Welcome, newborn entity! On handling newly generated entities in ontology transformation

Vojtěch Svátek[1,*], Ondřej Zamazal[1,*], Kateřina Haniková[1,*], David Chudán[1,*], Mohammad Javad Saeedizade[2,*] and Eva Blomqvist[2,*]

[1]*Prague University of Economics and Business, Czechia*
[2]*Linköping University, Sweden*

Abstract
Modeling can be seen both as an engineering and a design task. Even when clear requirements are available for an ontology modeling endevour, many times the requirements can be solved in several different ways. Even further, when requirements change and evolve, a certain modeling style, or pattern, may no longer be the best choice. However, refactoring an ontology due to such changes in requirements, or due to the desire to align better with other external ontologies or data sets, is a complex and tedious process, also requiring extensive expertise. Attempting to automate part of the ontology transformation process, by identifying typical transformation patterns, and creating tool support for their semi-automated application, is therefore an important research topic. One specific sub-task of such automation is the naming of new entities (e.g. classes or properties) that are generated through the pattern application. In this paper we discuss the need for such automated naming support, and show the feasibility of introducing Large Language Models (LLMs) for taking the automation one step further.

Keywords
Ontology transformation, OWL, entity naming, Large language models,

1. Introduction

When developing an ontology, the ontology engineers must choose one of, sometimes, many possible ontological representations of the abstract situation being modeled [1]. The different alternatives can be characterized as conforming to certain *modeling styles* or *modeling patterns*. For example, a complex relationship can be expressed as a chain of simpler relationships (e.g., a person *wasBornIn* a city (or, settlement), which is, in turn, linked by *locatedIn* to a country) or we can create a shortcut using a compound relationship (e.g., *wasBornInCountry* directly linking a person with a country). The choice of modeling style impacts down-stream applications that consume the ontology and apply its ontological viewpoint, e.g., a knowledge graph *visualization* tool can either be capable of employing the chain property to downsize a diagram or not – depending on whether this property is an explicit part of the ontology. One solution to the ontology rigidity problem (entailed by once-made modeling decisions) is to enable posterior *transformation* of an ontology across different modeling styles/patterns. To manage the complexity and resource needs of such transformations, ontology transformation could be fully automated, or, more likely, semi-automated. A pre-requisite of automation is the availability of reusable *transformation patterns* – either a priori or, potentially, via on-the fly discovery of such patterns by advanced AI methods.

The idea of automated transformation of an ontology into its (fully or partially) semantically equivalent structural variants, fitting particular tasks, was probably first coined by Zamazal et al. in the

EKAW 2024: EKAW 2024 Workshops, Tutorials, Posters and Demos, 24th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2024), November 26-28, 2024, Amsterdam, The Netherlands.
*Corresponding author.
svatek@vse.cz (V. Svátek); ondrej.zamazal@vse.cz (O. Zamazal); katerina.hanikova@vse.cz (K. Haniková); david.chudan@vse.cz (D. Chudán); javad.saeedizade@liu.se (M. J. Saeedizade); eva.blomqvist@liu.se (E. Blomqvist)
https://nb.vse.cz/~svatek/ (V. Svátek); https://nb.vse.cz/~svabo/ (O. Zamazal)
0000-0002-2256-2982 (V. Svátek); 0000-0002-7442-9016 (O. Zamazal); 0009-0009-7162-5925 (K. Haniková); 0000-0002-3678-6223 (D. Chudán); 0009-0000-0812-6167 (M. J. Saeedizade); 0000-0003-0036-6662 (E. Blomqvist)

PatOMat project [2]. The user interaction was supported by a graphical user interface, GUIPOT, and a graphical editor of transformation patterns was also provided [3]. The tools developed in PatOMat were used in a number of pilot use cases [4]. However, the high amount of user interaction in the to-be-transformed entity selection phase as well as the need to manually tune the new entities' lexical labels (generated via simple lexico-syntactic heuristics) after the transformation, prevented the whole approach from larger uptake. The *EvoPat* tool [5], in contrast, handled not only the schema transformation but also the instance data transformation. However, due to its RDF-triple granularity, expressing high-level logical patterns in it would have led to low comprehensibility of the transformation pattern representations. It also did not address entity naming, neither in detection nor in transformation.

Aside from such early endeavors, the ontology transformation task has not been extensively researched in recent years, presumably due to the challenges of formulating the transformation patterns, identifying the cases when they can be applied and managing the effects of their application. However, with the growing base of potential transformation patterns as well as with current capabilities of generative AI, we argue this is the time to attempt at such automation anew. Specifically, Large Language Models (LLMs) should be capable of supporting the matching between ontologies and transformation patterns as well as of handling some parts of the transformation effect, such as the generation and naming of new entities. The value of such methods is also supported by the fact that the amount of published RDF datasets is steadily increasing, and their reuse could bring tangible benefits – if not hindered by discrepancies between their modeling styles.

In this paper we focus on a specific problem within ontology transformation: handling the new entities (e.g. properties or classes) that result from the application of the transformation pattern. In Section 2 we illustrate a number of ontology transformation patterns, and characterize their 'new entity generation' effect. In Section 3 we then present the results of preliminary experiments in providing meaningful names for such entities via LLMs.

2. Transformation patterns and their newly generated entities

A formal framework for describing ontology transformation patterns has already been provided in earlier work [2]. Here we only provide the intuition that a transformation pattern (in the context of OWL2OWL transformation) consists of (i) the *source pattern* and (ii) *target pattern*, and of (iii) *semantic links* between the source and target. Source and targets both being OWL patterns, i.e., syntactically fragments of ontologies containing placeholders in the position of some or all entities, and the semantic links being primarily responsible for providing a name (i.e., typically, an `rdfs:label` value and/or a human readable IRI fragment) for every newly generated entity.[1]

Bootstrapping a set of transformation patterns can be done thanks to their similarity to *alignment patterns* [6] used to align existing, independently developed ontologies. Specifically, so-called *complex alignments* define correspondences not between atomic entities but between (class or property) expressions of which at least one is compound. Note however that while the complex alignment patterns structurally correspond to transformation patterns, they do not tackle the problems of handling new entities, central in this paper, since no new entity comes into being in the ontology alignment process.

Table 1 shows seven verbally described transformation patterns (some having alignment patterns as pre-cursor). Note that some are representatives of larger transformation pattern families, the members of which share some properties, in particular, the nature of output entities.

Given the lack of space, we only explain the patterns semi-formally, providing (for each pattern, which we number from 1 to 7):

1. A tentative *name* for each pattern.
2. The meta-type of the *new entities* – class, object property (OP) or data property (DP) – and their connection to pre-existing ontology entities via RDFS axioms (subclass, domain and range).

[1] Our current implementation of transformation patterns thus actually calls this third part of the pattern 'naming transformation', for simplicity.

Table 1
Verbal overview of prominent transformation patterns; OP/DP = object/data property, D = domain, R = range

#	Pattern name	New entity(-ies)	Entailment axiom(s)
1	2-chain shortcutting	new OP for given D, R	property chain
2	class by attribute value	new subclass of given class	property restriction
3	relationship de-reification	new OP for given D, R	property chain with inverse
4	property inversion	new OP for given D, R	inverse property
5	OP to DP conversion	new DP for given D	NONE
6	link counting property	new DP for given D	NONE

7	role relationship shortcutting	(1) new subclass of given class
		(2) new OP for given R;
		D = the new class
			(1) compound property restriction
			(2) *Horn rule*

3. The nature of the *entailment* (OWL) *axiom* that can be used to entail the ABox statements – class instantiation or property assertion – for the new entity.

A more formal description of each pattern is in an auxiliary page.[2]

Pattern 1 is the shortcutting of 2 chained[3] properties by *one new property* [7], as in the initial 'city/-country of birth' example. Pattern 2 is derived from a published alignment pattern [6]; if we have an object property with defined domain and range (such as `:hasReviewDecision` with domain `:Paper` and range `:Decision`), and the range class has a subclass (e.g., `:AcceptanceDecision`) then we can create a *new corresponding subclass* of the domain class ('class of papers with acceptance decision'), equivalent to a restriction of the property to the subclass of the range property. Pattern 3 is similar to Pattern 1, but with slightly different input and somewhat different background semantics. The properties in this patterns do not form a 'chain', since the direction of the two source properties are opposite, thus requiring an inverse property restriction in the entailment. For example, we may have a class `:Marriage` that is in the domain of two properties, `:involvesHusband` and `:involvesWife`; from this we derive a *new object property* expressing the direct link ('married to') between the husband and wife. The meaning of Patterns 4 and 5 is straightforward, e.g., simply replacing one property with another, either its inverse (e.g., `:livesIn` with `:hasInhabitant`) or another kind (DP instead of OP). Note that no OWL entailment can assure the ABox entailment for Pattern 5, as it requires the production of a string literal for the target data property substituting the corresponding IRI value of the source object property (e.g., instead of the IRI `dbpedia:Amsterdam` there would be `"Amsterdam"^^xsd:string`). The literal could be produced either by parsing the IRI string or by fetching the value of a 'naming' property such as `rdfs:label` or `dcterms:title`, but neither approach is guaranteed to work. Pattern 6 also consists in generating a data property from an object property such that both have the same domain (e.g., 'Person') , however, with a different meaning: the target, integer-valued, data property (e.g., 'numberOfcitizenships') counts the cardinality of the source object property (e.g., 'isCitizenOf') assertions for the given RDF subject. Finally, Pattern 7 is more complex and more content-oriented compared to the previous 'purely structural' ones. Semantically, the input is a ternary relation expressing the fact that an object A plays a role R with respect to an object B (for example, Bob plays the role of researcher at CWI); however, syntactically, the relation R is reified to a class (e.g., `:RolePlaying`, with outgoing relations `:involvesRolePlayer`, `:involvesRole` and `:involvesRelatedObject`). The transformation then generates two output entities, as a kind of combination of Pattern 2 and Pattern 1. The first new entity is a subclass of the class of A (e.g., `:Researcher` as subclass of `:Person`), as a 'typification' of the role. The new subclass can be defined through an OWL property restriction, but a compound one (such as $Researcher \equiv \exists\ {}^{\wedge}involvesRolePlayer\ .\ (\exists\ involvesRole\ .\ ResearcherRole)$, in description logic notation), with link between A and the reified relationship node inverted (^). The second new entity is an object property (e.g., 'researches at') directly connecting the new class

[2]https://github.com/Onto-DESIDE-VSE/TransformationPatterns/tree/main/experiments/LLMs%20experiments/EKAW24
[3]In the sense that the range of one is the domain of the other.

(`:Researcher`)[4] with the class of B (say, `:Institute`). Here, we cannot even express the entailment in OWL, but would need a rule language (formally, a Horn clause), since there are multiple non-chained antecedent atoms needed (e.g., `researchesIn(X,Y) :- involvesRolePlayer(Z,X), involvesRole(Z,`*`ResearcherRole`*`), involvesRelatedObject(Z,Y)`[5]).

Note that we only discussed simple transformation patterns that are merely applied on a single ontology fragment in one match. Some other patterns, such as conversions of class hierarchies to SKOS (instance-level) taxonomies, require bulk operation and would be more difficult to describe.

3. Large Language Model (LLM) as namegiver

All mentioned transformation patterns feature the creation of new entities, and except for OP to DP conversion (where likely – although not certainly – the data property could retain the name of the object one), the labeling of them is not trivial. Earlier approaches [4] employed template-based NLG techniques for such entity naming, sometimes with a grain of linguistic knowledge (such as WordNet derivative forms, allowing, e.g., nominalization of verbs). Nowadays it however seems natural to resort, for this task, to LLMs, which should assure *fluency* of the output, and possibly also some degree of *semantic abstraction*, leading to *parsimony* in terms of the number of tokens involved. The last point is particularly important if the transformed entities are to appear in analytical applications, whether knowledge graph visualization, or, say, knowledge graph mining, where output hypotheses/models may consist of numerous interconnected entities.

To assess the feasibility of the approach we have carried out preliminary, merely qualitative experiments on generating names for new entities in the context of Pattern 1 (a new property) and Pattern 2 (a new class). For each pattern we gathered a dataset of randomly selected examples from the Archivo repository.[6] After that a human annotator (one of the authors of this paper) suggested a name for each new property (Pattern 1) and for each new class (Pattern 2). For Pattern 1 we selected 16 examples that were deemed the most interesting and suitable for the pattern. For Pattern 2 there were 25 examples selected. These examples were used to execute the experiments with LLMs and assess their output suggestions.

For the LLM-based naming of generated entities, we performed two experiments. The first (a) was prompting using natural language only, and the second one (b) used prompts including the input ontology structures in Turtle syntax. There intuition being to explore whether the LLMs can handle requests involving the actual pattern structure, as expressed in a formal language (possibly benefiting from the LLM having been also trained on code samples in this language), or whether such structures need to first be transformed into natural language (involving an additional step with potential sources of error).

Even though these experiments are very preliminary, only a first attempt to assess the quality of LLM support capabilities, the results are promising.[7] However, in the experiments with Pattern 2, it seems that the LLM is able to provide better results to this task when it is prompted with natural language only (experiment a), rather than in Turtle syntax (b).

A certain level of agreement was observed between the LLM suggestions and the human naming, and also between different LLMs. For example, suggestions made for the chain of properties (Pattern 1) with the following classes and properties *"Meeting - meeting notes - Meeting Notes - contains action item - Action Item"* included:

- Suggested property (human) – assigned action item
- GPT-4o (natural language prompt, without output specification) – generates action item *or* has action item

[4]Note the change of property's domain to the subclass of the original domain, with no counterpart in Pattern 1 or Pattern 2.
[5]We alter the entity notation in the examples according to the standard syntax (RDF vs. description logic vs. Horn rules); the identity of entities should be however clear from the lexical content of the tokens.
[6]https://archivo.dbpedia.org/
[7]Full set of results available from https://github.com/Onto-DESIDE-VSE/TransformationPatterns/tree/main/experiments/LLMs%20experiments/EKAW24

- GPT-4o (natural language prompt, 3 outputs required) – action item assigned in *or* outcome of meeting
- GPT-4o (Turtle representation) – has action item *or* includes task
- GPT-4o (Turtle representation, three-shot prompting) – has action item *or* includes action item

In some cases, the results gained from LLMs appeared even more suitable than the names suggested by a human. For example, for the chain of properties (Pattern 1), with the following example *"Organization - delivery location - Residence Object - address - Basic Address"*. The human annotator suggested to call the new property between *"Organization"* and *"Basic Address"* *"address of delivery location"*, while Gemini Pro and GPT-4o suggested *"has delivery address"*, which is more natural. This indicates that apart from the fully automated case, even in a semi-automated setting, using LLM suggestion may allow human users to think of better formulations.

Of course, there were also some 'hallucinations'[8] returned by some of the models, i.e. suggestions that do not make sense. For example, for Pattern 2 with the example *"Person - has period - Period of Life - Honour"* a suggestion fo the LLM for the new subclass of person was *"Honored Period Person"*. Although, with different prompt, the same model suggested *"Honored Person"*. This is also the reason why we decided to prompt each model for more than one option, so that we could observe if it is capable of coming up with some suitable name for a new entity even in light of occasional random responses.

It should be noted that while we for simplicity referred to the mentioned research activities as to 'experiments', there was no rigorous experimental protocol present; the nature of the analysis of results was rather that of group discussion over the individual results, followed by seeking agreements on whether the overall observation on the models' performance is positive or not.

4. Conclusions and future work

Adequately treating, and plausibly naming, newly generated entities is a crucial challenge in the automation of ontology transformations. Such challenges need to be overcome to make RDF knowledge graphs adaptable to new requirements and applications, in a scalable way. Throughout the paper, we have illustrated the need for handling of new entities in many common transformation patterns, and provisionally demonstrated the feasibility of using LLMs for the naming of these new entities.

We plan to extend the preliminary attempts to use LLMs for new entity naming to more rigorous experiments with multiple language models and across a larger set of transformation patterns. Upon the successful completion of these experiments, we will integrate the invocation of properly configured LLMs into our new ontology transformation tool currently under development [8][9]. A further challenge to be investigated is the mentioned interference between the application of different transformation patterns (or manifold application of the same pattern) on the same ontology. In order to better grasp this phenomenon, as well as for various other purposes, we plan to design an ontology and knowledge graph of OWL2OWL transformation patterns, capturing both syntactic and semantic (linking to relevant foundational ontologies) aspects of these patterns.

Acknowledgments

This work has been supported by the EU's Horizon Europe grant no. 101058682 (Onto-DESIDE).

References

[1] C. Shimizu, K. Hammar, P. Hitzler, Modular ontology modeling, Semantic Web 14 (2023) 459–489. URL: https://doi.org/10.3233/SW-222886. doi:10.3233/SW-222886.

[8]Since we are dealing with names for general entities and not with facts, using the term 'hallucination' for inadequate output might not be the best fit.

[9]The PatOMat2 tool, https://github.com/Onto-DESIDE-VSE/patomat2, is a follow-up of the original PatOMat, with numerous enhancements.

[2] O. Sváb-Zamazal, V. Svátek, L. Iannone, Pattern-based ontology transformation service exploiting OPPL and OWL-API, in: P. Cimiano, H. S. Pinto (Eds.), Knowledge Engineering and Management by the Masses - 17th International Conference, EKAW 2010, Lisbon, Portugal, October 11-15, 2010. Proceedings, volume 6317 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 105–119. URL: https://doi.org/10.1007/978-3-642-16438-5_8. doi:10.1007/978-3-642-16438-5_8.

[3] O. Sváb-Zamazal, M. Dudás, V. Svátek, User-friendly pattern-based transformation of OWL ontologies, in: A. ten Teije, J. Völker, S. Handschuh, H. Stuckenschmidt, M. d'Aquin, A. Nikolov, N. Aussenac-Gilles, N. Hernandez (Eds.), Knowledge Engineering and Knowledge Management - 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings, volume 7603 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 426–429. URL: https://doi.org/10.1007/978-3-642-33876-2_39. doi:10.1007/978-3-642-33876-2_39.

[4] O. Zamazal, V. Svátek, Patomat - versatile framework for pattern-based ontology transformation, Comput. Informatics 34 (2015) 305–336. URL: http://www.cai.sk/ojs/index.php/cai/article/view/1138.

[5] C. Rieß, N. Heino, S. Tramp, S. Auer, EvoPat - pattern-based evolution and refactoring of RDF knowledge bases, in: P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Z. Pan, I. Horrocks, B. Glimm (Eds.), The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I, volume 6496 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 647–662. URL: https://doi.org/10.1007/978-3-642-17746-0_41. doi:10.1007/978-3-642-17746-0_41.

[6] F. Scharffe, O. Zamazal, D. Fensel, Ontology alignment design patterns, Knowl. Inf. Syst. 40 (2014) 1–28. URL: https://doi.org/10.1007/s10115-013-0633-y. doi:10.1007/s10115-013-0633-y.

[7] A. Krisnadhi, N. Karima, P. Hitzler, R. Amini, V. Rodríguez-Doncel, K. Janowicz, Ontology design patterns for linked data publishing, in: P. Hitzler, A. Gangemi, K. Janowicz, A. Krisnadhi, V. Presutti (Eds.), Ontology Engineering with Ontology Design Patterns - Foundations and Applications, volume 25 of *Studies on the Semantic Web*, IOS Press, 2016, pp. 201–232. URL: https://doi.org/10.3233/978-1-61499-676-7-201. doi:10.3233/978-1-61499-676-7-201.

[8] O. Zamazal, M. Ledvinka, V. Svátek, PatOMat2: A Tool for Pattern-Based Ontology Transformation using SPARQL, in: I. Novalija, C. Badenes-Olmedo (Eds.), Companion Proceedings of the 24th International Conference on Knowledge Engineering and Knowledge Management, Amsterdam, Netherlands, CEUR Workshop Proceedings, To Appear. CEUR-WS.org, 2024.