

Reusing JessTab Rules in Protégé

D Corsar and D Sleeman

Computing Science Department, University of Aberdeen, Aberdeen AB24 3UE, UK
{dcorsar, sleeman}@csd.abdn.ac.uk

1 Introduction

Due to their ability to provide an explicit definition of a domain, ontologies are one of the most commonly used forms of domain knowledge representation. The popular Protégé environment [11] provides an extensive ontology and knowledge base authoring and maintenance environment in one tool. One way to reason over this knowledge base is provided by the JessTab Protégé [1] plugin, which provides a bridge between the Java Expert System Shell (JESS) [3] rule engine and the Protégé knowledge base. However these rules are explicitly tied to the knowledge base for which they were developed as they are required to name particular classes and slots. Although an unavoidable requirement, this can greatly hinder reusing a set of JessTab rules developed for one ontology/knowledge base with additional knowledge bases. In particular, the developer could manually define mappings between the class and slot names in the JessTab rules and those in the second and subsequent knowledge bases. However this would be a tedious and very error prone process.

We have developed a Protégé plugin which supports the user in achieving reuse of a JessTab rule set. By automating many of the procedures, particularly the mapping stage, that the user would otherwise be required to perform manually, our plugin simplifies and accelerates the process of modifying a JessTab rule set for reuse.

In section 2 we discuss some current ontology mapping and merging tools; in section 3 we outline two scenarios where our tool could be used; in section 4 we briefly outline our tool's functionality; in section 5 we describe experiments we have performed and report some results. In section 6 we discuss some modifications inspired by our experiments which should improve our tool's performance and in section 7 we provide a summary of our findings.

2 Related Work

Our semi-automated reuse approach makes use of techniques originally developed in the ontology mapping, merging and alignment fields. There have been various approaches to these tasks including use of specially designed algebras [7], use of lexical analysis of the concept names in the two ontologies, as well as having the user manually define mappings with the aid of a specially designed user interface. Below we focus on tools using the latter two approaches.

A popular web-based ontology management tool with the facility to assist with merging is Stanford KSL's Chimaera [5]. Chimaera provides limited support with ontology mapping, by allowing multiple ontologies to be loaded, automatically examining their concept names and providing the user with a list of similarly named concepts. The principal mapping approach used here is the detection of common substrings. This approach is very effective for pairs of concepts that have related names, but clearly is ineffective when the same concepts are expressed using synonyms, for example "person" and "individual."

Another Stanford product, PROMPT [9] provides a suit of ontology management tools, in the form of a further Protégé-2000 plugin. One of these tools, iPROMPT [8], was developed as an interactive ontology-merging tool, which assists the user by providing suggestions for merging, analyzing any resulting conflicts and suggesting relevant conflict resolution strategies.

When determining its suggested mappings, iPROMPT makes use of two factors: a measure of linguistic similarity between names, combined with the internal structure of concepts and their location in the ontology. The similarity measures of iPROMPT are currently based solely on substring matching of the various concept names; and so consequently it suffers similar problems to Chimaera.

One tool which uses lexical databases is ONION [6]. The ONION algorithm was designed to address the problem of resolving semantic heterogeneity amongst ontologies. Mitra and Wiederhold implemented two methods of addressing this task: one is based, like ours, around a lexical database, while the other uses domain specific corpora. Briefly ONION's algorithm evaluates two expressions for similarity and assigns them a similarity value. When the two terms being compared are not identical, ONION makes use of either a thesaurus (WordNet [2]) or a corpus in assigning the value. If the value is above a user set threshold, then the mapping is accepted.

The results of Mitra and Wiederhold's experiments indicated that generally the corpus based mapping approach produced more accurate mappings than the thesaurus technique. However, the main difficulty with this approach is generating the corpus. Mitra and Wiederhold searched the internet to find sources that were both relevant to the domain of the ontologies and contained some of their key concepts. Various sized corpora were used ranging from 50 to 1000 webpages. Although this approach provided the more accurate results, the initial costs necessary to acquire such a corpus would no doubt be very high, making this an infeasible approach for supporting mappings between many domains.

3 Example Scenarios

There are many scenarios where reuse of an existing set of rules could be beneficial. Practically any developer building a new application requiring similar or identical rules to those used in a previously developed application could benefit from the use of our tool. By semi-automating the rule reuse process, the developer could benefit from, amongst other things, reduced design, implementation and testing costs.

Below we outline two examples where our tool has been used. The first example involves building simulations of two seemingly disparate areas - shopping and water treatment, while the second emphasises the reuse of a problem solving method (ruleset) in two distinct route planning applications.

3.1 Simulating Shopping and Water Treatment

Our first example involves the creation of two different simulations, which at their core use similar procedures, meaning their development could be achieved by using our approach. At a very abstract level, the day to day activity of a retail store is

1. it receives stock from some supplier(s),
2. it stores this stock in a store room or on shelves,
3. customers buy items reducing the stock level,
4. it orders new stock,
5. return to step 1.

In this problem, the environment (the retail store) can be easily represented by an ontology. Likewise the processes noted above can be simulated by a series of JessTab rules coded to run against that ontology.

At a similarly abstract level, water treatment plants feature a similar underlying processes

1. receives water from sources,
2. stores it in treatment tanks,
3. passes the results onto the desired locations,
4. signals that it can process more water,
5. return to step 1.

Again, the environment (the water treatment plant) can easily be represented by an ontology and the processes by JessTab rules. However, as the underlying processes in water treatment are in essence very similar to those of the store, our tool could be used to reconfigure the JessTab rules implemented for the store to provide similar functionality for the water treatment plant. Although these rules will not provide all the detail required by the simulation, it could form the basis of it and in so doing reduce its development costs.

3.2 Route Planning

Our previous example illustrated reuse of the same set of rules in building simulations in two different domains. Our second shows the reuse of a route planning problem solver with two distinct applications, based on two different ontologies.

Route planning has been a highly research field in computing for many years, during which numerous algorithms have been developed to address, what is essentially a search problem. Typically the main objective is to find the best path from one location to another within some environment. The notion of “best” path is typically determined by the inclusion of a suitable configured evaluation metric, applied when deciding which potential path to investigate first.

Although typically a graph is used, it is possible to represent the environment with an ontology, in which classes represent locations whose attributes provide details of locations reachable from it. Given such an ontology, a route planning algorithm could be implemented in the form of a set of JessTab rules to carry out various planning tasks using the ontology.

For example, if a retail company were to represent its warehouse in an ontology, describing the items stored in it and the location within the warehouse, a JessTab rule base could calculate the shortest path around the warehouse for collecting items to fulfil a series of orders. The same rules could be reused with a second ontology representing a courier company's deliveries and their locations. Our tool could help tailor the original planning algorithm for use against the deliveries ontology to calculate the cheapest route for delivering orders.

4 The JessTab Rule Reuse Process

We have developed a rule reuse process composed of two distinct phases. The first phase consists of two actions: the extraction of class names, slot names, and slot types from the relevant ontology¹; and the generalisation of this information to produce a set of ontology independent rules. During the second phase the independent rules are mapped to the new ontology to which they are to be applied. Fig. 1 shows the processes involved in our reuse process and the relationships between them.

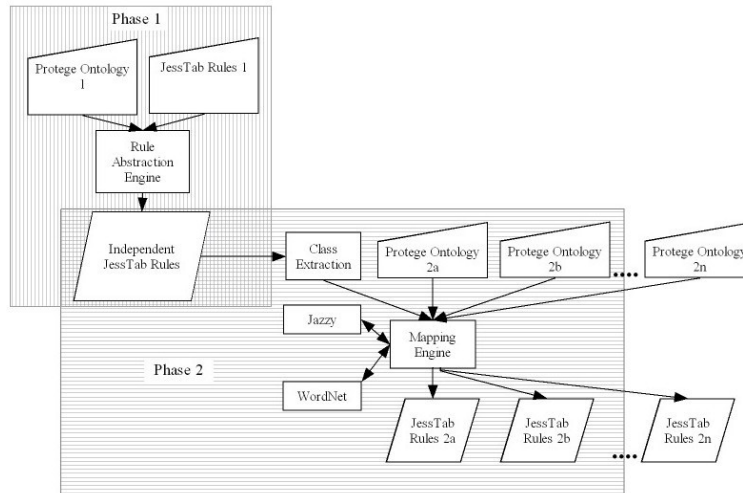


Fig. 1. Illustration of the JessTab rule reuse process

¹ If the relevant ontology is not available, the class and slot names could be extracted from the rule set, however the slot data type information would not be available.

4.1 Phase 1 - Rule Abstraction

The rule abstraction phase is the first stage and is identified in Fig. 1 by the vertical stripe filled box in the upper left corner labelled “Phase 1.” During this phase a set of JessTab rules and the corresponding ontology are passed to the Rule Abstraction Engine. Here the tool extracts from the ontology all the class names along with all the slot names and each slot’s data type. The JessTab rules are then rewritten replacing each class and slot name with a more general form. This removes all the references to the original ontology, making the resulting rule set more abstract and ontology-independent.

The class and slot names have similar abstract names. Abstracted class names consist of a predefined prefix (the default is “XX_”) concatenated with the original class name. For example the class name “Person” is given the abstract name “XX_Person”. Slots are treated similarly; their abstracted name consists of the prefix, the slot name, and the slot data type. For example an Integer slot with the name “age” is given the abstract name “XX_age_Integer”.

4.2 Phase 2 - Rule to Ontology Mapping

After completion of Phase 1, the independent rules can be mapped to further ontologies during Phase 2. First the Rule Extraction component builds a list of the classes and slots the independent rules refer to. The tool then attempts to map these classes to those in the second ontology. This phase is shown in the horizontal stripe filled box in the lower right corner of Fig. 1 labelled “Phase 2.”

The basic mapping algorithm is:

1. Extract all the classes and their associated slot information from the abstract JessTab rules and store them in a list.
2. For each class extracted from the abstract JessTab rules
 - (a) Find the most suitable class in the new ontology to map to, and suggest the individual (slot and class name) mappings.
3. Update the rules to reflect suggested mappings and allow the user to complete and/or correct the mappings.

Clearly the key step is 2a - finding the most suitable mappings for the classes. We derive our suitability ratings by applying the commonly used lexical analysis techniques of partial and exact string comparisons, spell checking (provided by Jazzy [12]) and synonym lookup in WordNet (accessed via JWord [4]). For a mapping to be suggested the calculated similarity rating for two (class or slot) names must exceed a threshold set by the user. Furthermore, to avoid data type errors when the rules are run, slot mappings are only suggested between slots with the same type. When determining potential mappings, our tool compares every slot of every extracted class, along with the class name, with those in the ontology. If two slots are suitable for mapping they are given a classification based on how that mapping was achieved (the classification mechanisms are discussed below). At this level a mapping simply consists of a *from string* (the name from the abstract JessTab rules), a *to string* (the name from the new ontology) and the mapping classification. Once this process has been performed

for all the classes the “best” mapping (if multiple mappings have been found) is proposed to the user as the most likely mapping for his configuration.

We define three distinct mappings operators (for examples of each see Table 1); these are:

1. *Direct mapping.* These consist of two identical strings, or where one string is a minor variant of the other.
2. *Constituent mapping.* These consist of two strings that share at least a (settable) percentage of constituents.²
3. *WordNet mapping.* Here the constituents of each string are extracted, and their synonyms found in WordNet. In a Wordnet mapping the percentage of synonyms in the *from string* appearing in the constituents of the *to string* (and vice versa) must be greater than a (settable) threshold percentage.

In principle each mapping between an extracted class and a new ontology class consists of direct, constituent and WordNet mappings. As noted earlier each of the mappings will return a value, the largest value (providing it’s over a threshold) is the one which is recommended to the user. In the case of a numerical tie, the algorithm selects the mapping by applying the following precedence rule:

Direct Mappings > Constituent Mappings > WordNet Mappings

Currently the tool supports mapping each extracted class to a single class in the ontology. Given the above procedure it is understood that it may result in some classes not having a mapping as the algorithm searches to optimise the value of each particular pair. A slightly more sophisticated algorithm might suggest sub-optimal mappings for some pairs, and hence produce some additional acceptable mappings. This more sophisticated mapping algorithm will be developed subsequently.

5 Results

To evaluate the performance of our tool, we conducted a series of experiments designed to test first the rule abstraction process and second the mapping capabilities. Each test involved the development of an initial JessTab rule set, which was based around an initial ontology. This ontology was either created by ourselves, or downloaded from the Stanford KSL OKBC server with the Protégé OBKB Tab plugin[10]. Once each rule set had been developed to a satisfactory level, we built further ontologies with which to test the mapping functionality.

Table 1³ provides a selection of the results from our tests with the Document ontology downloaded from the Stanford KSL OKBC server’s Ontolingua section.

These results offer a good illustration of the type of support the user can expect from our tool. The “Document” concept in Table 1 demonstrates nicely the direct mapping: in Application A the name is identical, in Application B the

² The constituents of a string are the words that make up that string. We use the symbols ‘-’, ‘_’ and (in mixed case strings) upper case letter to denote the start of new words. For example the string *date-of-birth* has constituents *date*, *of* and *birth*.

³ The spelling mistakes in Table 1 are intentionally left as they illustrate the minor variant checking feature of direct mappings.

Name	Data Type	Abstract Name	Mapped in to			
			Application A		Application B	
			Name	Mapping Type	Name	Mapping Type
Document	n/a	XX_Document	Document	Direct	Docment	Direct
Thesis	n/a	XX_Thesis	Thesis	Direct	Dissertation	WordNet
Doctoral- Thesis	n/a	XX_Doctoral- Thesis	DoctoralThesis	Constituent	Dissertation- Doctoral	WordNet
Miscellaneous- Publication	n/a	XX_Miscellaneous- Publication	AssortedPublish- ing	WordNet	Miscellaneous- Publichings	WordNet
Artwork	n/a	XX_Artwork	Art	WordNet	Artistry	Manual
Computer- Program	n/a	XX_Computer- Program	ComuterProgram	Direct	Computer- Programme	WordNet
Has-Author	String	XX_Has- Author_String	Author	Direct	HasWriter	WordNet
Has-Editor	String	XX_Has- Editor_String	HasEditor	Direct	Editor-Of	Manual
Title-Of	String	XX_Title-Of_String	Title	Constituent	TitleOf	Direct
Publication- Date-Of	String	XX_Publication- Date-Of_String	IssueDate	WordNet	Date-Of- Publication	Constituent
Publisher-Of	String	XX_Publisher- Of_String	PublicationHouse	Manual	PublisherOf	Direct

Table 1. Suggested mappings for selected classes and slots of the OKBC Document ontology

minor spelling error in “Document” is picked up by our tool and has no adverse effect on the suggested mapping. Further Table 1 exhibits many examples of constituent mappings, while the WordNet mappings also feature prominently both with one-word name examples (“Thesis” to “Dissertation”) and multi-word concept names (“Miscellaneous-Publication” and “AssortedPublishing”).

6 Discussion

Overall we were very pleased with the outcome of our experiments. However, they do suggest that some modifications to the mapping phase could be beneficial. One significant change would be to allow the user to specify how deep the tool searches WordNet for synonyms. This is best described by means of an illustration. Take the “Artwork” concept in Table 1 line 5. In Application B the user was required to manually map this concept to “Artistry.” When suggesting a mapping, the tool looked up synonyms of artwork in WordNet, which returned “artwork,” “art,” “graphics” and “nontextual matter.” Had the tool searched another level, it would have found “artistry” is a synonym of “art” and suggested a mapping. However, performing deeper searching like this could dramatically increase the time taken by the automatic mapper.

The mappings proposed between the rule set and the new ontology are strictly 1:1 at present. As noted in section 4.2, choosing a suboptimal mapping for one

pair, might, in fact, allow more variable pairs to be mapped; that is, evaluating the effectiveness of mappings should be subjected to global optimisation and not just to optimising mappings between particular variable pairs.

7 Summary

The experiments we have performed have provided favourable results, although there are some enhancements which could be made. However, by automating part of the reuse process, our tool can be of considerable use to a developer wishing to reuse a set of JessTab rules with further ontologies.

8 Acknowledgments

This work is supported under the Advanced Knowledge Technologies (AKT) IRC (EPSRC grant no. GR/N15764/01). See <http://www.aktors.org>

References

1. Henrik Eriksson. The JESSTAB Approach to Protégé and JESS Integration. In *Proceedings of the IFIP 17th World Computer Congress - TC12 Stream on Intelligent Information Processing*, pages 237–248. Kluwer, B.V., 2002.
2. Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, Cumberland, RI, 1998.
3. E. Friedman-Hill. *Jess In Action: Rule-Based Systems in Java*. Manning Publications Co., Greenwich, CT, 2003.
4. Kunal Johar and Rahul Simha. Jword 2.0. <http://www.seas.gwu.edu/~simhawe/software/jword/index.html>.
5. Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder. An Environment for Merging and Testing Large Ontologies. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, pages 483–493. Morgan Kaufmann, 2000.
6. Prasenjit Mitra and Gio Wiederhold. Resolving Terminological Heterogeneity in Ontologies. In *Proceedings of Workshop on Ontologies and Semantic Interoperability at the 15th European Conference on Artificial Intelligence (ECAI 2002)*, 2002.
7. Prasenjit Mitra and Gio Wiederhold. An Ontology-Composition Algebra. In Stefan Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 93–116. Springer, 2004.
8. Natalya F. Noy and Mark A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, 2000.
9. Natalya F. Noy and Mark A. Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, 2003.
10. Michael Sintek. OKBC Tab Website. <http://protege.stanford.edu/plugins/okbctab/okbc.tab.html>.
11. Stanford Medical Informatics, Stanford University. Protégé Website. <http://protege.stanford.edu>.
12. Tom White. Can't Beat Jazzy: Introducing the Java Platform's Jazzy New Spell Checker API. <http://www-106.ibm.com/developerworks/java/library/j-jazzy/>.