An Introduction to using Relation Algebra with FCA

Uta Priss

www.upriss.org.uk

Version 1.2, May 2009

Abstract. This paper provides an introduction to Relation Algebra, in particular, with respect to its use with FCA. It also serves as an introduction to several research papers I wrote on the subject.

1 Introduction

Relation Algebra (RA) and Relational Algebra (RLA) provide a foundation for query languages. While RLA is usually used for many-valued tables in relational databases (using SQL), RA is suitable for binary matrices as used in Formal Concept Analysis (FCA). RLA is more expressive than RA, but RA has some interesting features for the use with formal contexts.

I wrote several papers on using RA with FCA (see References). Because people keep sending me emails with questions about these papers, I decided to write an introduction to this topic, which provides more examples and explanations. This introduction is purely technical. I won't repeat any general comments and references because they can be found in my other papers on this topic.

2 Basic operations

RA can be defined in a purely axiomatic fashion and can be used with many different applications. For this paper, only the application to Boolean matrices is of interest. Thus, the operations are defined with respect to Boolean (or binary) matrices, which are matrices that only contain 0s and 1s.

Figure 1 shows some of the basic operations. Union $I \cup J$, intersection $I \cap J$, complement $\overline{I}$ and dual I^d are applied coordinate-wise. For example, for union, a coordinate of the resulting matrix is 1, if a coordinate in the same position of any of the original matrices is 1. For intersection, the resulting matrix has a 1 in positions where both intersected matrices have a 1. Complementation converts 0s into 1s and 1s into 0s. The dual of a matrix is a mirror image of the original matrix, mirrored along the diagonal. There are three special matrices: the matrix one contains just 1s; nul contains just 0s; and dia, the identity matrix, contains 1s along the diagonal, 0s otherwise.

Figure 2 shows the composition operation $I \circ J = K$, which is a form of relational composition or Boolean matrix multiplication. If one conducts this operation by hand, it is a good idea to write the matrices in a schema as shown in the middle of Figure 2. The example on the right shows how an individual coordinate is calculated. The coordinate in the ith row and jth column is calculated by using the ith row of the left matrix and jth

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \cup \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \cap \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\overline{\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}^{\text{d}} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{array}{ccc} \text{one:} & \text{nul:} & \text{dia:} \\ \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

Fig. 1. Union, intersection, complement, dual matrix; the one, nul and dia matrices

column of the right matrix. The individual coordinates are multiplied (using Boolean AND: $1 \times 1 = 1$; $1 \times 0 = 0 \times 1 = 0 \times 0 = 0$) and then added (using Boolean OR: $1 + 1 = 1 + 0 = 0 + 1 = 1$; $0 + 0 = 0$).

The bottom part of Figure 2 shows that non-square matrices can also be composed. But non-square matrices are not part of the usual RA definition.

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* $nul := \overline{one}$.

Non-square matrices do not form an RA, because these require:

Special rules for non-square matrices:

- For $\cup$ and $\cap$: the dimensions of the right matrix must be the same as the dimensions of the left matrix.
- For $\circ$: the number of columns in the left matrix must equal the number of rows in the right matrix.
- dia, one and nul refer to sets of matrices whose actual dimensions depend on the matrices and operations with which they are used.

A further operation called “transitive closure” is sometimes defined. It should be noted that when the expressivity of RAs is discussed, transitive closure is not part of

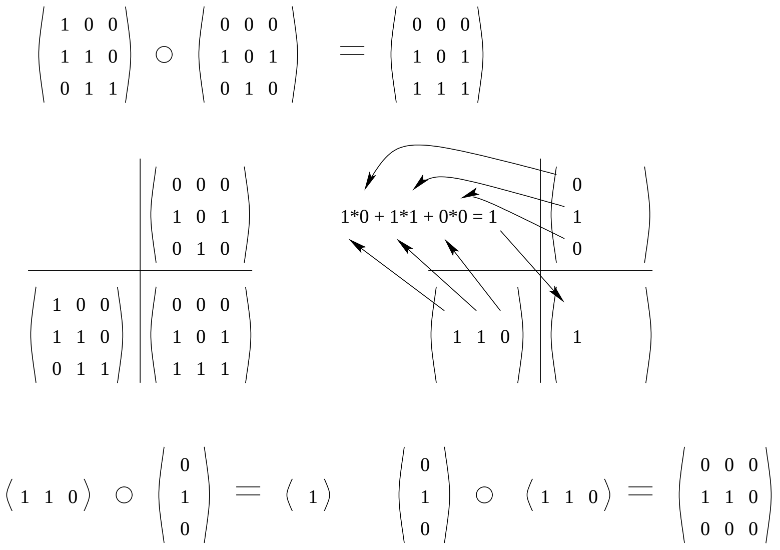


Fig. 2. Relational composition

RA because it cannot be expressed by the basic RA operations. This is because although it only uses $\cup$ and $\circ$ in its definition, the dots (...) in its definition indicate some sort of infinity, which cannot be expressed by the other operations. (The proof for this is well-known and beyond this paper.)

Figure 3 shows the transitive closure of the composition operation, which is defined as $I^{trs} := I \cup I \circ I \cup I \circ I \circ I \cup \ldots$. If the matrix I has 1s on the diagonal, I^{trs} is calculated by composing I with itself until it does not change anymore (as shown in the top half of Figure 3). If the matrix I does not have all 1s on the diagonal, I is still composed with itself until it does not change anymore, but I and the results at each stage are unioned (as shown in the bottom half of Figure 3).

3 Some properties of RA operations

For any RA, the substructure $(R, \cup, \cap, ^-, nul, one)$ is a Boolean algebra. This means that the operations $\cup, \cap, ^-$ have the same properties as union, intersection and complement for sets; and nul and one are like the empty set $\emptyset$ and its complement $\overline{\emptyset}$. The other operations that are used for sets ($\subseteq$, $\subset$, $=$, $\supseteq$, $\supset$) can be defined as usual: $I \subseteq J :\Longleftrightarrow I \cap J = I$ and $I = J :\Longleftrightarrow I \subseteq J, I \supseteq J$, and so on. Laws, such as de Morgan's law ($\overline{I \cap J} = \overline{I} \cup \overline{J}$), the associative, commutative and distributive laws are all valid as usual.

Below are some properties that apply to $\circ$ and ^{d}. The operation $\circ$ is not commutative ($I \circ J$ is not usually the same as $J \circ I$). The fourth property in the list below does not hold for $\cap$ (i.e., $(I \cap J) \circ K$ is not always the same as $I \circ K \cap J \circ K$). Properties 1-7 below can also be used as axioms for defining RAs (together with the Boolean algebra

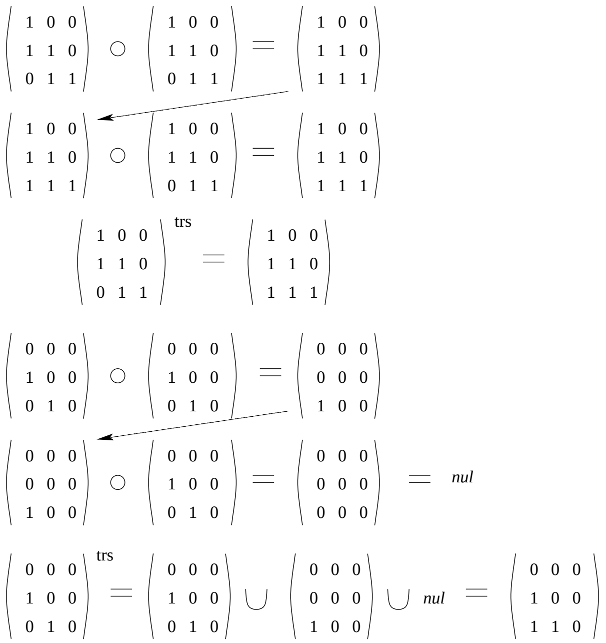


Fig. 3. Transitive closure

axioms). Properties 1-7 even hold for non-square matrices as long as the special rules listed after Definition 1 are observed.

1. $I \circ (J \circ K) = (I \circ J) \circ K$
2. $I \circ dia = I = dia \circ I$
3. $(I^d)^d = I$
4. $(I \cup J) \circ K = I \circ K \cup J \circ K$
5. $(I \cup J)^d = I^d \cup J^d$
6. $(I \circ J)^d = J^d \circ I^d$
7. $I^d \circ \overline{I \circ J} \subseteq \overline{J}$

4 Using RA with formal contexts

Formal Concept Analysis (FCA) uses the notion of a formal context $(\mathtt{G}, \mathtt{M}, I)$ which consists of a set `G`, a set `M` and a binary relation between `G` and `M` represented[1] by the Boolean matrix I. Figure 4 shows two formal contexts (K_I and K_J). The elements of `G` are called *(formal) objects*; the elements of `M` are called *(formal) attributes*. RA should

[1] Because sets can be encoded as matrices, typewriter font (`H`) is used to denote sets, italics (H) is used for matrices (but not in the figures). The matrices of formal contexts are written with crosses instead of 1s.

be applicable to the matrices of formal contexts, but because the matrices need not be square, this is not completely straightforward. Furthermore, the rows and columns in the matrices have interpretations: each row corresponds to an object; each column corresponds to an attribute. RA operations on formal contexts are only meaningful if they take these interpretations into consideration.

In FCA, concept lattices are produced from the formal contexts. This is not relevant for this paper, but it should be pointed out that the RA operations on contexts in general do not translate into the same operations on lattices. For example, a union of contexts does not produce a union of lattices. Some RA operations may not have any useful applications. In my opinion, the most useful RA operation for FCA is composition because it can be used for combining formal contexts and for expressing quantifiers as explained further below.

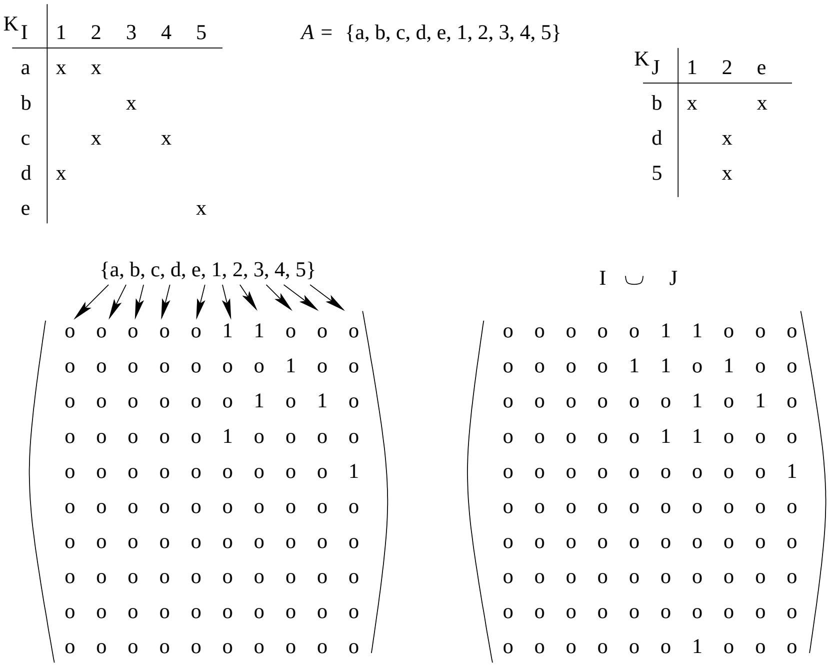
Because the use of RA operations for formal contexts is intuitive, but the construction of an RA for FCA is not completely straightforward, I developed two different suggestions to model this. The "unnamed perspective" is mostly of theoretical value because it makes it easy to form an RA. It would not be very efficient, however, to implement RA operations for FCA in that manner. The second suggestion is called the "named perspective" and describes a more practical approach that can be directly implemented in software, but which is more remote from the theoretical aspects of RAs. The terms "named" and "unnamed" are chosen in analogy to their use in relational database theory.

4.1 The unnamed perspective

Most FCA applications use not one, but many formal contexts which are often stored in a database. In the case of the unnamed perspective all objects and attributes of all of the contexts stored in a database are gathered into one linearly ordered set called an *active domain* $\mathcal{A}$ (in analogy to how this term is used in relational database theory). In Figure 4, the active domain of K_I and K_J is shown at the top. Even though $\mathcal{A}$ is just written as a set in Figure 4, the order must be fixed. In this case, the first element is a, the second element is b, and so on. It does not matter which order is chosen for $\mathcal{A}$, but the order must be unchanged while the RA operations are applied.

All matrices are then formed as square matrices based on $\mathcal{A}$ so that the first row and column in the matrix both correspond to the first element of the active domain and so on. This is shown for K_I, which is transformed into a matrix I based on $\mathcal{A}$ in the bottom left of Figure 4. Clearly, this is not a practically useful solution because the active domain of a fairly small database might already contain hundreds or more elements. Objects and attributes are both elements of $\mathcal{A}$. Therefore some of the rows of I in Figure 4 correspond to objects in K_I; some correspond to attributes in K_I. The bottom right of Figure 4 shows the union of I and J. (As mentioned before, these are theoretical constructs, not instructions for software implementations.)

Figure 5 shows how to calculate H' in the unnamed perspective. Sets and elements can be encoded in several ways: as rows, as columns or on the diagonal. The notation $sqr(\mathtt{N},\mathtt{H})$ (for "square") means that 1s are where the row position $\in \mathtt{N}$ and the column position $\in \mathtt{H}$. Using the active domain instead of $\mathtt{N}$ or $\mathtt{H}$ yields equal rows or columns. Along the diagonal, $dia(\mathtt{H})$ has a 1 where row position equals column position and $\in \mathtt{H}$.



K_I	1	2	3	4	5
a	x	x			
b			x		
c		x		x	
d	x				
e					x

A = {a, b, c, d, e, 1, 2, 3, 4, 5}

K_J	1	2	e
b	x		x
d		x	
5		x	

{a, b, c, d, e, 1, 2, 3, 4, 5}

$$\begin{pmatrix} o&o&o&o&o&1&1&o&o&o\\ o&o&o&o&o&o&o&1&o&o\\ o&o&o&o&o&o&1&o&1&o\\ o&o&o&o&o&1&o&o&o&o\\ o&o&o&o&o&o&o&o&o&1\\ o&o&o&o&o&o&o&o&o&o\\ o&o&o&o&o&o&o&o&o&o\\ o&o&o&o&o&o&o&o&o&o\\ o&o&o&o&o&o&o&o&o&o\\ o&o&o&o&o&o&o&o&o&o \end{pmatrix}$$

$I \cup J$

$$\begin{pmatrix} o&o&o&o&o&1&1&o&o&o\\ o&o&o&o&1&1&o&1&o&o\\ o&o&o&o&o&o&1&o&1&o\\ o&o&o&o&o&1&1&o&o&o\\ o&o&o&o&o&o&o&o&o&1\\ o&o&o&o&o&o&o&o&o&o\\ o&o&o&o&o&o&o&o&o&o\\ o&o&o&o&o&o&o&o&o&o\\ o&o&o&o&o&o&o&o&o&o\\ o&o&o&o&o&o&1&o&o&o \end{pmatrix}$$

Fig. 4. Union in the unnamed perspective

Matrices can easily be converted between these formats: $dia(\mathtt{S}) = sqr(\mathcal{A}, \mathtt{S}) \cap dia$; $sqr(\mathcal{A}, \mathtt{S}) = one \circ dia(\mathtt{S})$.

The ordering of the active domain is fixed and the same for the rows and columns of the matrices based on $\mathcal{A}$. This ordering is not changed by any RA operations. Forming the dual ^{d} is not a problem because rows and columns have the same order. There is no notion of "permutation" in this modelling. Usually, a composition with a matrix that contains at most one 1 per row and column is seen as a permutation in matrix theory because such a composition appears to change the order of rows or columns. But in the unnamed perspective, this is considered to only change the values within the matrix (which may not be useful for any applications). It does not alter the correspondence of the first element of the matrix to the first element of $\mathcal{A}$, and so on.

After the formal contexts have been transformed into matrices based on $\mathcal{A}$, all RA operations can be performed on these matrices. Thus, there is a natural definition of an RA for formal contexts:

Definition 2. *For an active domain $\mathcal{A}$, the context-RA based on $\mathcal{A}$ for a set of formal contexts is defined as the smallest matrix-RA based on $\mathcal{A}$ that contains these contexts.*

The reason why this perspective is called "unnamed" is because once the active domain has been determined, the names of the original objects and attributes can be completely omitted from any calculations. The correspondence between elements of

K_I	1	2	3	4	5
a	x	x			
b			x		
c		x		x	
d	x				
e					x

A = {a, b, c, d, e, 1, 2, 3, 4, 5}

H = {a, c}

$$\mathrm{dia}(H') = \mathrm{dia} \cap \overline{\mathrm{sqr}(A,H) \bigcirc \overline{I}}$$

$$\begin{array}{ccccc}
\begin{array}{cccccccccc}
\{a, & c\} & & & & & & & & \\
\downarrow & & \searrow & & & & & & & \\
\end{array} \mathrm{sqr}(A,H) & & \overline{I} & & \mathrm{sqr}(A,H) \bigcirc \overline{I} \\
\begin{pmatrix}
1 & o & 1 & o & o & o & o & o & o & o \\
1 & o & 1 & o & o & o & o & o & o & o \\
1 & o & 1 & o & o & o & o & o & o & o \\
1 & o & 1 & o & o & o & o & o & o & o \\
1 & o & 1 & o & o & o & o & o & o & o \\
1 & o & 1 & o & o & o & o & o & o & o \\
1 & o & 1 & o & o & o & o & o & o & o \\
1 & o & 1 & o & o & o & o & o & o & o \\
1 & o & 1 & o & o & o & o & o & o & o \\
1 & o & 1 & o & o & o & o & o & o & o
\end{pmatrix} & \bigcirc &
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & o & o & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & o & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & o & 1 & o & 1 \\
1 & 1 & 1 & 1 & 1 & o & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & o \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix} & = &
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & o & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & o & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & o & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & o & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & o & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & o & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & o & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & o & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & o & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & o & 1 & 1 & 1
\end{pmatrix}
\end{array}$$

$$\begin{array}{ccccc}
\mathrm{dia} & & \overline{\mathrm{sqr}(A,H) \bigcirc \overline{I}} & & \mathrm{dia}(H') \\
\begin{pmatrix}
1 & o & o & o & o & o & o & o & o & o \\
o & 1 & o & o & o & o & o & o & o & o \\
o & o & 1 & o & o & o & o & o & o & o \\
o & o & o & 1 & o & o & o & o & o & o \\
o & o & o & o & 1 & o & o & o & o & o \\
o & o & o & o & o & 1 & o & o & o & o \\
o & o & o & o & o & o & 1 & o & o & o \\
o & o & o & o & o & o & o & 1 & o & o \\
o & o & o & o & o & o & o & o & 1 & o \\
o & o & o & o & o & o & o & o & o & 1
\end{pmatrix} & \cap &
\begin{pmatrix}
o & o & o & o & o & o & 1 & o & o & o \\
o & o & o & o & o & o & 1 & o & o & o \\
o & o & o & o & o & o & 1 & o & o & o \\
o & o & o & o & o & o & 1 & o & o & o \\
o & o & o & o & o & o & 1 & o & o & o \\
o & o & o & o & o & o & 1 & o & o & o \\
o & o & o & o & o & o & 1 & o & o & o \\
o & o & o & o & o & o & 1 & o & o & o \\
o & o & o & o & o & o & 1 & o & o & o \\
o & o & o & o & o & o & 1 & o & o & o
\end{pmatrix} & = &
\begin{array}{cccccccccc}
 & & & & & \{1, & 2, & 3, & 4, & 5\} \\
 & & & & & & \uparrow & & & \\
\end{array}
\begin{pmatrix}
o & o & o & o & o & o & o & o & o & o \\
o & o & o & o & o & o & o & o & o & o \\
o & o & o & o & o & o & o & o & o & o \\
o & o & o & o & o & o & o & o & o & o \\
o & o & o & o & o & o & o & o & o & o \\
o & o & o & o & o & o & o & o & o & o \\
o & o & o & o & o & o & 1 & o & o & o \\
o & o & o & o & o & o & o & o & o & o \\
o & o & o & o & o & o & o & o & o & o \\
o & o & o & o & o & o & o & o & o & o
\end{pmatrix}
\end{array}$$

Fig. 5. Calculating $\mathtt{H}' = \{2\}$ in the unnamed perspective

$\mathcal{A}$ and rows and columns of matrices based on $\mathcal{A}$ is determined by the position of the elements, not by their names.

4.2 The named perspective

Because the unnamed perspective is theoretically straightforward, but not practically useful, another solution has to be found for practical implementations. The "named perspective" uses names of rows and columns for all its matrices. In other words, the matrices used in this perspective all belong to formal contexts. Figure 6 shows the same contexts and their union as in Figure 4, but this time according to the named perspective. In this perspective, the ordering of rows and columns can be changed while the names are in use (top half of Figure 6), but not while matrices are used (bottom half of Figure 6). Definition 3 shows the operations which are defined for formal contexts.

K_I	1	2	3	4	5
a	x	x			
b			x		
c		x		x	
d	x				
e					x

K_J	1	2	e
b	x		x
d		x	
5		x	

$K_I \sqcup K_J$	1	2	3	4	5	e
a	x	x				
b	x		x			x
c		x		x		
d	x	x				
e					x	
5		x				

$$\{a, b, c, d, e, 5\}\ \overset{\{1, 2, 3, 4, 5, e\}}{\begin{pmatrix} 1&1&0&0&0&0\\ 0&0&1&0&0&0\\ 0&1&0&1&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&1&0\\ 0&0&0&0&0&0 \end{pmatrix}} \sqcup \begin{pmatrix} 0&0&0&0&0&0\\ 1&0&0&0&0&1\\ 0&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&0&0&0&0\\ 0&1&0&0&0&0 \end{pmatrix} = \begin{pmatrix} 1&1&0&0&0&0\\ 1&0&1&0&0&1\\ 0&1&0&1&0&0\\ 1&1&0&0&0&0\\ 0&0&0&0&1&0\\ 0&1&0&0&0&0 \end{pmatrix}$$

Fig. 6. Union in the named perspective

Definition 3. *For formal contexts* $\mathcal{K}_1 := (\mathtt{G}_1, \mathtt{M}_1, I)$ *and* $\mathcal{K}_2 := (\mathtt{G}_2, \mathtt{M}_2, J)$*, the following context operations are defined:*
$\mathcal{K}_1 \sqcup \mathcal{K}_2 := (\mathtt{G}_1 \cup \mathtt{G}_2, \mathtt{M}_1 \cup \mathtt{M}_2, I \sqcup J)$ *with* $\mathtt{g} I \sqcup J \mathtt{m} :\Longleftrightarrow \mathtt{g} I \mathtt{m}$ *or* $\mathtt{g} J \mathtt{m}$
$\mathcal{K}_1 \sqcap \mathcal{K}_2 := (\mathtt{G}_1 \cup \mathtt{G}_2, \mathtt{M}_1 \cup \mathtt{M}_2, I \sqcap J)$ *with* $\mathtt{g} I \sqcap J \mathtt{m} :\Longleftrightarrow \mathtt{g} I \mathtt{m}$ *and* $\mathtt{g} J \mathtt{m}$
$\mathcal{K}_1 \diamond \mathcal{K}_2 := (\mathtt{G}_1, \mathtt{M}_2, I \diamond J)$ *with* $\mathtt{g} I \diamond J \mathtt{m} :\Longleftrightarrow \exists_{\mathtt{n} \in (\mathtt{M}_1 \cap \mathtt{G}_2)} : \mathtt{g} I \mathtt{n}$ *and* $\mathtt{n} J \mathtt{m}$
$\overline{\mathcal{K}_1} := (\mathtt{G}_1, \mathtt{M}_1, \overline{I})$
$\mathcal{K}_1^d := (\mathtt{M}_1, \mathtt{G}_1, I^d)$

The operations in Definition 3 can be used with all formal contexts. This is in contrast to the operations in Definition 4, which can only be applied in cases where special conditions are met (such as, $\mathtt{G}_1 = \mathtt{G}_2, \mathtt{M}_1 = \mathtt{M}_2$).

Definition 4. *The following additional operations for formal contexts are defined for formal contexts* $\mathcal{K}_1 := (\mathtt{G}_1, \mathtt{M}_1, I)$ *and* $\mathcal{K}_2 := (\mathtt{G}_2, \mathtt{M}_2, J)$*:*

1. $\mathcal{K}_1 \cup \mathcal{K}_2 := \mathcal{K}_1 \sqcup \mathcal{K}_2$ *if* $\mathtt{G}_1 = \mathtt{G}_2, \mathtt{M}_1 = \mathtt{M}_2$
2. $\mathcal{K}_1 \cap \mathcal{K}_2 := \mathcal{K}_1 \sqcap \mathcal{K}_2$ *if* $\mathtt{G}_1 = \mathtt{G}_2, \mathtt{M}_1 = \mathtt{M}_2$
3. $\mathcal{K}_1 \circ \mathcal{K}_2 := \mathcal{K}_1 \diamond \mathcal{K}_2$ *if* $\mathtt{M}_1 = \mathtt{G}_2$

Definition 3 provides a set-theoretic definition for $\sqcup, \sqcap, \diamond$, which can be translated into matrix-based operations as shown below and in the bottom half of Figure 6 because Definition 4 fulfills the "Special rules for non-square matrices" from page 2. In contrast to the unnamed perspective, the matrices used here are of minimal dimensions (according to $\mathcal{K}_1^*$ and $\mathcal{K}_2^*$ below). Rows and columns relate to subsets of $\mathcal{A}$, which are ordered according to $\mathcal{A}$.

- $\mathcal{K}_1 \sqcup \mathcal{K}_2 = \mathcal{K}_1^* \cup \mathcal{K}_2^*$ with $\mathcal{K}_1^* = (\mathtt{G}_1 \cup \mathtt{G}_2, \mathtt{M}_1 \cup \mathtt{M}_2, I); \mathcal{K}_2^* = (\mathtt{G}_1 \cup \mathtt{G}_2, \mathtt{M}_1 \cup \mathtt{M}_2, J)$.
- $\mathcal{K}_1 \sqcap \mathcal{K}_2 = \mathcal{K}_1^* \cap \mathcal{K}_2^*$ with $\mathcal{K}_1^* = (\mathtt{G}_1 \cup \mathtt{G}_2, \mathtt{M}_1 \cup \mathtt{M}_2, I); \mathcal{K}_2^* = (\mathtt{G}_1 \cup \mathtt{G}_2, \mathtt{M}_1 \cup \mathtt{M}_2, J)$.
- $\mathcal{K}_1 \diamond \mathcal{K}_2 = \mathcal{K}_1^* \circ \mathcal{K}_2^*$ with $\mathcal{K}_1^* = (\mathtt{G}_1, \mathtt{M}_1 \cup \mathtt{G}_2, I); \mathcal{K}_2^* = (\mathtt{M}_1 \cup \mathtt{G}_2, \mathtt{M}_2, J)$.

Figure 7 shows how basic FCA operations can be formed in the named perspective, calculating $\mathtt{c}' = \{2, 4\}$ and $\mathtt{H}' = \{2\}$ for $\mathtt{H} = \{a, c\}$. The resulting algebraic structure is described in the next definition.

Definition 5. *A context algebraic structure (CAS) based on* $\mathcal{A}$ *is an algebra that implements the context operations from Definition 3. (See Priss (2006) for the complete definition.)*

The active domain $\mathcal{A}$ is not needed for Definition 3, but it is needed if the operations are matrix-based as above because the linear order of $\mathcal{A}$ is used for ordering the objects and attributes of any context. A CAS is not an RA. There are no unique dia, one and nul matrices because these matrices need to change their dimensions and their sets of objects and attributes depending on what other matrices and operations they are used with. Furthermore, if negation is used in combination with composition, the results can be different from the ones in the unnamed perspective. There are ways to modify the CAS operations so that they yield an RA which is equivalent to a context-RA, but the details of that are beyond this introductory paper.

5 Eight quantifiers

Figure 7 shows how to calculate H' as $\overline{H^d \circ \overline{I}}$. Analogously, for $\mathtt{N} \subseteq \mathtt{M}$, $N' = \overline{\overline{I} \circ N^d}$, which retrieves objects from $\mathtt{G}$ that relate to all attributes in $\mathtt{N}$. The table below shows that there are seven other quantifiers formed similarly.

$\mathtt{N}^+$	$I \circ N^d$	at least one, some
$\mathtt{G} \setminus \mathtt{N}^+$	$\overline{I \circ N^d}$	none
$\mathtt{N}'$	$\overline{\overline{I} \circ N^d}$	relates to all
$\mathtt{G} \setminus \mathtt{N}'$	$\overline{I} \circ N^d$	does not relate to all
$(\mathtt{M} \setminus \mathtt{N})^+$	$I \circ \overline{N^d}$	relates to those that are not only
$\mathtt{G} \setminus ((\mathtt{M} \setminus \mathtt{N})^+)$	$\overline{I \circ \overline{N^d}}$	relates to those that are only
$(\mathtt{M} \setminus \mathtt{N})'$	$\overline{\overline{I} \circ \overline{N^d}}$	relates to all outwith
$\mathtt{G} \setminus ((\mathtt{M} \setminus \mathtt{N})')$	$\overline{I} \circ \overline{N^d}$	does not relate to all outwith

K_I	1	2	3	4	5
a	x	x			
b			x		
c		x		x	
d	x				
e					x

$$c' = c^d \circ I = (0\,0\,1\,0\,0) \circ \begin{pmatrix} 1\,1\,0\,0\,0 \\ 0\,0\,1\,0\,0 \\ 0\,1\,0\,1\,0 \\ 1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1 \end{pmatrix} = (0\,1\,0\,1\,0)$$

$$H := \{\,a, c\} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$H' = \overline{H^d \circ \overline{I}} = \overline{(1\,0\,1\,0\,0) \circ \overline{\begin{pmatrix} 1\,1\,0\,0\,0 \\ 0\,0\,1\,0\,0 \\ 0\,1\,0\,1\,0 \\ 1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1 \end{pmatrix}}} = \overline{(1\,0\,1\,0\,0) \circ \begin{pmatrix} 0\,0\,1\,1\,1 \\ 1\,1\,0\,1\,1 \\ 1\,0\,1\,0\,1 \\ 0\,1\,1\,1\,1 \\ 1\,1\,1\,1\,0 \end{pmatrix}} = \overline{(1\,0\,1\,1\,1)} = (0\,1\,0\,0\,0)$$

Fig. 7. Basic FCA operations in the named perspective

Figure 8 shows four of the quantifiers from the table above. The first example, $I \circ N^d$ retrieves all objects that relate to any of the attributes in N, which are a, c, and d. The second one, $\overline{\overline{I} \circ N^d}$, retrieves objects that related to both 1 and 2, which is only a. The third one, $I \circ \overline{N}^d$, retrieves objects that do not only relate to 1 and 2, which are b, c and e. The last one, $\overline{\overline{I} \circ \overline{N}^d}$, retrieves objects that relate to all attributes that are not in N, i.e., objects that relate to 3, 4 and 5, which is only e.

6 RA as a query language

Developing a full query language for relational databases from RA is quite complex because RA is restricted to binary relations. N-ary or many-valued relations cannot be represented with RA without the introduction of some further structures (for example, a Fork operation as described by Priss (2006)). A simpler solution described by Priss (2005) creates a separate binary relation for each key attribute/attribute pair. Figure 9 shows an example of a table with Employee data. A relational schema holds the data from all tables of a database (in this case only the Employee table). The formal context C_{Emp} is many-valued. A corresponding binary matrix I_{Emp} contains a 1 for every non-null value of C_{Emp} and a 0 for every null value (in this case a *one* matrix). The contexts V_{ename} and V_{eaddr} model the name and address attributes, respectively. These contexts were derived by using "nominal scaling", but any other of the standard FCA means for scaling many-valued contexts could be used as well.

The bottom half of Figure 9 calculates the RA equivalent of the RLA query "`select ename,eaddr from Emp where ename = 'mary' and eaddr = 'UK'`". Using RA this corresponds to $dia((V_{ename} \circ \underline{mary}^d) \cap (V_{eaddr} \circ \underline{UK}^d)) \circ I_{Emp} \circ dia(\underline{ename}^d \cup \underline{eaddr}^d)$ where $\underline{mary}$ is a row matrix indicating the position of "mary" in V_{ename}; similarly $\underline{UK}$ for V_{eaddr} and $\underline{ename}$, $\underline{eaddr}$ for I_{Emp}. The result is a matrix

K_I	1	2	3	4	5
a	x	x			
b			x		
c		x		x	
d	x				
e			x	x	x

N = { 1, 2} = (1 1 0 0 0)

$$I \bigcirc N^d = \begin{pmatrix} 1&1&0&0&0\\ 0&0&1&0&0\\ 0&1&0&1&0\\ 1&0&0&0&0\\ 0&0&1&1&1 \end{pmatrix} \bigcirc \begin{pmatrix}1\\1\\0\\0\\0\end{pmatrix} = \begin{pmatrix}1\\0\\1\\1\\0\end{pmatrix}$$

$$\overline{\overline{I} \bigcirc N^d} = \overline{\begin{pmatrix} 0&0&1&1&1\\ 1&1&0&1&1\\ 1&0&1&0&1\\ 0&1&1&1&1\\ 1&1&0&0&0 \end{pmatrix} \bigcirc \begin{pmatrix}1\\1\\0\\0\\0\end{pmatrix}} = \overline{\begin{pmatrix}0\\1\\1\\1\\1\end{pmatrix}} = \begin{pmatrix}1\\0\\0\\0\\0\end{pmatrix}$$

$$I \bigcirc \overline{N}^d = \begin{pmatrix} 1&1&0&0&0\\ 0&0&1&0&0\\ 0&1&0&1&0\\ 1&0&0&0&0\\ 0&0&1&1&1 \end{pmatrix} \bigcirc \begin{pmatrix}0\\0\\1\\1\\1\end{pmatrix} = \begin{pmatrix}0\\1\\1\\0\\1\end{pmatrix}$$

$$\overline{\overline{I} \bigcirc \overline{N}^d} = \overline{\begin{pmatrix} 0&0&1&1&1\\ 1&1&0&1&1\\ 1&0&1&0&1\\ 0&1&1&1&1\\ 1&1&0&0&0 \end{pmatrix} \bigcirc \begin{pmatrix}0\\0\\1\\1\\1\end{pmatrix}} = \overline{\begin{pmatrix}1\\1\\1\\1\\0\end{pmatrix}} = \begin{pmatrix}0\\0\\0\\0\\1\end{pmatrix}$$

Fig. 8. Some quantifiers

Employee table

key	ename	eaddr
e1	paul	UK
e2	mary	UK
e3	carl	USA
e4	sue	USA

a relational schema:

	Emp	eID	ename	eaddr
tEmp	1	1	1	1
e1	1	1	paul	UK
e2	1	2	mary	UK
e3	1	3	carl	USA
e4	1	4	sue	USA

C_{Emp}	eID	ename	eaddr
e1	1	paul	UK
e2	2	mary	UK
e3	3	carl	USA
e4	4	sue	USA

V_{ename}	paul	mary	carl	sue
e1	1			
e2		1		
e3			1	
e4				1

V_{eaddr}	UK	USA
e1	1	
e2	1	
e3		1
e4		1

$$\text{dia}\left(\begin{pmatrix} 1&o&o&o\\ o&1&o&o\\ o&o&1&o\\ o&o&o&1 \end{pmatrix} \bigcirc \begin{pmatrix}o\\1\\o\\o\end{pmatrix} \cap \begin{pmatrix} 1&o\\ 1&o\\ o&1\\ o&1 \end{pmatrix} \bigcirc \begin{pmatrix}1\\o\end{pmatrix}\right) = \text{dia}\left(\begin{pmatrix}o\\1\\o\\o\end{pmatrix} \cap \begin{pmatrix}1\\1\\o\\o\end{pmatrix}\right) = \begin{pmatrix} o&o&o&o\\ o&1&o&o\\ o&o&o&o\\ o&o&o&o \end{pmatrix}$$

$$\begin{pmatrix} o&o&o&o\\ o&1&o&o\\ o&o&o&o\\ o&o&o&o \end{pmatrix} \bigcirc \overset{I_{Emp}}{\begin{pmatrix} 1&1&1\\ 1&1&1\\ 1&1&1\\ 1&1&1 \end{pmatrix}} \bigcirc \overset{\text{dia(ename,eaddr)}}{\begin{pmatrix} o&o&o\\ o&1&o\\ o&o&1 \end{pmatrix}} = \begin{pmatrix} o&o&o\\ o&1&1\\ o&o&o\\ o&o&o \end{pmatrix} \qquad \text{mv}\left(\begin{pmatrix} o&o&o\\ o&1&1\\ o&o&o\\ o&o&o \end{pmatrix}\right) = \begin{pmatrix} o&o&o\\ o&\text{mary}&\text{UK}\\ o&o&o\\ o&o&o \end{pmatrix}$$

Fig. 9. Calculating $dia((V_{ename} \circ \underline{mary}^d) \cap (V_{eaddr} \circ \underline{UK}^d)) \circ I_{Emp} \circ dia(\underline{ename}^d \cup \underline{eaddr}^d)$

that has 1s in the positions of “mary” and “UK”. The $mv()$ function maps this onto a submatrix of C_{Emp} with the values “mary” and “UK”.

There are probably other ways of translating RLA into RA. I consider this topic to be unfinished research. It is not clear in what way, if at all, this could be practically implemented. I suspect users would find RA queries at least as difficult as RLA queries. Thus, some more user-friendly, maybe graphical representation would need to be found. On the other hand, there are useful applications of RA with respect to the modelling of lexical databases (as described in several of my papers). These applications do not develop query languages, but focus on developing relational schemata using relational composition that capture essential structures for a domain. Furthermore, relational schemata are used quite frequently under different names by other FCA authors as well.

7 References

I maintain two websites with further information, links to papers, software, etc:

- RA resources: `http://www.upriss.org.uk/fca/relalg.html`
- FCA website: `http://www.upriss.org.uk/fca/fca.html`

Priss, U. (2005). *Establishing connections between Formal Concept Analysis and Relational Databases.* In: Dau; Mugnier; Stumme (eds.), Common Semantics for Sharing Knowledge: Contributions to ICCS 2005, p. 132-145.

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.

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.

Priss, Uta (2009). *Relation Algebra Operations on Formal Contexts.* In: Proceedings of the 17th International Conference on Conceptual Structures, ICCS’09, Springer Verlag.

Priss, Uta (2009). *The FcaFlint Software for Relation Algebra Operations on FCA Contexts.* In: Conceptual Structures Tool Interoperability Workshop (CS-TIW 2009), Moscow, Russia.