FCART: A New FCA-based System for Data Analysis and Knowledge Discovery

A.A. Neznanov, D.A. Ilvovsky, S.O. Kuznetsov

National Research University Higher School of Economics,
Pokrovskiy bd., 11, 109028, Moscow, Russia
ANeznanov@hse.ru, DIlvovsky@hse.ru, SKuznetsov@hse.ru

Abstract. We introduce a new software system called Formal Concept Analysis Research Toolbox (FCART). Our goal is to create a universal integrated environment for knowledge and data engineers. FCART is constructed upon an iterative data analysis methodology and provides a built-in set of research tools based on Formal Concept Analysis techniques for working with object-attribute data representations. The provided toolset allows for the fast integration of extensions on several levels: from internal scripts to plugins.
FCART was successfully applied in several data mining and knowledge discovery tasks. Examples of applying the system in medicine and criminal investigations are considered.

Keywords: Data Analysis, Formal Concept Analysis, Knowledge Discovery, Software.

1 Introduction

We introduce a new software system for information retrieval and knowledge discovery from various data sources (textual data, database queries). The Formal Concept Analysis Research Toolbox (FCART) was designed especially for the analysis of unstructured (textual) data. In case studies we applied FCART for analyzing data in medicine, criminalistics, and trend detection.

The core of the system supports knowledge discovery techniques, including those based on Formal Concept Analysis [1], clustering [2], multimodal clustering [2, 3], pattern structures [4, 5] and other.

2 Motivation

Currently, there are several well-known open source FCA-based tools, such as ConExp [6], Conexp-clj [7], Galicia [8], Tockit [9], ToscanaJ [10], FCAStone [11], Lattice Miner [12], OpenFCA [13], Coron [14]. These tools are Java-based, cross-platform, rather easy to use, and they do not need to be installed. However, they cannot completely satisfy the growing demands of the community. There are common drawbacks of these systems which have to be addressed: poor data preprocessing,

extensibility, and scalability, as well as non-universal character of existing software tools and their command line interface (CLI) or “old-fashioned” graphical interface (GUI) in terms of usability. There is a lack of universal integrated environment for knowledge discovery based on FCA, although some attempts were made in the Tockit project [9] (the development of ToscanaJ has been forked into a separate project from Tockit). Therefore, our main task was to create an effective software implementation of full research cycle of data analysis and knowledge discovery.

A new system should provide:

1. Universal integrated environment for knowledge and data engineers.
2. Built-in set of research tools based on FCA and multimodal clustering techniques for working with object-attribute data representation.
3. Additional tools for import/export of data and data preprocessing.
4. Extendibility of the research tools on several levels: from internal scripts to plugins.
5. Generation of rich, visually appealing reports.

3 Methodology

3.1 Goal

The goal of developing the software package FCART is to create a universal extensible integrated environment. The methodology of using this software is based on modern methods and algorithms of data analysis, technologies for manipulating big data collections, data visualization, reporting and interactive processing techniques. It allows one to obtain new knowledge from data with full process tracking and reproducibility. Some ideas of data preprocessing were inherited from the CORDIET project [15].

3.2 Fundamentals

Analytic artifacts

Some of FCA entities appear to be fundamental to information representation. In FCART we use the term “*analytic artifact*” which denotes the definition of abstract interface, describing the entity of the analytic process.

The basic artifact for FCA-based methods is that of “*formal context*”, i.e., object-attribute representation of a subject domain. Most important artifacts include “*concept lattice*” and “*formal concept*”.

All artifact instances are linked by “origination”. For example, we can generate the concept lattice from the formal context. In this case the formal concept will be an "origin artifact" for the lattice. Another example is lattice and “association rules” – the lattice is the origin of the rules. Any artifact instance is *immutable*. It means that an instance cannot be changed after creation, but can be visualized in various ways.

If we have the predefined set of artifacts in most cases we can use the term "artifact" instead of "artifact instance" without ambiguity. Collection of all artifacts in current analytic cycle forms so-called *analytical session*.

Another interesting option of our system is that of *multicontext* [16] artifact. In the most general case a multicontext can be considered as a network of contexts. In the particular case each context is assigned a specific time point, – then one can use this artifact for describing statements of a dynamic system. An example of applying this artifact is the problem of finding trends. Consider the context where objects are documents and attributes are terms. We find chains of intentionally related concepts (other variants of relations can be used). The terms included in the intersection of these concepts will form the core of the trend, the remaining terms and their number will characterize the stage in the life cycle and the popularity of this trend at a particular time point. Relations between concepts in contexts have suitable visual interpretation: sequence of diagrams with labeled links between related concepts.

Solvers

All types of artifacts are generated by *solvers*. Each solver requires one or many artifact instances of preassigned types as input and produces one artifact instance of preassigned type as output.

Having predefined types of artifacts and links (assigned by solvers) between immutable artifact instances we can check an integrity of data of particular analytical session. Without explicit user action a session cannot lose any artifact instances and links, and guarantees integrity of a session.

Visualizers

Artifact visualizer is a special solver that generates user-oriented visual representation of input artifact instance. From a technical point of view visualizer produces interactive or non-interactive window with some elements of user interface. Of course, one artifact can have different kinds of visual appearance.

Usually, visualizer is the last in a chain of solvers. But we can get a visual representation of each artifact in a session. For example, lattice browser generates a diagram of a lattice and allows a user to manipulate the diagram, but this browser does not generate new artifacts. We need to distinguish generation of new artifact and drawing of existing artifact for various purposes: working in the batch mode, increasing efficiency of long chains of solvers, benchmarking, etc.

Reports

Report is a final result of research. Every scientific environment must provide a report rich text editor with additional functionality to avoid mistakes while converting and moving multiple results with metadata to an external editor. The main feature of the editor is an automatic insertion of fully decorated artifact representation in the resulting report.

3.3 Main principles

1. Iterative process of data analysis using FCA entities and methods.
2. Separation of processes of *data querying* (from various data sources), *data preprocessing* (of locally saved immutable snapshots), *data analyzing* (in interactive visualizers of immutable analytic artifacts), and *results formalizing* (in a report editor).
3. Explicit definition of analytic artifacts and their types. It allows checking integrity of the session data and provides links between artifacts for an end-user.
4. Integrated performance estimation tools.
5. Integrated documentation of software tools and data analysis methods.

4 Software properties

4.1 Common information

At this moment we introduce the version 0.7 of FCART in the form of local Windows application. We use Microsoft and Embarcadero programming environments and different programming languages (C++, C#, Delphi, Python and other). For scripting we use Delphi Web Script and Python. Native executable (the core of the system) is compatible with Microsoft Windows 2000 and later and has not any additional dependences.

Another line of development is Web-version of system based on Microsoft .NET platform. For now architecture and some key components are ready, but we are going to focus on Web-development after finishing local version 0.9.

4.2 Architecture

FCART constructed as multicomponent application. Current version consists of the following components:

- Core component
  - multiple-document user interface of research environment with session manager,
  - snapshot profiles editor (SHPE),
  - snapshot query editor (SHQE),
  - query rules database (RDB),
  - session database (SDB),
  - report builder.
- Local XML-storage for preprocessed data.
- Internal solvers and visualizers.
- Additional plugins and scripts.

4.3 Data preprocessing in FCART

Obtaining initial artifacts
There are several ways to obtain initial artifacts.

- Load from ready data files of supported formats.
- Generate by plugin or script.
- Query from data snapshots.

Data snapshot (or snapshot) is a data table with structured and text attributes, loaded in the system by accessing external SQL, XML or JSON data sources. Snapshot is described by a *profile*. FCART provides one with a snapshot profile editor (SHPE) and local storage of snapshots with metadata.

Constructing binary contexts
Initial formal contexts can be imported from data files in standard format like CXT or CSV. The system has query language for transforming snapshot into formal context. This language describes so-called rules. Main rule types are the following.

- *Simple rule* generates one attribute from atomic fields of a snapshot.
- *Scaling rule* generates several attributes from atomic fields based on nominal or ordinal scale.
- *Text mining rule* generates one attribute from unstructured text fields.
- *Multivalued rule* generates one or many attributes from multivalued field (arrays and sets).
- *Compound rule* merges rules of all types into single rule. This rule uses standard logical operations and brackets to combine elements.

We also implement additional rule types: *Temporal rules* are used for manipulating date and time intervals and *Filters* are used for removing objects from context.

In most cases, it is not necessary to write a query from scratch. One can select some entities in rules DB and automatically generate a query. It is possible because the rule DB is aware of dependencies between rules. Separate queries or full DB of rules can be imported and exported as XML-files.

FCART uses Lucene full text search engine [17] to index the content of unstructured text fields in snapshots. The resulting index is later used to validate quickly whether the text mining or compound rule returns true or false. It is useful for dealing with dynamic data collections, including texts in natural language.

4.4 Sessions, solvers and visualizers

Session
Multiple-document interface allows one to have each solver in its own window. User can view all artifacts in the *session browser* (independent task pane) in the form of a tree. The main mode of user interaction in FCART is interactive work in various visu-

alizers. Our software manages links between artifacts and guarantees valid state of a working session in case of deleting some objects and restarting the system.

Main solvers in the current version can produce clusters; concept lattices and sublattices; association rules and implications; calculate stability indices, similarity measures for contexts and concepts. All those artifacts can be visualized and inserted into the report. The set of solvers and visualizers can be appended by plugins and scripts.

Any artifact can be exported in several formats. For example, concept lattice can be saved as graph (XGMML) or as picture (EMF, PNG, and JPG). We plan to extend the set of admissible formats on demand of future users.

Interactive visualization of concept lattice

The *concept lattice visualizer* is an example of visualizer. It can be used to browse the collection of objects with binary attributes given as a result of query to snapshot (with structured and text attributes). The user can select and deselect objects and attributes and the lattice diagram is modified according. The user can click on a concept. The screen shows in a separate window names of objects in the extent and names of attributes in the intent. Names of objects and attributes are linked with initial snapshot records and fields. If the user clicks on the name of an object or an attribute, the content of the object or attribute description is shown in a separate window according to snapshot profile.

Fig. 1 demonstrates the result of building sublattice from concept lattice. The Multiple-document interface allows us to inspect several artifacts, so a sublattice will be opened in a new window.

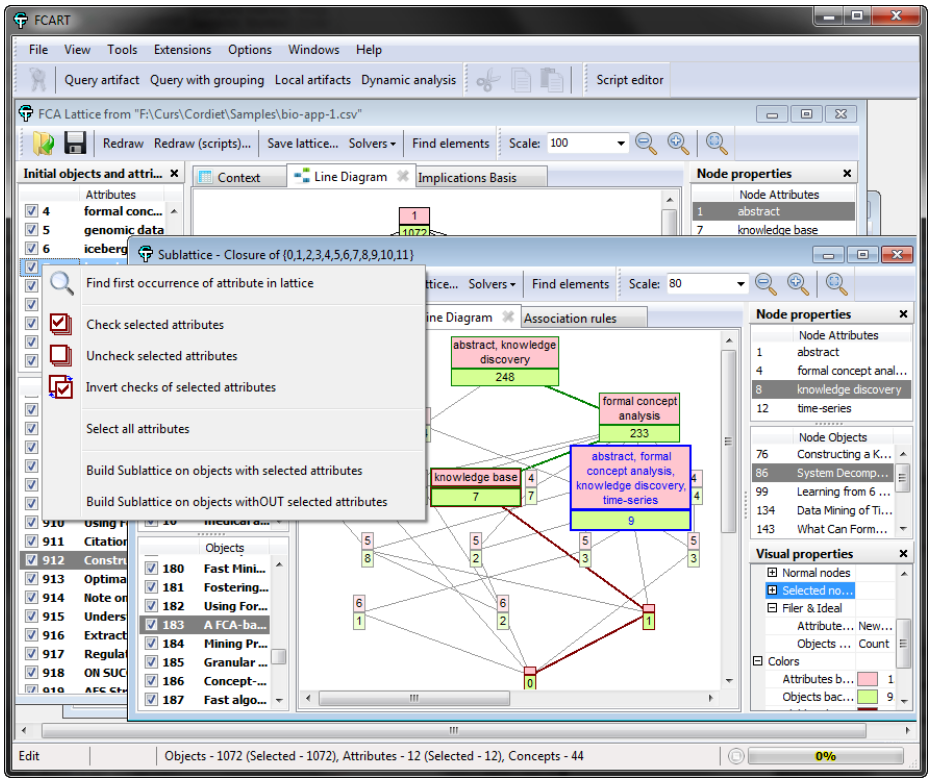

Fig. 1. Concept lattice visualizer

The user can customize settings of lattice browsing in various ways. The user can specify whether nodes corresponding to concepts show numbers of all (or only new) objects and all (or only new) attributes in extent and intent respectively, or names of all (or only new) objects and all (or only new) attributes. Separate settings can be specified for the selected concept, concepts in the order filter, and the remainder of the lattice. The visual appearance can be changed: zooming, coloring, and other tools are available.

Right clicking on the name of an attribute user can choose several options: he can build a sublattice containing only objects with selected attribute; build a sublattice containing only objects without selected attribute; or find the highest concept with selected attribute. Right clicking on the name of object allows the same actions.

Report generation

FCART supports editing several reports at the same time. A user can add any of valid artifacts from the current session to the report. Source file can be added as text with syntax highlighting (XML or other schemes); snapshot – as a table with profile definition; context – as a table or a bipartite graph; concept lattice – as an XGMML-text or a vector diagram (Fig. 2); and so on. Reports are part of the session and are stored

automatically. The final report can be copied to the clipboard with full content and formatting. Also it can be saved to a file in RTF or HTML format.

Same report engine is used to edit and render documentation: descriptions of solvers, comments to artifacts, and other.

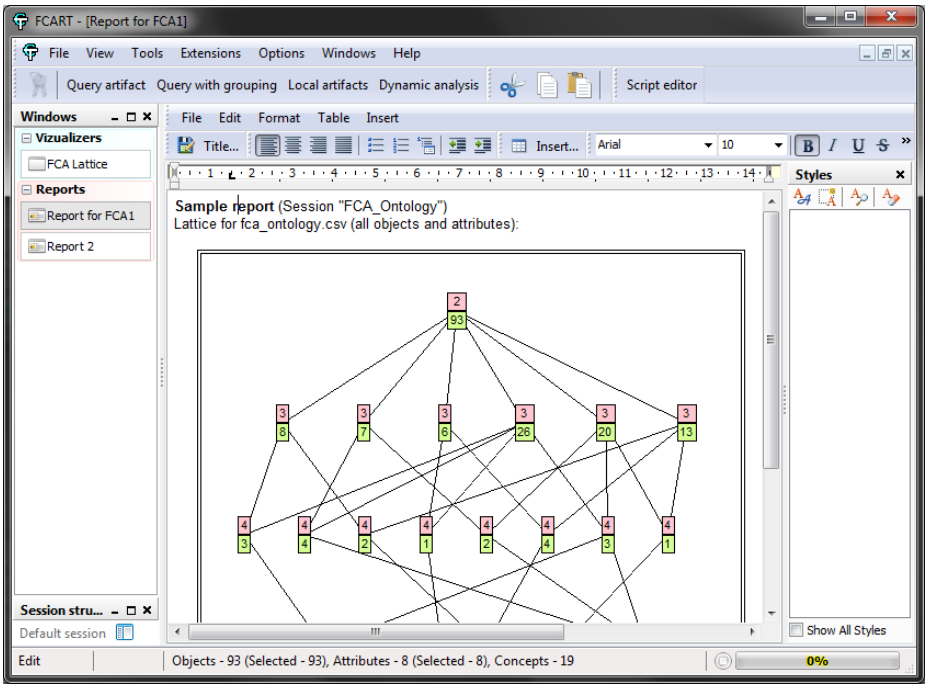

Fig. 2. Report editor with inserted lattice diagram

4.5 Extensibility

Scripts

Scripts (macros) are small internal programs, written in Delphi Web Script [18] (already implemented) or Python [19] (implementing now). In the current version following tasks can be automated by scripts:

- Generating artifacts (for example, building contexts on the fly, randomizing and generating of test samples). Of course, the user can generate only artifacts of predefined types.
- Formatting reports.
- Drawing lattice (layouting).
- Calculating similarity measures of artifacts of same types.

The system provides the script editor with syntax highlighting and debugging. The set of possible tasks will be extended in the next versions of FCART.

Plugins
The system can be extended by special modules called plugins using low-level API. The plugin's API is designed to reach maximum performance. In the current version we tested generators of artifacts.

5 Comparison with existing systems

The study of big analysis software like IBM i2 Analyst's Notebook or QSR NVivo shows that this software do not have FCA tools and have a completely different methodology of data analysis as compared to FCA software systems. So we need to compare functionality of the system with well-known tools for building and visualizing FCA artifacts (Table 1). Some criteria for comparison:

- Basic functionality: support for contexts editing, formal concept lattice generation and drawing, sublattice construction, association rules generation and other.
- Performance of basic operations and scalability.
- Rich set of supported formats.
- Data preprocessing capabilities.
- Changing visualization schemes.
- Reporting capabilities.
- Ease of extensibility.

All of the tools mentioned in Table 1 have unique features. For example, Concept Explorer was an important milestone in the development of FCA software tools. It has interesting modes of visualization of a lattice and good default layout. Galicia introduces the generic MultiFCA approach to deal with a set of contexts. ToscanaJ can visualize nested lattices and involves an editor of conceptual schemas on relational databases. FcaStone was primarily intended for file format conversion and other low level operations. Unfortunately, most of useful tools for end-user did not have official updates starting from 2006. Last version of Coron was released in 2010. Only two actively developed projects can be noted: ToscanaJ and Conexp-clj.

Table 1. Some accessible FCA software tools

Program title	Authors	Web-site
Concept Explorer (ConExp)	S.A. Evtushenko et al [6]	conexp.sourceforge.net
Galicia	P. Valtchev et al [8]	www.iro.umontreal.ca/~galicia
ToscanaJ (with Siena and Elba)	University of Queensland, Technical University of Darmstadt [10]	toscanaj.sourceforge.net
FcaStone	U. Priss et al [11]	fcastone.sourceforge.net
Lattice Miner	Boumedjout Lahcen [12]	lattice-miner.sourceforge.net
Conexp-clj	TU-Dresden, Daniel Borchman	daniel.kxpq.de/math/conexp-clj
OpenFCA	P. Borza, O. Sabou, at al [13]	code.google.com/p/openfca

(Conflexplore)		
Lattice navigator	M. Radvansky, V. Sklenar	www.fca.radvansky.net
Coron	Szathmary, L., Kaytoue, M., Marcuola, F., Napoli, A. [14]	coron.loria.fr

The common problem for these tools is low limits of size of interactively analyzed artifacts (for example, lattices with more than 8000 concepts can hardly be operated and visualized on modern hardware). This is mainly due to the use Java (or other high-level languages) and cross-platform GUI.

Let's look at an example of scalability. Consider real context (707 objects and 257 attributes) and generate concept lattice (10568 concepts) in different software[1].

ConExp generated concepts, spent 90.0 MB of memory, and *could not* produce layout the lattice (Fig. 3).

ToscanaJ Siena generated concepts, spent 203.2 MB of memory, produced layout (Fig. 4), but worked *very slowly* even when viewing initial context.

FCART generated concepts, spent 23.5 MB of memory, produced layout (Fig. 5), and provided normal interactive manipulations with context and concepts.

The current version of FCART can construct and manipulate big lattices (more than 16000 concepts), also in interactive mode. After all planned optimizations in version 0.8 we will present deep comparison of implementations of all basic FCA algorithms in the form of compiled components and scripts (the system has built-in tools for benchmarking) on synthetic tests and real data.

FCART is built on top of modern platform, provides powerful preprocessing tools, rich reporting capability, and two levels of extensibility.

[1] All tests were conducted on a computer with Intel Core i7-3770 3,4 GHz CPU, 16 GB of RAM, Microsoft Windows 7 Professional x64 (system info is tracked by Process Explorer).

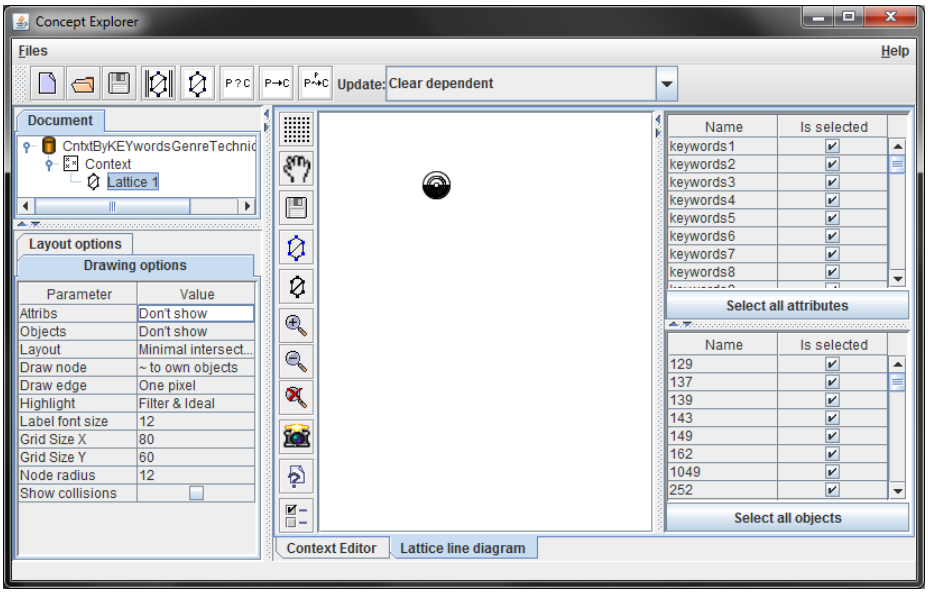

Fig. 3. Sample lattice layout in ConExp

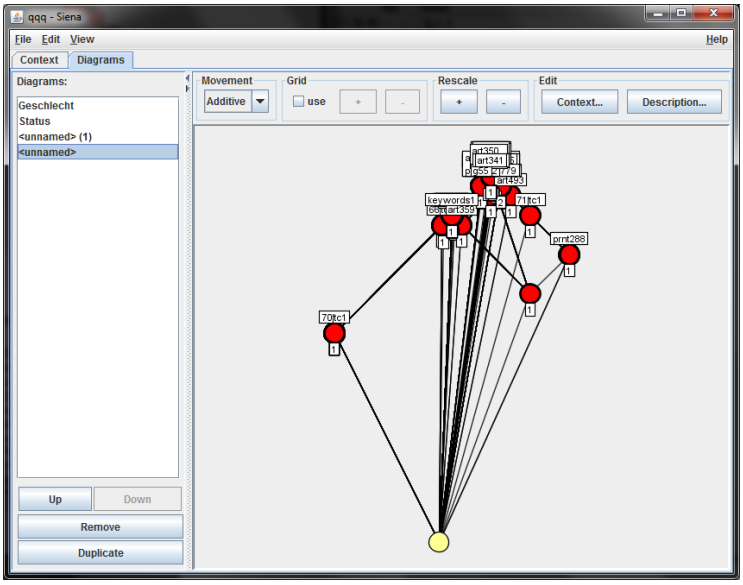

Fig. 4. Sample lattice layout in ToscanaJ Siena

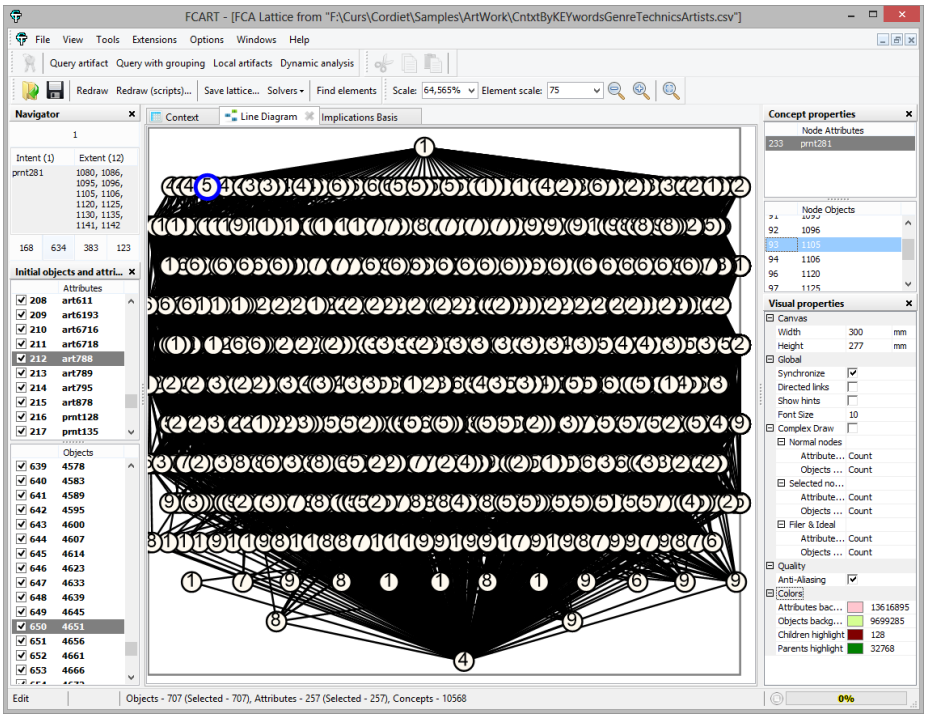

Fig. 5. Sample lattice layout in FCART (simple drawing scheme is used)

6 Conclusion and future work

FCART is a powerful environment being in active developing state. The next major release of the local version 0.8 is planned for March 2013. Then this system will be freely available to the FCA community.

We assume to improve methodology, extend the set of solvers, optimize some algorithms, and use proposed system in different knowledge discovery tasks. We already test new solvers based on concept stability [20] and similarity [10]. Biclustering techniques [2, 22] are also being actively tested; we are going to extend our platform to triadic concept analysis and noise-robust triclustering methods [3].

Acknowledgements

The work of the authors on the project “Mathematical Models, Algorithms, and Software Tools for Intelligent Analysis of Structural and Textual Data” was supported by the Basic Research Program of the National Research University Higher School of Economics.

References

1. Ganter, B., Wille R. Formal Concept Analysis: Mathematical Foundations, Springer, 1999.
2. Mirkin, B. Mathematical Classification and Clustering, Springer, 1996.
3. Ignatov, D.I., Kuznetsov, S.O., Magizov, R.A., Zhukov, L.E. From Triconcepts to Triclusters. Proc. of 13th International Conference on rough sets, fuzzy sets, data mining and granular computing (RSFDGrC-2011), LNCS/LNAI Volume 6743/2011, Springer (2011), pp. 257-264.
4. Ganter, B., Kuznetsov, S.O. Pattern Structures and Their Projections. Proc. of 9th International Conference on Conceptual Structures (ICCS-2001), 2001, pp. 129-142.
5. Kuznetsov, S.O. Pattern Structures for Analyzing Complex Data. Proc. of 12th International conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing, Conference (RSFDGrC-2009), 2009, pp. 33-44.
6. Yevtushenko, S.A. System of data analysis "Concept Explorer". (In Russian). Proceedings of the 7th national conference on Artificial Intelligence KII-2000, p. 127-134, Russia, 2000.
7. Conexp-clj (http://daniel.kxpq.de/math/conexp-clj/)
8. Valtchev, P., Grosser, D., Roume, C. Mohamed Rouane Hacene. GALICIA: an open platform for lattices, in Using Conceptual Structures: Contributions to the 11th Intl. Conference on Conceptual Structures (ICCS'03), pp. 241-254, Shaker Verlag, 2003.
9. Tockit: Framework for Conceptual Knowledge Processing (http://www.tockit.org)
10. Becker, P., Hereth, J., Stumme, G. ToscanaJ: An Open Source Tool for Qualitative Data Analysis, Proc. Workshop FCAKDD of the 15th European Conference on Artificial Intelligence (ECAI 2002). Lyon, France, 2002.
11. Priss, U. FcaStone - FCA file format conversion and interoperability software, Conceptual Structures Tool Interoperability Workshop (CS-TIW), 2008.
12. Lahcen, B., Kwuida, L. Lattice Miner: A Tool for Concept Lattice Construction and Exploration. In Suplementary Proceeding of ICFCA'10, 2010.
13. Borza, P.V., Sabou, O., Sacarea, C. OpenFCA, an open source formal concept analysis toolbox. Proc. of IEEE International Conference on Automation Quality and Testing Robotics (AQTR), 2010, pp. 1-5.
14. Kaytoue, M., Marcuola, F., Napoli, A., Szathmary, L., Villerd, J. The Coron System. Proc. of the 8th Intl. Conference on Formal Concept Analysis (ICFCA 2010), 2010, pp. 55-58.
15. Poelmans, J., Elzinga, P, Neznanov, A., Viaene, S., Kuznetsov, S., Ignatov, D., Dedene, G. Concept Relation Discovery and Innovation Enabling Technology (CORDIET) // CEUR Workshop proceedings Vol-757, Concept Discovery in Unstructured Data, 2011.
16. Wille, R.: Conceptual structures of multicontexts. In Eklund, P., Ellis, G., Mann, G., eds.: Conceptual Structures: Knowledge Representation as Interlingua. Volume 1115 of LNAI, Springer (1996), pp. 23–29.
17. Apache Lucene (http://lucene.apache.org)
18. Grange, E. DelphiWebScript Project (http://delphitools.info/dwscript)
19. Python Programming Language – Official Website (http://www.python.org)
20. Kuznetsov, S.O. On Stability of a Formal Concept // Annals of Mathematics and Artificial Intelligence, Vol. 49, 2007, pp.101-115.
21. Klimushkin, M.A., Obiedkov, S., Roth C. Approaches to the Selection of Relevant Concepts in the Case of Noisy Data // 8th International Conference on Formal Concept Analysis (ICFCA 2010), 2010, pp. 255-266.

22. Ignatov, D.I., Kuznetsov, S.O., Poelmans, J. Concept-Based Biclustering for Internet Advertisement. Proc. of 12th IEEE International Conference on Data Mining Workshops, 10 December 2012, Brussels, Belgium, pp. 123-130.