A practical exploration of ontology interoperability

POLOVINA, S., COOKE, J. and LOKE, J.

Published version

POLOVINA, S., COOKE, J. and LOKE, J. (2009). A practical exploration of ontology interoperability. In: Conceptual Structures: Leveraging Semantic Technologies. Lecture notes in computer science (5662). Berlin, Springer, 247-256.

A Practical Exploration of Ontology Interoperability

Simon Polovina, Jeremy Loke, and James Cooke

Cultural, Communication & Computing Research Centre (CCRC)
Sheffield Hallam University, Sheffield, United Kingdom
{s.polovina,j.loke}@shu.ac.uk, jcooke3@hera.shu.ac.uk

Abstract. ISO Common Logic (CL, ISO/IEC 24707:2007) offers the Semantic Web (SW) a new and powerful dimension in achieving the effective discovery, automation, integration, and reuse across applications, data and knowledge. The paper shows how it is possible to explore such interoperability through small scale exemplar projects. As Conceptual Graphs (CG) is a key technology in CL, we focused on the Amine CG software and for the SW we focused on the Protégé OWL software, exploring the possible mappings between ontologies captured in OWL and in Amine. Through this practical exercise the dimensions and extent of the desired interoperability could be demonstrated. This small but significant experiment provided a practical insight into how CG Tools can actually interoperate towards achieving the wider goal of Ontology interoperability between CL and the SW.

1 Introduction and Motivation

At the Panel Session during the Conceptual Structures Tools Interoperability Workshop (CS-TIW) at ICCS 2008 (www.inra.fr/iccs08/workshops) one of us (Polovina) expressed concern over the ongoing poor progress of Conceptual Graphs (CG) software tools' ability to interoperate with one another. It was shown how interoperability could be progressed in a practical way through small-scale exemplar projects such as an Ontology Importer for Amine [1]. One of the key ensuing comments that if we wanted interoperability was for individuals simply to do it for themselves. Also arising from the session was how CG tools could interoperate with the Semantic Web (SW), reflecting the desires and developments towards this goal [3,8]. Thus we had a spectrum from exploring interoperability that ranged from a being a 'cottage industry' to globally interoperating CG tools with the Web itself.

In this larger view, CG is a key technology in the ISO Common Logic (CL) standard (ISO/IEC 24707:2007)[4]. ISO CL propels 'non-SW' technologies like CG (and their tools) from being disparate cottage industries into the global arena of the SW. This arises from the fact that standards play a key part in the adoption of any technology however promising that technology is. Standards diminish the risk of being locked into a non-interoperable technology. Interoperability across standards offers any technology the most appropriate standard

S. Rudolph, F. Dau, and S.O. Kuznetsov (Eds.): ICCS 2009, LNAI 5662, pp. 247–256, 2009.

for it to belong to whilst allowing it to be used with technologies in other standards according to their individual strengths. ISO CL accordingly offers the SW a new and powerful dimension in achieving its aims of the effective discovery, automation, integration, and reuse across applications, data and knowledge. Interoperating W3C's SW recommendations with ISO CL offers the most lucrative route in best realising these aims.

The 'challenge' of 'doing it ourselves' may nonetheless continue to offer a worthwhile route in this larger endeavour. Given the previous experience of the Amine Ontology Importer project referred to earlier, and in taking up this remark, we present a further small-scale exemplar practical project. Namely we investigated how an ontology produced in an emergent ISO CL software tool can interoperate with a SW one through interoperating CG's Amine software (amine-platform.sourceforge.net) with the SW's Protégé OWL (Web Ontology Language) software (protege.stanford.edu).

2 CG and OWL

CG and OWL provide various layers of functionality for supporting ontologies. CG are a way of structuring knowledge in a form readable by both humans and computers. CG can be read in a graphical or linear manner [9]. CG are constructed from Concepts, Relations and Arcs. Concepts consist of a type name within a rectangle, they may also have a referent which refers to an individual or instance of that type. Relations are shown as the relation type name within a circle or ellipsis and refer to the relationship between the two concept types. Arcs are shown as arrows between the concepts and the relations, the direction of the arrow dictates which way the formal logic should be read. If the arrow is directed towards the relation then the relationship will generally be 'has a', if the arrow is directed away from the relation then the relationship will generally be 'which is/who is/is a'.

Thus `[Concept-1] -> (Relation) -> [Concept-2]` states that "Concept-1 has a Relation which is a Concept-2". `[Cut] -> (Inst) -> [Knife]` denotes that "Cut has an Inst(rument) which is Knife". (In CG is generally easier to start the description with the relation, so for this example we would state that "The Instrument of Cut is a Knife".) An introduction to CG provides a description in more detail including the use of referents [6].

In CG, an ontology is a hierarchy of Concept Types and Relations with Universal at the topmost super-type of all types and Absurd at the bottommost subtype of all types; as such an ontology denotes a 'catalogue of modes of existence' [9], and the purpose of an ontology is to model knowledge in a formally logical structure so that facts can be derived from it.

OWL is a SW technology (www.w3.org/TR/owl-features/) that uses Web technologies, such as XML and Uniform Resource Identifiers (URI, www.ietf.org/rfc/rfc2396.txt) which uniquely identify ontologies and elements within ontologies across the Web. The OWL language is based on the SW's XML/RDF Schema (www.w3.org/RDF). OWL is a high level mark-up language that can easily read by humans as well as computers in its raw form.

2.1 The Amine Suite

The Amine CG platform provides an entire suite of applications aimed at creating complex intelligent systems. Amine ontologies allow the use of CG in their structure both as 'Definitions' and 'Canons'. The difference between a Canon and a Definition is essentially that the former describes a good use of a CG whilst the latter is a definition of a given Concept or a Relation, and examples distinguishing the two can be found (e.g. www.huminf.aau.dk/cg/Module_III/1152.html). Canons and Definitions provide an invaluable and powerful basis to structure a model of knowledge based on the CG ontology format described above. Each of these nodes can contain a Canon or a Definition in the form of a CG which relates to other nodes in the structure to form the inherent conceptual links which the ontology carries. The Amine ontologies are stored as XML files, and it is through parsing between Amine's Ontology XML format and Protégé OWL's XML format that interoperability across the two ontology formats is explored.

2.2 Protégé OWL

Protégé OWL is a platform which enables creation and manipulation of ontologies in the W3C's OWL format. This platform provides an intuitive interface enabling a graphical representation of an OWL ontology. It may include descriptions of classes, properties and their instances that are used to build the model of knowledge used to reason facts. There are three subsets of the OWL language: OWL lite, OWL DL and OWL full. OWL DL was judged to be the most appropriate level by which to investigate interoperability across the two ontology formats as it supports Description Logic (The 'DL' in OWL DL), given the previous correspondence between CG and DL [2]. In passing, OWL uses URI to identify objects it allows inter ontology transfer of data over the Web, a simple functionality that does not exist in the Amine platform at present but could easily be implemented.

3 Mapping between Amine Ontology and Protégé OWL

We based our approach on JXML2OWL, a project using a Java XML DOM parsing approach to migrate data from a standard XML data structure to the OWL Web Ontology Language. However we used "LINQ To XML" parsing in VB.NET to conduct the mappings [5,7]. This choice capitalised on the particular experience that one of us had (Cooke). We accordingly explored the pathways between a Protégé OWL and Amine ontology, creating it in one of them and testing the parsed ontology in the other. Our findings were as follows.

3.1 Amine Concepts to OWL Classes

Amine Concept Types. In Amine, concept types represent a type of a concept. The concept types build up a hierarchy of concepts in Amine through their children and fathers. For example, the following XML describes the concept type 'City', and its father (Universal):

```
<Child>
  <Key>City</Key>
</Child>
<Fathers>
<Father>
  <Key>Universal</Key>
</Father>
```

OWL Classes. To construct the same hierarchy, OWL uses Classes, and the subClassOf construct to define the Super-Class (or in Amine terms, the 'father'). For example, the following XML describes the classes Universal and City and the hierarchy between them via the subClassOf construct:

```
<owl:Class rdf:ID="Universal" />
<owl:Class rdf:ID="City">
 <rdfs:subClassOf rdf:resource="#Universal"/>
</owl:Class>
```

Analysis. Concept types in Amine can be mapped across to OWL easily as Classes, in addition the structure of the Concept Types can be translated using OWL's (or to be specific, RDFs) subClassOf feature. Using this simple mapping, the Amine ontology type hierarchy can be translated into an OWL file and still maintain its Sub-Type/Super-Type structure, this file can then be modified or expanded using the Protégé OWL Suite. This is achieved by parsing the Amine XML and finding all of the `<Type>.<Key>` nodes. Then within each node find all of the `<Fathers>.<Father>.<Key>` nodes. If there are no `<Fathers>.<Father>.<Key>` nodes present in a `<Type>` node then there is no subClassOf construct in the OWL XML, illustrated by Figure 1.

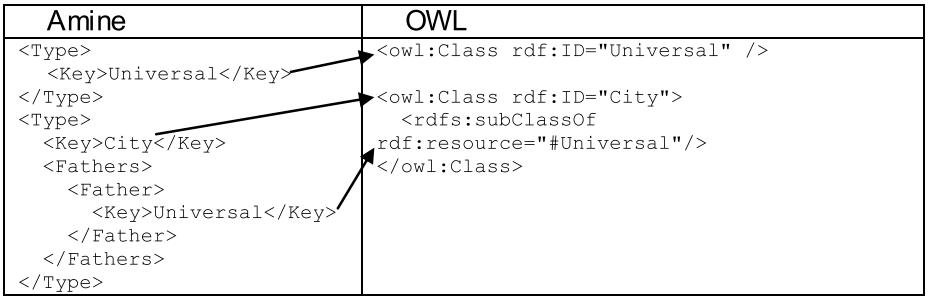

Fig. 1. Amine Concept to OWL Classes

3.2 Amine Relation Types to OWL Properties

Amine Relation Types. In Conceptual Graph theory a relation type is a type of relation which can contain a CG. Amine defines a Relation Type as a conceptual structure which is always a subtype of the special 'Relational Root'. In line with CG theory, Amine allows a relational type to have a CG. For example,

the following XML is direct output from Amine and shows the definition of the Relation Type 'Agnt' and that it's Super-Type is 'Relation':

```
<RelationType>
  <Key>Agnt</Key>
  <Fathers>
    <Father>
      <Key>Relation</Key>
    </Father>
  </Fathers>
</RelationType>
```

OWL Properties. To construct the same hierarchy, OWL uses Properties, and the subPropertyOf construct to define the Super-Class (or in Amine terms, the "father"). For example, the following XML describes the Properties 'Relation' and 'Agnt' and the hierarchy between them via the subPropertyOf construct:

```
<owl:ObjectProperty rdf:ID="Relation" />
<owl:ObjectProperty rdf:ID="Agnt">
  <rdfs:subPropertyOf rdf:resource="#Relation"/>
</owl:ObjectProperty>
```

Analysis. An Amine ontology can include conceptual Relation Types, integrating a relational type hierarchy with the Concept Type hierarchy to capture the relations between the concept types [9]. These can be mapped to OWL properties and using RDF's subPropertyOf feature, enabling the Sub-Type/Super-type structure of the type hierarchy to be maintained. These similarities lead us to believe that there is a pathway to convert Amine Relation Types to OWL Properties through 2 stages:

1. Parse the XML to find the `<RelationRoot>` node and its value, then create a OWL property with no domain or range as the root property.
2. Parse the XML and find all of the `<RelationType>.<Key>` nodes (except for the Relation Root) then search within each of the `<RelationType>` nodes (except for the Relation Root) for `<Fathers>.<Father>.<Key>` nodes then writing the OWL ObjectProperty and subPropertyOf.

Figure 2 provides an example.

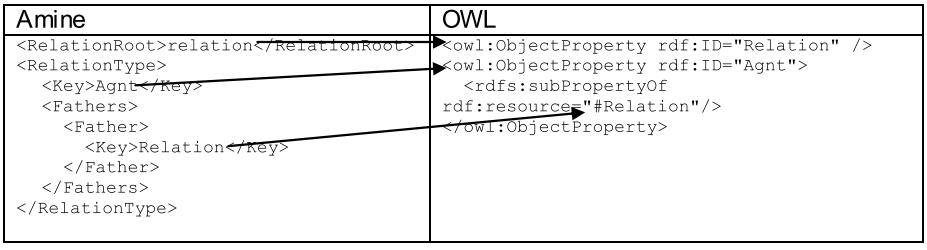

Fig. 2. Amine Relation Types to OWL Properties

3.3 Amine Individuals to OWL Individuals

Amine Individuals. In Amine, an individual is a specific member of a group. For example, the following XML describes the Individual 'John' and that he is a member of the type 'Person':

```
<Individual>
  <Key>John</Key>
  <Fathers>
    <Father>
      <Key>Person</Key>
    </Father>
  </Fathers>
</Individual>
```

OWL Individuals. To construct the same hierarchy OWL uses the coincidently named Individuals, the definition of OWL Individuals is simpler than Classes and Properties. For example:

```
<Person rdf:ID="John"/>
```

Analysis. In an Amine ontology an individual is an instance of a concept type, OWL also has individuals which are defined as instances of classes. Following our mappings thus far this is a natural conversion. First we must parse the XML and find all of the `<Individual>` nodes, for each node write the `<Individual>.<key>` value as the Individual's name then Write the `<Individual>.<Fathers>.<Father>.<key>` value as the name of the class this individual belongs to. Figure 3 provides an example.

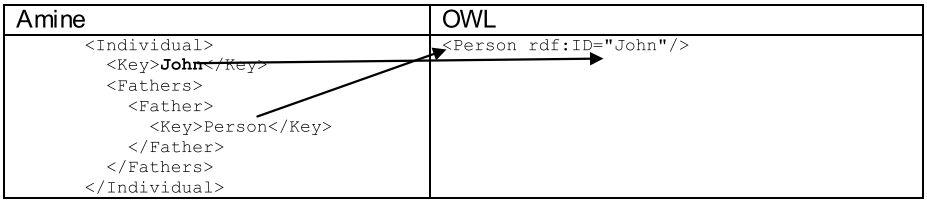

Fig. 3. Amine Individuals to OWL Individuals

3.4 Amine CG to OWL Mapping

So far we have proposed pathways/mappings to convert between an Amine ontology and Protégé OWL. However we noted that these mappings do not take advantage of the expressiveness of CG and their use within Amine. Therefore we lay out a method for translating CG in an Amine ontology to an appropriate structure within an OWL ontology.

In an Amine ontology Concept Types and Relation Types have the capability to contain both definitions and canons in the form of CG; these CG connect the nodes together and define the relationships between them. Here a parallel can be drawn with properties of OWL.

Figure 4 tabulates the mappings.

Amine ->	-> OWL	Comments
Concept Type	Class	Mapping is simple although Universal name will be lost.
Relation Type	Properties	In OWL, properties are dyadic; in CG, relations are n-adic. We thus used Class conditions to assert relations.
Individuals	Individuals	Maps
Definition	Class Conditions	Definition CG is parsed and the relations are created as a class condition.
Canon	No support	A canon cannot be translated as only one model of knowledge may be defined, but this can be saved as an RDF comment
Situation	No support	No support in OWL for situations as they model a specific moment in time; but this can be saved as an RDF comment.
Synonyms	No support	Could be added as an RDF comment.

Fig. 4. Amine-OWL mappings

4 Binary Relations vs. n-adic Relations

Considering we have a larger more complex CG with many arcs we would break them up into smaller, binary CG and model them as OWL properties one by one. With the nature of the n-adic relations of CG and the Binary relations of OWL it may not seem as simple for these two technologies to interoperate as this paper implies. However using the pathways identified in this paper the complex many arced CG are broken down into binary OWL Properties.

4.1 The n-adic Dimension

OWL properties along with their domain and range can be used to express a dyadic relationship between individuals. However as CG may be n-adic we need to employ Class Conditions in OWL in order lay out the logic as it appears in CG. Using necessary asserted conditions we can map individuals, properties and classes together and translate some of the model of knowledge contained within the Amine ontology.

In order to achieve this we must first programmatically set the Domain and Range of all of the Properties. This can only be done if the property (or in Amine, the Relation) is used somewhere in the ontology. For example if we had in a Amine ontology with the relation 'Agnt' but this relation was never used in a CG, the computer would have no way to know what concepts it exists to relate. For this reason we will assume that all of the Relation types in the Amine ontology we are converting to OWL are referenced by a CG somewhere.

Let us take a CG from the previous example: `[Go] -> (Inst) -> [Bus]` This CG is the Definition of the 'Go' concept in an Amine ontology, from this CG we must get the Domain and Range for the "Inst" Property.

Domain. It is logical that the domain will be a Super-Type of the 'Go' Concept Type, however there could be n-amount of Super-Types for the Concept Type in question, its direct Super-Type could be 'Action', should this be the domain?, the Super-Type of 'Action' could be 'Movement', should this be the domain? The computer has no way of knowing what the correct Domain for a given Concept Type should be with reference to creating an OWL Property, it is for this reason that we have decided to use the direct Super-Type as it cannot be incorrect and will be valid for other uses of the Concept Type. For example now that we have established that the Domain of the property 'Inst' is 'Action' this domain will still be valid when used in a CG like this: `[Return] -> (Inst) -> [Bus]` as the Super-Type of 'Return' is also 'Action'.

Range. The same can be said of the Range of the Property 'Inst' as if we use the Super-Type of 'Bus' which is 'Vehicle', the property would still be valid for a CG like this `[Return] -> (Inst) -> [Car]`. However this will cause an issue if we encounter a CG like this: `[Cut] -> (Inst) -> [Knife]` as 'Knife' is not a Sub-Type of 'Vehicle'. It is for this reason that when the parser encounters a scenario like this the Domain or Range (whichever caused the error) will be 'pushed up' from its existing hierarchical level to a level which will satisfy both of the CG. For example if the above CG is encountered and causes a conflict, the Range will be changed from 'Vehicle' to a concept type (or in OWL Class) which is a Super-Type of 'Car', 'Bus' and 'Knife', in this case it would be 'Entity'.

4.2 Class Restrictions

Now that we have established a method of getting the Domain and Range of the for the OWL properties we can begin asserting conditions on the Classes based on CG from the Amine Ontology.

To continue with our simple example: `[Go] -> (Inst) -> [Bus]` In order to translate this knowledge 'The instrument of Go is Bus' we need to use a restriction on the 'Go' Class in the OWL ontology, this is because this statement is changing the structure of the 'Go' Class. Programmatically this is decided by which ever concept in the CG is logically read first.

When the parser reaches the point of asserting Class Conditions and it encounters a CG it will take the first logical concept, in this case 'Go' and add an OWL Restriction. The onProperty attribute will set the property which restricts the Class, in this case the 'Inst' Property. As the second relation in this CG ('Bus') is a Concept and not an Individual, the construct someValuesFrom is used to allow any Sub-Type of 'Bus' to be accepted as valid.

Figure 5 illustrates.

CG	OWL
[Go] > (Inst) > [Bus]	<owl:Class rdf:about="#Go"> <owl:Restriction> <owl:onProperty rdf:resource="#Inst"/> <owl:someValuesFrom rdf:resource="#Bus"/> </owl:Restriction> </owl:Class>

Fig. 5. Class restrictions

5 Levels of Conceptual Interoperability

We informally assessed the mappings according to the Levels of Conceptual Interoperability Model (LCIM) [10,11]. We were pleased to detect that even within this small-scale project LCIM level 3 (the semantic level) could be achieved, and elements of level 4 (the pragmatic/dynamical level) may be present. In our view Level 5 (the conceptual level, a common view of the world is established i.e. a way to formalize the knowledge about a given domain) would not presently be attained. For this we would turn to the larger-scale CL-SW interoperability projects alluded to towards the beginning of this paper. But considering the ambitious nature of such a small project, "$3\frac{1}{2}$ out of 5 isn't bad".

6 Concluding Remarks

Through this practical exercise we have managed to demonstrate the actual dimensions and extent of useful interoperability. Whilst originally taken in good humour, 'doing it yourself' can nonetheless provide useful results and a context and direction for larger scale projects. The success of our small 'cottage industry' project demonstrates that work in the interoperability arena is not as impossible a task as it may seem. Ours indeed was a small but significant experiment that provided a practical insight into how CG Tools can actually interoperate towards achieving the wider goal of Ontology interoperability between CL and the SW.

References

1. Abdulrub, S., Polovina, S., Sandberg-Petersen, U., et al.: Implementing interoperability through an ontology importer for Amine. In: Croitoru, M., Jäschke, R., Rudolph, S. (eds.) CS-TIW 2008 Proceedings, CEUR-WS, Germany, vol. 352, pp. 1–6 (2008)
2. Coupey, P., Faron, C.: Towards correspondences between Conceptual Graphs and Description Logics. In: Mugnier, M.-L., Chein, M. (eds.) ICCS 1998. LNCS (LNAI), vol. 1453, pp. 165–178. Springer, Heidelberg (1998)
3. Corby, O.: Web, Graphs and Semantics. In: Eklund, P., Haemmerlé, O. (eds.) ICCS 2008. LNCS (LNAI), vol. 5113, pp. 43–61. Springer, Heidelberg (2008)

4. Delugach, H.S.: Towards Conceptual Structures Interoperability Using Common Logic. In: Croitoru, M., Jäschke, R., Rudolph, S. (eds.) CS-TIW 2008 Proceedings, Germany. CEUR-WS, vol. 352, pp. 13–21 (2008)
5. Meijer, E., Beckman, B., Bierman, G.: LINQ: Reconciling Object, Relations and XML in the.NET Framework. In: SIGMOD 2006: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, p. 706. ACM, New York (2006)
6. Polovina, S.: An introduction to conceptual graphs. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 1–15. Springer, Heidelberg (2007)
7. Rodrigues, T., Rosa, P., Cardoso, J.: Moving from Syntactic to Semantic Organizations using JXML2OWL. Computers in Industry 59(8), 808–819 (2008)
8. Rudolph, S., Krötzsch, M., Hitzler, P.: Quo Vadis, CS? - On the (non)-Impact of Conceptual Structures on the Semantic Web. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 464–467. Springer, Heidelberg (2007)
9. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading (1984)
10. Tolk, A., Muguira, J.: The Levels of Conceptual Interoperability Model (LCIM). In: Fall Simulation Interoperability Workshop, Washington DC (April 2004)
11. Tolk, A., Turnitsa, C., Diallo, S.: Implied Ontological Representation within the Levels of Conceptual Interoperability. Intelligent Decision Technologies 2, 3–19 (2008)