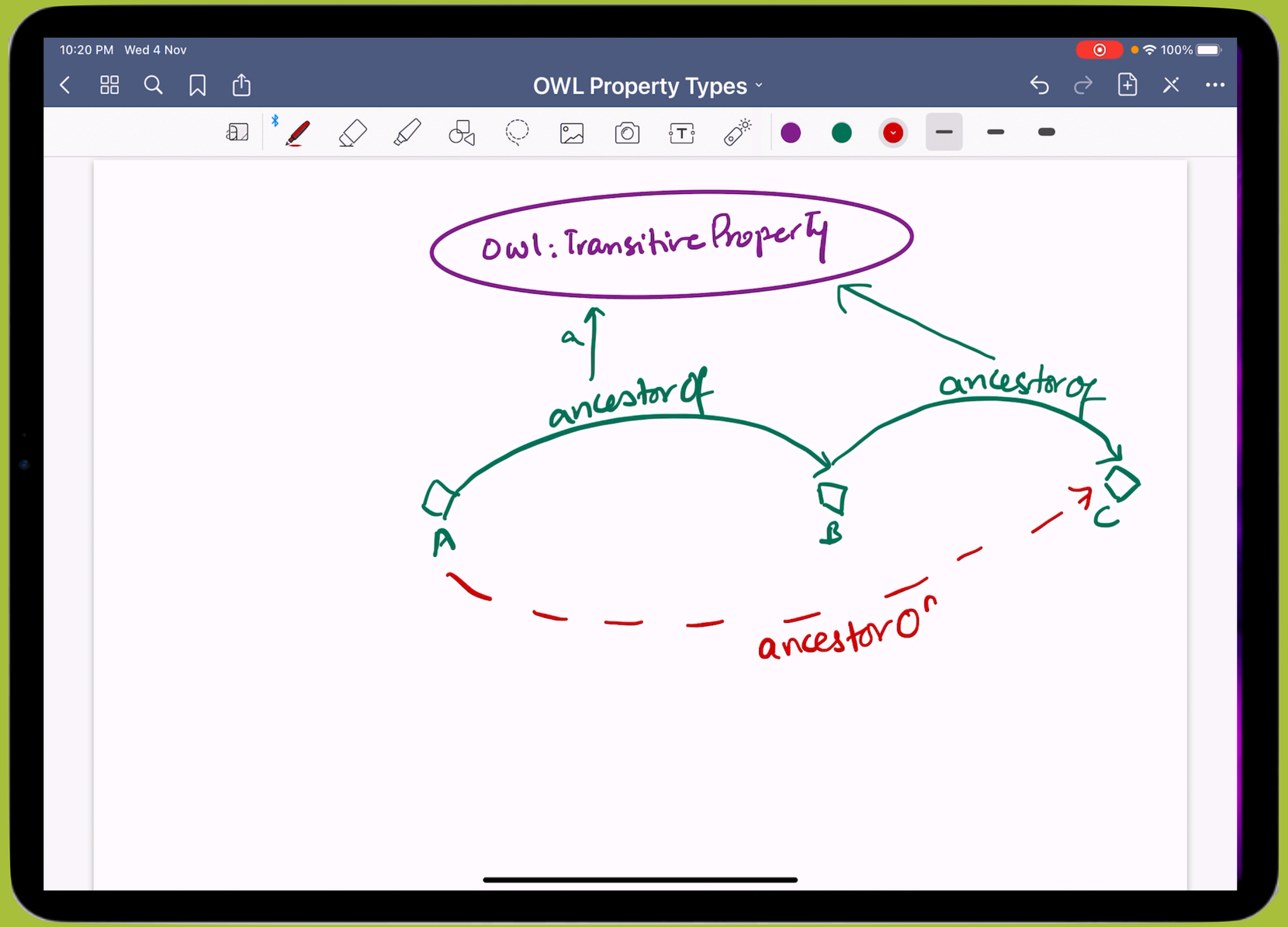
- owl:TransitiveProperty
  - e.g. if isPartOf(a,b) and isPartOf(b,c) then it holds that isPartOf(a,c)
- owl:SymmetricProperty
  - e.g. if isNeighbourOf(a,b) then it holds that isNeighbourOf(b,a)
- owl:FunctionalProperty
  - e.g. if hasMother(a,b) and hasMother(a,c) then it holds that b=c
- owl:InverseFunctionalProperty
  - e.g. if isMotherOf(b,a) and isMotherOf(c,a) then it holds that b=c



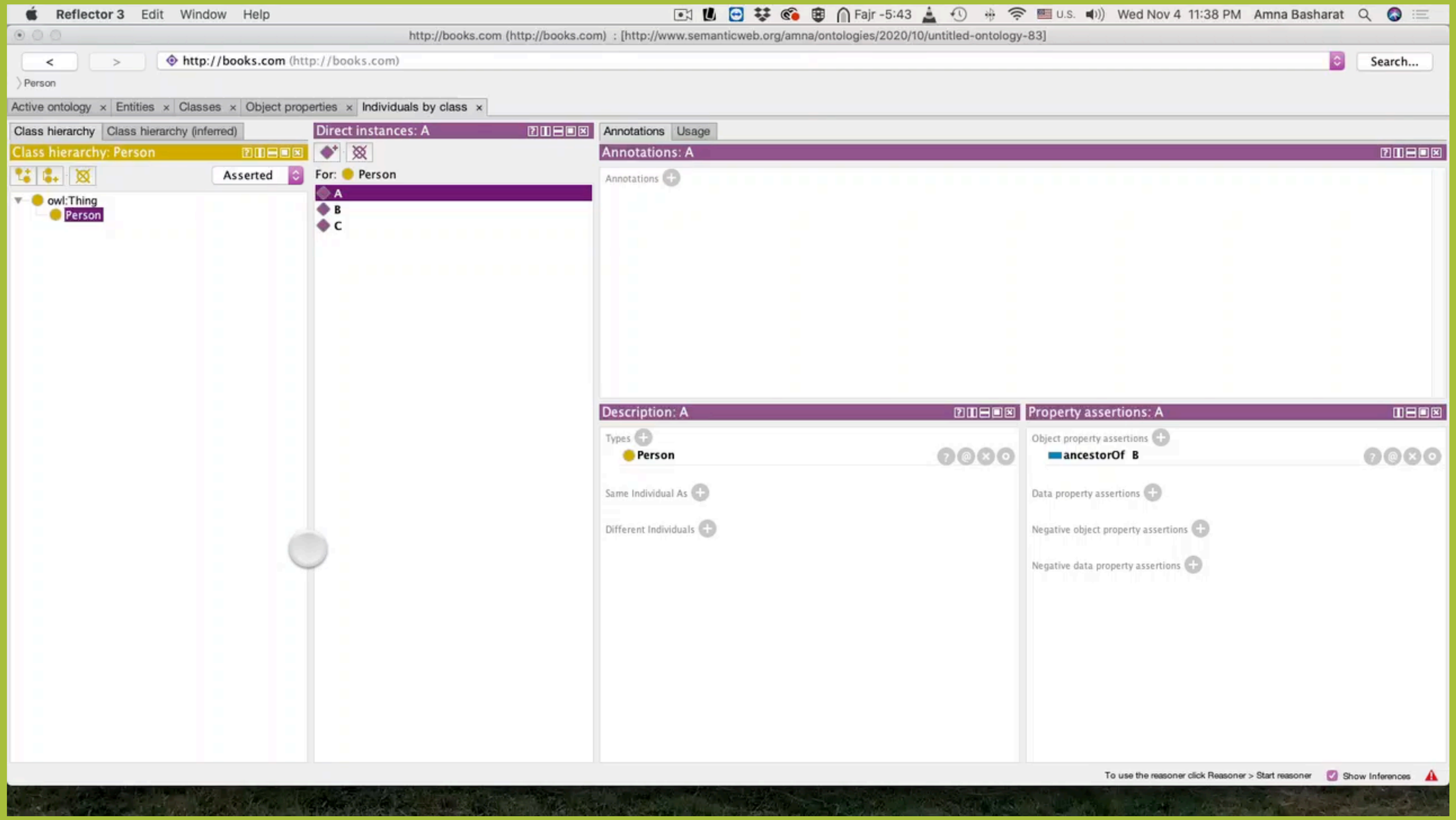
OWL Transitive Properties

```
:isPublishedBefore a owl:ObjectProperty ;
                   a owl:TransitiveProperty ;
                   rdfs:domain owl:Book ;
                   rdfs:range owl:Book .

:AnimalFarm a owl:Book ;
            :isPublishedBefore :NineteenEightyfour .

:BraveNewWorld a :Book ;
               :isPublishedBefore :AnimalFarm .
```

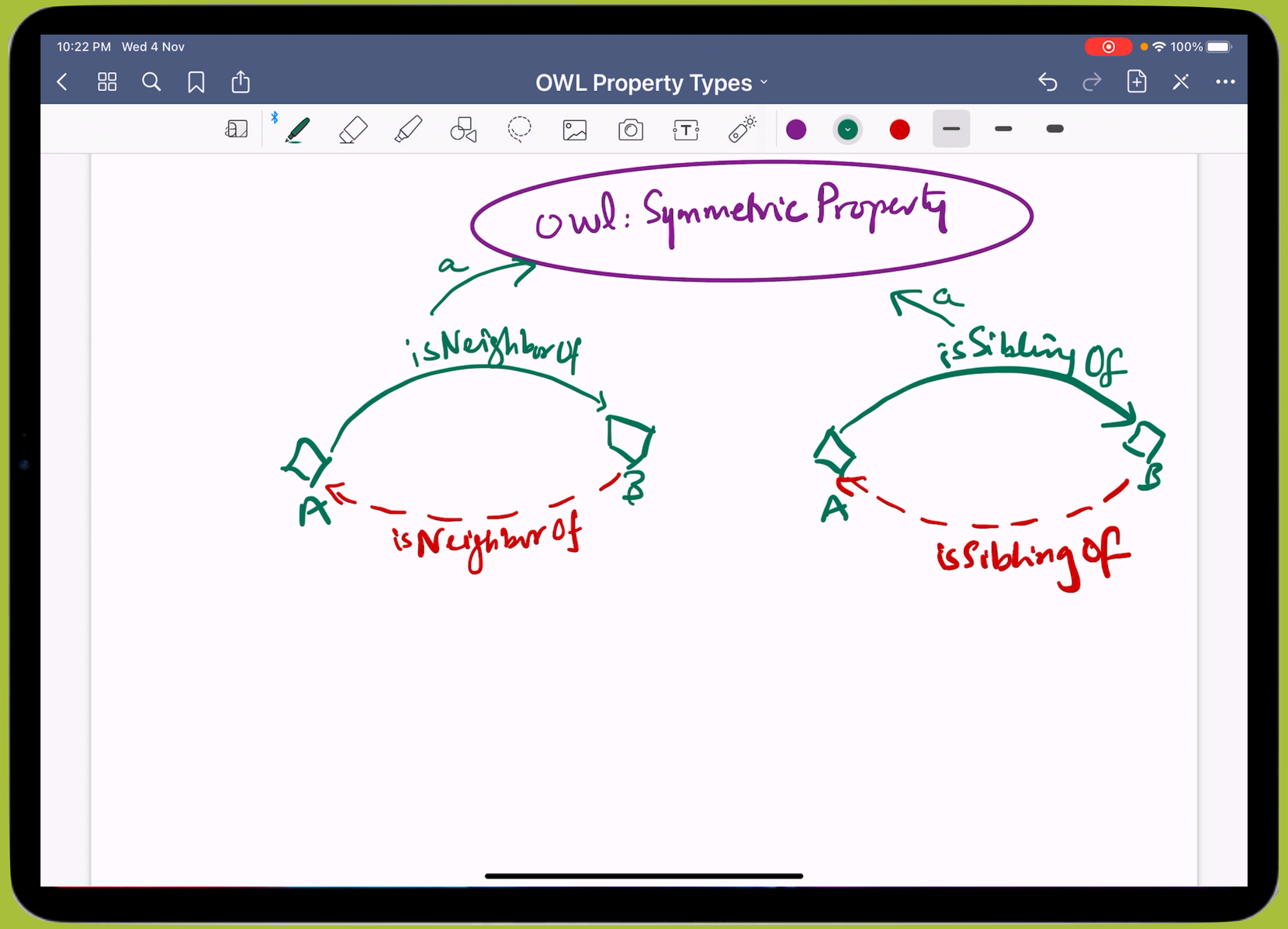
- Via inference it can be entailed that :BraveNewWorld has been published before :NineteenEightyFour



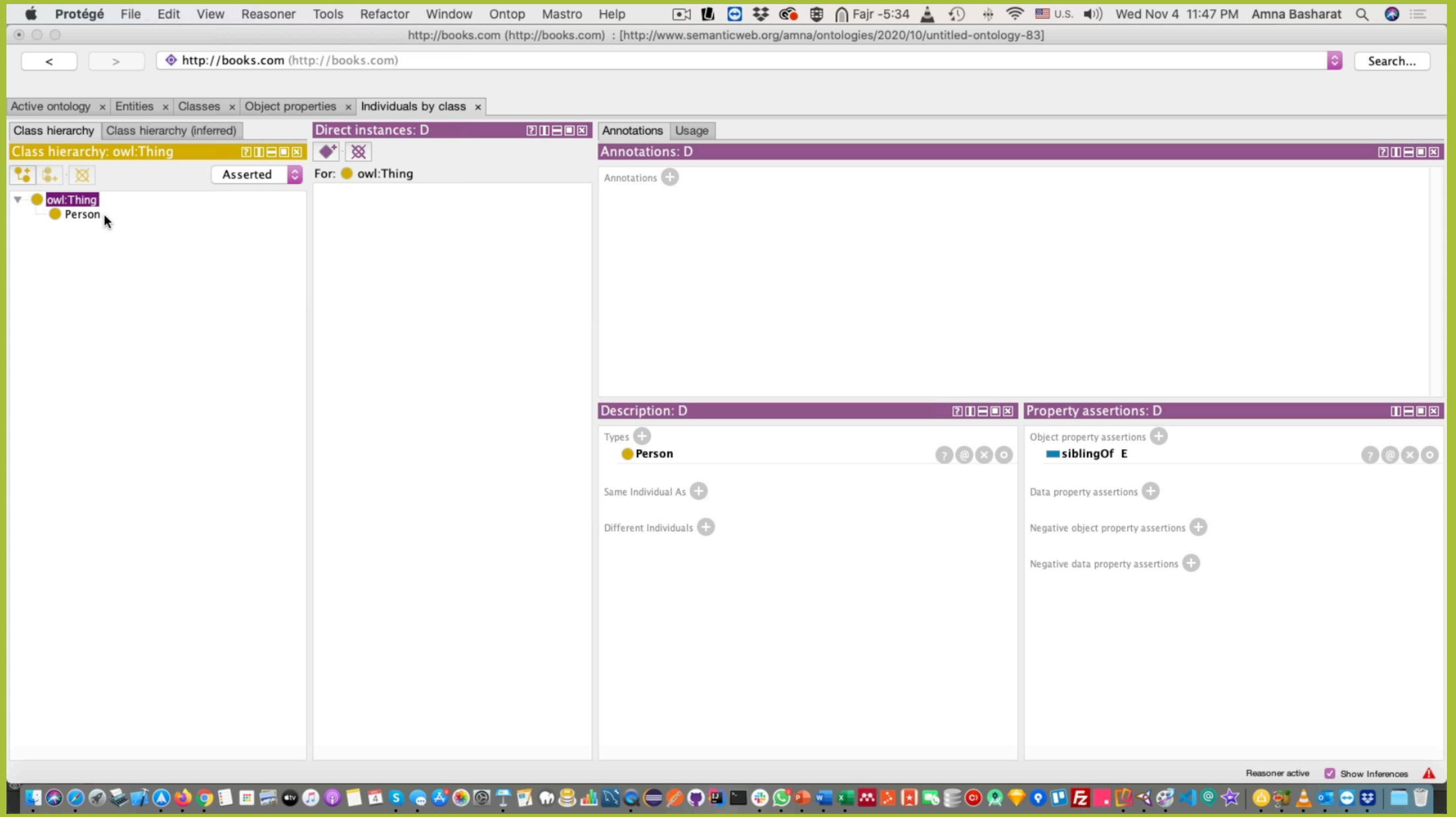
Transitive Property Illustration

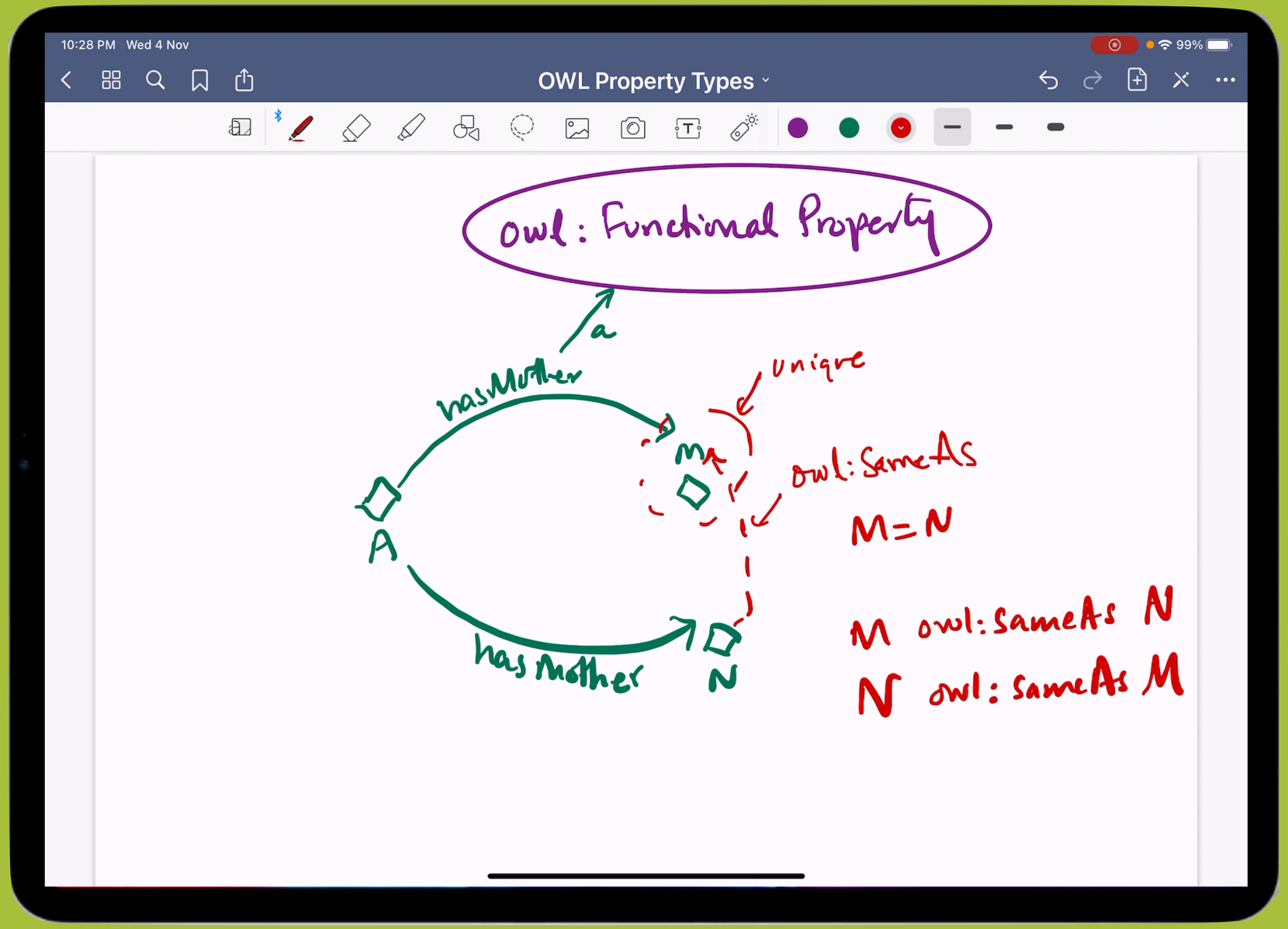


Transitive Property Illustration (Protege)

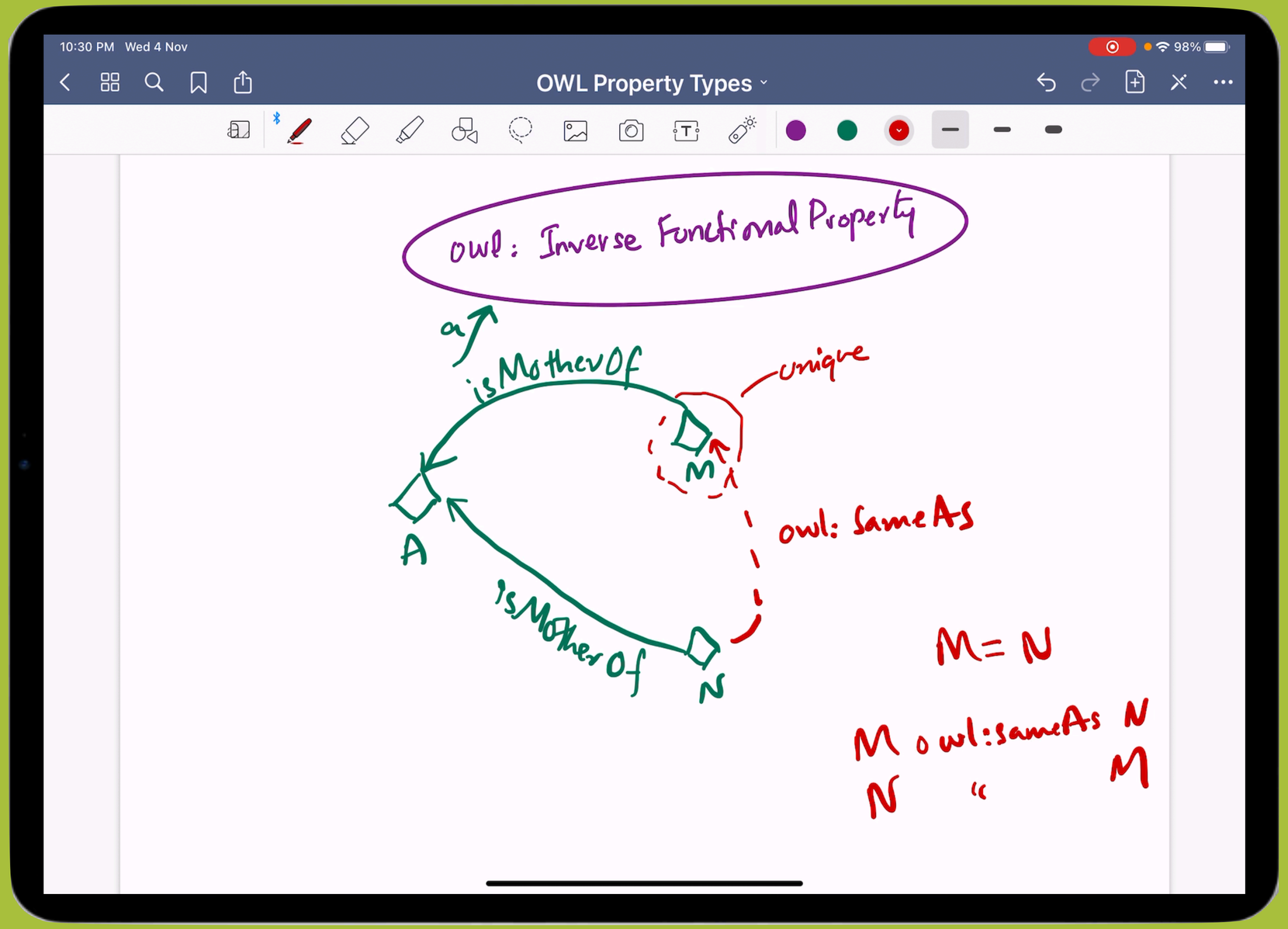


Symmetric Property Illustration

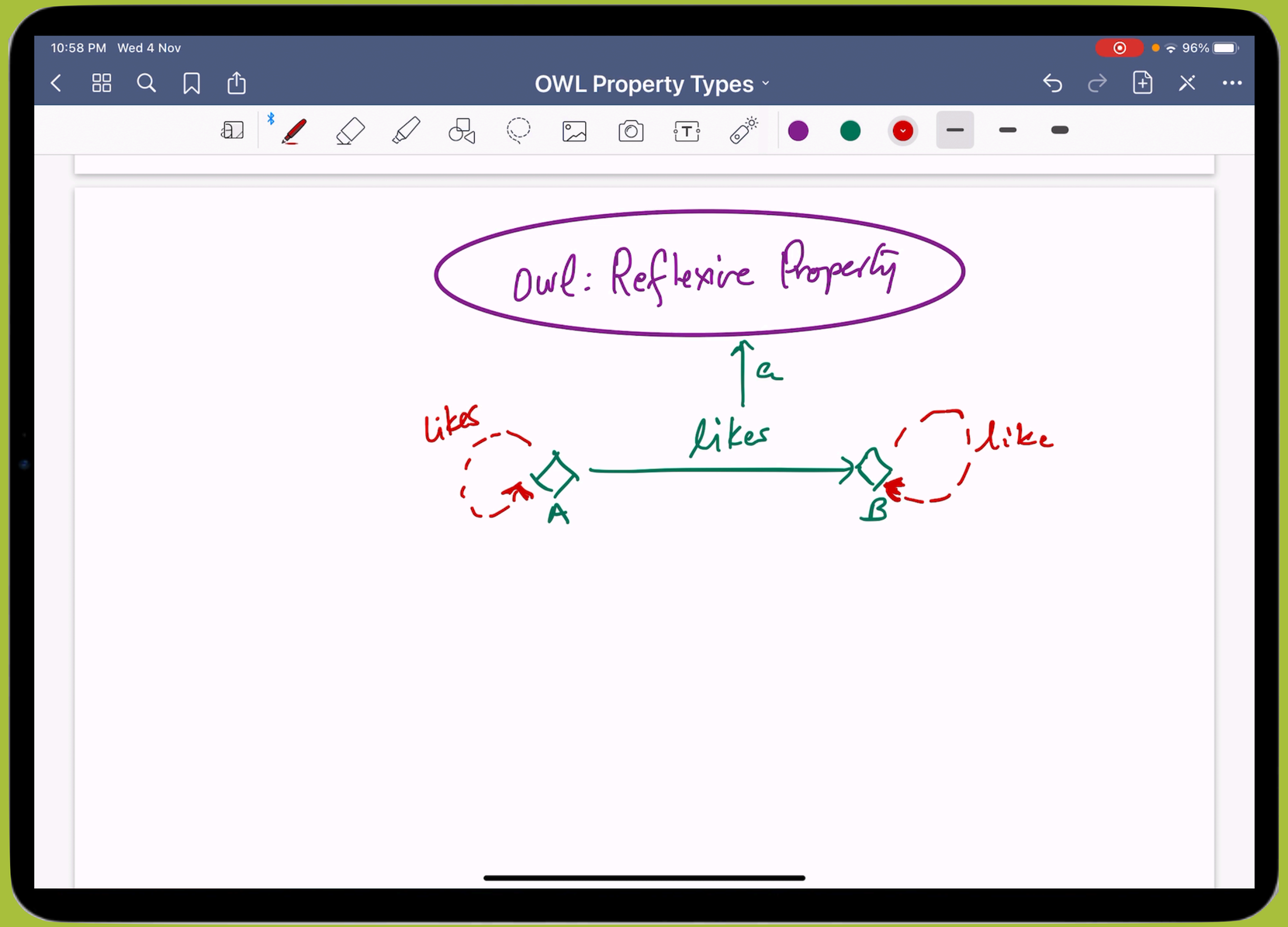

Protégé File Edit View Reasoner Tools Refactor Window Ontop Mastro Help
Wed Nov 4 11:47 PM Amna Basharat
http://books.com (http://books.com) : [http://www.semanticweb.org/amna/ontologies/2020/10/untitled-ontology-83]
http://books.com (http://books.com)
Search...
Active ontology
Entities
Classes
Object properties
Individuals by class
Class hierarchy
Class hierarchy (inferred)
Class hierarchy: owl:Thing
Asserted
owl:Thing
Person
Direct instances: D
For: owl:Thing
Annotations
Usage
Annotations: D
Annotations
Description: D
Types
Person
Same Individual As
Different Individuals
Property assertions: D
Object property assertions
siblingOf E
Data property assertions
Negative object property assertions
Negative data property assertions
Reasoner active
Show Inferences



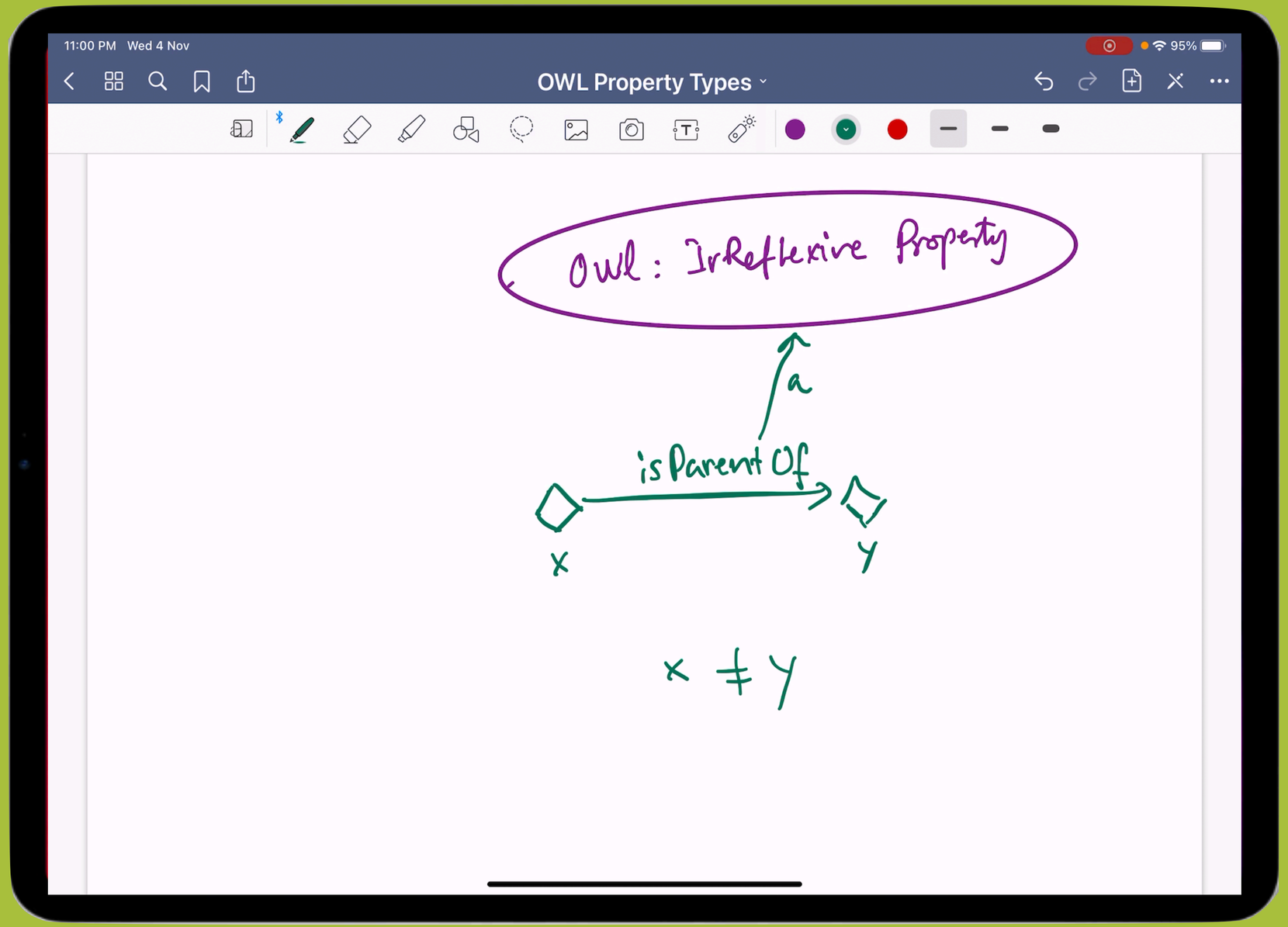
Functional Property Illustration



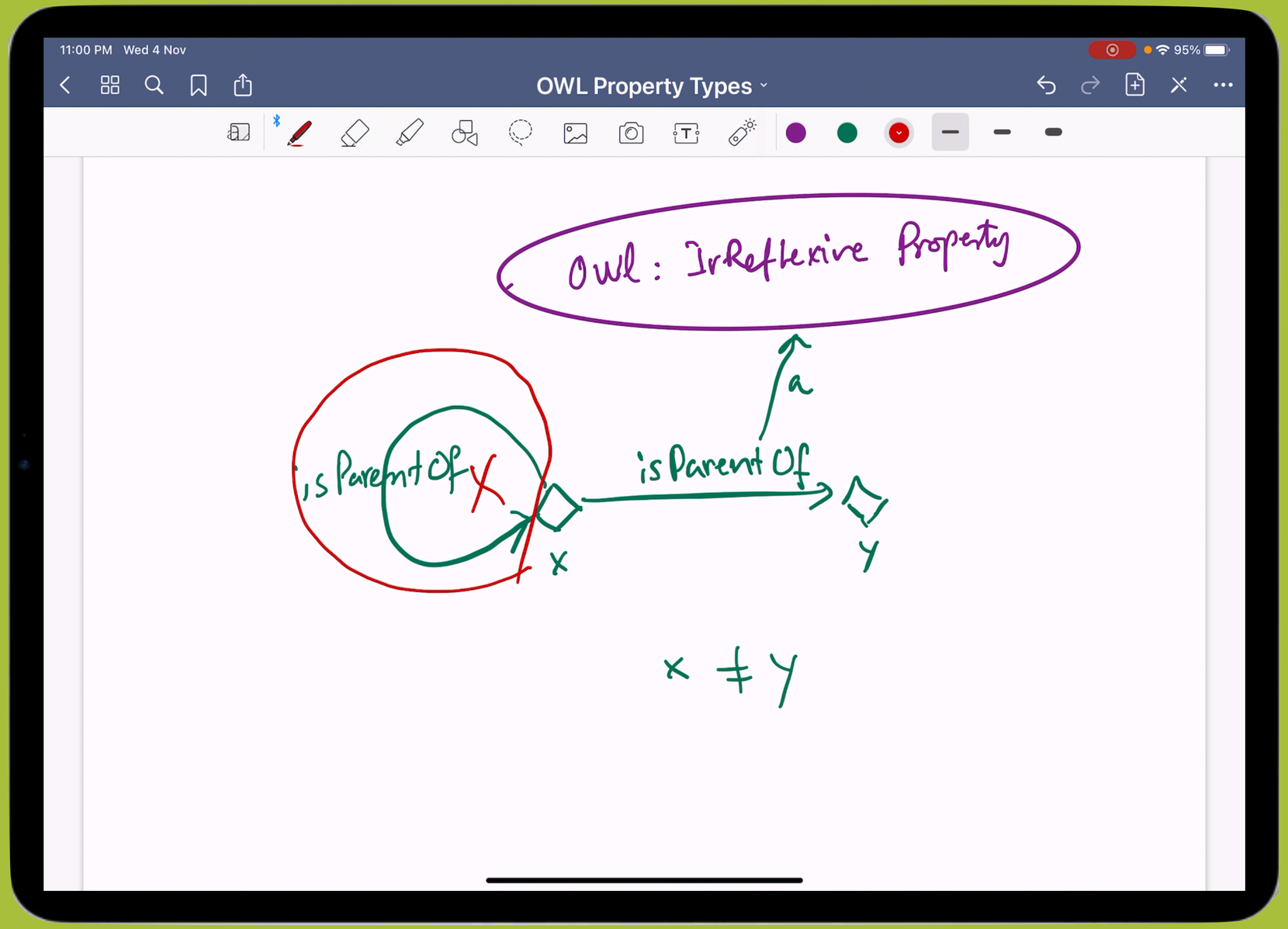
Inverse Functional Property Illustration



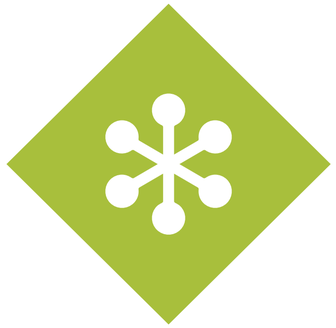
Reflexive Property Illustration



IrReflexive Property Illustration

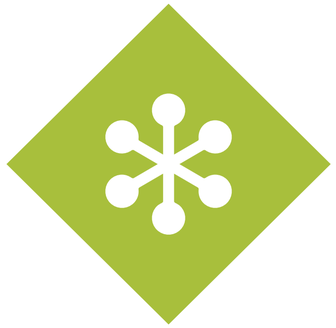


IrReflexive Property



OWL More Property Relationships

- **Asymmetric properties** via owl:AsymmetricProperty
  - e.g. if it holds that isLeftOf(a,b) then it is NOT isLeftOf(b,a)
- **Reflexive properties** via owl:ReflexiveProperty
  - e.g. isRelatedTo(x,x)
- **Irreflexive properties** via owl:IrreflexiveProperty
  - e.g. If isParentOf(x,y) then x≠ y



OWL Disjunctive Properties

- Two properties R and S are **disjunctive**, if two individuals x, y are never related via both properties

  :hasParent a owl:opbjectProperty ;

  owl:propertyDisjointWith :hasChild.

- Shortcut for several **disjunctive properties**

  [] rdf:type owl:AllDisjointProperties ;

  owl:members

  (:hasParent

  :hasChild

  :hasGrandchild).

OWL Negated Property Instantiation

- Two individuals can be explicitly defined to be not related with each other via a given property.

¬isBrother(GeorgeOrwell,AldousHuxley)

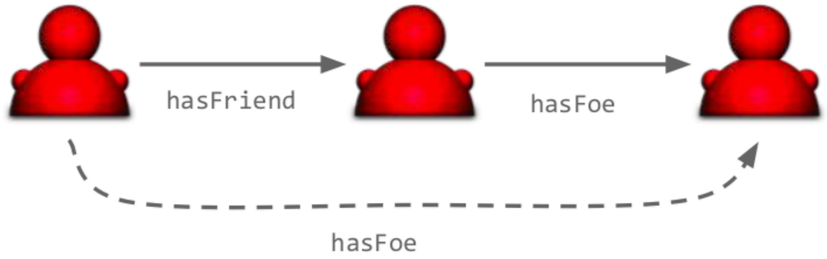
```
[] rdf:type owl:NegativePropertyAssertion ;
   owl:sourceIndividual :GeorgeOrwell ;
   owl:assertionProperty :isBrother ;
   owl:targetIndividual :AldousHuxley .
```

OWL Property Chaining

- **Complex Roles** (properties) can be constructed from simple roles (**RBox**)
  - *'the friends of my friends are also my friends'*

```
hasFriend a owl:TransitiveProperty .
```

- But: „*The foes of my friends are also my foes.*“
  - cannot be expressed in that way



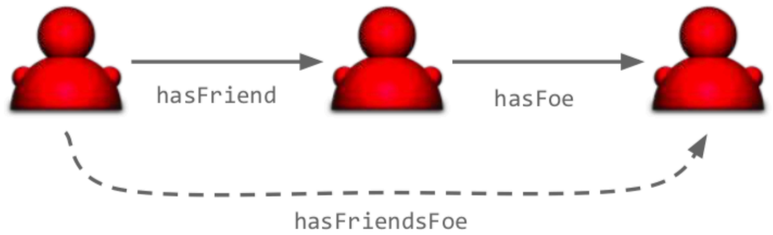
- in FOL it can be expressed as a rule (axiom):
  - $\forall x,y,z: hasFriend(x,y) \wedge hasFoe(y,z) \rightarrow hasFriendsFoe(x,z)$

OWL Property Chaining

- General Role Inclusion (**Property chaining**)

```
:hasFriendsFoe a owl:ObjectProperty ;
   owl:propertyChainAxiom  ( :hasFriend :hasFoe ) .
```

- not allowed for datatype properties



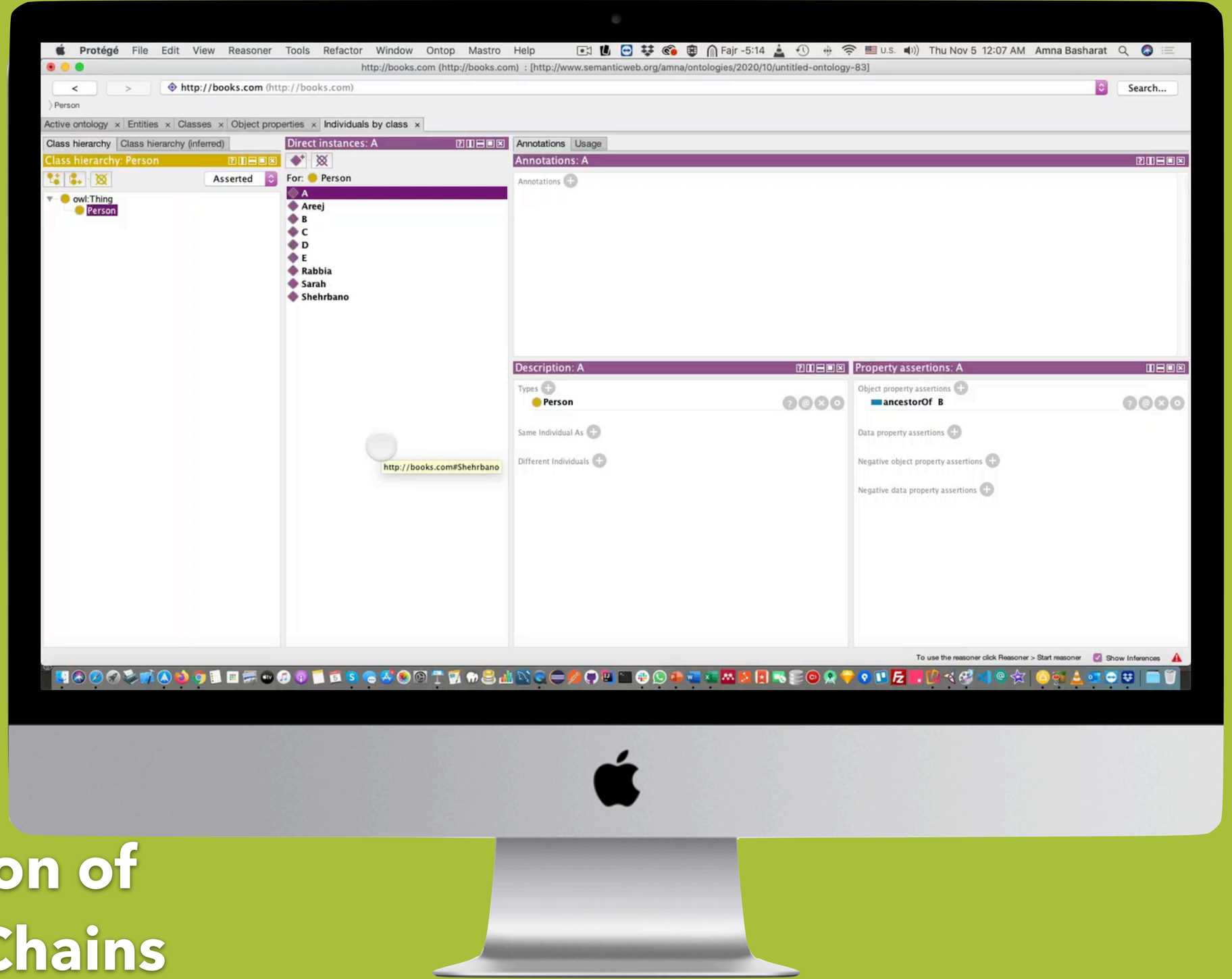


Illustration of PropertyChains

OWL Qualified Number Restrictions

- Class constructors with **number restrictions on properties connected with a range constraint**

```
:Tetralogy a owl:Class ;
           rdfs:subClassOf
             [ a owl:Restriction ;
               owl:onProperty :hasVolumes ;
               owl:cardinality 4 ] .
```

Tetralogy ⊑ (=4)hasVolumes

SuccessfulAuthor ⊑ ≥1 notableWork.Bestseller

```
:SuccessfulAuthor a owl:Class;
    rdfs:subClassOf [
      a owl:Restriction ;
      owl:onProperty :notableWork ;
      owl:minQualifiedCardinality  "1"^^xsd:nonNegativeInteger ;
      owl:onClass :Bestseller ] .
```

owl:maxQualifiedCardinality, owl:minQualifiedCardinality, owl:qualifiedCardinality

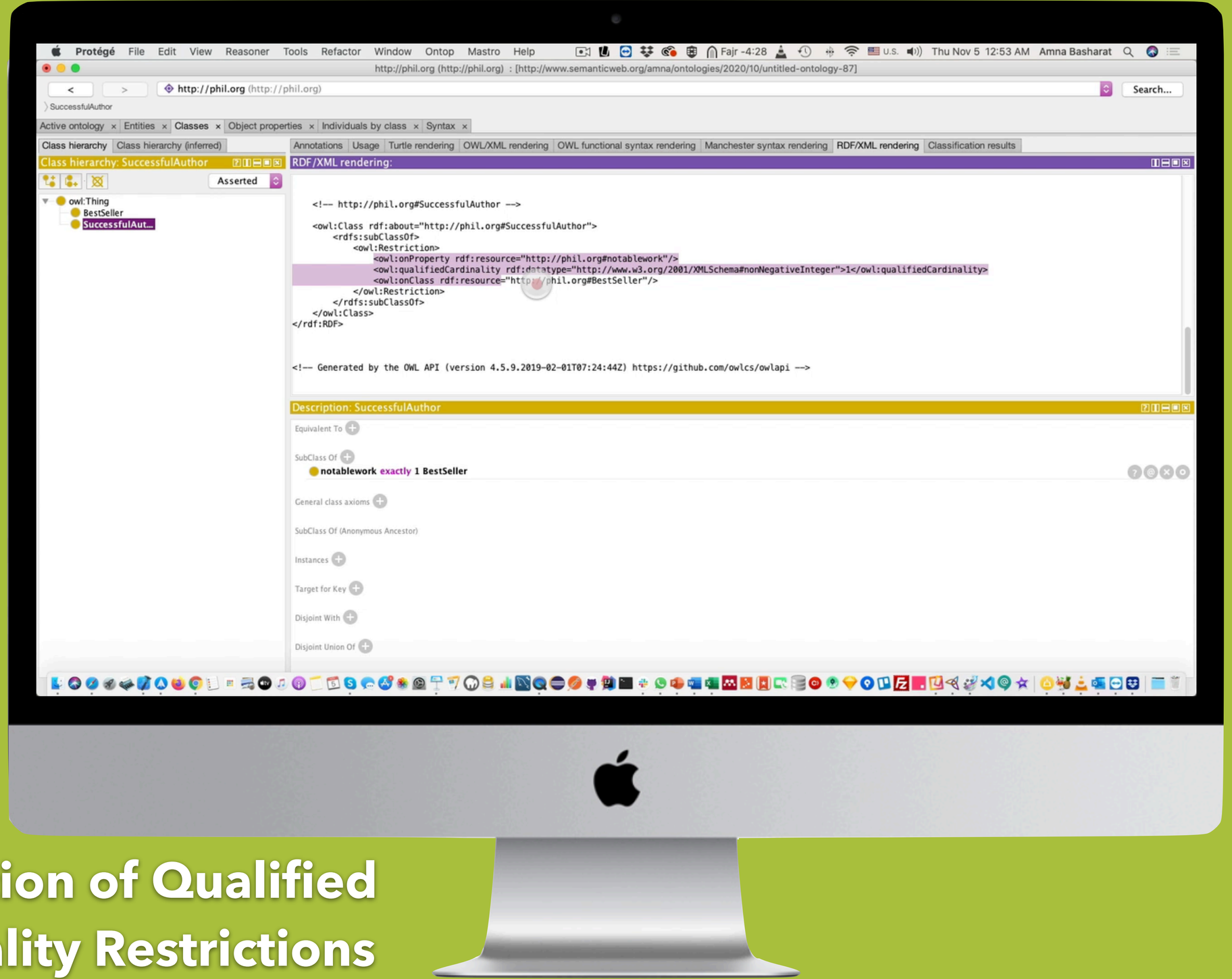


Illustration of Qualified Cardinality Restrictions

OWL Reflexive Property Restriction

- Classes that contain individuals that are related to themselves for specific properties

$$Philosopher \sqsubseteq knows.self$$

```
:Philosopher a owl:Class ;
    rdfs:subClassOf [
      a owl:Restriction ;
      owl:onProperty :knows ;
      owl:hasSelf "true"^^xsd:boolean ] .
```

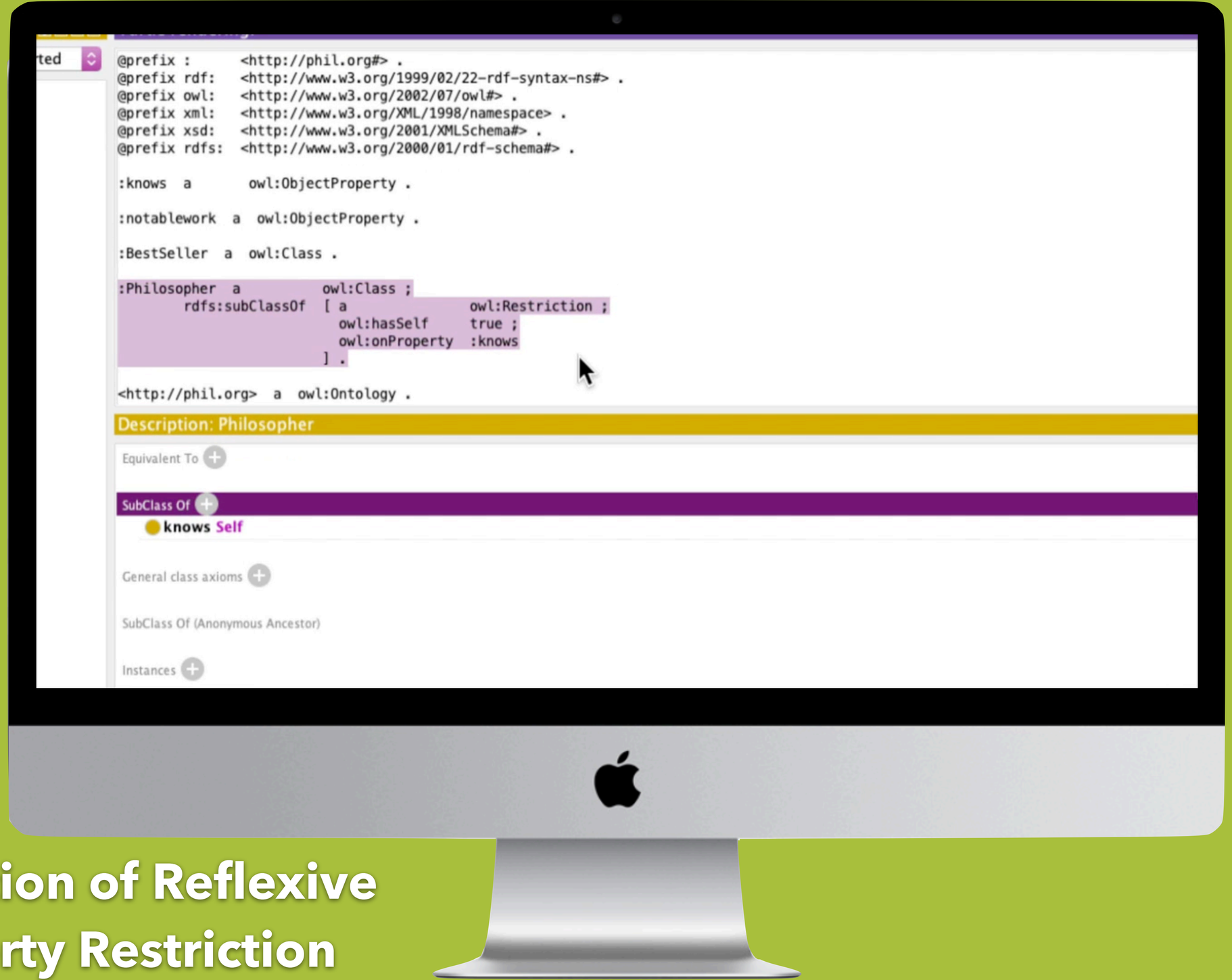


Illustration of Reflexive Property Restriction

Syntax Variation Examples

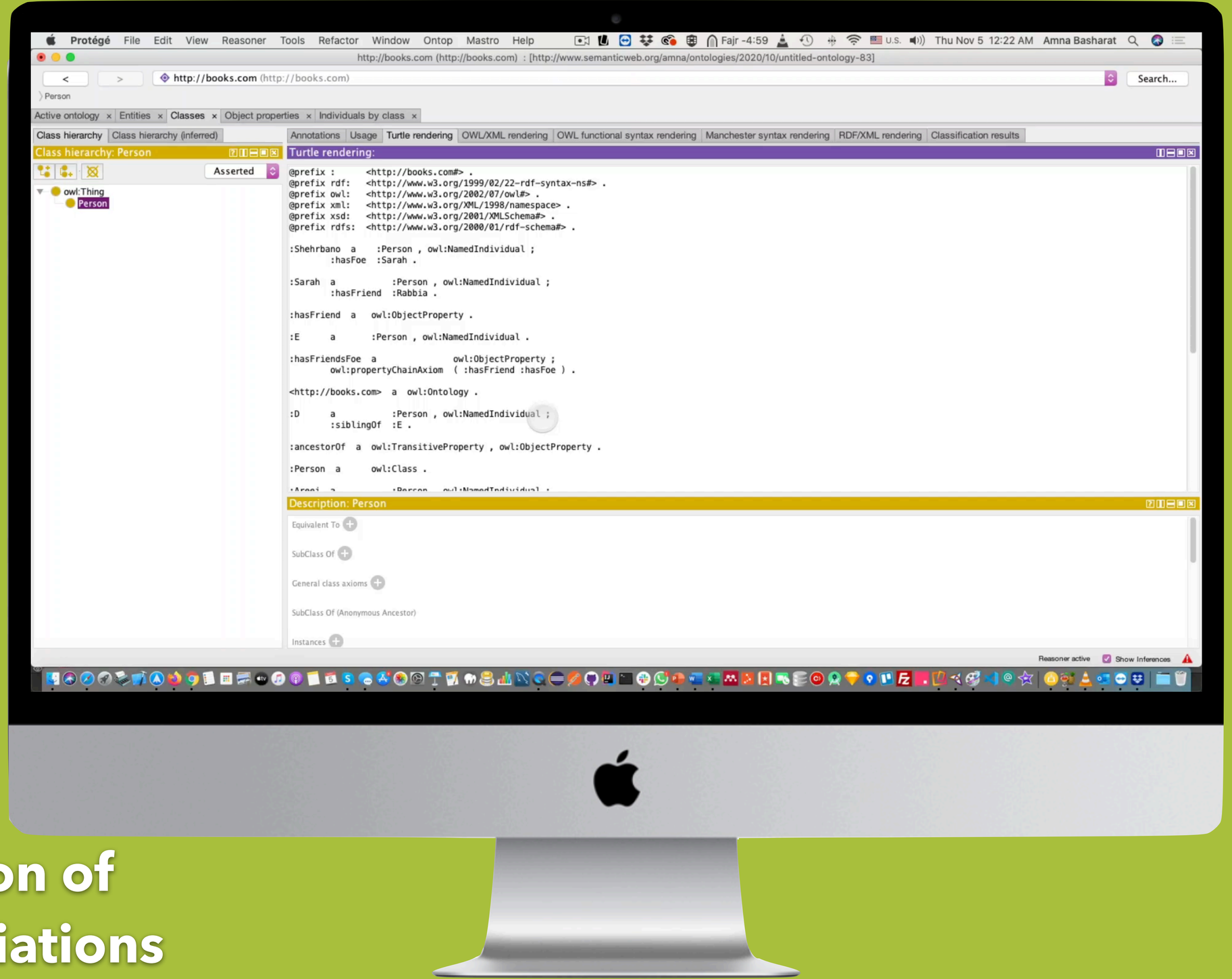


Illustration of Syntax Variations

Beyond OWL

Beyond OWL

- More **expressivity** also means more **complexity**
  - This might lead to **undecidability** (as for FOL)
- Do we really need more expressivity than OWL DL offers?
- Consider the following example:
  - ‘A squanderer is a person whose expenses are higher than his income’

- $\mathsf{Squanderer} \sqsubseteq \mathsf{Person}$
- $\mathsf{Squanderer} \sqsubseteq \mathsf{hasExpenses}.\top$
- $\mathsf{Squanderer} \sqsubseteq \mathsf{hasIncome}.\top$

?

- We need a constructor to combine Classes and Properties
- **Problem:** Mixing of TBox and ABox

Rules - Beyond OWL

- The following example can be expressed via a FOL-Rule:

  ‘A squanderer is a person whose expenses are higher than his income’

$$\text{Person}(x) \wedge \text{hasIncome}(x,y) \wedge \text{hasExpenses}(x,z) \wedge (z > y) \rightarrow \text{Squanderer}(x)$$

- Arithmetics can be part of rules and modelled like a predicate
  - $(z > y) \triangleq \text{greaterThan}(z,y)$