FCA Software Interoperability

Uta Priss

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

Abstract. This paper discusses FCA software interoperability from a variety of angles: because the central FCA structures, formal contexts and concept lattices, can be represented in non-FCA software, interoperability with such software is of relevance. The non-FCA software in question is spreadsheet, relational database, graph and vector graphics software. The simplest approach to interoperability consists of providing file format conversion tools, such as FcaStone, which is therefore also discussed in this paper. Interoperability can be hindered by social factors, i.e. if the FCA researchers do not want to use non-FCA software. This issue is investigated with respect to software-derived graph layouts of lattice diagrams. An experiment that compares different software-derived lattice diagram layouts is conducted and leads to a surprising result.

1 Introduction

There appears to be some controversy among Formal Concept Analysis (FCA) researchers in how far FCA software should interoperate with other software. Some researchers complain about the lack of interoperability among FCA tools and the lack of connections between FCA and non-FCA applications. For example, formal contexts might be presentable in spreadsheet and relational database software whereas lattice diagrams might be edited in graph and vector graphics software. Other researchers express the view that the quality of FCA will be diminished if non-FCA approaches are applied, for example, with respect to non-FCA graph layout algorithms. This paper discusses different aspects of FCA interoperability and in particular investigates the use of non-FCA algorithms for graph layouts. A small experiment is conducted by deriving layouts of five well-known examples of formal contexts using FCA and non-FCA graph software. The experiment has a surprising result.

Section 2 of this paper provides a brief overview of the interoperability discussions in the FCA community. Section 3 discusses the relationship between some FCA and non-FCA tools. Section 4 describes the FCA file format conversion software FcaStone. Section 5 compares graph layouts derived with different FCA tools.

2 The FCA interoperability discussion

This section provides a brief overview of the discussion of interoperability in the FCA community and in the slightly broader conceptual structures (CS) community. In past

ISBN 978–80–244–2111–7, Palacký University, Olomouc, 2008.

years, several ICCS authors expressed disappointment with the lack of progress in CS research with respect to applications and software (Chein & Genest (2000) and Keeler & Pfeiffer (2006)). Members of the ICCS community tend to agree that CS ideas are in principle extremely relevant to modern information representation tasks and, for example, the Semantic Web, but for some reason CS research has not been able to influence mainstream research communities (Rudolph, Krötzsch & Hitzler, 2007). In particular the software that is currently available for conceptual graphs (CG) and FCA does not reach the full potential of CS research and is not yet of commercial quality. Several suggestions have been made by the CS community to improve the situation. Keeler & Pfeiffer (2006) suggest to employ a pragmatic methodology for tool development using a "game" metaphor. Rudolph et al. (2007) suggest to establish connections with larger existing related communities (for instance, the Semantic Web community). Others have organised CS tool interoperability workshops[1] and challenges[2].

Dobrev (2006) presents an overview of interoperability issues of CG tools. He argues that although limited data exchange between CG tools is possible at the syntactic level using the standard exchange format, exchange at a semantic level, which incorporates contextual and background knowledge is not yet possible. In contrast to the CG community which has an ISO approved standard for Common Logic[3], there is no similar standard for FCA. The rest of this paper is only concerned with FCA software, not with the broader field of CS software. Tilley (2004) provides an overview of FCA software as described in FCA research papers. Interoperability between FCA software is low. Each software has different storage formats and different input/output options, which are not necessarily compatible with other software. Most of the FCA software appears to be at a somewhat "prototypical" stage and not of the same quality as commercial software. Although there is an overlap of features between different FCA software tools, certain features are only available in certain software. Thus in order to use all features that are currently implemented, a user would need to download and install several different tools and then try to figure out how to export data from one tool so that it can be incorporated into other tools, which is not always possible. In theory, it should be easy to convert between the different XML formats, but in practice, all of the current FCA XML formats have a completely different semantics. Because of the lack of interoperability among the tools, new developments, such as newly discovered faster algorithms, have to be implemented separately by the developers of each tool. There is no plug-in architecture that would allow algorithms to be easily incorporated into different tools.

3 Interoperability with non-FCA software

This section argues that FCA software shares a number of features with non-FCA software. More specifically, software for representing and operating on formal contexts shares features with database and spreadsheet software. Software for displaying and

[1] http://www.kde.cs.uni-kassel.de/ws/cs-tiw2008
[2] https://skyhawk.cs.uah.edu/concept/index.php/ICCS_Challenge
[3] http://cl.tamu.edu/

editing concept lattices shares features with vector graphics and graph drawing software. The difference between graph drawing and vector graphics software is that vector graphics is more general. Graphs consist 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 editors, 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 editors 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 software is of interest to FCA, but because of the differences between them, not all FCA features can be represented with such software.

It should be noted that the discussion in this section focuses on software, not mathematical modelling. Thus some of the mathematical aspects, such as the difference between abstract lattices, Hasse diagrams and general graphs are ignored if they are not immediately relevant for what is implemented in software tools. Furthermore, the list of FCA features that is discussed is not complete and depends on the current state of the art of FCA research and software technology.

Fig. 1 lists FCA features which are currently provided by FCA software. The context-related features are grouped into features that are shared with spreadsheet and database software. Spreadsheet software allows to create cross tables in which data can be entered, rows can be permuted and display parameters (font, colour, etc) can be changed. Relational database software also allows to store and edit objects, attributes and their relationships (crosses). But tables in relational databases need not be binary relations; databases are more akin to power context families. Before lattices can be drawn, users need to build binary contexts from the data in the database. FCA software should interoperate with spreadsheet and database software. Of course, not all context features are provided by spreadsheets and databases. Thus, although data can be imported from spreadsheets and databases, such software is not suitable as a sole interface for formal contexts.

With respect to displaying concept lattices, both vector graphics software and graph editing software have many features that are commonly used to modify lattice diagrams. Several FCA tools allow for lattice diagrams to be exported in SVG (scalable vector graphics) format. If minor edits are required that are not supported by the FCA software, it is possible to create a lattice using the FCA software and then to export the diagram and use a vector graphics program for further editing. For example, the graph layout algorithms, the manner in which the objects and attributes are displayed and so on could be implemented as options that the user chooses when exporting a diagram. Graph editing software has two important features that are not necessarily available in vector graphics software: the availability of graph layout algorithms and the feature of clicking on a node to move it in a manner that the attached edges stay attached. Modern vector graphics editors, such as Inkscape[4] and Dia support this to some degree. But because

[4] The URLs for all tools mentioned in this paper can be found on the last page of this paper.

there is no universally accepted graph representation format and Inkscape and Dia have their own formats, it is difficult for FCA software to export the lattice diagrams in formats that preserve sufficient information and can be read by graphics software. Older vector graphics editors (such as xfig) tend not to have graph functionality and are thus not as suitable for lattice editing.

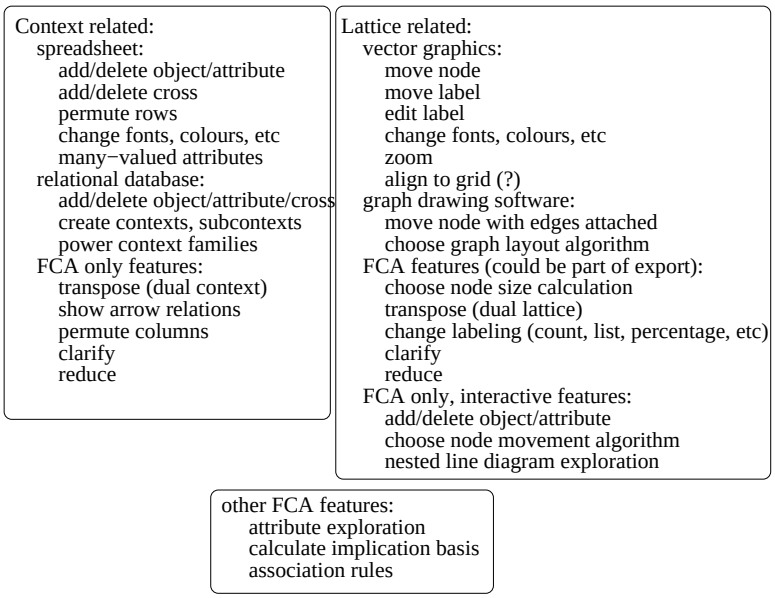

Fig. 1. Tasks for FCA software

Although vector graphics and graph editors provide means for adding and deleting, such features may not be consistent with the FCA features for adding or deleting objects, attributes and concepts. It can be a danger that inexperienced users might modify a diagram using the editor's add/delete features in such a manner that the diagram is no longer a lattice. Experienced FCA users might miss the ability to choose "node movement algorithms", i.e. the ability to move a whole filter or ideal in a lattice by dragging a node. It seems unlikely that current vector graphics and graph editors have such functionality. But this would be an opportunity for FCA developers to communicate with graphics editor developers. Maybe it would be possible to add such functionality to the editors. The Dia software, for example, supports different application modes (e.g. ER diagrams, flow charts). Maybe it would be possible to add an FCA mode to that program. Maybe the developers in the vector graphics communities would also be interested in layout algorithms that have been developed by FCA researchers. This might be a good opportunity for collaboration.

Complex FCA features, such as the exploration of nested line diagrams will maybe never be supported by traditional vector graphics editors. But Priss (2008a) discusses

nested line diagrams as a means of "faceting", as used in library and information science. Several software tools for manipulating facets exists. Thus there could be some overlap in technology between software for faceted classification and FCA software. Other FCA features, such as association rules and implications are shared with data mining approaches. There could be opportunities for interoperability for FCA software in that area as well.

The simplest means of interoperability for FCA software with non-FCA software is to allow the import and export in compatible formats. With respect to spreadsheets and databases, FCA software should support comma-separated value files and with respect to vector graphics, the SVG format should be supported as an export option. It would be convenient to also allow input from graphics formats, but that is a difficult challenge because a lattice graph can be encoded in many different ways.

The question of whether non-FCA graph layout algorithms are useful for FCA software will be discussed in more detail further below. One obvious advantage for using external graph layout algorithms is that it eases the burden on the FCA programmers. The first popular non-FCA graph layout program that was used by FCA software was probably Graphplace (Eijndhoven, 1994), which converts a binary relation into a coordinate representation in a postscript format. A more modern program which implements many different graph layout algorithms and all kinds of features is Graphviz. The "directed graph" option in Graphviz provides layouts for lattices in a top-down manner. Graphviz also converts into many other graph, raster and vector graphics formats. Thus if FCA software exports lattices in a Graphviz format, then all these other formats are automatically accessible as well. The FCA tools Colibri and FcaStone make use of Graphviz.

4 FCAStone: FCA file format conversion software

A simple approach for allowing FCA software to interoperate with non-FCA software is by providing means for converting between the file formats of the different tools. FcaStone (named in analogy to "Rosetta Stone") is a command-line utility that converts between the file formats of commonly-used FCA tools (such as ToscanaJ, ConExp, Galicia, Colibri[5]) 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. FcaStone does not intend to compete with or replace the Java-based tools (ToscanaJ, ConExp, Galicia, etc) but instead to provide a different type of functionality, which is aimed more at server-side applications and conversion. FcaStone does not have a graphical user interface (GUI).

The emphasis of FcaStone is on converting file formats, but FcaStone can also convert formal contexts into lattices. It uses the Graphviz software to calculate the graph

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

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 somewhat difficult to produce concept lattice diagrams in a graph format, because the dual labelling of nodes with objects and attributes is not easily supported in non-FCA graph formats. Priss (2008b) discusses how lattices can be represented using Graphviz's format. Another problem is that Graphviz's layout of lattices produces curved lines, which is not usually accepted in the FCA community. Thus, some FCA researchers may not approve of using FcaStone and Graphviz to produce lattice diagrams. We argue that FcaStone's diagrams are produced without manual editing. There are applications where manual editing of lattices is not feasible, for example, if the lattice diagrams are produced on-line as a response to user queries. An advantage of Graphviz's layouts is that they can be generated in an overlapping-free manner. Out of the three open-source FCA tools, ToscanaJ, ConExp, and Galicia, only Galicia produces lattices which are overlapping-free (see the next section). If it was possible to export the lattice layouts from Galicia, FcaStone could use such layouts instead of Graphviz layouts. But as far as we know the graph coordinates cannot be exported in Galicia. More details about FcaStone and the formats it supports can be found in Priss (2008b).

5 Graph layout for lattices

The previous sections have highlighted different aspects of FCA interoperability with non-FCA software. This section concentrates on comparing graph layouts produced by different tools. Five of Rudolf Wille's (the founder of FCA) well known examples of formal contexts have been selected. The five examples are fairly randomly chosen from an overview lecture given by Wille at the 2007 KPP conference[6]. The first example, "digits" was originally published in Stahl & Wille (1986). The "bodies of water" and the "live in water" examples were published in Wille (1984) and the "tea ladies" and the "lattice properties" examples were published in Wille (1992). The background of these lattices shall not be discussed in this paper because we are only interested in the representation of the line diagrams of these lattices. All five examples have reasonably complex line diagrams.

It is not the aim of the experiment conducted here to rank FCA software with respect to the "quality" of their diagrams. All FCA tools that were used here have different purposes. For example, the diagrams produced by Siena (part of the ToscanaJ suite) are intended for manual editing. Siena's initial layout contains many overlapping nodes. But because the initial layout contains many parallel edges, it only requires a few nodes to be moved manually in order to obtain a diagram that preserves the parallel edges. In general, there is some disagreement among researchers as to what diagrams should look like, whether they should have parallel edges, symmetries or whether the nodes

[6] http://www.fbi.h-da.de/kpp2007.html

should be arranged on levels. This paper does not intend to provide any judgement on these issues.

This paper is only interested in what we call "graphical similarity" of line diagrams. First, we define the "position" of a node in a line diagram as follows: if the nodes are arranged in levels starting from the top, then the position of a node refers to the level it is on and the distance it has from the side. Position "0,0" is the top node. Position "1,0" refers to the nodes that are the furthest to the left and right among all neighbours of the top node, and so on. This could either be one node, if the top has only one lower neighbour, or two nodes. Two line diagrams A and B are called "graphically similar" if a) they contain the same number of edge crossings and b) a node that is in the same "position" in A and B has the same number of upper and lower neighbours in A as in B.

Wille uses a particular method for drawing line diagrams, called the "geometric method" (Ganter & Wille, 1999). This paper intends to test whether any of the default lattices produced by commonly used FCA tools produce lattices that are similar to the lattice layouts that Wille derived with his geometric method. Again, it should be stressed that this is not intended as a value judgement with respect to the quality of these diagrams. But since some users may want to produce layouts that are similar to Wille's it would be useful if software existed that produced such layouts on demand.

We conducted the following experiment. We derived the default layouts of the five formal contexts in ConExp, Galicia, Siena (part of ToscanaJ) and using the Graphviz layout of FcaStone. The three FCA tools were chosen because they are open-source and freely available. Furthermore, we manually reproduced Wille's layouts. For the automatically derived layouts, we allowed ourselves only to change font sizes and node sizes. None of the nodes were moved. The production of the pictures was helped by the FcaStone software because with this software it took only seconds to convert the formal contexts into formats that can be read by the different tools. We apologise that the fonts in the pictures are too small to read. Only the layouts matter for this paper. We have provided a website[7] where researchers can find larger scale pictures and the contexts in "cxt" format so that this experiment can be reproduced and be extended to other FCA tools.

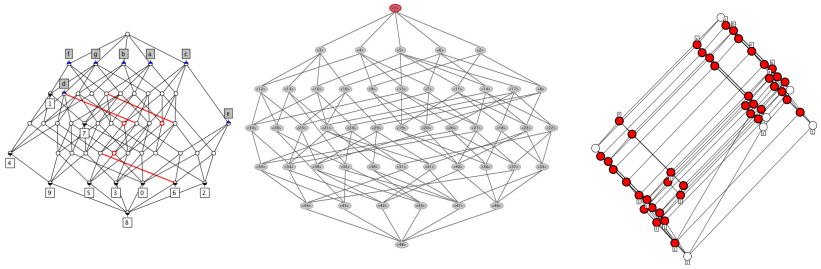

Fig. 2. The "digits" example: ConExp, Galicia, Siena

[7] `http://www.upriss.org.uk/fca/examples.html`

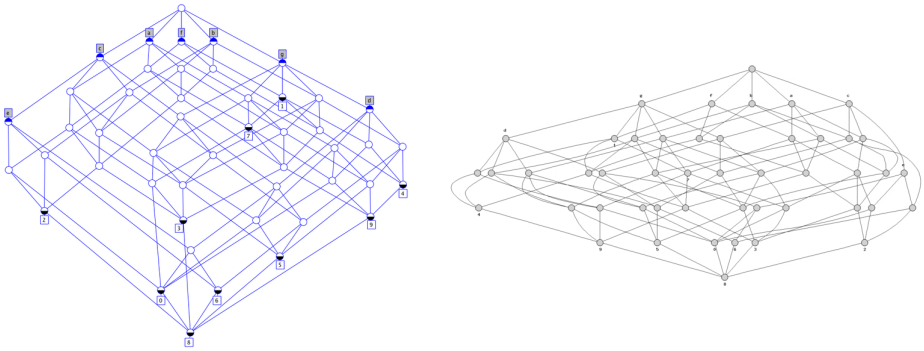

Fig. 3. The “digits” example: Wille’s layout, Graphviz

In our opinion, the result is surprising. The layouts that are produced by Graphviz are from an FCA view very unconventional because the edges are not parallel and in many cases even curved. Nevertheless, across all five examples, using our definition of “graphical similarity”, the lattices produced by Graphviz are similar to Wille’s layouts. It seems to us that if an algorithm was found that started with the Graphviz layouts and then straightened the edges and looked for parallel edges, it might be possible to automatically produce Wille-style layouts. In our opinion, this little experiment is an argument for increased interoperability between FCA and non-FCA tools. Even if non-FCA tools produce something that initially does not look appropriate (such as the curved edges in the Graphviz diagrams), it may ultimately have a functionality that is useful for FCA purposes. Only if FCA tools interoperate with non-FCA tools, it is possible to explore such features.

In the “digits” example, Graphviz’s and Wille’s layout are graphically similar because they are almost mirror images of each other. In ConExp and Galicia, the nodes are more permuted and not in the same positions. In all examples, Siena is difficult to see because of the strong degree in overlap. In the “bodies of water” example, ConExp’s, Wille’s and Graphviz’s layouts are similar and are different from Galicia and Siena. In the “lattice properties” example, both Wille’s and Graphviz’s layout have 7 edge crossings, ConExp has 6, Galicia has more. In the “live in water” example, Wille’s, Graphviz’s and ConExp’s layouts differ by a few switched nodes and by one edge crossing. Galicia is very different and has more edge crossings. In the “tea ladies” example, the nodes neighbouring the top node are roughly (but not exactly) in the same positions in Graphviz’s and Wille’s layouts, but not in ConExp and Galicia.

6 Conclusion

This paper analyses FCA software interoperability from a variety of angles. It is argued that interoperability with non-FCA software can be challenging because non-FCA applications have entirely different aims and purposes. But there can be benefits. For example, it appears that the graph layouts provided by a non-FCA tool are in some sense

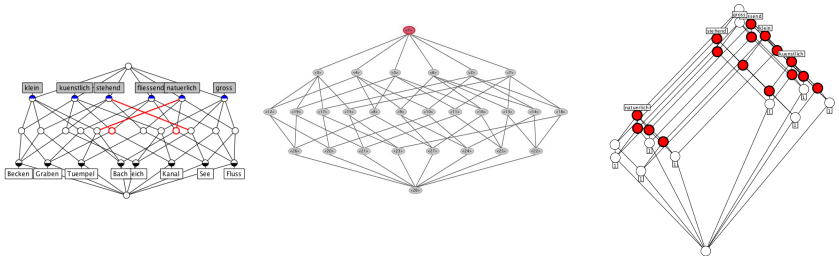

Fig. 4. The "bodies of water" example: ConExp, Galicia, Siena

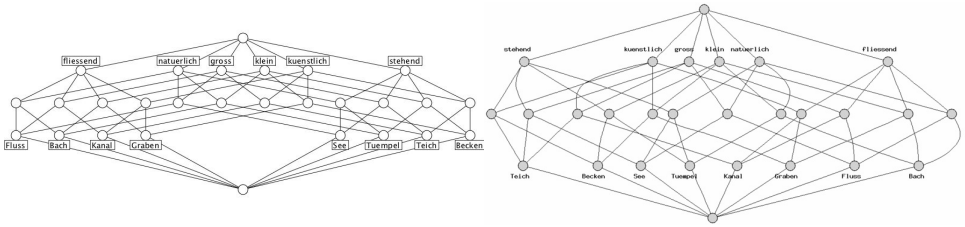

Fig. 5. The "bodies of water" example: Wille's layout, Graphviz

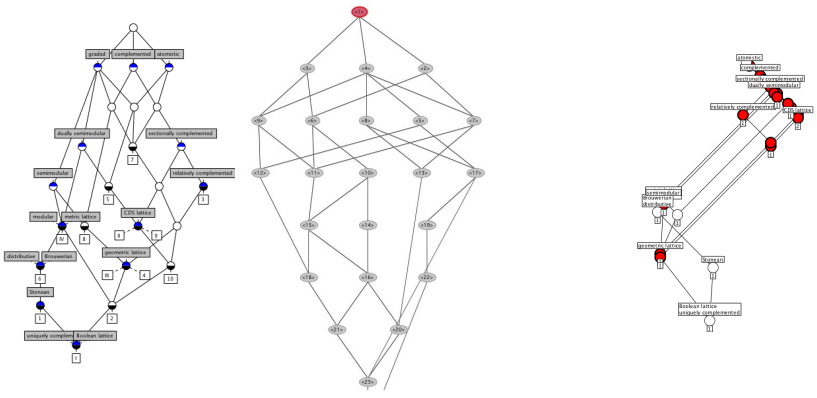

Fig. 6. The "lattice properties" example: ConExp, Galicia, Siena

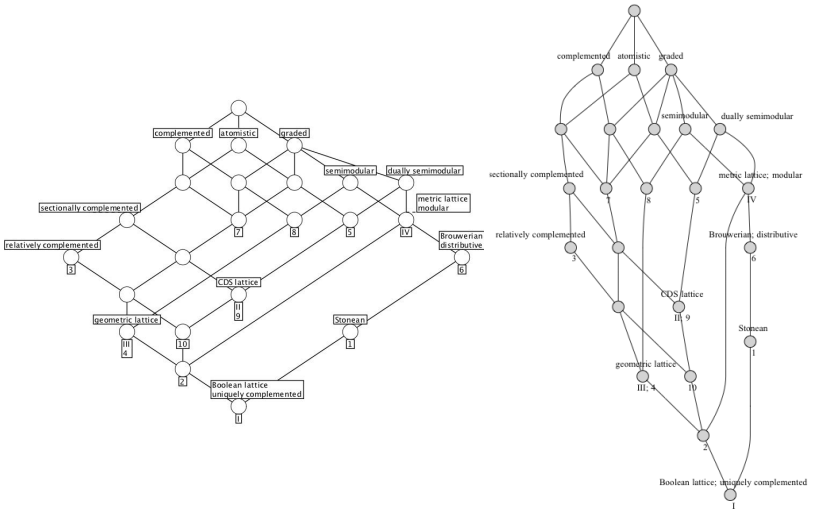

Fig. 7. The "lattice properties" example: Wille's layout, Graphviz

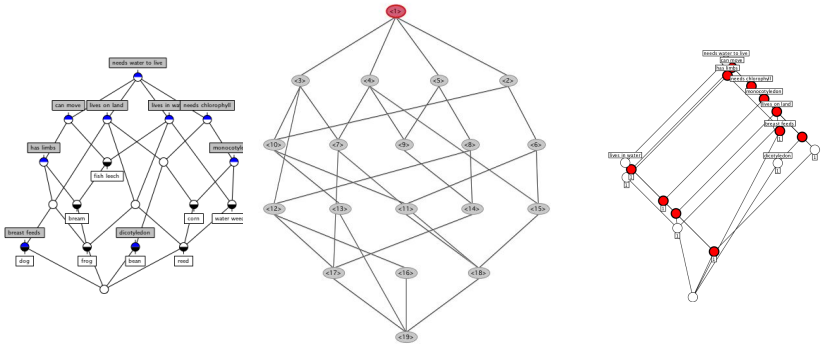

Fig. 8. The "live in water" example: ConExp, Galicia, Siena

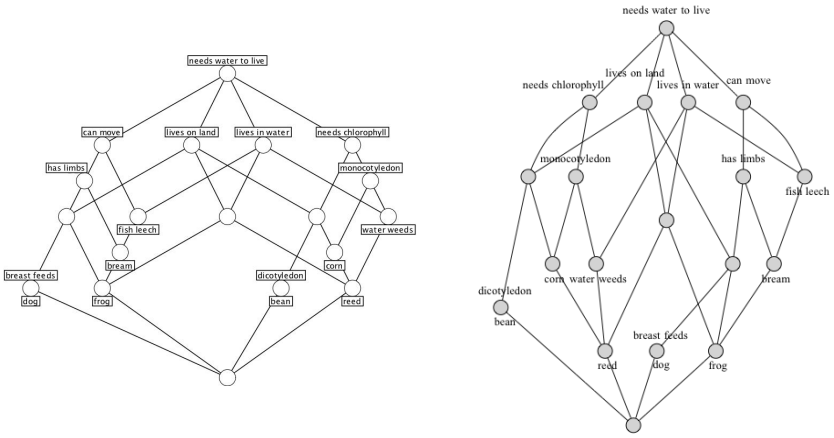

Fig. 9. The "live in water" example: Wille's layout, Graphviz

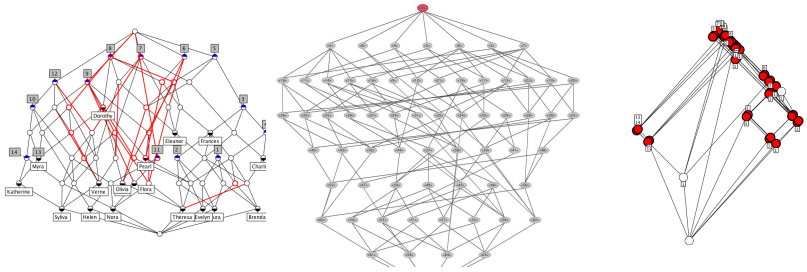

Fig. 10. The "tea ladies" example: ConExp, Galicia, Siena

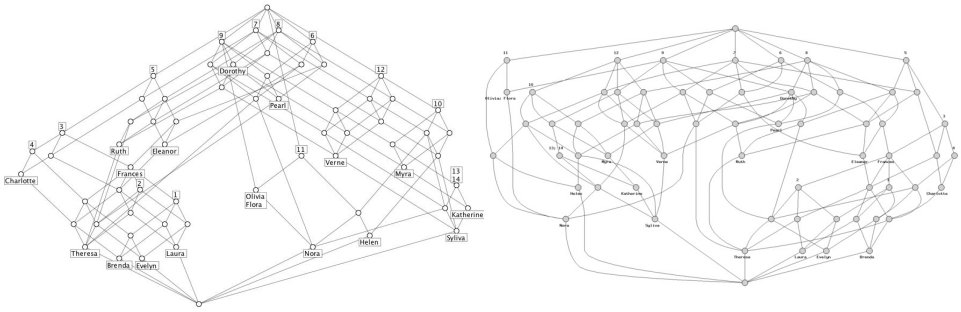

Fig. 11. The "tea ladies" example: Wille's layout, Graphviz

similar to manually derived layouts from researchers in the FCA community. Thus combining FCA software with non-FCA software can provide new insights and inspirations.

URLs for the Tools mentioned in this paper

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. Inkscape: `http://www.inkscape.org`
9. ToscanaJ: `http://toscanaj.sourceforge.net`
10. Tockit (related to ToscanaJ): `http://tockit.sourceforge.net`
11. Xfig: `http://www.xfig.org`

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. Eijndhoven, Jos van (1994). *Graphplace - a graph layouter*. Software, Eindhoven University of Technology, The Netherlands. Available via anonymous ftp from several sites.
4. Ganter, Bernhard; Wille, Rudolf (1999). *Formal Concept Analysis.* Mathematical Foundations. Springer Verlag.
5. 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.
6. Priss, Uta (2008a). *Facet-like Structures in Computer Science.* Axiomathes, Vol 14, Springer-Verlag.
7. Priss, Uta (2008b). *FcaStone - FCA file format conversion and interoperability software.* Conceptual Structures Tool Interoperability Workshop (CS-TIW).
8. 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.
9. Stahl, J.; Wille, R. (1986). *Preconcepts and set representation of contexts.* In: Gaul & Schader (eds): Classification as a tool of research.
10. 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.
11. Wille, Rudolf (1984). *Liniendiagramme hierarchischer Begriffssysteme.* Studien zur Klassifikation. Indeks Verlag.
12. Wille, Rudolf (1992). *Concept Lattices and Conceptual Knowledge Systems.* Computers Math. Applic., 23, 6-9, p 493-515.