Constructs Replacing and Complexity Downgrading via a Generic OWL Ontology Transformation Framework

Ondřej Šváb–Zamazal, Vojtěch Svátek
University of Economics, Prague, CZ

Anne Schlicht, Heiner Stuckenschmidt
University of Mannheim, Germany

[ondrej.zamazal@vse.cz]

Agenda

- Context and motivations
- (Ontology) Transformation patterns
- Transformation workflow and implementation
- Language Profiling Scenario
  - Construct replacement
  - Complexity downgrading
- Experiment
- Ongoing and future work

Agenda

- Context and motivations
- (Ontology) Transformation patterns
- Transformation workflow and implementation
- Language Profiling Scenario
  - Construct replacement
  - Complexity downgrading
- Experiment
- Ongoing and future work

Context: PatOMat project

- Semantic Web ontology language (OWL) enables to describe the same “state of affairs” in many different ways (by various formal means)
- This creates “heterogeneity of styles” issue
- Ontological structures corresponding to different modelling style variants can be captured as patterns
- Fragments of ontologies can be automatically transformed so as to still reflect the same “state of affairs”

Examples of different modelling style variants

AcceptedPaper SubClassOf: Paper.
RejectedPaper SubClassOf: Paper.
AcceptedPaper DisjointWith: RejectedPaper.

accepts Domain: PCChair.
rejects Domain: PCChair.
accepts DisjointWith: rejects.

accepts Range: Paper.
rejects Range: Paper.

hasPCChairDecision Domain: Paper.
hasPCChairDecision Range: (EquivalentTo {acceptance, rejection}).
hasPCChairDecision Characteristics: FunctionalProperty.

hasPCChairDecision Domain: Paper.
Acceptance SubClassOf: Decision.
Acceptance DisjointWith: Rejection.

hasPCChairDecision Range: Decision.
Rejection SubClassOf: Decision.

Context: PatOMat motto

„Let's make ontologies metamorphic”

- Metamorphosis in nature: the same individual but in different form
- Ontology Metamorphosis: “state of affairs” behind the ontology remains the same but modelling style is different

Context: PatOMat project

- Types of transformation use-cases:
  - Modelling-style Transformation applied on correct ontology
  - Repairing Transformation applied on incorrect ontology

Context: PatOMat project

- Types of transformation use-cases:
  - Transformation of modelling-style applied on correct ontology
    - ✓ Simple context-less modelling-style transformation, e.g. transformation of ontology taxonomies to thesaurus taxonomies (OWL → SKOS and vice versa)
    - ✓ Ontology Transformation into a modelling style enabling better matching of two ontologies
    - ✓ Ontology Transformation into a modelling style enabling smooth importing of content ontology design pattern

Context: PatOMat project

- Types of transformation use-cases:
  - Repairing Transformation applied on incorrect ontology
    - ✓Repairing of entity naming along a taxonomy

PCChair SubClassOf: ProgramCommittee.

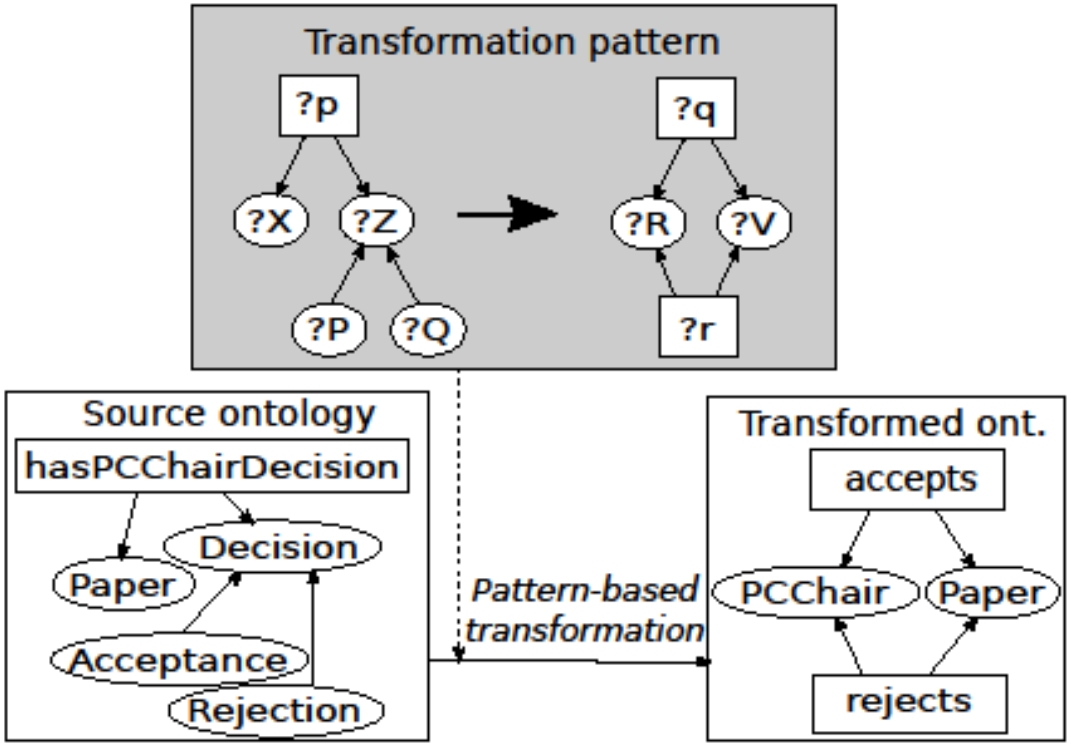
Accepted SubClassOf: Paper.

InvitedTalk SubClassOf: Presentation

Agenda

- Context and motivations
- (Ontology) Transformation patterns
- Transformation workflow and implementation
- Language Profiling Scenario
  - Construct replacement
  - Complexity downgrading
- Experiment
- Ongoing and future work

Basic shape of transformation



PatOMat and patterns

- Alternative modelling styles are captured via (logical/structural) ontology patterns: OWL structures (mostly) containing placeholders instead of real entities
  - source OP
  - target OP
- Transformation of (occurrences of) one OP into another is defined by a transformation pattern
  - namely, in its pattern transformation (PT) part
- Both ontology patterns and transformation patterns may contain naming patterns with linguistic grounding
  - naming detection patterns
  - naming transformation patterns

Agenda

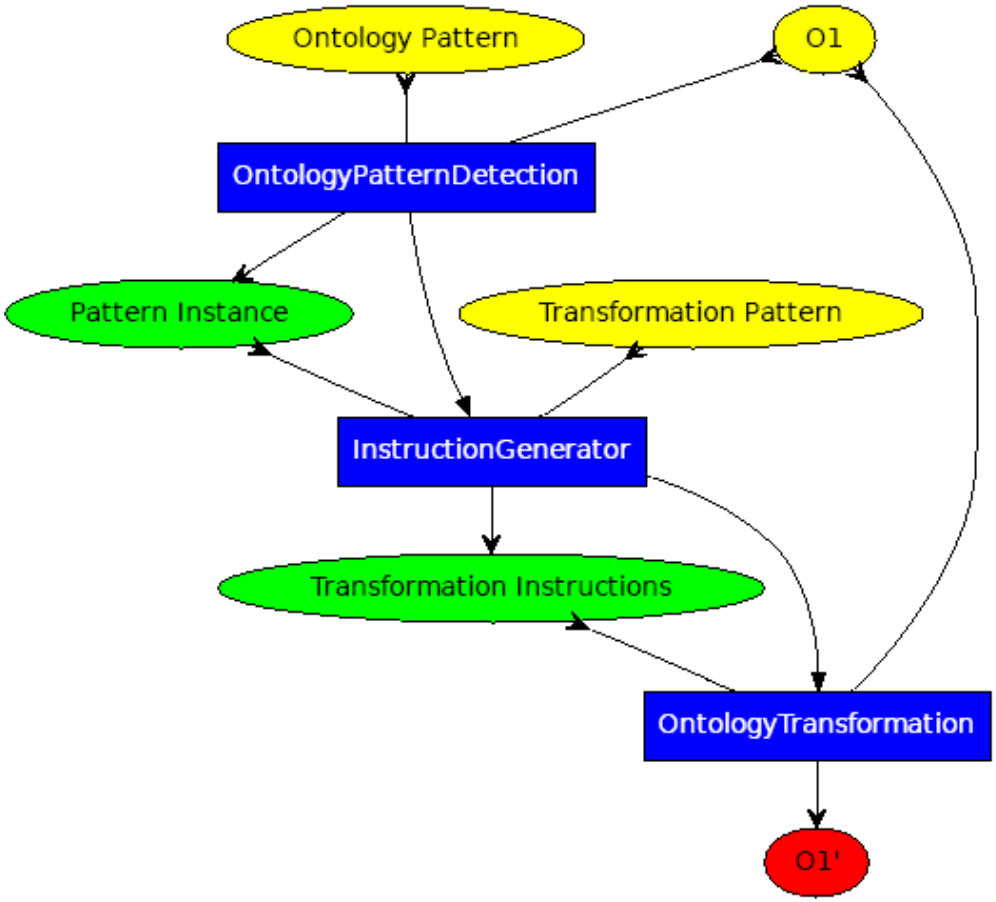
- Context and motivations
- (Ontology) Transformation patterns
- Transformation workflow and implementation
- Language profiling scenario
  - Construct replacement
  - Complexity downgrading
- Experiment
- Ongoing and future work

Prototype implementation

- Three-phase transformation
  - detection of source pattern in ontology
  - generation of transformation instructions
    - ✓ instantiation of the transformation part of the pattern
  - actual transformation
    - ✓ using OWL-API
- The user can interact in each step

- *Services available via POST method at http://owl.vse.cz:8080*
- *Tutorial available http://owl.vse.cz:8080/tutorial/*

Pipeline of RESTful Services



Agenda

- Context and motivations
- (Ontology) Transformation patterns
- Transformation workflow and implementation
- Language Profiling Scenario
  - Construct replacement
  - Complexity downgrading
- Experiment
- Ongoing and future work

Language profiling scenario

- Motivation: generic means how to replace forbidden or unsupported constructs outside of some tool
- Advantage: such a transformation can be re-used over many tools (not hard-coded there)

Language profiling scenario

- In comparison with other use-cases: fully automatic
- Two phases:
  - An analysis of the source ontology → which transformations should be applied
  - Dynamic composition of selected transformation patterns in a sequence
  - Post-processing for ensuring completeness
- Three possible transformations:
  - equivalent replacement,
  - approximate replacement
  - removing

1st use-case: Construct replacement

- Task: replacing a specific language construct
  - e.g. removing nominals (enumerated classes)

```
Continent equivalentTo {Africa, America,
Antarctica, Asia, Australia, Europe}.
```

```
AfricanRedSlip subClassOf
(hasContinentOfOrigin value Africa).
```

Construct replacement

- Solution #1: removing of nominals
  - But then we would lose part of the description

~~Continent equivalentTo {Africa, America, Antarctica, Asia, Australia, Europe}.~~

~~AfricanRedSlip subClassOf (hasContinentOfOrigin value Africa).~~

Construct replacement

- ## Solution #2: replacing of nominals

```
Continent equivalentTo {Africa, America, Antarctica,
Asia, Australia, Europe}.
```

→

```
OneOfContinent equivalentTo (Africa_nc or America_nc or
Antarctica_nc or Asia_nc or Australia_nc or Europe_nc).
Africa a Africa_nc. ...
```

```
AfricanRedSlip subClassOf (hasContinentOfOrigin value
Africa).
```

→

```
AfricanRedSlip subClassOf (hasContinentOfOrigin some
Africa_nc).
```

http://nb.vse.cz/~svabo/patomat/tp/lr/tp_nominals-6a.xml

Agenda

- Context and motivations
- (Ontology) Transformation patterns
- Transformation workflow and implementation
- Language Profiling Scenario
  - Construct replacement
  - Complexity downgrading
- Experiment
- Ongoing and future work

Complexity downgrading

- Task: downgrading according to ontology complexity requirement
- E.g.: to OWL2EL profile
- Forbidden constructs: universal quantifications to a class expression, cardinality restrictions, class negations, enumerations, disjoint properties,...
- We can replace: complement of universal restriction, minimum cardinality, enumerations

Complexity downgrading – minimum cardinality

```
Ax1: AcceptedPaper subClassOf (hasDecision min 2
Acceptance).
→
AcceptedPaper subClassOf (hasDecision some
Acceptance).

EvaluatedPaper = hasDecision some Decision.
Acceptance subClassOf Decision.
Ax2: AcceptedPaper subClassOf EvaluatedPaper.
```

Ax2 is preserved if Ax1 is replaced and not removed.

Complexity downgrading – enumerations

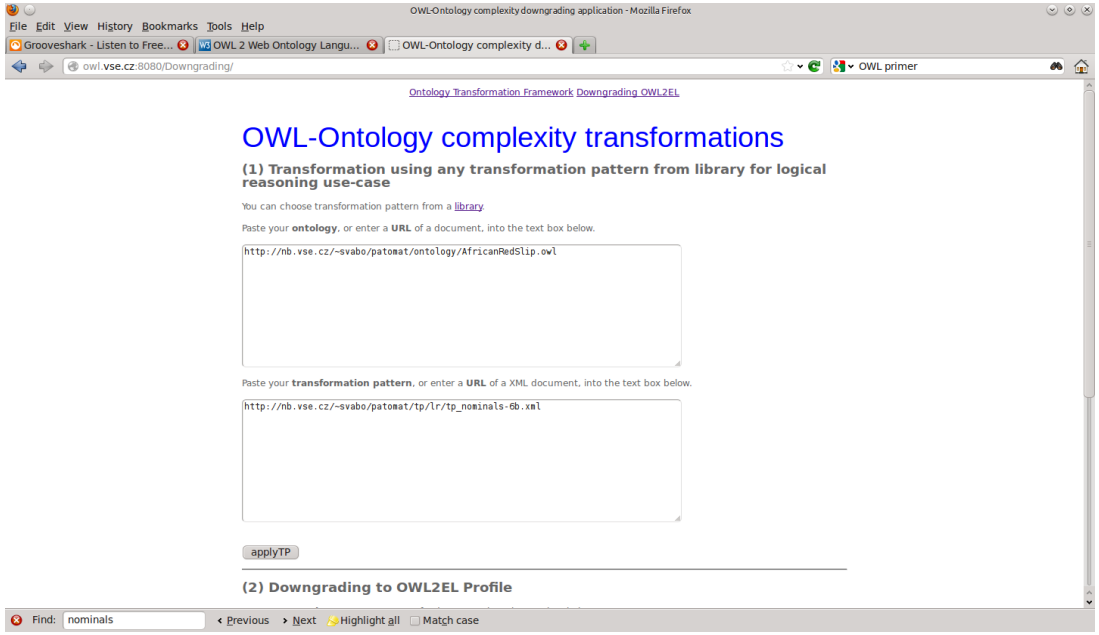
```
Ax1: EurAsia = {europe, asia}.
→
Europe_nc = { europe }. Asia_nc = { asia }.
Europe_nc subClassOf EurAsia
Asia_nc subClassOf EurAsia

EuropeanWatch = ( hasContinentOfOrigin hasValue
europe )
EurAsiaWatch = ( hasContinentOfOrigin some
EurAsia )
Ax2: EuropeanWatch subClassOf EurAsiaWatch.
```

Ax2 is preserved if Ax1 is replaced and not just removed.

Complexity downgrading – application

http://owl.vse.cz:8080/Downgrading/



Agenda

- Context and motivations
- (Ontology) Transformation patterns
- Transformation workflow and implementation
- Language Profiling Scenario
  - Construct replacement
  - Complexity downgrading
- Experiment
- Ongoing and future work

Complexity downgrading - experiment

- Comparison of effects: replacement vs. removal
- Effect measured as number of preserved subsumption relations (just one aspect)
  - Using query answering by SPARQL:

    ASK Class1 SubClassOf: Class2
- Ontology collection gathered by Watson search engine:
  - Criteria: OWL language, >10 classes, >5 properties, wo imports
  - 328 → 63 → 38 ontologies

Complexity downgrading - experiment

O variant means original ontologies

R variant goes from removal transformation

	number of ontologies
no difference between O and R variants	17
no positive effect wrt. saved subsumption relations	13
saved subsumptions due to simple modifications	7
saved subsumptions due to replacement transformations	1

Limitations of this experiment:

- Current small set of problematic issues are covered
- Positive effect can only be measured if there are also further axioms which enable derivation of subsumption relations

Advantage for future:

- Any newly designed transformation pattern can be plugged in

Agenda

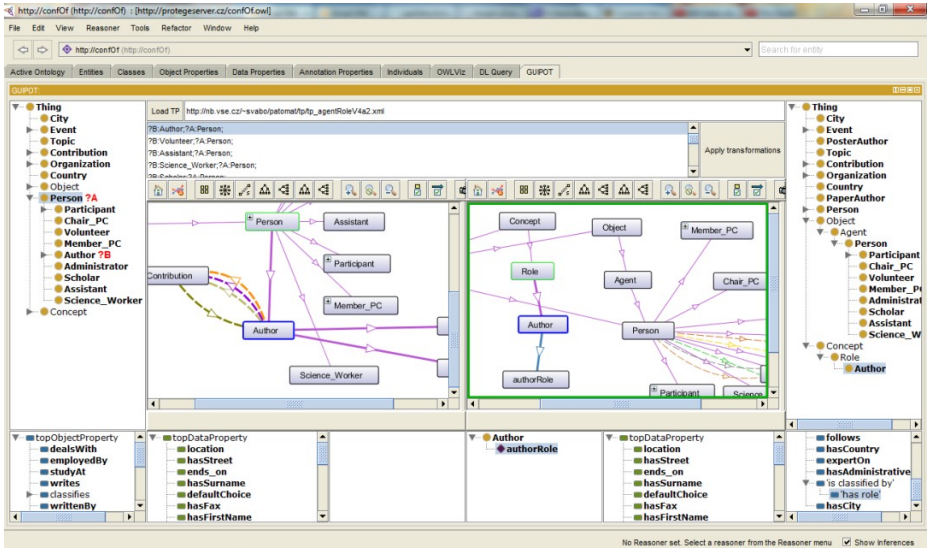
- Context and motivations
- (Ontology) Transformation patterns
- Transformation workflow and implementation
- Language Profiling Scenario
  - Construct replacement
  - Complexity downgrading
- Experiment
- Ongoing and future work

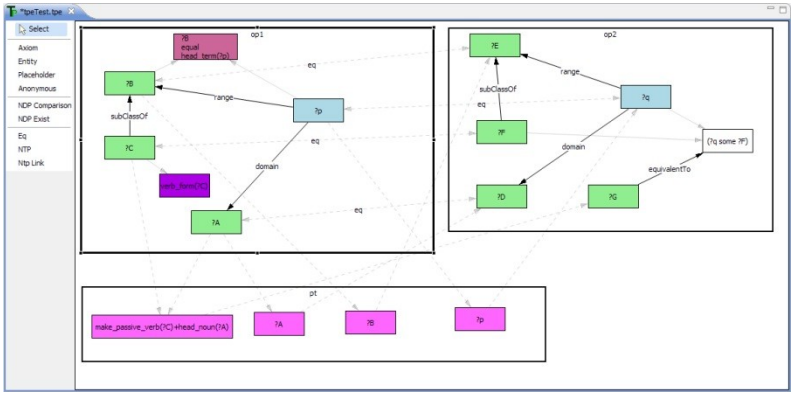
Ongoing and future work

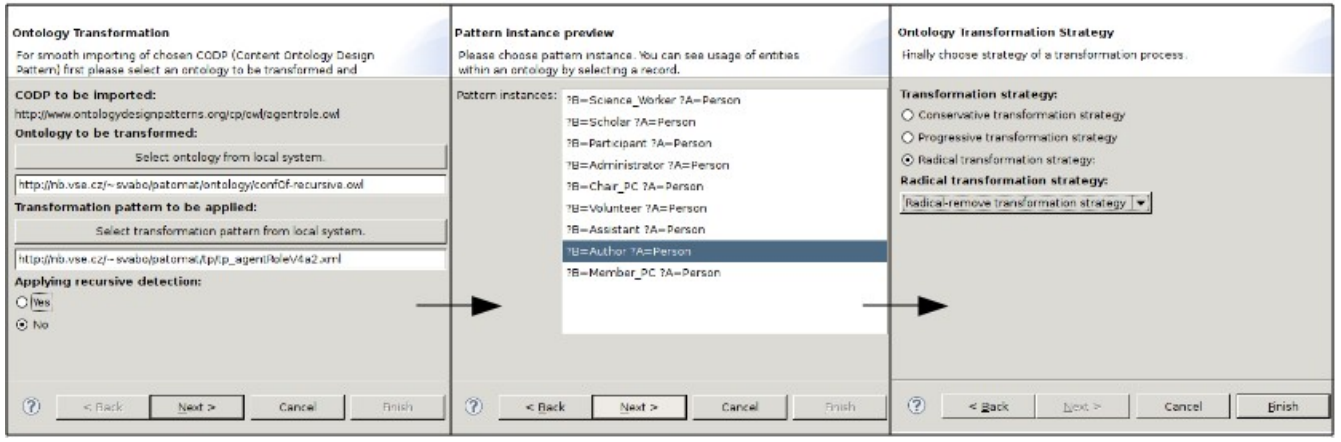
- Automatic generation of TPs and downgrading
- More advanced experiment also wrt. 1st use-case

- Comprehensive library of naming patterns relevant for ontology style transformation
  - Implementation on top of existing lexical sources
- Canonical methods for swapping info between logical and annotation spaces while transforming
- Ontologies of logical/structural patterns
  - Patterns structure; categorisation facets
  - Patterns usage, esp. matching to modelling issues
- Data-driven ontology transformation
  - other CPs; matching settings; reasoning settings

Applications for ontology transformation







Best Demo Award at EKAW 2012.

THANKS FOR YOUR ATTENTION