The FcaFlint Software for Relation Algebra Operations on FCA Contexts

Uta Priss

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

Abstract. FcaFlint is a tool that implements context operations for FCA formal contexts. It is part of the FcaStone package. This paper provides an introduction to Relation Algebra operations as used by FcaFlint and discusses the command-line interface and implementation details of FcaFlint.

1 Introduction

FcaFlint[1] is a tool that implements context operations for FCA formal contexts. It can be used as a stand-alone tool (in which case the context must be formatted in the ".cxt" or "Burmeister" format), or it can be used in combination with FcaStone[2] (using any context format supported by FcaStone). The context operations are modelled using Relation Algebra (RA), which is an algebra that was invented by Peirce, Tarski and others and should not be confused with Codd's Relational Algebra (RLA). Both RA and RLA provide a foundation for query languages. While RLA is normally 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. Although some context operations (such as calculating dual contexts) are already provided by other FCA tools, as far as we know, none of the available tools implement a larger set of operations.

This paper provides a brief introduction to RA (Sections 2 and 3) and discusses FcaFlint's command-line options and the implementation of RA operations in FcaFlint (Section 4). The RA operations are described at a very basic level which assumes no prerequisite knowledge about matrix operations. The use of RA for FCA, in particular for linguistic applications has been described by Priss (2005), Priss (2006) and Priss & Old (2006). Those papers also provide more background and references which are omitted in this paper.

There are many examples in the FCA literature where contexts are constructed from other contexts in a systematic manner, often involving RA operations (although RA may not be explicitly mentioned). Without having RA software, each application that uses RA operations requires special purpose code written for the application or manual editing of the formal contexts. With FcaFlint, RA context constructions (and also some additional non-RA constructions) can be much more easily derived. Some examples

[1] FcaFlint will be bundled with the next edition of FcaStone.

[2] http://fcastone.sourceforge.net/

of using FcaFlint for context constructions are shown by Priss (2009). A very brief example of using RA as a query language is given at the end of Section 3 below. A more extended introduction to RA (which contains Sections 2 and 3 below, but has more details) can be found on-line[3].

2 Introduction to RA: 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 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. There are many ways to define RAs, abstractly or with respect to specific applications. The FCA-oriented RA definitions used in this paper are adapted from Priss (2006).

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.

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

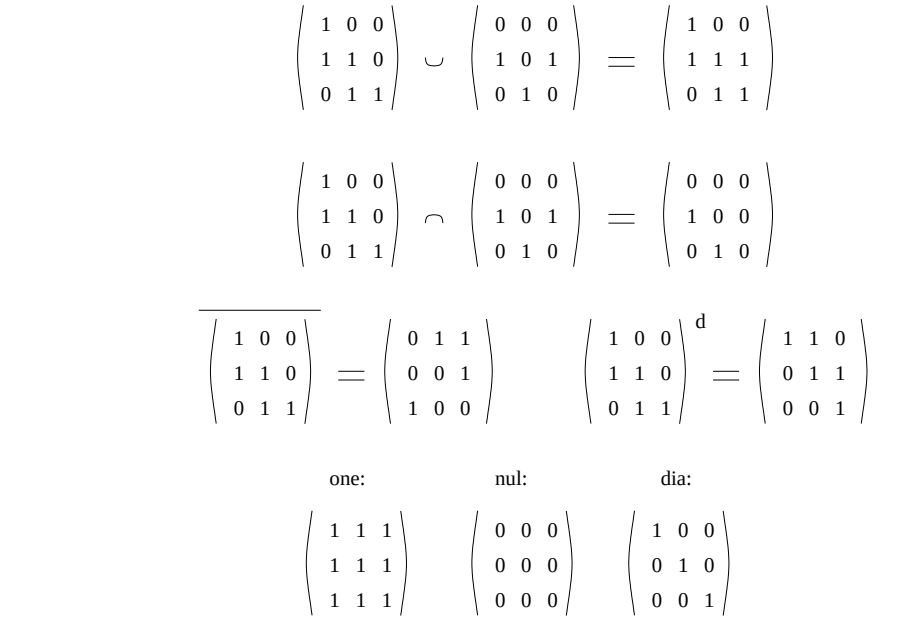

Fig. 1. Union, intersection, complement, dual matrix; the one, nul and dia matrices

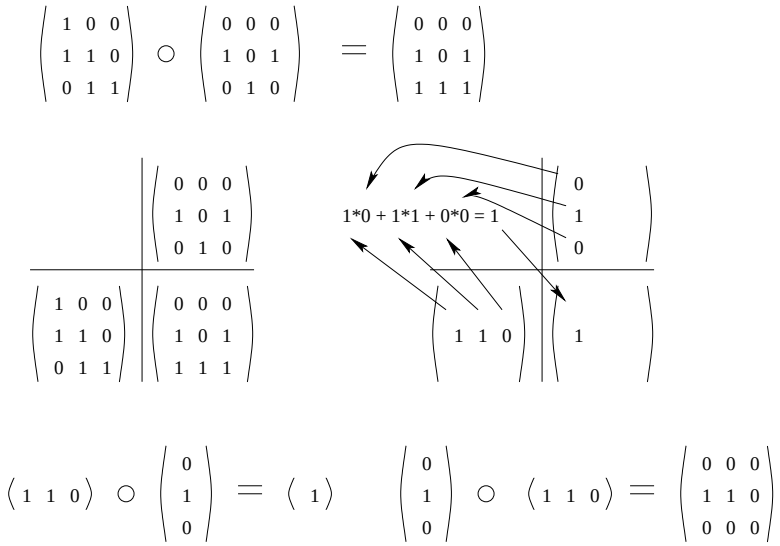

Fig. 2. Relational composition

– 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.

These special rules mainly refer to the theoretical definition of matrix-RAs. It is possible to extend the definition of union, intersection and composition to matrices of arbitrary dimensions (if one is not concerned about creating an RA). But because there is more than one way to extend these definitions, it needs to be carefully considered which operations are meaningful for a particular application. This is further discussed in the next section.

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 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 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[4] 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 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 for FCA.

Because the use of RA operations for formal contexts is intuitive, but the construction of an RA for FCA is not completely straightforward, Priss (2006) provides two

[4] 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.

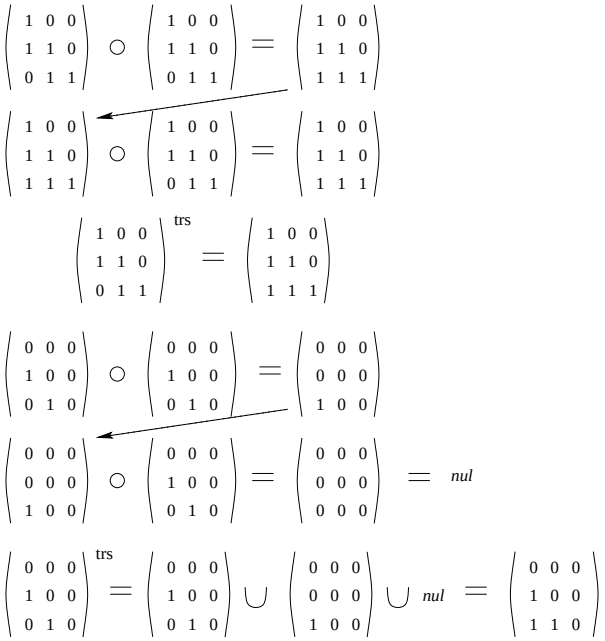

Fig. 3. Transitive closure

different suggestions to model this. The "unnamed perspective" is mostly of theoretical value because it makes it easy to form an RA, but is not practically useful. This perspective is called "unnamed" because objects and attributes are assigned to positions in a matrix, but their names are not used. Many 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* $\mathcal{A}$. The matrices of the contexts are transformed into $|\mathcal{A}|$-dimensional matrices. It would not be practically useful to actually implement RA operations for FCA in this manner. Therefore this perspective is not further discussed in this paper[5].

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. This perspective is discussed in the next section. The terms "named" and "unnamed" are chosen in analogy to their use in relational database theory.

[5] Further details on this perspective can be found in "Introduction to using Relation Algebra with FCA" available at: http://www.upriss.org.uk/fca/relalg.html

3.1 The named perspective

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 4 shows the union of contexts according to the named perspective. In this perspective, operations can be defined in a set-theoretic manner or in a matrix-based manner, where each row and column corresponds to a "named" object or attribute. The ordering of rows and columns can be changed while the names are explicit (top half of Figure 4), but not while matrices are used (bottom half of Figure 4). Definition 2 shows the set-theoretic definitions of context operations (according to Priss (2006)). Apart from complement and dual, these operations are different from Ganter & Wille's (1999) definitions, although they are similar.

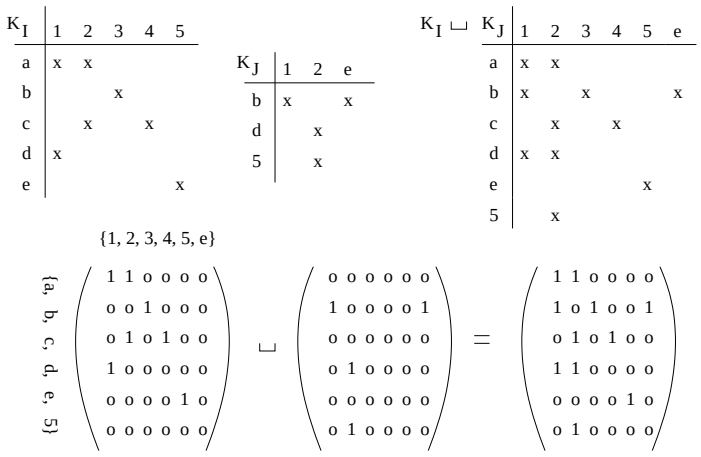

Fig. 4. Union in the named perspective

Definition 2. *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 2 can be used with all formal contexts. This is in contrast to the operations in Definition 3, 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 3. *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* $\mathtt{M}_1 = \mathtt{G}_2$

Using Definition 3, $\sqcup, \sqcap, \diamond$ can be translated into matrix-based operations as shown below and in the bottom half of Figure 4 because Definition 3 fulfills the "Special rules for non-square matrices". In contrast to the unnamed perspective, the matrices used here are of minimal dimensions (according to $\mathcal{K}_1^*$ and $\mathcal{K}_2^*$ below). The active domain $\mathcal{A}$ is used in this perspective as well but only in order to determine the order of the rows and columns: the objects and attributes corresponding to each matrix must be ordered in the same order as they appear in $\mathcal{A}$.

- $\mathcal{K}_1 \sqcup \mathcal{K}_2 = \mathcal{K}_1^* \cup \mathcal{K}_2^*$ 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 5 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 4. *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.)*

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, which is a problem because the unnamed perspective does form an RA. There are ways to modify the CAS operations so that they yield an RA which is equivalent to the unnamed perspective, but this is complicated. Basically, CAS operations enlarge contexts as needed, by adding rows and columns. These rows and columns are usually filled with 0s, but if a context was previously negated once, they should be filled with 1s.

It is possible to solve this by distinguishing 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.

All contexts start out with their outside containing all 0s. Negation changes the outside to "all 1s". Union and intersection may enlarge the inside, but the outside is still either "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.

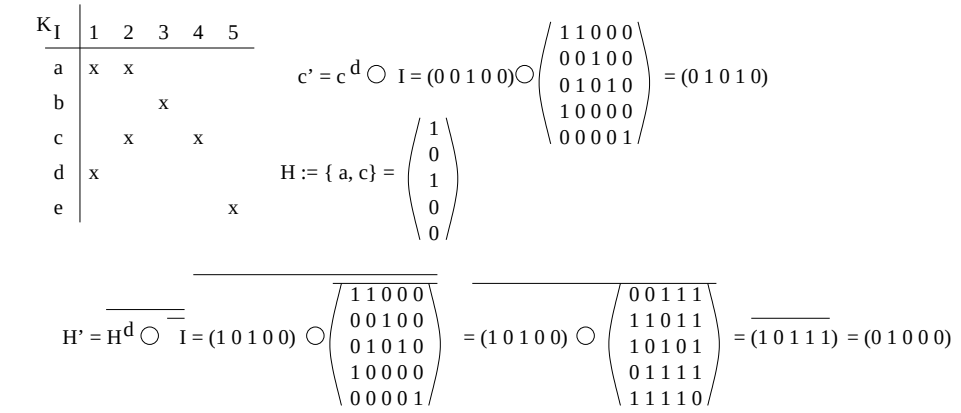

Fig. 5. Basic FCA operations in the named perspective

While it is beyond this paper to discuss applications in detail, Figure 6 provides a glimpse of how CAS can be used to model databases (Priss, 2005). In this example, a table with Employee data is translated into a relational schema, a many-valued context C_{Emp} and value scales V_{ename} and V_{eaddr}. 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 bottom half of Figure 6 calculates the RA equivalent of the RLA query "`select ename,eaddr from Emp where ename = 'mary' and eaddr = 'UK'`". In this modelling, $\underline{mary}$ is a row matrix indicating the position of "mary" in V_{ename}; similarly $\underline{UK}$ for $\overline{V_{eaddr}}$ and $\underline{ename}, \underline{eaddr}$ for I_{Emp}. The result is a matrix 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". This example shows how RA can be used as a query language, although whether this is practically useful still needs to be determined. Other, definitely useful applications are discussed in Priss (2009).

4 The implementation of RA operations in FcaFlint

The FcaFlint software implements all of the RA operations discussed in this paper. In the first version, the operations are implemented mostly as matrix operations as described in Section 2. The only checks that are implemented refer to the "Special rules for non-square matrices". (For example, the dimensions of $one, dia, nul, \overline{dia}$ are automatically determined where possible.) Furthermore, computing a dual matrix switches the object and attribute set and the result of composition selects the set of objects from the first matrix and the set of attributes from the second matrix. Otherwise, it is up to the user to make sure that the operations are meaningful with respect to formal contexts, i.e. that objects and attributes are ordered correspondingly and so on.

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 ap-

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

$$\mathrm{dia}\left(\left\langle \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix} \bigcirc \begin{pmatrix}0\\1\\0\\0\end{pmatrix} \right\rangle \cap \left\langle \begin{pmatrix}1&0\\1&0\\0&1\\0&1\end{pmatrix} \bigcirc \begin{pmatrix}1\\0\end{pmatrix} \right\rangle\right) = \mathrm{dia}\left(\begin{pmatrix}0\\1\\0\\0\end{pmatrix} \cap \begin{pmatrix}1\\1\\0\\0\end{pmatrix}\right) = \begin{pmatrix}0&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}$$

I_{Emp} dia(ename,eaddr)

$$\begin{pmatrix}0&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix} \bigcirc \begin{pmatrix}1&1&1\\1&1&1\\1&1&1\\1&1&1\end{pmatrix} \bigcirc \begin{pmatrix}0&0&0\\0&1&0\\0&0&1\end{pmatrix} = \begin{pmatrix}0&0&0\\0&1&1\\0&0&0\\0&0&0\end{pmatrix} \qquad \mathrm{mv}\left(\begin{pmatrix}0&0&0\\0&1&1\\0&0&0\\0&0&0\end{pmatrix}\right) = \begin{pmatrix}0&0&0&0\\0&\text{mary}&\text{UK}\\0&0&0&0\\0&0&0&0\end{pmatrix}$$

Fig. 6. Calculating $dia((V_{ename} \circ \underline{mary}^d) \cap (V_{eaddr} \circ \underline{UK}^d)) \circ I_{Emp} \circ dia(\underline{ename}^d \cup \underline{eaddr}^d)$
or: `select ename,eaddr from Emp where ename='mary' and eaddr='UK'`

position 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 composition function also implements the (non-RA) operations of requiring at least n values to be shared in the composition.

The operations are applied to a context stored in an input file (e.g. ex1.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 as shown in Table 1. Table 2 shows examples of command-line usage of FcaFlint.

FcaFlint function	Meaning
dual(ex1.cxt)	Dual context
invers(ex1.cxt)	Invers context
union(ex1.cxt, ex2.cxt)	Union
inters(ex1.cxt, ex2.cxt)	Intersection
compos(ex1.cxt, ex2.cxt)	Relational composition
appos(ex1.cxt, ex2.cxt)	Apposition: ex1 on the left of ex2
subpos(ex1.cxt, ex2.cxt)	Subposition: ex2 underneath of ex1
equal(ex1.cxt, ex2.cxt)	Prints "Matrices are equal" or "Matrices are not equal"
trans(ex1.cxt)	Transitive closure of composition
<ONE>	one
<NUL>	nul
<DIA>	dia
<AID>	$\overline{dia}$

Table 1. RA operations in FcaFlint

FcaFlint command-line	Meaning or result
fcaflint "inters(ex1.cxt,invers((ex1.cxt))" output.cxt	$I \cap \overline{I}$
fcaflint "inters(ex1.cxt,compos(ex1.cxt,ex2.cxt))" output.cxt	$I \cap (I \circ J)$
fcaflint "equal(ex1.cxt,(dual(dual(ex1.cxt))))" output.cxt	$I = (I^d)^d$, prints: "Matrices are equal"
fcaflint "invers(<ONE>)" output.cxt	prints: "Result is NUL matrix"
fcaflint file.bat output.cxt	Reads the operations from a batch file "file.bat".

Table 2. Examples of FcaFlint command-lines

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 aimed at expert users because using RA requires some expertise.

The second version of FcaFlint intends to support RA operations according to the named perspective described in Section 3.1. In particular, it is fairly easy to implement checks for objects and attributes ensuring that they are compatible. The implementation

of “inside” and “outside” conditions is slightly more complicated. The approach that is currently envisioned 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.

Websites

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

References

1. Ganter, Bernhard, & Wille, Rudolf (1999). *Formal Concept Analysis. Mathematical Foundations.* Berlin-Heidelberg-New York: Springer.
2. 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.
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 (2009). *Relation Algebra Operations on Formal Contexts.* In: Proceedings of the 17th International Conference on Conceptual Structures, ICCS’09, Springer Verlag.