PatOMat ontology transformation framework in gallop: inventory check and experiments plans

Ondřej Šváb-Zamazal and Vojtěch Svátek

University of Economics, Prague
Department of Information and Knowledge Engineering
ondrej.zamazal@vse.cz

March 1, 2011

Agenda

- Introduction
- Use Cases: Ontology Matching and Import Scenario
- Content Pattern Import: AgentRole pattern
- PatOMat Ontology Transformation Framework
- Ontology Transformation Workflow
- Future Work

Introduction

Ontology transformation means a process of (semi-)automated transformation of relevant parts of an ontology into a different, in some sense more suitable, 'shape' or modelling style.
It can help in many different tasks such as:

- *Ontology Matching* - creating alignments between ontologies may be difficult due to heterogeneity of styles
- *Reasoning* - some features of ontologies cause performance problems for certain reasoners
- *Importing* - import is quite difficult when the source and target ontology are modelled using different styles, i.e. **adaptation of the source ontology is need**

Motivating example

Example (Different modelling styles)

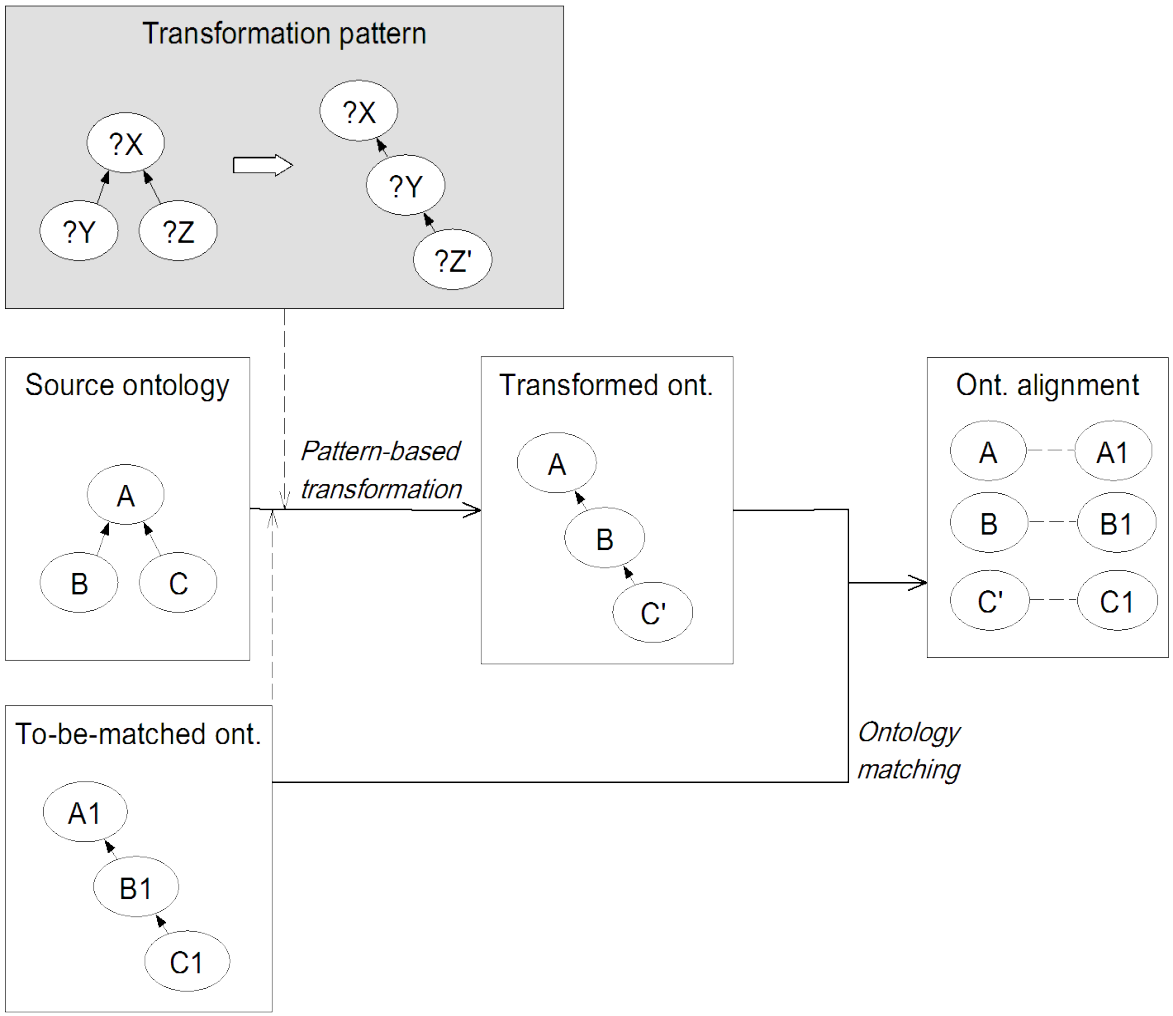
In one ontology, the possibility of an acceptance or rejection of paper can be expressed by concepts

- using **siblings**: (*PaperAcceptanceAct/PaperRejectionAct SubClassOf: ReviewerAct*).
- In another ontology it can be captured with **object properties**: (*Reviewer accepts/rejects Paper*).
- Another possibility is the use of **enumerations**: (*Paper reviewerDecision acceptance/rejection*)
- and another possibility is using **open class**:
  - *hasDecision Domain: Paper. hasDecision Range: Decision*
  - *Acceptance SubClassOf: Decision*
  - *Rejection SubClassOf: Decision*

Use Cases

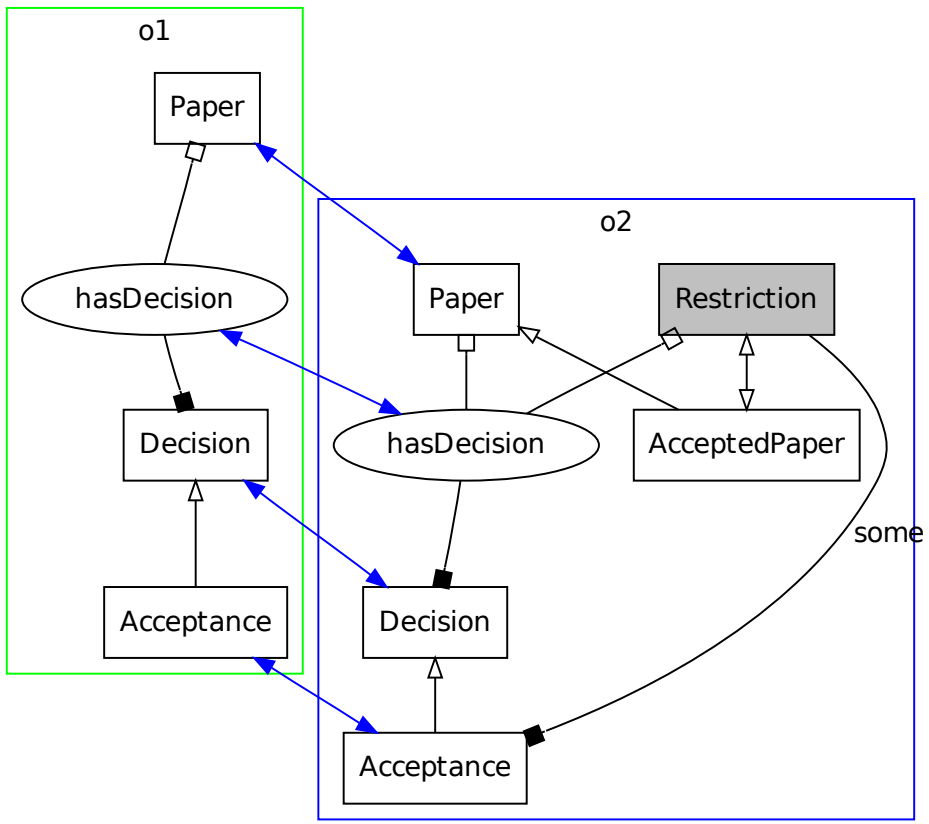
1. Matching of style-wise heterogeneous ontologies
   - It is alternative to building complex 'Mannheim-style' correspondences o2#AcceptedPaper = o1#hasDecision some o1#Acceptance
2. Solving structural problems when importing an ontology into another
   - currently investigated for content patterns (CPs) from the OntologyDesignPatterns.org

Transformation for Matching Scenario



Example of Transformation Pattern within Ontology Matching context

Matching two ontologies: ekaw and cmt. Cmt is transformed using transformation pattern tp_hasSome2.

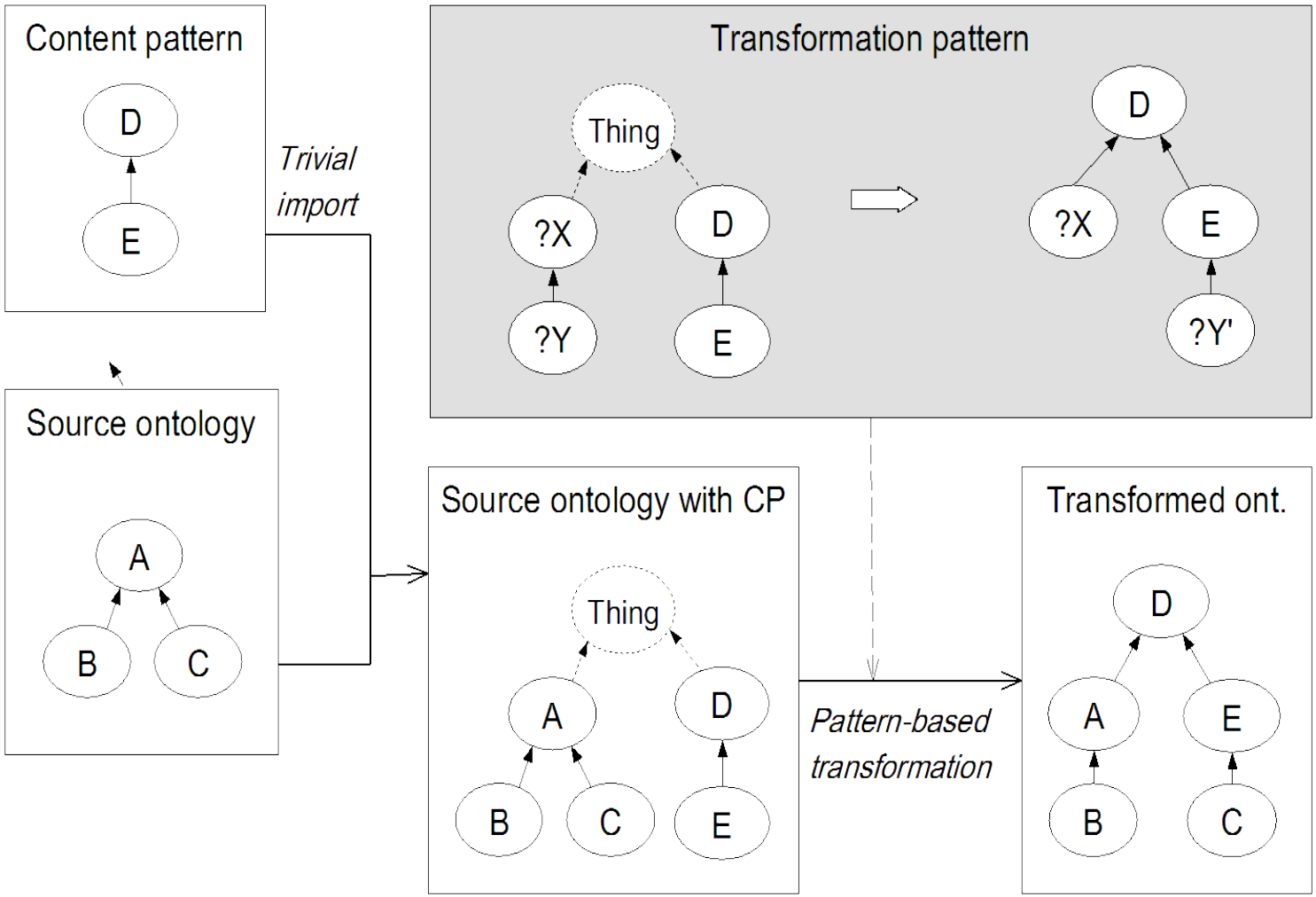


Ontology transformation cont'd - matching effect

1. cmt ontology enriched with new entity: AcceptedPaper: `AcceptedPaper equivalentTo: (hasDecision some Acceptance)`
2. Applying string-based matching technique we can get: `cmt#AcceptedPaper=ekaw#Accepted_Paper`
3. This can be used for getting complex correspondence: `(cmt#hasDecision some cmt#Acceptance) = ekaw#Accepted_Paper`

This corresponds with instance of 'Class by Attribute Type Pattern' from Scharffe's work. This demonstrates a benefit stemming from ontology transformation within ontology matching context.
See at `http://owl.vse.cz:8080/tutorial/`

Content pattern importing scenario



Example: Importing AgentRole content pattern

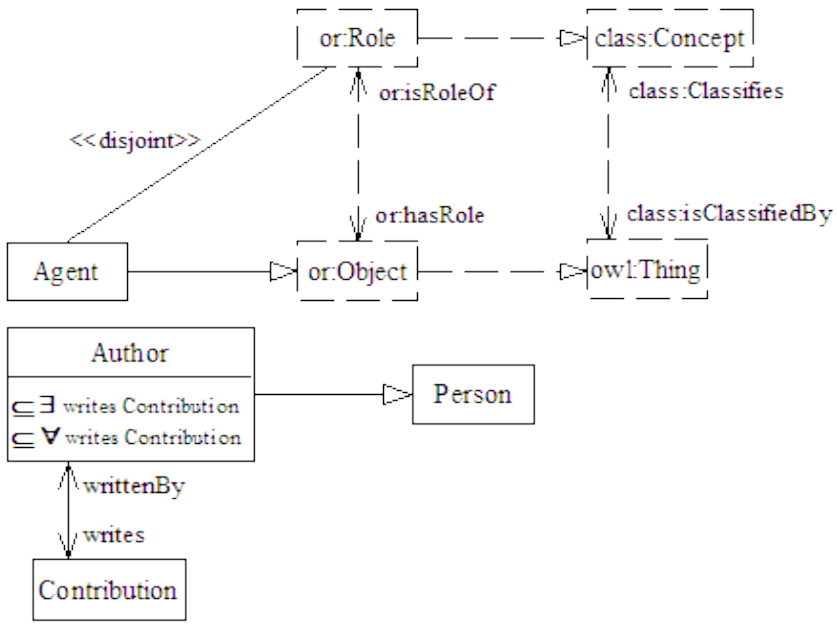
Case study: Importing AgentRole CP into confOf ontology

AgentRole (with own imports)

- OntologyDesignPatterns.org

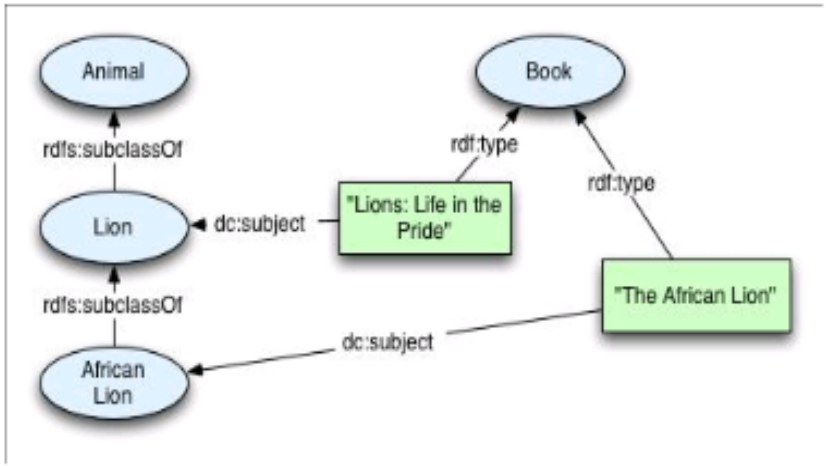
(Fragment of) ConfOf ontology from OntoFarm collection

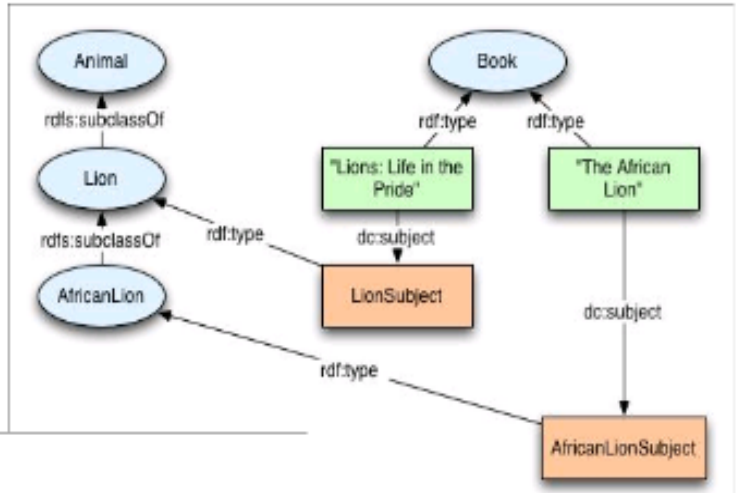
- modelling the 'conference organisation' domain

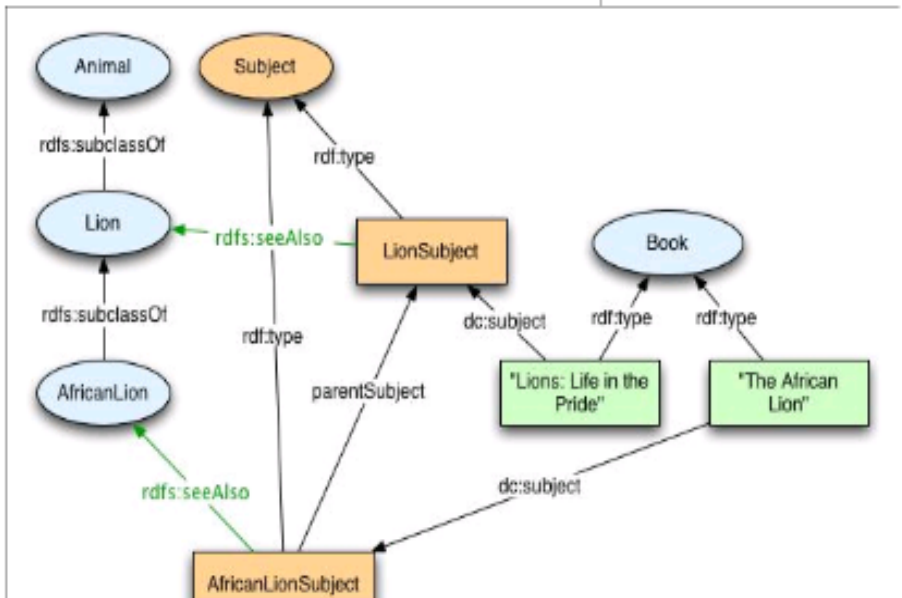


Need for adaptation: we should be able to say that a person *has the role* of author (rather than just 'is author')

Approaches for Classes as Property Values

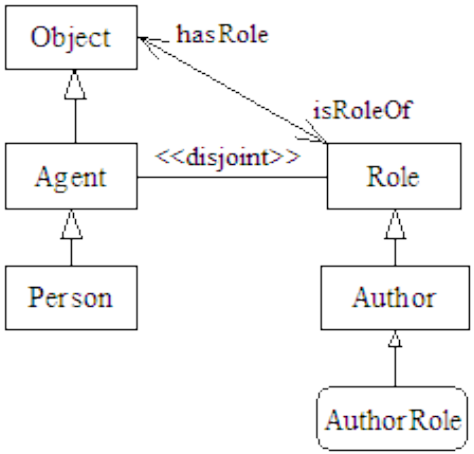




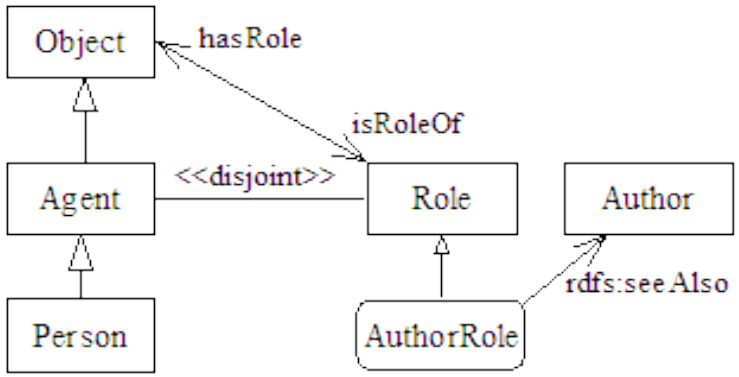


Target Patterns for AgentRole import

Approach 2: Create special instances of the class to be used as property values



Approach 3: Create a parallel hierarchy instances as property values



Corresponding transformation patterns are at `http://nb.vse.cz/~svabo/patomat/tp/`.

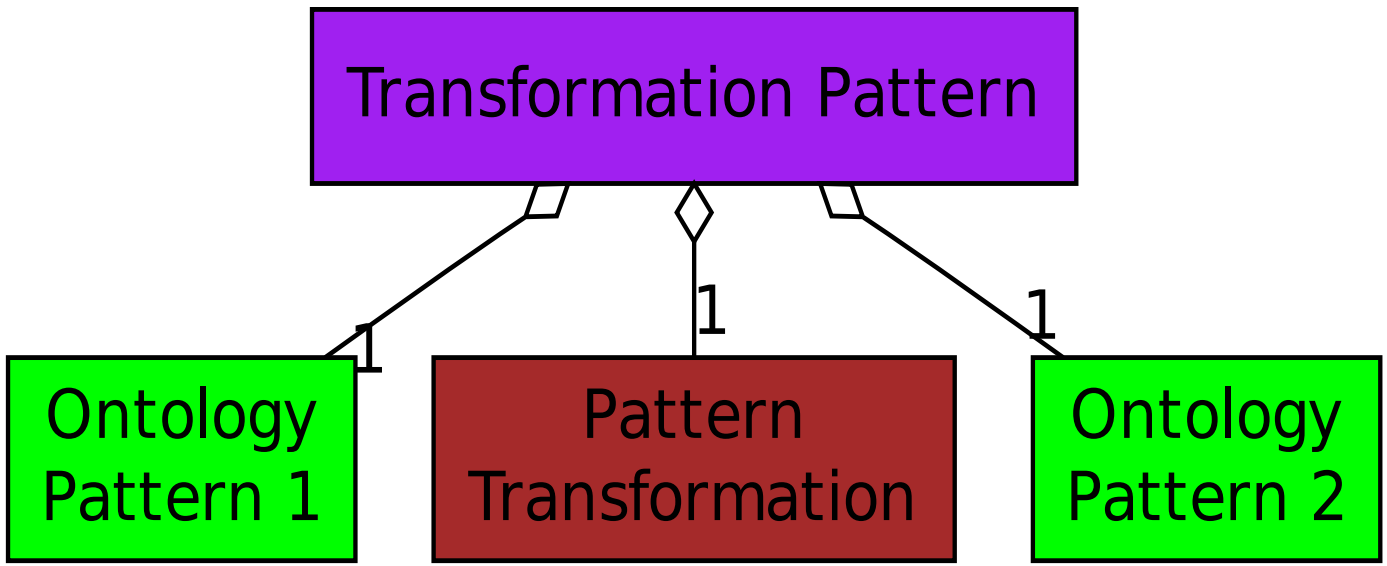
PatOMat Transformation Framework

Central notion is a *transformation pattern*.

- 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* can contain *naming patterns*

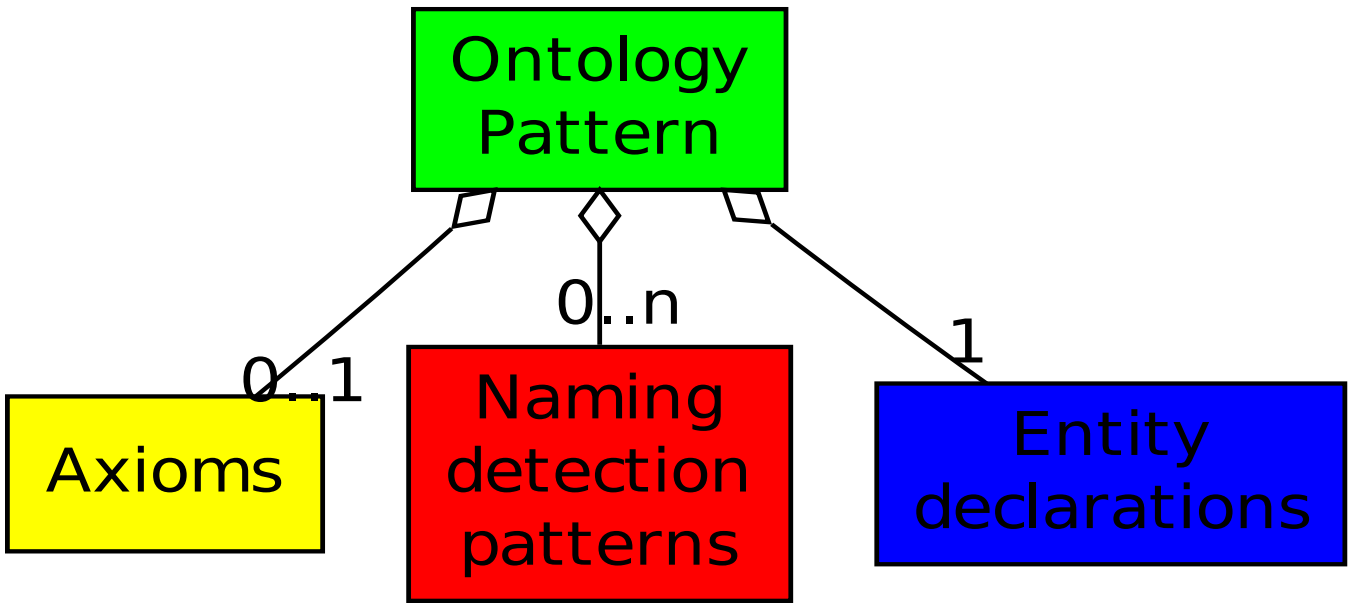
Transformation Pattern Representation

Transformation Pattern



Transformation Pattern Representation cont'd

Ontology Pattern



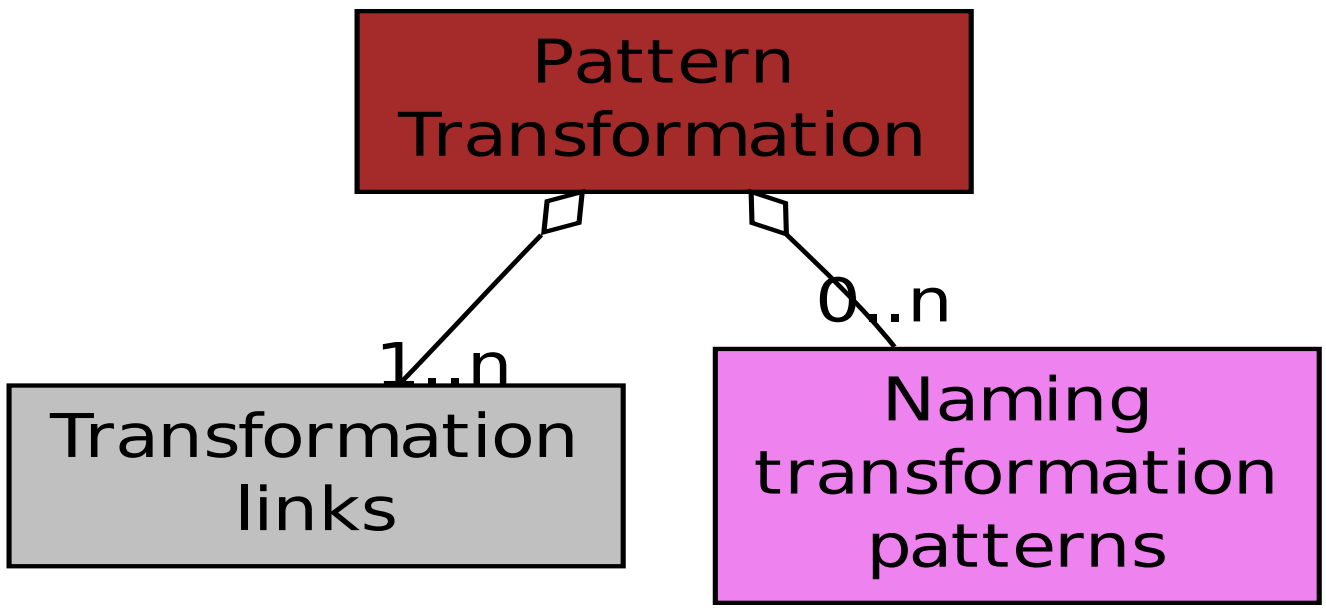
Transformation Pattern Representation cont'd

Ontology Pattern

- *Entity declarations*
  - types: Class, ObjectProperty, DatatypeProperty, AnnotationProperty, Individual, Literal
  - entity placeholders, e.g. Class: ?A
  - (specified) entities, e.g. Class: http://www.ontologydesignpatterns.org/cp/owl/agentrole.owlAgent
- *Axioms* in OWL Manchester syntax, e.g. ?G equivalentTo (?q some ?F)
- *Naming Detection Patterns*, e.g. comparison(?B, head term(?p)), exists(verb form(?C))

Transformation Pattern Representation cont'd

Pattern Transformation

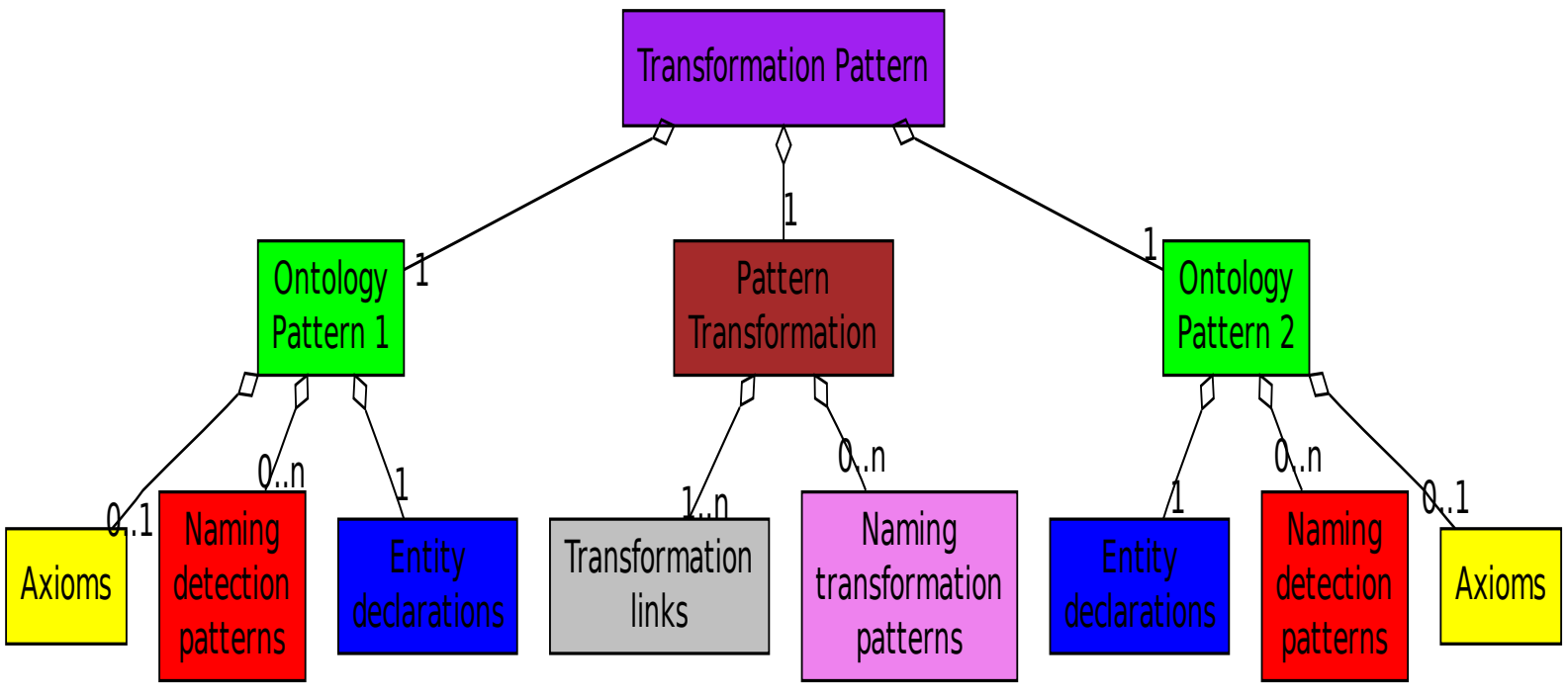


Transformation Pattern Representation cont'd

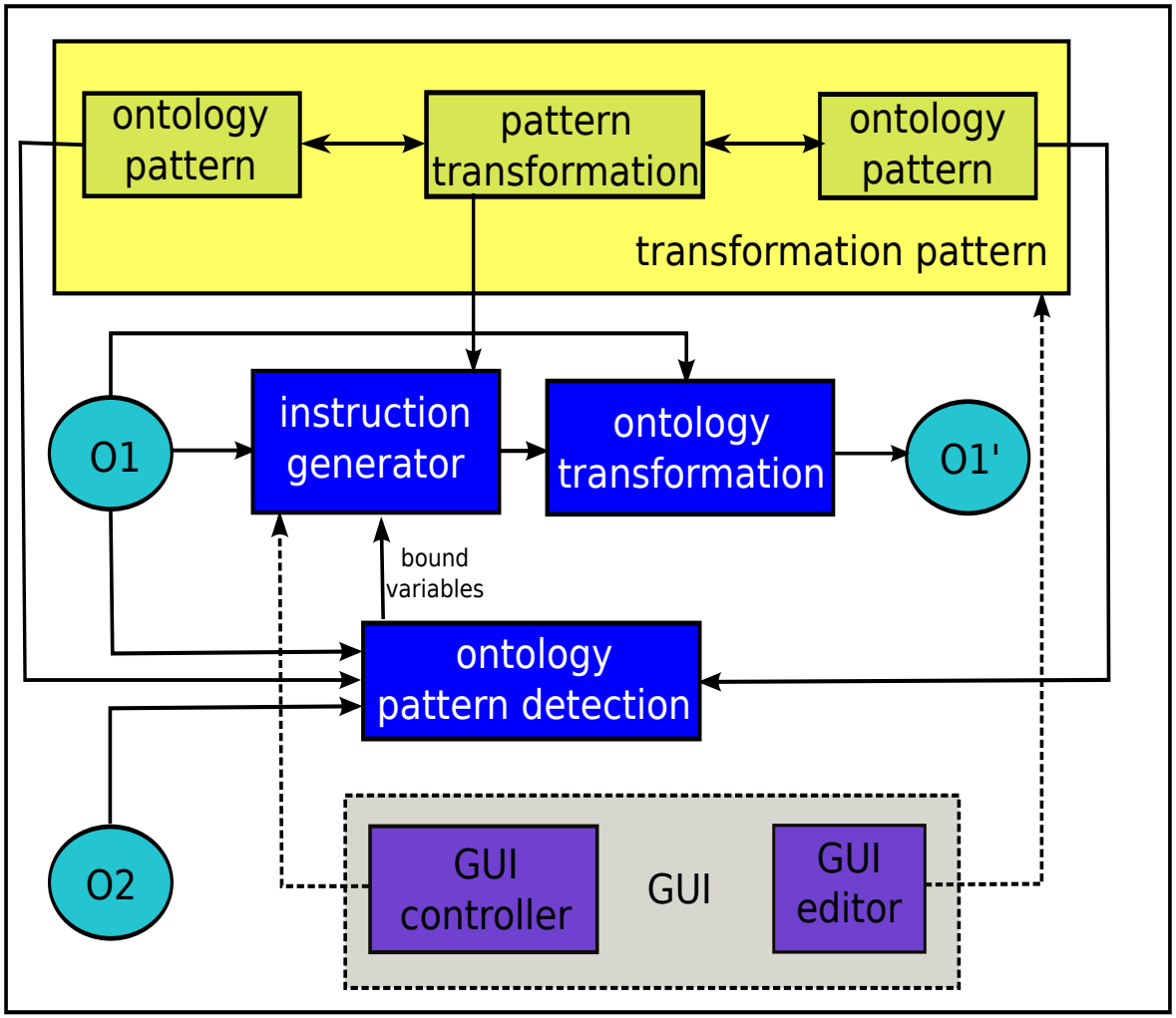
Pattern Transformation

- *Transformation links*
  - logical equivalence correspondence, e.g. ?A EquivalentTo: ?D
  - extralogical *eqAnn* between annotation literal and real entity
  - extralogical *eqHet* between heterogeneous entities
- *Naming Transformation Patterns* e.g.
  ?G, make passive verb(?C) + head noun(?A)

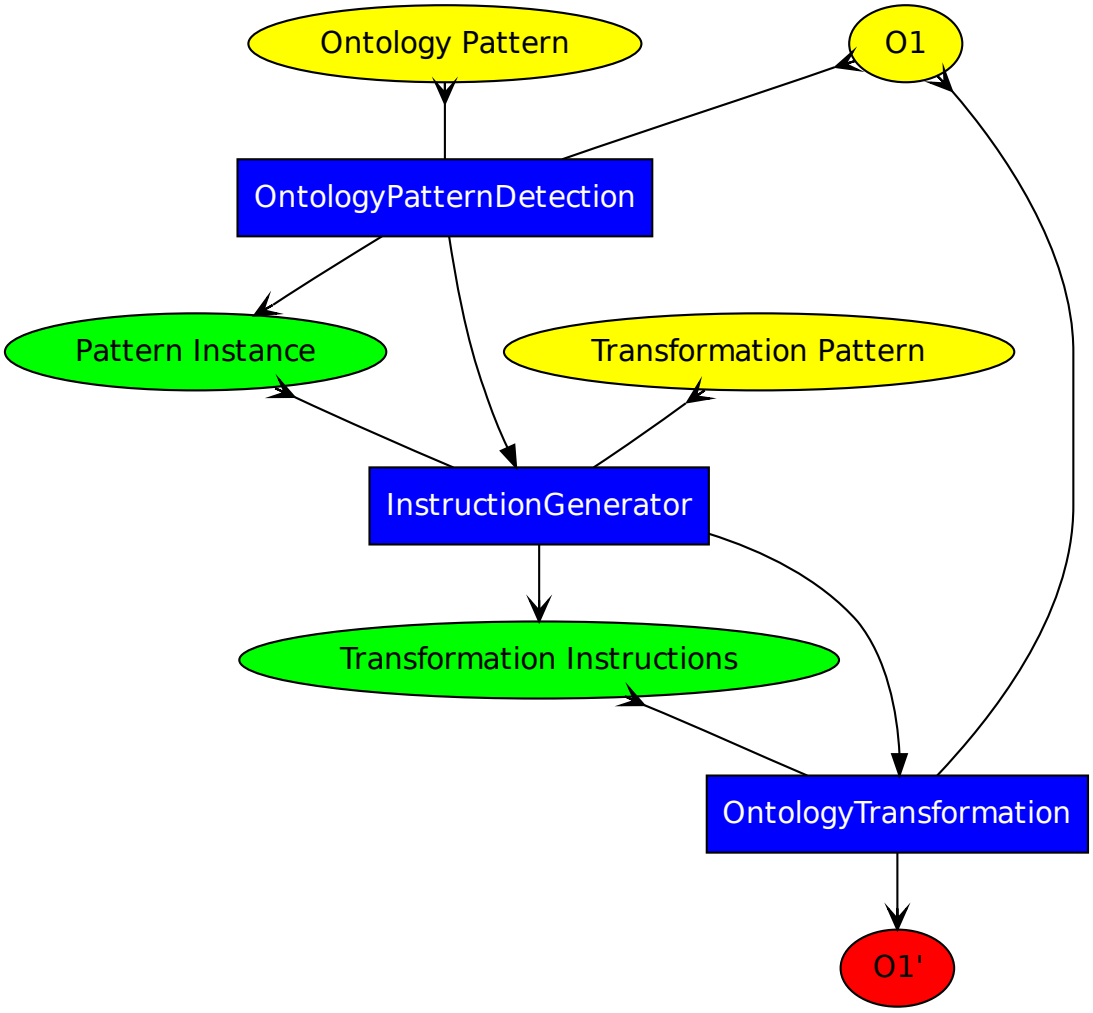
Transformation Pattern Representation cont'd



Overall Framework and Mechanism



Ontology Transformation RESTful services



Ontology Transformation Workflow

- Ontology Pattern Detection: SPARQL query generated based on OP1 of TP
  - Jena ARQ
  - experiments with Terp
  - recursive semantics supported with reasoner?
- Instruction generator: preparation of all instructions in one XML output, see later on
- Ontology Transformation: instructions processing and ontology storing

Transformation Pattern – example AgentRole TP

`http://nb.vse.cz/~svabo/patomat/tp/tp_agentRoleV4a2.xml`

Ontology Pattern Detection (1st service) output example:

```
<pattern_instance>
  <binding placeholder="?B">Assistant</binding>
  <binding placeholder="?A">Person</binding>
</pattern_instance>
<pattern_instance>
  <binding placeholder="?B">Social_event</binding>
  <binding placeholder="?A">Event</binding>
</pattern_instance>
<pattern_instance>
  <binding placeholder="?B">Author</binding>
  <binding placeholder="?A">Person</binding>
</pattern_instance>
```

Instructions generator (2nd service)

Transformation instructions consists of several parts:

- *entities operations*: re/naming and removing of entities instructions (using OWL-API)
- *axioms operations*: OPPL instructions for removing and adding axioms and adding entities
- *annotation operations*: instructions for adding annotations (using OWL-API)

Instruction generator – output example:

```
<instructions tp="tp_agentRoleV4a2.xml">
  <entities>
    <rename type="Class" original_name="Person">Person
    </rename>
    <remove type="Class">Author</remove>
  </entities>
  <oppl_script>
    <add>!authorRole types !Author</add>
    <add>Person subClassOf &or;Agent</add>
    <remove>Author subClassOf Person</remove>
    <add>!Author subClassOf &ar;Role</add>
  </oppl_script>
  <annotations>
  </annotations>
</instructions>
```

Transformation instructions processing workflow

Transformation instructions processing workflow:

1. Removing axioms (OPPL)
2. Removing entities (OWL-API)
3. Adding axioms and entities (OPPL)
4. Adding annotation axioms (OWL-API)
5. Renaming entities (OWL-API)

Ontology Transformation (3rd service)

Transformation instructions can be applied on the ontology.
Two problems:

1. Additional Axioms
2. Recursive Structures in pattern

Ontology Transformation – Additional Axioms

Additional Axioms: external axiom touching the pattern by reffering to one of its entities
Problem: what about removing those entities?
Example: This can be demonstrated on tp_agentRoleV4a2.xml and confOf ontology.

- Author class is removed
- additional axiom e.g. writes Domain: Author

Ontology Transformation – Additional Axioms cont'd

- Solution 1: neither removing entities nor axioms - *conservative strategy*
- Solution 2: only removing axioms - *progressive (default) strategy*
- removing both entities and axioms - *radical*
  - Solution 3: entities are kept and additional axioms are annotated (annotation:remove_warning_by, annotation:remove_warning_for) - *radical-keep strategy*
  - Solution 4: entities are replaced either equivalently (Author equivalentTo Person and (writes some Paper)) or similarly (domain/range example) using TP in future - *radical-neutral strategy*
  - Solution 5: entities are removed and additional axioms are annotated but... - *radical-remove strategy*

Ontology Transformation – Recursive Structures in patterns

Recursive Structures in patterns: some pattern can be applied in recursive way on its part, e.g. recursive taxonomies
Problem: how to cope with recursion in transformation process?

- Solution: recursive detection supported with additive instruction generation wrt. numerous pattern instances

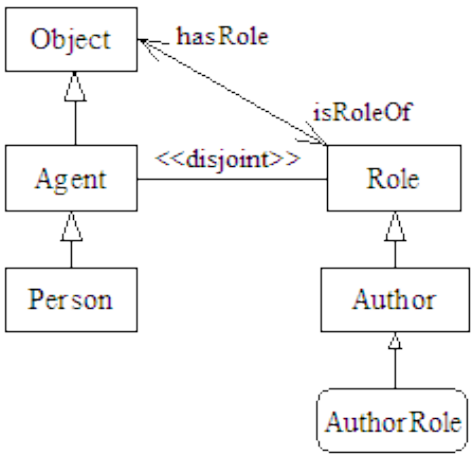
Example: This can be demonstrated on tp_agentRoleV4a2.xml and confOf-recursive ontology.
confOf enriched with:

- Author subClassOf Person.
- PersonAuthor subClassOf Author.
- PosterAuthor subClassOf Author.

Ontology Transformation – Recursive Structures in patterns cont'd

Problem: how to fix placeholders in pattern instances, e.g. ?A=Person or ?A=Author

- {?A=Person, ?B=Author; ?A=Person, ?B=PaperAuthor; ?A=Person, ?B=PosterAuthor}
- {?A=Author, ?B=PaperAuthor; ?A=Author, ?B=PosterAuthor}



Ontology Transformation RESTful conclusion

- Deployed on glassfish application server
- At: `http://owl.vse.cz:8080`
- Services accessible via POST method
- Implementation of Ontology Transformation framework in Java available at: `http://patomat.sourceforge.org/`

- *OntologyPatternDetection*:
  `http://owl.vse.cz:8080/ontologyTransformation/detection/`
- *InstructionGenerator*:
  `http://owl.vse.cz:8080/ontologyTransformation/instructions/`
- *OntologyTransformation*:
  `http://owl.vse.cz:8080/ontologyTransformation/transformation/`

Future Work and Conclusions

- *Bulk transformation* over recursive structures (e.g. taxonomies)
- Elaborate more use cases: other CPs; matching settings; reasoning settings
- Eclipse plug-in for import scenario into XD tool
- *Graphical interface* for *transformation pattern* authoring and instruction generation monitoring
- More advanced *detection techniques*
- Comprehensive library of *naming patterns* relevant for ontology style transformation (based on existing lexical sources)
- Canonical methods for *swapping* information between logical and annotation spaces while transforming
- Ontologies of logical/structural patterns
  - Patterns structure
  - Patterns usage, i.e. matching to modelling issues

Thank you for your attention

http://patomat.vse.cz/