Guided ENtity reuse and class Expression geneRATOR

C. Maria Keet
School of Mathematics, Statistics, and Computer Science, University of KwaZulu-Natal, and UKZN/CSIR-Meraka Centre for Artificial Intelligence Research, South Africa
keet@ukzn.ac.za

Muhammad Tahir Khan, Chiara Ghidini
Fondazione Bruno Kessler
Trento, Italy
{tahirkhan, ghidini}@fbk.eu

ABSTRACT

Generic, reusable ontology elements, such as a foundational ontology's categories and part-whole relations, are essential for good and interoperable knowledge representation. Ontology developers face the challenge to figure out which category their class belongs to and which relationship to choose for their ontology authoring tasks. To reduce this bottleneck, there is a need to have guidance to handle these Ontology-laden entities. We solve this with the generic approach GENERATOR: *Guided ENtity reuse and class Expression geneRATOR* that enables easy reuse of already represented knowledge such that it guides the modeller toward the comparatively best options of possible axioms to add. This has been realised with DOLCE, part-whole relations, and an automated reasoner that is used during the authoring process to propose feasible axioms, and the software has been integrated in the MoKi ontology development tool.

1. INTRODUCTION

Effectively supporting ontology developers—including domain experts and novices—in the process of authoring OWL ontologies, is increasingly recognised in a number of works [2, 7] as a crucial step to make the construction of ontologies more agile and apt to the needs of organisations and business enterprises. Many challenges need to be faced at ontology design time, and in particular at the level of *ontology authoring* when deciding *how* to formalise knowledge. Three crucial questions that an ontology developer could ask herself while writing an ontology are: *where do I start?* and *where do I add classes?* and *how do I relate them?*. Generic, reusable ontology elements and domain-independent relations have the potential to both help ontology developers in answering these questions and also speeding up the ontology development and improving the quality of the ontology [3, 4]. Nevertheless, reusing entities and selecting relations from existing ontologies is a complex task (e.g., [4, 6]). Very little methodological and tool-based support is given to ontology developers on how (where) to link a domain ontology to a foundational one and/or how to reuse any of the knowledge represented in the foundational ontology (few exceptions are [5, 4]).

We tackle this problem with a general method that caters for several combinations of manual design and automated support, which makes use of a question and answer system to link the domain ontology to a general and/or foundational ontology and an automated reasoner for selection of relations from an ontology. Automated reasoning services, at present, are used *post hoc* when the axioms have been added to the ontology already; instead, we use a reasoner to provide support *during* ontology authoring—i.e., in the process of adding an axiom—by narrowing down the possible set of relations. In this paper, we summarize GENERATOR: *Guided ENtity reuse and class Expression geneRATOR*. We will demonstrate an implementation of GENERATOR for DOLCE, part-whole relations, and use of the automated reasoner, which is independent of the ontology development environment, and its integration in MoKi to achieve usability.

2. THE GENERATOR METHOD

The general method we propose is independent from its implementation and independent of the chosen ontology language; i.e., it equally well can be realised manually with one or more ontologies formalised in a higher-order logic or semi-automated with OWL ontologies and OWL automated reasoners. We describe first its materials and then the method.

Materials. There are various scenarios (see below), yet the materials have to be set up only once, not for each axiom one contemplates to add to an ontology. One will need the following, with key items in italics:

1. A *domain ontology* that will be used as a base ontology to which the axioms will be added, O_d;
2. Optionally, a *foundational or a high-level relation ontology*, O_f, to which the domain ontology has been, or will be, aligned either permanently or only during the selection process, where one can either select a category (high-level entity) from O_f directly or through a *decision diagram* that asks the user one or more closed questions that guides her through the structure of the ontology upon which the system will propose the category;
3. The ontology or ontologies (O_d and O_f) should have one or more relationships (object properties) whose *domain and/or range have been declared*;
4. Optionally, an ontology development environment;
5. Optionally, an *automated reasoner* with the minimum

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
K-CAP '13, June 23-26, 2013, Banff, Canada.
Copyright 2013 ACM 978-1-45-03-2102-0/13/06 ...$10.00.

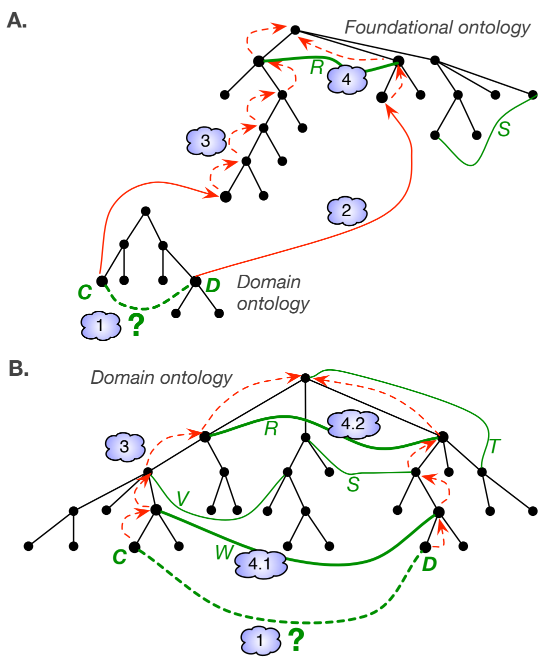

Figure 1: Depiction of the general idea of GENERATOR. A: a domain ontology and a foundational ontology; B: procedure when using only one ontology.

capability of traversing the taxonomy upwards;

The method. There are two main scenarios, and the general idea for both cases is depicted in Figure 1. For scenario A, the modeller has a domain ontology and a foundational ontology. Upon selection of a class in the domain ontology and the desire to relate that to another class (step (1) in Figure 1-A), it has to be aligned somehow to the foundational ontology (step 2). Either this class alignment already exists, so this step is completed automatically in the background, it can be carried out manually if the modeller knows the foundational ontology well, or can be guided by the decision diagram specific to that foundational ontology. Subsequently, one moves up in the taxonomy (step 3) to (automatically) find any possible object properties. While step (3) can be done manually or with a script to retrieve a class's parent class, the main advantage of the reasoner (besides saving oneself analysis and coding time) is taxonomic classification to have the most up-to-date class hierarchy and therewith avoid spurious candidates. Thus, one only has to select which classes to relate, and, optionally, align the ontology, and the software can handle the rest, as each time it finds a domain and range axiom of a relationship in which the parents of C and D participate, it is marked as a candidate property that can be used in the axiom. Finally, the candidate properties are returned to the user (step 4). Not shown in the diagram but equally possible, is the use of a separate relation ontology, which is a variation of scenario A: the modeller uses a relation ontology to axiomatise the base ontology. Also here the base ontology is traversed upwards and on each iteration the base ontology class is matched against relational ontology to find relations where the (parent of the) class is defined in a domain and range axiom, also until the top is reached before returning candidate relations. Scenario B is similar to A, but then without the step of alignment to the foundational ontology.

Given the labels in Figure 1, the computed suggestion to relate C and D for scenario A is object property R, and it is both W and its super property R in scenario B. It is then up to the modeller to choose and save the axiom in the ontology.

Thus, GENERATOR exploits already declared knowledge in an ontology—be that its own or a borrowed one—and manual or automated reasoning to compute the rest, thereby narrowing down the possible set of relations that can exist between two classes. Neither manual searching and assessment is strictly necessary anymore, nor will the selection result in an inconsistency based on that relation between the two selected classes alone. Hence, it can greatly reduce the cognitive overload during ontology authoring, it fosters reuse of already well-researched knowledge (especially in scenario A), and by using a foundational ontology, also anticipates easy mappings with other ontologies.

3. REALIZATION AND IMPLEMENTATION

We have developed an instantiation of GENERATOR with DOLCE, part-whole relations, a domain ontology, a novel decision tree to categorise a domain class as a subclass of a DOLCE class (named D3), and a novel algorithm that uses an automated reasoner to compute the applicable part-whole relation(s) between the selected domain classes (OntoPartS-2). With GENERATOR, we thus avoid the common post-hoc checking and instead use the reasoner to guide the 'trial' phase and reduce errors. We developed a proof-of-concept implementation that can be reused across ontology development environments, and integrated it into the MoKi tool [1].

4. REFERENCES

[1] C. Di Francescomarino, C. Ghidini, and M. Rospocher. Evaluating wiki-enhanced ontology authoring. In *Proc. of EKAW'12*, volume 7603 of *LNAI*, pages 292–301. Springer, 2012. Oct 8-12, Galway, Ireland.

[2] V. Dimitrova et al. Involving domain experts in authoring OWL ontologies. In *Proc. of ISWC'08*, volume 5318 of *LNCS*, pages 1–16. Berlin, 2008.

[3] C. M. Keet. The use of foundational ontologies in ontology development: an empirical assessment. In *Proc. of ESWC'11*, volume 6643 of *LNCS*, pages 321–335. Springer, 2011.

[4] C. M. Keet, F. C. Fernández-Reyes, and A. Morales-González. Representing mereotopological relations in OWL ontologies with ONTOPARTS. In *Proc. of ESWC'12*, volume 7295 of *LNCS*, pages 240–254. Springer, 2012.

[5] A. Oltramari. A tutoring methodology for the enrichment of ontologies. In *Cahiers de Lexicologie: Revue internationale de lexicologie et lexicographie*, pages 221–229, 2011.

[6] I. Opalicki and S. Lovrencic. How well are domain and upper ontologies connected? In *Central European Conference on Information and Intelligent Systems*, pages 17–22, 2012. Sept. 19-21, 2012, Varazdin, Croatia.

[7] T. Tudorache et al. Ontology development for the masses: Creating ICD-11 in WebProtégé. In *Proc. of EKAW'10*, volume 6317 of *LNCS*, pages 74–89. Springer, 2010. Lisbon, Portugal.