PURO: Providing Deeper Ontological Background to OWL ontologies and Linked Data Vocabularies (Introductory part)

LAAOS

Vojtěch Svátek, Martin Homola, Miroslav Vacura

Knowledge Engineering Group
University of Economics, Prague, CZ

Knowledge Representation Group
Comenius University, Bratislava, SK

Overview of this and subsequent presentations

- Motivations and overview of PURO (*Svátek*)
- PURO modelling primitives (*Homola*)
- Relationship to foundational ontologies (*Vacura*)

- B-Annot: Generic and Dataset-Specific Background Annotation of Linked Data Vocabularies (*Serra*)

- Typed Higher-Order Description Logics and its PURO Compliance (*Kľuka*)

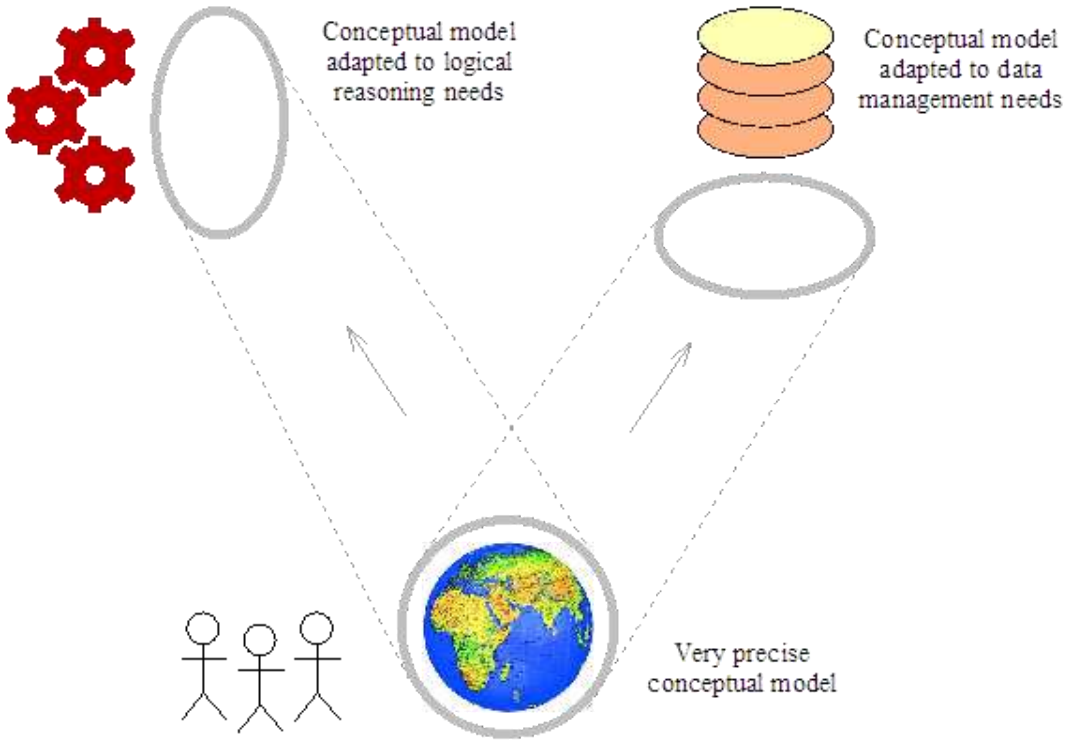
LAAOS and PURO

- The PURO model/methods is a major outcome of the 2-year Czecho-Slovak collaboration project LAAOS
  - Logical Aspects of Adaptable Ontological Schemas
- The project aims at intersection of multiple areas covered by the two collaborating groups (UEP, CU)
  - Ontological modelling
  - Description logics
  - Applied (philosophical) ontology
  - Linked data on the web
  - Semantic web tool design
  - Computational intelligence
- PURO relates to the first five areas
  - 2nd LAAOS sub-project is on using CI techniques (esp. Genetic Algorithms) for generation/selection of ontology transformation patterns aiming at ontology profiling

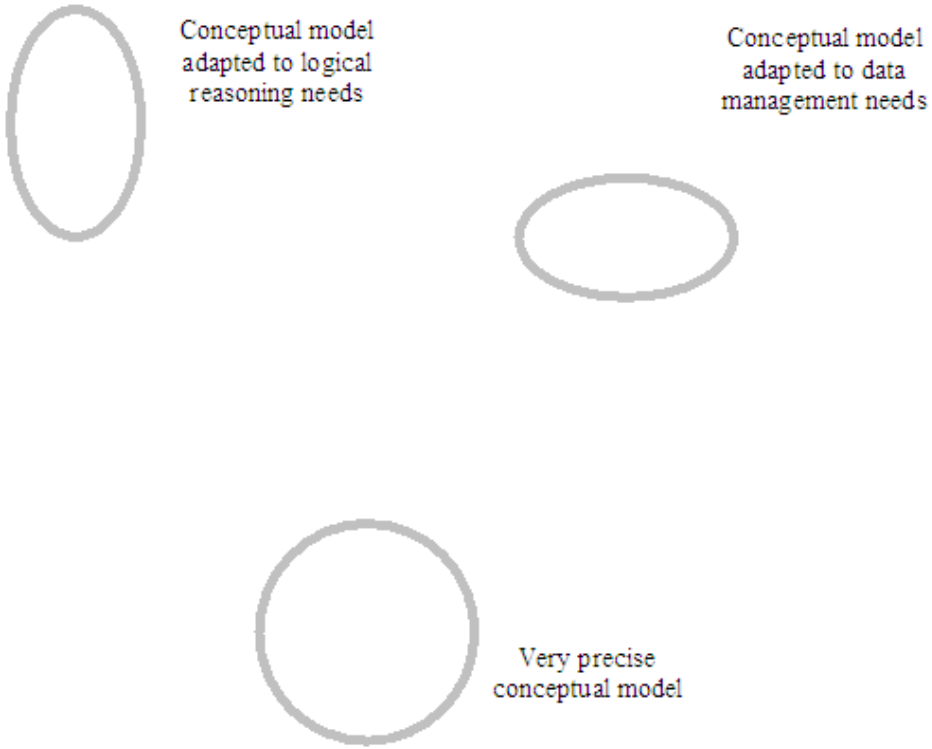
Evolution of ontology engineering

- **1990s:** few models, but rich and carefully crafted both in terms of logics and conceptualization
  - populated by AI knowledge bases
- **2000s:** many new models (thanks to W3C endorsement of OWL and its predecessors)
  - often rich T-box but no or little A-box
- **2010s:** an important subset of ontologies are linked data vocabularies, mostly populated by large data derived from non-semweb resources
  - i.e., large, distributed and heterogeneous A-box
  - usually simple T-box; schema tuned for easy management of data

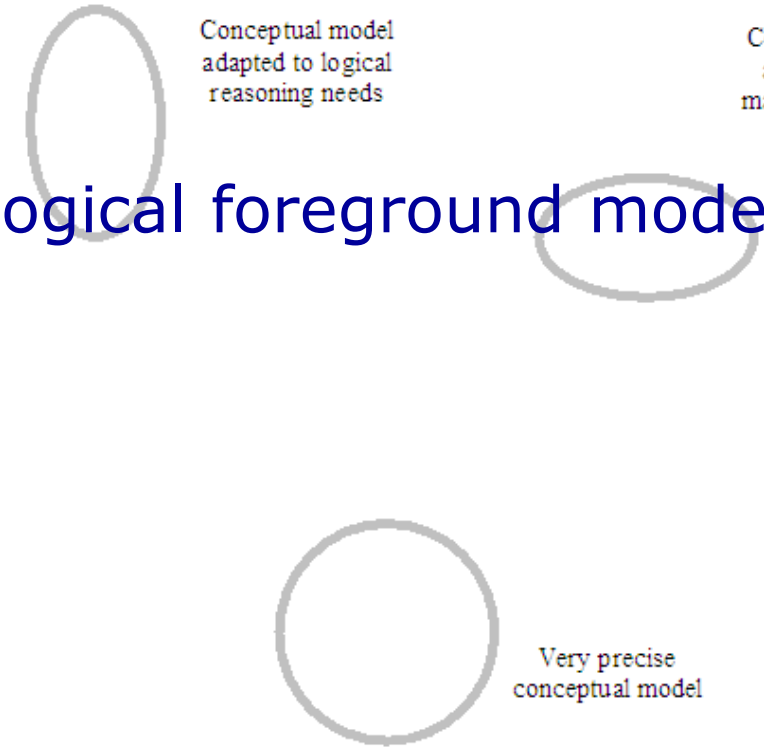
Conceptualization, logics and data management



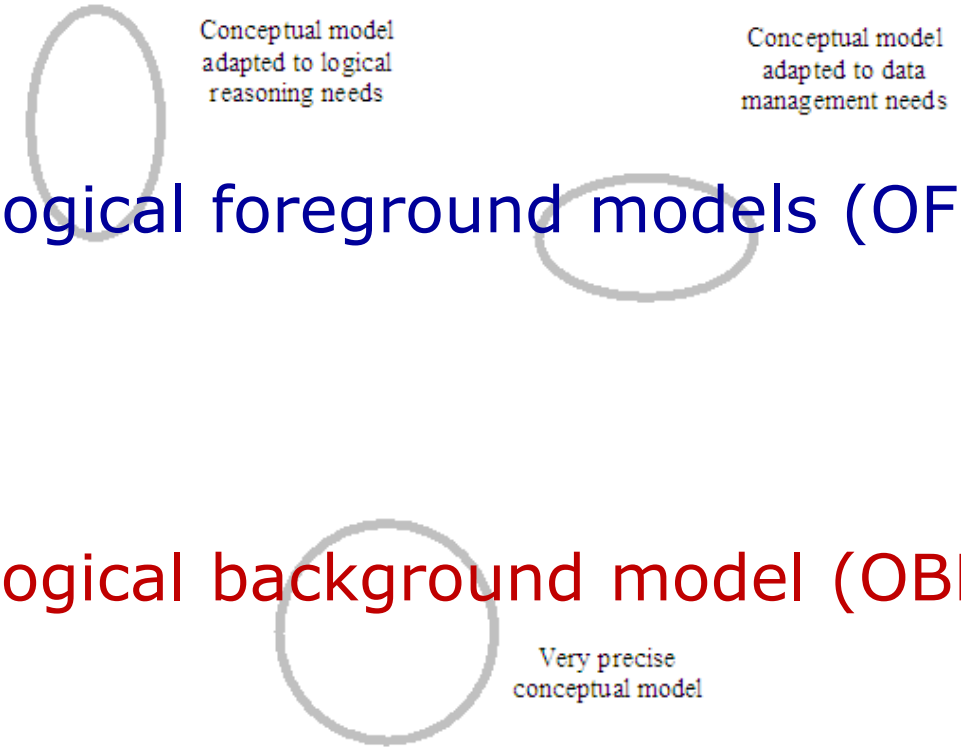
Conceptualization, logics and data management



Conceptualization, logics and data management



Conceptualization, logics and data management

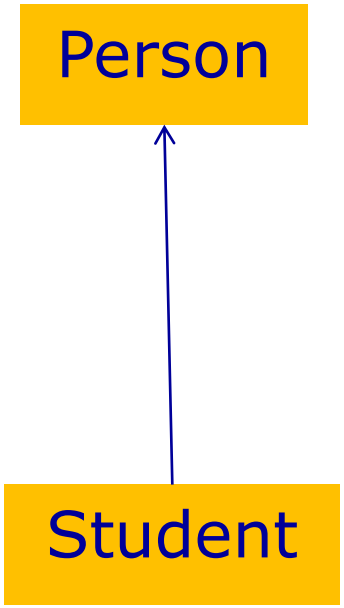


OFM / OBM Languages

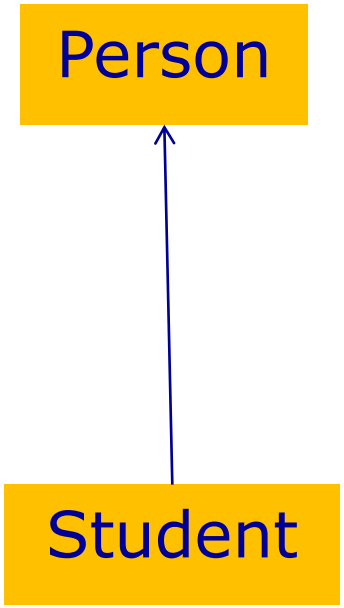
- Both kinds of models are expressed using a certain language, containing certain representation primitives
  - Ontological forground model language (OFML)
  - Ontological background model language (OBML)

- As OFML we can consider OWL
  - It subsumes simpler languages used for linked data, such as RDFS
- How about OBMLs?

OntoClean: OBML for taxonomy-centric ontologies



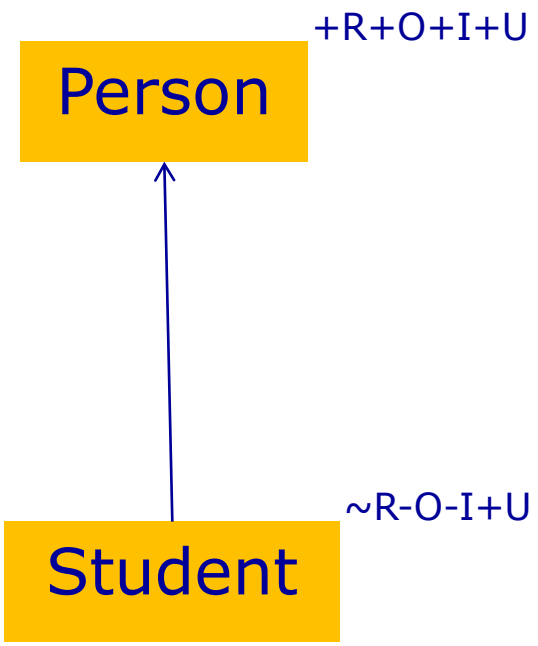
OntoClean: OBML for taxonomy-centric ontologies



- Rigid class
- Has own identity condition
- Has unity condition

- Anti-rigid class
- No identity condition
- Has unity condition

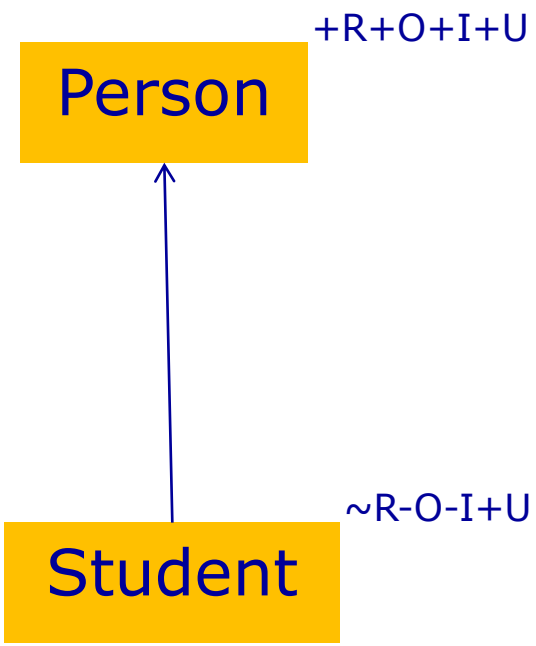
OntoClean: OBML for taxonomy-centric ontologies



- Rigid class
- Has own identity condition
- Has unity condition

- Anti-rigid class
- No identity condition
- Has unity condition

OntoClean: OBML for taxonomy-centric ontologies



- Rigid class
- Has own identity condition
- Has unity condition

- Anti-rigid class
- No identity condition
- Has unity condition

- Coherence checking: e.g., $P^{\sim R}$ can't subsume Q^{+R}

PURO: OBML for relationship-centric ontologies

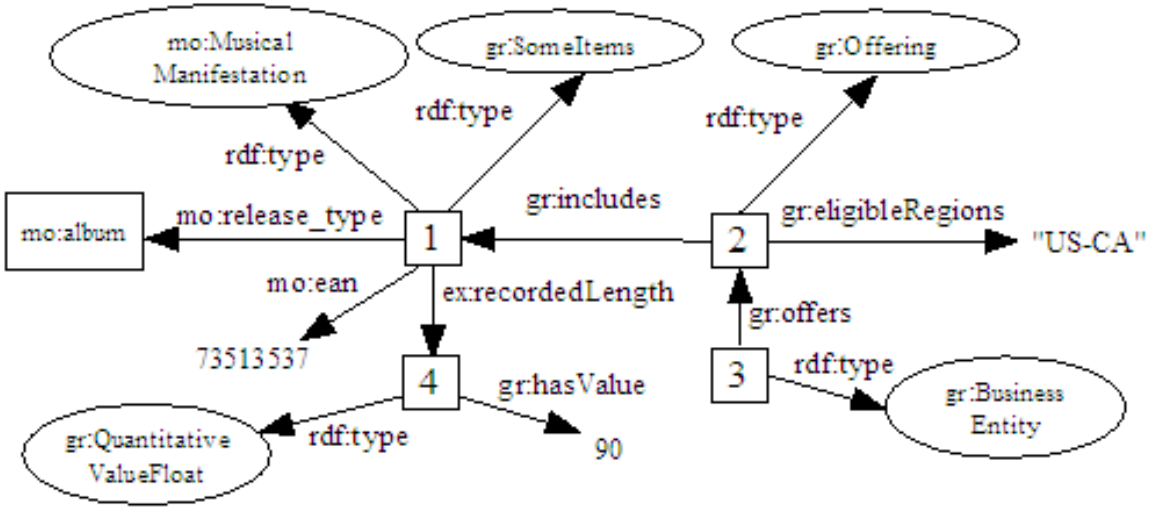
- PU: particulars vs. universals
  - or individuals (that cannot be instantiated) vs. types
- RO: relationships vs. objects

- Why these distinctions?
  - Relatively easy to understand
  - Together similar to the core OWL OFM inventory
    - ✓ individuals vs. classes; classes vs. properties

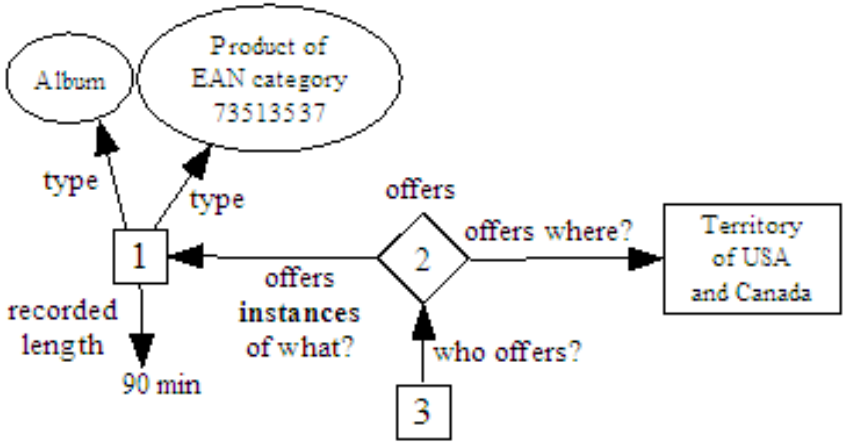
- But also challenges
  - The distinction is sometimes blurry
  - The structure of underlying OFMs is not uniform
  - The OBM structure is best manifested at the A-Box level, while the connection to OFM is required at the T-Box level

OFM vs. OBM („offering a music album“ example)

- OFM graph



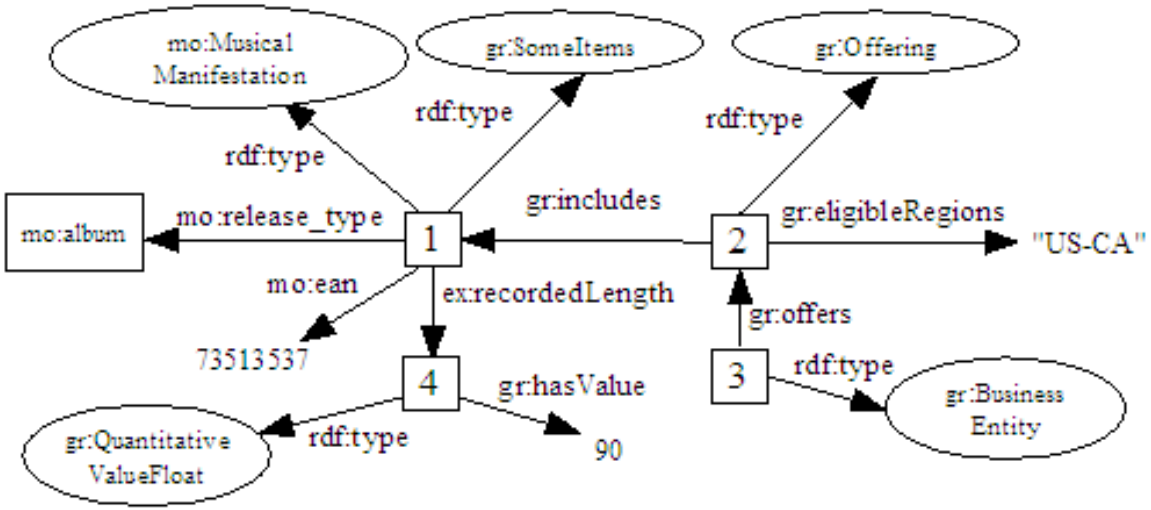
- PURO OBM graph



OFM vs. OBM („offering a music album“ example)

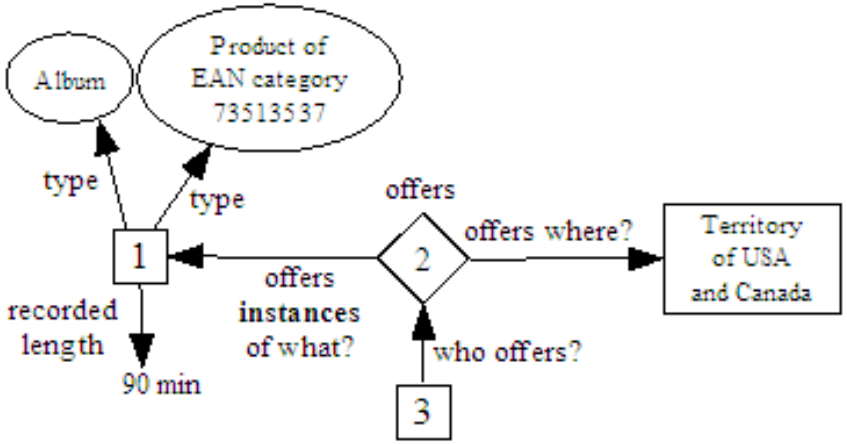
- OFM graph

  OWL A-Box



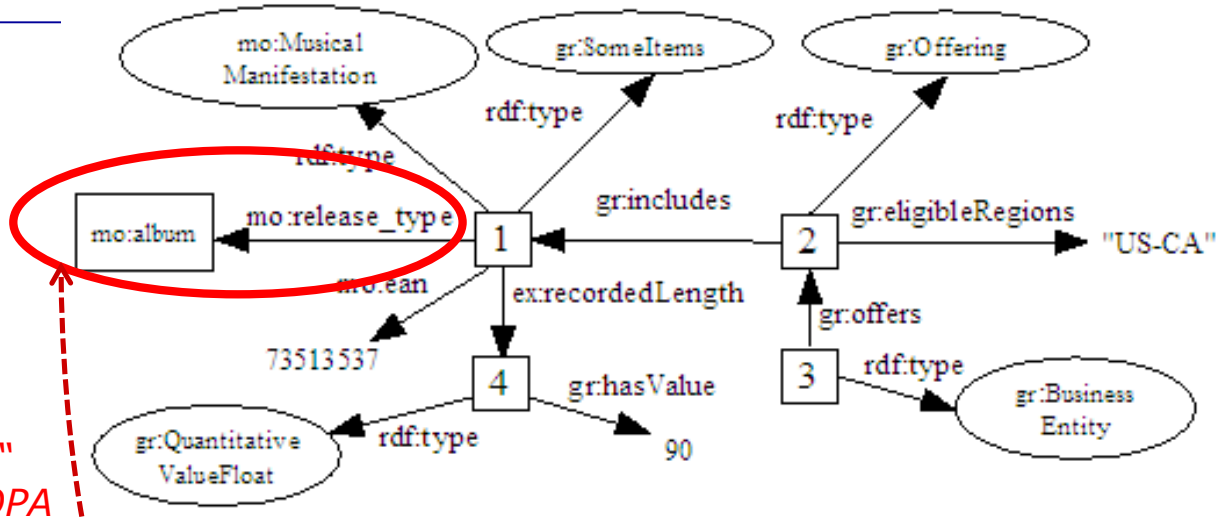
- PURO OBM graph

  Not OWL!



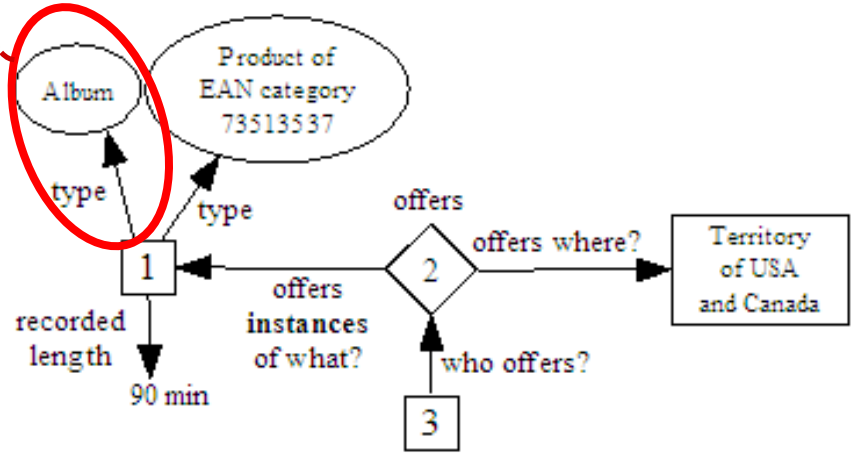
OFM vs. OBM („offering a music album“ example)

- OFM graph



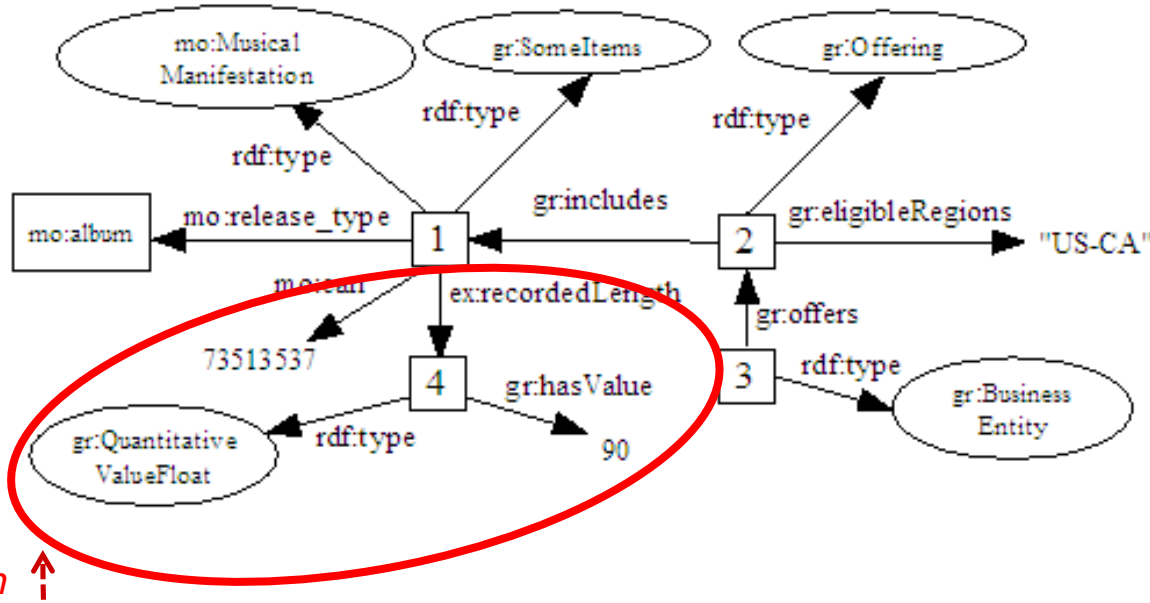
„Instantiation“ manifested as OPA

- PURO OBM graph



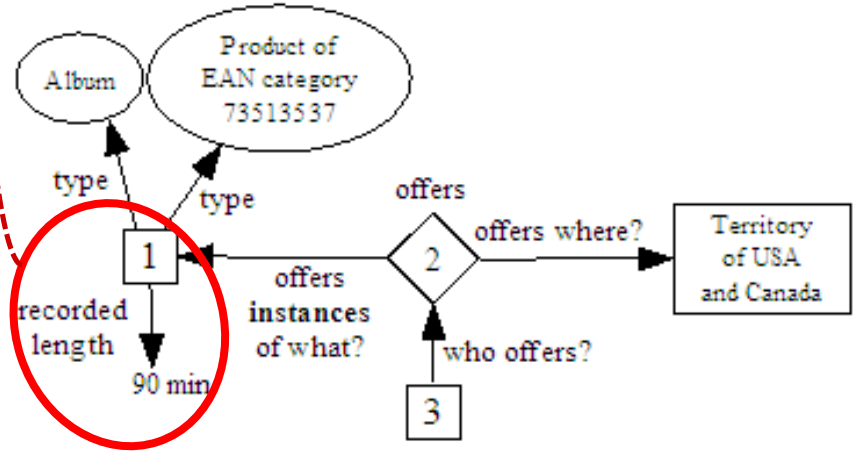
OFM vs. OBM („offering a music album“ example)

- OFM graph



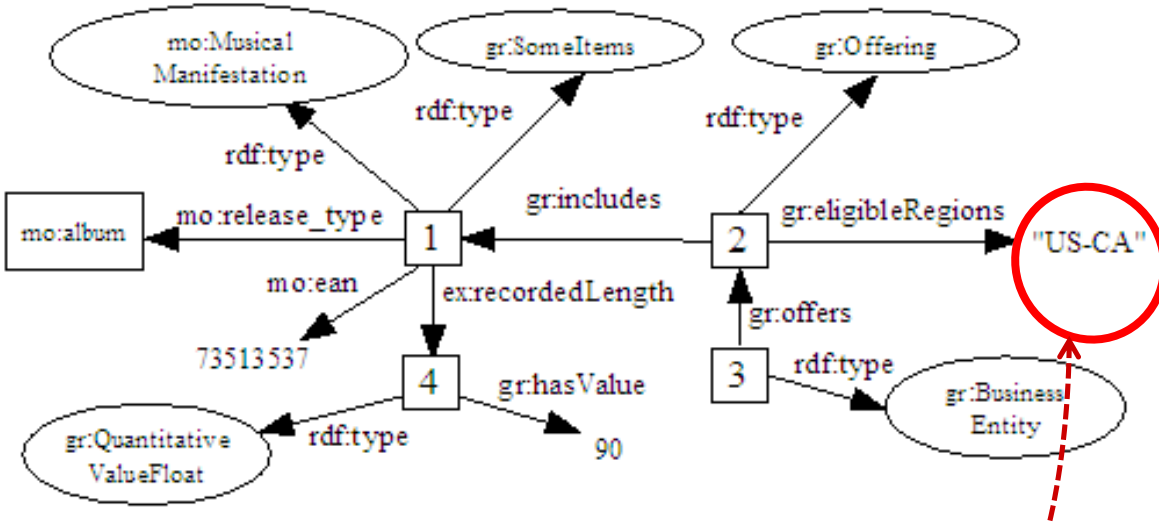
„Valuation“ manifested as chain of OPA+DPA

- PURO OBM graph



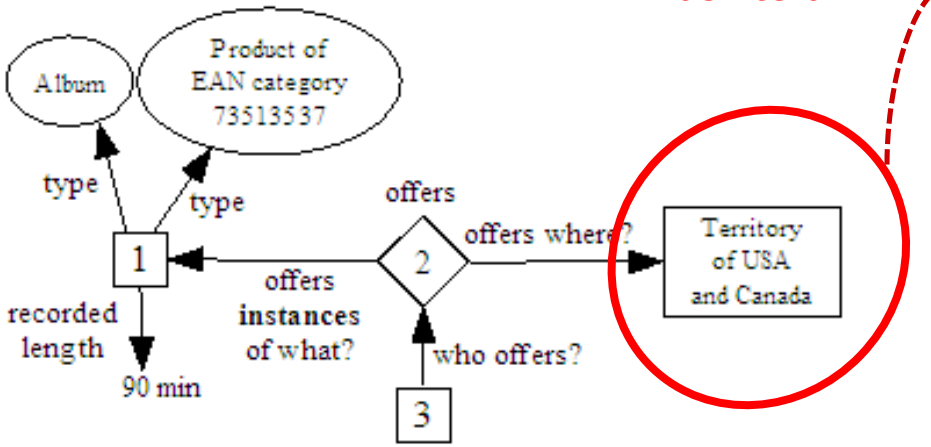
OFM vs. OBM („offering a music album“ example)

- OFM graph



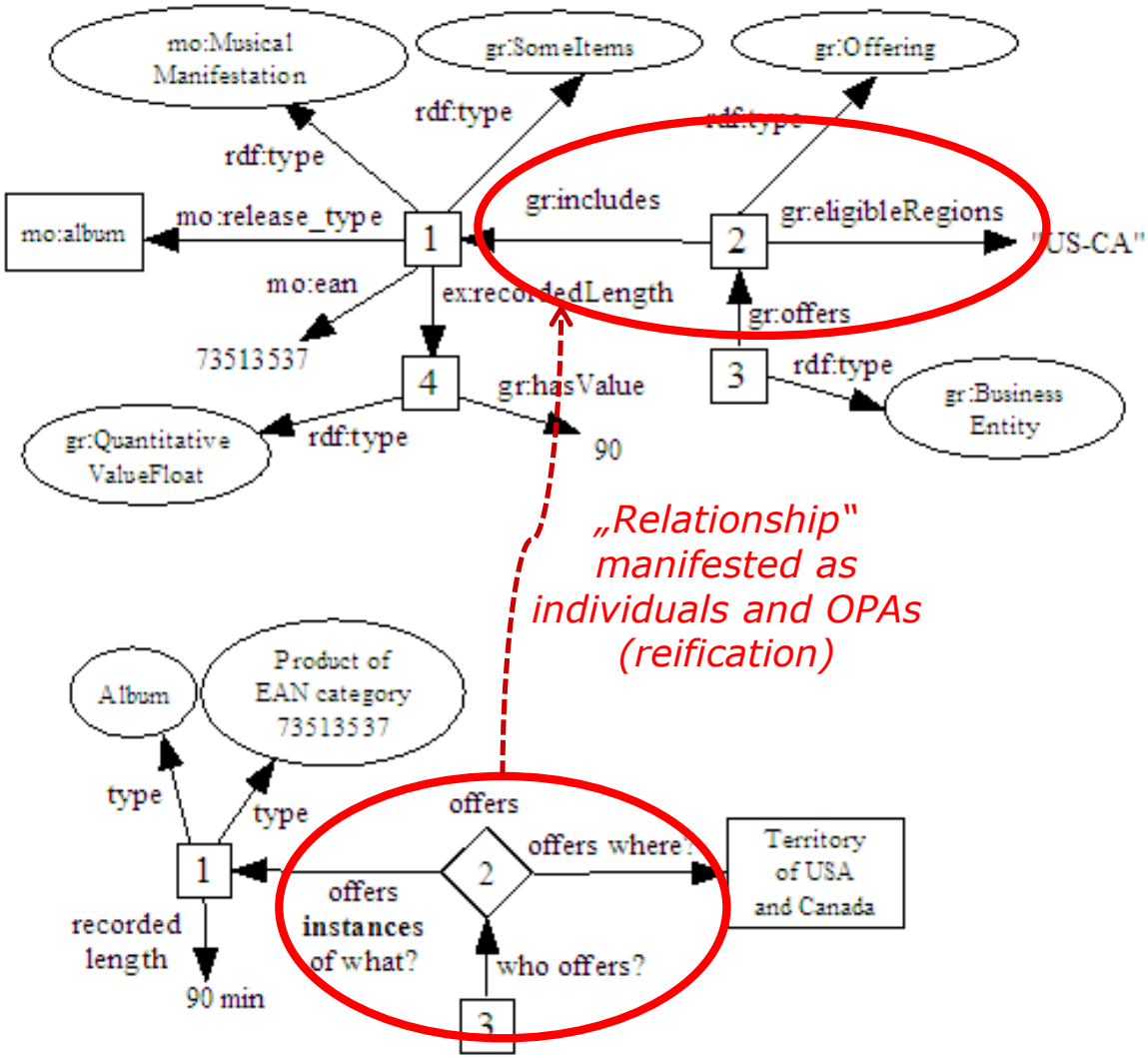
„Object“ manifested as literal

- PURO OBM graph



OFM vs. OBM („offering a music album“ example)

- OFM graph
- PURO OBM graph



Linking OFM and OBM

- OFM can contain links to OBM entities
  - Values of OWL annotation properties
- OBM and OFM can be both meta-modelled inside a common knowledge base
  - the T-box of which is a PURO ontology

Use cases for PURO annotations

- Checking the conceptual coherence
  - of a single vocabulary or of a vocabulary mapping
  - based on the annotations (OFM→OBM)
- Authoring of an OWL ontology starting from OBM
  - choice of OFM based on additional requirements of the designer (DL-enabled ontology, LD vocabulary, HTML-centric vocabulary, …)
- Alternative visualizations of an ontology
- Unifying framework for several logical/structural patterns
- Extraction of 'concise bounded descriptions' (CBD) from RDF data

Ongoing and future work

- Linking the PURO meta-model to higher-order logic
- Annotation experiments for multiple LD vocabularies
- Visualization of PURO OBMs

Thanks for your attention

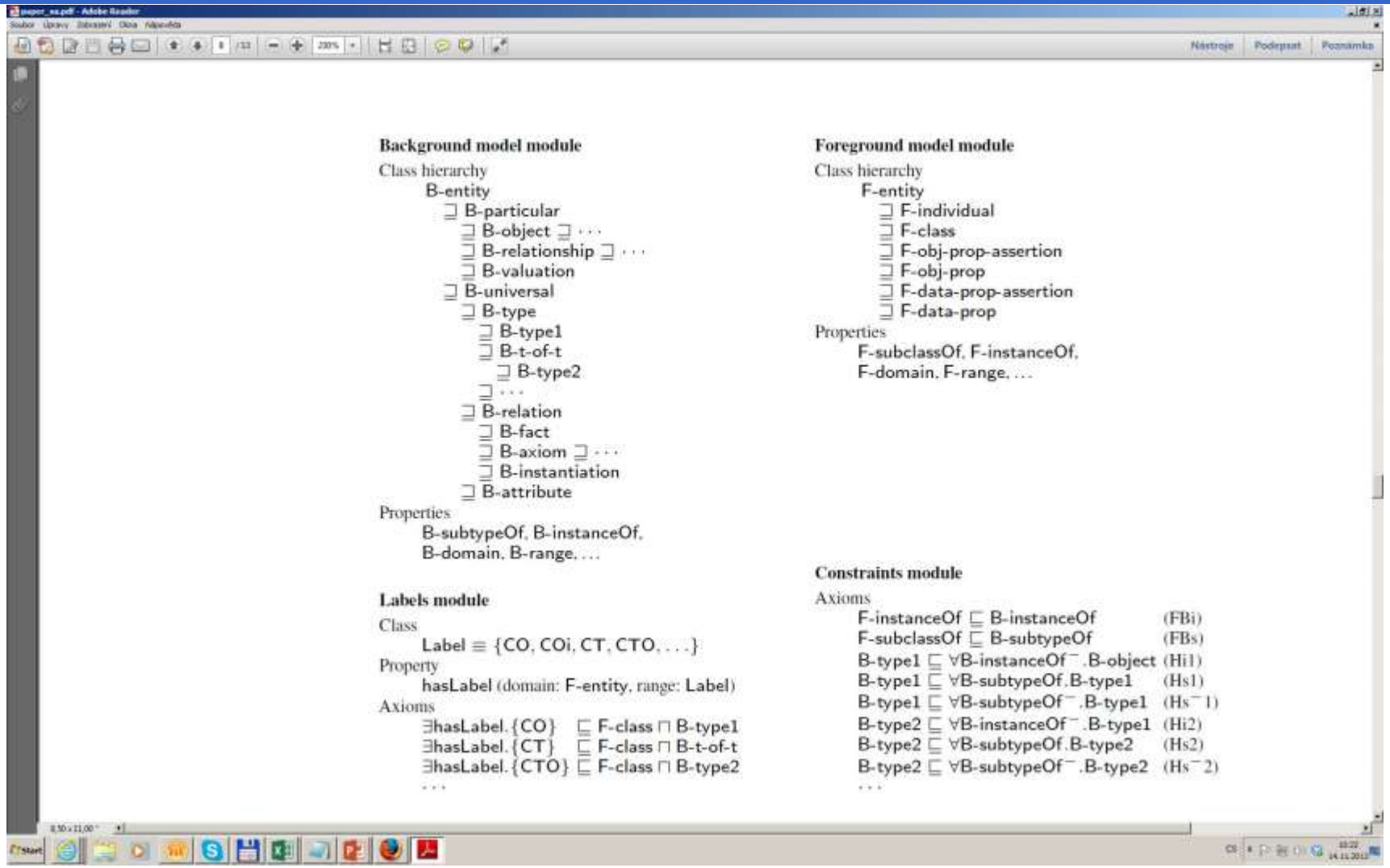
(further talks to follow)

LAAOS

Conceptual coherence checking process

- Annotation of the vocabulary with PURO primitives
- Meta-modelling of the vocabulary entities
  - using individuals: instances of concepts from the PURO (meta-)ontology
- Actual coherence checking, by means of DL consistency checking
  - For P-U incoherency, the meta-modelling individual of an incoherent class becomes instance of a specific 'diagnostic' class

PURO ontology



Background model module

Class hierarchy

- B-entity
  - ⊒ B-particular
    - ⊒ B-object ⊒ ···
    - ⊒ B-relationship ⊒ ···
    - ⊒ B-valuation
  - ⊒ B-universal
    - ⊒ B-type
      - ⊒ B-type1
      - ⊒ B-t-of-t
        - ⊒ B-type2
      - ⊒ ···
    - ⊒ B-relation
      - ⊒ B-fact
      - ⊒ B-axiom ⊒ ···
      - ⊒ B-instantiation
    - ⊒ B-attribute

Properties

B-subtypeOf, B-instanceOf, B-domain, B-range, ...

Labels module

Class

Label ≡ {CO, COi, CT, CTO, ...}

Property

hasLabel (domain: F-entity, range: Label)

Axioms

∃hasLabel.{CO} ⊑ F-class ⊓ B-type1
∃hasLabel.{CT} ⊑ F-class ⊓ B-t-of-t
∃hasLabel.{CTO} ⊑ F-class ⊓ B-type2
...

Foreground model module

Class hierarchy

- F-entity
  - ⊒ F-individual
  - ⊒ F-class
  - ⊒ F-obj-prop-assertion
  - ⊒ F-obj-prop
  - ⊒ F-data-prop-assertion
  - ⊒ F-data-prop

Properties

F-subclassOf, F-instanceOf, F-domain, F-range, ...

Constraints module

Axioms

F-instanceOf ⊑ B-instanceOf	(FBi)
F-subclassOf ⊑ B-subtypeOf	(FBs)
B-type1 ⊑ ∀B-instanceOf⁻.B-object	(Hi1)
B-type1 ⊑ ∀B-subtypeOf.B-type1	(Hs1)
B-type1 ⊑ ∀B-subtypeOf⁻.B-type1	(Hs⁻1)
B-type2 ⊑ ∀B-instanceOf⁻.B-type1	(Hi2)
B-type2 ⊑ ∀B-subtypeOf.B-type2	(Hs2)
B-type2 ⊑ ∀B-subtypeOf⁻.B-type2	(Hs⁻2)
...	