

Tool Support for FCA

Thomas Tilley

School of Information Technology and Electrical Engineering
University of Queensland, Brisbane, Australia
tilley@itee.uq.edu.au

Abstract. Over the last two decades a number of tools have been developed to support the application of Formal Concept Analysis (FCA) to a wide variety of domains. This paper presents an overview of tool support for FCA.

1 Introduction

Over the last two decades a number of tools have been developed to support the application of Formal Concept Analysis (FCA) to a wide variety of domains. These tools range from the early DOS-based implementations of Duquenne's GLAD tool to Java-based tools currently under active development like ConExp and ToscanaJ. Both commercial and open-source software appears in the list which also includes general-purpose and application specific tools.

The next section of the paper introduces the general purpose tools. Application specific tools are then discussed in Section 3 before Section 4 concludes the paper.

2 FCA Tools

Duquenne's tool for General Lattice Analysis and Design (GLAD) is possibly the earliest software tool that facilitates the analysis of formal concept lattices [1]. GLAD is a DOS-based program written in FORTRAN that has been under development since 1983. The tool facilitates the editing, drawing, modifying, decomposing and approximation of finite lattices in general and is not restricted to the analysis of concept lattices. The lattices to be analysed can be derived from abstract mathematics or applied statistics using techniques like Analysis of Variance. Single-valued data can also be analysed by exploiting the classic correspondence between lattices and binary relations identified by Birkhoff.

GLAD contains a large number of features, many of which are undocumented and it also supports "scenarios" which represent a form of macro. These scenarios can be used regenerate and manipulate a lattice by recalling the list of commands used to construct it. Diagrams can also be output directly from GLAD in the Hewlett Packard Graphics Language (HPGL)¹ — a vector based language designed for plotters.

ConImp (**C**ontexts and **I**mplications) is another DOS-based tool implemented by Peter Burmeister [2] who started development in 1986 on an Apple II computer. While ConImp is purely text based and provides no graphical output for lattices it also supports

¹ See <http://www.piclist.com/techref/language/hpgl.htm>

a wide range of features for manipulating contexts and provides concept listings which can be used for drawing line diagrams by hand.

The *Duquenne-Guigues-base* represents a canonical base of valid implications for a given context and this is computed and used extensively within ConImp. Interactive attribute exploration is supported which can be used to derive both the Duquenne-Guigues-base and a typical set of objects. In addition, a three-valued logic that allows for *true*, *false* and *unknown* values can also be used.

While ConImp supports single-valued contexts another tool called *MBA* (possibly from the German for “Many-valued FCA”: “*Mehrwertige BegriffsAnalyse*”) can be used to scale and pre-process many-valued contexts². In addition, contexts can be exported from ConImp in the so called “Burmeister Format” (“*.CXT*”) and rendered using another DOS-based tool called *Diagram* [3]. The use of separate tools for the tasks of data-preparation, context creation, and line diagram rendering (*Diagram*) is also reflected in the classic FCA tools Anaconda and TOSCANA.

Anaconda and TOSCANA (**TO**ols of **C**oncept **A**NALysis) are tools used for building conceptual knowledge systems on top of data stored in relational databases. A conceptual system engineer uses knowledge from a domain expert to create queries in the form of conceptual scales using a *conceptual system editor*. These scales essentially capture the expert’s knowledge and the information is stored in a *conceptual system file*. A user can then exploit the conceptual scales to retrieve or analyse data from the database using a *conceptual schema browser* [4]. In traditional TOSCANA systems Anaconda is the conceptual system editor, TOSCANA is the conceptual system browser, and the data is stored in a Microsoft Access database.

Anaconda is a tool for the creation and editing of contexts, line-diagrams and scales. The context, scales and line-diagrams are saved in a conceptual schema file which is then used by TOSCANA to analyse the data in the database. While TOSCANA users cannot create new scales, the scales can be composed to produce nested line diagrams. There are three versions of TOSCANA based on Vogt’s C++ FCA libraries [5,6] and more recently a Java-based version — ToscanaJ.

ToscanaJ [4] is a platform-independent implementation of TOSCANA that supports nested line diagrams, zooming and ideal/filter highlighting. Originally part of the Tockit project³ — an open source effort to produce a framework for conceptual knowledge processing in Java — ToscanaJ is now a separate project⁴.

In the context of the workflow described earlier, ToscanaJ represents the conceptual schema browser and the conceptual system editor role is filled by two tools — *Siena* and *Elba*. The two tools can be seen as Anaconda replacements that are both used for preparing contexts and scales, however, each represents a different workflow. Elba is used for building ToscanaJ systems on top of relational databases while Siena allows contexts to be defined using a simple point and click interface.

ToscanaJ can be used to analyse data in relational databases via ODBC (Open Database Connectivity)/JDBC (Java Database Connectivity) or, alternatively, an embedded relational database within ToscanaJ can be used. Line diagrams can also be

² See <http://www.mathematik.tu-darmstadt.de/ags/ag1/Software/>

³ See <http://tockit.sourceforge.net/>

⁴ See <http://toscanaj.sourceforge.net/>

exported in a variety of raster and vector-based formats including Portable Network Graphics (PNG), Joint Photographic Expert Group (JPEG), Encapsulated PostScript (EPS), Portable Document Format (PDF), and Scalable Vector Graphics (SVG).

An XML-based conceptual schema file (.CSX) is used to store the context and scales produced by Siena and Elba. In addition, an extensible viewer interface allows custom views to be defined as well as allowing external data viewers to be specified. This feature is exploited by the formal specification browser SpecTrE described in Section 3.2.

The line diagrams in Siena and Elba use an n -dimensional layout algorithm in which each attribute in the purified context is assigned to a vector [7]. The layout is then projected onto the Cartesian plane using standard parallel projection and the approach is based on the algorithm used in Cernato.

Cernato is a commercial FCA tool developed by Navicon⁵ that combines some of the features of Anaconda and TOSCANA into a single tool. Users are presented with a familiar spreadsheet-like interface for creating contexts and data can be imported and exported in Comma Separated Value (CSV) format which facilitates the analysis of data from genuine spreadsheet applications.

Line diagrams are constructed incrementally in Cernato and the layout is animated by default. Zooming and the construction of scales, which are known as “views” in Cernato, are also supported, however, nested line-diagrams are not. In addition to the CSV import/export facility a custom XML format can also be used. Furthermore, line diagrams can be exported in a number of raster-based image formats, contexts can be saved as HTML-tables and Cernato is also able to export complete TOSCANA systems.

ConExp (**C**oncept **E**xplorer)⁶ is another Java-based, open-source FCA project. Like Cernato, ConExp combines context creation and visualisation into a single tool. While ConExp does not support database connectivity, contexts can be imported and exported in ConImp’s ‘.CXT’ format. A number of lattice layout algorithms can be selected including chain decomposition and spring-force algorithms. The line diagrams also support various forms of highlighting including ideal, filter, neighbour and single concept highlighting and can be exported in JPEG or GIF format.

ConExp currently implements the largest set of operations from Ganter and Wille’s FCA book [8] including calculation of association rules and the Duquenne-Guigues-base of implications. Interactive attribute exploration is also supported and the context can display the arrow relations $g \nearrow m$ and $g \searrow m$.

GaLicia⁷, the Galois Lattice Interactive Constructor is another Java-based FCA tool that provides both context creation and visualisation facilities [9]. GaLicia’s heritage lies in a series of incremental data mining algorithms originally entitled the **G**ALOIS **L**ATTICE-BASED **I**NCREMENTAL **C**LOSED **I**TEMSET **A**PPROACH and also a *trie* data-structure based version called GALICIA-T. These incremental algorithms were used for mining association rules in transaction databases [10, 11] and form the basis for the incremental construction of lattices in GaLicia.

Both single and many-valued contexts can be analysed in GaLicia. In addition, binary relationships between objects can also be described via a context and stored

⁵ See <http://www.navicon.de>

⁶ See <http://sourceforge.net/projects/conexp>

⁷ See <http://www.iro.umontreal.ca/~valtchev/galicia/>

using GaLicia’s Relational Context Family [12]. A number of different lattice and Galois sub-hierarchy construction algorithms are also supported.

GaLicia provides two lattice layout mechanisms including a “magnetic” spring-force algorithm. The lattices can also be viewed using a novel, rotating 3-Dimensional view. GaLicia can be run as a stand-alone application or it can be used via the World Wide Web as a Java applet running in a Web browser.

3 Other Tools

Having introduced a number of generic FCA tools in the preceding sections, this section provides a brief overview of some application specific FCA tools. These tools can be broadly classified into two main groups: the *modular* and the *monolithic*. Tools that rely on other programs for part or all of their functionality will be classified as “modular”. For example, a number of the application specific tools make use of pre-existing graph drawing applications for lattice layout. In contrast the term “monolithic” will be used to describe those tools which do not rely on other applications to function. This does not, however, exclude the use of pre-existing libraries within the tools code. Additionally, the term should not infer that a tool is poorly engineered or necessarily massive, but rather that the tool has been constructed from scratch.

3.1 Monolithic Approaches

Düwel’s *BASE* [13] tool supports the identification of class candidates from use-cases. The name is taken from the German “*ein Begriffsbasiertes Analyseverfahren für die Software-Entwicklung*” which translates into English as “concept-based analysis during software development”.

Taran and Tkachev’s [14] tool SIZID is designed to support the analysis of sociological and psychological data. SIZID can handle multi-valued contexts and the calculation of implications.

Cole and Eklund have implemented a number of FCA based document management and information retrieval tools. *Warp-9 FCA* [15] is a tool for managing a collection of medical discharge documents that is implemented using the scripting and extension language Tcl/Tk⁸. A medical ontology to index documents and the visualisation supports folding line diagrams. The ideas in *Warp-9 FCA* are further refined and applied to the analysis of email in the tool CEM — the Conceptual Email Manager [16]. More recently a commercial descendant of CEM known as *Mail-Sleuth* has also been released⁹.

In Lindig and Snelting’s [17] paper on the structure of legacy code a footnote mentions an inference based software environment called NORA which was used to produce the analyses described in the paper. NORA stands for “NO Real Acronym”. While no details of the NORA environment are presented in the paper, both Snelting and Lindig have produced other tools to support the analysis of software using FCA. Snelting and Streckenbach’s *KABA* is a Java-based tool that implements the analysis earlier described

⁸ See <http://www.tcl.tk>

⁹ See <http://www.mail-sleuth.com>

by Snelting and Tip [18]. The name KABA is taken from the German “KlassenAnalyse mit BegriffsAnalyse” which translates as “class analysis via concept analysis” in English. Apparently “KABA” is also the name of a popular chocolate drink in Germany.

KABA combines concept lattices with dataflow analysis, and type inference. In particular the prototype tool supports the visualisation of horizontal-decompositions in Java classes and a 15 KLOC (“thousand Lines Of Code”) example is reported.

While another prototype tool that implements Lindig’s component retrieval ideas could be considered monolithic [19], there have been a number of modular tools developed using Lindig’s *concepts*¹⁰ framework.

3.2 Modular Approaches

Concepts is an updated version of Lindig’s *TkConcept* tool¹¹ implemented in Tcl/Tk. *TkConcept* is included here as an example of a modular tool because it makes use of a graph layout application called *Graphplace*¹² to draw lattice diagrams. *TkConcept* was intended as a framework for concept analysis applications that provides basic abstractions so that software designers can focus on the implementation of domain specific parts of an application.

Van Deursen and Kuipers [20] used Lindig’s *concepts* tool in conjunction with *Graphplace* in the analysis of a 100 KLOC COBOL program. A relational database was used to derive information about the application using a COBOL lexical analysis tool. The data was then extracted and formatted for analysis with *concepts*.

The *ConceptRefinery* tool described by Kuipers and Moonen [21] also uses *concepts* in conjunction with a COBOL parser and a relational database. *Concept refinery* is implemented using Tcl/Tk and a version of the *dot* directed graph drawing tool was used for visualisation. *Dot* is part of the *GraphViz* graph visualisation package¹³.

GraphViz and *concepts* are also used to render lattice diagrams in Eisenbarth et al.’s *Bauhaus* tool [22]. *Bauhaus* makes use of a number of components including the *gcc* compiler and *gprof* profiler which are glued together using Perl. In addition to their earlier work identifying features in web-browser code, Eisenbarth et al. have also used their tool to analyse a 1,200 KLOC production system.

The *Cable* tool implemented by Ammons et al. makes use of FCA to aid in the debugging of temporal specifications [23]. This visualisations presented to *Cable* users are implemented using the *Dotty* and *Grappa* graph visualisation tools which are also part of *GraphViz*.

Janssen’s *JaLaBA* tool¹⁴ is a novel on-line Java Lattice Building Application that uses Freese’s *LatDraw*¹⁵ program for lattice layout. *LatDraw* makes use of a 3-dimensional spring and force layout algorithm which produces line diagrams similar to *GaLicia* and *ConExp*.

¹⁰ See <http://www.eecs.harvard.edu/~lindig/src/concepts.html>

¹¹ See <http://sensei.ieec.uned.es/manuales/tkconcept/welcome.html>

¹² Available from

<ftp://ftp.dcs.warwick.ac.uk/people/Martyn.Amos/packages/graphplace/graphplace.tar.gz>

¹³ See <http://www.research.att.com/sw/tools/graphviz/>

¹⁴ See <http://juffer.xs4all.nl/cgi-bin/jalaba/JaLaBA.pl>

¹⁵ See <http://www.math.hawaii.edu/~ralph/LatDraw/>

The round-trip engineering work of Bojic and Velasevic [24] also makes use of a modular tool implemented using ConImp. By adapting the output from the Microsoft Visual C++ profiler, ConImp was able to analyse their data which was then used to update a UML model using the Rational Rose design tool¹⁶.

Richards and Boettger et al.'s RECOCASE tool [25] is also comprised of a number of other applications. RECOCASE uses the Link Grammar Parser¹⁷ to parse use-cases and ExtrAns¹⁸ is used to generate “flat logical forms” which are then analysed using FCA.

A paper by Tonella [26] describes the CANTO tool (**C**ode and **A**rchitecture **a**nalysis **T**ool) [27] which has a modular architecture composed of several subsystems. CANTO consists of a front-end for analysing C code, an architecture recovery tool, a flow analysis tool and a customised editor. The components communicate either via sockets or files and apart from the flow analysis tool each of the components is an external application. Visualisations produced by the architecture recovery tool are created using PROVIS — yet another graph drawing application based on Doty.

Another FCA framework implemented by Arévalo [28,29] and Buchli [30] is ConAn (**C**oncept **A**nalysis) [30]. ConAn is implemented in Smalltalk and consists of a number of tools for the creation and analysis of formal contexts. A tool called *ConAn PaDi* (**C**onAn **P**attern **D**isplayer) built using the ConAn framework is used for analysing patterns in data from the *Moose* Smalltalk re-engineering environment¹⁹. Beyond software engineering applications ConAn also represents a generic and extensible framework. Users can provide objects and attributes (known respectively as *elements* and *properties*) as labels in a table or custom Smalltalk objects can be implemented to represent the elements and properties used by ConAn.

The implementation of the authors own tool [31], SpecTrE (the **S**pecification **T**ransformation **E**ngine), mirrors the modular approach taken by Van Deursen and Kuipers. SpecTrE is used for visualising and navigating formal specification documents written in Z [32]. The tool makes use of a parser to extract information from specifications which is stored in a database. The information is then formatted for analysis and visualisation using ToscanaJ and ZML [33] — an XML-based Z representation.

4 Conclusion

This paper has presented an overview of FCA tools ranging from the early DOS-based implementations through to Java-based tools currently under active development. Section 2 introduced the general purpose tools which include both commercial and open-source software. Section 3 then discussed application specific tools. While the modular tools demonstrate the reuse of common components for both the analysis of concepts and visualisation, the diversity represents the applicability of FCA itself to wide-range of problems.

¹⁶ See <http://www.rational.com/products/rose/>

¹⁷ See <http://bobo.link.cs.cmu.edu/link>

¹⁸ See <http://www.ifi.unizh.ch/cl/extrants/overview.html>

¹⁹ See <http://www.iam.unibe.ch/~scg/Research/Moose/>

References

1. Duquenne, V., Chabert, C., Cherfouh, A., Delabar, J.M., Doyen, A.L., Pickering, D.: Structuration of phenotypes/genotypes through galois lattices and implications. In Nguifo, E., Liquière, M., Duquenne, V., eds.: CLKDD'01: Concept Lattices-based Theory, Methods and Tools for Knowledge Discovery in Databases. Volume 42., CEUR (2001) 21–32
2. Burmeister, P.: Formal concept analysis with ConImp: Introduction to the basic features. Technical report, TU-Darmstadt, Darmstadt, Germany (1996)
3. Hereth, J.: DOS programs of the Darmstadt research group on formal concept analysis (1999)
4. Becker, P., Correia, J.H.: The ToscanaJ suite for implementing conceptual info. systems. In Stumme, G., ed.: Proceedings of the First Int'l Conf. on Formal Concept Analysis - ICFCA'03, Springer-Verlag (2003) to appear.
5. Vogt, F., Wille, R.: TOSCANA - a graphical tool for analyzing and exploring data. In Tamassia, R., Tollis, I., eds.: Proceedings of the DIMACS Int'l Workshop on Graph Drawing (GD'94). Lecture Notes in Comp. Sci. 894, Berlin-Heidelberg, Springer-Verlag (1995) 226–233
6. Groh, B.: A Contextual-Logic Framework based on Relational Power Context Families. PhD thesis, Griffith Uni., School of Info. and Communication Tech. (2002)
7. Becker, P.: Multi-dimensional representations of conceptual hierarchies. In: Conceptual Structures—Extracting and Representing Semantics, Contributions to ICCS 2001. (2001) 145–158
8. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer-Verlag, Berlin (1999)
9. Valtchev, P., Gosser, D., Roume, C., Hacene, M.: Galicia: an open platform for lattices. In Ganter, B., de Moor, A., eds.: Using Conceptual Structures: Contributions to ICCS 2003, Shaker Verlag (2003) 241–254
10. Valtchev, P., Missaoui, R., Godin, R., Meridji, M.: Generating frequent itemsets incrementally: two novel approaches based on galois lattice theory. Journal of Experimental and Theoretical Artificial Intelligence (JETAI) : Special Issue on Concept Lattice-based theory, methods and tools for Knowledge Discovery in Databases **14** (2002) 115–142
11. Valtchev, P., Missaoui, R., Godin, R.: A framework for incremental generation of frequent closed itemsets. In: Proceedings of the Workshop on Discrete Mathematics and Data Mining (DM&DM2002), Arlington, VA (2002)
12. Huchard, M., Roume, C., Valtchev, P.: When concepts point at other concepts: the case of UML diagram reconstruction. In: Advances in Formal Concept Analysis for Knowledge Discovery in Databases, FCAKDD 2002. (2002) 32–43
13. Düwel, S.: BASE - ein begriffsbasiertes Analyseverfahren für die Software-Entwicklung. PhD thesis, Philipps-Universität, Marburg (2000)
14. Taran, T., Tkachev, O.: Applications of formal concept analysis in humane studies. In Ganter, B., de Moor, A., eds.: Using Conceptual Structures: Contributions to ICCS 2003, Shaker Verlag (2003) 271–274
15. Cole, R., Eklund, P.: Scalability in formal concept analysis. Computational Intelligence, **15** (1999) 11–27
16. R. Cole, P.E., Stumme, G.: CEM – a program for visualization and discovery in email. In Zighed, D., Kormorowski, J., Zytow, J., eds.: Proceedings of PKDD 2000. LNAI 1910, Berlin, Springer-Verlag (2000) 367–374
17. Lindig, C., Snelting, G.: Assessing modular structure of legacy code based on mathematical concept analysis. In: Proceedings of the Int'l Conf. on Software Eng. (ICSE 97), Boston (1997) 349–359
18. Snelting, G., Tip, F.: Reengineering class hierarchies using concept analysis. Technical Report RC 21164(94592)24APR97, IBM T.J. Watson Research Center, IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA (1997)

19. Lindig, C.: Concept-based component retrieval. In Köhler, J., F., Giunchiglia, Green, C., Walther, C., eds.: Working Notes of the IJCAI-95 Workshop: Formal Approaches to the Reuse of Plans, Proofs, and Programs. (1995) 21–25
20. van Deursen, A., Kuipers, T.: Identifying objects using cluster and concept analysis. In: Proceedings of the 21st Int'l Conf. on Software Eng., ICSE-99, ACM (1999) 246–255
21. Kuipers, T., Moonen, L.: Types and concept analysis for legacy systems. Technical Report SEN-R0017, Centrum voor Wiskunde en Informatica (2000)
22. Eisenbarth, T., Koschke, R., Simon, D.: Locating features in source code. *IEEE Transactions on Software Eng.* **29** (2003) 195–209
23. Ammons, G., Mandelin, D., Bodik, R., Larus, J.: Debugging temporal specifications with concept analysis. In: Proceedings of the Conf. on Programming Language Design and Implementation PLDI'03, ACM (2003)
24. Bojic, D., Velasevic, D.: Reverse eng. of use case realizations in UML. In: Symposium on Applied Computing - SAC2000, ACM (2000)
25. Böttger, K., Schwitter, R., Richards, D., Aguilera, O., Mollá, D.: Reconciling use cases via controlled language and graphical models. In: INAP'2001 - Proc. of the 14th Int'l Conf. on Applications of Prolog, Japan, Uni. of Tokyo (2001) 20–22
26. Tonella, P.: Concept analysis for module restructuring. *IEEE Transactions on Software Eng.* **27** (2001) 351–363
27. Antoniol, G., Fiutem, R., Lutteri, G., Tonella, P., Zanfei, S.: Program understanding and maintenance with the CANTO environment. In: Proceedings Int'l Conf. on Software Maintenance. (1997) 72–81
28. Arévalo, G.: Understanding behavioral dependencies in class hierarchies using concept analysis. In: Proceedings of LMO 2003 (Langages et Modèles á Object), Paris (France), Hermes (2003)
29. Arévalo, G., Ducass, S., Nierstrasz, O.: Understanding classes using x-ray views. In: MASPEGHI 2003, MAnaging SPEcialization/Generalization HIERarchies (MASPEGHI) Workshop at ASE 2003, Montreal, Canada (2003) Preliminary Version.
30. Buchli, F.: Detecting software patterns using formal concept analysis. Technical Report IAM-03-010, Institut für Informatik und angewandte Mathematik, Universität Bern, Switzerland (2003)
31. Tilley, T.: Towards an FCA based tool for visualising formal specifications. In Ganter, B., de Moor, A., eds.: Using Conceptual Structures: Contributions to ICCS 2003, Shaker Verlag (2003) 227–240
32. Spivey, J.: The Z notation : a reference manual. Prentice-Hall Int'l (1989)
33. Sun, J., Dong, J., Lui, J., Wang, H.: Object-Z web environment and projections to UML. In: WWW10 - 10th Int'l World Wide Web Conf., New York, ACM (2001) 725–734