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.

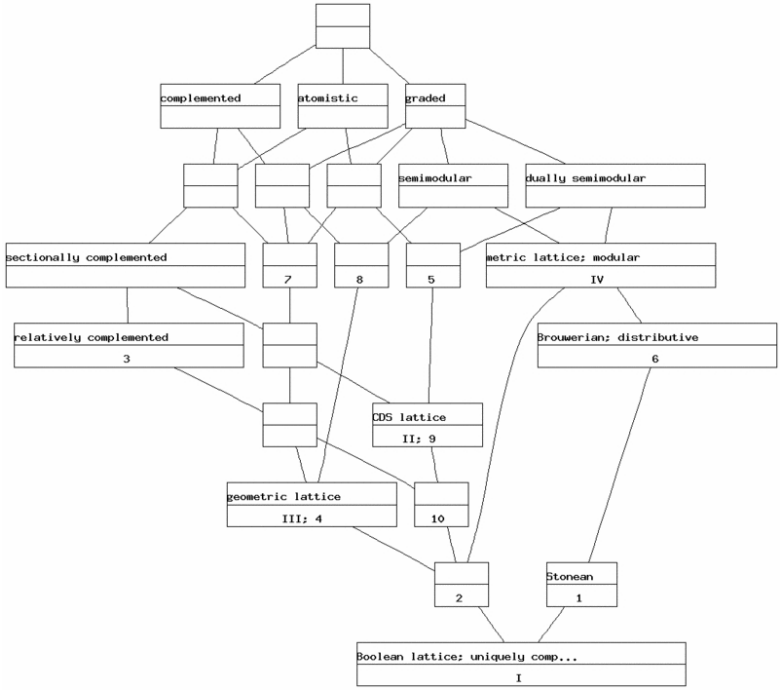

Fig. 2. An example of a layout with Graphviz (using data from Wille (1992)).

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.

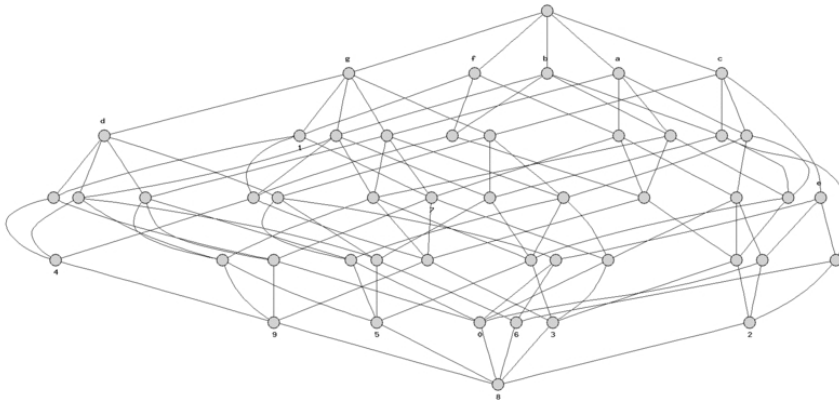

Fig. 3. A second example of a layout with Graphviz (using data from Stahl & Wille (1986)).

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`

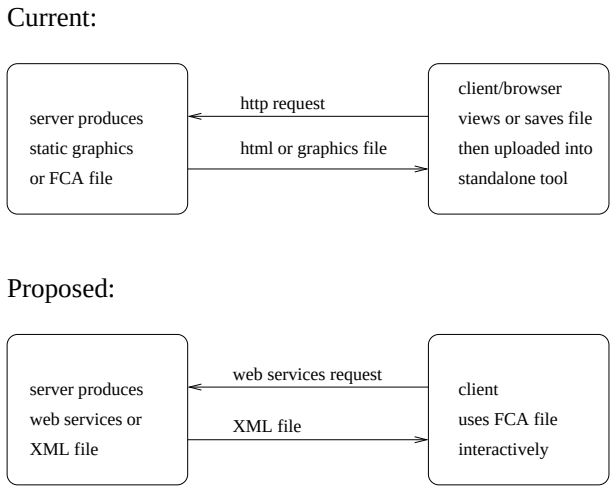

Fig. 4. Using FCA applications over the web

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.