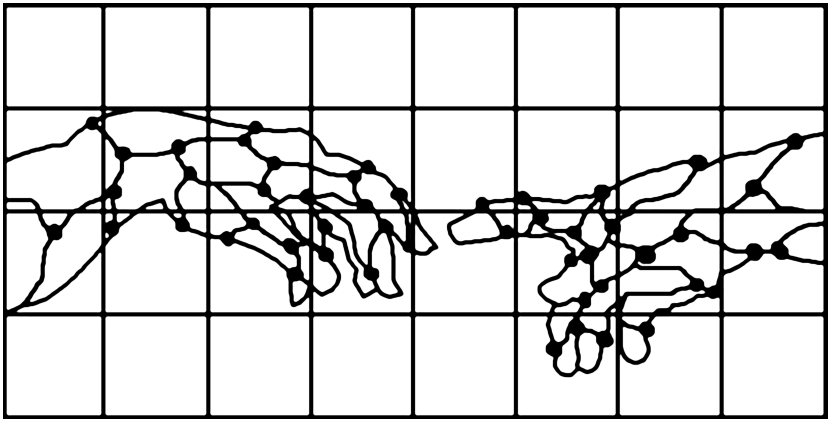
Madalina Croitoru
Robert Jäschke
Sebastian Rudolph (Eds.)

Conceptual Structures Tools and the Web

Third Conceptual Structures Tool Interoperability Workshop 2008



O programmer why seekest thou to hide
The mellow fruits of craftmanship and wit,
Sweets with sweets war not, coupled side by side,
Will software grant the greatest benefit.

Each single tool might let the user down,
Another look and feel, another trait,
To spread CGs' and FCA's renown
There is no choice but: interoperate.

Mark how one tool, good fellow to another,
Connected through a pleasant API
As forged to gladly complement each other,
The user's happiness will multiply,

Their seamless interaction, seeming one,
Conveying this: 'Thou single wilt prove none.'

Preface

The Third Conceptual Structures Tool Interoperability Workshop (CS-TIW 2008) was held in Toulouse, France on the 7th of July 2008, collocated with the 16th International Conference on Conceptual Structures (ICCS 2008) "Knowledge, Visualization and Reasoning". Information about the workshop can be found at: http://www.kde.cs.uni-kassel.de/ws/cs-tiw2008/.

The title of this year's workshop, "Conceptual Structures Tools and the Web" was chosen to emphasize the need for interoperability for building comprehensive, effective knowledge systems that are useful to communities and organizations. While the ongoing goal of the workshop is to explore how to improve the interoperability of Conceptual Structures (CS) tools amongst each other and with other established knowledge representation and reasoning technologies, this year, in particular, the focus has shifted specifically on the interaction of such tools with Web technologies.

The workshop brought together researchers from diverse communities with the common goal of a fruitful discussion on the above mentioned interoperability issues by raising mutual awareness of ongoing research and existing technologies from which each community could benefit.

All papers appearing in this volume were refereed by at three referees. The final decision to accept the papers was arbitrated by the Program Chairs based on the referee reports. We wish in full to express our appreciation to all the authors of submitted papers and to the members of the Program Committee for all their work and valuable comments.

July 2008

Madalina Croitoru
Robert Jäschke
Sebastian Rudolph

Workshop Organization

Programme Chairs

Madalina Croitoru
Robert Jäschke
Sebastian Rudolph

Programme Committee

Oscar Corcho
Raul Garcia-Castro
Peter Haase
Nathalie Hernandez
Marie Laure-Mugnier
Heather D. Pfeiffer
Camille Roth
Gerd Stumme

Local Organization

Nathalie Hernandez

Table of Contents

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.
```

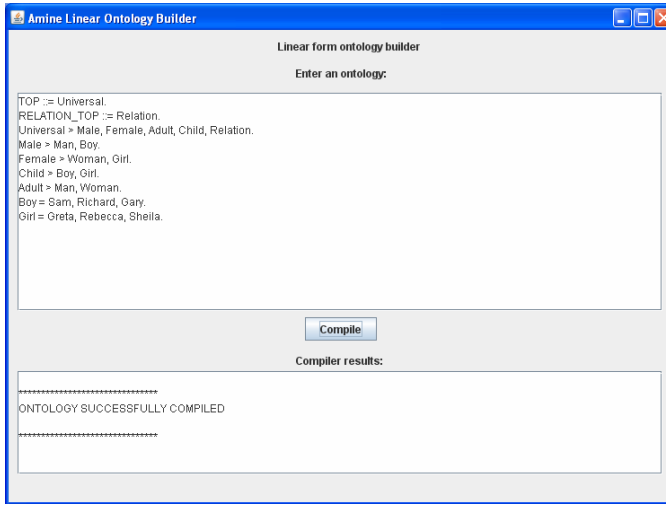


Fig. 1. Importing an ontology.

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.

The Role of FLIPP Explainers as a Tool to Assist the Visual Composition of Web Services for Enterprise Systems

Neil Adams, Simon Polovina, and Richard Hill

Faculty of Arts, Computing, Engineering & Sciences,
Sheffield Hallam University, Sheffield, United Kingdom
neil.g.adams@gmail.com,{s.polovina, r.hill}@shu.ac.uk

Abstract. This work considers process orchestration for software development on Service-Oriented Architectures (SOA). In this research the concept of FLIPP explainers are introduced to the web service community. The FLIPP explainer is a conceptual structure providing mass simplification to otherwise complex logic without the use of text or symbols. In this research the FLIPP explainer has been transformed into a workable tool to develop composite applications alongside SOAs as a direct alternative to some of the current products being offered by SAP and Oracle. Tests indicate that the tool has potential to assist in the development of applications but offers more real promise for the visualization of complex systems and processes. Also, the work highlights the fundamental issues that the FLIPP has for software development, including the initial complexity of designing the diagrams. Finally, guidelines for future enhancements and potential work are provided.

1 FLIPP Explainers and Service Composition

Part of SAP's strategy for improving the way enterprise applications are constructed requires a faster and more flexible development process [8]. With the major manufacturers now introducing Service-Oriented Architectures (SOA), tools are being provided that allow business analysts to create applications by dynamically composing web services in order to quickly produce systems.

It is only now that service-oriented architectures are a viable solution for enterprise computing that visual development has any real chance of succeeding due to the increased abstraction offered by the services as they are now semantically closer to the business than ever before[4]. SAP consider their NetWeaver suite to be the market leader in SOA systems and therefore Visual Composer (VC) to be the pioneering application for business analysts to use. This work aims to improve visual development through the use of FLIPP Explainers as a direct comparison the the SAP alternative.

FLIPP explainers are a logical method to provide massive simplification to complex systems[1]. Cox explains that the diagrams allow complex systems to be visualized "without language, symbols or formulae". He proposes that the

diagrams achieve this by creating 'scenarios' which portray the unambiguous options to the user. Sowa [7] comments that each diagram provides the user with an acrylic and-or graph in rectangular blocks, nested together to form the system. Cox and Polovina [2] imply that by using the diagram frames, it is possible to represent ideas that are not easily conveyed through semantics alone.

The most convenient way to describe FLIPP Explainers is through example and Figure 1 shows a sample FLIPP diagram by Sowa[7]. Cox explains how to read the diagrams as "read down, don't cross verticals". From this example it

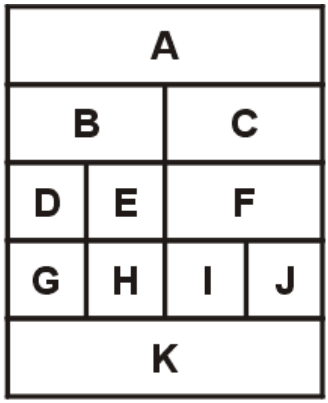


Fig. 1. A sample FLIPP diagram from Sowa[7]

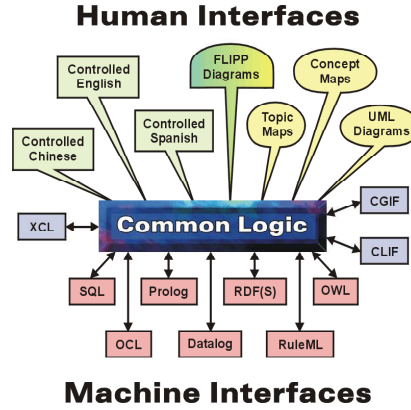


Fig. 2. Relationships of various languages to common logic, Sowa[7]

is possible to analyse the characteristics of the FLIPP diagrams. Sowa[7] notes that the 11 boxes, marked A to K, are grouped vertically, by implicit (AND) symbols, and horizontally, by implicit (OR) symbols. Therefore, according to Sowa, figure 1b can be represented logically through the following equivalent formula:

$$A \wedge ((B \wedge ((D \wedge G) \wedge (E \wedge H))) \wedge (C \wedge F \wedge (I \wedge J))) \wedge K \qquad (1)$$

In English the above diagram can be read as follows: *"Start with A, if you proceed to B as opposed to C then use D followed by G or E then followed by H. If you follow A with C then use F followed by either I or J. Regardless of route, finally finish with K."*

It is by this reasoning that it seems that web services can be modelled through these diagrams in order to improve user understanding and also to provide a logical framework for semi-automated decision making. Although the concept of FLIPP diagrams in relation to user interfaces has not been investigated yet, Sowa[7] has researched them in their original context. Figure 2 shows Sowa's positioning of FLIPP diagrams in relation to common logic and other structures

related to the semantic web. By mapping a FLIPP diagram to formats such as controlled English, processing can occur upon the logic by existing languages. What is attractive about the FLIPP explainer in relation to this work is the impact that it has upon data visualization. FLIPP diagrams immediately convey complex information to people without the need to learn symbols or notation.

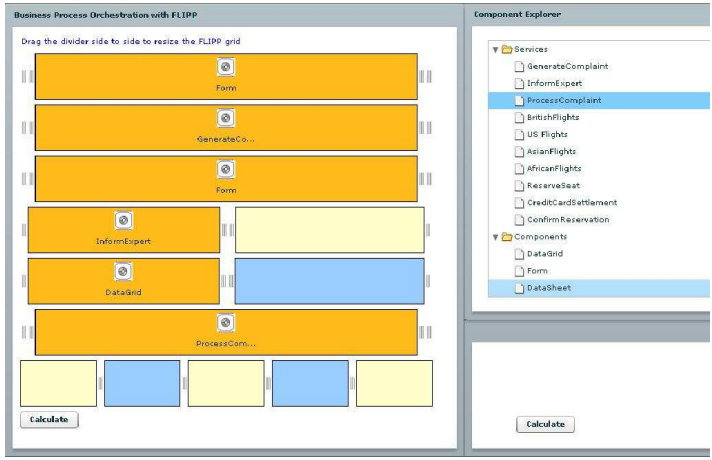


Fig. 3. Sample application modelled inside the FLIPP interface.

2 Discussion

The results of this research have been collated from a number of business and process analysts working at British Airways. In order to ascertain the results, the users were subjected to a range of case studies and asked for opinions regarding the usability and relevance of the interface for their current jobs. These results are outlined in the section below.

When comparing FLIPP explainers with SAP Visual Composer some of the claims made by Cox regarding FLIPP explainer appear to be subjective. Analyst 1 found that lines were easier to decipher than the box system whereas Analyst 2, although understanding the methodology, failed to see a real advantage at this stage. However, Analyst 3 found that reading the processes was much easier on the FLIPP interface than on VC. Therefore it appears that there are two distinct phases of using the FLIPP diagram in process orchestration; software composition, creating a FLIPP, and software modification, reading from a FLIPP.

Cox's[1] main claim, regarding FLIPP's simplicity of reading, is certainly shared by Analysts 2 and 3. Both users felt that it is simple to track processes

through the FLIPP grid yet they both struggled to produce the diagrams initially. Parush et al.'s [5] work note that the use of graphic stimuli in a structured interface help to create a 'holistic' view that users can interpret at a glance. This effect was enhanced significantly amongst inexperienced users which is also evident in the FLIPP testing analysis. Analyst 3 had little knowledge of the are prior to experimentation and afterwards understood the purpose of both the diagrams and the systems perfectly.

However, Analyst 1 had plenty of knowledge of reading diagrams from UML and other experience but failed to see the benefit of the holistic view. However, when it came to developing the interfaces all the users had at least some initial problems. There appear to be two obvious explanations for this trend. Firstly it could be down to the lack of a finalised interface that caused the user frustration with simple problems such as limitations with the colours or deleting items. However it is more likely that it is harder to plan a system in one's head prior to expressing it on a FLIPP interface. The development process of a FLIPP explainer is not covered in detail in Cox's material yet it is a fundamental part of the process. Therefore with this part of the process proving challenging it may deter other users as well.

A common criticism of the interface in the results is the inability for the user to create a series of loops or to use iteration to reach a particular goal. Analyst 3's suggestion of using bolder borders, seems like a particularly interesting idea, especially when compared to the idea of having linking arrows, thus effectively detracting much of the simplification from the diagram. However even this leads to some questions regarding usability as Cox[1] champions the FLIPP diagrams to reduce the complexity of standard arduous textual descriptions. Other systems, such as SAP's Visual Composer, have the luxury of being able to simple connect an arrow back to the start of the loop which is activated when a guard condition is met. Interestingly in the SAP system it is not possible to mark the guard condition on the line thus limiting the visualization that is available.

Analyst 2 raised the point that between cells in the process there is no method for identifying the condition that is met to trigger a certain path through the system. Inside the SAP interface there is generally either a description on a connecting line or the inputs/outputs to be connected are clearly shown. However, inside the FLIPP interface at present this information isn't available which causes uncertainty. Pautusso and Alonso[6] conclude that a visual approach is the natural complement to service selection only when the correct amount of information is provided in the visualization.

In order to read the FLIPP explainers the creator has derived a top to bottom methodology as standard. However it was commented on by Analysts 2 and 3 that processes are traditionally expressed from left to right and Analyst 1 automatically began developing in the left hand corner. Analyst 2, in particular, found that the processes should be expressed this way to promote usability amongst differing sets of people. Many of the results taken from the analysts appear to represent the FLIPP interface in a bad light. However the analysts didn't report that the interface contained any fundamental problems. All the

analysts agreed that the interface was easy to read from and many of the points mentioned above are purely improvements that would be necessary to use the interface for on a full scale implementation.

One major problem that has become apparent with the FLIPP explainer when used in this context is related to the connection of inputs and outputs. Each of the analysts liked the simple method of connection that the interface utilised by combining the inputs via a single screen. However the problems become apparent in the modification of the grid layout. When the grid is moved with the web services attached it can effectively disconnect any connections that have been made when the web services were placed in the grid.

3 Conclusions

The principle contribution is that the FLIPP explainer appears to have a future role in the composition of web services, assuming that some of the initial limitations can be resolved. In response to the original question, of whether it is possible to portray more information through a simpler visualization, the results indicate that it is possible to convey more information through a FLIPP based interface than through the current implementations. This is substantiated as follows:

- Although the results suggest that it is easier to read data from FLIPP diagrams, it is not clear whether it is a tangible effort or time saving approach for the user;
- Being able to compile FLIPP diagrams quickly and accurately appears to be an ability that requires some degree of skill;
- The benefit of process visualization via FLIPP diagrams also depends on the person involved to a certain extent. However it appears to be less important than when creating systems.

It is apparent that the FLIPP explainers original claim by Cox[1] that users prefer complex logic to be portrayed without 'language, symbols or formula' is partially true when used for developmental purposes. Users have a desire to feel empowered by the application and not restricted or frustrated with functional limitations. Therefore the challenge for implementing a successful FLIPP interface is to understand the balance between usability and simplicity. Building a FLIPP interface true to Cox's guidelines [1] would provide a logically superior product which would excel at demonstrating systems to people unaware of the method. However, as noted by Halipern and Tarr[3], too much simplification restricts the purpose of the interface, particularly for experienced users, thus negating any advantages that might have been achieved.

One of the limitations of the FLIPP interface is that it still fails to provide an adequate method in assisting users to select the correct services for their applications. In its current form it serves simply as a tool in the armory of the Visual Composer user, and would best be used as an optional construct dependent upon the developer's preferences.

However, the tool still has a much greater potential when assisting information visualization. Ultimately, this is the strength of the FLIPP explainer, and by implementing a system where the grid is used to assist in modifying previous composite applications, or for identifying useful components from other processes, its potential will be maximised.

4 Future Work

This work represents the first stages of a topic that could potentially lead in many disparate directions. Firstly, the immediate future would be to create an interface with sufficient functionality that it could be used inside an SOA environment to provide genuine 'like for like' tests. This, followed by more detailed analysis of user behaviour in the environment would provide a detailed insight into the real benefits of FLIPP diagrams for a variety of users. Another possible use of the FLIPP diagram is to combine the interface with other Common Logic tools. This would enable users to quickly and easily search for a process that would meet their exact needs whilst communicating the relevant output via the FLIPP interface. Sowa[7] describes a link between FLIPP Explainers and Common Logic, thus suggesting that this approach has potential. Since it is possible to represent the FLIPP data inside common logic notation, it would therefore be feasible to expose the composed service for placement within other conceptual structures. Future work involving FLIPP Explainers is to be focused around improving service composition and visualizing the processes for future modifications.

References

1. Cox, D. (2005) Explanation by Pattern Means Massive Simplification (an E-book), [online] Last accessed 20th June 2007 at http://www.flipp-explainers.org/
2. Cox, D. and Polovina, S (2007) Helping System Users to Be Smarter by Representing Logic in Transaction Frame Diagrams , In Proceedings: of15th International Conference on Conceptual Structures, ICCS 2007, Sheffield, UK, July 22-27, 2007.
3. Hailpern and Tarr (2006) Model-driven development: The good, the bad, and the ugly, In IBM Systems Journal, Vol 45, No 3.
4. Norton, D (2007) SOA is a Catalyst for Model-Driven Development, Gartner Research
5. Parush, A., Hod, A. and Shtub, A. (2006) Impact of visualization type and contextual factors on performance with enterprise resource planning systems, Computers & Industrial Engineering, 52, 133-142.
6. Pautasso, C. and Alonso, G. (2003) Visual Composition of Web Services, In: Proceedings of the 2003 IEEE Symposia on Human Centric Computing Languages and Environments, Auckland, New Zealand, October 2003.
7. Sowa, J. F. (2007) FLIPP Diagrams, [online] Last Accessed 20th November 2007 at: http://www.jfsowa.com/logic/flipp.htm
8. Woods, D & Mattern, T. (2006) Enterprise SOA: Designing IT for business innovation, O'Reilly.

Towards Conceptual Structures Interoperability Using Common Logic

Harry S. Delugach

Computer Science Department
Univ. of Alabama in Huntsville
Huntsville, AL 35899
delugach@cs.uah.edu

Abstract. The Common Logic (CL) ISO standard has been officially published and available. While its focus goes beyond the conceptual structures community, one of its components is an interchange format for conceptual graphs. Now the community has an opportunity to leverage the standard for tool usage. Current tools that support pre-ISO versions must now support the standard. Future tools will be much more useful if they too support the standard. This paper describes the CL effort, outlines its main features and issues a call to action.

Keywords: Common Logic, conceptual graph interchange format, CGIF, international standard

1 Introduction

Over the years, a number of knowledge representations have been proposed, e.g., the Knowledge Interchange Format (KIF) [1], conceptual graphs (CGs) [2, 3], the W3C's Resource Description Framework (RDF) [4], various forms of the W3C's Web Ontology Language (OWL) [5] and there have been occasional efforts to establish ways to translate between them, or else to develop techniques for interoperability between systems that use the various representations. Recently one such effort – Common Logic (CL) – has reached the status of an officially published international standard [6]. This standard is freely available to the public.

While the focus of CL goes beyond just the conceptual structures community, one of its components is an interchange format for conceptual graphs. Now for the first time the conceptual graph research community has an opportunity to leverage the standard for tool interoperability and further promulgation of CGs. Current tools that support pre-ISO versions should be updated to support the standard. This paper describes the effort, its main features and then suggests some of the potential benefits of its use.

2 A (partial) interoperability history of conceptual graphs

The story of conceptual structures interoperability has several chapters. Since conceptual graphs were first proposed [2], efforts have been made to support its interoperability. Indeed, that introduction of conceptual graphs contained a set of "standard" concept and relation types, along with a text-based format known as the "linear form" of conceptual graphs (LF). This linear form had the twin goals of being able to be automatically parsed by software and the ability to be understood by human readers. Consider the conceptual graph example in Figure 1:

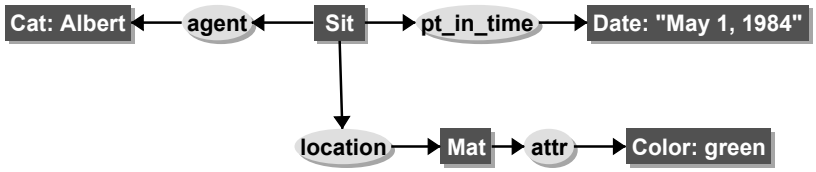


Figure 1. Example Conceptual Graph.

One linear form of this graph is the following:

```
[Cat: Albert]-> (agent) -> [Sit] –
   -> (pt-in-time) -> [Time: "May 1984"]
   -> (location) -> [Mat] -> (attr) -> [Color: green]
```

The linear form has been useful and versatile within online discussions and emails where small graphs need to be shown unambiguously.

The need for interoperable tools was already well understood even before the first ICCS conference. Gerard Ellis and Bob Levinson began an effort known as PEIRCE which was to create a standard workbench for conceptual graphs such that its tools could interoperate with each other [7]. The effort never realized its full potential, for several reasons; one of those was the lack of a standard (other than the LF). John Sowa began to explore options for a more interoperable standard.

A few years after CGs became popular, the Knowledge Sharing Effort (KSE) of the U.S. DARPA research agency was using another knowledge representation language. Their language, known as the Knowledge Interchange Format (KIF) was also a first-order logic language, with extensions for non-monotonic reasoning and definitions [1].

Around 1994, it became clear to researchers in both the CG and KIF communities that they were both addressing issues of interoperability. They realized it was in their best interests to work toward standardizing both CGs and KIF so that (a) they would be interoperable with each other and (b) they would express the same semantics. This effort became known as the Common Logic effort, but it was not successful in reaching the status of an approved standard.

A side effort during these activities was the creation of the CharGer CG editing package [8], along with a published XML version of conceptual graphs which included some meta-information such as graph layout, creation time-date, etc. for a conceptual graph.

One result of that early Common Logic effort was a new textual form of conceptual graphs, known as the Conceptual Graph Interchange Format (CGIF). This version was widely publicized by being available on John Sowa's website for many years (www.jfsowa.com). In fact, many people in the CG community are unaware that this version (which is still online) has been superceded by the ISO standard. Furthermore, many researchers are building tools that still support this superceded standard.

The graph in Figure 1 can be denoted in CGIF as follows:

```
[Cat: Albert ] [Sit: *s ] [Date: *d "May 1, 1984"]
[Mat: *m ] [Color: *c green]
(agent ?s Albert) (pt_in_time ?s ?d) (location ?s ?m )
(attr ?m ?c)
```

The main obstacle preventing the creation of the original Common Logic standard was that participants had different ideas about what features beyond strict first-order logic (modalities, sorts, etc.) ought to be included. In 2003, therefore, a new standardization effort was organized, led by Pat Hayes, Chris Menzel and John Sowa. This new effort was originally called Simplified Common Logic (SCL), which represented both the inauguration of the new effort, as well as a new philosophy of creating a standard with a smaller set of features that could be more easily agreed upon. This effort was aimed toward establishing an international standard, in conjunction with the W3C's desire to create a standard for interoperability of ontology information in conjunction with OWL and RDF.

In June 2003, I was asked to serve as editor for the Common Logic ISO standard. The project was once again called Common Logic (CL, not SCL) in order to match the standard's title. In October 2007, the ISO Common Logic standard became a full-fledged International Standard now designated as ISO/IEC 24707:2007. In April 2008, the standard was designated a publicly available standard, so that one copy can be downloaded from ISO by anyone who wants to use it.

Here is a brief summary of the CG representations discussed in this paper.

Name	Referred to in this paper	Source	Status
Linear Form	Linear Form (LF)	[2]	de facto standard
Sowa's CGIF	Pre-ISO CGIF	http://www.jfsowa.com/cg/cgstandw.htm (still online as of 1 Apr 2008)	Superceded
Conceptual Graph Interchange Format	CGIF	ISO/IEC 24707:2007 [6]	ISO standard freely-available
CharGer-format	CGX	http://projects.sourceforge.net/charger	Freely available

3 Description Of Common Logic

The Common Logic Standard is not just for conceptual graphs; indeed, its main purpose is to provide interoperability for both syntax and semantics for a family of first-order logic languages. The standard's main sections prescribe a set of syntactic and semantic categories: any language dialect that provides both the syntax and semantics is eligible for conformance. The semantics are based on well-understood model theory [9].

3.1 Common Logic Semantics

Common Logic semantics are based on model theory such that any CL text is required to be based on model theoretic interpretations. Simply put, a CL text is associated with a set of individuals, the "universe of discourse," that the text is "about". Because CL also allows functions and relations to be asserted, there is a (potentially larger) set of all things to which the text may refer (including functions and relations); this potentially large set is called the "universe of reference". According to the model theoretic requirements, every symbol in CL is assumed to have some procedure (outside of the model itself) that associates that symbol with specific element(s) in the universe of reference. This association is referred to as the *interpretation* of the symbol.

This means that every model that claims CL conformance must have a model theoretic interpretation over a set of individuals called the *universe of discourse* (*UD*). For CGs, that requirement is addressed by having a set of individual markers in a CG model, each of which denotes a specific individual in *UD*. Every concept has associated with it an

(explicit or implicit) individual marker, denoting the individual. Regardless of whether the marker is explicit or implicit, it is required for a CG model that there exist some consistent means of distinguishing the individual to which a marker refers. For example, the procedure must associate marker **#483** with some individual in *UD*; whenever marker **#483** appears, it must be associated with that same individual.

The impact of the model theory on the use of CG models is straightforward: every concept denotes an individual in some *UD*, so that there must always be a *UD* assumed for any given CG model. The impact on interoperability is more subtle: when a CG model is transferred, its particular *UD* is not explicitly included. Any other system that uses the model may only assume that there exists such a *UD*. This is further explained below.

An important feature for interoperability is CL's ability to incorporate comments in a CL text. These can serve much the same purpose as in any programming or formal specification language – as uninterpreted human-readable text to aid in understanding – but they can also serve as a means to include extra-logical information (i.e., beyond CL semantics). For example, a CGIF text could include any of the following (suitably structured for automated processing, of course) as comments:

- Information about the graphical layout of graph elements (relative page locations, color, fonts, etc.),
- Knowledge about the provenance, origin or custodian of the knowledge being represented
- Executable code for the specification of actors

Though CL comments are necessarily uninterpreted, CL's semantics nevertheless allow comments to be attached to particular phrases in a text. This means they can effectively annotate concepts, relations, or contexts.

3.2 Common Logic Dialects

The Common Logic standard does not prescribe a single syntax, but instead prescribes a common abstract semantics for CL syntaxes. Three separate concrete syntaxes are specified in the CL standard as appendices (each called an "annex" in ISO style).

Common Logic Interchange Format (CLIF)
This dialect is based on KIF and resembles LISP syntax. It is specified in Annex A of [6].

Conceptual Graph Interchange Format (CGIF)
This dialect is based on CGs; its syntax is based on the original pre-ISO CGIF which somewhat resembles the linear form but carefully crafted so that is can be parsed in one pass. It is specified in Annex B of [6].

Extended Common Logic Markup Language (XCL)

This is a new dialect expressed in XML but developed especially for this standard. It is specified in Annex C of [6].

As for all ISO/IEC standards, there is an expected review every five years. This gives a nice time frame during which we can explore the standard and have a better idea of how it might be revised.

3.3 Relationship to other representations

Since Common Logic grew partly out of the W3C community, it fills a technical niche that positions it next to the most popular representations of RDF [4] and OWL [10] [5]. In brief, since RDF expresses only binary relations, CL is more expressive. For two of the three main "species" of OWL, the comparison is straightforward. OWL Lite is primarily for describing classification hierarchies while OWL DL is based on description logics [11] which are a set of languages, each of which is a decidable subset of first-order logic. CL is therefore more expressive than any of these.

The third OWL species, OWL Full, is meant to be more expressive and therefore possess no computational guarantees; i.e., some queries on an OWL knowledge base are undecidable in polynomial time. OWL Full's expressiveness is comparable to CL's, but without the benefit of a century of study in dealing with first-order logic.

One difficulty with the OWL species is that practitioners must decide which of them to use for their purposes. Given a limited domain, it is certainly preferable to use a less expressive (and thereby more computable) representation, if that's all that is needed. It is often the case, however, that a domain's limited purposes may become expanded over time, requiring some rework of the knowledge base into some more expressive form.

Expressivity is obviously important for interoperability: if we transfer some text to another knowledge system that is less expressive, then we either must accept that some of our knowledge will be lost or (worse) that it will be misinterpreted in the less expressive system and lead to incorrect inferences. CL's expressiveness means that we can be sure that the meaning of any RDF, OWL Lite or OWL DL representation can be preserved. Efforts to accommodate OWL Full are ongoing.

4 Interoperability Issues

There are some significant differences between Common Logic and the CG theory as described in [2]. These differences may affect some current semantics of CG tools as they are currently constituted.

Implicit UD

It was mentioned above that when a CG model is transferred, its particular *UD* is not explicitly included. Any other system that uses the model may only assume that there exists such a *UD*. For example, if a CG model is transferred from its origin to a new system, the marker **`#483`** must denote some individual in the originating system's assumed *UD*, but the new system cannot "know" to what individual (e.g., some actual person, location, point-in-time, etc.) the marker refers. If the new system performs logical inferences on the model, e.g., inferring that the individual denoted by marker **`#483`** has the color green, then the results of that inference must be true in the originating system as well. Thus CL's semantics under interoperability are limited to knowing that there is *some* model theoretic interpretation that is valid for the model, without knowing exactly *what* that interpretation is.

Untyped logic

CL does not support types (technically referred to as "sorts" in logic; see [12]), nor any notion of a type hierarchy. In theory, we can treat a type as simply a function on individuals that returns their type; e.g., **`type(Albert)`** would return **`Cat`**. This does not, however, address the issue of how to declare type symbols in the first place, how to associate those type names with particular individuals, how to specify subsumption rules or how to specify reasoning with subsumption. All these issues need to be addressed.

Functions

CL includes the ability to specify functions, which are not specifically supported in CGs. A CL function is a formula whose evaluation results in a denotation of some individual in the *UD* (i.e., it "returns" that individual's identity). It is likely that CG actors can provide this capability, but this still must be clearly demonstrated to the research community, which has been reluctant to adopt actors into most CG tools.

5 Community Research Challenges

Considerable effort has been expended toward the creation of the standard. A dozen or so people have contributed or made substantive technical critique. Some dozen or so meetings in several countries have considered various aspects of the standard. A number of other organizations and initiatives have expressed interest in CL as a useful component for ontologies, knowledge-based systems, automated reasoners, metadata registries, etc.

Explore the standard!

There are some issues with respect to the standard that will need addressing. For example, do CGs provide a model theoretic interpretation of conceptual relations in accordance with the model theory prescribed? Do actors truly provide the capability to represent functions as specified in the standard? How should types and type

hierarchies be specified in a standard way, since they are not covered in CL? These and other questions will need the attention of the research community.

Support the standard!

Many researchers are interested in using CGIF (or already are!) as an interchange format. Obviously the more people who are interested, the more tools and software will emerge that support the standard. A standard is not like a legal statute – it is actually an agreement and a commitment: an agreement that it provides useful features and a commitment to adhere to the standard so that its utility can be realized. While the approval of a standard may seem like the end of a process (especially to those involved in the meetings and discussions!), in fact, it is really a beginning.

Critique the standard!

Users of a standard sometimes see it as a "done deal" – a process that takes place out of sight and whose participants are trying to persuade others to buy into their ideas. But a standard is only as good as its users think it is. Since this is the first version of CL, there are no doubt some issues to be identified and discussed.

Contribute to the standard!

One of the things I have learned is that anyone may contribute to the making of a standard. If you are interested, you can easily communicate with the people involved in the standard's development. Most standards really have only a few people involved in the technical details; if these interest you, then you are encouraged to get involved.

6 Conclusion

The completion of the CL effort is an exciting development for conceptual structures. Now that a completed standard is widely available, the conceptual structures community, especially those interested in conceptual graphs, have an opportunity to build on its potential. One measure of success in a community is whether it supports tools and interoperability. It is therefore imperative that the community rises to the challenge and show all that a standard can help us accomplish.

Acknowledgements. I would to thank the anonymous reviewers for their helpful comments and suggestions. I would also like to thank Pat Hayes, John Sowa and Chris Menzel for their guidance and patience during the lengthy standardization process, in both helping me understand the standard itself and also understanding the power and pitfalls of logic.

References

1. M.R. Genesereth and R.E. Fikes, *Knowledge Interchange Format Version 3.0 Reference Manual*, Technical Report KSL-92-86, Computer Science Department, Stanford University, 1992.
2. J.F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, 1984, p. 481.
3. J.F. Sowa, *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks/Cole, 2000.
4. D. Brickley and R.V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," 2004; http://www.w3.org/TR/2004/REC-rdf-schema-20040210.
5. G. Antoniou and F. van Harmelen, "Web Ontology Language: OWL," *Handbook on Ontologies*, S. Staab and R. Studer, eds., Springer, 2004, pp. 67-92.
6. ISO/IEC, "ISO/IEC 24707:2007 - Information technology - Common Logic (CL) - A framework for a family of logic-based languages," 2007; http://standards.iso.org/ittf/PubliclyAvailableStandards/c039175_ISO_IEC_24707_2007(E).zip.
7. G. Ellis and R. Levinson, "The Birth of PEIRCE: A Conceptual Graphs Workbench," *Conceptual Structures: Theory and Implementation*, H. D. Pfeiffer and T. E. Nagle, eds., Spinger-Verlag, 1992, pp. 219-228.
8. H. Delugach, "CharGer: Some Lessons Learned and New Directions," *Working with Conceptual Structures: Contributions to ICCS 2000*, G. Stumme, ed., Shaker Verlag, 2000, pp. 306-309.
9. W. Hodges, *A Shorter Model Theory*, Cambridge University Press, 1997.
10. M.K. Smith, et al., "OWL Web Ontology Language Guide," 2004; http://www.w3.org/TR/owl-guide/.
11. F. Baader, et al., "Description Logics," *Handbook on Ontologies*, S. Staab and R. Studer, eds., Springer, 2004, pp. 3-28.
12. M. Davis, "First Order Logic," *Handbook of Logic in Artificial Intelligence and Logic Programming*1, D. M. Gabbay, et al., eds., Oxford Univ. Press, 1993, pp. 31-65.

Pseudo-conceptual text and web Structuring

Ali Jaoua

Computer Science and Engineering Department
College of Engineering, Qatar University
jaoua@qu.edu.qa

Abstract: Structuring huge documents with a high speed is still a challenge. In this paper, we propose a heuristic based on pseudo-concepts to derive a tree of words reflecting in decreasing "importance" order the semantic macro-structure of the space of documents or the micro-structure of a document. Both macro and micro structures are used to browse inside the space of documents. The advantage of the proposed methods with respect to previous ones using exact formal concepts [2,4,11], is that by only selecting approximate formal concepts associated to the different pairs of a binary relation linking documents or sentences inside a document to indexing words, we improve the structuring process in terms of time complexity while keeping acceptable meaning of generated text structure. Experimentation [12] realized with documents with big size showed that response time of the structuring system as well as the browsing trees are very helpful for users to get the global structured view of the space of documents and the detailed view inside a selected document. Starting from an already created conceptual meta-search engine merging Google and Yahoo search results [4,11], we now have a way to compile more web pages in much shorter time.

Keywords: Macro and micro document structuring, pseudo formal concepts, approximate associations, pseudo Galois connection

1 Introduction

While browsing through a documentary database, Internet or simply in a text, the most important need for the user is to find pertinent information in the shortest possible time, or the main structure of significant keywords related to the content. Generally, extracting pertinent information from data requires mainly the two following tasks: first read and classify data, second select the most suitable information related to the user interest. Most of previous systems using conceptual analysis are only able to analyze a small number of documents or web pages [2,4], because most of classification methods are NP-complete and are not able to compile a high number of documents in an acceptable time for the users, at real time. Computers and communication systems are mainly used to search and retrieve URLs with very

high speed from allover the world, creating obviously the need for developing a layer of information engineering software (i.e. “intelligent software”) which main task is to read and organize data for the user, at real and acceptable time. These intelligent systems have the precious task to classify dynamically and incrementally new arriving URLs or data. They are dedicated to make repetitive classification activities, preparing the work to the human browser, and presenting it with a more understandable and structured view. During the last three years, two text structuring systems for English and Arabic languages have been implemented [9,10], and a meta-search engine for English "Insighter" [4,11]. These systems are based on the following steps: first, creation of a context from the text by decomposing the text into different sentences, and the sentence into non "empty" words, where two similar words are assimilated to only one representative word; second the coverage of the context by a minimal number of concepts[1] with the greatest density; third associating to each concept a significant title (i.e a word with a maximum weight selected from the domain of the concept), finally organizing the words into a heap (i.e an almost complete binary tree where words with greater weight appear at the higher level in the tree). Because of the nature of conceptual clustering (NP-complete problem), even if we used a branch and bound algorithm, the system was only able to process efficiently texts with small size. However the quality of the derived tree of words is very good and reflects in most of the tested texts their main ideas. In this paper, we propose an approximate approach for documentary database or a text structuring that should only require a linear time in terms of the size of the binary context C linking documents to indexing words or sentences inside a same document to words indexing these sentences. The proposed method is based on a heap data structure ordering pairs (d,w) of binary relation C in decreasing strength order, where d is a reference to a document and w is an indexing word.

The next section includes some relational algebra, and formal concept analysis, the mathematical foundations used in this work. We also give a definition of the strength of a pair (d,w) in the next section. As a matter of fact, we often merge the two backgrounds through the context that we assimilate to a binary relation.

In the third section, we present an approximate algorithm to build a heap of words through which user can browse easily to find the most pertinent documents. In section 4, we present some experimental results of the heuristics for text structuring [12] on a developed system. We also anticipate its utilization for improving our conceptual meta-search engine last[11].

2. Background and Foundation

2. 1 Relational Algebra [8]

A binary relation R between two finite sets D and T is a subset of the Cartesian product $D \times T$. An element in R is denoted by (x,y), where x designates the antecedent and y the image of x by R. For a binary relation we associate the subsets given as

follows: The set of images of e defined by: e.R= $\{e' \mid (e, e') \in R\}$. The set of antecedents of e´ is defined by:

$$e'.R^{-1} = R.e' = \{e \mid (e, e') \in R\};$$

The domain of R is defined by: Dom(R) = $\{e \mid (e, e') \in R\}$.; The range of R is defined by: Cod(R) = $\{e' \mid (e, e') \in R\}$; The cardinally of R defined by: Card(R) is the number of pairs in R. Let R and R' be two binary relations, we define the relative product (or setting up) of R and R', the relation R o R' = $\{(e, e') \mid$ It exists t in cod(R)$\mid (e, t) \in R$ & $(t, e') \in R'\}$, where the symbol "o" represents the relative product operator. The inverse relation of R is given by: $R^{-1} = \{(e, e') \mid (e', e) \in R\}$. The relation I, identity of a set A is given by: $I(A) = \{(e, e) \mid e \in A\}$.

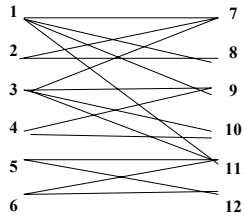
Definition 1*: Gain or economy of relation R*

The gain W (R) of binary relation R is given by:

$$W(R) = (r/(d.c))\ (r-(d+c))$$

Where, r is the cardinality of R (i.e. the number of pairs in binary relation R), d is the cardinality of the domain of R, and c is the cardinality of the range of R.

Example1: If R1 is the following binary relation:



r=16 (i.e. number of pairs in R1)
d=6 (i.e. cardinality of the domain of R1)
c=6 (i.e. cardinality of the range of R1)
W(R1)= (16/(36))(16-12)= 1.77

Remark: The quantity (r/dc) provides a measure of the density of relation R. The quantity (r-(d+c)) is a measure of the economy of information.

2.2. Previous Developed Structuring Algorithm

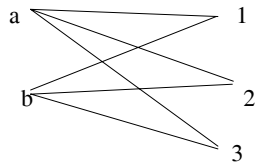
In last developed tools already running with acceptable quality[11,13], implemented algorithm was based on "optimal concept" clustering, using a branch and bound heuristic, where at the first step we calculate "the elementary relation ER(x,y) " associated to each pair (x,y) of relation R. ER(x,y) is given by the following relational expression:

$$ER(x,y) = I(y.R^{-1}) \text{ o } R \text{ o } I(x.R)$$

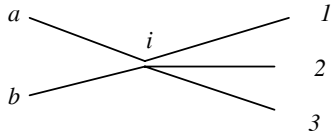
At the second step we calculate the weight W(ER(x,y)) of each elementary relation ER(x,y). Third, we select the pair (xmax, ymax) such that W(ER(xmax,ymax)) is the maximum weight and we continue to give a priority to explore ER(xmax,ymax) to find the concept with maximum weight in R. We continue by the same way to select other concepts until covering R. We then give a name to each concept, by selected the word with the maximum rank in the range of the concept. We finally build a tree of words, where each word is linked to the associated cluster of URLs.

In the following section, in order to accelerate the tree of words generation, we will extrapolate the definition of W(ER(d,w)) to only calculate W(ER(d,w)) = $W(w.R^{-1} \text{ x } d.R)$.

Example2: In the case of the following relation R2 corresponding to a complete bipartite graph: W(R2)=(6/6)(6-5)=1.



W(R2)=1, because we may replace the 6 pairs by only 5 pairs by the creation of an intermediate object (i), saving one link:



If we assume that a document is composed of several sentences, and that ideas are associated to sentences in the text, then pairs (document d, word w), where w belongs to d plays a major role for discovering the main ideas contained in the text. We may sort all the possible pairs (d,w) in decreasing strength order, or only creating a heap of pairs (d,w), that we can update in an incremental way, in a logarithmic time in terms of the total number of pairs in the binary context relating each sentence to all indexing words.

In the following definition, we use function W to define the strength s(d,w) of a pair (d,w) in relation C (or context) as equal to $W(w.C^{-1} \text{ x } d.C)$, as a approximation of the weight of the corresponding elementary relation seen in section 2.2 (i.e. $\mathbf{W}(ER(x,y)) = \mathbf{W}(I(w.C^{-1}) \text{ o } C \text{ o } I(d.C))$).

Definition 2: **Strength of a pair (d,w)**

If w is indexing a document d then w is weakly associated to all words of w contained in document d, with strength:

$$s(d,w) = ((|d.C| \times |w.C^{-1}|) - (|d.C| + |w.C^{-1}|)).$$

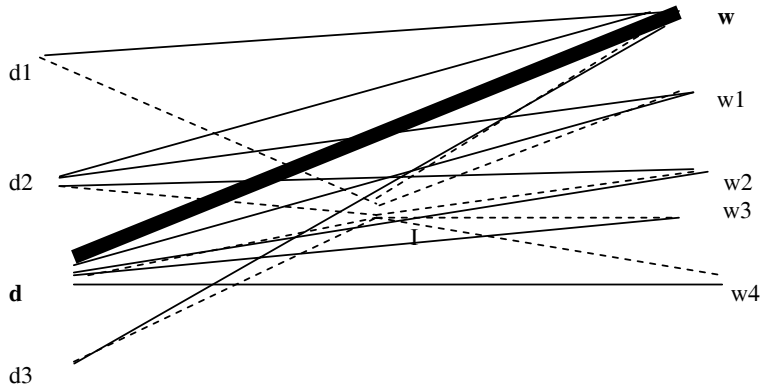


Fig. 1 Strength of a pair (d,w)

In fig1, s (d,w) = (5 x 4 – (5+4))= 11. In this example, discontinued pairs linking $w.C^{-1}$ to d.C through a new created intermediate object (I): i.e. the pseudo-concept representative replaces continued pairs of the initial elementary relation included in C and supporting pair (d,w).

2.2. Formal Concept Analysis [5]

Formal Concept Analysis (FCA) is a theory of data analysis which identifies conceptual structures among data sets. It was introduced by Rudolf Wille [1,5] and has since then grown rapidly.

2.2.1 Usual Definition of the two operators of Galois Connection

Let G be a set of objects and M be a set of properties. Let C be a binary relation defined on the set E. For two sets A and B such that $A \subseteq E$ and $B \subseteq E$, we defined two operators $f(A)= A^R$ and $h(B)= B^Q$ as follow:

$$f(A) = A^R = \{m | \forall g \in A \Rightarrow (g, m) \in C\}$$
$$h(B) = B^Q = \{g | \forall m \in B \Rightarrow (g, m) \in C\}$$

A formal context k :=(G,M,C) consists of two sets G (objects) and M (Attributes) and a relation C between G and M. Formal Concept of the context (G,M,C) is a pair (A,B) with: $A \subseteq G$, $B \subseteq M$, $A^R = B$ and $B^Q = A$. We call A the extent and B the intent of the concept (A, B). IF (A_1, B_1) and (A_2, B_2) are two concepts of a context, (A_1, B_1)

is called a sub concept of (A_2, B_2), provided that $A_1 \subseteq A_2$ and $B_2 \subseteq B_1$. In this case, (A_2, B_2) is a super concept (A_1, B_1) and it is written $(A_1, B_1) < (A_2, B_2)$. The relation **"<"** is called the hierarchical order of the concepts .The set of all concepts of (G, M, C) ordered in this way is called the concept lattice of the Context (G, M, C).

2.2.2 Definition 3: Pseudo-concept associated to a pair (d,w)

Let C be a binary relation linking documents to words, then an elementary relation associated to a pair (d,w) is defined by ER= $I(w.C^{-1})$ o C o I(d.C). Where I(A) is the identity relation restricted to set A. and o is the operator for relational composition. A pseudo-concept is an approximation of an elementary relation by the concept: PC= $w.C^{-1}$ x d.C, (i.e. the smallest concept including RE: fig1). In order to avoid to compute ER and calculate its economy by function W given in definition 1, we set W(PC)=strength (d,w) = s(d,w) as the economy of PC.

In the following section, we define the two following operators f' and h' instead of the classical ones f and h reminded in section 2.2.1.

2.2.3 Definition of two new operators f' and h'

Galois connection operators (f,h) defined in the previous subsection 2.2.1 are of course suitable to build the lattice of concepts, and to find associations between the attributes, useful during the browsing process. In our case, we define a "bridging pair $(g,x) \in C$ with the strongest strength associating a set A to a another set B, such that g.C = B and $A \subseteq x.C^{-1}$. The two following operators (f',h') are designed for the purpose of generating such pair (g,x):

$f'(A) = A^{R'} = \{m \in g.C \mid$ *it exists a pair (g, x)* $\in C$, $A \subseteq x.C^{-1}$, $(|g.C|. |x.C^{-1}| - (|g.C| + |x.C^{-1}|$ *) is maximal}* where |A| is the cardinality of set A., g.C is the set of images of g in the binary relation C, and $x.C^{-1}$ is the set of antecedents of x by C the binary relation associating a set of documents to a set of terms (or words). f(A) is defined as the range of the most economical pseudo-concept containing A. Starting with a set A of words, we may find some additional information as an association rule $A \rightarrow x.C^{-1}$ with the weight s(g,x).

$h'(B) = B^{Q'} = \{y \in x.C^{-1} \mid$ *it exists a pair (g, x)* $\in C$, $B \subseteq g.C$, $(|g.C|. |x.C^{-1}| - (|g.C| + |x.C^{-1}|$ *) is maximal}*. As additional information, we can say that $B \rightarrow$ g.C with the weight s(g,x). The set of documents g.C is associated to the set B with some strength through the pair (g,x). Operators f' may be calculated with a quadratic time, n x m, where n is the cardinality of A and m is the cardinality of g.C , where g is an element in A. For example to find f'(A), we select the set of images A' of some element g of A, then for each element x of A' we check if $(A \subseteq x.C^{-1})$ and calculate the weight ($|g.C|. |x.C^{-1}| - (|g.C| + |x.C^{-1}|)$. We finally select the maximum weight as the strength of the association $A \rightarrow w.C^{-1}$. The proposed definitions of (f',h') are useful because they have a similar advantage to the classical operators (f,h), even less precise, they define an interesting definition of the strength of an association we could use for fuzzy reasoning. We expect to implement function h' to display some associations

relating the keywords of analyzed documents. The proposed new operator will be used to give additional information to the user by adding new words to his/her request.

In the following section 3, we present globally an efficient algorithm based on ordering of the pairs using the heap data structure where the pair with highest strength is stored in the root of the heap.

3. An Efficient Algorithm for building a browsing tree

The idea of the algorithm is that pairs (d,w) in the binary relation C with highest strength should very probably be central because they are making the bridge between a set of documents and a set of words, such that pair (d,w) may at most enable us to save:

$$(((|d.C| \times |w.C^{-1}|) - (|d.C| + |w.C^{-1}|)) \text{ links or pairs}$$

because it replaces the sub-relation of C pre-restricted by the set w. C^{-1} and post-restricted by the set d.C (i.e. $I(w.C^{-1}) \circ R \circ I(d.C)$) by the Cartesian product: $w.C^{-1} \times d.C$, when this sub-relation is a complete bipartite graph.

The proposed algorithm is used twice, at the first step it generates a tree for building a high level structure of documents through a heap of words (macro-structuring). At the a second level, it generates a micro-structure of any selected document. The different steps of the algorithm are the following:

3.1 Macro-structuring and browsing algorithm

1- Create a weighted matrix S corresponding to R, where if each pair (i,j) we save its strength in S(i,j). This step is O(n) in terms of the number of pairs in R. For that we can first calculate the number of elements in each row and column in R.
2- Build a heap of pairs (i,j) containing the information about their strength. This step is also known to be O(n).
3- Cleaning the heap by only visualizing the words (w) without repetitions, avoiding to show the associated document (d) in the pair (d,w).
4- If a user decides a word for browsing then all the list of documents indexed by this word will be proposed.

3.2 Micro-structuring and browsing algorithm

5- If a user decides to select a specific document d, then a tree of words will be built by the same way as in the macro structuring step, but the difference is that the sentence inside the document is used instead of the document itself.
6- The user may now browse inside the document using selected keywords to be able to get the most suitable view corresponding to his needs, using the new

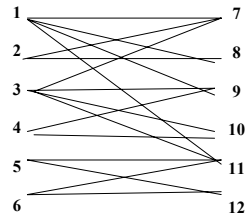
operators (f',h'), we can even calculate some approximate closure of the request to expand user query.

3.3. Incremental heap reorganization and structuring algorithm

7. If a new document is introduced in the system, then a new row is created in relation R, updating of S will require a maximum of O(n) iterations. While updating the heap will require a maximum of O(n) operations.
8. If we change a text, by adding, removing or updating a sentence in the text, then we will need to update R, S and the heap H in a linear time.
9.

3.4 Illustration for a structuring in general (Macro or micro):

Here we may find a binary tree R linking documents (1,2,3,4,5,6) to words (7,8,9,10,11,12):

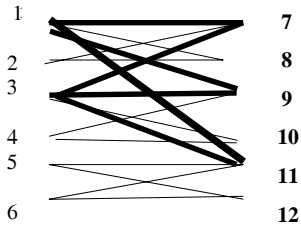


After sorting of the different pairs of R in increasing order in terms of their calculated strength, and removing word redundancy, we obtain the following heap of words:

(11), (9), (7), (8), (10), (12).

Which means that the most pertinent word is 11, then 9, 7, 8, 10 and 12, in decreasing importance order. When the user clicks on 11, then references to documents $11.C^{-1}$ ={1,3,5,6} are shown.

Remark: What might be considered as a good result is that the 6 first selected pairs among 16 pairs of R, belong to the most economical concept in relation R, totally, as you can see it in the figure below. This is a meaningful result, because greatest concept represents the most significant cluster of documents sharing the maximum number of terms.



In bold you may notice that a concept is built from the 6 first pairs with the highest strength. Therefore, the presented heuristic is promising as an alternative to find optimal concepts coverage (NP-complete problem) in a binary relation with an acceptable time.

4. Improvements of existing structuring browsing

The proposed heuristics enabled us to generate the structure of texts with big size, for English and Arabic texts, in much better time complexity compared to previous methods using either the lattice of concepts or a heap of concepts with respect to function gain seen in definition1. Credo [2] is very slow because it needs the calculus of the global latticed of concepts. In [11,4], we developed "Insighter" a running meta-search engine that only classifies documents into a minimal coverage of the context by the most economical concepts. But used heuristics were still not efficient, even if the quality is acceptable. In the current approach presented in this paper, implemented and tested with a good number of texts, in most of case the first words represent the main keywords of the text and reflect really the main concepts discussed in the text, with an excellent speed. However, the classification of documents behind each representative keywords should be implemented by use a similarity distance and threshold definition. The proposed method will be incorporated in a conceptual meta-search engine already realized by replacing concepts with meta-concepts. For example, if we submit as input the abstract and introduction of this paper to the structured browser, it gives as first indexing words: "document, space, structuring, micro, macro, data, text, selection, browsing, searching, relational, ….which are surely among the most significant words in the paper, as you can see it in the presented screen below. In another document about mind and brain, obtain words are: brain, synapses,.. The new realized system is now suitable for structuring electronic books in an acceptable time. The speed of the new developed tool is much better than the previous one, replacing minutes with seconds, in all tested cases and allowing the structuring of huge documents [12]. The quality of the first words is almost always the same as the previous one. However, the new system does not classify as accurately as the first developed one. So, we are already trying to improve the quality of the cluster of documents related to some representative word, used as a reference to group only documents enough close to each other. In the following two figures 2 and

3, we can find respectively a first screed related to the same text related to brain and mind, both systems selected and recognized what was exactly the title of the document. However the trees of words at lower levels become very different. While our initial system takes 30 seconds, the new one takes less than a second to do a similar task.

5. Conclusion and perspectives

This system employs some pseudo-formal concept analysis as an approach for knowledge discovery and clustering. A heuristic process of finding coverage of the domain of knowledge using the idea of concepts [1] is here replaced by pseudo-concept ordering. Our approach is experimented for text structuring, and it will be used to update a conceptual meta-search engine to enable it to classify more search results. The presented methods may also be used as a base to improve proposed heuristics for solving the NP-complete problem of binary relation coverage with a minimal number of concepts. We should also now explore the incremental version of these algorithms. The new methods will automatically improve the efficiency of our developed structuring conceptual meta-search engine [11,4], however we are trying to improve the quality of the browsing trees to have a better clusters of URLs.

Acknowledgements: We thank Qatar University for having granted this research (# 05013CS).

REFERENCES

1. Bernhard Ganter and Rudolf Wille , (1999). Formal concept analysis: mathematic foundations. Springer, Berlin/Heidelberg, 1999.
2. Carpineto, C., & Romano, G. (2004b). Exploiting the Potential of Concept Lattices for Information Retrieval with CREDO. Journal of Universal Computing, 10, 8, 985-1013 .
3. Godin, R., Gecsei, J., & Pichet, C. (1989). Design of browsing interface for information retrieval. In N. J. Belkin, & C. J. van Rijsbergen (Eds.), Proc . SIGIR '89, 32-39
4. A. Jaoua, M. Lazhar Saidi, Ahmed Hasnah, jihad M. Al-Jaam, Sahar Ahmed, Baina Salem, Noura Rashid Shereen Shareef, Suad Zaghlan. Structured Conceptual Meta-Search Engine, Fourth International Conference on Concept Lattices and Their Applications (CLA2006), Hammamet-Tunisia 2006 , 30/10 au 1/11/2006.
5. Wille, R. (1982). Restructuring lattice theory: an approach based on hierarchies of concepts. In I. Rival (Ed.), Ordered sets. Reidel, Dordrecht-Boston .470-445 ,
6. Ahmed Hasnah, Ali Jaoua, Jihad Jaam. , Conceptual Data Classification: Application for Knowledge Extraction. Chapter 23: Computer Aided Intelligent Recognition Techniques and Applications, Editor Muhammad Sarfraz, Wiley, 2005.

7. Jaoua A. and Elloumi S., Galois Connection, Formal Concepts and Galois Lattice in Real Relations: Application in a Real Classifier, The Journal of Systems and Software, 60, pp. 149-163,2002.
8. Schmidt, and Strohlein, Relation and Graphs, Discrete Mathematics for Computer Scientists. EATCS-Monographs on Theoretical Computer Science. Springer, 1993.
9. T. Mosaid, F. Hassan, H. Saleh, F, Abdullah, Conceptual Text Mining: Application for Text Summarization, Senior Project, University of Qatar, January 2004.
10. A. Jaoua, M. A. Al-Saidi, A. Othman, F. Abdulla, I. Mohsen, Arabic Text Structuring by Optimal Concept Extraction and its Utilization for Browsing, The fourth International Conference on the Use of Arabic Language in Computer Science, CSPA, Doha, 31 March, 2008. pp 74-82.
11. A.Jaoua, Conceptual Structured Browsing: applications for structuring meta-search engine, summarization and text processing. ICFCA, International Conference on Formal Concept Analysis, Clermont Ferrand, France, February 2007.
12. Aisha A. Al Ibrahim, Mariam A. Al Abdulla, Fatma A. Al Rasheed, Mashael R. Al-Mansouri, Automatic Structuring of Electronic Book, Senior Project, Qatar Univesity, 2008.

FcaStone - FCA file format conversion and interoperability software

Uta Priss

Napier University, School of Computing,
u.priss@napier.ac.uk
www.upriss.org.uk

Abstract. This paper presents FcaStone - a software for FCA file format conversion and interoperability. The paper both describes the features of FcaStone and discusses FCA interoperability issues (such as the specific difficulties encountered when connecting FCA tools to other tools). The paper ends with a call to the FCA community to develop some sort of standard FCA Interchange Format, which could be used for data exchange between stand-alone applications and across the web.

1 Introduction

FcaStone[1] (named in analogy to "Rosetta Stone") is a command-line utility that converts between the file formats of commonly-used FCA[2] tools (such as ToscanaJ, ConExp, Galicia, Colibri[3]) and between FCA formats and other graph and vector graphics formats. The main purpose of FcaStone is to improve the interoperability between FCA, graph editing and vector graphics software. Because it is a command-line tool, FcaStone can easily be incorporated into server-side web applications, which generate concept lattices on demand. FcaStone is open source software and available for download from Sourceforge. FcaStone is written in an interpreted language (Perl) and thus platform-independent. Installation on Linux and Apple OS X consists of copying the file to a suitable location. Installation on Windows has also been tested and is also fairly easy.

The need for interoperability software was highlighted in discussions among ICCS participants in recent years. Several ICCS authors expressed disappointment with the lack of progress in conceptual structures (CS) research with respect to applications and software (Chein & Genest (2000); Keeler & Pfeiffer (2006)) and with respect to current web developments (Rudolph et al., 2007). Although several sophisticated CS tools exist, each of them has different features. If a user wants to use features that are supported by different tools, it can be difficult to move the data from one tool to the other because each tool has different file formats. APIs are missing that would allow for the different existing tools to interoperate. Interoperability has been discussed by

[1] http://fcastone.sourceforge.net/

[2] FCA stands for Formal Concept Analysis. This paper provides no background on FCA. See http://www.fcahome.org.uk for links to FCA references, introductions and software.

[3] The URLs for all tools mentioned in this paper are listed at the end of the paper.

Dobrev (2006) and, implicitly in Tilley's overview of existing FCA tools and has been the topic of ICCS tools workshops. Recently, the Griwes project[4] has been developing a framework for conceptual graph (CG) interoperability. Griwes is a very promising project, but it focuses more on CGs than FCA and is still in its early stages.

FcaStone provides a first step towards interoperability: it allows to convert between different FCA file formats. Ideally, this conversion should not have to be performed at the command-line, but instead should be integrated into existing tools. But until such APIs exist that allow full interoperability, FcaStone provides a simple workaround. Furthermore, it is quite unlikely that APIs will ever be written for all possibly relevant tools, especially if this includes other non-CS tools. Thus there may always be a need for a file format conversion tool. It should be stressed that so far FcaStone only supports FCA formats, not CG formats. It is planned in the future to extend FcaStone also to CG formats, but it has to be investigated, first, in how far that is feasible, because FcaStone focuses mainly on lattice representations which are not the main concern of CGs.

While developing FcaStone, it became apparent that there are a few non-FCA file formats, which are also suitable for representing concept lattices. In particular, graph formats and vector graphics formats are of relevance. The difference between graph and vector graphics formats is that vector graphics are more general. Graph formats usually focus on graphs consisting of nodes and edges. Graph editors normally provide graph layout algorithms. The connection between a node and its edges is usually fixed, so that clicking on a node and moving it around will move the connected edges with that node. Vector graphics formats, on the other hand, can be used for any sort of graphics (not just nodes and edges). Although vector graphics editors usually have some grouping mechanism that allows to create complex objects which can be moved around and edited as a whole, it is not always possible to connect edges to nodes in such a manner. While vector graphics formats can represent graphs and provide many editing features, they often do not provide the specific editing features that more specialised graph editors have. Both graph and vector graphics formats are of interest to FCA, but because of the differences between them, not all FCA features can be represented in these formats.

The following list summarises the formats and features that are currently implemented for FcaStone (the URLs for the tools are at the end of this paper):

- Commonly used FCA file formats (cxt, cex, csc, slf, bin.xml, and csx).
- Conversion between FCA file formats and comma separated value (csv) files as exported from and imported into databases and spreadsheets.
- Export into Bernhard Ganter's latex format (only for contexts at the moment).
- Graph formats (dot, gxl, gml, ...) for use by graph editors (yEd, jgraph, ...) and vector graphics formats (fig, svg, ...) for use by vector graphics editors (Xfig, Dia, Inkscape, ...).
- Creating lattice diagrams (using Graphviz's layout) from contexts.
- Serve as a component of a server-side script for generating lattices on webpages.

The emphasis of FcaStone is on converting file formats, not on fast algorithms for lattice construction which are already provided by other FCA software. FcaStone can convert formal contexts into lattices. It uses the Ganter algorithm (Ganter, 1984), but in

[4] `http://www-sop.inria.fr/acacia/project/griwes/`

a very simple string implementation that is not efficient. FcaStone then uses Graphviz to calculate the graph layouts. Graphviz is open source graph visualisation software, which contains several graph layout algorithms. In this respect, FcaStone is similar to the Colibri software, which also relies on Graphviz for lattice layouts. Because Graphviz provides a large number of file conversion options, FcaStone only needs to produce a single format (called "dot format") which can then be further converted by Graphviz into a large number of other formats.

It is possible with FcaStone to convert a formal context into a concept lattice (for example as a gif file) by just running FcaStone on the command-line. In many cases the resulting pictures are surprisingly good without manual editing of the diagrams - although the diagrams do not exactly follow all of the traditional FCA style conventions. FcaStone is a very slim program. Its interaction with Graphviz and with other scripts (if used on a web-server) is designed to be achieved via system calls (pipes). Although this is a fairly basic form of interaction which may not be very efficient, it should be easy for a programmer to incorporate FcaStone into other scripts or to modify it to suit other needs. FcaStone is work in progress. The following features are not yet available, but are planned for future releases:

- Converting lattices between different formats in a manner that preserves the graph layout of the lattice. (This would also mean that other FCA software such as Galicia or Colibri could be called from FcaStone in order to obtain layouts and efficiency.)
- Database connectivity.
- Convert into other knowledge representation formats (conceptual structures, semantic web), if they are suitable for representing lattices.

The remainder of this paper provides an overview of the file formats that are supported by FcaStone (Section 2); discusses the specific difficulties related to presenting concept lattices in graph formats (Section 3); presents more details on which tools are suitable for editing FCA data and how these are supported by FcaStone (Section 4); and, finally, discusses FCA web applications (Section 5), which includes a call to the FCA community to develop some sort of standard FCA Interchange Format, which could be used for data exchange between stand-alone applications and across the web.

2 Supported formats

Fig. 1 provides an overview of the supported file formats. The top half of the figure shows formats that are supported for input and output; the bottom half show formats that are only supported for output. The top half of the output-only formats could also be supported for input in future versions of the software, but only if certain constraints are observed by the formats. This is because these formats are not FCA-specific and allow for the representation of other graphs or vector graphics. The raster graphics and page description formats are totally unsuitable for input, because reading those files would require image recognition techniques.

The FCA formats should be the easiest to handle. Unfortunately, there are many different ways to encode a formal context or concept lattice. For example, a context can be stored as a cross table or as a two column table; a concept lattice can be stored

extension	I/O	type	scope	Graphviz required?	comments
cxt	input/output	FCA format	only context	no	P. Burmeister's format
con					Colibri format
slf					Galicia format
bin.xml					Galicia format
tuples		tab separated values			Tupleware format (like csv, but tab instead of comma + additional first line) only two column files supported
csv		comma separated values			used by databases/spreadsheets
csc		FCA format	context + lattice		F. Vogt's Anaconda format (lattice not implemented)
cex				no	ConExp, lattice not yet implemented
csx				no	ToscanaJ, lattice not yet implemented
fig	output only	vector graphics		yes for lattice	xfig
tex		latex			to be used with B. Ganter's fca.sty, lattice not yet implemented
dot		graph format	only lattice	no	Graphviz format
gml				no	
gxl				yes	format availability depends on local Graphviz installation
svg		vector graphics			
jpg		raster graphics			
gif					
png					
ps		page description format			
pdf					

Fig. 1. Supported formats

as nodes and edges at a concrete level with coordinates or at an abstract level without coordinates or purely as a graph or as partial information that can only be read in combination with the context. Thus even though all of the more modern FCA formats are XML formats, translating between formats is not just a simple XML transaction because different information is stored. For example, if one XML format stores only the context and a second format only the lattice, then the complete lattice needs to be calculated for the conversion from the first to the second format.

As mentioned before, Graphviz is used to produce the graph layouts in FcaStone. Because Graphviz has the ability to convert its "dot" format into a variety of formats (the ones on the bottom third of Fig. 1), the conversion into these formats is automatically provided without FcaStone having to do any work. The disadvantage is that FcaStone does not have control over these file types. Graphviz produces slightly different results in different settings. For example, one user reported problems with the xfig output of FcaStone which have not been encountered in other installations of the software. If users do not want to install Graphviz, an alternative is provided by the gml format which is very similar to the dot format. Software exists that can produce graph layouts for this gml format (such as yEd).

3 The representation of lattices in non-FCA graph formats

For interoperability between FCA and non-FCA software, it is essential to represent lattices in graph formats. Since there is not any graph format that is universally accepted as a standard by a variety of tools, it is difficult to decide which graph formats to convert to. Currently FcaStone supports the non-XML formats dot and gml. In later versions XML formats (GraphML, XGMML) may be added. It is difficult to represent FCA lattices in these non-FCA graph formats because concept lattices use many labels per node (objects and attributes), which require special placement (below or above the node). Other graph applications usually only have one label per node. Thus the placement of several labels per node is a challenge.

The default in FcaStone is to concatenate all objects belonging to the same node into one string which is then cut off after 30 characters. The same is done with the attributes. If the -c option is used, the objects and attributes are "clarified", i.e., only at most one object and at most one attribute of each node is represented. The -t option places the objects of each node (and, separately, the attributes of each node) on top of each other. This is only useful if the output file is of type fig or svg and the file is afterwards edited in a vector graphics editor.

Two lattice designs are available[5]: Using the -b option, each node is represented as a box (see Fig. 2). This is similar to the Graphviz files generated by Colibri. The objects are listed in the bottom half, the attributes in the top half. The advantage of this format is that the labels never overlap because Graphviz will adjust the box sizes depending on the label sizes. The disadvantage is that this is not the standard FCA way of representing lattices. Also, some boxes in Fig. 2 are too large and the text is too close to the left

[5] The data for Fig. 2 and Fig. 3 and for a few other examples is available at `http://www.upriss.org.uk/fca/examples.html` together with sample pictures produced with FcaStone and Graphviz.

side. This is because in the particular Graphviz installation that was used to create this diagram, Graphviz had problems with some fonts, which seemed to affect some output formats more than others. Fig. 2 shows the default output. The output can be modified by changing Graphviz's attributes. But such modifications depend on the settings of the specific computer that is used. Most users will find that they need to experiment with their particular Graphviz installation in order to determine which combination of fonts and output formats works best.

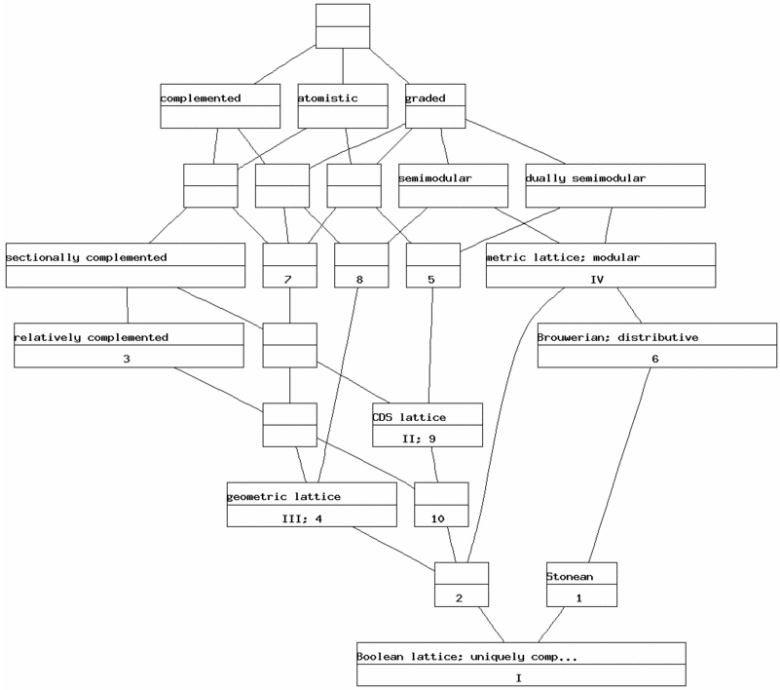


Fig. 2. An example of a layout with Graphviz (using data from Wille (1992)).

The other design (in Fig. 3) is more similar to traditional FCA lattices, with the disadvantage that labels can overlap. Again, the default output is shown, which in this case has very small fonts compared to the size of the lattice. These design choices are based on what can be stored in Graphviz's dot format. Just attaching two labels to a single node is difficult in the dot format. A hint found on the Graphviz mailing list suggests to attach an invisible self-edge to each node and then position the labels with respect to this invisible edge. This "hack" appears to be the only current solution to the labelling problem. Because the other vector and raster graphics formats are generated

by conversion from the Graphviz format, they all have the same layout. In the future, we plan to let FcaStone generate some output files (such as svg) directly by using only the graph coordinates, but not the rest of the file content, as provided by Graphviz. This will give more freedom in the ways how the lattices can be represented.

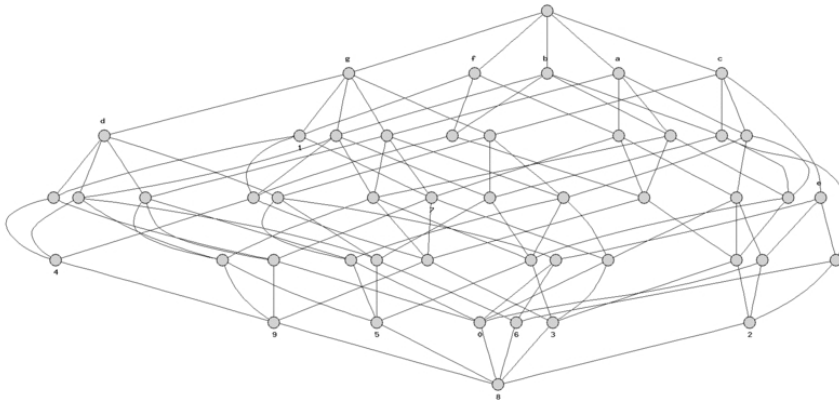


Fig. 3. A second example of a layout with Graphviz (using data from Stahl & Wille (1986)).

The gml output of FcaStone uses yet another type of design because of the lack of design options available with that format. In this format the labels are placed across the nodes, which is not satisfactory. Therefore, gml output should only be used if the lattice is afterwards manually edited. As far as we can see, gml has even fewer options for placing labels in different locations. The trick of using invisible self-edges appears not to be supported by this format.

In general, automatically generated lattices will probably never be as perfect as hand drawn ones. Graphviz's layouts are surprisingly good and in the case of Figs. 2 and 3 surprisingly similar to the published hand drawn ones. In both examples, some of the Boolean sublattices are visible, even though the edges are not parallel and some are even curved. If non-FCA tools are used for layout or editing, then one has to work with whatever features for graphs are provided by such tools. If perfect lattice pictures are desired, then traditional FCA tools should be used for editing. FcaStone's primary aim is to help users to produce the input formats for such tools. FcaStone's facility for lattice generation is more aimed at applications in which the lattices cannot be hand drawn (such as automatically generated ones on webpages) or do not need to be perfect (for example, because they are just used to provide a rough sketch of what the lattice looks like).

4 Using FcaStone with other tools

This section discusses interoperability with non-FCA tools. There are many reasons why such tools might be used. It may sometimes be necessary to edit lattices in ways not supported by traditional FCA software. For example, if one just wants to insert additional information for illustrative purposes, using traditional tools one would have to convert the lattice to a raster graphics format, then upload it into a vector graphics editor where one could add the additional information[6]. With FcaStone lattices can be exported straight into a format that can be read by vector editors. Another reason for using formats that can by processed by non-FCA tools is that other research communities have developed algorithms that are relevant for lattices, such as graph layout algorithms, and that FCA techniques are relevant for other communities. It would be desirable if there was greater exchange between FCA and related communities.

The following tools can be used with formats generated by FcaStone (the URLs for the tools are listed at the end of the paper):

- Text formatting
  - Latex is a document mark-up and preparation system. The FCA latex output is to be used with Bernhard Ganter's fca.sty. At the moment FcaStone can only generate latex files for contexts. In the future it is planned to also generate lattices in latex format.
- Graph layout and editors
  - Graphviz is an open source (CPL licence) graph layout program with a native format called "dot". It provides a variety of file conversion options. FcaStone calls it in order to produce the lattice layouts. Graphviz comes with the XWindows program "dotty", which can be used to edit the lattices (stored in the dot format). A list of other tools that can be used with dot files, is available on the Graphviz website.
  - yEd is a closed source, proprietary, but free to download, Java-based graph editor with GraphML as its native format. It is easy to install on Windows, Mac OS X and Unix/Linux. It can import gml files. yEd has its own graph layout functionality. FcaStone can produce gml files without Graphviz being installed.
  - Jgraph is an open source, proprietary, but free to download, Java-based graph editor. Presumably it can read gxl files, but we have not tested this.
- Vector graphics editors
  - Xfig is a Unix (Linux, Mac OS X) vector graphics editor with fig as its native format. WinFig is a version that runs on PCs; jfig is platform independent. Without the "-g" option, FcaStone produces a fig file of the context. With the "-g" option, the lattice is produced.
  - Inkscape is a an open-source, vector graphics editor with svg as its native format. It can be downloaded and installed via sourceforge as binary versions for Linux, Mac OS X, and Windows. Lattices in svg format can be uploaded into Inkscape. Inkscape has a connector facility which would make it possible to

[6] Some FCA software does support svg output, but, for example, in the case of ToscanaJ a special plugin is needed.

edit graphs so that moving a node also moves the connected edges. Unfortunately, the connections are stored using special Inkscape-only xml tags, which do not correspond to the svg files that are generated by Graphviz. This could be addressed in future versions of FcaStone.

- Dia is a GTK+ based diagram creation program for Linux, Unix and Windows released under the GPL license. It is pre-installed on many Linux distributions. A Windows executable is available. On other Unix and Mac OS X, it has to be installed from source. Graphviz can convert dot files into dia files, if the required libraries are installed.

5 FCA web applications

Ideally, it should be possible to use FCA tools on the WWW and in distributed environments in order to allow FCA modelling of remote data sources, sharing of FCA files across different client applications and improved tool interoperability in general. The top half of Fig. 4 shows the current use of FCA software on the web by applications such as Roget's Thesaurus[7]. At the moment, webserver FCA functionality is mostly restricted to either producing static graphics (raster graphics or svg), which are only viewed but not changed by a user, or to producing FCA files which are then saved by the user and uploaded into a stand-alone FCA application. On-line FCA software, such as Jon Ducrou's Surfmachine[8], exists which allows to interactively explore lattices, but does not allow to upload, download and save data. As far as we know there is no on-line FCA software which has the same kind of functionality as the stand-alone tools (ConExp, ToscanaJ, Galicia, etc) because it would be quite difficult to implement such a tool in an efficient and secure manner. In our opinion a better solution is a distributed approach where the server-side software focuses on producing data in an FCA format while the client-side software performs the resource-intensive operations of the GUI interface. It should not be too difficult to implement such an approach using the current tools, if the FCA community would agree on a shared FCA interchange format.

An FCA interchange format in XML could be extended to provide web services as shown in the bottom half of Fig. 4. Web services are a means for communication between servers and clients using XML messages. An example of the use of web services is a client interface that accesses information from an on-line search engine or e-commerce website by sending messages to a web API on the server. This technology is usually implemented using WSDL/SOAP or REST (which cannot be discussed in the framework of this paper). Credo[9] and SearchSleuth[10] are two examples of FCA front-ends which connect to the API of a search engine in this manner.

If there was an FCA interchange format, then FCA software, such as Roget, Credo and SearchSleuth, could produce an XML file in addition to its current html output. This XML file could then be read by any stand-alone FCA tool that supports web services technology. Users could use the tool of their choice to draw and explore lattices which

[7] `http://www.roget.org/`
[8] `http://www.kvocentral.org/software/surfmachine.html`
[9] `http://credo.fub.it/`
[10] `http://www.kvocentral.org/software/searchsleuth.html`

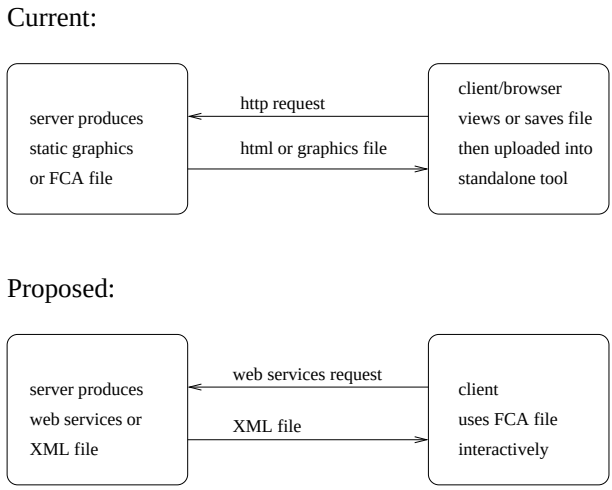


Fig. 4. Using FCA applications over the web

are produced from data that resides on the server. It would become much easier to develop new web-based FCA applications, because application developers would only need to write programs that extract and present the data as formal contexts, without having to worry about how to draw the lattices. The FCA algorithms would be provided by existing software.

The challenge of creating an FCA interchange format resides as much in deciding the content of the format as in obtaining an agreement from the developers and getting the format accepted. It might be that a currently existing format could be used as an interchange format if it was widely supported by FCA software. Part of the problem is to decide how much information to include in the interchange format. For example, should information about fonts and colours of the lattice diagrams be included? It is not the purpose of this paper to make any concrete suggestions, but hopefully this paper will help stimulating discussions about these issues. In the meantime, FcaStone attempts to fill the gap and provide some rudimentary means for interoperability by file format conversion!

URLs for the tools (FCA and non-FCA)

1. Colibri: `http://www.st.cs.uni-sb.de/~lindig/#colibri`
2. ConExp: `http://sourceforge.net/projects/conexp`
3. Dia: `http://live.gnome.org/Dia`
4. FcaStone: `http://fcastone.sourceforge.net`
5. fca.sty: `http://www.math.tu-dresden.de/ganter/fca`
6. Galicia: `http://www.iro.umontreal.ca/~galicia`

7. Graphviz: http://www.graphviz.org
8. Jfig: http://tech-www.informatik.uni-hamburg.de/applets/javafig
9. Jgraph: http://www.jgraph.com
10. Inkscape: http://www.inkscape.org
11. ToscanaJ: http://tockit.sourceforge.net
12. Winfig: http://www.schmidt-web-berlin.de/winfig
13. Xfig: http://www.xfig.org
14. yEd: http://www.yworks.com/en/products_yed_about.html

References

1. Chein, M.; Genest, D. (2000). *CGs Applications: Where Are We 7 Years After the First ICCS?* In: Ganter; Mineau (eds.): Lecture Notes in Artificial Intelligence 1876, Springer, p. 127-139.
2. Dobrev, P. (2006). *CG Tools Interoperability and the Semantic Web Challenges.* Contributions to ICCS 2006, 14th International Conference on Conceptual Structures, Aalborg University Press.
3. Ganter, Bernhard (1984). *Two basic algorithms in concept analysis.* Technische Hochschule Darmstadt, FB4-Preprint, 831, 1984.
4. Keeler, M.; Pfeiffer, H. (2006). *Building a Pragmatic Methodology for KR Tool Research and Development.* In: Schaerfe, Hitzler, Ohrstrom (eds.), Conceptual Structures: Inspiration and Application, Proceedings of the 14th International Conference on Conceptual Structures, ICCS'06, Springer Verlag, LNAI 4068, p. 314-330.
5. Rudolph, S.; Krötzsch, M.; Hitzler, P. (2007) *Quo Vadis, CS? On the (non)-impact of Conceptual Structures on the Semantic Web.* In: Priss, Polovina, Hill (eds.), Proceedings of the 15th International Conference on Conceptual Structures, ICCS'07, Springer Verlag, LNAI 4604, p. 464-467.
6. Stahl, J.; Wille, R. (1986). *Preconcepts and set representation of contexts.* In: Gaul & Schader (eds): Classification as a tool of research.
7. Tilley, Thomas (2004). *Tool Support for FCA*. In: Eklund (ed.), Concept Lattices: Second International Conference on Formal Concept Analysis, Springer Verlag, LNCS 2961, p. 104-111.
8. Wille, Rudolf (1992). *Concept Lattices and Conceptual Knowledge Systems.* Computers Math. Applic., 23, 6-9, p 493-515.

A visual mapping tool for database interoperability: the HealthAgents case

Roman Roset[1], Miguel Lurgi[1], Madalina Croitoru[2], Bo Hu[2],
Magi Lluch i Ariet[1], Paul Lewis[2]
[1] MicroArt, Barcelona, Spain
[2] ECS, University of Southampton, UK
{rroset,mlurgi,mlluch}@microart.cat, {mc3,bh,phl}@ecs.soton.ac.uk

Abstract In this paper we present a visual mapping tool for creating mappings between a relational database schema and an ontology. Our work is motivated twofold: on one hand mapping scripts are very difficult to be written by hand by non computer experts, on the other hand practical scenarios require such mappings to be created "on-the-fly" by domain experts. We believe that our automated tool clearly illustrates the use of visual tools in the context of Semantic Web practical applications in general and eHealth applications in particular.

1 Introduction

In this paper we present a visual mapping tool for creating D2RQ [1] mappings between a relational database schema and an OWL [8] ontology. We present the practical rationale that led to the development of this tool and its functionality.

With a larger and larger number of ontologies available the need for accessing existing data stored in various relational databases has dramatically increased. This called for different languages being developed in order to support this integration via a translation between ontological and database concepts. D2RQ is one of the most widely used mapping languages due to its flexibility and compatibility easiness. It allows to specify which concepts on the ontology correspond to which concepts in the database. Once the correspondences have been established, the ontology could allow access to the database via ontology based querying languages (RDQL [2], SPARQL [4] etc.). However, mapping scripts are very difficult to be written by hand by non computer experts. In some practical scenarios such mapping should be created "on-the-fly" by domain experts who might not be necessarily accustomed to D2RQ syntax. This calls for an automated tool that allows such mappings to be developed visually. To the best of our knowledge such tool does not currently exist. This is the reason why we developed the mapping tool presented in this paper. We believe that such a tool clearly illustrates the use of visual tools in the context of Semantic Web practical applications.

The domain where we applied this tool is eHealth, in the concrete context of the HealthAgents [1] system. One of the main objectives of HealthAgents is the

[1] http://www.healthagents.net

integration of a large number of geographically distributed clinical databases of brain tumour cases. The potentially large number of acquired cases will then allow for better tumour classification using machine learning algorithms (classifiers) also distributed in a multi-agent framework. The obvious interoperability problems (different database schemas across different hospitals) was overcome by adopting a "common language": an ontology. This ontology will smooth out the integration of the schemas while also allowing for reasoning capabilities over the mediated schema. The classifiers could then select the cases that they want to train on / classify by querying the distributed databases via the ontology. This querying will require, however, a translation between the concepts of the ontology and the concepts of the database schema. If a new hospital wants to join the HealthAgents network this mapping will need to be created from scratch, usually by medical staff. At this point the need for an automated tool for creating such mapping became impetuous with clear clinical usability requirements.

Regarding related work we feel that ours was inspired by an unique challenge of mapping disparate relational databases from diverse clinical partners onto the HealthAgents ontology. Despite of this, we did some research on the literature about the current state of the art on these kind of tools.

We found that certain existing tools like KAON Reverse [3] did not handle latest standards (OWL, D2RQ) and we experienced great difficulty in using those tools for our technological needs. On the other hand, more modern tools employing the latest standards, like for example: TopBraid Composer [5]; although offering great flexibility and functionality, and in its case in particular the capability to perform mappings between ontologies, were not designed for ontology - database mapping.

For these reasons, and the lack in the literature of a tool that could be used for the purposes we wanted it for, and in the way wanted us to be used in order to facilitate the work for the clinicians using their own databases, we decided to take the challenge of designing and developing our own tool.

We will present this tool in the remainder of the paper. First, we introduce the HealthAgents system and its particular clinical requirements, in section 3 we describe the functionality of the developed mapping tool, and we conclude by presenting the ongoing work aimed at improving its usability.

2 HealthAgents

HealthAgents [6] is an agent-based, distributed decision support system (DSS) that employs clinical information, Magnetic Resonance Imaging (MRI) data, Magnetic Resonance Spectroscopy (MRS) data and genomic DNA profile information. The aim of this project is to help improve brain tumour classification by providing alternative, non invasive techniques. A predecessor project, Interpret [7], has shown that single voxel MRS data can aid in improving brain tumour classification. HealthAgents builds on top of these results and further employs multi voxel MRS data, as well as genomic DNA micro-array information for better classification results. Moreover, HealthAgents is decentralizing the Interpret

DSS by building a distributed decision support system (d-DSS). This way, the number of cases to be studied is increased, improving classifiers accuracy.

At the moment, the data in the HealthAgents system is stored in relational databases at the various participating European clinical centers. An uniform vocabulary needed for interoperability reasons is provided by means of HADOM (HealthAgents Domain Ontology). It conceptualises the parameters of the employed techniques (MRI, MRS, DNA microarrays etc.), the clinical information (age, sex, tumor location etc.) and the known brain tumor classes compliant to WHO (World Health Organisation) [2].

The patient concept is at the center of HADOM (see Figure 1(a)). Each visit of a patient is given an unique ID to be differentiated from other EHR regarding the same person. A particular patient instance, therefore, has several associated patient records. Tissue focus defines instances of the concerned areas under two sub groups, namely Primary_Focus and Secondary_Focus. A particular focus is related to one visit of a patient via Patient_Record (see Figure 1(b)). Many medical instruments and methods have been developed to diagnose brain tumour. In HADOM, we enumerate the following approaches and define them as sub-concepts of Medical_Control: Biopsy, HRMAS, Magnetic_Resonance, and Microarrays.

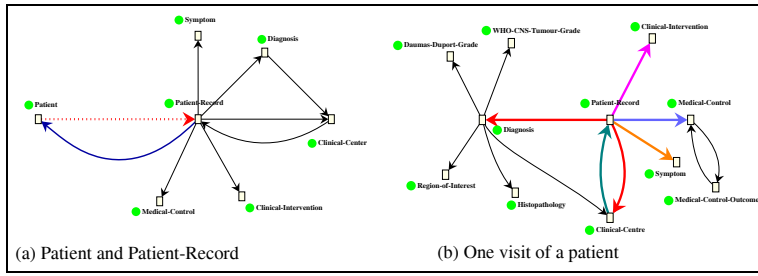


Figure 1. Conceptual view of HealthAgents HADOM

The HADOM ontology provides the basic terminology for the HealthAgents database schema and allows for interoperability at the terminological level. This is illustrated in Figure 2.

At the middle of the development we found ourselves in the necessity of integrating a number of databases (distributed over the nodes belonging to the HealthAgents network), each one with its own schema and different from each other. Obviously, the ontology framework provided by the system itself was the straightforward way to do this, however, it was a tedious job to map all the data stored in each database to the ontology, and so was born the idea of having a visual tool for doing this job in an easier way.

[2] Available from Harvard Medical School at: http://neurosurgery.mgh.harvard.edu/newwhobt.htm

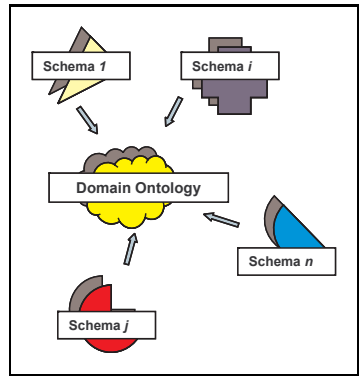


Figure 2. Ontological interoperability of HealthAgents database schema

3 Approach

The HealthAgents Mapping Tool is a software application developed for mapping between a given relational database and a given OWL ontology. Based on the idea of automating the mapping process between an ontology (concepts and properties) and a relational database schema, we designed a tool inspired by the drag and drop paradigm. This tool allows the user to relate concepts on a given ontology to entities present within a relational database with the final goal of obtaining a mapping description, using the D2RQ language for its representation.

The D2RQ framework contains a mapping language for treating non-RDF relational databases as virtual RDF graphs, and a platform that enables applications to access these graphs through the Jena and Sesame APIs, as well as over the Web via the SPARQL Protocol and as Linked Data.

The D2RQ language offers great flexibility from the point of view of mapping relationships, since it allows for a great range of properties and relations between concepts to be represented. The full specifications of the D2RQ language as well as the description of the entire platform are described in the D2RQ specifications web page [3]; where all the concepts, properties, and relationships that can be represented within a mapping description created using this language are presented.

The mapping description introduced above will allow any given user, who does not need to know the organisational schema of the database, to query the database via the SPARQL language on the ontology.

Basically, the tool facilitates the production of a mapping description between an OWL ontology and a relational database. This need for automated mapping tools is even greater in a Semantic Web era when ontologies are commonly used for interoperability and most of the data still resides in local database schemas.

[3] Available at: http://www4.wiwiss.fu-berlin.de/bizer/d2rq/spec/

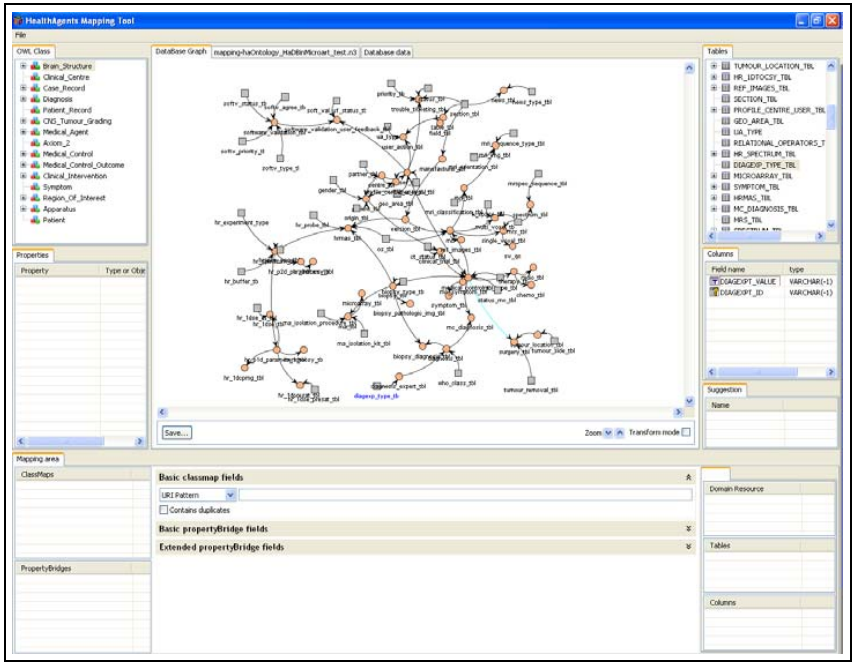


Figure 3. The HealthAgents Mapping Tool.

In the following, we present the functionality of the tool we developed and its usage in a practical setting. This will allow us to illustrate the approach taken for its design, and will also convey the full extent of the tool's capabilities. In doing this, we will start by roughly describe a typical workflow of an user working with the application.

The user is presented, on one hand, with the possibility of loading an OWL ontology that will be visualised through the built in interface. At the same time, the user can load a relational database schema specified either by an XML file or by access to the location of the actual database server. The database schema will be visually available through the interface provided by the application.

Once the ontology specification and the database schema are loaded into the application, the workflow begins by presenting the user with a directed graph that shows the entities (nodes) within the database and the relationships (e.g. foreign keys etc.) amongst them. Apart from this, a series of windows appear which are used to specify the concepts and entities to be related in a drag and drop manner. Figure 3 shows a screenshot of the application with an ontology and a database loaded. At the center of the window, the graph representing the database schema is displayed.

For the sake of clarity, in order to improve the application usability, the workspace is divided into four different areas used to present the different type of information involved in the mapping process:

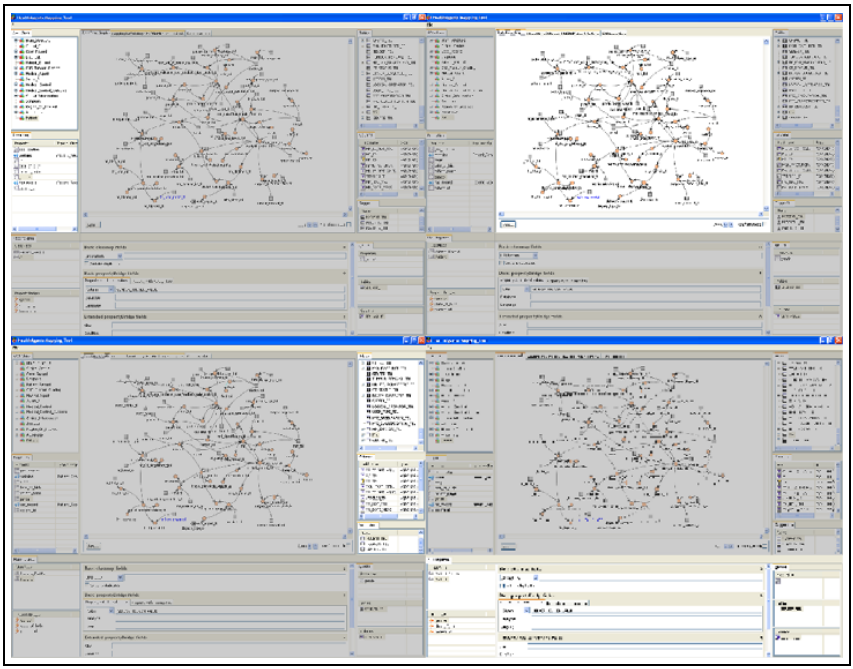


Figure 4. HealthAgents Mapping Tool spaces. Top left: the ontology area; top right: the visualisation area; bottom left: the database area; and bottom right: the mapping area

- **The ontology area:** Shows, in two windows, the concepts available within the ontology, and the attributes of the currently selected concept;
- **The visualisation area:** Apart from the graph mentioned above it presents two more tabs, displaying: the D2RQ file being generated by the mapping process, and a table presenting the data available on the database for the selected entities over the schema;
- **The database area:** Shows the schema of the specified database (tables and their fields); also makes available a window with suggestions of database schema tables for the mapping of the currently selected ontology concept;
- **The mapping area:** Displays the D2RQ specifications and the way of presenting the information of the mapping. In this space all the D2RQ specifications can be filled in to obtain a complete mapping description. This area

also fosters two subspaces: one on the side of the ontology, and the other on the side of the database; where the ontological concepts and properties, and the database tables and fields, that are being related within the current mapping description, are respectively presented.

For illustrative purposes figure 4 shows the different organisational areas used for the presentation of the information within the application. Within the figure: the top left screenshot shows the ontology area, the top right screenshot shows the visualisation area, the database area is presented on the bottom left screenshot, and finally, the bottom right screenshot presents the mapping area.

Given the intuitive character of the application's interface, the only thing to be done in order to relate a concept in the ontology (or any of its attributes) to an entity within the database is to select the desired object, drag it to the correct window within the mapping area, and do the same for the corresponding object in the database. Figure 5 shows how easily the mapping process is carried out, with a few concepts already mapped and an ontology concept being dragged to the mapping area to relate it with its counterpart within the database.

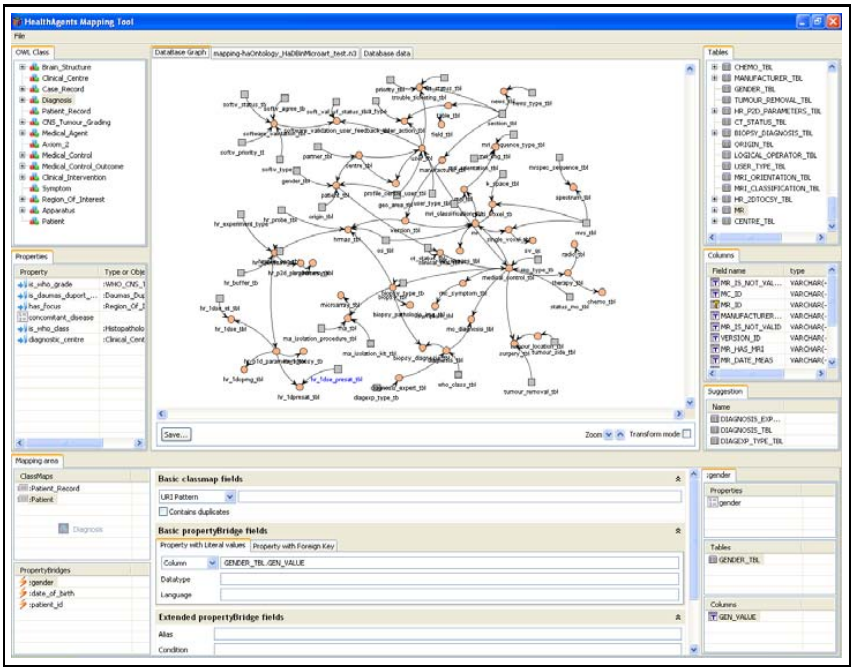


Figure 5. The mapping process. Dragging an ontology concept to the mapping area.

In order to achieve the completion of the mapping description the user must repeat the action described above for each one of the concepts on the ontology that need to be mapped. At any time during the development of the mapping, the user is allowed to visualise the final D2RQ file, which stores the mapping description. Another useful functionality is the possibility of querying the database, at any time, using the tab window provided for that specific purpose. This might help the user to decide, based on the information stored in the database, to which concept on the ontology any given entity within the database is related.

Although the mapping is done complying with the minimum specifications of the D2RQ declarative language, the application offers the possibility to create a complete mapping by presenting the user with a complete set of relations and attributes that can be specified for each of the concepts that are being mapped (following the D2RQ language specifications).

At the end of the mapping process carried out using the HealthAgents Mapping Tool application, the user has, within a D2RQ file, all the mapped concepts and the corresponding relationships with the original database entities, i.e. the mapping description (figure 6 shows an example of the D2RQ file). This allows to query that particular database at any given time, through the D2RQ platform; and also enables applications to access the database over the Web, using the mapping file generated and the SPARQL protocol as an intermediate layer between the database and the application wishing to access the data.

In the case of HealthAgents, accessing the information stored on a relational database, and mapping it to an ontology, is the main purpose for the utilization of this kind of application, and the methodology described above (using the mapping file and the SPARQL protocol) is the one which the HealthAgents system employs for accessing the data on the different databases connected to the HealthAgents network.

It is easy to see how this tool is useful for different types of applications and for people working in different areas, since the problem of accessing information stored within a relational database over the Web has been always present and in most of the cases has been sorted out with the use of ontologies.

One particular domain in which the HealthAgents Mapping Tool is essential is eHealth, given the large number of centers and clinicians willing to join the HealthAgents network for sharing data. The information they possess is stored in their local databases, hence they need a way to make that data available on the network. One way of achieving this, currently provided by the HealthAgents network, is mapping their own schemas into the HealthAgents ontology. The HealthAgents Mapping Tool offers the possibility to make this mapping process easy, by providing the clinicians with an intuitive tool that displays the schema of the local database along with the ontology they aim to make their data compatible with.

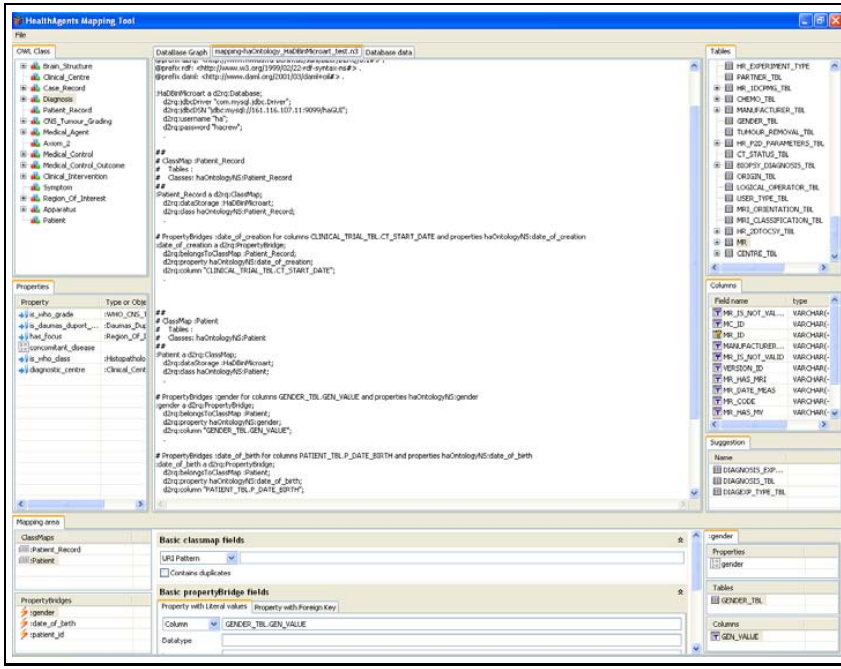


Figure 6. D2RQ file containing the mapping description.

4 Ongoing work and Drawbacks

Although the tool presented above has been successfully deployed in the context of HealthAgents, there are a number of drawbacks which we have identified. Addressing these drawbacks is current ongoing work and it would give added value for the researchers and developers using it. The following list presents some features that we think could enhance the capabilities of the tool and increase its usability:

– **The direction of the mapping:** At the moment, the mapping is done in just one direction, from the ontology to the database; in practical terms this means that the final result of the mapping process will allow the user to query the information on the database via the ontology, but will not be possible to access the information on the instances of the ontology concepts in order to use this data for storing it in the database. Looking at this fact, we think that it could be of interest to be able to perform the mapping from the database to the ontology as well, since there might be cases in which users would like to add the data in the instances of the ontology concepts to their own databases.

- **Visualisation of the ontology representation:** As mentioned in the last section, one major feature of the tool is that it shows the database schema in the form of a graph, i.e. the entities and their relationships are displayed using a directed graph on the visualisation screen, where the entities are the nodes and their relationships are the edges. But this visualisation is only available for the database schema, giving to the user a visual perspective of the database in order to facilitate the mapping process. For this purpose of facilitating the process, it could also be interesting to be able to visualize the ontology representation. There are several tools on the market that offer this functionality, and we think that it is of interest for the HealthAgents Mapping Tool to incorporate this functionality as well, allowing the user to visualize not only one side of the mapping, but both of them, for increasing the intuitiveness of the process and making it more natural.
- **Visualisation of intra mapping relationships:** When carrying out the mapping, it can be useful for the user to visualise the relationships that have been created so far, until a given state of the mapping process, between the entities within the database and the concepts on the ontology, with the objective of making easier for the user to visualise the concepts that have already been mapped, the ones that still remain to be mapped, and the relationships that exist between the objects of the two data representations. Again, as the previously presented desired features, this will facilitate and accelerate the mapping process.

5 Conclusions

In this paper we presented a visual mapping tool for creating correspondences between ontological and database concepts using the D2RQ language. The need of such tool arose in the context of the HealthAgents system where clinical partners are required to map their database schema to the system's ontology in order to support interoperability. We presented the functionality of the tool and a real case use of such system where an existing database (in Microart, Barcelona) was mapped to the developed HealthAgents ontology, HADOM. We concluded the paper with ongoing directions of work addressing different usability aspects aimed at improving the tool both from a practical view point (according to clinical feedback) but also from a research direction (ontology visualisation, ontology querying completeness).

Acknowledgements

This work is supported under the HealthAgents STREP project funded by EU Framework 6 under Grant number IST-FP6-027213.

References

1. The D2RQ Platform- Treating Non-RDF Databases as Virtual RDF Graphs. http://www4.wiwiss.fu-berlin.de/bizer/d2rq/. Last visisted 11/04/2008.

2. RDQL - A Query Language for RDF. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/, January 2004. Last visited 11/04/2008.
3. KAON Reverse. http://kaon.semanticweb.org/alphaworld/reverse/, January 2008. Last visited 11/04/2008.
4. SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/, January 2008. Last visited 11/04/2008.
5. Top Braid Composer. http://www.topquadrant.com/topbraid/composer/index.html, May 2008. Last visited 26/05/2008.
6. C. Arús, B. Celda, S. Dasmahapatra, D. Dupplaw, H. González-Vélez, S. van Huffel, P. Lewis, M. Lluch i Ariet, M. Mier, A. Peet, and M. Robles. On the design of a web-based decision support system for brain tumour diagnosis using distributed agents. In *WI-IATW'06: 2006 IEEE/WIC/ACM Int Conf on Web Intelligence & Intelligent Agent Technology*, pages 208–211, Hong Kong, December 2006. IEEE.
7. A. R. Tate, J. Underwood, D. M. Acosta, M. . Julia-Sape, C. Majos, A. Moreno-Torres, F. A. Howe, M. van der Graaf, M. M. Lefournier, F. Murphy, A. Loosemore, C. Ladroue, P. Wesseling, J. L. Bosson, A. W. Simonetti, W. Gajewicz, J. Calvar, A. Capdevila, P. Wilkins, A. C. Bell, C. Remy, A. Heerschap, D. Watson, J. R. Griffiths, and C. Arus. Development of a decision support system for diagnosis and grading of brain tumours using in vivo magnetic resonance single voxel spectra. *NMR Biomed*, 19:411–434, 2006.
8. W3C OWL Reference. OWL Web Ontology Language Reference — W3C Recommendation 10/2/2004. http://www.w3.org/TR/owl-ref/, 2004.

Author Index