Importing from Functional Knowledge Bases
– A Preview –

Alex Borgida
Dept. of Computer Science
Rutgers University, USA
borgidacs.rutgers.edu

Fausto Giunchiglia
DIT
University of Trento, Italy
fausto@dit.unitn.it

Abstract: We review several proposals for reusing knowledge from existing ontologies by importing concepts and related axioms, which start from a set Ω of "identifiers of interest". In order to compare these and other potential proposals, we offer a formal definition of the notion of "importing knowledge" based on Levesque's functional characterization of knowledge bases using a Tell/Ask interface. This definition is parameterized by a set $\mathcal{A}$ of Ask operators. In this preliminary work we consider ways in which this definition can be used to capture aspects of prior proposals, and therefore provide a framework for comparison between them.

1 Introduction

Motivated in part by the need to re-use ontologies, and to organize them/make them more understandable, there have been numerous proposals for approaches to modularize ontologies, especially ones using Description Logics, including [1, 5–7, 11, 14]. The techniques proposed are based on a variety of intuitions of what are reasonable properties for "self-standing" collections of identifiers and axioms, including the idea that modules should partition the knowledge base into disjoint groups of identifiers and axioms, and that if concept name appears in some module, then so should all its subsuming concepts in the ontology.

While the above proposals start from basic principles of what a module should be, there are a number of examples of actual re-use of knowledge from existing ontologies, several of which are reviewed below. In these cases, rather than importing pre-defined modules, one starts from a set Ω of concepts of interest, and then extracts a portion of an existing ontology that is deemed relevant to it. Though the details of how this is to be done vary, we abstract certain shared desiderata, and discuss to what extent two proposals for defining knowledge import for Description Logics [8, 2] meet them. We then offer a definition of importing knowledge that is based on general Tell/Ask interfaces for knowledge bases, and show how this can be used to capture aspects of the proposals reviewed earlier, thereby providing a framework for comparison.

2 Examples of Importing Terms

The following are examples from the literature of cases where ontology reuse is not by importing modules that have been computed *a priori*, but instead the material imported is determined by the set of terms that is desired.

2.1 Trimming WordNet

Navigli [12] and Swartout et al [15] are interested in situations where several small terminologies with *domain-specific terms* have been extracted (possibly semi-automatically, using text processing techniques). These need to be integrated, and one way to do so is by tying them together through a general ontology, such as WordNet, where terms are already organized in an IsA (hyperonym/hyponym) hierarchy. In particular, the idea is to link the roots of the various domain-specific terminology trees to the closest synsets in WordNet. (Let this be the set Ω in this case.) One is left with the problem that there are many irrelevant terms in the resulting joint terminology. Instead, what is wanted is a small subset of terms in WordNet that subsume and tie together the domain specific terms. This subset can then be viewed as the part of WordNet that is to be imported.

Swartout et al [15] use for this purpose all the nodes in WordNet that subsume Ω, plus children of these that have many siblings already included. We focus here on Navigli's more complex algorithm [12]. It starts from the nodes for Ω, and constructs the final set of imported nodes $\Omega_\star$ in several steps:

$S_1 := \text{IsA}^*(\Omega)$; /* *Nodes in WordNet that subsume* Ω.[1] */

$S_2 := \{$Nodes in S_1 that have 2 or more children (hyponyms) in $S_1\}$

$S_3 := \{$Immediate children of nodes in S_2 that belong to S_1 and have siblings in $S_1\}$

$S_{top} := \{$Nodes in S_1 that are at the very top of WordNet (top 2 levels)$\}$

The final imported fragment of WordNet then consists of the concepts $\Omega \cup S_{top} \cup S_2 \cup S_3$ together with all IsA edges between these *entailed* by the original WordNet taxonomy.

The motivation for including S_2 (or actually excluding the complement of S_2) is that *"a node with only 1 hyponym gives no additional information and provides no additional classification"*. The reason for including elements in S_3 is less clear ("avoid collapsing the hierarchy"), since this over-rides the principle enunciated for S_2.

Note that both here, and below we have rephrased the original algorithms to some extent, since (as the names indicate), they are originally couched in the terminology of *removing* parts of the exporting ontology.

2.2 KnowledgeBus

Petersen et al [13] consider the problem of generating an Information System from a large knowledge base such as Cyc, starting from a seed set of concepts. Their ultimate goal is to generate a deductive database implemented in XSB, and encapsulated in Java. Note that Cyc contains much more than just concept

[1] We use IsA^+ to refer to the transitive closure of the IsA relation, and IsA^* to the reflexive transitive closure. For binary relation ρ, we use $\rho(x)$ to refer to the set $\{y \mid \rho(x, y)\}$ when x belongs to the domain of ρ, and generalize this to sets: $\rho(T) = \bigcup_{x \in T} \rho(x)$.

hierarchies: there are general predicates, and formulas in CycL (a variant of FOL).

The idea of the approach in [13] is to first identify all axioms in Cyc which might contribute to proofs about instances of concepts in the seed set of identifiers S_0 (which correspond to our set Ω). It relies on Cyc predicates p being typed ($p : T_1 \times \ldots \times T_k$ indicates that the j-th argument of p must belong to concept T_j); and the hypothesis that multiple occurrences of a variable, such as y in the axiom $p(x, y), q(y, z) \rightarrow r(x, z)$, require the corresponding arguments of the predicates to be IsA related, ensuring type-compatibility.

Starting from Ω, the algorithm computes by a fix-point iteration (using operators Π and Γ below) a set of concepts of interest S_*, and associated predicate names P_*. The final axioms retrieved are those involving only predicates in P_*.

- for a set of concepts S, $\Pi(S) = \{p \mid p : T_1 \times \ldots \times T_k \ \wedge\ \exists j.[T_j \in S \wedge \neg dataType(T_j)]\}$ */* Gather predicates that have some argument typed by a concept in S. */*
- for a set of predicates P, $\Gamma(P) = \{T \mid p \in P \ \wedge\ p : T_1 \times \ldots \times T_k \ \wedge\ \exists j.genls(T_j, T)\}$ */* Gather generalizations of concepts used to type arguments of predicates in P. */*

2.3 Pruning Ontologies to obtain Database Conceptual Schemas

Conesa and Olivé [4] start with a list of terms (entity, attribute, and relationship names) that are of interest in a particular domain. This list may be developed by analyzing software requirements, for example. (This is the set Ω in this case.) They wish to enrich it with semantic integrity constraints derived from some large existing general ontology, such as OpenCyc translated into UML – the material to be imported.

In [3], the task is *specified* as finding a minimal sub-ontology O_P of the exporting ontology O_X that contains Ω and all formulas/constraints in the set

Axioms1 := {formulas ψ in O_X such that $vocab(\psi) \subseteq IsA^*(\Omega)$ }

and that preserves IsA$^+$ relationships derivable in O_X. Presumably, the formulas in Axioms1 are of interest since, by inheritance, they also "talk about" instances of terms in Ω.

The actual algorithm presented in [4, 3] is considerably more complex, and can be rephrased as a sequence of set specifications:

- $S_{SurelyNeeded} := \Omega \cup vocab(\text{Axioms1})$. */* These terms are the ones definitely wanted. */*
- S2 := $\{\ y \mid \exists x, z \in S_{SurelyNeeded}\ .\ IsA^*(x, y) \wedge IsA^*(y, z)\ \}$ */* These are all the terms in $S_{SurelyNeeded}$ or on inheritance paths between them. */*
- S3 := remove from S2 concepts on so-called "redundant paths" G, consisting of concepts belonging to $S2 - S_{SurelyNeeded}$, but only as long as (i) the existence of some subclass path between x and z in $S_{SurelyNeeded}$ is preserved, (ii) the nodes in G have only single parents and children. */* These concepts only provide redundant inheritance paths. */*

The general intuition here appears to be that the set of original identifiers should be enlarged to include entities/relationships from which relevant axioms can be inherited, plus sufficient intermediate nodes to maintain the inheritance relationship. Note that condition (ii) above is suggestive of Navigli's trimming.

3 Formalizing Knowledge Import

The examples examined above, and others, indicate that in these cases the interest is in

1. obtaining a *minimal* set of *additional* concepts that need to be understood in order to understand the original "terms of interest" Ω being imported;
2. but this set should usually not include concepts that have only one parent and one child in the resulting IsA hierarchy, and no relevant axioms;
3. preserving all *derivable* IsA relationships between the imported concepts;
4. expecting that the result of reasoning with the imported knowledge is somewhat equivalent to including the full exporting knowledge base;
5. the logical language of the exporting ontology (e.g. CycL, UML) need not be a description logic, and may be different from that of the importing knowledge base.

Several papers have tried to formalize the notion of knowledge import, particularly in the context of Description Logics. In particular, Cuenca-Grau et al [8] provide the following definition

Definition 1. *[8] Let $Q1 \subseteq Q$ be two ontologies, and S a set of identifiers. We say that Q1 is an* S-module *in Q* w.r.t. a language L*, if for every ontology P and every axiom α expressed in L, with $vocab(P \cup \{\alpha\}) \cap vocab(Q) \subseteq S$, we have $P \cup Q \models \alpha$ iff $P \cup Q1 \models \alpha$. A* minimal S-module *contains no sub-S-modules.*

The notion of "`minimal` Ω`-module of KB`$_{expt}$ `w.r.t. L`" then corresponds to our import, and deals directly with items 4 and 5 above. Item 1 in the desiderata is handled elegantly, but indirectly, by requiring the imported axioms to be a subset of the original ontology: this usually forces more axioms to be included than those whose vocabulary is contained in S, because of item 4. The negative aspect of this is that the material imported depends on the syntactic formulation of Q. For example, having $A \sqsubseteq (B \sqcap C)$ in Q will behave differently from having the logically equivalent $\{A \sqsubseteq B\ , A \sqsubseteq C\}$, and may force unnecessary concepts to be included.

Borgida [2] also offers a formal definition (which we do not give here), and argues that two other aspects of importing knowledge are relevant:

- in addition to the different logic used in the importing ontology, there may be syntactic restrictions on the occurrence of imported concepts (for example, in many cases the imported concepts are used only as super-concepts of local concepts); */* This may further reduce the set of axioms that need to be imported since they may not be useful even if they concern Ω. */*
- there is a need to preserve explanations rather than exact syntactic form of axioms in the exporting ontology. */* This is useful to handle in part item 2. */*

For purposes of comparison, as well as a guideline for future proposals, it would be desirable to have a general yet formal framework to cover the knowledge import schemes described so far. We follow the observation that the descriptions in Section 2 are often couched in terms of operators that are applied to (sets of) concepts.

We therefore start from Levesque's general functional approach to knowledge representation [10], which provides for users to interact with a KB through a set of Tell and Ask operators. The answering process relies on the use of a particular language L_{rep} to represent internally the information told to the KB, and ideally is *specified* in a logical manner, rather than just procedurally.

For a UML-based KB, typical Tell operations would be ones for building a UML diagram, while there would be Ask operations to read off information, such as getAttributes($<Class>$) — returning the list of locally specified attribute identifiers; and getMinAtSource($<Association>$) — returning the minimum cardinality at the source end of the association link. In addition, there might be Ask operators that provide derived information, such as subsumedBy?($<Class$, $Class>$) or disjointFrom?($<Class, Class>$).

For a Description Logic-based KB, the Tell operations would introduce the identifiers and build a TBox, while Ask operations might include the obvious subsumedBy?($<Concept$, $Concept>$) and isConsistent?($<Concept>$), but also getRange($<Role>$,$<Concept>$) — returning the term describing the range of values for *Role* when the domain is restricted to *Concept*, and getClassified-Parents($<ConceptId>$) — returning the identifiers of immediately subsuming concepts in the classification hierarchy typically computed by DL reasoners.

Using the word "term" to refer to concepts, classes, properties, associations, predicates, etc. let us distinguish *term-returning operators* — ones that return either a term or an aggregate of terms (set,list, or tuple); getRange and getImmediateParents are examples of such operators. By analogy with Abstract Data Types, we will call *observers* the other operators, especially ones checking the truth of some formula (such as subsumption).

We will say that a set of identifiers S is closed under term-returning operator *op* for KB if, whenever *op* is invoked on KB with arguments from S, and it returns an identifier B, this also belongs to S. (B might be an element of the list or tuple returned by *op*.)

The material to be imported will then be specified in two steps: first, the set of names Ω of interest will be enlarged, using term-returning operators; then, a minimal set of formulas, over this extended vocabulary, is selected from those entailed by or contained in the exporting knowledge base, so that it produces the same answers for observer Ask operators.

Ignoring the issue of differing logics in the exporting and importing ontologies, we propose the following definition:

Definition 2. *Let KB_{expt} be the exporting knowledge base represented in some formalism L_{export}; $\mathcal{A}$ a set of Ask operators; and $\Omega \subseteq vocab(KB_{expt})$ a set of term identifiers.*

Let $\Omega_\star$ be the minimal set of identifiers from KB_{expt} that contains Ω and is closed under term-returning operators in $\mathcal{A}$. "`import` Ω `from` $\mathtt{KB}_{expt}$" is then a minimal set F of formulae having the properties that (i) vocab(F) $\subseteq \Omega_\star$; (ii) $KB_{expt} \models F$; and (iii) for every KB_{impt} such that $vocab(KB_{impt}) \cap vocab(KB_{expt}) \subseteq \Omega_\star$, the answers to all observer operators in $\mathcal{A}$ are identical for $KB_{impt} \cup F$ and $KB_{impt} \cup KB_{expt}$ as long as only symbols in $\Omega_\star$ are considered.

4 Reconstructing Some Previous Examples of Importing

Different import situations are described by the form of the contents of the knowledge base (L_{export}), the term-returning operators in $\mathcal{A}$ that generate $\Omega_\star$, and the observer operators in $\mathcal{A}$ that determine what formulas to import. Of course, a premium is placed on using Ask operators that are independently motivated!

In the case where KB_{expt} is a set of formulas, and we want to import its logical consequences concerning the symbols in Ω, we get the notion of *uniform interpolant* of $(\mathrm{KB}_{impt} \cup \mathrm{KB}_{expt})$ w.r.t. Ω [9]. Such uniform interpolants are guaranteed to exist for less expressive formalisms such as propositional logic and the μ-calculus, though not for FOL. To get a uniform interpolant, it is sufficient to have $\mathcal{A}$ be the obviously useful observer operator isEntailed?($<Formula>$). In some sense, this is the "gold standard" sought by all of the other proposals, though it is usually only approximated.

On the other hand, if one seeks only the formulas syntactically involving the identifiers in Ω, then a simple listToldFormulas() operator would do the job, since the above definition minimizes the material imported from KB_{expt}.

When importing information about Ω from standard taxonomies, such as WordNet, the simplest proposal is to import all subsuming concepts and their IsA-relationships, as in [15]. Assuming that WordNet is represented as the collection of assertions $isa(B, C)$, representing direct super/sub-concept relationships in the taxonomy, $\Omega_\star$ can be specified by the closure of the getImmediateParents($<ConceptNode>$) term-returning operator. In turn, the projection of IsA^* on the set $\Omega_\star$ can be obtained by using the observer subsumedBy?($<ConceptNode>$, $<ConceptNode>$).

In Navigli's proposal, the situation is more delicate because not all IsA-ancestors are included in $\Omega_\star$. The crucial observation is that nodes with 2 or more children in a taxonomy (the set S_2 in Section 2.1) correspond (mostly) to least common subsumers (lcs) of pairs of concepts whose subsumption paths meet at that node for the first time.[2] Therefore, if we use getLeastCommonSubsumer($<ConceptNode>$,$<ConceptNode>$) as term-returning operator in $\mathcal{A}$, the least fixedpoint starting from Ω will produce S_2. Note that the lcs operator is of independent interest, and has been used, among others, in computing concept similarity in term taxonomies.

[2] Cases where two paths from one concept C meet at an ancestor A seem to us to be suspicious since children in taxonomies tend to be disjoint.

In KnowledgeBus, the knowledge base contains information about the concept taxonomy (in the form of $genls(B, C)$ atoms), the signature of predicates, and general FOL axioms ψ. As in Section 2.2, we need to obtain the sets of concepts and general predicates S^* and P^*. Since these are of different sorts it is possible to mix them together in the same set, and obtain their union by repeatedly applying the following 3 term-returning operators: getImmediateParents($<Concept>$) as before; getPredicates($<Concept>$) — returning predicates that have $Concept$ as the type of at least one of their arguments; and getType($<Predicate>$) — returning the tuple of concepts representing the type signature of the predicate.[3] Once $\Omega_\star$ is computed, [13] retrieves existing formulas involving these symbols, which can be simulated using listToldFormulas(), as above.

The proposal by Conesa and Olivé differs from KnowledgeBus because in Axioms1 *all* concepts in the formula must be generalizations of Ω. This is justified by the fact that the formulas are used not for deduction (in which case other, intermediate concepts could be needed) but for integrity checking. It means that $\Omega_\star$ will not contain all concepts obtainable via getImmediateParents. So we need to take two useful operators: getInheritedConstraints($<Concept>$) — returning all formulas inherited from super-classes; and hasVocabulary($<Formula>$); and then compose them into getInheritedIC, which will return only Axioms1. hasVocabulary and getInheritedIC are then sufficient to generate $\Omega_\star$, and subsumedBy? ensures, as usual, sufficient subclass axioms, while listToldFormulas over $\Omega_\star$ produces the other desired axioms. Note that the "implementation" of this approach (see S_2, S_3 in Section 2.3) is even harder to capture in our framework, though S_3(ii) is lcs again. We might take this to be less a fault of our definition, and more a sign of problems with the proposal.

Please note that if we are willing to accept specially crafted term-returning operators, then we can also capture previous modularization proposals such as those of Noy&Musen [11] and Seidenberg&Rector [14]. Lack of space prevents further discussion of this.

5 Summary

To summarize, we have proposed a generic definition for importing from knowledge bases having Levesque-style Ask operators. The definition relies on term-returning operators to obtain an enlarged set of concepts to consider, and observer operators to select axioms or their consequences. We have shown how the definition captures interesting aspects of some prior approaches, and in the process discovered the important role played by the notion of lcs in some schemes. The main work, including the use of this definition to study importing from specific kinds of knowledge bases, and connecting lcs with predicate/theory abstraction, is still ahead of us.

[3] Note that this does not eliminate data type arguments, but arguably this is in fact correct: predicates $p : Person \times Integer$ and $q : Office \times Integer$ can in fact be "joined" on their second argument, connecting Person and Office.

Acknowledgments: Borgida's work is supported in part by the U.S. DHS under ONR grant N00014-7-1-0150. Giunchiglia's work is supported in part by the EU-funded Open Knowledge project (http://www.openk.org/).

References

1. J. Bao, D. Caragea, V. Honavar: On the Semantics of Linking and Importing in Modular Ontologies. *ISWC 2006*: 72-86
2. A. Borgida "On Importing Knowledge from DL Ontologies: intuitions and problems", *DL 2007*
3. J. Conesa, A. Olivé: A Method for Pruning Ontologies in the Development of Conceptual Schemas of Information Systems. *J. Data Semantics V*: 64-90 (2006)
4. J. Conesa, A. Olivé: Pruning Ontologies in the Development of Conceptual Schemas of Information Systems. *ER 2004*: 122-135
5. B. Cuenca Grau, B. Parsia, E. Sirin: Working with Multiple Ontologies on the Semantic Web. *ISWC 2004*: 620-634
6. B. Cuenca Grau, B. Parsia, E. Sirin, A. Kalyanpur: Modularity and Web Ontologies. *KR 2006*: 198-209
7. B. Cuenca Grau, I. Horrocks, Y. Kazakov, U. Sattler. "A Logical Framework for Modularity of Ontologies", *IJCAI 2007*: 298-303.
8. B. Cuenca Grau, I. Horrocks, Y. Kazakov, U. Sattler. "Just the Right Amount: Extracting Modules from Ontologies", *WWW 2007*: 717-726.
9. G. D'Agostino, M. Hollenberg. "Uniform interpolation, automata and the modal mu-calculus". *Advances in Modal Logic*, Vol. 1, 1996.
10. H. J. Levesque: "Foundations of a Functional Approach to Knowledge Representation". *Artif. Intell. 23(2)*: 155-212 (1984)
11. N. Fridman Noy, M. A. Musen: "Specifying Ontology Views by Traversal". *ISWC 2004*: 713-725
12. R. Navigli, "Extending, Pruning and Trimming General Purpose Ontologies". *IEEE SMC 2002*.
13. B. J. Peterson, W. A. Andersen, J. Engel, "Knowledge Bus: Generating Application-focused Databases from Large Ontologies", *KRDB'98*.
14. J. Seidenberg, A. L. Rector: "Web ontology segmentation: analysis, classification and use", *WWW 2006*: 13-22
15. W. R. Swartout, R. Tatil, K. Knight, and T. Russ, "Toward Distributed use of Large-Scale Ontologies",. *Proc. Knowledge Acquisition for Knowledge-Based Systems Workshop*, 1996.