# Knowledge discovery through creating formal contexts

ANDREWS, Simon and ORPHANIDES, Constantinos



## Published version

ANDREWS, Simon and ORPHANIDES, Constantinos (2010). Knowledge discovery through creating formal contexts. In: XHAFA, F, ABRAHAM, A, DEMETRIADIS, S and CABALLE, S, (eds.) First international workshop on computational intelligence in networks and systems (CINS 2010). IEEE Computer Society, 455-460.

## Repository use policy

# Knowledge Discovery through Creating Formal Contexts

Simon Andrews* and Constantinos Orphanides†
*Conceptual Structures Research Group, Communication and Computing Research Centre*
*Faculty of Arts, Computing, Engineering and Sciences, Sheffield Hallam University, Sheffield, UK*
*s.andrews@shu.ac.uk †corphani@my.shu.ac.uk

***Abstract*—Knowledge discovery is important for systems that have computational intelligence in helping them learn and adapt to changing environments. By representing, in a formal way, the context in which an intelligent system operates, it is possible to discover knowledge through an emerging data technology called Formal Concept Analysis (FCA). This paper describes a tool called FcaBedrock that converts data into Formal Contexts for FCA. The paper describes how, through a process of guided automation, data preparation techniques such as attribute exclusion and value restriction allow data to be interpreted to meet the requirements of the analysis. Creating Formal Contexts using FcaBedrock is shown to be straightforward and versatile. Large data sets are easily converted into a standard FCA format.**

***Keywords*-Formal Context; Formal Concept Analysis; guided automation; visualisation; Concept Lattice**

## I. Introduction

Formal Concept Analysis (FCA) was introduced in the 1990s by Rudolf Wille and Bernhard Ganter [17] building on applied lattice and order theory developed by Birkhoff and others in the 1930s and was initially developed as a subsection of Applied Mathematics, based on the mathematisation of concepts and concepts hierarchy. The following description of FCA is that used in [4].

A *Formal Concept* is constituted by its *extension*, comprising of all objects which belong to the Concept, and its *intension*, comprising of all attributes (properties, meanings) which apply to all objects in the extension. The set of all objects and attributes together with their relation to each other form a *Formal Context*, which can be represented by a cross-table.

| Airlines | Latin America | Europe | Canada | Asia Pacific | Middle East | Africa | Mexico | Caribbean | USA |
|---|---|---|---|---|---|---|---|---|---|
| Air Canada | × | × | × | × | × | | × | × | × |
| Air New Zealand | | × | | × | | | | | × |
| Nippon Airways | | × | | × | | | | | × |
| Ansett Australia | | | | × | | | | | |
| Austrian Airlines | | × | × | × | × | × | | | × |

The cross-table above is a Formal Context representing the destinations of five airlines, where the elements on the left are formal objects (airlines) and the elements at the top are formal attributes (destinations). If an object has a specific attribute, it is indicated by placing a cross in the corresponding cell of the table. An empty cell indicates that the corresponding object does not have the corresponding attribute. In the Airlines Context, Air Canada flies to Latin America but does not fly to Africa.

A central notion of FCA is a duality called a '*Galois connection*'. This connection is often observed between two types of items that relate to each other. A Galois connection implies that "if one makes the set of one type larger, they will be related to a smaller set of the other type, and vice versa" [14]. In the airlines example, the combination of destinations, Asia Pacific, Europe and USA, are flown to by four airlines. If Middle East is added to the list of destinations, the number of airlines reduces to two.

The definition of a Formal Concept is extended by the idea of closure: the extension contains all objects that have the attributes in the intension, and the intension contains all attributes shared by the objects in the extension. In the example of the two airlines that fly to Asia Pacific, Europe, USA and the Middle East, it can be seen from the table that Canada is also flown to by the same two airlines. Adding Canada to the list of destinations completes (closes) that particular Formal Concept.

A strength of FCA is that the Galois connections between the Formal Concepts can be visualised in a *Concept Lattice* (Figure 1), which is an intuitive way of discovering hitherto undiscovered information in data and portraying the natural hierarchy of Concepts that exist in a Formal Context.

Each node in the lattice is a Formal Concept. The objects (airlines) are labeled below the nodes, while the attributes (destinations) are labeled above the nodes. Extracting information from a lattice is straightforward. In order to see which attributes are featured by an object, one begins from the node where the object is located and moves upwards. Any attributes one meets along the way are the attributes featured by that object. For example, if one heads upwards in the lattice from Air New Zealand (object), one will collect the attributes USA, Europe and Asia Pacific. This can be interpreted

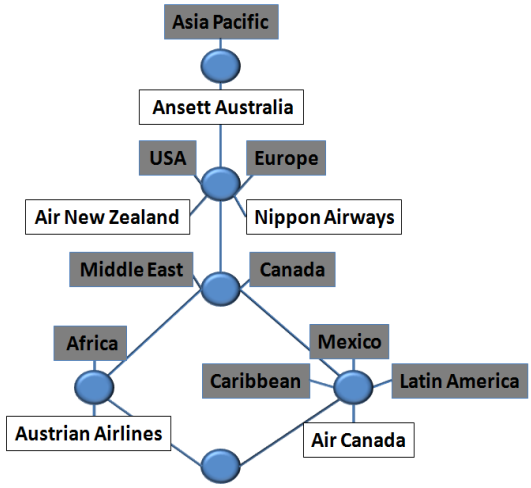



Figure 1. A Lattice corresponding to the Airlines Context

as 'Air New Zealand flies to USA, Europe and Asia Pacific'. Similarly, in order to see which objects have a specific attribute, one heads downwards in the lattice. Any objects one meets along the way are objects which feature that particular attribute. For example, heading downwards from Canada (attribute), one will collect the objects Austrian Airlines and Air Canada. This can be interpreted as 'Canada is flown to by Austrian Airlines and Air Canada'. Although the Airline Context is only a small example of FCA, it is evident that the Concept Lattice provides information that is not evident from looking at the table alone. It has been shown that Formal Concept Analysis (FCA) can be usefully applied to large sets of data [9], [10], [16] and that FCA has applications in data mining [6], [15]. Algorithms exist that are capable of processing large Formal Contexts in reasonable time [2], [11]. However, data is rarely in a form immediately suitable for FCA tools and applications. Data often exists in the form of flat-file tables of comma separated values (csv). For FCA to be carried out, these data must be converted into Formal Contexts. The existing, typically many-valued, attributes must be converted into Formal Attributes. This can be an involved and time consuming task, usually requiring a programming element. The task is, essentially, a process of discretizing and Booleanizing data; taking each many-valued attribute and converting it into as many Boolean attributes as it has values. The incorporation of many-valued attributes into FCA is well known [7], [18] and continuous data can be used for FCA by being hierarchically scaled [14] or discretized as disjoint ranges of values [10]. However, there is less work describing how these techniques can be applied in an automated, reproducible way.

Dealing with existing data sets raises a number issues: Free-text data (such as peoples' names and addresses) is not usually suitable for Booleanizing; missing values are common and must be dealt with in some way; data classes are sometimes included in the original data, but it may be more appropriate to think of theses as sub-contexts rather than attribute values; continuous data can be discretized but the method to use and the boundary values need to be decided; attributes with two values may sometimes be interpreted as being Boolean and thus converted using a single Formal Attribute; attribute values that are unused may be interpreted as not being required in the Formal Context.

The possibility of such diverse interpretation of data can, without careful documentation, lead to inconsistent and incomparable analysis. Several analyses may each use a different interpretation of the same original data set. This can also lead to problems in measuring the performance of FCA algorithms [12].

A tool to address these issues was proposed at CSTIW'09 [3] where an early prototype of what follows was demonstrated.

## II. FcaBedrock Overview

FcaBedrock is an open-source tool for automating the conversion of existing, flat-file csv data sets into Formal Contexts[1] (Figure 2)

FcaBedrock uses a process of guided automation. The user supplies the tool with appropriate meta-data for conversion, such as the names of the attributes and their values and with decisions as to what to convert and how to convert it. After reading in the original data file, this meta-data is used by FcaBedrock to create a Formal Context file in a standard format for FCA. The meta-data is stored in a separate text document called a Bedrock file. This can be used for subsequent conversions and acts as a record of the interpretation made of the data set. Bedrock files are loaded into FcaBedrock allowing the reproduction of Context files and allowing changes in the interpretation to be made. Different interpretations can be made and constraints applied to a data set allowing different analyses. Each Formal Context produced can be documented with its corresponding Bedrock file. Multiple data files with the same attributes can be converted using the same Bedrock file.

FcaBedrock currently supports two input data file formats: many column csv and three column csv. FcaBedrock outputs the Formal Context in the widely *Burmeister* FCA format.

Many column csv is a traditional flat-file format for many-valued data. Each row represents an instance (object) and each column a many-valued attribute.

[1]https://sourceforge.net/projects/fcabedrock/

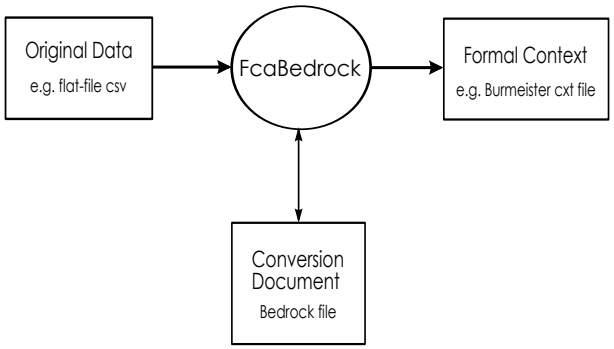



Figure 2. FcaBedrock Process

The three column format is becoming popular as a standardized format and is the approach used, for example, in the Resource Description Format (RDF) [1], [13].

FcaBedrock assumes that, when reading a three column data file, the first value is a Formal Object, the second is a many-valued attribute and the third is an attribute value. The first values are used by FcaBedrock as object names when outputting a Formal Context in the *Burmeister* format (see below). If auto-detection is used, the second values are detected as attribute names. The order in which triples appear in a data file is not significant.

The *Burmeister* Formal Context format (cxt) is a popular format in FCA. A cxt file begins with the number of objects and number of attributes and then lists the objects followed by the attributes. It then stores the body of the Formal Context as a grid using crosses for *True* values and dots for *False* values.

The sections that follow describe how FcaBedrock operates. Examples are given to illustrate how existing data can be interpreted for FCA using public data sets from the UCI Machine Learning Repository [5].

## III. Attribute Values

Figure 3 shows FcaBedrock. There are windows for the names and types of the original data attributes, whether they are to be converted or not, their categories and the corresponding category values found in the data file. All these must initially be entered by the user (unless FcaBedrock's auto-detection feature is being used - see Section VIII), although copy and paste is available and useful if the names of the categories and the category values are the same. This is often but not always the case, so FcaBedrock uses both; the category values are required for the computation but the category names appear in the Context file. The example shown is of the well-known *Mushroom* data set from UCI [5], where arbitrary letters are used in the data file to represent attribute categories. The *cap-surface* attribute, for example, has four possible categories, *fibrous*, *grooves*, *scaly* and *smooth*, represented by the values *f*, *g*, *y* and *s*, respectively, in the data file. In another UCI example, the *Adult* data set [5] has category values in the data file that each contain a leading space character. The spaces are better omitted from the category names but are required for converting the data.

## IV. Attribute Types

FcaBedrock uses three types of attribute: categorical, continuous and Boolean. The type is set by the user entering *c* for categorical, *o* for continuous or *b* for Boolean in the Type window.

### A. Categorical Attributes

The predominant attribute type is categorical. This is the typical many-valued attribute and is converted by creating one formal attribute for each of the attribute categories. In the conversion process, each row of comma separated values read in from the data file is split into a list of individual values. Each value is compared with the category values listed for the corresponding attribute in the Values (File) window. Attribute zero corresponds to the first column in the data file and so on. A match is recorded as a *True* value in the Formal Context.

In the *Mushroom* example shown in Figure 3, all but one of the attributes are categorical. The *gill-spacing* attribute, for example has three categories: *close*, *crowded*, and *distant*. FcaBedrock will create three corresponding Formal Attributes and name them by concatenating the attribute and category names, thus *gill-spacing-close*, *gill-spacing-crowded*, and *gill-spacing-distant*.

### B. Boolean Attributes

A Boolean attribute can be interpreted as a single Formal Attribute. Typically, a Boolean attribute in a data set is one that has two categories that represent *True* and *False*. In the *Mushroom* example, the Boolean type is being used for the *bruises?* attribute. The attribute has two categories, *bruises* and *no*, represented in the data file by the values *t* and *f*. For a Boolean attribute, FcaBedrock uses the first category value as the *True* value, *t* in this example. During the conversion process, it compares the corresponding value read in from the data file with the *True* value. If they match, this is recorded as a *True* value in the Formal Context. FcaBedrock names the Formal Attribute using the original attribute name, without concatenating the name of the *True* category. So, in this example, the original *Bruises?* attribute becomes a single Formal Attribute also called *Bruises?*.

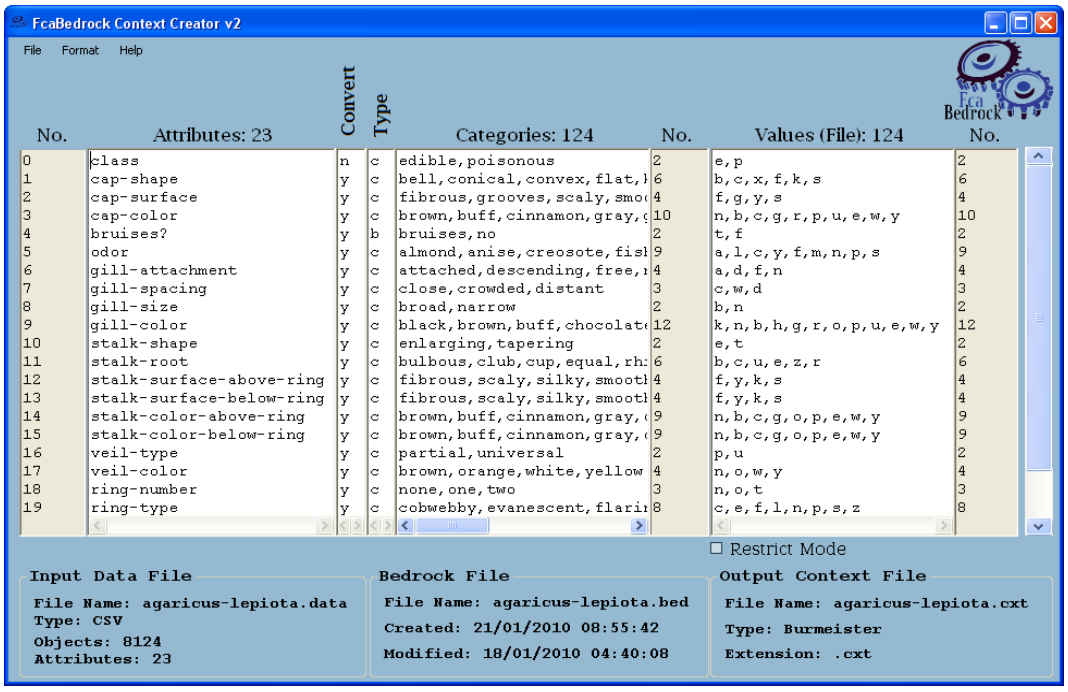



Figure 3. FcaBedrock

### C. Continuous Attributes

Continuous attributes are dealt with in FcaBedrock by discretizing the data using user defined ranges. Although hierarchical scaling (e.g. *>0*, *>10*, *>20*...) is often used in FCA to describe continuous attributes [14], this can lead to dense contexts and, consequently, a large number of large concepts. Hence, in FcaBedrock, a continuous attribute is divided into a number of ranges with disjoint boundaries.

Figure 4 shows the meta-data of the *Adult* data set in FcaBedrock. The *age* attribute is to be converted as a continuous attribute. The boundaries of the ranges are entered as comma separated values. In this case, the boundaries are *<*, *20*, *40*, *60* and *>*. FcaBedrock will apply these boundaries as the ranges *less than 20*, *20 to less than 40*, *40 to less than 60* and *greater than or equal to 60*, giving a total of four ranges (categories) for the five boundaries. The *<* and *>* boundaries are optional; literal end-boundary values will mean that any values outside the overall range will be ignored in the conversion process. The Formal Attribute names are created by concatenating values appropriately; in this case *age<20*, *age20to<40*, *age40to<60* and *age>=60*. During the conversion process, FcaBedrock compares a continuous value read in from the data file with the appropriate boundary values, assigning a *True* value to the corresponding Formal Attribute in the Context. Fractional values are allowed; *height1.567to<2.890*, for example, would be possible as a Formal Attribute.

## V. Guided Automation

### A. Exclusion of Attributes and Attribute Values

It may be necessary or desirable to exclude attributes from a Formal Context for a number of reasons: the attribute may be unsuitable for conversion (if it is a free-text attribute for example) or it may be that a particular attribute is not of interest in the analysis being undertaken. The user of FcaBedrock can decide which attributes in the data set to include in the Formal Context using the Convert window. Entering a *y* (for *yes*) will mean that the corresponding attribute will be included in the conversion. Entering an *n* (for *no*) will mean that the corresponding attribute will be excluded from the conversion; no Formal Attributes will be created for this attribute. Individual attribute categories can be excluded from the Formal Context by simply not including them in the list.

Note that these attribute and category exclusions do not exclude *objects* from the Formal Context; all the rows (objects) of the data file will be included in the conversion.

### B. Attribute Value Restriction

Sub-contexts can be created by restricting the conversion to user-specified attribute values. By specifying one or more category values of one or more

| No. | Attributes: 15 | Convert | Type | Categories: 103 | No. | Values (File): 103 | No. |
|---|---|---|---|---|---|---|---|
| 0 | age | y | o | <,20,40,60,> | 4 | <,20,40,60,> | 4 |
| 1 | workclass | y | c | Private,Self-emp-not-inc,S | 8 | Private, Self-emp-not-i | 8 |
| 2 | fnlwgt | n | c | | 0 | | 0 |
| 3 | education | y | c | Bachelors,Some-college,11t | 16 | Bachelors, Some-college, | 16 |
| 4 | education-num | n | c | | 0 | | 0 |
| 5 | marital-status | y | c | Married-civ-spouse,Divorc | 7 | Married-civ-spouse, Div | 7 |
| 6 | occupation | y | c | Tech-support,Craft-repair, | 14 | Tech-support, Craft-rep | 14 |
| 7 | relationship | y | c | Wife,Own-child,Husband,Not | 6 | Wife, Own-child, Husban | 6 |
| 8 | race | y | c | White,Asian-Pac-Islander,A | 5 | White, Asian-Pac-Island | 5 |
| 9 | sex | y | c | Female,Male | 2 | Female, Male | 2 |
| 10 | capital-gain | n | c | | 0 | | 0 |
| 11 | capital-loss | n | c | | 0 | | 0 |
| 12 | hours-per-week | n | c | | 0 | | 0 |
| 13 | native-country | y | c | United-States,Cambodia,Eng | 41 | United-States, Cambodia, | 41 |
| 14 | class | n | c | >50K,<=50K | 2 | >50K, <=50K | 2 |

Figure 4. *Adult* meta-data in FcaBedrock

attributes, the Formal Context will only contain objects which those values. For example, taking the *Mushroom* data set, a sub-context containing only poisonous mushrooms can be created simply by specifying the category value *p* for the *class* attribute. This can be repeated for any number of attributes. So, for example, a mushroom sub-context could be created containing only poisonous mushrooms that have brown, buff or cinnamon caps.

### C. Missing Values

Data sets often contain missing values. Typically these are interpreted as *False* values in FCA, although *missing* more correctly means *unknown*. A symbol such as *?* is often used in a data set to indicate a missing value and this symbol can be used in FcaBedrock if missing values are of interest in the analysis. If the missing value symbol is included as a category value of an attribute then a corresponding missing value Formal Attribute will be created.

### D. Auto-Detection of Meta-Data

If a data file is read without first loading or creating a Bedrock file, FcaBedrock can detect attribute values and create meta-data automatically. It will assume that all attributes are categorical. It will add each new value it finds in a column in the data file to a corresponding list of category values. It will assume that each attribute is to be converted. It will set the category names to the category values. If it finds more than 100 values for an attribute, FcaBedrock will list the first 100 values and indicate that the detection of values for the attribute has been truncated. This is most likely to occur when the attribute being detected is a continuous one, free-text or some form of ID value. The meta-data obtained through auto-detection can then be edited to provide the required interpretation. If a three-column format input file is being used, attribute names will also be detected.

## VI. A Mini-Adult example

The following is an example adapted from the UCI *Adult* data set. The example uses a data file (see File 1) called `mini-adult.data` with just eight instances and five attributes: *age*, *education*, *employment*, *sex* and *US-citizen*. There are two classes indicating a salary above or below $50k.

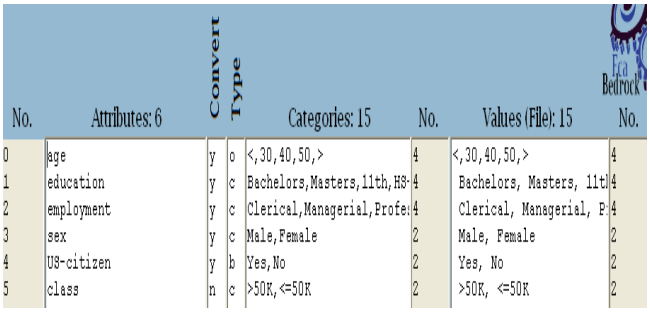
```
39, Bachelors, Clerical, Male, Yes, <=50K
50, Bachelors, Managerial, Female, Yes, <=50K
38, HS-grad, Unskilled, Male, Yes, <=50K
53, 11th, Unskilled, Male, Yes, <=50K
28, Bachelors, Professional, Female, Yes, >50K
37, Masters, Managerial, Female, No, <=50K
49, ?, Clerical, Female, No, <=50K\\
52, HS-grad, Managerial, Male, Yes, >50K
```

File 1. `mini-adult.data`

The *mini-adult* meta-data in FcaBedrock are shown in Figure 5. The output *Burmeister* context file, `mini-adult.cxt`, is shown in File 2. The Concept Lattice is shown in figure 6, visualised by inputting the cxt file to a tool called Concept Explorer (ConExp) [19]. Note that the object labels refer to the number of the adult instance (row) in the data file.

## VII. Conclusion

FcaBedrock has been used successfully to convert large data sets into Formal Contexts: The *Mushroom*



| No. | Attributes: 6 | Convert Type | | Categories: 15 | No. | Values (File): 15 | No. |
|---|---|---|---|---|---|---|---|
| 0 | age | y | o | <,30,40,50,> | 4 | <,30,40,50,> | 4 |
| 1 | education | y | c | Bachelors,Masters,11th,HS- | 4 | Bachelors, Masters, 11t | 4 |
| 2 | employment | y | c | Clerical,Managerial,Profe | 4 | Clerical, Managerial, P | 4 |
| 3 | sex | y | c | Male,Female | 2 | Male, Female | 2 |
| 4 | US-citizen | y | b | Yes,No | 2 | Yes, No | 2 |
| 5 | class | n | c | >50K,<=50K | 2 | >50K, <=50K | 2 |

Figure 5. *mini-adult* meta-data in FcaBedrock

```
B             education-Bachelors
              education-Masters
8             education-11th
15            education-HS-grad
              employment-Clerical
0             employment-Managerial
1             employment-Professional
2             employment-Unskilled
3             sex-Male
4             sex-Female
5             US-citizen
6             .X..X...X...X.X
7             ...XX....X...XX
age<30        .X.....X...XX.X
age30to<40    ...X..X....XX.X
age40to<50    X...X.....X..XX
age>=50       .X...X...X...X.
              ..X.....X....X.
              ...X...X..X.X
```

File 2. `mini-adult.cxt`, *Burmeister* context file.

data set has 8124 objects and around 125 Formal Attributes (depending on the interpretation). The *Adult* data set has 32561 objects and around 106 Formal Attributes and the *Internet Advertisements* data set has 3279 objects and around 1570 Formal Attributes.

FcaBedrock is straightforward to use: the meta-data required is usually easy to obtain (data sets usually come with descriptive documentation) and enter, particularly with the use of the repeat and auto-detection functions. There are only three types of attribute to consider and each type is straightforward to convert.

FcaBedrock is a versatile tool. It currently supports two standard input formats and two popular output formats (the use of *FcaStone* makes other FCA formats possible, too). FcaBedrock gives the user control of the conversion process and freedom to interpret data as seen fit, including the creation of sub-contexts, to suit the needs of analysis. Multiple data sets of the same type (data samples, for example) can easily be converted into Formal Contexts for comparative analysis.

The Formal Contexts created by FcaBedrock are clearly documented by corresponding Bedrock files. These files allow the reproduction of Formal Contexts. Different interpretations of the same data are possible though the editing of meta-data in FcaBedrock, with a corresponding Bedrock file being created for each interpretation, if desired.

FcaBedrock is an open source project. It may be developed further to support other input and output formats. Additional functionality may be added, such as auto-creation of ranges for continuous attributes as part of auto-detection, grouping of categories and hierarchical scaling of continuous attributes. A more ambitious goal is to incorporate categorisation of free-text values into formal attributes; capturing similarities in free-text values such as addresses from the same town or names in alphabetical order. Such enhancements to the process of Context creation will open further possibilities for FCA to be carried out on existing data sets.

## References


[1] Abadi, D.J., Marcus, A., Madden, S.R., Hollenbach, K.: SW-Store: a vertically partitioned DBMS for Semantic Web data management. In: VLDB, vol. 18, pp. 385-406. Springer-Verlag (2009)

[2] Andrews, S.: In-Close, A Fast Algorithm for Computing Formal Concepts. In: Rudolph, S., Dau, F., Kuznetsov, S.O. (eds.) ICCS 2009, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-483/paper1.pdf (2009)

[3] Andrews, S.: Data Conversion and Interoperability for FCA. In: CS-TIW 2009, pp. 42-49, http://www.kde.cs.uni-kassel.de/ws/cs-tiw2009/proceedings_final_15July.pdf (2009)

[4] Andrews, S., Orphanides, C., Polovina, S.: Visualising Computational Intelligence through converting Data into Formal Concepts. In the First International Workshop on Emerging Data Technologies for Collective Intelligence (EDTCI-2010), to be held in Fukuoka, Japan, Nov. 4-6, 2010. IEEE Conference Publishing Services, in press

[5] Asuncion, A., Newman, D.J.: UCI Machine Learning Repository http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html. Irvine, CA: University of California, School of Information and Computer Science (2007)

[6] Boulicaut, J-F., Besson, J.: Actionability and Formal Concepts: A Data Mining Perspective. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 14-31. Springer-Verlag, Berlin/Heidelberg (2008)

[7] Ganter, B., Wille, R.: Formal Concept Analysis - Mathematical Foundations. Springer-Verlag (1999)

[8] Goethals, B., Zaki, M.: Advances in Frequent Itemset Mining Implementations: Report on FIMI'03. In: SIGKDD Explorations Newsletter, vol. 6(1), pp. 109-117. ACM New York (2004)

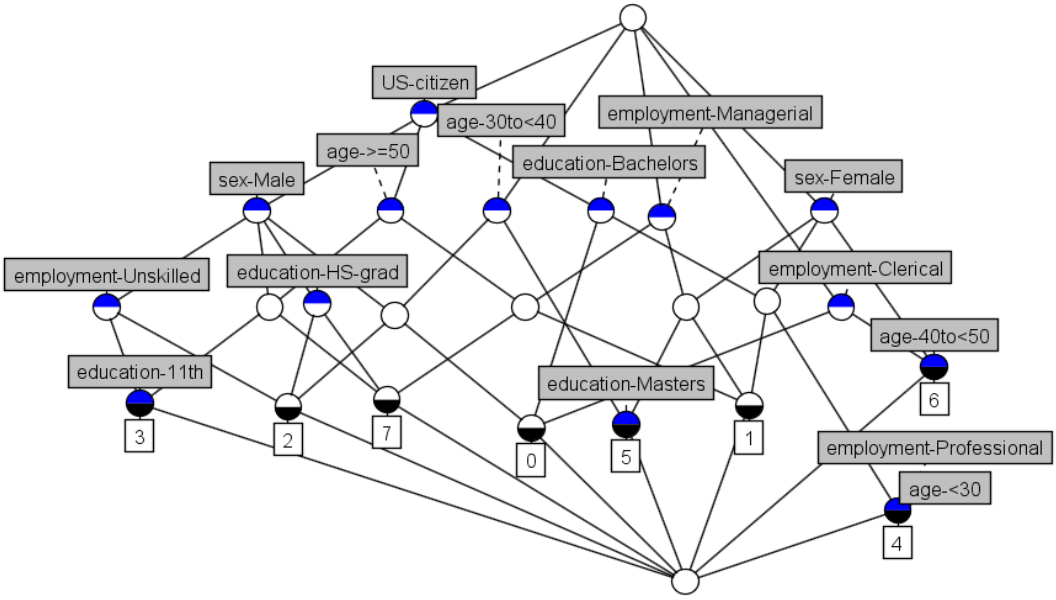



Figure 6. mini-adult Concept Lattice


[9] Ignatov, D.I., Kuznetsov, S.O.: Frequent Itemset Mining for Clustering Near Duplicate Web Documents. In: Rudolph, S., Dau, F., Kuznetsov, S.O. (eds.) ICCS 2009. LNCS (LNAI), vol. 5662, pp. 185-200. Springer-Verlag, Berlin/Heidelberg (2009)

[10] Kaytoue-Uberall, M., Duplesssis, S., Napoli, A.: Using Formal Concept Analysis for the Extraction of Groups of Co-expressed Genes. In: Le Thi, H.A., Bouvry, P., Pham Dinh, T. (eds.) MCO 2008. CCIS vol. 14, pp. 439-449. Springer-Verlag, Berlin/Heidelberg (2008)

[11] Krajca, P., Outrata, J., Vychodil, V.: Parallel Recursive Algorithm for FCA. In: Belohlavek, R., Kuznetsov, S.O. (eds.) CLA 2008, pp. 71-82. Palacky University, Olomouc (2008)

[12] Kuznetsov, S.O., Obiedkov, S.A.: Comparing Performance of Algorithms for Generating Concept Lattices. In: Journal of Experimental and Theoretical Artificial Intelligence, vol. 14, pp. 189-216 (2002)

[13] Passin, T. B.: Explorer's Guide to the Semantic Web. Manning, Greenwich, CT (2004)

[14] Priss, U.: Formal Concept Analysis in Information Science. In: Cronin, B. (ed.), Annual Review of Information Science and Technology, vol. 40, p. 521-543 (2006)

[15] Rioult, F., Robardet, C., Blachon, S., Crémilleux, B., Gandrillon, O., Boulicant, J-F.: Mining concepts from large SAGE gene expression matrices. In: Boulicant, J-F., Dzeroski, S. (eds.) Workshop on Knowledge Discovery in Inductive Databases 2003, pp. 107-118. Catat-Dubrovnik, Croatia (2003)

[16] Sawase, K., Nobuhara, H., Bede, B.: Visualizing Huge Image Databases by Formal Concept Analysis. In: Barjiela, A., Pedrycz, W. (eds.) Human-Centric Information Processing Through Granular Modelling. Studies in Computational Intelligence, vol. 182. Springer, Berlin/Heidelberg (2009)

[17] Ganter, B., Wille, R. (1998) *Formal Concept Analysis: Mathematical Foundations*, Springer-Verlag, Berlin. Translated by C. Franzke.

[18] Wolff, K.E.: A First Course in Formal Concept Analysis: How to Understand Line Diagrams. In: Faulbaum, F. (ed.) SoftStat 1993. Advances in Statistical Software vol. 4, pp. 429-438 (1993)

[19] Yevtushenko, S. (2006). *ConExp*. Available at: http://sourceforge.net/projects/conexp