

## **Knowledge discovery through creating formal contexts**

ANDREWS, Simon <<http://orcid.org/0000-0003-2094-7456>> and  
ORPHANIDES, Constantinos

Available from Sheffield Hallam University Research Archive (SHURA) at:

<https://shura.shu.ac.uk/5073/>

---

This document is the

**Citation:**

ANDREWS, Simon and ORPHANIDES, Constantinos (2012). Knowledge discovery through creating formal contexts. *International Journal of Space-Based and Situated Computing*, 2 (2), 123-138. [Article]

---

**Copyright and re-use policy**

See <http://shura.shu.ac.uk/information.html>

# Knowledge Discovery through creating Formal Contexts

Simon Andrews and Constantinos Orphanides

Communication and Computing Research Centre  
Faculty of Arts, Computing, Engineering and Sciences  
Sheffield Hallam University, Sheffield, UK  
`s.andrews@shu.ac.uk`, `c.orphanides@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. Examples are given of how formal contexts can be created using *FcaBedrock* and then analysed for knowledge discovery, using real datasets. Creating formal contexts using *FcaBedrock* is shown to be straightforward and versatile. Large data sets are easily converted into a standard FCA format.

## 1 Introduction

Formal Concept Analysis (FCA) was introduced in the 1990s by Rudolf Wille and Bernhard Ganter (Ganter and Wille, 1998), building on applied lattice and order theory developed by Birkhoff and others in the 1930s. It was initially developed as a subsection of Applied Mathematics based on the mathematisation of concepts and concepts hierarchy, where a 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 of the extension (Wille, 2005). The set of 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		×	×	×	×	×			×

### 1.1 Formal Contexts

The cross-table above shows a formal context representing destinations for five airlines. The elements on the left side are formal objects; the elements at the top are formal attributes. If an object has a specific property (formal 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 above, Air Canada flies to Latin America (since the corresponding cell contains a cross) but does not fly to Africa (since the corresponding cell is empty). However, an empty cell might also mean that it is unknown whether the corresponding object has the corresponding attribute (Wolff, 1993).

In mathematical terms, a formal context is defined as a triple  $\mathbb{K} := (G, M, I)$ , with  $G$  being a set of objects,  $M$  a set of attributes and  $I$  a relation defined between  $G$  and  $M$ . The relation  $I$  is understood to be a subset of the cross product between the sets it relates, so  $I \subseteq G \times M$ . If an object  $g$  has an attribute  $m$ , then  $g \in G$  relates to  $m$  by  $I$ , so we write  $(g, m) \in I$ , or  $gIm$ . For a subset of objects  $A \subseteq G$ , a derivation operator  $'$  is defined to obtain the set of attributes, common to the objects in  $A$ , as follows:

$$A' = \{m \in M \mid \forall g \in A : gIm\}$$

Similarly, for a subset of attributes  $B \subseteq M$ , the derivation operator  $'$  is defined to obtain the set of objects, common to the attributes in  $B$ , as follows:

$$B' = \{g \in G \mid \forall m \in B : gIm\}$$

### 1.2 Formal Concepts

Now, a pair  $(A, B)$  is a *Formal Concept* in a given formal context  $(G, M, I)$  only if  $A \subseteq G$ ,  $B \subseteq M$ ,  $A' = B$  and  $B' = A$ . The set  $A$  is the extent of the concept and the set  $B$  is the intent of the concept. A formal concept is, therefore, a closed set of object/attribute relations, in that its extension contains all objects that have the attributes in its intension, and the intension contains all attributes shared by the objects in its extension. In the Airlines example, it can be seen from the cross-table that Air Canada and Austrian Airlines fly to both USA and

Europe. However, this does not constitute a formal concept because both airlines also fly to Asia Pacific, Canada and the Middle East. Adding these destinations completes (closes) the formal concept:

( $\{\text{Air Canada, Austrian Airlines}\}, \{\text{Europe, USA, Asia Pacific, Canada, Middle East}\}$ ).

### 1.3 Computing Formal Concepts

For the computation of formal concepts, a Boolean matrix is frequently used to represent the formal context. In the boolean matrix, the rows and columns are computationally interchangeable; the same will apply if ‘A’, ‘extent’, ‘object’ and ‘row’ are substituted for ‘B’, ‘intent’, ‘attribute’ and ‘column’ (and vice-versa).

There are several algorithms for computing the formal concepts in a formal context – reviews of such algorithms, that take a variety of approaches, can be found in (Arevalo et al, 2007; Kuznetsov and Obiedkov, 2002; Zaki and Hsiao, 2005).

An example of such an algorithm is given in Figure 1, implemented in the formal concept miner In-Close<sup>1</sup>, developed by one of the authors. In-Close is based conceptually on Kuznetsov’s *Close-By-One* algorithm (Kuznetsov, 1999). To better understand the algorithm,  $I$  is a Boolean matrix representing a formal context, with  $m$  rows, representing a set of objects, and  $n$  columns, representing a set of attributes. A formal concept is represented by an extent  $A[r]$  (an ordered list of objects) and an intent  $B[r]$  (an ordered list of attributes), where  $r$  is the index of the concept. In the algorithm,  $j$  is the current attribute,  $r$  is the index of the concept being closed,  $r_{new}$  is a global index of the candidate new concept and  $y$  is a starting column.  $\text{InClose}(r, y)$  means ‘incrementally close concept  $r$ , beginning at attribute  $y$ ’.  $\text{IsCanonical}$  checks if a concept is new by examining its canonicity (lexicographical order) to circumvent the need to explicitly search for repeated results (Ganter, 1984). For a more thorough and technical explanation of how the algorithm works, refer to (Andrews, 2009b, 2011).

### 1.4 Galois Connections

Another central notion of FCA is a duality called a ‘Galois connection’, which is often observed between items that relate to each other in a given domain, such as objects and attributes. A Galois connection implies that “if one makes the sets of one type larger, they correspond to smaller sets of the other type, and vice versa” (Priss, 2008). Using the formal concept above as an example, if Africa is added to the list of destinations, the set of airlines reduces to  $\{\text{Austrian Airlines}\}$ .

### 1.5 Concept Lattices

The Galois connections between the formal concepts of a formal context can be visualized in a *Concept Lattice* (Figure 2), which is an intuitive way of discovering

<sup>1</sup> <http://sourceforge.net/projects/inclose/>

---

InClose( $r, y$ )

---

```

1 begin
2    $r_{new} \leftarrow r_{new} + 1$ ;
3   for  $j \leftarrow y$  upto  $n - 1$  do
4      $A[r_{new}] \leftarrow \emptyset$ ;
5     foreach  $i$  in  $A[r]$  do
6       if  $I[i][j]$  then
7          $A[r_{new}] \leftarrow A[r_{new}] \cup \{i\}$ ;
8     if  $|A[r_{new}]| > 0$  then
9       if  $|A[r_{new}]| = |A[r]|$  then
10         $B[r] \leftarrow B[r] \cup \{j\}$ ;
11      else
12        if IsCannonical( $r, j - 1$ ) then
13           $B[r_{new}] \leftarrow B[r] \cup \{j\}$ ;
14          InClose( $r_{new}, j + 1$ );
15 end

```

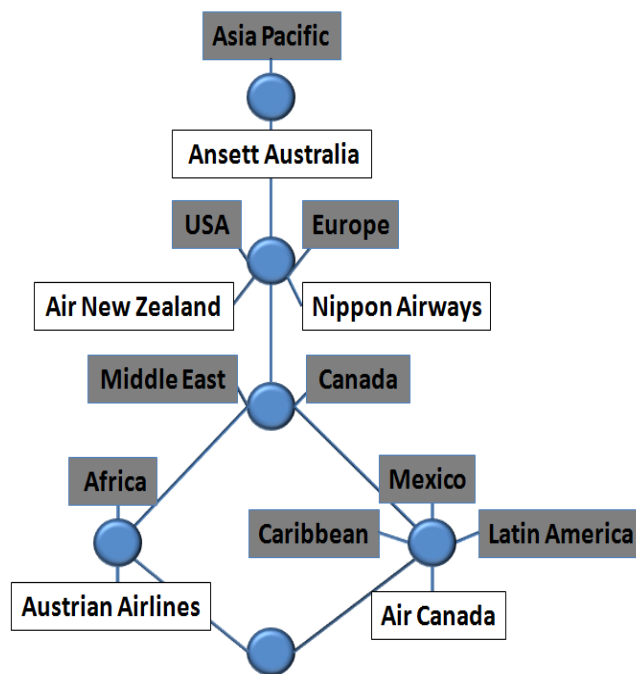
---

**Fig. 1.** The In-Close algorithm.

hitherto undiscovered information in data and portraying the natural hierarchy of concepts that exist in a formal context.

A concept lattice consists of the set of concepts of a formal context and the subconcept-superconcept relation between the concepts. The nodes in Figure 2 represent formal concepts. It is conventional that formal objects are noted slightly below and formal attributes slightly above the nodes, which they label.

A concept lattice can provide valuable information when one knows how to read it. As an example, the node which is labeled with the formal attribute ‘Asia Pacific’ shall be referred to as *Concept A*. To retrieve the extension of Concept A (the objects which feature the attribute ‘Asia Pacific’), one begins at the node where the attribute is labeled and traces all paths which lead down from the node. Any objects one meets along the way are the objects which have that particular attribute. Looking at the lattice in Figure 2, if one takes the attribute ‘Asia Pacific’ and traces all paths which lead down from the node, one will collect all the objects. Thus Concept A can be interpreted as ‘All airlines fly to Asia Pacific’. Similarly, the node which is labeled with the formal object ‘Air New Zealand’ shall be referred to as *Concept B*. To retrieve the intension of Concept B (the attributes of ‘Air New Zealand’), one begins at the node where the object is labeled and traces all paths which lead up from the node. Any attributes one meets along the way, are the attributes of that particular object. Looking at the lattice once again, if one takes the object ‘Air New Zealand’ and traces all paths which lead up from the node, one will collect the attributes ‘USA’, ‘Europe’, and ‘Asia Pacific’. This can be interpreted as ‘The Air New Zealand airline flies



**Fig. 2.** A lattice corresponding to the Airlines context.

to USA, Europe and Asia Pacific’. The concept that we formed previously by inspecting the cross-table,

$$(\{\text{Air Canada, Austrian Airlines}\}, \{\text{Europe, USA, Asia Pacific, Canada, Middle East}\}),$$

is the node in the center of the lattice; the one labeled with ‘Middle East’ and ‘Canada’. It becomes quite clear, for example, that although Air New Zealand and Nippon Airways also fly to Europe, USA and Asia Pacific, only Air Canada and Austrian Airlines fly to Canada and the Middle East as well.

Although the Airline context is a small example of FCA, visualising the formal context clearly shows that concept lattices provide richer information than from looking at the cross-table alone. This type of hierarchical intelligence that is gleaned from FCA is not so readily available from other forms of data analysis.

## 1.6 Converting Data for FCA

It has been shown that FCA can be usefully applied to large sets of data (Poelmans et al, 2011; Kaytoue-Uberall et al, 2008; Sawase et al, 2009; Ignatov and Kuznetsov, 2009) and that it has applications in data mining (Boulicaut and Besson, 2008; Rioult et al, 2003). Algorithms exist that are capable of processing large formal contexts in good time (Andrews, 2011; Krajca et al, 2008). However, data is rarely in a form immediately suitable for FCA tools and applications. 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 Booleanising 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 (Wolff, 1993; Ganter and Wille, 1998) and continuous data can be used for FCA by being hierarchically scaled (Priss, 2008) or discretized as disjoint ranges of values (Kaytoue-Uberall et al, 2008). However, it is not always obvious how or even if a particular attribute should be converted. If an attribute has hundreds of values, converting it into hundreds of formal attributes may be pointless if this leads to a hugely complex lattice - it is probably not suitable for FCA. For continuous values the question arises as to how to form the ranges to discretise them; how many and how large? Consequently, there is less work describing how these techniques can be applied in an automated, reproducible way.

A process of data conversion leads to the possibility of diverse interpretation of data. Different choices of which of the data to convert and how to convert them can, without careful documentation, lead to inconsistent and incomparable analyses and lead to problems in measuring the performance of FCA tools and algorithms (Kuznetsov and Obiedkov, 2002). It was thus proposed at CSTIW’09 that a tool should be developed to apply automation to the process of conversion and to address these issues (Andrews, 2009a).

## 2 FcaBedrock Overview

FcaBedrock<sup>2</sup> is an open-source tool for automating the conversion of existing, flat-file csv datasets into formal contexts. Creating formal contexts is a common process which, until now, was done manually. Several FCA tools, such as the ToscanaJ kit for browsing conceptual schemas and the lattice builder ConExp<sup>3</sup> (Yevtushenko, 2000), exist which are capable of creating formal contexts, but are time consuming and not ‘automated’, as they were not built for this purpose.

FcaBedrock implements a novel approach, using a process of guided automation (Andrews and Orphanides, 2010a), meaning that while most processes of the tool are automated, it is the user who has the final say on how to interpret the data (guided automation is explained in section 5 of this article). The user supplies the tool with appropriate metadata 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, these metadata are used by FcaBedrock to create a formal context file in a standard form for FCA (Figure 3). The metadata is stored in a separate text document called a *Bedrock* (.bed) file. This can be used for subsequent conversions and act as a record of the interpretation made of the dataset. Bedrock files can be loaded into FcaBedrock, allowing the reproduction of context files and allowing changes in the interpretation to be made. User-defined constraints applied to the data allow different analyses to be carried out. Each analysis can be documented with a Bedrock file. Multiple data files with the same attributes can be converted using the same Bedrock file.

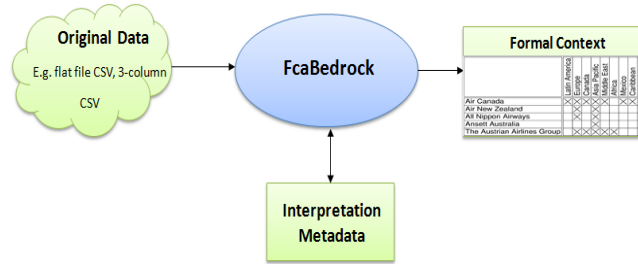


Fig. 3. FcaBedrock Process

### 2.1 Input Formats

FcaBedrock currently supports two input data file formats: many-column and three-column delimiter separated values (dsv). The comma and tab characters

<sup>2</sup> <https://sourceforge.net/projects/fcabedrock/>

<sup>3</sup> <https://sourceforge.net/projects/conexp/>

are currently supported as value delimiters. FcaBedrock outputs the formal context in the widely used *Burmeister* (.cxt) and *FIMI* (.dat) file formats.

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

The three-column format is becoming popular as a standardized data format and is the approach used, for example, in triple-stores (Rohloff et al, 2007) and the Resource Description Format (RDF) (Abadi et al, 2009). 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 (this feature shall be explicated later on in this article), the second values are detected as attribute names. The order in which triples appear in a data file is not significant.

## 2.2 Output Formats

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 *Frequent Itemset Mining Implementations* (FIMI) data format (dat) is used in data mining, particularly in testing the efficiency of algorithms (Goethals and Zaki, 2004). Unlike *Burmeister*, FIMI does not include object or attribute names, but just the formal attributes featured in each formal object.

## 3 Attribute Values

Figure 4 shows FcaBedrock. There are three sections on top, for specifying how continuous attributes should be scaled, what delimiter is used in the input file and if the first row in the input file is to be used as attribute names. 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 5.2), 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 because these are the values that appear in the input file, but the category names are the ones that will appear in the output context file. This means that human-readable names can be given to code values, for example, to make the resultant concept lattice understandable. The example shown is of the well-known *Mushroom* data set from UCI (Frank and Asuncion, 2010), 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 has category values in the data file that each contain a leading space character. The spaces are better omitted from the category names to make them more readable, but are required for converting the data.

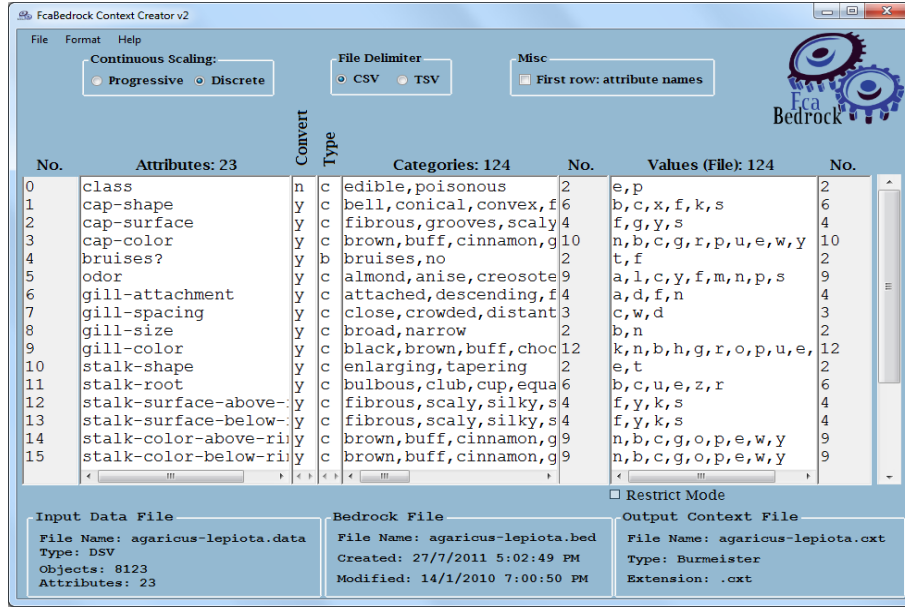


Fig. 4. FcaBedrock showing metadata for the UCI *mushroom* data set

## 4 Attribute Types

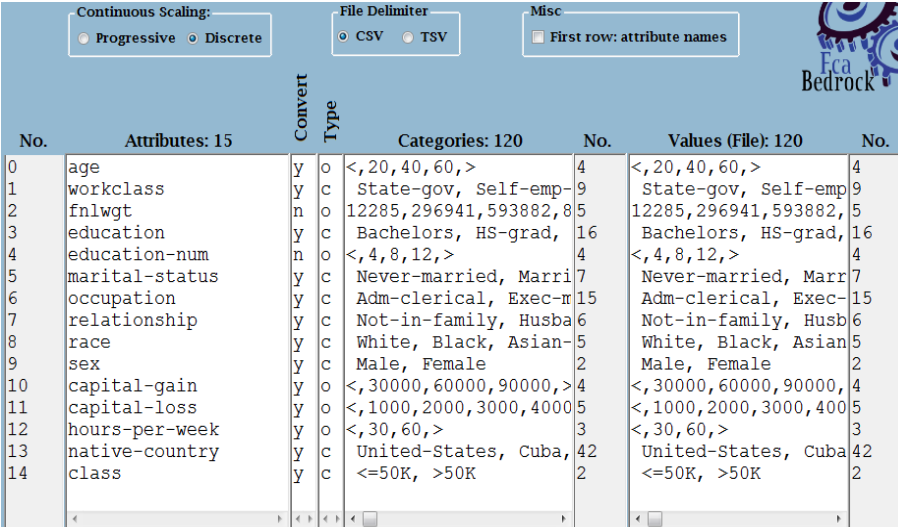
### 4.1 Categorical Attributes

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 4, 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*.

## 4.2 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?*.



No.	Attributes: 15	Convert	Type	Categories: 120	No.	Values (File): 120	No.
0	age	y	o	<,20,40,60,>	4	<,20,40,60,>	4
1	workclass	y	c	State-gov, Self-emp-	9	State-gov, Self-emp	9
2	fnlwgt	n	o	12285,296941,593882,8	5	12285,296941,593882,5	5
3	education	y	c	Bachelors, HS-grad, 16	16	Bachelors, HS-grad, 16	16
4	education-num	n	o	<,4,8,12,>	4	<,4,8,12,>	4
5	marital-status	y	c	Never-married, Marri	7	Never-married, Marr	7
6	occupation	y	c	Adm-clerical, Exec-n	15	Adm-clerical, Exec-	15
7	relationship	y	c	Not-in-family, Husba	6	Not-in-family, Husb	6
8	race	y	c	White, Black, Asian-	5	White, Black, Asian	5
9	sex	y	c	Male, Female	2	Male, Female	2
10	capital-gain	y	o	<,30000,60000,90000,>	4	<,30000,60000,90000,4	4
11	capital-loss	y	o	<,1000,2000,3000,4000	5	<,1000,2000,3000,400	5
12	hours-per-week	y	o	<,30,60,>	3	<,30,60,>	3
13	native-country	y	c	United-States, Cuba,	42	United-States, Cuba	42
14	class	y	c	<=50K, >50K	2	<=50K, >50K	2

Fig. 5. UCI *Adult* metadata in FcaBedrock

## 4.3 Continuous Attributes

Continuous attributes are dealt with in FcaBedrock by discretising the data using user-defined ranges (e.g. *0-10*, *10-20...*), or by hierarchical scaling (e.g. *>0*, *>10*, *>20...*) (Priss, 2008). Figure 5 shows the metadata 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 four categories: *less than 20*, *20 to less than 40*, *40 to less than 60* and *greater than or equal to 60*. The formal attribute names are created by concatenating values appropriately: *age-<20* and *age-20to<40*, for example. 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 formal context.

#### 4.4 Ordinal Attributes

Ordinal attributes have been introduced as a new feature, in a version of FcaBedrock not available online during the writing of this article (although it may be available, of course, by the time this article has gone to press). Ordinal attributes are categorical attributes where the ordering of the attribute values is important. Taking as an example an attribute *Education* containing higher education degrees, it is important that a foundation course comes before a Bachelors degree, a Doctorate comes after a Masters degree and so on. Ordinality also allows the creation of sensible ranges to summarize values for non-continuous attributes. Using the same example, creating ranges for the *Education* attribute can be used to group education levels to undergraduate and postgraduate: Foundation–Bachelors, Masters–Doctorate. Capturing the order of non-numerical data is a feature that allows analyses that are not easy using traditional means of data analysis. FCA of such ordinal data can provide novel intelligence from a concept lattice, as will be seen later in this article.

Declaring ranges for ordinal attributes is possible in FcaBedrock using the same process used for declaring ranges for continuous attributes. For example, the boundaries *<*, *Bachelors*, *Masters* and *>* will be used by FcaBedrock to create three categories: *less than Bachelors*, *Bachelors to less than Masters* and *greater than or equal to Masters*. Discrete and hierarchical scaling is supported for ordinal attributes as well.

When an ordinal attribute has been declared, FcaBedrock assigns a numerical value to each of the attribute values. During the conversion process, FcaBedrock reads an attribute value, finds its numerical counterpart and compares it with the appropriate boundary values, assigning a *True* value to the corresponding formal attribute in the formal context.

#### 4.5 Date Attributes

Date attributes have been introduced as a new feature, in a version of FcaBedrock not available online during the writing of this article. The most common date formats are supported. As an example, creating ranges for a *Dates* attribute can be used to group months into seasons: 01/12/2010–28/02/2011, 01/03/2011–31/05/2011, 01/06/2011–31/08/2011 and 01/09/2011–30/11/2011. Declaring ranges for date attributes is possible in FcaBedrock using the same process used for declaring ranges for continuous attributes. For example, the boundaries *01/12/2010*, *01/03/2011*, *01/06/2011*, *01/09/2011* and *>* will be used by FcaBedrock to create four categories: *01/12/2010 to less than 01/03/2011*, *01/03/2011 to less than*

*01/06/2011*, *01/06/2011 to less than 01/09/2011* and *greater than or equal to 01/09/2011*. Adding hours, minutes and seconds to a date is also possible – *23/09/2011 01:55:40 AM* is valid as a date value.

During the conversion process, FcaBedrock reads a date value and compares it with the appropriate boundary values, assigning a *True* value to the corresponding formal attribute in the formal context.

## 5 Guided Automation

### 5.1 Attribute Exclusion and Restriction

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. Individual attribute categories can be excluded from the formal context by simply not including them in the list.

Sub-contexts can be also created by restricting the conversion to user-specified attribute values. By specifying one or more category values of one or more attributes, the formal context will only contain objects with 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.

### 5.2 Auto-detection of Metadata

If a data file is read without first loading or creating a Bedrock file, FcaBedrock can detect attribute values and create metadata automatically. It initially assumes that all attributes are categorical. It adds each new value it finds in a data-column to a corresponding list of category values. If FcaBedrock determines that there is a high proportion of different numerical values for an attribute, it provides the option of automatically creating ranges and declares the attribute as continuous. This also applies for dates – if FcaBedrock determines that there is a high proportion of different dates for an attribute, it provides the option of automatically creating ranges and declares the attribute as date.

## 6 A Mini-Adult example

The following is an example adapted from the UCI *Adult* data set, to better illustrate the transformation of data into a formal context.

```

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 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.

The *mini-adult* metadata in FcaBedrock are shown in Figure 6. The output *Burmeister* context file, `mini-adult.cxt`, is shown in File 2.

```

B
8
15

0
1
2
3
4
5
6
7
age-<30
age-30to<40
age-40to<50
age->=50
education-Bachelors
education-Masters
education-11th

```

No.	Attributes: 6	Convert Type	Categories:	No.	Values (file):	No.
0	age	y	o <,30,40,50,>	4	<,30,40,50,>	4
1	education	y	c Bachelors,Masters,11t	4	Bachelors,Masters,11t	4
2	employment	y	c Clerical,Managerial,F	4	Clerical,Managerial,	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

Fig. 6. *mini-adult* metadata in FcaBedrock

```

education-HS-grad
employment-Clerical
employment-Managerial
employment-Professional
employment-Unskilled
sex-Male
sex-Female
US-citizen
.X..X...X...X.X
...XX...X...XX
.X.....X...XX.X
...X..X...XX.X
X...X.....X..XX
.X...X...X...X.
..X.....X...X.
...X...X.X...X.X

```

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

## 7 Analysing Formal Contexts for Knowledge Discovery

To illustrate how knowledge discovery is possible through analysing formal contexts, practical examples are given, using real datasets from the UCI Machine Learning Repository.

### 7.1 Mushroom Dataset

The Mushroom dataset contains information of 8124 edible and poisonous mushrooms of the *agaricus* and *lepiota* families. These are many-valued categorical data which have attributes that describe properties such as veil type, veil colour, population and habitat.

Figure 7 shows two mushrooms of the *agaricus* family; *Agaricus Arvensis* and *Agaricus Xanthodermus*. On first sight, both mushrooms seem to be edible. However, *Agaricus Xanthodermus* has poisonous properties. This is a good example of how mushroom identification can be a confusing process for non-experts who are unable to distinguish subtle but important differences between mushrooms which resemble to each other. Figure 8 shows how the attribute restriction and



**Fig. 7.** *Agaricus Arvensis* (to the left) and *Agaricus Xanthodermus* (to the right)

attribute exclusion capabilities of FcaBedrock can be applied to create a sub-context of this dataset, in order to determine if a mushroom is safe to eat or not based on some of its characteristics. In this analysis, 20 attributes were excluded from this conversion, except from the *class* (edible and poisonous), *cap-color* and *habitat* attributes. In addition, the cap-color attribute was restricted to the values *red*, *pink*, *green* and *purple* and the habitat attribute was restricted to the values *woods*, *urban*, *meadows* and *grasses*. The restrictions set resulted in 996 objects (out of the initial 8124) and 19 formal attributes (out of the initial 125). Essential to the analysis is the fact that this sub-context contains few enough formal concepts (nodes) to make the visualisation practical.

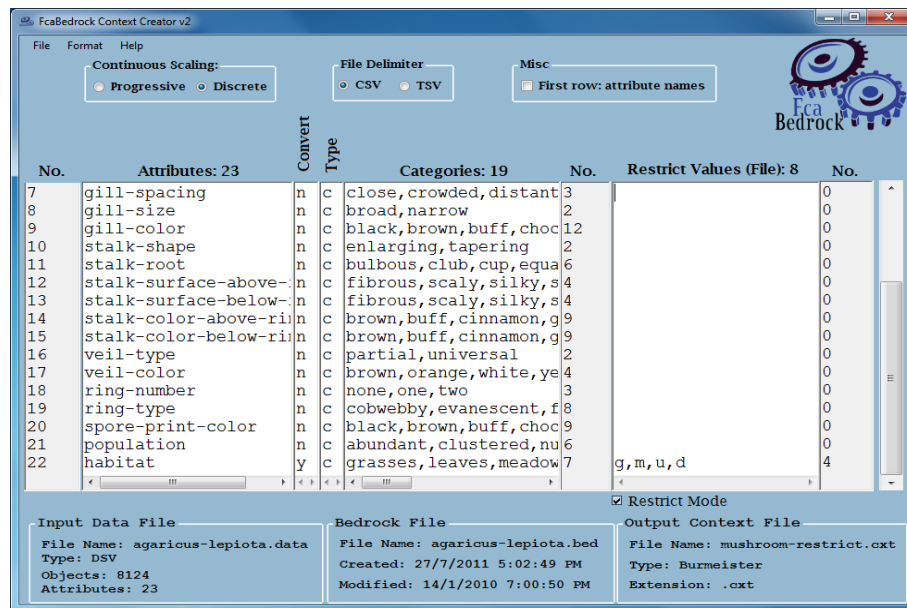
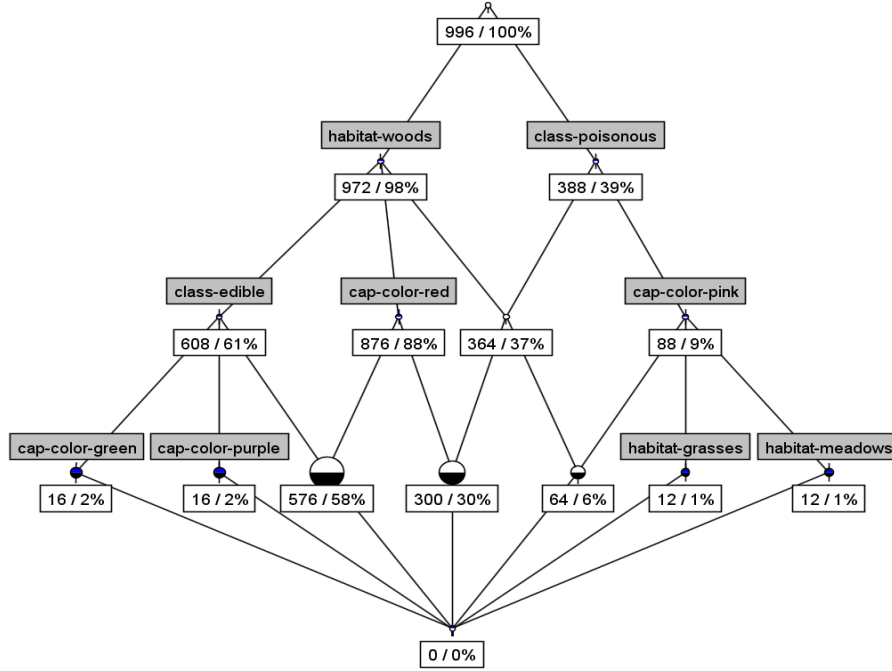


Fig. 8. Creating a mushroom *class-habitat-cap color* sub-context in FcaBedrock

Visualising the sub-context in ConExp (Figure 9) produced some interesting information. By labeling each concept in the lattice with the object count and percentage of the overall number of objects, in ConExp, it appears that, in the sample, 98% of all the mushrooms live in woods, out of which 61% are safe to eat. It also appears that the most dominant mushrooms have a red cap, of which 576 are edible and 300 are poisonous, perhaps indicating that mushroom cap colors are not the safest indicator for determining their edibility. In contrast, pink-capped mushrooms can be found in woods and less frequently in grasses or meadows and all of them are poisonous. Mushrooms with a green or purple cap are extremely rare, but if one is lucky enough to find some they are safe to eat.

Let us now say that we are interested in the relationship between mushroom



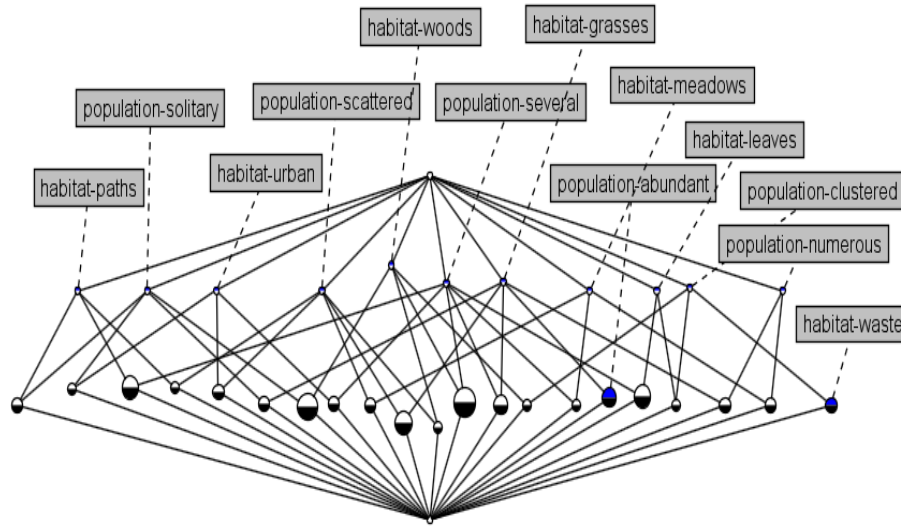
**Fig. 9.** A mushroom *class-habitat-cap color* sub-context, created by FcaBedrock, visualised in ConExp

habitat and population type. Once again, FcaBedrock’s attribute exclusion feature was used to create a sub-context containing only the habitat and population type attributes. There were 13 formal attributes in all, corresponding to the 7 categories of habitat and the 6 categories of population type.

The concept lattice of this sub-context is shown in Figure 10. In ConExp, the size of the node in the lattice was made proportional to the number of own objects (mushrooms), so it can be seen that, for example, clusters of mushrooms are found in similar numbers in woods, leaves and waste ground, but not in other habitats. Solitary mushrooms are usually found in woods, although they can be occasionally found in paths, urban areas and grassland.

## 7.2 Adult Dataset

The Adult dataset comprises of US Census data of 32,561 adults, describing properties such as age, education, employment type, race and sex.



**Fig. 10.** A mushroom *habitat-population* sub-context, created by FcaBedrock, visualised in ConExp

Converting all the suitable attributes of this dataset results in a formal context with more than 100,000 concepts – far too many to visualise. However, let us say we are interested in comparing how pay is affected by gender in adults who have had a higher education. For this analysis to be carried out, only the sex (*male*, *female*), class ( $\leq 50k$ ,  $> 50k$ ) and education attributes. The attribute value restriction feature was also used to convert only those objects (adults) with the education attribute value *Bachelors*, *Masters* or *Doctorate* (3 out of the 16 possible categories of education in the dataset). This resulted in a sub-context with 7,491 (out of the initial 32,561) and 7 formal attributes (out of the initial 129).

The resulting concept lattice in ConExp is shown in Figure 12, containing 37 concepts. The number of objects (and the percentage of the whole) is being displayed for the concepts of interest. Because only objects with Bachelors, Masters or Doctorate education categories were included in the conversion, there were 13 education categories (such as 10th grade and high school graduate) left with zero objects associated with them. Such unsupported formal attributes are normally labeled at the infimum of the concept lattice, but these labels can be hidden in ConExp.

Although a little cluttered with lines, the lattice is still readable, aided by a ConExp feature whereby information regarding a concept is displayed when a node is pointed to with the mouse. An examination of the concepts allows us to compare the percentage of males and females who earn more than \$50k, for each type of higher education. With a Bachelor's degree, 21% of females

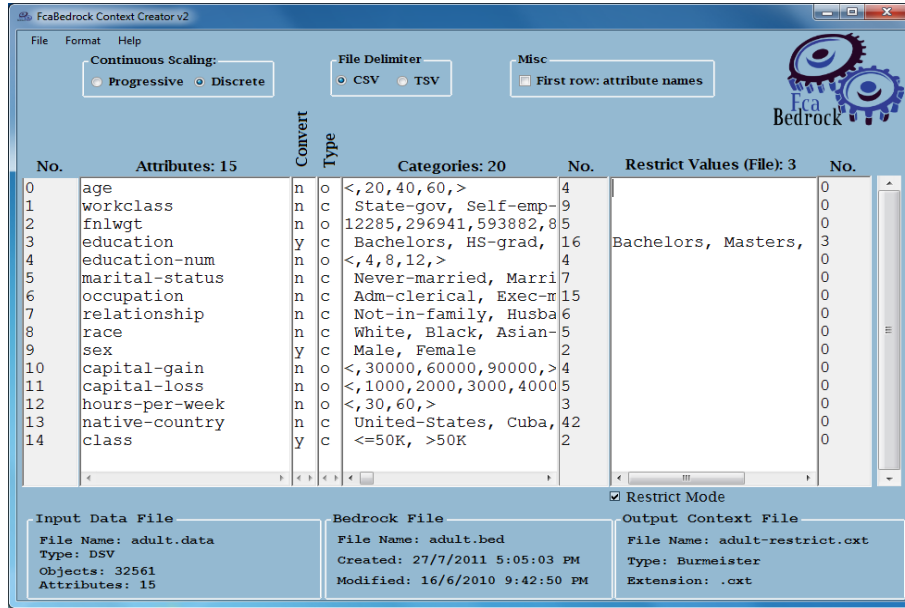


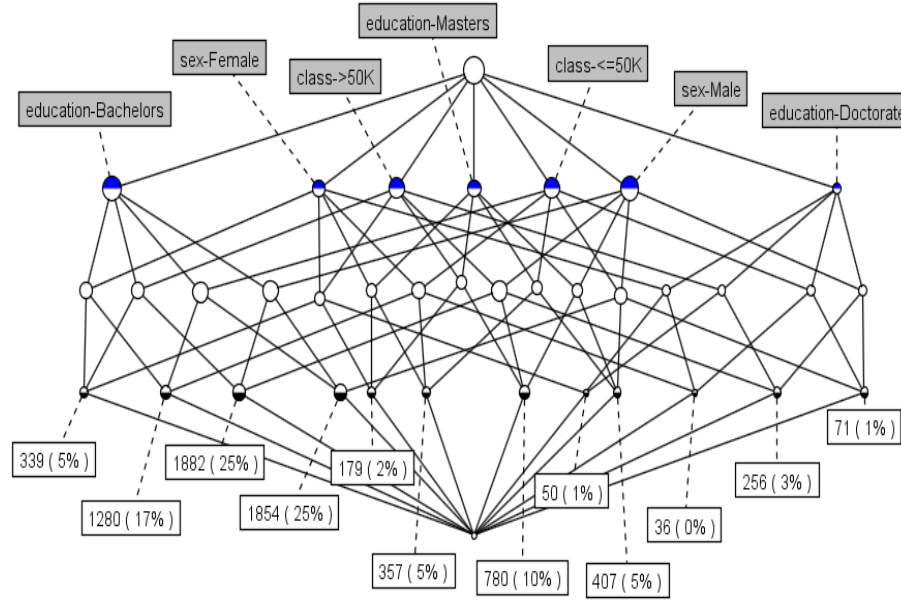
Fig. 11. Adult *degree-sex-pay* sub-context, created by FcaBedrock

and 50% of males earned more than \$50k. This ‘gender gap’ was maintained at Master’s level, with 33% of females and 65% of males earning more than \$50k, but narrowed slightly at Doctorate level, with 58% of females and 78% of males earning more than \$50k.

Let us say that we are now interested in how age relates to the number of hours worked per week. For this analysis, only the *age* and *hours-per-week* attributes are converted, using progressive scaling. The lattice for this sub-context is shown in Figure 13. The first scale represents the age of the adults, using the ranges *<20*, *<40*, *<60* and *all*. The second scale shows the hours worked per week, using the ranges *<20*, *<40*, *<60*, *<80* and *all*.

By reading the concept lattice, intelligence concerning the hours worked per week, according to the age of the population, can be determined. For example, it can be clearly seen that 8% of the population work more than 60 hours per week. However, when it comes to young adults (less than 20 years old) about a quarter of them work less than 20 hours per week and about three-quarters work less than 40 hours per week. This can be compared to the population as a whole, where only 5% work less than 20 hours and only a quarter work less than 40 hours.

An interesting analysis is possible if the *education* attribute is considered as an ordinal attribute: In the analysis in Figures 11 and 12, the education attribute was declared as categorical and restricted to the 3 values *Bachelors*, *Masters* and *Doctorate*, out of the initial 16. However, the ordinality of educational attainment

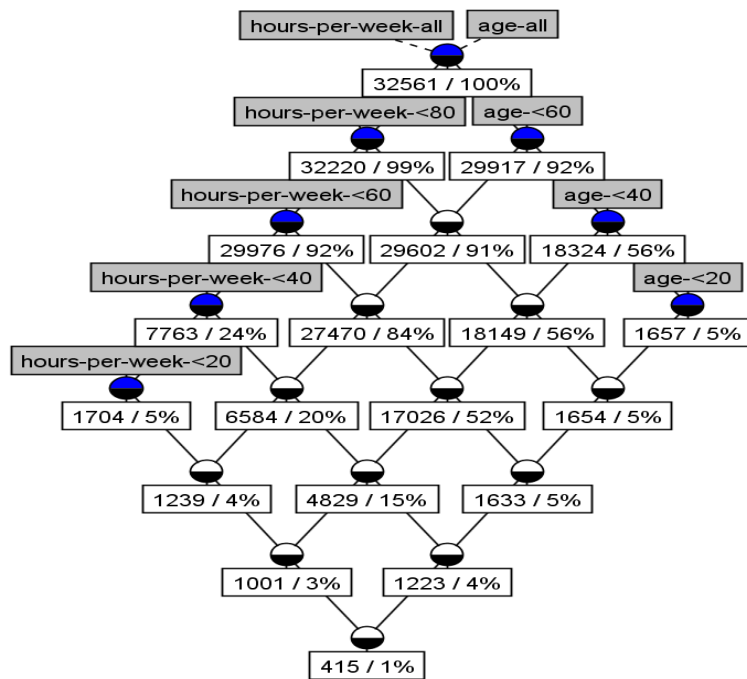


**Fig. 12.** Adult *degree-sex-pay* sub-context, created by FcaBedrock, visualised in Con-Exp

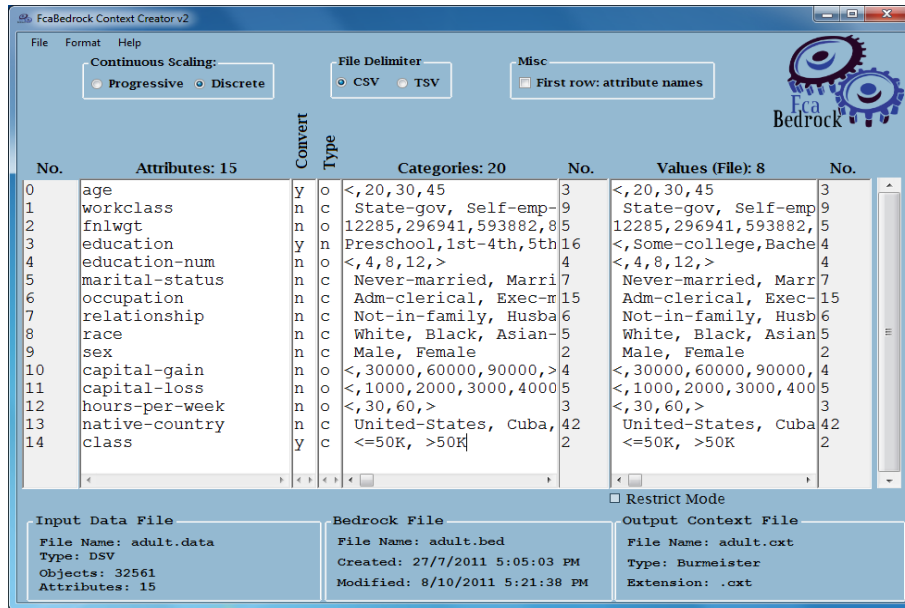
can be used to capture all the education levels as sensible groupings. Furthermore, as opposed to the previous analysis where restricting the conversion to specific attribute values only converts objects which have those particular attribute values, in this analysis all objects are included in the analysis since no restrictions are set. This allows us to have a broader conceptual overview of the data.

Let us now say that we are interested in how does education and income relate for all the people in the dataset. For this analysis, only the *age*, *class* and *education* attributes were converted. For the education attribute to make sense as ordinal, the values of the education attribute were re-ordered from lower to higher education levels, as shown in Figure 14. The boundaries  $<$ , *Some-college*, *Bachelors* and  $>$  were set for the education attribute, resulting in three ranges:  $<Some-college$ , *Some-college* to  $<Bachelors$  and  $>=Bachelors$ . For the age attribute, the boundaries  $<$ , 20, 30 and 45 were set, resulting in three ranges:  $<20$ , 20 to  $<30$  and 30 to  $<45$ .

Although somewhat cluttered with lines, the resulting concept lattice in Figure 15 still shows some interesting results. It appears that 55% of the entire population has not attended college whatsoever, while the rest 45% is almost evenly distributed between attending some college and earning one or more degrees. Looking at the obvious, larger concepts (nodes) in the lattice, it can be seen that 45% of the population did not attend college and earn less than 50k,



**Fig. 13.** Adult *age-hours per week* sub-context, created by FcaBedrock, visualised in ConExp



No.	Attributes: 15	Convert	Type	Categories: 20	No.	Values (File): 8	No.
0	age	y	o	<, 20, 30, 45	3	<, 20, 30, 45	3
1	workclass	n	c	State-gov, Self-emp	9	State-gov, Self-emp	9
2	fnlwgt	n	o	12285, 296941, 593882, 8	5	12285, 296941, 593882, 5	5
3	education	y	n	Preschool, 1st-4th, 5th	16	<, Some-college, Bache	4
4	education-num	n	o	<, 4, 8, 12, >	4	<, 4, 8, 12, >	4
5	marital-status	n	c	Never-married, Marri	7	Never-married, Marr	7
6	occupation	n	c	Adm-clerical, Exec-m	15	Adm-clerical, Exec-	15
7	relationship	n	c	Not-in-family, Husba	6	Not-in-family, Husb	6
8	race	n	c	White, Black, Asian-	5	White, Black, Asian	5
9	sex	n	c	Male, Female	2	Male, Female	2
10	capital-gain	n	o	<, 30000, 60000, 90000, >	4	<, 30000, 60000, 90000, 4	4
11	capital-loss	n	o	<, 1000, 2000, 3000, 4000	5	<, 1000, 2000, 3000, 400	5
12	hours-per-week	n	o	<, 30, 60, >	3	<, 30, 60, >	3
13	native-country	n	c	United-States, Cuba,	42	United-States, Cuba	42
14	class	y	c	<=50K, >50K	2	<=50K, >50K	2

**Input Data File**  
File Name: adult.data  
Type: DSV  
Objects: 32561  
Attributes: 15

**Bedrock File**  
File Name: adult.bed  
Created: 27/7/2011 5:05:03 PM  
Modified: 8/10/2011 5:21:38 PM

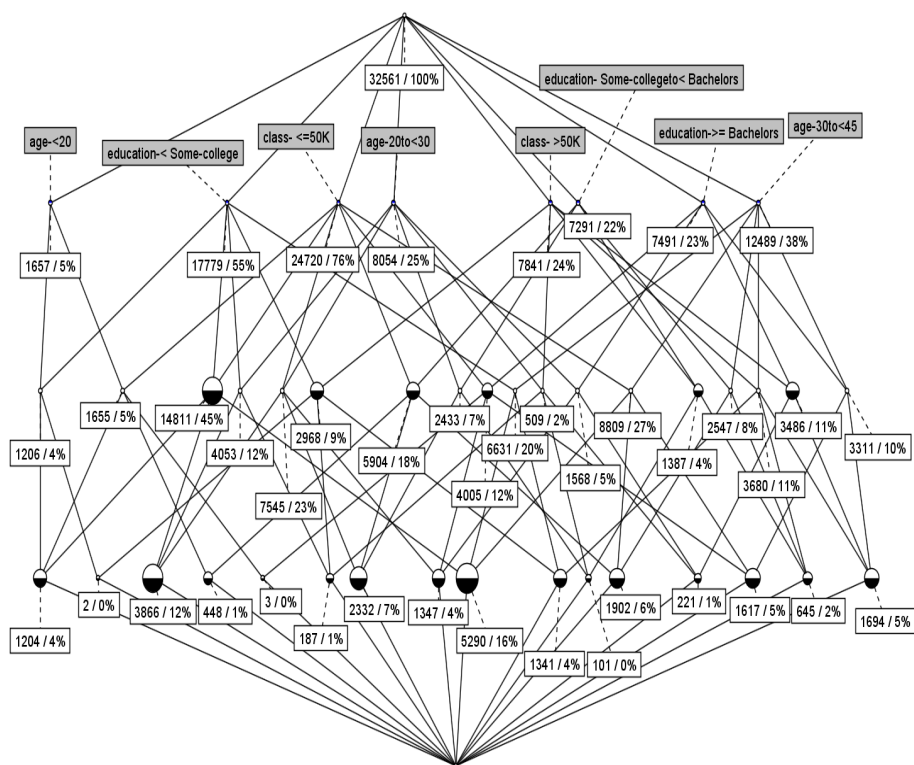
**Output Context File**  
File Name: adult.cxt  
Type: Burmeister  
Extension: .cxt

**Fig. 14.** Adult *age-pay-degree* sub-context, created by FcaBedrock. Note the ordering of the attribute values of education: *Preschool, 1st-4th, 5th...*

out of which 35.7% are in the 30–45 age group, 26.1% are in the 20–29 age group and 8.1% are less than 20 years old. This could possibly indicate the economic and social difficulties of attending college for older generations and how education is considered an essential advantage in terms of career evolution for younger generations.

On the other hand, only 24% of the entire population earns more than 50k, out of which 29% belongs to the 30–45 group, with 48% of them having a college degree. For the 20–30 group, 19.4% have a college degree, out of which only 14% earn more than 50k, although a similar percentage of this age group earns more than 50k without attending college whatsoever.

Some of the smaller concepts yield useful intelligence too. For example, it appears that 187 people of the 20–30 age group and 2 people younger than 20 earn more than 50k and have not attended college. Similarly, there are 101 people of the 20–30 group which are college dropouts and earn more than 50k. It appears that, for some, lack of educational attainment has not been a barrier to gainful employment. Indeed, a significant few have forgone further education to pursue early success.



**Fig. 15.** Adult *age-pay-degree* sub-context, created by FcaBedrock, visualised in Con-Exp

### 7.3 Sheffield Parking Tickets Dataset

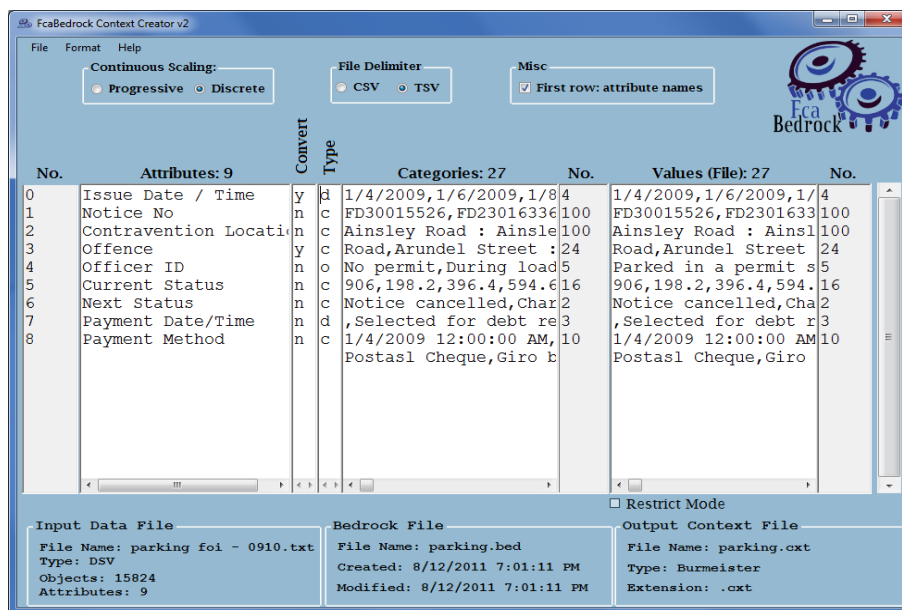
This dataset is provided by the City Council of Sheffield (United Kingdom) and contains raw data of parking tickets issued in the name of the council within the last five year period, broken down by date, time, location, narrative of infraction, current status of case, next status of case, transaction details (i.e. date, time and payment methods) and the IDs of the officers issuing the tickets.

Let us say that we are interested in seeing what kind of offenses have taken place for the reporting period of April 2009 to August 2009. For this analysis to take place, the *Issue Date/Time* and *Offense Description* attributes were converted. The *Issue Date/Time* attribute was grouped into three ranges: April–June, June–August and greater than August. Furthermore, in order to avoid long labels in the lattice, the *Offense Description* attribute was shortened to *Offense*, and each offense description was given a shorter title than the ones appearing in the dataset; for example, “Parked in a permit space without displaying a valid permit” was renamed to “No permit”. These modifications were made directly in FcaBedrock without altering the original dataset whatsoever, by just modifying the names of the offense descriptions, as they would appear in the formal context, in the *Categories* window (Figure 16).

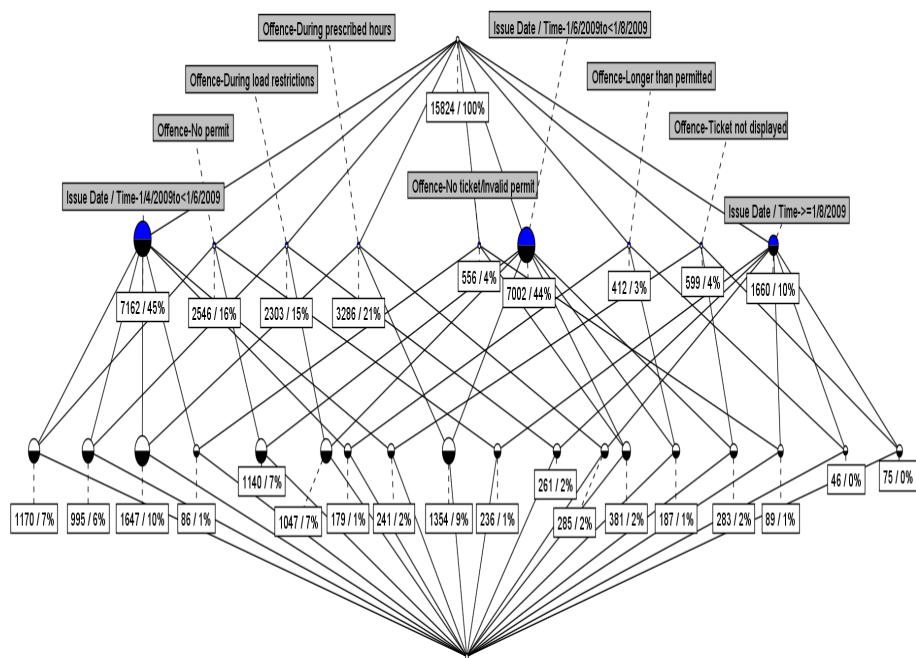
The resulting concept lattice is shown in Figure 17. Only the first 6 offenses are displayed – the rest have been deselected, in ConExp, to make the lattice more readable. The lattice shows that the most dominant of the six offenses is parking in a restricted street during prescribed hours, with a percentage of 21%, while the least dominant offense is parking for longer than permitted, with a percentage of 3%. The lattice also shows that a similar amount of parking tickets have been issued for the periods 1/4/2009–1/6/2009 and 1/6/2009–1/8/2009 (7162 and 7002 respectively). Only 1660 tickets have been issued from 01/08/2009 onwards, but it should be noted that these data only contain entries up to 10/8/2009. The same offense ‘popularity’ applies when observing each time period separately, apart from the 10-day reporting period of August, where the most popular offenses are evenly distributed between parking during prescribed hours, parking during load restrictions and parking without a permit.

## 8 Conclusion

Knowledge Discovery is possible through creating formal contexts and then visualising them. FcaBedrock has been used successfully to convert large datasets into formal contexts. Where the concepts in a formal context become far too many to visualise, FcaBedrock’s attribute restriction and exclusion features have been used to create sub-contexts of the same data, so as to focus the analysis on specific business questions and to make the resulting concept lattices readable and manageable. Furthermore, datasets containing different kinds of attribute have been used to demonstrate how FcaBedrock can handle datasets of different nature, although free-text data have proven to still be a problem and no means for converting them has been implemented yet. For converting free-text data,



**Fig. 16.** Parking Tickets *issue date-offense* sub-context, created by FcaBedrock. Note how the *Offense Description* attribute was renamed to *Offense* and how it's values in the *Categories* window were shortened, when compared to the values as they appear in the original data, in the *Values (File)* window.



**Fig. 17.** Parking Tickets *issue date-offense* sub-context, created by FcaBedrock, visualised in ConExp

further work is required, such as implementing a Natural Language Processing (NLP) technique, in order to identify values.

The tool is straightforward to use: the metadata required are usually easy to obtain (datasets usually come with descriptive documentation) and enter, but even without descriptive documentation, the auto-detection and repeat features can be used to obtain the initial metadata, after which the user can modify them as required. In terms of converting attributes, the five attribute types categorical, boolean, continuous, ordinal and dates are straightforward and easy to convert, although more means for dealing with attributes are being considered for the next version of FcaBedrock, such as giving the user the ability to create ranges for continuous attributes based on certain characteristics of the numerical data under investigation. For example, if during the auto-detection phase the tool determines that the attribute being read is logarithmic, it could let the user know that this is the case and suggest ranges suitable for demonstrating the true distribution of that attribute.

FcaBedrock is a versatile tool. It currently supports two standard input formats and more formats will be added to the next version of the tool, such as XML and the ability to read and query RDBMS data. The two output formats Burmeister and FIMI allow interoperability between various FCA tools, in order to avoid unnecessary and time-consuming conversions between FCA file formats.

Through a process of guided automation, FcaBedrock gives the user control over the conversion process to interpret data and create sub-contexts, to suit the needs of an analysis.

The further development of FcaBedrock will form a core part of the CUBIST project (see Acknowledgments, below). CUBIST aims to develop an approach for semantic and user-friendly Business Intelligence (BI) by augmenting semantic technologies with BI capabilities and providing conceptually relevant, user-friendly, visual analytics. To this end, FcaBedrock is currently being developed to accept RDF formats as input and pulling data directly from triple stores. In addition, the natural language processing element of CUBIST will make intelligence available from unstructured data sources. The project is also providing industrial use cases to demonstrate that the creation of formal contexts from industrial data can yield important business intelligence. The data provided by the use cases are large in volume (millions of objects, thousands of attribute values). Optimisations are taking place in FcaBedrock to significantly reduce the time required for parsing the data and converting them to formal contexts: faster methods for handling attributes with many attribute values, indexing the data using integers rather than strings, better algorithms for reading triples, and making the tool scalable to the processing cores of the environment it is running on, so as to make FCA of large-scale data less time-consuming and less resource-intensive.

## 9 Acknowledgments

This work is part of the CUBIST<sup>4</sup> project (“Combining and Uniting Business Intelligence with Semantic Technologies”), a research project funded by the European Commission’s 7th Framework Programme of ICT, under topic 4.3: Intelligent Information Management.

This article is an extended version of the paper with the same name presented at the second International Conference on Intelligent Networking and Collaborative Systems (INCoS) 2010 (Andrews and Orphanides, 2010b).

---

<sup>4</sup> <http://www.cubist-project.eu>

## Bibliography

- Abadi, D.J., Marcus, A., Madden, S.R. and Hollenbach, K. (2009) *SW-Store: a vertically partitioned DBMS for Semantic Web data management*. In: VLDB, vol. 18. Springer-Verlag. pp. 385–406.
- Andrews, S. (2011) *In-Close2, a High Performance Formal Concept Miner*. In: Proceedings of the 19th International Conference on Conceptual Structures (ICCS) 2011. LNAI 6828. Berlin: Springer-Verlag. pp. 50–62.
- Andrews, S. (2009a) *Data Conversion and Interoperability for FCA*. In: CS-TIW 2009, [http://www.kde.cs.uni-kassel.de/ws/cs-tiw2009/proceedings\\\_final\\\_15July.pdf](http://www.kde.cs.uni-kassel.de/ws/cs-tiw2009/proceedings\_final\_15July.pdf). pp. 42–49.
- Andrews, S. (2009b) *In-Close, a Fast Algorithm for Computing Formal Concepts*. In: Rudolph, Dau, Kuznetsov (Eds.): Supplementary Proceedings of ICCS'09, CEUR WS 483.
- Andrews, S. and Orphanides, C. (2010a) *FcaBedrock, a Formal Context Creator*. In: Croitoru, M., Ferre, S. and Lukose, D. (eds.) Conceptual Structures: From Information to Intelligence, Proceedings of the 18th International Conference on Conceptual Structures (ICCS) 2010. LNAI 6208. Berlin: Springer. pp. 181–184.
- Andrews, S. and Orphanides, C. (2010b) *Knowledge Discovery through Creating Formal Contexts*. In: Hill, R. (ed.) First International Workshop on Computational Intelligence in Networks and Systems (CINS 2010), in Xhafa, F., Demetriadis, S., Caballe, S., Abraham, A. (eds.) Second International Conference on Intelligent Networking and Collaborative Systems (INCoS) 2010. ISBN: 978-0-7695-4278-2/10. DOI 10.1109/INCOS.2010.53. IEEE Computer Society. pp. 455–460.
- Arevalo, G., Berry, A., Huchard, M., Perrot, G., Sigayret, A. (2007) *Performances of Galois Sub-hierarchy-building algorithms*. In: Kuznetsov, S.O., Schmidt, S. (eds), ICFCA 2007, LNAI 4390. Berlin: Springer-Verlag. pp. 166–180.
- Boulicaut, J-F. and Besson, J. (2008) *Actionability and Formal Concepts: A Data Mining Perspective*. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933. Berlin/Heidelberg: Springer-Verlag. pp. 14–31.
- Frank, A. and Asuncion, A. (2010) *UCI Machine Learning Repository*. [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California, School of Information and Computer Science.
- Ganter, B. (1984) *Two Basic Algorithms in Concept Analysis*. Technical Report FB4-Preprint No. 831, TH Marstadt.
- Ganter, B. and Wille, R. (1998) *Formal Concept Analysis: Mathematical Foundations*. Berlin: Springer-Verlag.
- Goethals, B. and Zaki, M. (2004) *Advances in Frequent Itemset Mining Implementations: Report on FIMI'03*. In: SIGKDD Explorations Newsletter, vol. 6(1). ACM New York. pp. 109–117.
- Ignatov, D.I. and Kuznetsov, S.O. (2009) *Frequent Itemset Mining for Clustering Near Duplicate Web Documents*. In: Rudolph, S., Dau, F., Kuznetsov, S.O. (eds.) ICCS 2009. LNCS (LNAI), vol. 5662. Berlin/Heidelberg: Springer-Verlag. pp. 185–200.

- Kaytoue-Uberall, M., Duplessis, S., and Napoli, A. (2008) *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. Berlin/Heidelberg: Springer-Verlag. pp. 439–449.
- Krajca, P., Outrata, J. and Vychodil, V. (2008) *Parallel Recursive Algorithm for FCA*. In: Belohlavek, R., Kuznetsov, S.O. (eds.) CLA 2008. Olomouc: Palacky University. pp. 71–82.
- Kuznetsov, S.O. (1999) *Learning of Simple Conceptual Graphs from Positive and Negative Examples*. In: Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery. Lecture Notes in Computer Science, Vol. 1704. London: Springer-Verlag. pp. 384–391.
- Kuznetsov, S.O. and Obiedkov, S.A. (2002) *Comparing Performance of Algorithms for Generating Concept Lattices*. In: Journal of Experimental and Theoretical Artificial Intelligence, vol. 14. pp. 189–216.
- Poelmans, J., Elzinga, P., Dedene, G., Viaene, S. and Kuznetsov, S. O. *A Concept Discovery Approach for Fighting Human Trafficking and Forced Prostitution*. In: Andrews, S., Polovina, S., Hill, R. and Akhgar, B. (eds.) ICCS 2011. LNCS (LNAI), vol. 6828, pp. 201–214.
- Priss, U. (2008) *Formal Concept Analysis in Information Science*. In: Cronin, B. (ed.) Annual Review of Information Science and Technology, ASIST, vol. 40.
- Rioul, F., Robardet, C., Blachon, S., Crémilleux, B., Gandrillon, O. and Boulicant, J-F. (2003) *Mining concepts from large SAGE gene expression matrices*. In: Boulicant, J-F., Dzeroski, S. (eds.) WKDID 2003. Catat-Dubrovnik, Croatia: Catat-Dubrovnik. /hpp. 107–118.
- Rohloff, K., et al. (2007) *An Evaluation of Triple-Store Technologies for Large Data Stores*. In: On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, LNCS vol. 4806/2007, pp 1105-1114.
- Sawase, K., Nobuhara, H. and Bede, B. (2009) *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. Berlin/Heidelberg: Springer.
- Wille, R. (2005). *Formal Concept Analysis as Mathematical Theory of Concepts*. In: Ganter, B., Stumme, G. and Wille, R. (eds.) Formal Concept Analysis: Foundations and Applications. Berlin: Springer. pp. 1–6.
- Wolff, K.E. (1993). *A First Course in Formal Concept Analysis*. Available: [http://www.fbmn.h-da.de/home/wolff/Publikationen/A\\_First\\_Course\\_in\\_Formal\\_Concept\\_Analysis.pdf](http://www.fbmn.h-da.de/home/wolff/Publikationen/A_First_Course_in_Formal_Concept_Analysis.pdf). Last accessed 05 September 2011.
- Yevtushenko, S.A. (2000) *System of data analysis "Concept Explorer"* (In Russian). In: Proceedings of the 7th National Conference on Artificial Intelligence KII-2000, Russia. pp. 127–134.
- Zaki, M.J. and Hsiao, C-J. (2005) *Efficient Algorithms for Mining Closed Itemsets and Their Lattice Structure*. In: IEEE Transactions on Knowledge and Data Mining. 17(4), IEEE Computer Society.