Implementing Interoperability through an Ontology Importer for Amine

Saleh Abdulrub[1], Simon Polovina[1],
Ulrik Sandberg-Petersen[2], and Richard Hill[1]

[1] Faculty of Arts, Computing, Engineering & Sciences,
Sheffield Hallam University, Sheffield, United Kingdom
Saleh.Abdulrub@student.shu.ac.uk,{s.polovina, r.hill}@shu.ac.uk
[2] Kaj Munk Research Center,
Department of Communication & Psychology,
Kroghstræde 3,
Aalborg University, DK-9220,
Aalborg, East Denmark
ulrikp@hum.aau.dk

Abstract. This paper investigates the need for tools that will facilitate a higher-order demand for interoperability between disparate systems. An ontology importer for Amine is described which enables ontologies written in linear form to be used with Amine version 4. Furthemore, the ontology importer is shown as an intermediary between CharGer and Amine, thus demonstrating interoperation between two conceptual structures tools, as well as discussed in a wider context by means of a Web service or by interoperating with Protégé-OWL.

1 Introduction

The Amine platform provides a comprehensive software suite for symbolic programming, intelligent system programming and intelligent agents programming[7]. It is primarily based on Conceptual Graphs (CG)[8]. As a core component it includes an ontology builder that is based upon a graphical user interface (GUI). The role of an ontology is key as it describes the concepts in a domain and any relationships between these concepts in this particular domain[3]. An ontology thus describes an existence of some particular objects in some domain and work can be derived from studying this domain. In CG an ontology consists of a type hierarchy that contains types that represent groups of entities with similar traits[5].

2 The Need

Whilst Amine's ontology builder is sophisticated, the technique used to develop an ontology using the Amine GUI can be quite time consuming as it involves much manual mouse pointing and clicking by the user. There are two further possible ways an ontology that on first sight appear to be more attractive. The first of these is generating or writing XML that is compatible with Amine. This method however is rather intricate as the XML needs to match the atypical ontology storage structure of Amine. Thus even with XSLT, in practical terms this is too low-level[1], [4]. The second way is by building an ontology programmatically by using Amine's API. Again this method is difficult and time consuming as, like the XML option, it requires the Amine-specific structure. The ontology importer addresses these concerns. It provides easier ontology development by giving, in contrast to Amine's current ontology builder, the ability to develop an ontology directly in CG in its linear form.

3 The Ontology Importer

The importer addresses the desire for CG software tools to interoperate. To have these tools in some sense compatible and functioning with each other is a key goal that would be welcomed by many in the CG community. This ontology importer is a small experiment that could be taken further in the future to achieve the end goal of interoperability between other CG tools.

3.1 As an Implementation

Currently the ontology importer consists of a GUI with a text-editor. An input in straight ASCII format is complied and accepted, then linking it to Amine's APIs. The ontology importer is fully functioning with error handling facilities. It is a Java library, and as such can be accessed from any Java application. One application that comes with the ontology importer for example is a text editor to input the text, which also contains output for informative messages. Currently the prototype accepts the input as a Java String. However the ontology importer will soon provide the facility for a user to enter the URL of a file and have the file compiled and if there are no errors in the input create an ontology. It is significant that the input to the ontology importer is a Java String. This is useful for the following reasons; firstly it enables easy creation of ontologies that have been automatically constructed from other sources, thereby enabling interoperability between Amine and other knowledge representation software. Secondly it enables quicker and easire creation of ontologies by using the keyboard rather than the mouse.

4 Prolog+CG 2.0

The syntax used by the ontology importer is modelled upon that of Prolog+CG 2.0 (hereafter referred to as Prolog+CG). Prolog+CG is a GC tool built in Java. It is similar to Prolog except it has extensions for handling CG. Professor Adil Kabbaj, the creator of Amine, was also the creator of Prolog+GG[6]. Prolog+CG's extensions enable it to handle CG ontologies that can be created, queried and results given. Prolog+CG is thus a CG programming environment. In Prolog+CG for example the following ontology could be written into a Prolog+CG text editor without any further programming:

```
Universal > Male, Female, Adult, Child.
Male > Man, Boy.
Female > Woman, Girl.
Child > Boy, Girl.
Adult > Man, Woman.
```

Instances can also be created easily in Prolog+CG. For example the following instances can easily be created in Prolog+CG:

```
Boy = Sam, Richard, Gary.
Girl = Greta, Rebecca, Sheila.
```

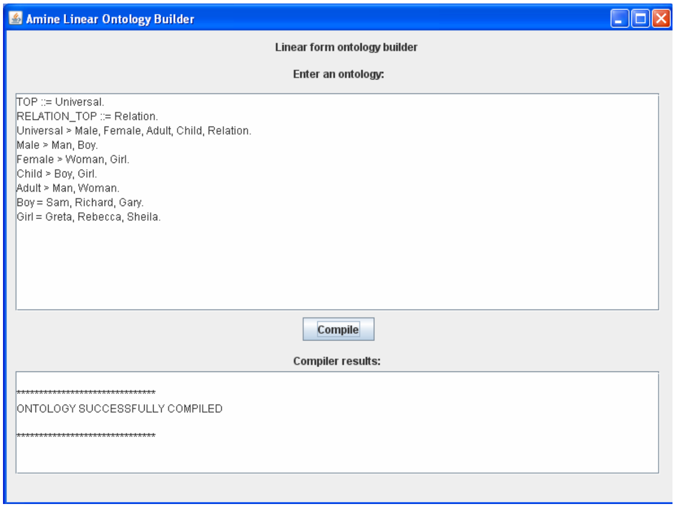
Like the ontology importer, Prolog+CG enables ontologies to be entered in CG linear form. However it is not interoperable in that it lacks a cohesive, distinct component for creating and interoperating ontologies. It also does not return an Amine API ontology or lexicon object that can then be used by Amine's ontology builder. Given all these factors, the ontology importer was designed as a distinct component from the outset rather than to retro-fit Prolog+CG's legacy version to the problem in hand.

5 Creating an Ontology with the Ontology Importer

As in Amine the topmost type and the topmost relation type must be specified at the start of the ontology. This is achieved by simply entering the word 'TOP' and specifying its identifier after the '::=' assignment of symbol. This process is the equivalent to the dialogue box that appears on Amine's ontology builder GUI that asks for the topmost type in the hierarchy. The topmost relation in the hierarchy must also be specified by entering 'RELATION_TOP' and passing the identifier name after the '::=' assignment symbol. The ontology importer then expects these to be the first types in the hierarchy. (NB If an identifier other than the TOP's identifier is the first in the hierarchy an error is flagged. If RELATION_TOP's identifier is a subtype of any type other than TOP's identifier an error is flagged. Similarly if TOP and RELATION_TOP are omitted the ontology would not compile). Adding an ontology using the prototype is thereby much simpler than using Amine's GUI; indeed one can simply paste an ontology into the text editor. The following ontology can then be created as shown in Figure 1:

```
TOP ::= Universal.
RELATION_TOP ::= Relation.
Universal > Male, Female, Adult, Child.
Male > Man, Boy.
Female > Woman, Girl.
```

```
Child > Boy, Girl.
Adult > Man, Woman.
Boy = Sam, Richard, Gary.
Girl = Greta, Rebecca, Sheila.
```



```
TOP ::= Universal.
RELATION_TOP ::= Relation.
Universal > Male, Female, Adult, Child, Relation.
Male > Man, Boy.
Female > Woman, Girl.
Child > Boy, Girl.
Adult > Man, Woman.
Boy = Sam, Richard, Gary.
Girl = Greta, Rebecca, Sheila.
```

```
*********************************
ONTOLOGY SUCCESSFULLY COMPILED
*********************************
```

Fig. 1. Importing an ontology.

6 Automating across Applications

An example of how to automate the ontology importer can be illustrated using CharGer as an example. CharGer is a CG software tool[2]. CharGer is used for CG operations similar to Amine. CharGer creates ontologies in graphical form and generates a linear form ontology from this graphical form. In CharGer instances are created by having the link in text format. Instances are shown in a rectangular box with the concept type name and the instance as the referent. The link is produced by giving the name of the concept type, thus in text format. The following linear form ontology is generated by CharGer:

```
Type Man is a kind of Male
Type Boy is a kind of Child
Type Female is a kind of Universal
Type Adult is a kind of Universal
Type Relation is a kind of Universal
Type Male is a kind of Universal
Type Child is a kind of Universal
Type Woman is a kind of Adult
Type Man is a kind of Adult
Type Boy is a kind of Male
Type Woman is a kind of Female
Type Girl is a kind of Female
Type Girl is a kind of Child

There is a Proposition where Boy Sam Boy Richard Girl Rebecca Girl
Sheila Boy Gary Girl Greta
```

The above code can be either automatically transformed or manually edited to produce the type hierarchy in the format accepted by the ontology importer. One current shotcoming of the ontology importer is that it requires the types to be specified top-down. Hence we need to reorder the list before processing with the ontology importer.

```
TOP ::= Universal.
RELATION_TOP ::= Relation.
Universal > Female.
Universal > Male.
Universal > Child.
Male > Man.
Child > Boy.
Universal > Adult.
Universal > Relation.
Adult > Woman.
Adult > Man.
Male > Boy.
Female > Woman.
Female > Girl.
Child > Girl.
Boy = Sam, Richard, Gary.
Girl = Rebecca, Sheila, Greta.
```

The above ontology now can be accepted and compiled by the ontology importer. Achieving the goal of automation across applications has been met, though there were some amendments that need to be done to the ontology in order for it compile. This is clearly a shortcoming which we intend to address in the near future.

7 Automating as a Web Service

The primary goal for this work was to be able to use Amine ontologies in Web Services, without using a mouse for data entry. Part of this goal has been achieved; since the ontology importer is a Java library, it can be incorporated into any Java

Web Service together with Amine. Similarly it would be useful to enable the creation of Amine ontologies as a web service. Once implemented this will achieve the goal of interoperability. Any application can then add a reference specifying the location of the Web Service and connect to the ontology importer's API. When connected to the importer's API the user is indirectly accessing Amine's APIs to create an ontology. An Amine ontology would then be returned, thus achieving the goal which was originally formulated.

8 Conclusion

The ontology importer presented in this paper provides Amine with a convenient plug-in tool for its ontology builder. This extends to Web Services and mainstream non-CG ontology tools such as Protégé-OWL. Future research would include the following. First, the restriction on the order of the lines needs to be lifted. Second, a convenient method needs to be added, such that a URL can be given to the ontology importer, which is then fetched and used as the input to the method which accepts a String. Third, a Web Service needs to be created which provides Amine ontology creation services through a Web Service interface. The ontology importer will be released as Open Source software, thereby benefitting the CG and broader knowledge representation communities.

References

1. Brady, N., Polovina, S., Shadija, D., Hill, R., (2006) 'Experiences and Lessons from the Practical Interoperation of CharGer with SeSAm', *Proceedings of the First Conceptual Structures Tool Interoperability Workshop (CS-TIW 2006)*, July 16, 2006, Aalborg, Denmark. de Moor, A., Polovina, S., Delugach, H., (Eds.), Aalborg University Press (ISBN: 87-7307-769-0), 32-47.
2. Delugach, H., (2006). 'CharGer - Conceptual Graph Editor'. [online] last accessed 8 April 2008 at: http://sourceforge.net/projects/charger/
3. Horridge, M.,(2008). 'A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools', [online] last accessed 25 March 2008 at: http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial-p4.0.pdf
4. Maybery, P., Polovina, S., (2007). 'The Extent to which Interoperability between CG and non-CG Tools can be Assessed against the Semiotic Ladder', *Proceedings of the 2nd Conceptual Structures Tool Interoperability Workshop (CS-TIW 2007)*, July 2007, Sheffield, UK. Pfeiffer, H., Kabbaj, A., Benn, D., (Eds.), Research Press International (ISBN: 1-897851-16-2), 35-44.
5. Petersen, U., (2008). Online Course in Knowledge Representation using Conceptual Graphs [online] last accessed 22nd March 2008 at http://www.huminf.aau.dk/cg/.
6. Petersen, U., (2004-2007). Prolog+CG 2.0 website. last accessed Accessed 26 March 2008 at: (http://prologpluscg.sourceforge.net/.
7. Pfeiffer, H., Kabbaj, A., & Benn, D., (2007). *Proceedings of the 2nd Conceptual Structures Tool Interoperability Workshop (CS-TIW 2007)*, July 2007, Sheffield, UK. Pfeiffer, H., Kabbaj, A., Benn, D., (Eds.), Research Press International (ISBN: 1-897851-16-2), 65-70
8. Sowa, J.F., (1984). Conceptual Structures, Addison-Wesley.