Relation Algebra Operations on Formal Contexts

Uta Priss

Edinburgh Napier University, School of Computing,
www.upriss.org.uk

Abstract. This paper discusses the use of relation algebra operations on formal contexts. These operations are a generalisation of some of the context operations that are described in the standard FCA textbook (Ganter & Wille, 1999). This paper extends previous research in this area with respect to applications and implementations. It also describes a software tool (FcaFlint) which in combination with FcaStone facilitates the application of relation algebra operations to contexts stored in many formats.

1 Introduction

Relation algebra (RA) operations provide a generalisation of some of the context operations that are described in the standard Formal Concept Analysis (FCA) textbook (Ganter & Wille, 1999). Using relation algebra is of interest because it provides an alternative to SQL-based conceptual modelling. Currently, some FCA software tools allow users to perform a limited number of context operations, for example, generating the dual context which switches objects and attributes. But the complete set of RA operations did not use to be available. Furthermore, software exists[1] which allows to extract formal contexts from relational databases via SQL queries. But FCA users who are not relational database experts may find it difficult to translate their natural language queries into SQL. In many applications, users therefore have to manually edit their formal contexts until they have the desired format and FCA visualisations can be applied. There is software for RA operations[2], but this is not designed for FCA and cannot easily be used for formal contexts. As far as we know up to now there has not been an extensive analysis of RA operations on FCA contexts and there has not been a software tool that implements RA operations systematically. This situation is changed by the new FcaFlint software[3] which provides an interface for performing RA operations directly on formal contexts stored in a variety of formats.

This paper starts with an introduction to relation algebra (Section 2). It then continues with an exploration of using RA for formal contexts (Section 3) and context schemata (Section 4). Section 5 considers RA for the modelling of lexical databases. Previously, the use of relation algebra for the modelling of lexical databases has been described (Priss & Old, 2006). Because, lexical databases are usually quite large, it is

[1] Tupleware available at http://tockit.sourceforge.net/

[2] http://www.informatik.uni-kiel.de/~progsys/relview.shtml

[3] FcaFlint will be bundled with the next edition of FcaStone available at http://fcastone.sourceforge.net/

not practical to generate the lattice of a formal context that contains all of the data of the lexical database. Thus mechanisms must be provided that allow for the extraction of meaningful smaller contexts. Furthermore over time, users of lexical databases may want to extract different types of formal contexts, thus the extraction of formal contexts must be achieved in a fairly flexible manner. Relation algebra provides a useful framework for modelling the context extraction for such databases. Results presented in previous research (Priss & Old, 2006) are extended and elaborated in this paper.

The paper continues (in Section 6) with a discussion of the FcaFlint software tool which in combination with FcaStone facilitates the application of RA operations to contexts stored in a variety of formats. FcaFlint allows to enter RA queries in a textual format. The implementation of FcaFlint is based on the description of a theoretical means for establishing an RA for formal contexts by Priss (2006). But the RA operations described in that paper are more theoretically than practically useful because they are inefficient (and to some degree incomplete). For practical implementations some modifications are needed as discussed in Section 7.

2 Relation algebra

Relation algebra was invented and developed by Peirce, Tarski and others. Priss (2006) discusses the modelling of FCA with RA and provides some background and references which shall not be repeated in this paper. An introduction to using RA with FCA, which contains detailed examples of the operations is available elsewhere[4]. RA can serve as an alternative to the more common modelling of FCA operations with Peirce Algebraic Logic (PAL) or Relational Algebra[5] (RLA). The difference between RA and RLA is that RA operates on binary relations while RLA operates on n-ary relations. Both RA and RLA can be used for the modelling of many-valued contexts or power context families. Since concept lattices correspond to binary relations, at the visualisation stage a binary relation must be extracted from the many-valued data, for example, by using the process of conceptual scaling (Ganter & Wille, 1999). The difference is that with RLA the binary context is extracted at the end after the context operations have been applied to n-ary relations, whereas with RA (with the addition of a Fork algebraic operation) n-ary relations are encoded in a binary format from the start. The details of this process are described by Priss (2006). In any case, both RA and RLA are relevant for FCA and ideally there should be FCA software for RA and RLA operations. This paper focuses on RA operations, which can be seen as a generalisation of some of the context operations discussed by Ganter & Wille (1999).

A detailed formal notation of FCA with RA is quite complex because all operations need to consider the sets of formal objects, attributes and the matrices of the contexts. In this paper, we simplify the notation and focus mostly on the Boolean matrices representing the binary relations of the contexts. It is assumed that the operations are applied

[4] An "Introduction to using Relation Algebra with FCA" can be downloaded from: http://www.upriss.org.uk/fca/relalg.html

[5] The similarity in the names of Relation versus Relation**al** Algebra is unfortunate, but these are the names that are established in the literature. RLA is Codd's algebra for relational databases.

sensibly with respect to the sets of formal objects and attributes. For example, in the case of union and intersection, the matrices need to have the same dimensions.

Definition 1. *A matrix-RA is an algebra* $(R, \cup, ^-, one, \circ, ^d, dia)$ *where* R *is a set of square Boolean matrices of the same size;* one *is a matrix containing all 1s;* dia *is a matrix, which has 1s on the diagonal and 0s otherwise;* $\cup, ^-, \circ, ^d$ *are the usual Boolean matrix operations.* $\cap$ *and* nul *are defined as* $I \cap J := \overline{\overline{I} \cup \overline{J}}$ *and* $nul := \overline{one}$.

Boolean (or binary) matrices are elaborated by Kim (1982). Boolean matrix operations are exactly like normal matrix operations except that the matrices contain only 0s and 1s and use Boolean OR, AND, and NOT for the componential operations. For example, Fig. 1 illustrates relational composition. The operations $\subseteq$ and $=$ are as usual: $I \subseteq J :\Longleftrightarrow I \cap J = I$ and $I = J :\Longleftrightarrow I \subseteq J$ and $J \subseteq I$. It can be shown that a matrix-RA is an RA and fulfills all the axioms of an RA, such as $(R, \cup, \cap, ^-, nul, one)$ is a Boolean algebra; $\circ$ is associative and distributive with $\cup$; dia is a neutral element for $\circ$ (but unique inverse elements need not exist); $(I^d)^d = I$; $(I \cup J)^d = I^d \cup J^d$; $(I \circ J)^d = J^d \circ I^d$; and so on (cf. Priss, 2006). RA has the expressive power of First Order Logic (FOL) with three variables. For finite matrices a transitive closure of composition can be defined: $I^{trs} := I \cup (I \circ I) \cup (I \circ I \circ I) \cup ....$ But this is not an RA operation and not FOL.

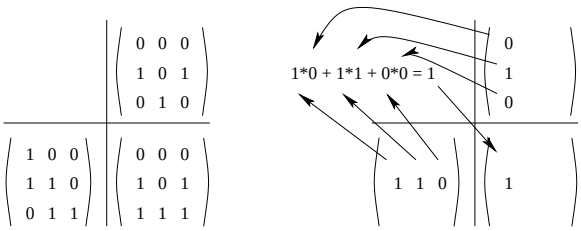

Fig. 1. Binary relational composition, $I \circ J$

Boolean matrices have many interpretations. If they are interpreted as binary relations, they can be used to check properties. A binary relation is:

- symmetric if $I = I^d$,
- reflexive if $dia \subseteq I$,
- transitive if $I \circ I \subseteq I$,
- antisymmetric if $I \cap I^d \subseteq dia$,
- a partial ordering if $I \cap I^d = dia$ and $I \circ I = I$,
- surjective if $dia \subseteq I^d \circ I$ (or I has at least one 1 per column),
- injective if $I \circ I^d \subseteq dia$ (or I has at most one 1 per column).

If Boolean matrices are interpreted as the incidence relation of a directed graph, the transitive closure I^{trs} shows the reachability for travelling along the graph's edges. The transitive closure $(I \circ I^d)^{trs}$ shows the connectedness in an undirected graph. The next section discusses Boolean matrices interpreted as formal contexts.

3 Formal contexts as Boolean matrices

Ganter & Wille (1999) discuss context constructions and operations, some of which are RA operations, some are not. In their Definition 30, the complementary context K^c ($\overline{I}$ in our notation) and the dual context are RA operations. Ganter & Wille write X and Ø instead of one and nul, respectively. Ganter & Wille's apposition and subposition are non-RA operations, but are frequently used with FCA (see the next section). Ganter & Wille's disjoint union is not an RA operation because it involves apposition and subposition.

As mentioned above, in order for RA formalisms to be meaningful for FCA, matrix operations should only be applied if the involved contexts have appropriate sets of objects and attributes. For example in the case of composition, the set of attributes of the left matrix should correspond to the set of objects of the right matrix. This means that the sets need to contain the same elements and need to be in the same linear order. In the rest of this paper, it shall always be assumed that the RA operations are applied sensibly without explicitly mentioning the details about the sets of objects and attributes. Furthermore, the RA operations are also applied to non-square matrices and it is silently assumed that the dimensions of the matrices are compatible as needed. The matrix dia always needs to be square, but nul and one adjust their dimensions to the matrices they are unioned or intersected with.

An interesting question, which as far as we know has not yet been discussed elsewhere, is to investigate the expressiveness of RA with respect to FCA. Because the number of concepts in a concept lattice tends to be different from the numbers of objects and attributes, but RA operations do not radically change the matrix dimensions, it is immediately apparent that it is not possible to generate concept lattices from formal contexts using RA operations. But, as will be shown below, it is possible to calculate basic FCA facts about the objects and attributes and their ordering using just RA operations.

This paper does not contain an introduction to FCA (which can be found in Ganter & Wille (1999)). But a few notions shall be mentioned here: for a formal context (G, M, I), the *prime operator* retrieves *intensions* of concepts if applied to sets of objects and *extensions* of concepts if applied to sets of attributes. For $G_1 \subseteq G$, $G_1' := \{m \in M \mid gIm \text{ for all } g \in G_1\}$. Dually, for $M_1 \subseteq M$, $M_1' := \{g \in G \mid gIm \text{ for all } m \in M_1\}$. The *plus operator* uses an EXISTS-quantifier instead of an ALL-quantifier: $G_1^+ := \{m \in M \mid gIm \text{ for one } g \in G_1\}$. An *object concept* γg is the smallest concept to which that object belongs. It is the concept which is labelled by the object in the usual graphical representation of a concept lattice. An *attribute concept* μm is the largest concept to which that attribute belongs. The *object order* $\leq_o$ of a concept lattice is an order on the set of objects derived from the order among the object concepts in the concept lattice: $g_1 \leq_o g_2 \iff \gamma g_1 \leq \gamma g_2$. The *attribute order* is defined analogously.

For a formal context (G, M, I), the following examples indicate what kind of information can be derived about the conceptual structure by using just RA operations (described here for the set of objects, but dually for the set of attributes):

1. Sets of objects that share at least one attribute with each other: $(G, G, I \circ I^d)$. This matrix is symmetric: $(I \circ I^d)^d = (I^d)^d \circ I^d = I \circ I^d$. Each row or column contains g^{++}. It represents applying the plus operator twice.
2. An equivalence relation on the objects based on the horizontal decomposition of the lattice (see Priss & Old, 2006): $(G, G, (I \circ I^d)^{trs})$.
3. Objects which have no attributes in common: $(G, G, \overline{I \circ I^d})$.
4. The object order: $(G, G, \overline{\overline{I} \circ I^d})$. Each column in this context shows the extension of the object concept of that object: g''. It represents applying the prime operator twice. The lattice of this context displays the object order. Fig. 2 shows an example.

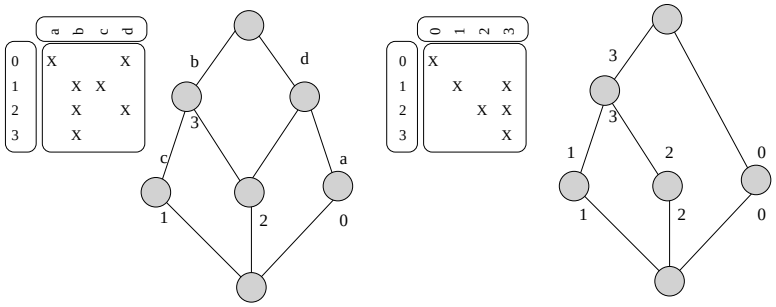

Fig. 2. A formal context (left) and its object order (right), $\overline{\overline{I} \circ I^d}$

Thus the RA operations are able to extract some information about the relationships among the objects (and attributes, respectively) but not the full lattice. Also, although the arrow relations (Ganter & Wille, 1999) are representable within a formal context, it is unlikely that these can be generated with RA operations because two different formal contexts with the same object orders can have different arrow relations. Thus, if RA can only extract the object and attribute orders, it is not sufficient to calculate the arrow relations.

4 Examples of context schemata

This section and the next one show some more examples of how RA operations can be used for FCA applications. Apposition and subposition are not RA operations, but they can be used to iteratively build a context from smaller contexts. A *context schema* consists of four (or nine etc) contexts built via apposition and subposition, such as the examples in Table 1 and Fig. 3. Quite often some of the contexts in a context schema are functionally dependent on other ones in the same schema. For example, in schemata 5 and 6 in Table 1 the context in the lower right corner is derived by composition. In general, the functions used for building context schemata are not restricted to RA operations. If the functions used for building a context schema are solely RA operations and

nul, one, dia, these are called *RA context schemata*. It might be an interesting research question, to examine the algebraic structure that is formed by RA context schemata.

Table 1 shows examples of RA context schemata. The simplest examples (1-3) are listed in Ganter & Wille (1999) and yield the horizontal and vertical sums and the direct product of concept lattices. Many other constructions of RA and non-RA context schemata are known (e.g. Ganter & Wille, Section 1.4).

Table 1. RA context schemata

Context schema	Effect on the lattices
1) $\begin{array}{c\|c} nul & I \\ \hline J & nul \end{array}$	Horizontal sum (Ganter & Wille, 1999).
2) $\begin{array}{c\|c} one & I \\ \hline J & nul \end{array}$	Vertical sum (Ganter & Wille, 1999).
3) $\begin{array}{c\|c} one & I \\ \hline J & one \end{array}$	Direct product (Ganter & Wille, 1999).
4) $\begin{array}{c\|c} dia & I \\ \hline nul & dia \end{array}$	All object and attribute concepts are turned into atoms/co-atoms.
5) $\begin{array}{c\|c} dia & J \\ \hline I & I \circ J \end{array}$	Composition of two lattices.
6) $\begin{array}{c\|c} dia & I^d \\ \hline I & I \circ I^d \end{array}$	Vertical gluing of a lattice and its dual.

The following examples demonstrate how classification can be represented with context schemata and RA operations. In Fig. 3, one context (J) is a classification on the attributes of another context (I). The relational composition $I \circ J$ means that the objects of the first context are classified with the attributes of the second context. This is an example of a context schema of type 5 from Table 1. In this case, the first context I contains types of living beings as objects and food sources as attributes. The second context J contains a classification of the food sources into food types. Using the dia matrix in the upper left corner has the effect that both lattices are glued together along the object-attribute concepts of the shared set. The lattice of J can be embedded join-preservingly into the combined lattice, which retains the attribute order of J. The lattice of I is meet-preservingly mapped and retains the object order of I. In the lattice diagram in Fig. 3, the concepts of I (except top and bottom) can be identified by their thicker node-borders. The concepts of J have been shaded in grey.

Wille proposed a different modelling of the relational composition[6] of two contexts, which represents a classification. Instead of the dia matrix, a context is used that represents the union of the object/attribute orders. Translated into RA notation, this context is $\begin{array}{c\|c} \overline{(\overline{I^d} \circ I)^d} \cup \overline{\overline{J} \circ J^d} & J \\ \hline I & I \circ J \end{array}$. Fig. 4 shows this lattice and a dual/invers construction, which was not mentioned by Wille. The difference between the left-hand side lattice in Fig. 4 and the lattice in Fig. 3 is that Fig. 4 maintains the object and attribute orders of I and J. This usually means that the lattice has fewer concepts.

[6] Rudolf Wille suggested this in a seminar presentation in 1995. I am not sure if this has ever been published.

	meat	French fries	milk	mice	gras	ANIMAL PRODUCT	VEGETARIAN FOOD	FAST FOOD	MEAT PRODUCT	PLANT-BASED
meat	×					×			×	
French fries		×					×	×		×
milk			×			×	×			
mice				×		×			×	
gras					×					×
human	×	×	×			×	×	×	×	×
vegetarian		×	×			×	×	×		×
cat	×		×	×		×	×		×	
eagle	×			×		×			×	
cow					×					×

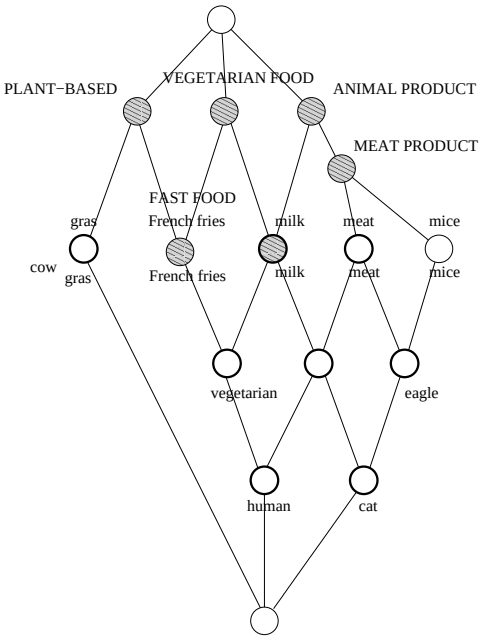

Fig. 3. An example of a context schema for relational composition: $\frac{dia \mid J}{I \mid I \circ J}$

The dual construction on the right-hand side assumes that the relational composition uses an implicit ALL-quantifier. In this case, an object of context I only belongs to a class (i.e. an attribute in J), if *all* of its attributes in I are in that class. In this case, neither the object concepts, nor the attribute concepts can be mapped using order-preserving mappings. The object and attribute concepts are still highlighted in the lattice diagram as before. Together the lattices in Fig. 4 show the differences in the interpretations of the classifications. For example, although humans eat some instances of each of the different food types (left lattice), humans eat all instances of fast food and vegetarian food in this context, but not all instances of meat, plant-based and animal products (right lattice).

The relationship between the EXISTS- and the ALL-quantifier is also explored in the next example, but in a single lattice. Fig. 5 shows another example of classification. In this case, a formal context (C, C, I) represents a classification (bird, mammal, vertebrate and songbird). Another context contains some instances (G, C, J) of that classification (penguin, sparrow, kiwi and bat). The instances have properties (egg-laying, wings, flightless, flippers and nocturnal) represented by a third context (G, M, K). The properties are generalised to the classes using an ALL-quantifier. A class has a property if all of its instances have that property $(C, M, \overline{(\overline{J} \circ I)^d} \circ K)$. The resulting concept lattice in Fig. 5 shows the wider and narrow senses of each class. This is indicated by the dashed lines for the class BIRD. The concepts under the attribute concept of a class name show the wider sense of the class which contains any instances that have some

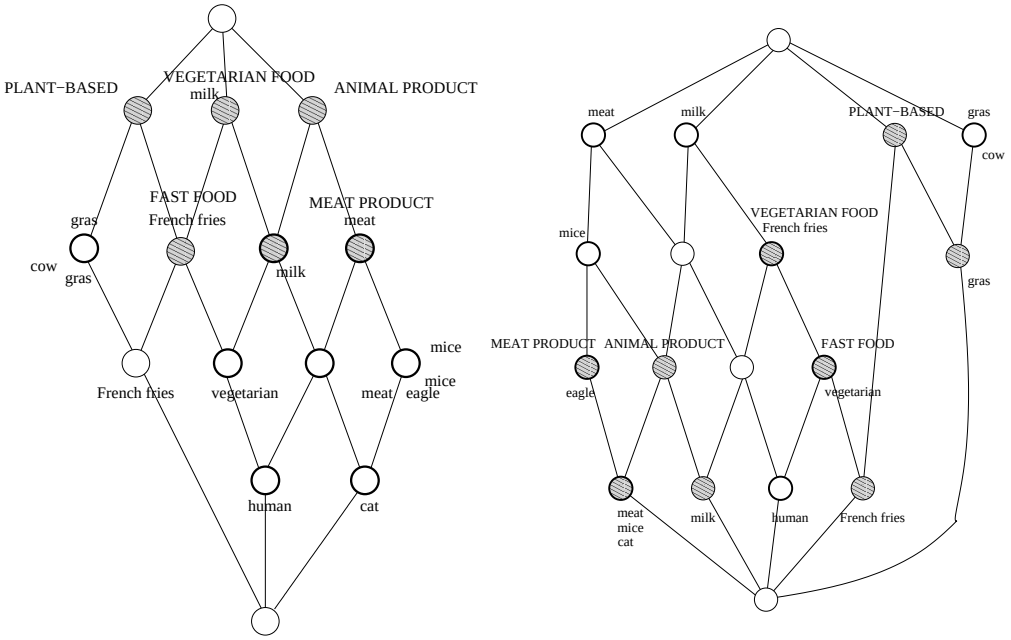

Fig. 4. Left-hand side: $\dfrac{\overline{\overline{I}^{d} \circ I}^{d} \cup \overline{\overline{J} \circ J^{d}} \;\Big|\; J}{I \;\Big|\; I \circ J}$. Right-hand side: $\dfrac{I^{d} \circ I \cap J \circ J^{d} \;\Big|\; J}{I \;\Big|\; \overline{\overline{I} \circ J}}$

	BIRD	MAMMAL	VERTEBRATE	SONGBIRD	egg laying	wings	flightless	flippers	nocturnal
bird	×		×		×	×			
mammal		×	×			×			×
vertebrate			×						
songbird	×		×	×	×	×			
penguin	×		×		×		×	×	
sparrow	×		×	×	×	×			
kiwi	×		×		×	×	×		×
bat		×	×			×			×

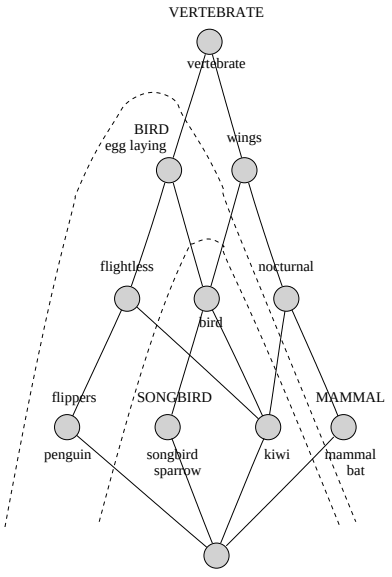

Fig. 5. The wider and narrower senses of a class, $\dfrac{I \;\Big|\; \overline{\overline{J} \circ I}^{d} \circ K}{J \;\Big|\; K}$.

of the class attributes. The concepts under the object concept of a class name show the narrower sense of the class which contains only instances which have properties common to all instances in the class. The prototypical instances of each class are given by $\overline{(\overline{J} \circ I)^d}$; in this case: the prototypical bird and songbird is sparrow; the prototypical mammal is bat. It is not claimed that this construction is novel (because it is similar to common FCA-models of Fuzzy and Rough Set Theory), but it is demonstrated how this can easily be expressed with RA operations.

In this example in Fig. 5, if each class has at least one instance that is distinguished from all other instances, then I is the attribute order of J, i.e. $I = \overline{(\overline{J^d} \circ J)^d}$. Turning this argument around, if the set of attributes in a formal context can be partitioned into a set of class C of type indicators and a set M of other attributes (i.e., the context is of the form $(G, C \cup M, J|K)$), then the narrower and wider senses can be calculated for these classes using $\begin{array}{c|c} \overline{(\overline{J^d} \circ J)^d} & \overline{\overline{\overline{J^d} \circ J} \circ \overline{J^d} \circ K} \\ \hline J & K \end{array}$.

5 RA for modelling of neighbourhood lattices in lexical databases

Lexical databases are usually too large to be visualised as a single concept lattice. For example, a lexical database, such as WordNet or Roget's Thesaurus, contains more than 100,000 words. Therefore techniques are required that allow to extract smaller formal contexts from the lexical database. Priss & Old (2006) discuss the use of neighbourhood lattices, which use the plus operator to extract a neighbourhood around a word in Roget's Thesaurus. Starting with one word (or small set of words), all senses of this word are extracted, and then all other words that have the same senses. Priss & Old (2006) show how this process can be modelled with RA operations.

Simple neighbourhood lattices are not always the best way to represent the data because they can be too broad and include too many unrelated words. Simple neighbourhood lattices tend to include homographs which are words that have the same spelling, but are otherwise unrelated, such as "lead" as a verb and as a metal. A mechanism for excluding homographs is to use *restricted neighbourhood lattices*. These were invented by Old and shown by Priss & Old (2008) to be the most useful types of concept lattices for Roget's Thesaurus compared to other methods of reducing the complexity of concept lattices. Independently, a lattice-based model of Roget's Thesaurus was invented in the 1950's by Margaret Masterman and shown to be very similar to restricted neighbourhood lattices by Priss & Old (2009). Masterman's work precedes FCA and does not use formal contexts or concept lattices. Her algorithms are not mathematically formulated, but described semi-formally and with examples. Thus, the relationship between her work and FCA is not entirely obvious without careful analysis and was only recently discovered (Priss & Old, 2009). Nevertheless, the fact that Old and Masterman independently and starting from different theoretical backgrounds discover a similar method for modelling Roget's Thesaurus, seems to indicate that this method is appropriate for its domain.

Because Priss & Old (2006) describe neighbourhood lattices with RA, but not restricted ones, the RA modelling of restricted neighbourhood lattices is added here. For a context (G, M, I), neighbourhoods of objects are computed as $(I \circ I^d) \circ (I \circ I^d) \circ ...$,

and for attributes as $(I^d \circ I) \circ (I^d \circ I) \circ ....$ Each additional $(I \circ I^d)$ corresponds to applying the plus operator two more times. The resulting matrices are symmetric. Thus the rows or columns of these matrices show which objects (or attributes) belong to the same neighbourhood at that stage. The plus operator is not a closure operator, but the transitive closure can be formed. The transitive closures $(I \circ I^d)^{trs}$ and $(I^d \circ I)^{trs}$ show the neighbourhood closures for the sets of objects and attributes, respectively. If all objects are connected to all attributes via transitive closure then both matrices are one. Otherwise, the matrices show equivalence relations on the objects and attributes. Priss & Old (2006) show that $(I \circ I^d)^{trs} \circ I = I \circ (I^d \circ I)^{trs}$. This is a matrix which contains the neighbourhood closures of the objects as columns and of the attributes as rows.

A restricted neighbourhood context of degree n means that instead of "at least one" as for the plus operator, an object must have at least n attributes: $G_1^{(I,\geq n)} := \{m \in M \mid gIm \text{ for at least } n \text{ elements } g \in G_1\}$. For attributes: $M_1^{(I,\geq n)} := \{g \in G \mid gIm \text{ for at least } n \text{ elements } m \in M_1\}$. We use the notation $\circ^{\geq n}$ to express that at least n elements need to be in common in the composition. Thus, $I \circ^{\geq 2} I^d$ shows objects that have at least two attributes in common. It is not possible to model this with just RA operations because this is equivalent to an FOL expression with more than 3 variables: $\exists_{a,b,x,y} : (a,b),(a,x),(b,x),(a,y),(b,y), x \neq y$. It is possible to express the condition that each row or column must have at least two 1s using RA: $((I \circ \overline{dia} \circ I^d) \cap dia) \circ I$ selects rows which have at least two 1s. $(((I \circ \overline{dia} \circ I^d) \cap dia) \circ I) \cap (I \circ ((I^d \circ \overline{dia} \circ I) \cap dia)) = (I \circ \overline{dia} \circ I^d) \circ I \circ (I^d \circ \overline{dia} \circ I)$ keeps only those 1s from the matrix I which belong to rows with at least two 1s and columns with at least two 1s. This is the form of a restricted neighbourhood context which has been shown to be useful for Roget's Thesaurus (Priss & Old, 2008).

6 FcaFlint

The FcaFlint software, which will be bundled with the next edition of FcaStone, implements all of the RA operations discussed in this paper. The operations are applied to a context stored in an input file (e.g. input.cxt) and the result is saved in a new file (e.g. output.cxt). The default format of the contexts is the Burmeister format, but in combination with FcaStone, any context format can be used that is supported by FcaStone. The RA operations are entered as functions. For example, $\overline{I} \circ I^d$ is executed from the command-line as:

```
fcaflint 'compos(invers(input.cxt),dual(input.cxt))' output.cxt
```

The matrices $one, dia, nul, \overline{dia}$ are entered as "<ONE>","<DIA>","<NUL>" and "<AID>". The dimensions of these matrices are automatically determined where possible. For example, in a union or intersection, one will use the same dimensions as the matrix it is unioned or intersected with. FcaFlint also provides the non-RA operations apposition, subposition, equality and transitive closure of composition. The $one, dia, nul, \overline{dia}$ matrices can be used for apposition and composition with other matrices, but not in combination with each other. This is because in that case, the dimensions of the matrices would be unknown. The 'equal()' function determines whether two

matrices are equal. The 'trans()' function calculates the transitive closure of a matrix with respect to composition ($\circ$). The composition function also implements the (non-RA) operations of requiring at least n values to be shared in the comparison (written as $\circ^{\geq n}$ in the previous section).

Because context schemata are used frequently in FCA-based modelling, FcaFlint should be useful in many applications. FcaFlint has been tested on matrices of sizes of up to 50×400. It returns reasonably fast results, with the exception of the transitive closure function, which should only be used for smaller matrices. It should be stressed that FcaFlint is not aimed at end users (because using relation algebra requires some expertise) but is meant as an intermediate representation - as a formal language for conceptual modelling. In the future, it is intended to produce a graphical interface that allows users to enter certain natural language queries which are then translated internally into a relation algebra representation used for further processing.

7 Implementing RA operations for FCA contexts

According to Definition 1, square Boolean matrices form an RA, but FCA contexts are not usually square (i.e. have $G = M$.) Non-square matrices cannot form an RA because, for example, union and intersection require the matrices to have the same dimensions. Furthermore, the nul, one and dia matrices need to change their dimensions depending on which matrices they are used with. But it can be shown that if RA operations are applied to non-square matrices of appropriate dimensions then they fulfill the RA axioms. Thus, non-square Boolean matrices almost form an RA and the RA operations can be meaningfully used with such matrices.

FcaFlint intends to implement two modes for FCA contexts. In the first mode, only the matrices of the contexts are considered, not the sets of objects and attributes; and the RA operations are just matrix operations. FcaFlint only checks whether matrices have appropriate dimensions. If yes, the operations are executed. Otherwise a warning is printed. In this mode it is up to the user of FcaFlint to make sure that the RA operations are actually meaningful with respect to the formal objects and attributes.

The second mode for FcaFlint attempts to implement RA operations which consider objects and attributes as well as the matrices. Priss (2006) describes context-RAs and context algebraic structures (CAS) as a means for using RA with FCA. A context-RA assumes an *active domain* which is a set of all objects and attributes of all contexts of an application. Each context is then transformed into a square, $|\mathcal{A}|$-dimensional matrix. A context-RA is a matrix-RA where each row and column corresponds to an element of the active domain. This context-RA is only of theoretical interest because it is not practically useful to create $|\mathcal{A}|$-dimensional matrices. Apart from the size problems, such matrices would need to be recalculated each time new data is added to an application.

Priss (2006) therefore also describes CAS as a means of implementing RA operations for FCA contexts in a more efficient manner. For example, in order to calculate the union of two contexts their sets of objects and attributes are unioned (using a normal union, not a disjoint union as suggested by Ganter & Wille (1999)). The problem with this approach is that a CAS does not form an RA and, in fact, if the operations are used

in combination with negation, the CAS operations may yield different results from the context-RA operations (which is undesirable).

The problem with the CAS operations is that because they enlarge contexts as needed, rows and columns may need to be added to a context. These rows and columns are usually filled with 0s, but if a context was previously negated once, they should be filled with 1s. The idea for FcaFlint is to consider both the *inside* of a formal context (which consists of the relation between objects and attributes that is currently defined for the context) and the *outside* of the context (which collects conditions about potential elements from the active domain that might be added to the context at a later stage). Only the inside is stored as a matrix. The outside is stored as a set of conditions (e.g., "all 0", "all 1") without having a complete list of which elements belong to the outside. This is much more efficient than using context-RAs and allows to add new elements to an active domain over time.

All contexts start out with their outside containing all 0s. If the context is once negated, the outside contains all 1s. For union and intersection, both inside and outside conditions contribute to the new inside of the context. It can be shown that the axioms of a Boolean algebra are fulfilled by union, intersection, negation, one, and nul applied to formal contexts as defined for CAS and by additionally keeping track of whether the outside of a context is filled with all 0s or all 1s.

Unfortunately, composition can change the outside of a context into conditions which are more complicated than "all 1" or "all 0". Thus, it is still not easy to create an RA in this manner. But the complexity of these conditions increases slowly. For many applications, the outside conditions of the contexts will be simple or irrelevant. For now, the approach that is taken with FcaFlint is to store simple conditions and to stop the program with a warning if the conditions are getting too complex. A warning would tell users that they need to manually check the sets of objects and attributes of their formal contexts and to verify whether the CAS operations that they are attempting to use are actually meaningful.

8 Conclusion

In summary, this paper presents a discussion of the use of RA operations on formal contexts. The paper was motivated by the development of the FcaFlint software which allows to apply RA applications directly to formal contexts stored in a variety of formats. With FcaFlint, RA operations can be used for conceptual modelling in the same way as RLA operations can be used using the Tupleware software[7]. But the implementation of FcaFlint highlighted some problems with previous models of RA/FCA that needed to be overcome. This paper also provides some insights into the expressivity of RA for formal contexts and discusses examples in the area of classification and lexical databases. A more systematic evaluation of RA operations on formal contexts might be of interest, but this is left for future research.

[7] `http://tockit.sourceforge.net/`

References

1. Ganter, Bernhard, & Wille, Rudolf (1999). *Formal Concept Analysis. Mathematical Foundations.* Berlin-Heidelberg-New York: Springer.
2. Kim, K. H. (1982). *Boolean Matrix Theory and Applications.* Marcel Dekker Inc.
3. Priss, Uta (2006). *An FCA interpretation of Relation Algebra.* In: Missaoui; Schmidt (eds.), Formal Concept Analysis: 4th International Conference, ICFCA 2006, Springer Verlag, LNCS 3874, p. 248-263.
4. Priss, Uta; Old, L. John (2006). *An application of relation algebra to lexical databases.* In: Schaerfe, Hitzler, Ohrstrom (eds.), Conceptual Structures: Inspiration and Application, Proceedings of the 14th International Conference on Conceptual Structures, ICCS'06, Springer Verlag, LNAI 4068, p. 388-400.
5. Priss, Uta; Old, L. John (2008). *Data Weeding Techniques Applied to Rogets Thesaurus.* In: Knowledge Processing in Practice. (In publication).
6. Priss, Uta; Old, L. John (2009). *Revisiting the Potentialities of a Mechanical Thesaurus.* In: Ferre & Rudolph (eds.), Proceedings of the 7th International Conference on Formal Concept Analysis, Springer Verlag. (In publication).