Suite of Tools for Pattern-Based Transformation of OWL Ontologies

Ondřej Zamazal, Marek Dudáš, Ján Černý, and Vojtěch Svátek

University of Economics, W. Churchill Sq.4, 130 67 Prague 3, Czech Republic
{ondrej.zamazal|xdudm12|xcerj121|svatek}@vse.cz

Motivation The high expressivity of OWL enables us to express the same conceptualization in different ways. A simple example is the difference between 'class-centric' and 'property-centric' modeling styles, such that the same notion is modeled as a class in the former (e.g. 'Purchase') and an object property in the latter (e.g. 'bought_from'). Similarly, concept subordination can be expressed via a subclass hierarchy or via individuals connected by a dedicated property (*ontology simplification* as in SKOS). *Style heterogeneity* contributes to *conceptual heterogeneity* on the Semantic Web and thus represent an obstacle to reusing ontologies in advanced semantic web scenarios. In particular, two ontologies modeled in different styles are difficult to *match* or to *import* to one another. Furthermore, opting for a style when designing an ontology may have impact on the usability and performance of *reasoners*, as some features cause performance problems for certain reasoners. Finally, ontology designers may need help with inspection and repair of entity *naming* in an ontology.

PatOMat In our approach [2] we come up with ontology transformation framework *PatOMat*, which is based on *transformation patterns* (TP).[1] A TP contains two *ontology patterns* (the source OP and the target OP) and the description of transformation betweem them, called pattern transformation (PT). The representation of OPs is based on OWL 2, except that *placeholders* are allowed in addition to concrete OWL entities. Furthermore, there is the naming aspect of the OP, which is important for its detection. A PT consists of a set of *transformation links* and a set of *naming transformation patterns*. Naming transformation patterns serve for generating names for target entities. The framework prototype implementation is available either as a *Java library* or as three *RESTful services*.[2] The whole transformation is divided into three steps that correspond to the three core services: *OntologyPatternDetection* service assigning entity placeholders, *InstructionGenerator* generating general and specific transformation instructions, and *OntologyTransformation* service transforming an ontology according to transformation instructions. These services are based on our specific implementation over OWL-API[3] and Jena.[4] Transformation process is decomposed into parts in order to enable an user intervention.

[1] There is also fully-fledged tutorial at `http://owl.vse.cz:8080/tutorial/`.
[2] All accessible via the web interface at `http://owl.vse.cz:8080/`.
[3] `http://owlapi.sourceforge.net/`
[4] `http://jena.sourceforge.net/`

Tools In order to support smooth application of TPs we implemented several graphical tools. *Transformation Pattern Editor* (TPEditor) [1] supports authoring and updating of transformation patterns. It allows their graphical modelling and export/import from/to the (XML-based) TP notation. TPEditor is available as a plugin for Eclipse. *Graphical User Interface for Pattern-based Ontology Transformation* (GUIPOT) is a Protégé plugin allowing the user to go through all steps of transformation via a standard working environment of a knowledge engineer [3]. The *Importing transformation wizard* supports the *ontology import* use case in the sense that the user can select a *content ontology design pattern*, an ontology and a transformation pattern, and a specific process will be performed. This wizard is integrated into the *eXtreme Design* tool supporting pattern-based ontology development and this can be plugged[5] into Eclipse as well as into the NeOn toolkit.[6] The *Naming repair plugin* supports the naming repair use case. It is integrated into the *Ontology Repair and Enrichment* toolkit. (ORE)[7] Next, there is a web-based *Downgrading application* supporting the language profiling use case.[8] Following the input of the source ontology URI (and selected TP in the 'one construct transformation' use case), the transformed ontology is output together with a brief transformation log. Finally, we are working on web-based variants of tools for editing and applying transformation patterns. This will be available as *Portal of Applicable Transformation Patterns* integrating the *Web-based Catalogue of Ontology Transformation Patterns* (WebCOP) consisting of detailed information about each TP, *Web-based Editor of Ontology transformation Patterns* (WEdOP) and RESTful services for launching the transformation steps on demand.

Demo This demo is successor of the demo [3] presented at EKAW 2012. The main enhancement since then will be the integration into ORE and the web-based integration of tools (i.e. WebCOP, WEdOP and RESTful services). While the former has already been completed, we are still working on the latter (it will be available at the time of workshop). Additionally, there are two new use-cases on which we can demonstrate ontology transformation: *Ontology Naming Repair* and *Ontology Simplification*.

References

1. Šváb-Zamazal O., Daga E., Dudáš M., Svátek V.: Tools for Pattern-Based Transformation of OWL Ontologies. Presented as demo at ISWC'11, Bonn, 2011.
2. Šváb-Zamazal O., Svátek V., Iannone L.: Pattern-Based Ontology Transformation Service Exploiting OPPL and OWL-API. In: EKAW-2010, Lisbon, Portugal, 2010.
3. Šváb-Zamazal O., Dudáš M., Svátek V.: User-Friedly Pattern-Based Transformation of OWL Ontologies. Demo session at EKAW 2012, Galway, Ireland, 2012.

[5] Information about these tools is available at `http://owl.vse.cz:8080/tools.html`
[6] `http://neon-toolkit.org/wiki/Main_Page`
[7] `http://ore.aksw.org/ore`
[8] Available from `http://owl.vse.cz:8080/Downgrading/`.