Formal Concept Analysis with ConImp: Introduction to the Basic Features

Peter Burmeister[*]

April 3, 2003

Contents

[*]Department of Mathematics (Arbeitsgruppe Allgemeine Algebra und Diskrete Mathematik), Darmstadt University of Technology, Schloßgartenstr. 7, D-64289 Darmstadt, Germany, email: burmeister@mathematik.tu-darmstadt.de. — This is in part a translation, but mainly an extension and update of the article "ConImp — *Ein Programm zur Formalen Begriffsanalyse*" from this author (cf. [B00]). In particular among other changes, hints to the keystrokes in ConImp necessary in order to start the corresponding subprogam have been added. Special thanks are due to our secretary, Mrs Cornelia Franzke, for her assistance with the translation.

0 Preliminary remarks

For more than two decades Formal Concept Analysis (FCA for short) now provides conceptual tools for the analysis of data, and FCA has already had many successful applications. One of the main objectives of the method of FCA is to visualize the data in form of concept lattices and thereby to make them more transparent and more easily discussible and criticizable (see the example below of a repertory grid for an anorectic patient in Tables 1, 2 and 3 and the corresponding Figures 1, and 4). Another important tool provided in connection with Formal Concept Analysis is the method of "interactive attribute exploration", which allows knowledge acquisition from (or by) an expert by putting very precise questions to him, which either have to be confirmed or to be refuted by a counterexample. This method will be described in detail in section 6. Among the programs for the support of Formal Concept Analysis ConImp is one of those most widely spread all over the world and the most frequently used so far, and starting from Version 4.16 from August 2001 it is also available for Linux-platforms. ConImp allows input and "manipulation" of (so-called one-valued) contexts and of implications between the attributes and calculations of related data from these inputs, which are mainly needed in applications. In particular all the data needed for the drawing of the related concept lattices by hand (and also for the so-called geometric method) can be provided. As kernel of ConImp one may consider the subprogram on the so-called **interactive attribute exploration**, which — like many other parts — is constantly developed further — in particular w.r.t. the part concerning incomplete data or incomplete knowledge (cf. section 6).

In the following we try to explain the basic concepts of formal concept analysis, which is a theory that has been developed since about 1980 by R. Wille (cf. [W82]) and members of the Research Group on Formal Concept Analysis at the Technische Hochschule Darmstadt; and we shall do this in connection with some examples. And at the same time we want to present the main features of ConImp. In particular all kinds of input and output, all manipulations of contexts and implications as well as all calculations mentioned in this text can be carried out by using ConImp. And we try to give in parenthesis the key strokes needed to reach the option under consideration (the path is usually indicated by starting from the main menu, or, when preceded by . . . , starting from the submenu under discussion).

All those parts which directly refer to an example are printed in slanted style; and those parts more or less presenting output of the program on screen or printer in the format used by ConImp are given in typewriter style.

Before we start with the basic concepts let us add some remarks concerning the development of ConImp: The first programming on and first applications of the program "ConImp" (the name abbreviates "$\underline{\text{Con}}$texts and $\underline{\text{Imp}}$lications") have started in 1986 — still under the operating system CPM on an APPLE II computer[1] and

[1]The names of computer systems and programs mentioned here and in what follows are mainly protected by law and copyright and used here only for the purpose of identifying and referencing these products.

still under the name "BAmn"[2] —, mainly in order to have the so-called attribute exploration (cf. section 6) and the necessary calculations in formal concept analysis at hand on this type of computer. Later on there have also been compiled versions, which ran under MSDOS or PCDOS (on so-called IBM-compatible computers) or which ran under TOS (on ATARI ST computers). As programming languages TURBO PASCAL 3.0 and later 6.0 by BORLAND[3] were used and — for ATARI ST computers — ST PASCAL (by Creative Computer Design (CCD)) and lately Pure PASCAL (by Application Systems Heidelberg). By and by the program was enlarged by new features, mainly caused by requests of users and by own interests. At the moment (September 2001) version 4.16 exists with English texts, and for DOS- and Windows3x, -95, -98, -ME and -NT-computers as well as for the operating system Linux (compiled with fc-Pascal (from First Publisher)).[4] This text refers to this current version 4.16.[5] The programming has not been done by professional programmers, and the surface is still quite old-fashioned, however — except for the usual minor errors, which occur in every larger program, and for crashes that occur in some versions when some of the lists of computed data become too large and the computer then runs out of memory — the program has always worked quite reliably and is the program which has so far been used most frequently in formal concept analysis.

1 (Formal) Contexts

In [W82] R. Wille has introduced Formal Concept Analysis — in connection with his attempt to restructure order and lattice theory — as an application of order and lattice theory in connection with so-called Galois connections induced by relations. The theory is based on a set theoretical model for conceptual hierarchies. This model mathematizes the philosophical understanding of a **concept** as a unit of thoughts consisting of two parts: the **extension** and the **intension** (comprehension). The **extension** covers all objects (or entities) belonging to the concept, while the **intension** comprises all attributes (or properties) valid for all the objects under consideration.[6] In connection with this philosophical background a **formal context** of the form $\mathbb{K} := (G, M, I)$ has become one of the basic notions of formal concept analysis. Here G and M are sets; the elements of G are called **objects**, and the elements of M are

[2]Begriffs-Analyse handling contexts with about 10*m attributes and 10*n objects. — The almost eqivalent role, which by and by contexts and implications began to play, when the program was extended and modified later on, has then led to the change of the name into ConImp.

[3]Starting from version 4.9 up to version 4.15 there could also be provided on request a version compiled in the so-called "protected mode" with Borland Pascal 7.0, in which the generated lists can become much longer than in the usual version.

[4]Unfortunately, those versions compiled with Borland Pascal 7.0 do not run under the most recent versions of Windows. Whoever has such a version should contact the author for a more recent and appropriately compiled version.

[5]Starting from version 3.3 in 1991 only the English version has been extended, while up to then there were also German ones. However the German mnemonics for the keystrokes are still used.

[6]All those who want to learn more about formal concept analysis and in particular about its mathematical background, should read the book [GW96] by B. Ganter and R. Wille.

called **attributes**. Moreover, I ($\subseteq G \times M$) is a binary relation between the sets of objects and attributes, respectively. In formal contexts, which usually refer to some application, the relation $(g, m) \in I$ (often also written as gIm) is read as follows: The object g is in relation I to the attribute m, if "the object g **has** the attribute m", or, equivalently, if "the attribute m **applies to** the object g".

A formal context[7] can be considered as (the mathematical model of) a table, which relates objects and attributes of a "real situation". The entries in the table indicate by a cross (in the program ConImp by the letter "**x**") that the object, the name of which precedes the corresponding row, has the attribute, the name of which is at the top of the corresponding column (of the entry). And by an empty space (blanc: " ") or (in the program as default) by a period (full stop: ".") it is expressed that the corresponding object does not have that attribute — or that it is not known, whether or not this is the case. This refers to a so-called **one-valued context**.[8] The program ConImp uses in this case the usual two-valued attribute logic, i.e. points or blanks as entries are always interpreted in the way that the corresponding attribute does not apply to the corresponding object.

Grid O11E2V1	SE	ID	FA	MO	SI	BR	EL	JA	MA	JE	DI
rational – emotional	2	3	2	5	4	3	2	5	3	4	4
sincere – insincere	1	2	2	3	3	3	2	3	3	4	2
optimistic – pessimistic	5	1	3	5	3	3	4	4	3	2	2
interested – not interested	5	2	2	3	3	3	2	4	4	3	3
flexible – timid	5	1	3	3	2	3	4	4	3	2	2
materialistic – idealistic	5	5	4	2	4	2	3	1	4	3	4
not fash-consc – fash-consc	2	2	2	5	2	4	4	5	3	4	4
light hearted – depressive	6	1	3	4	3	3	3	3	4	2	2
resolute – insecure	5	2	3	4	4	3	4	4	5	2	3
unconstrained – constrained	4	2	2	3	2	3	4	5	4	3	2

Table 1: A repertory grid of a patient (example O11E2V1)

As a first example we consider a context in Table 2 which is derived from a repertory grid given in Table 1 and taken from [SpWo], where it is discussed in great detail by the involved psychotherapist. This repertory grid has been produced by an anorectic patient in a session with her psychotherapist on relations among her family and persons related to her:
SELF (SE – the patient herself), IDEAL (ID – her ideal, which she wants to be), FATHER (FA), MOTHER (MO), SISTER (SI), BROTHER (BR), ELVIS (EL), JANE (JA), MARY (MA), JENNIFER (JE) and DIANA (DI).

[7]In what follows we shall usually omit the addition "formal", since all our contexts will be formal ones.

[8]That one speaks of "one-valued contexts" refers also to the fact that in the set theoretical language used in formal concept analysis one cannot refer explicitly to the negations of attributes, when these negations do not occur as attributes by themselves (cf. the examples in the following sections).

Context	SE	ID	FA	MO	SI	BR	EL	JA	MA	JE	DI
rational	×	.	×	.	.	.	×	.	.	.	.
emotional	.	.	.	×	.	.	.	×	.	.	.
sincere	×	×	×	.	.	.	×	.	.	.	×
insincere	.	.	.	.	.	.	.	.	.	.	.
optimistic	.	×	.	.	.	.	.	.	.	×	×
pessimistic	×	.	.	×	.	.	.	.	.	.	.
interested	.	×	×	.	.	.	×	.	.	.	.
not interested	×	.	.	.	.	.	.	.	.	.	.
flexible	.	×	.	.	×	.	.	.	.	×	×
timid	×	.	.	.	.	.	.	.	.	.	.
materialistic	.	.	.	×	.	×	.	×	.	.	.
idealistic	×	×	.	.	.	.	.	.	.	.	.
not fashion conscious	×	×	×	.	×	.	.	.	.	.	.
fashion conscious	.	.	.	×	.	.	.	×	.	.	.
light hearted	.	×	.	.	.	.	.	.	.	×	×
depressive	×	.	.	.	.	.	.	.	.	.	.
resolute	.	×	.	.	.	.	.	.	.	×	.
insecure	×	.	.	.	.	.	.	.	×	.	.
unconstrained	.	×	×	.	×	.	.	.	.	.	×
constrained	.	.	.	.	.	.	.	×	.	.	.

Table 2: The one-valued scaled context for the above grid for example O11E2V1: "1" or "2": a cross for the first, "5" or "6": a cross for the second property

For samples of two persons chosen randomly among those the patient had mentioned to the therapist she had to give a pair of "opposite" adjectives somehow distinguishing these two persons and then to rate all the above persons on a scale from 1 to 6 with respect to all these pairs. The result was the grid shown in Table 1, which contains these ten pairs, which are now heading the rows of the table where "not fash-consc – fash-consc" is short for "not fashion conscious – fashion conscious".

*In order to analyse the information hidden in the grid, the therapist "***scaled***" the grid into a one-valued context by splitting each pair into the two attributes consisting of the first adjective and the second adjective of the pair, respectively, and by making a cross in the column for a person and the row of the first attribute, if the corresponding pair had obtained the values "1" or "2" for this person (otherwise a period for this first attibute), and by making a cross in the column of a person and the row for the second attribute, when the person was rated "5" or "6" for the corresponding pair, otherwise a period for the second attribute of the pair. For the values "3" and "4" each row got a period, since these values were rated as a kind of undecidedness of the patient — there are other kinds of translation ("scaling") of the grid into a one-valued context, but this one has been very informative to the expert and has prevented the* *analysis from becoming too complicated.*[9] *In this way Table 2 was obtained.*

Such a table can now be entered into ConImp (**B** . . .). However, in general in a formal context of Formal Concept Analysis the columns correspond to the attributes and the rows to the objects, which everyone might expect to be the persons in connection with our example. ConImp allows in the alteration menu (**A** . . .) to "interchange the roles of objects and attributes" (**A V**), i.e. to transpose the table (that means to take the mirror image of the table with respect to its main diagonal which runs from upper left to lower right).

In addition, ConImp also offers the possibility to enter question marks "?" in order to indicate that the user does not know, whether or not the attribute applies to the object. And such contexts are then treated by a modification of the usual three-valued KLEENE-logic (mainly in connection with the interactive subprogram of attribute exploration (cf. section 6)). One can tell ConImp to use a three-valued logic either when editing a new context (command: **B** . . . **3** . . .)[10] or by changing to the three-valued logic in the change menu (**A L**). If one is loading a context from a disk or harddisk (**L** . . .), then the logic is automatically chosen to be three-valued if a question mark occurs in the context; otherwise, the two-valued logic is chosen, even when the the context has been saved as one using a three-valued logic (then **A L** has to be pressed again, if one wants to return to it). — The use of the three-valued logic may be of particular interest if one enters counterexamples in connection with the so-called **attribute exploration**[11] (**I I I** . . .), if one wants to enter, in connection with the edited counterexample, only the application of those attributes which are asolutely necessary for the answer to the actual question (possibly in addition to those that are obvious).[12] A context containing question marks (or with three-valued logic chosen) can at the moment be transformed (by command) in at least two ways into a "one-valued" one:[13]

U T converts all question marks into blanks,

U P converts all question marks into crosses.

Such a change has to be done before most computations can be carried out. Starting

[9]There are many ways to transform (scale) a so-called **many-valued context** (like the grid of the example) into a one-valued context. There exists a public domain DOS-program "MBA.exe", by which such scaling can be done automatically, once the many-valued context and the scales for the attributes have been entered into MBA.exe by the user.

[10]When we indicate the keystrokes necessary to start a subprogram, we usually start from the main menu of ConImp. If it is meant to start from a menu that is clear from the surrounding text, then the keystrokes are preceded by " . . . ". If the given keystrokes are just the start of some kind of dialogue (or of several different choices following the start of the subprogram or submenu), then the indicated keystrokes are followed by " . . . ". Observe that in connection with commands ConImp does not distinguish between small and capital characters. However, under Linux for loading a file file names have to be entered observing small and capital characters.

[11]Cf. section 6.

[12]More about this can be found e.g. in BURMEISTER [B91] or HOLZER [H01].

[13]If one is lucky, also one of the transformations in connection with implications already entered or computed may convert all question marks (see below).

from version 4.15 one has to distinguish because of the results of Richard Holzer in [H01] between **ordinary objects** and **fictitious objects**.[14] However, before performing such a transformation one can first "prepare" the present context with questionmarks as follows:
enter known implications as "background implications"[15] (**I H** ...) (if it has not been done yet), and then choose

U U to eliminate all question marks, which have to be crosses or blanks, when the computed or entered implications including possibly existing "uncertain" implications are true in the resulting context, but such that fictitious objects remain unchanged;

U A to eliminate all question marks, which have to be crosses or blanks, when the computed or entered implications including possibly existing "uncertain" implications are true in the resulting context, applying them also to fictitious objects;

U C to eliminate all question marks, which have to be crosses or blanks, when the computed or entered "certain" implications (i.e. excluding possibly existing "uncertain" implications) are true in the resulting context, but such that fictitious objects remain unchanged;

U F to eliminate all question marks, which have to be crosses or blanks, when the computed or entered "certain" implications (excluding possibly existing "uncertain" implications) are true in the resulting context, applying them also to fictitious objects.

These options also use the "**Duquenne-Guigues base**" (cf. section 5), if this is has been computed or loaded before.

The present version 4.18 of ConImpcan handle contexts with at most 255 attributes and at most 255 objects.[16]

The connection between a table (context) and a "real situation" is mainly established by the names for the attributes and objects; at the moment these names can consist of at most nine characters (six in earlier versions). In order to enter names with ConImp one calls the corresponding submenu either directly from the main menu (**N** ...) or from the context editor (**B** ... **Control N**).[17]

[14]Fictitious objects occur in connection with the procedure of attribute exploration in order to encode undecided implications in connection with the option **I I I** ... **O**. They can also be entered directly by the user. Their name has to start with a questionmark in order that they are treated correctly e.g. in connection with the following transformations.

[15]Cf. section 5.

[16]Earlier versions could only handle m objects and n attributes with m and n strictly less than 256 and their product less than about 25000 (e.g. 98 attributes and 255 objects or 158 ojects and attributes).

[17]We shall abbreviate "**Control N**" by "**^N**" — and analogously all other commands, where one has to press the key of the corresponding character while pressing (holding) the control key.

In this way the data given in Table 2 can now be entered into the context editor of ConImp *(***B** *. . .) — we have named the context "RepGrid" —, and it shows already the transposed context in comparison to Table 2. Yet, because of the restrictions on the name lenghts to nine characters one either has to allow* ConImp *to cut the names after nine characters, as we have done for the context "RepGrid" (see Table 3 — the attribute names have to be read from top to bottom) or to think of good shorthands — in particular, when the first nine characters do not distinguish all the names. — Let us observe that in Table 3, which shows an original print-out of a context by* ConImp*, we have just added in parentheses the consecutive numbers of the objects; on the screen these numbers are shown at the beginning of the line of the corresponding object, while its name is shown at the end of the line. In this table we also show the "subcontext" "RepGridM" of "RepGrid" having as set of attributes all the attributes of "RepGrid" and having as objects only the four main reference persons SELF, IDEAL, FATHER, MOTHER (we shall discuss it later in some more detail).*

In the following we mainly use examples of contexts, where the objects are certain positive natural numbers, and where the attributes are some properties, which we have chosen from those which might be of interest and make sense for natural numbers.[18]

To begin with, we consider as objects all natural numbers from 1 to 9, i.e. $G := \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ *, and as attributes we choose*

even	*i.e.*	*"is an even number"*
	resp.	*" is divisible by 2"*
odd	*i.e.*	*"is an odd number"*
	resp.	*"is not divisible by 2 without rest"*
prime	*i.e.*	*"is prime"*
	resp.	*"is only divisible by 1 and by itself"*
square	*i.e.*	*"is square"*
	resp.	"$= a * a = a^2$ *for a natural number* a"
cubic	*i.e.*	*"is cubic"*
	resp.	"$= a * a * a = a^3$ *for a natural number* a"

Thus, $M = \{$ *even, odd, prime, square, cubic* $\}$ *; and the relation* I *applies to the number* z *and the attribute* m *if the number* z *has the attribute* m*.*

In the case that an attribute m from the set M of attributes applies to an object

[18]Most applications of ConImp are concerned with the analysis of tables, which have nothing to do with mathematics — as can be seen from our introductory example of the repertory grid, which is quite typical. However, mathematical examples are more suitable for the discussion of the main part of ConImp — as we see it — which concerns the so-called (interactive) **attribute exploration**; and there are so far only very few genuine non-mathematical examples of attribute exploration — like finding keywords for the classification of books in a (relatively small) library —, and in non-mathematical conceptual universes experts have to come to a consent about the validity of the (attribute) implications (see section 5, while in mathematical conceptual universes the validity of implications can usually be decided more objectively. In particular, we hope that every reader can easily follow our examples concerning natural numbers and some simple properties concerning them.

```
Print-out of the
 context RepGrid

          e iopin  minfld  uc
         rm npenof adoaierino
         aosststtl tetsgpencn
         ttiiise e ea hhrssos
         iinnmirixtrlfiteoent
         ooccimeniiiiao slcsr
         nneesistbmassnhsuuta
         aarrtstelilth eitrri
         lleeiterediiicaveean
         --------------------
     SELF!x.x..x.x.x.xx..x.x..! ( 1)
    IDEAL!..x.x.x.x..xx.x.x.x.! ( 2)
   FATHER!x.x...x.....x.....x.! ( 3)
   MOTHER!.x...x....x..x......! ( 4)
   SISTER!........x...x.....x.! ( 5)
  BROTHER!..........x.........! ( 6)
    ELVIS!x.x...x.............! ( 7)
     JANE!.x........x..x.....x! ( 8)
     MARY!.................x..! ( 9)
 JENNIFER!....x...x.....x.x...! (10)
    DIANA!..x.x...x.....x...x.! (11)
        ----------------------
```

```
Print-out of the
 context RepGridM

          e iopin  minfld  uc
         rm npenof adoaierino
         aosststtl tetsgpencn
         ttiiise e ea hhrssos
         iinnmirixtrlfiteoent
         ooccimeniiiiao slcsr
         nneesistbmassnhsuuta
         aarrtstelilth eitrri
         lleeiterediiicaveean
         --------------------
     SELF!x.x..x.x.x.xx..x.x..!
    IDEAL!..x.x.x.x..xx.x.x.x.!
   FATHER!x.x...x.....x.....x.!
   MOTHER!.x...x....x..x......!
        ----------------------
```

Table 3: The contexts "RepGrid" and "RepGridM" concerning example O11E2V1 (see Table 2) and for the first four persons in it

g from the set G of objects of the context under consideration, one enters a cross into the table in the row which is preceded by the object name "z",[19] and in the column at the top of which one has the attribute name "m".

In our example for a context, which we will call "NumE1"[20] *this will look as in Table 4. — In particular we get the following: Since 2 is an even prime number, there are crosses in the row preceded by the object name "2" and in the columns*

[19]In the context editor the row *ends* with the object name and *starts* with the "consecutive number" of the object under consideration.

[20]"NumE1" stands here as an abbreviation for "example No. 1 on natural numbers". The context names of the other examples below will have to be interpreted in a similar way. — Note that the name of a context will be taken by ConImp as default name (which could be changed, but usually it is not wise to do so, as will become obvious immediately) to save the context *and data related to it* into a file of this name — with an extension hinting to the kind of the data —; therefore a context name cannot have more than eight characters (limit for file names under DOS). Thus, our context will be saved as "NumE1.CXT", when we choose the option "Save context" (**S** . . .).

preceded by the attribute names “even” and “prime”, respectively. Since 2 is neither odd nor square nor cubic (within the set of all natural numbers) the corresponding entries are periods. — We present the context here in the same way as it will be represented in a print-out of the program — but basically also as it will appear in the context editor —, i.e. the attribute names must be read from top to bottom in the columns; furthermore, we represent the **clarified context** *“NumE1p” derived from the context “NumE1” and the* **reduced context** *“NumE1r”, both of which will be explained in section 4.*

```
NumE1                 NumE1p                  NumE1r
             s                      s                       s
            pqc                    pqc                     pqc
          e ruu                  e ruu                   e ruu
          voiab                  voiab                   voiab
          edmri                  edmri                   edmri
          ndeec                  ndeec                   ndeec
          -----                  -----                   -----
        1!.x.xx!               1!.x.xx!                1!.x.xx!
        2!x.x..!               2!x.x..!                2!x.x..!
        3!.xx..!               3!.xx..!                3!.xx..!
        4!x..x.!               4!x..x.!                4!x..x.!
        5!.xx..!               6!x....!                8!x...x!
        6!x....!               8!x...x!                9!.x.x.!
        7!.xx..!               9!.x.x.!                 -------
        8!x...x!                -------
        9!.x.x.!
         -------
```

Table 4: Original formal context NumE1, and the derived clarified context — NumE1p — and reduced context — NumE1r

In the context editor — contrary to the print-out — one has on the left of the respective row the consecutive number of the corresponding object and on the right the object name. And the object and attribute name belonging to the line and column in which the curser is situated are shown again in the first line of the screen — together with the corresponding consecutive numbers attached to them (which are identical with the row and column number, respectively).

2 Concepts and Concept Ordering

Before we can answer e.g. the question of what is meant by a “reduced context” in Table 4, we will first treat the formal concepts and the hierarchical relation of comparability (order relation $\leq$) between formal concepts. This "subconcept – superconcept relation" belongs to the essential basic concepts of formal concept analysis:

Let $\mathbb{K} := (G, M, I)$ be a context. Let A be a subset of the object set G — i.e. A consists only of objects and all these objects belong to G —, then A' denotes the set of all *attributes* from the attribute set M which apply to *each* object in A. Conversely, for a subset B of the attribute set M, B' denotes the set of all *objects* from G to which *each* attribute from B applies.

Again, let A be a subset of the object set G and let B be a subset of the attribute set M, then the (ordered) pair (A, B) formed with these two sets is called a **(formal) concept** (for the context $\mathbb{K}$) if one has $A' = B$ and $B' = A$, i.e. if B consists of precisely those attributes from M which apply to all objects from A, and if conversely A consists of precisely those objects from G which have all attributes from B. If (A, B) is a formal concept, then A is called the **extent** and B is called the **intent** of (A, B). Moreover, one then always has $A'' = A$ and $B'' = B$. Frequently, formal concepts of the form $(\{g\}'', \{g\}')$ or $(\{m\}', \{m\}'')$ are of special interest, which are "generated" by a single object g from G or by a single attribute m from M, respectively. In such a case of one-element sets, we usually omit the set brackets (braces) and write (g'', g') and (m', m''), respectively. The first case is called an **object concept**, the second case an **attribute concept**. In general, one can say that the formal concepts correspond to maximal rectangles of crosses in the formal context — after appropriate permutations of the rows and columns. ConImp is able to compute the list of all concepts of the present context (**V B**) or only their number (**Z** ...) — this option should be used first, if one does not know whether there is still enough memory left (the memory still available for lists is shown at the bottom of the main menu[21]). By choosing **D B** ... **B** <**R**eturn> and by confirmation with **Y** (for "Yes") one can see row by row the extent on the left and intent on the right hand side of the screen — as long as the number of objects plus the number of attributes does not exceed 69, otherwise several lines of the screen are used and the assignment of the object or attribute names to the crosses in the corresponding row becomes almost unrecoverable — one then might use the option **D F** ... **B** to write the list of concepts into a text file and view it later with an editor (as plain ASCII text).

Thus, one can easily see that for the context NumE1 in Table 4 $(\{3, 5, 7\}, \{\mathit{odd}, \mathit{prime}\})$ *is a formal concept, having precisely the numbers "3", "5" and "7" in its extent and exactly the attributes "prime" and "odd" in its intent. One can easily realize that it is an object concept which is generated by any one of the objects in its extent (that it equals e.g.* $(3'', 3')$*). A further formal concept of this context is* $(\{2, 3, 5, 7\}, \{\mathit{prime}\})$*, which obviously equals the attribute concept* $(\mathit{prime}', \mathit{prime}'',)$*. Furthermore,* $(\{2, 4, 6, 8\}, \{\mathit{even}\}) = (\mathit{even}', \mathit{even}'')$ *is also an attribute concept. At a close look, one will discover that in our example every concept*

[21]Starting from version 4.16 of ConImp the available memory may be greater than shown, since in the versions compiled by fc-pascal only that memory is shown, that is granted to the program just at the moment, and this can dynamically be enlarged within the available memory, if necessary.

distinct from ($\emptyset''$, $\emptyset'$) and ($\emptyset'$, $\emptyset''$)[22] is an object or attribute concept, but this is not always the case, as one can see in the later example NumE2de of a context (cf. Table 8), the "concept lattice" (cf. section 4) of which is presented in Table 5. Another example, where it is not the case, is the "concept lattice" (see Figure 1 below) of the scaled repertory grid in Table 3, e.g. with the concept

({ SELF, FATHER }, { rational, sincere, not fashion conscious }).

— In this example the psychotherapist can realize e.g. from the two object concepts (see the corresponding lines of the context)
({ SELF }, { timid, depressive, not interested, pessimistic, insecure, idealistic, rational, sincere, not fashion conscious }) and
({ IDEAL }, { sincere, optimistic, interested, flexible, idealistic, not fashion conscious, light hearted, resolute, unconstrained })
in Table 2 or 3 that his patient identifies herself with most of the negatively rated attributes and wishes to have most of the positively rated attributes, having only three from nine attributes in common with her ideal (this can be seen more clearly from the corresponding line diagram of the "concept lattice" (and part of it) in Figure 1 (and Figure 4) below) — see Spangenberg, Wolff [SpWo] for a more extensive discussion of this context and its "concept lattice".

As already mentioned in connection with contexts, we will generally omit in the following the addition "formal" in front of the word "concept", if it is clear that we are dealing with formal concepts.

In the following we denote by $\mathfrak{B}(G, M, I)$ or $\mathfrak{B}(\mathbb{K})$ the set of all concepts of the context $\mathbb{K} := (G, M, I)$. We now define a binary relation $\leq$ — to be read as "less or equal" or "is a subconcept of" — on the set $\mathfrak{B}(G, M, I)$ of all concepts of $\mathbb{K} := (G, M, I)$ as follows: If (A_1, B_1) and (A_2, B_2) are concepts of $\mathbb{K}$, then $(A_1, B_1) \leq (A_2, B_2)$ (i.e. the concept (A_1, B_1) is less or equal to — respectively a **subconcept** of — the concept (A_2, B_2)) if (and only if), A_1 is a subset of A_2 (or equivalently, if and only if B_2 is a subset of B_1) [23] — (A_2, B_2) is then also called **superconcept** of the concept (A_1, B_1) . This definition corresponds to the philosophical convention that a concept always has a smaller extension and a larger intension than any of its superconcepts.

Thus, for example, among the concepts belonging to the context NumE1 one has that

$$(\{3, 5, 7\}, \{\textit{odd, prime}\}) \leq (\{2, 3, 5, 7\}, \{\textit{prime}\}) ,$$

while $(\{2, 4, 6, 8\}, \{\textit{even}\})$ *is not comparable with either of the two concepts above w.r.t. the relation* $\leq$.

The subconcept–superconcept relation "$\leq$" on the set $\mathfrak{B}(\mathbb{K}) = \mathfrak{B}(G, M, I)$ of all concepts of a context $\mathbb{K} := (G, M, I)$ is always

[22] $\emptyset$ denotes the empty set, i.e. the set which does not contain any element, and which consequently is contained as a subset in every set.

[23] Note the reversed order in connection with the intents.

reflexive	i.e.	$\mathfrak{b} \leq \mathfrak{b}$ for all concepts $\mathfrak{b}$ in $\mathfrak{B}(\mathbb{K})$,
transitive	i.e.	for all concepts $\mathfrak{b}_1$, $\mathfrak{b}_2$ and $\mathfrak{b}_3$ in $\mathfrak{B}(\mathbb{K})$:
		$\mathfrak{b}_1 \leq \mathfrak{b}_2$ and $\mathfrak{b}_2 \leq \mathfrak{b}_3$ always imply $\mathfrak{b}_1 \leq \mathfrak{b}_3$,
antisymmetric	i.e.	for all concepts $\mathfrak{b}_1$ and $\mathfrak{b}_2$ in $\mathfrak{B}(\mathbb{K})$:
		$\mathfrak{b}_1 \leq \mathfrak{b}_2$ and $\mathfrak{b}_2 \leq \mathfrak{b}_1$ always imply $\mathfrak{b}_1 = \mathfrak{b}_2$.

A set P, on which a binary relation $\leq$ is defined which is reflexive, transitive and antisymmetric, is called an **ordered set**, and the relation $\leq$ is called an **order relation** on P. And in particular, the pair $(P, \leq)$ is then called an **ordered set**.

As we shall see in section 3 below, ordered sets can be represented by a line diagram, and it will be explained, how to read them. Moreover, we shall learn that the ordered sets of all concepts of a context are indeed "lattices", and therefore one speaks of the "**concept lattice**" of a given context.

The concept lattice of the context "RepGrid" is shown in Figure 1[24]

3 Digression: Some basic notions of the theory of ordered sets

Before entering more deeply into the different features of ConImp we first want to explain some basic notions of order theory to those readers who are not quite familiar with this theory, and we want to provide them with some of the relevant facts, too.

Since the program ConImp only enables the user to work on finite contexts corresponding to finite sets of concepts (starting from 255 attributes one can maximally have 2^{255} concepts, i.e. considerably less than 10^{77} — a 1 with 77 zeroes), we will (mainly) restrict ourselves to finite sets (and lattices (see below)).

In the following, let P always be a set and $\leq$ an order relation on P, i.e. $(P, \leq)$ is assumed to be an ordered set.

- If p and q are elements of P, we say that p and q are **comparable** if $p \leq q$ or $q \leq p$; otherwise, p and q are called **incomparable**.

- Given two elements p and q of P, we say that p is a **lower neighbour** of q and accordingly that q is an **upper neighbour** of p if (1) and (2) below hold, where:

  (1) $p \leq q$ and $p \neq q$,

  (2) for all r in P, $p \leq r \leq q$ always implies $r = p$ or $r = q$;

  i.e. in other terms: "p is strictly less than q, and there is no element in P, which lies between p and q and is different both from p and q". If p is a lower neighbour of q, we write "$p \prec q$".

[24] See section 3 below, how to read the line diagram.

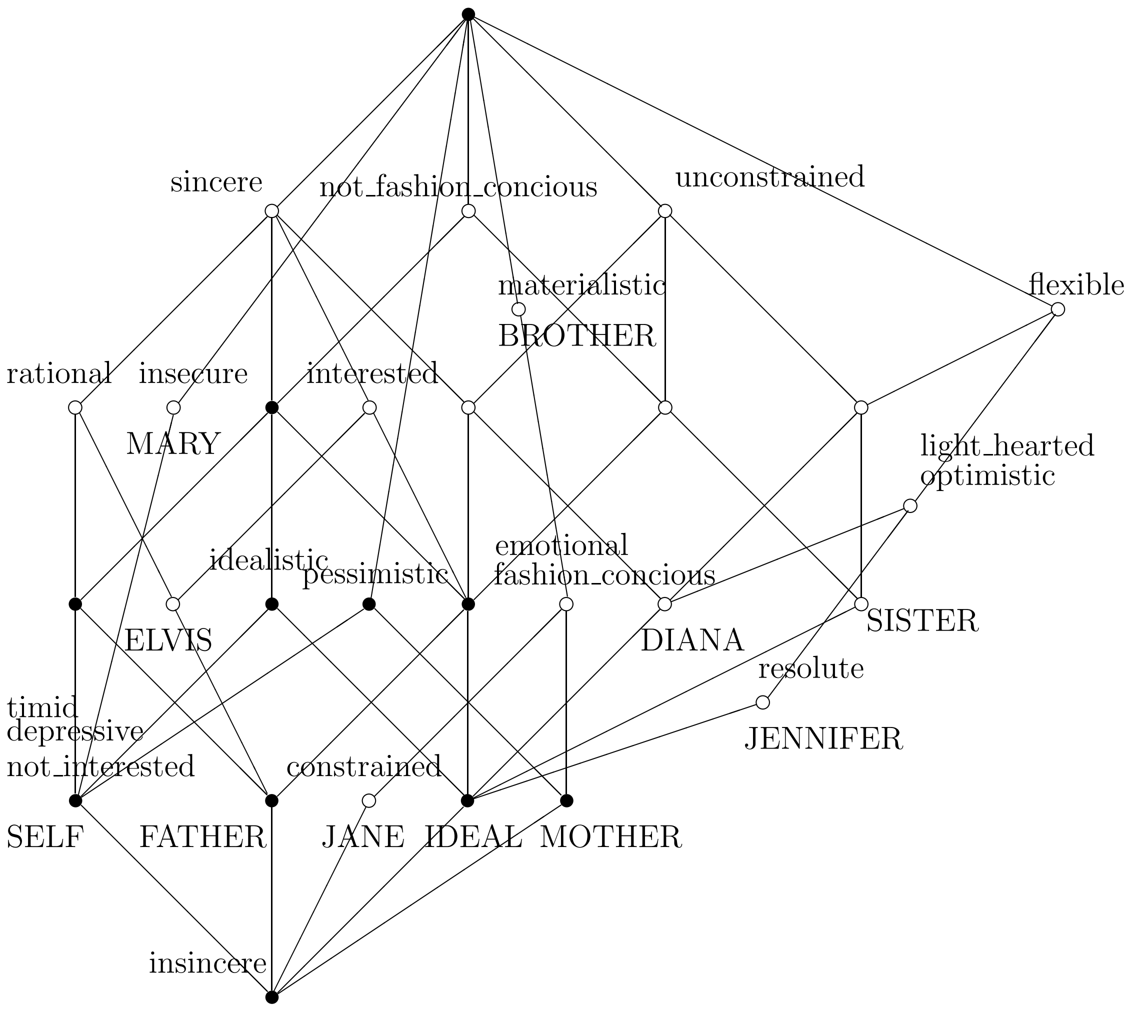

Figure 1: The concept lattice for the context "RepGrid" in Table 3

- If $(P, \leq)$ is a **finite** ordered set, then — due to transitivity and finiteness — the order relation $\leq$ is uniquely determined by the neighbourhood relation $\prec$ defined by $\leq$ as described above.

- In ConImp there are two ways to compute the order relation of a concept lattice corresponding to the actual context, and one has several options which really do it:[25]

  - **V V** computes for each concept a list of all lower neighbours,
  - **V N** computes for each concept a list of all upper neighbours,
  - **V L** computes for each concept a list of all upper neighbours as well as a list of all lower neighbours,
  - **V A** first computes the list of all concepts (together with aditional information as to which concepts the attributes and objects have to be assigned (cf. the attribute and object concepts)) and then computes for each concept a list of all upper neighbours and a list of all lower neighbours.

- Every finite ordered set $(P, \leq)$ can be represented graphically by a so-called **line diagram** (often also called **Hasse diagram**). The elements of P are depicted by small circles. The drawing is to be made in such a way that in each (and only such) case, when p is a lower neighbour of q in $(P, \leq)$, i.e. when $p \prec q$ holds, then the circle for q is drawn above the circle for p, and the circle for q is connected to the circle for p by a downward directed (mostly straight) line (on which there is no additional circle). Then one can read the relation $v \leq w$ between elements v and w of P in the respective line diagram from the existence of a sequence of connected line segments strictly moving upwards from the circle for v to the circle for w, or one has $v = w$.

- *As an example see the line diagram of the concept lattice for the context "RepGrid" in Figure 1. One can read from it that "SELF" has the properties "timid", "depressive", "not interested", "rational", "sincere", "insecure", "idealistic", "not fashion conscious" and "pessimistic".*

- *As a further example let us consider the set* $\{1, 2, 3, 5, 6, 10, 15, 30\}$ *of all divisors of the number* 30, *and let* $\leq$ *be the relation "is a divisor of" (in symbols:* $|$ *). Then, obviously,* 2 *and* 3 *are lower neighbours of* 6, *but they are not lower neighbours of* 30. *Nevertheless,* $2|30$ *is true, as one can also read from the line diagram for* $(\{1, 2, 3, 5, 6, 10, 15, 30\}, |)$ *represented in Figure 2.*

- Two ordered sets $(P, \leq)$ and $(Q, \sqsubseteq)$ are called **isomorphic**, if there is a one-to-one assignment, say f, assigning to each element of P an element of Q in such a way that $p \leq q$ in $(P, \leq)$, if and only if $f(p) \sqsubseteq f(q)$ in $(Q, \sqsubseteq)$, and such that to

[25] Observe that except for the option **V A** the concept lattice has to be computed in advance (by **V B**).

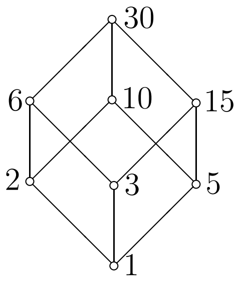

Figure 2: Example of a line diagram: Divisors of the number 30

each element of Q some element of P has been assigned (this last requirement means that f has to be "onto"). This means, too, that every element of Q is assigned by f to exactly one element of P.

- Let N be a subset of P. An element q of P is called an **upper bound** of N, if $n \leq q$ is true for every element n of N (by analogy p of P is called a **lower bound** of N, if $p \leq n$ for every n in N).

  If there is a **least upper bound** of N — that means: if there is an upper bound q_0 of N, for which $q_0 \leq q$ for all upper bounds q of N —, then q_0 is called the **supremum** of the set N and it is uniquely determined by this property; the supremum of N, if it exists, is denoted by sup N. By analogy, the **infimum** p_0 of N is defined as the **greatest lower bound** of N, if this exists, and is denoted by inf N.

  For example, in Figure 1 supremum and infimum exist for every subset of concepts. In particular the attribute concept for "idealistic" is the supremum of the object concepts for "SELF" and "IDEAL", while the object concept for "IDEAL" is the infimum of the attribute concepts for "idealistic" and "flexible".

  And in Figure 2 supremum and infimum exist for every subset of the set of divisors of 30, *i.e. in the ordered set* $(\{1, 2, 3, 5, 6, 10, 15, 30\}, |)$. *For example, one has that* $10 = \sup\{2, 5\}$ *und* $2 = \inf\{6, 10\}$.

- An ordered set $(P, \leq)$ is called a **lattice**, if supremum and infimum exist for every two-element subset of P. In this case, supremum and infimum also exist for every finite, non-empty subset of P.

  Let P be finite and let $\emptyset$ denote the empty subset of P; then sup $\emptyset$ and inf $\emptyset$ also exist.[26]

[26]sup $\emptyset$, if it exists in $(P, \leq)$, always is the least element of P, since every element of P is an upper bound of the empty subset — and, analogously, inf $\emptyset$, if it exists in $(P, \leq)$, is the greatest element of P, since every element of P is also a lower bound of $\emptyset$: sup $\emptyset$ = inf P and inf $\emptyset$ = sup P.

Lattices are of interest for us in so far as for every context $\mathbb{K}$ the ordered set $(\mathfrak{B}(\mathbb{K}), \leq)$ of all its concepts is always a lattice.

The line diagram in Figure 2 also represents a lattice.

Moreover, every finite lattice $(V, \leq)$ can be represented as the lattice of all concepts of some context — for example of the context $(V, V, \leq)$ — (i.e. there is a canonically defined isomorphism between these lattices). And if one wants to check whether a "small" finite ordered set $(P, \leq)$ is a lattice, one just has to enter the context $(P, P, \leq)$ into ConImp (**B** . . .) and to compute for it the number of concepts (**Z** . . .). If (and only if) this is equal to the number of elements in P, then $(P, \leq)$ is a lattice.

In particular, the lattice in Figure 2 is the concept lattice of the context

$$\textit{Div30} := (\{1, 2, 3, 5, 6, 10, 15, 30\}, \{1, 2, 3, 5, 6, 10, 15, 30\}, \mid),$$

which is represented in Table 5. In this connection every symbol for a number must be considered both as an object name and as an attribute name.

```
Div30        311           Div30ar     11         Div30cr     11
             05065321                  506                    506
             --------                  ---                    ---
          30!x.......!              30!...!                  5!xx.!
          15!xx......!              15!x..!                  3!x.x!
          10!x.x.....!              10!.x.!                  2!.xx!
           6!x..x....!               6!..x!                   -----
           5!xxx.x...!               5!xx.!
           3!xx.x.x..!               3!x.x!
           2!x.xx..x.!               2!.xx!
           1!xxxxxxxx!               1!xxx!
            ----------                -----
```

Table 5: Formal context Div30 of all divisors of the number 30 with "attribute-reduced" and "completely reduced" contexts (Div30ar) and (Div30cr), resp.

We continue with some additional notions and facts which are of importance for the understanding of certain operations in concept analysis — connected with the features of ConImp. In the following let $(V, \leq)$ always be a finite lattice (that is, in particular let V be a finite set).

- An element v of V is called **join-irreducible** or sup-**irreducible**, if v cannot be represented as the supremum of a subset of V which does not contain v as an element. A sup-irreducible element can also be characterized by the fact that it has exactly one lower neighbour. $J(V)$ designates the set of all join-irreducible elements of $(V, \leq)$.

  By analogy, an element w of V is called **meet-irreducible** or inf-**irreducible**, if it cannot be represented as the infimum of a subset of V which does not contain w, i.e. if it has exactly one upper neighbour. $M(V)$ designates the set of all meet-irreducible elements of the lattice V.

In the lattice shown in Figure 1 exactly the object concepts are join-irreducible, and except for the attribute concept generated by "timid", "depressive" or "not interested" all attribute concepts are meet-irreducible.

In Figure 2 one can see that in the lattice of all divisors of the number 30 *precisely the concepts corresponding to the numbers* 2, 3 *and* 5 *are join-irreducible and precisely the concepts corresponding to the numbers* 6, 10 *and* 15 *are meet-irreducible.*

- A (finite) lattice $(V, \sqsubseteq)$ is always uniquely determined — up to isomorphism — by the sets $J(V)$ of its join-irreducible and $M(V)$ of its meet-irreducible elements and by the restriction of the order relation from $(V, \sqsubseteq)$ to the set $J(V) \cup M(V)$ comprising all elements of V which are either join- or meet-irreducible. — In particular $(V, \sqsubseteq)$ is "in a canonical way" isomorphic to the lattice $(\mathfrak{B}(J(V), M(V), \sqsubseteq), \leq)$ of all concepts of the context $(J(V), M(V), \sqsubseteq)$.

  The **Basic Theorem of Formal Concept Analysis** (see [W82], p. 449 or [GW96], p. 20 in the English edition) says among other things that the ordered set of all concepts of a context is always a ("complete") lattice and that all the join-irreducible elements in it have to be object concepts and all the meet-irreducible elements have to be attribute concepts. This corresponds to what we have already observed in our examples.

4 Concept Lattice, Reduction and Purification of a Context, Subcontexts

As already mentioned, the ordered set $(\mathfrak{B}(G, M, I), \leq)$ of all concepts of a context $\mathbb{K} = (G, M, I)$ is always a lattice, which is also called the **concept lattice** of the context $\mathbb{K}$. By convention, in its line diagram the object names are always written below the circle which represents the object concept generated by the respective object — the object concept (g'', g') being the smallest concept having the object g in its extent. Correspondingly, the attribute names are written above the circle which represents the attribute concept generated by the respective attribute — the attribute concept (m', m'') being the largest concept having the attribute m in its intent. One can read from the line diagram that the object g has the attribute m, since the circle labelled with the name of the object g — i.e. representing the (object) concept (g'', g') — is connected with the circle labelled with the name of the attribute m — i.e. representing the (attribute) concept (m', m'') — by a sequence of line segments moving upwards, if and only if the object g has the attribute m. Thus, the context can be reconstructed from its line diagram.

The reader should compare Figure 1 with Table 3.

In ConImp the **concept list** is calculated (**V B** or **V A**) by means of the algorithm "**next concept**" developed by B. GANTER (cf. e.g. [G87] or [GW96]). After the computation of this list, the so-called **assignment lists** are produced, which assign to each object, say g, the number of the corresponding object concept (g'', g'), and to each attribute, say m, the number of the corresponding attribute concept (m', m'') — these lists enable the user to assign the correct labels to the concepts in the line diagram, when he/she is drawing the line diagram by hand. Furthermore, for every concept the set of all upper and lower neighbours is calculated in the "predecessor list" (meaning the list of all lower neighbours) and "successor list"[27] (meaning the list of all upper neighbours), if option **V A** has been chosen, otherwise these lists have to be computed separately. The concept list can be saved together with the predecessor and successor list (**T** ...) in order to reload it in a later session (**K** ...) or to use it in other programs. — If one fears that the concept list might become too long and that it might not fit into the memory (in this case the program would "crash"), one can calculate only the number of concepts without keeping the list in memory (**Z** ...). This computation can be interrupted, whenever the program shows how many concepts it has found so far (one has to fix the size of these intervals before the start), and after an interruption the results can be saved in order to resume the computation at any other time — one is asked for it on leaving the subprogram.

The list of concepts by itself is not of great importance for the drawing of the line diagram by hand, while the other lists just mentioned (i.e. of upper and lower neighbours) are very useful for this purpose: While looking for instance, which lower neighbours a given concept has, the list of upper neighbours is used to find out a good position for the corresponding circle in order to produce not too long lines. In addition, the list of concepts looses even more of its importance once one has drawn a line diagram of the concept lattice (cf. section 4), since then extent and intent of each concept can be read from this line diagram (see below).

For the context "NumE1" in Table 4, the assignment lists and the predecessor and successor lists are listed in Table 6 in the format used by ConImp *("–" marks the end of a list for one concept).*

Now one can draw the respective concept lattice by hand. [28]

In this connection it is useful to know that the first concept (concept no. 1) is always the greatest one and that the last concept is always the smallest one.

A line diagram of the context NumE1 is represented in Figure 3. The small numbers are the consecutive numbers of the concepts representing the order in which they are produced by ConImp.

From this line diagram, one can now also realize that for example the intent of the concept no. 8 consists of the attributes "odd" and "prime", since from the

[27]The lists are referred to like that in ConImp.

[28]Cf. also the literature on line diagrams, in particular [W84] or [S89]. In particular there is a method of "geometric preparation" by means of the predecessor and successor lists. For a description of this method one can consult e.g. [GW96], p. 70ff. For this method at least one of the lists of predecessors or successors is needed.

```
List of the predecessors of         List of the successors of
the concepts in the context NumE1   the concepts in the context NumE1
(first: number of the respective    (first: number of the respective
 concept                             concept
 then: concept numbers of            then: concept numbers of
         the predecessors)                   the successors)
  1:    2    3    4    5    9  -    1:    -
  2:    7   10    -                 2:    1    -
  3:    6   11    -                 3:    1    -
  4:    8   12    -                 4:    1    -
  5:    6    8    -                 5:    1    -
  6:    7    -                      6:    3    5    -
  7:   13    -                      7:    2    6    -
  8:   13    -                      8:    4    5    -
  9:   10   11   12    -            9:    1    -
 10:   13    -                     10:    2    9    -
 11:   13    -                     11:    3    9    -
 12:   13    -                     12:    4    9    -
 13:    -                          13:    7    8   10   11   12  -
```

```
Assignment list for the context  Assignment list for the context
NumE1: objects                   NumE1: attributes
(first: concept number)            (first: concept number)
  6 :     9                         2 :  cubic
  7 :     1                         3 : square
  8 :     3                         4 :  prime
  8 :     5                         5 :    odd
  8 :     7                         9 :   even
  9 :     6
 10 :     8
 11 :     4
 12 :     2
```

Table 6: Predecessor and successor lists for the context NumE1, and assignment lists for the objects and attributes

circle corresponding to this concept there exist ascending paths exactly to the circles corresponding to these attribute concepts (and to no other ones). And one can see, too, that this concept has the extent $\{3, 5, 7\}$*, since from the circle corresponding to this concept there exist descending paths exactly to the circles corresponding to these object concepts, respectively that it corresponds by itself to these object concepts, and that the concept no. 6 has precisely the objects "1" and "9" in its extent and precisely the attributes "square" and "odd" in its intent.*

Namely, the intent of a concept $\mathfrak{b}$ consists of all the attributes, the names of which are attached to a circle which can be reached by an ascending line path from the circle which is assigned to the concept $\mathfrak{b}$ (possibly including the circle of $\mathfrak{b}$), while the extent of $\mathfrak{b}$ consists of all objects the object concept of which is less than or equal to the concept $\mathfrak{b}$ (i.e. can be reached by a descending line path from the circle for $\mathfrak{b}$).

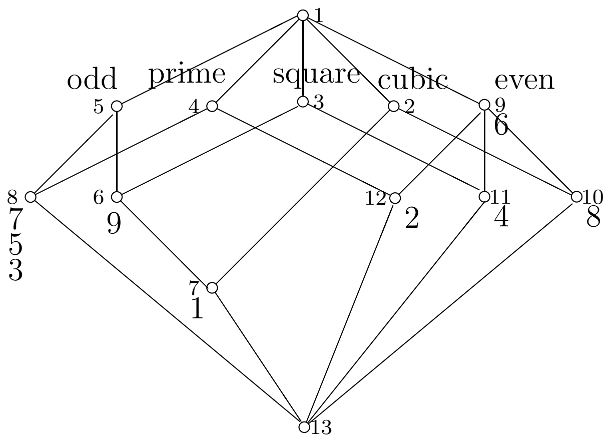

Figure 3: A line diagram for the concept lattice of the context NumE1

In the line diagram in Figure 3 one can see — what one might have deduced already from the context — even more clearly: The objects 3, 5 and 7 generate the same object concept.

The replacement of all those lines (columns) of a context which are identical except for the object name (attribute name) by one single line (column) is called **purification** of the context (we shall discuss the different options of ConImp for this process below in connection with the discussion of the "reduction" of a context). What happens in this process can be found out for the attributes without drawing a line diagram, if one computes the attribute order with ConImp (**Q**). As an output one first obtains a list of lists of attributes, the induced attribute concept of which has the same extent in each case (in the subsequent lists of predecessors or successors, from each list only the first attribute is used further). In order to determine this for the objects as well, one temporarily has to "exchange the role of objects and attributes" (**A V**; in mathematical terms this means to "transpose" the incidence table for I); and the attribute order which is computed in this connection is actually the object order of the original context. The orders computed this way can often be used sucessfully when drawing the lattices in order to distinguish certain "chains of attribute or object concepts" by choosing the same direction for the line segments connecting them. — In general it has turned out to be useful in connection with drawing a line diagram to use as little directions as possible for the necessary line segments.

Because of what we have learnt in the last section about the role of the join- and meet-irreducible elements of a finite lattice, we know for the concept lattice of the context $\mathbb{K}$ that the set $J(\mathfrak{B}(\mathbb{K}))$ of join-irreducible concepts is always contained in the set of all object concepts of $\mathbb{K}$ and that the set $M(\mathfrak{B}(\mathbb{K}))$ of all meet-irreducible concepts is always contained in the set of all attribute concepts. And as already mentioned in the last section, too, the context $(J(\mathfrak{B}(\mathbb{K})), M(\mathfrak{B}(\mathbb{K})), \leq \cap (J(\mathfrak{B}(\mathbb{K})) \times M(\mathfrak{B}(\mathbb{K}))))$ is sufficient to uniquely determine the structure of the concept lattice $(\mathfrak{B}(\mathbb{K}), \leq)$ except for isomorphy (some information about the original context being lost in the process (concerning the reducible object and attribute concepts)). We say, that **an object g is reducible**, if the respective object concept is join-reducible (i.e. not join-irreducible); correspondingly, **an attribute is called reducible**, if the respective attribute concept is meet-reducible.

The limitation to the irreducible objects and/or attributes is called **reduction** of the context (which is usually done in accordance with certain criteria, as seen below). The process of reduction is frequently useful or even necessary, if otherwise the context is getting too big to be handled by ConImp. ConImp allows the purification and reduction of the sets of objects and attributes in different combinations, the reduction of a set always includes its purification as well:

- **R R** for reducing the set of objects as well as the set of attributes,
- **R G** for reducing the set of objects,
- **R M** for reducing the set of attributes,
- **R H** for purifying the set of objects,
- **R N** for purifying the set of attributes,
- **R S** for purifying the set of objects and the set of attributes simultaneously,
- **R Q** for purifying the set of objects and reducing the set of attributes,
- **R P** for purifying the set of attributes and reducing the set of objects.

If one wants to find out what has happened in the course of a reduction of attributes, one can consider — besides the attribute order — the attribute implications treated in the next section (by means of the "transposed context" one obtains the corresponding "object implications").

From the line diagram in Figure 3 one can now tell that in the concept lattice for the context NumE1 the only join-reducible object concept is $(6'', 6')$ *with the number 9 (it has three lower neighbours) — however, purification is part of the reduction process, and therefore two of the three objects "3", "5" and "7" vanish as labels under reduction of objects. There is no meet-reducible attribute (concept). The clarified and reduced context for NumE1 is presented in Table 4. Table 5 presents both, the attribute-reduced and the completely reduced context for the context Div30. In this case there is nothing that might be clarified.*

Some readers may have observed that in the concept lattice in Figure 1 some circles are fully black. These correspond to all the concepts which are generated via joins from the object concepts for the objects "SELF", "IDEAL", "FATHER" and "MOTHER". Namely, it has turned out for the psychotherapist that these persons *and their relationship to the attributes used by the patient give him already most of the information he can draw out of the whole concept lattice (the attributes then have to be assigned to the "nearest black concept" below their attribute concept in the whole lattice). The other persons usually only round off the picture he has obtained from this smaller context. Since this part of the information is not so easily seen within the whole lattice — in particular if the latter is even more complicated than the one shown in Figure 1 —, we show the concept lattice corresponding to the* **subcontext** *"RepGridM" of the context "RepGrid" — both are shown in Table 3 — in Figure 4.*

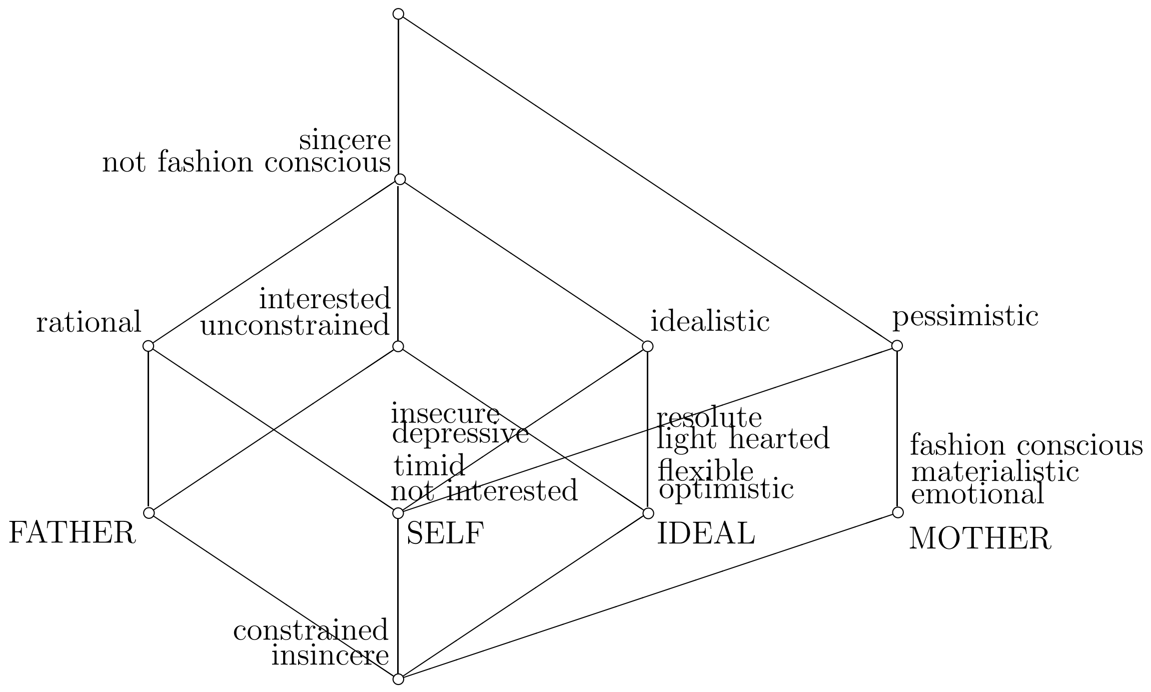

Figure 4: The concept lattice for the first four persons in Table 3

In order to be able to work on parts of a given context, ConImp has several commands in the alteration menu **A** . . . .[29]

- With **A S** . . . one can select a subcontext with only part of the objects of the given one or one can change the order of the objects by successively entering the actual (consecutive) number of the object which one wants to keep — or to place in this special position. A "0" (zero) ends the subprogram. One then has still the possibility to give the obtained context a new name. This subcontext is then immediately taken as "main context" to which all the other options of ConImp can be applied. As long as one has not yet reduced or clarified the

[29]Starting from version 4.16 one can also add and delete objects and attributes one by one at wished places directly within the context editor (**B** . . .). This will be discussed in more detail in section 6.

new context one can return to the old context with **A A**. This last option is in particular quite useful, when one wants to produce several subcontexts from the given one.

- With **A R** one can do this with respect to attributes (choose some attributes for a subcontext or just change their sequence).

5 (Attribute) Implications

An **attribute implication** "$P \Longrightarrow C$" for a context $\mathbb{K} = (G, M, I)$ consists of two subsets P and C of the attribute set M of the context (G, M, I), the set P being called the **premise** and C being called the **conclusion** of this implication. The implication $P \Longrightarrow C$ is **valid** for (**holds** in) the context (G, M, I) if the following is true:

> For every g from G: If every attribute from the premise P applies to the object g, then every attribute from the conclusion C also applies to g.

This is equivalent to C being a subset of P'', the closure of the premise P with respect to the context $\mathbb{K}$. Furthermore, the validity of the implication $P \Longrightarrow C$ for the context $\mathbb{K}$ can easily be read from the line diagram of the concept lattice of $\mathbb{K}$, since it is equivalent to the fact that the infimum of the set of the attribute concepts being elements of C can be reached through an ascending line path from the infimum of the attribute concepts being elements of P (i.e. that $\inf P \leq \inf C$ is true).

So one can read for example from Figure 3 that in the context NumE1 the implication

$$(*) \qquad \{\textit{cubic, odd}\} \Longrightarrow \{\textit{square}\}$$

is true; or that the attributes "odd" and "even" together imply all other attributes (since there is no object in the context which has both attributes at the same time) and thus that the infimum of the two corresponding attribute concepts is the smallest concept of the concept lattice.

In connection with the implications computed for a context with ConImp one should be aware that one is dealing with relations between the attributes *in the present context* and not necessarily with relations between the attributes "in general".

Thus, for example in our context NumE1 of numbers the implication

"square, cubic $\Longrightarrow$ odd "

holds, as one can see from the Table 7 below. This is, however, not an implication which holds for all positive integers, since e.g. 64 is a square and a cubic number, but not an odd one (cf. also section 6 on attribute exploration).

In the case of empirically obtained contexts it is very important for the users of ConImp to be aware of the fact that the implications computed by ConImp heavily depend on the actual context, and that there is no guarantee that they really hold in the whole (conceptual) universe of interest for the user. The display of the number of “supporting examples” (see below) can often only provide very weak evidence for the fact that an implication might be valid beyond the object range of the context. Neither does the “attribute exploration” (see below) provide “absolute certainty”, since in the case of non-mathematical problems, it will in general only be based on a certain degree of consensus among experts, who may have cooperated in the matter. However the method requires a totally reliable opinion (decision).

Since the set of all implications valid in a context is usually very big, it is attempted to produce lists of implications from which all implications valid in a certain context can be *generated*, i.e. derived in some prescribed way. For example, this generation can be effected by means of rules which we do not want to describe in this paper and for which we refer for example to [M83] or [B91].

It is, however, also possible to understand the matter without knowing about the rules: A set T of attributes from M is said to **respect** an implication $P \Longrightarrow C$, if either P is not a subset of T (i.e. at least one attribute contained in P is not contained in T), or if C is a subset of T. We say that an implication $P \Longrightarrow C$ **follows** from a set $\mathcal{I}$ of implications, if every set T of attributes respecting every implication from $\mathcal{I}$ also respects the implication $P \Longrightarrow C$. A set $\mathcal{I}$ is called

- a **generating subset** of the set $\mathcal{I}(\mathbb{K})$ of all implications valid (holding) in the context $\mathbb{K}$, if $\mathcal{I}$ is a subset of $\mathcal{I}(\mathbb{K})$ and every implication from $\mathcal{I}(\mathbb{K})$ follows from $\mathcal{I}$;
- a **base** of $\mathcal{I}(\mathbb{K})$, if it is a generating subset of this set of implications and looses this property whenever any implication from $\mathcal{I}$ is omitted;
- a **minimal base** of $\mathcal{I}(\mathbb{K})$, if $\mathcal{I}$ is a base and no attribute can be omitted from any implication from $\mathcal{I}$ — neither from a premise, nor from a conclusion — without loosing the property to generate $\mathcal{I}(\mathbb{K})$.

Now in Formal Concept Analysis there is a canonical base for the implications valid in a given context $\mathbb{K}$, the so-called **Duquenne-Guigues base** (cf. [DGu86] or [G87], who describes an algorithm for its computation, which is intensively used in ConImp).[30] This base has the property that for each implication occurring in it the set of all attributes being part of this implication constitutes an intent of a concept. With ConImp it is possible to compute for an existing context $\mathbb{K}$ both the **Duquenne-Guigues-base** (**I I D**) and (from that) a **minimal base** (**I B**) for the

[30]This computation is based on the so-called pseudo-intents: A subset P of the attribute set M of a context (G, M, I) is called a **pseudo-intent** of this context, if P itself is not a intent, i.e. if $P \neq P''$ but if for every pseudo-intent Q strictly contained in P it is true that Q'' is already contained in P. In [DGu86] Duquenne and Guiges have shown that the set

$$\{\, P \Longrightarrow C \mid P \text{ is a pseudo-intent of } \mathbb{K} \,\}$$

is a base for the implications of the context $\mathbb{K}$.

implications valid in the context $\mathbb{K}$.[31]

Table 7 represents the Duquenne-Guigues-base and a minimal base for the implications of the context NumE1 from Table 4.

In this example we assume that the concept list has been computed with ConImp before one has computed the implications. In this case of a Duquenne-Guigues-base the number in round brackets (parentheses) in front of an implication "$P \Longrightarrow Q$" shows the consecutive number of the concept the intent I of which is given by the set of attributes contained in this implication, i.e. for which $I = P \cup Q$ (in the case of other lists of implications — for example in the case of a minimal base for the set of all implications valid in the context NumE1, which is also shown in this table — it shows the consecutive number of that concept the intent I of which is "generated" by the attributes of the premise P (i.e. $P'' = I$)). The number in "angular" brackets (like $< n >$) placed in front of an implication denotes the number of objects in the context for which the premise of this implication is part of the intent of the corresponding object concept; "$< 0 >$" in this case means that there is no such object, i.e. the corresponding concept is the smallest concept of the concept lattice. At the beginning of every row there is the consecutive number of the respective implication. [32] In the output of the program the set brackets (braces) around premise and conclusion are being omitted.

The effect of shortening in particular the premises when using an irredundant minimal base instead of the Duquenne-Guigues-base however does not show in the example NumE1. Yet one may compare the minimal base of implications shown in Table 8 for the example NumE2de with the list of the Duquenne-Guigues implications to be found as the set of accepted implications in the minutes of the attribute exploration of the context NumE2d in section 6 — which leads to the context NumE2de.

Moreover, it is possible to compute with ConImp two further generating subsets for the set $\mathcal{I}(\mathbb{K})$ of all implications valid in $\mathbb{K}$, which are substantially larger than the Duquenne-Guiges-base, but which can nevertheless be quite useful in connection with various problems. These are the **list of proper implications** (**E**) and the **list of implications with an independent premise** (**M**).

Before dealing with these lists, we would like to mention that the set of intents of a context consists of precisely those subsets of the set M of attributes which respect all implications of a generating subset of the set $\mathcal{I}(\mathbb{K})$. For a set $\mathcal{I}$ of attribute implications valid in $\mathbb{K}$ and for a subset T of M, let $T^{\mathcal{I}}$ denote the smallest subset of M comprising T and respecting all implications from $\mathcal{I}$. Then $T'' = T^{\mathcal{I}}$ is true for each subset T of M, if (and only if) $\mathcal{I}$ is a generating subset of $\mathcal{I}(\mathbb{K})$ (i.e. then the closure T'' generated with the help of the context and the closure $T^{\mathcal{I}}$ of T generated with the help of $\mathcal{I}$ are always equal). Therefore, the concept lattice of a context

[31]If **I B** has been chosen and a Duquenne-Guigues-base has not been computed before, then it will also be computed in connection with this subprogram.

[32]For many — but not all — contexts this will mean that the premise P contains attributes contradictory to one another.

```
Duquenne-Guigues-base of the implications valid in the context NumE1:
  1. (   7) <  1> :     square     cubic  ===>        odd
  2. (   7) <  1> :        odd     cubic  ===>     square
  3. (  13) <  0> :      prime    square  ===>       even        odd      cubic
  4. (  13) <  0> :      prime     cubic  ===>       even        odd     square
  5. (  13) <  0> :       even       odd  ===>      prime     square      cubic


A minimal base of the implications valid in the context NumE1:
  1. (   7) <  1> :     square     cubic  ===>        odd
  2. (   7) <  1> :        odd     cubic  ===>     square
  3. (  13) <  0> :      prime    square  ===>        odd
  4. (  13) <  0> :      prime     cubic  ===>       even        odd
  5. (  13) <  0> :       even       odd  ===>      prime      cubic
```

Table 7: Two lists of implications for the context NumE1

can always be computed from a generating subset of the set of all implications valid in a context — up to isomorphism, and up to knowing the assignments from the attributes and objects, respectively, on one side, to the meet- and join-irreducible elements, respectively, on the other side.

We call a subset U of M **independent** (with respect to $\mathbb{K}$), if $V'' \neq U''$ holds for every proper subset V of U (i.e. for every subset V of U, which contains at least one element less than U).

The two lists of implications mentioned above can thus be described as follows:

- The list of **implications with an independent premise** for a context $\mathbb{K} = (G, M, I)$ consists of all implications $P \Longrightarrow C$ which are valid in $\mathbb{K}$ and for which P is an independent subset of M satisfying $P \neq P''$. Furthermore, one always has here $C = P'' \setminus P$, i.e. the conclusion C consists of all attributes in P'' which are not already contained in P. If the concept numbers have been printed, too (i.e. if before computing this list of implications, the concept list has been computed), one can read from this list all the independent subsets which are generating the intent of a particular concept. For the concepts which do not appear in the list, the corresponding intent is independent by itself. However, this list of independent intents must be extracted from the list of implications with independent premise and assigned concept numbers by hand (cf. the example NumE2d below).

- In ConImp the list $\mathcal{E}(\mathbb{K})$ of **proper implications** of the context $\mathbb{K} = (G, M, I)$ is obtained at present from the list of implications with an independent premise (therefore, in the case of longer lists, problems with the memory may occur).[33]

[33]Recently a direct method for the computation of all proper implications of a context has been found. This method uses the so-called "arrow relation" (cf. [GW96], p. 83). Neither this method nor the computation of the arrow relation have so far been implemented in ConImp.

The proper implications constitute a generating system for the set of all implications valid in the context, and this list is minimal with regard to the following property:

For any subset T of the attribute set M the intent T'' generated by T can be obtained adding to T all those conclusions of implications from this list for which the premise is a subset of T, i.e.

$$T \cup \bigcup \{\, C \mid P \Longrightarrow C \text{ is contained in } \mathcal{E}(\mathbb{K}) \text{ and } P \subseteq T \,\} = T'' = T^{\mathcal{I}(\mathbb{K})}\,.$$

We will give examples of these lists in connection with another context. For this purpose, we will consider (a reduced version of) the context NumE2de, which in the next section will be seen to be "representative" for the positive integers with the attribute set

$$M_{2d} := \{\, \textit{even, odd, prime, square, cubic,} \\ \textit{not-prime, no-square, not-cubic}\,\}.$$

The corresponding context together with a minimal base of implications is shown in Table 8. In Table 9 there is a list of all "implications with an independent premise" valid in NumE2de and in Table 11 there is shown a list of all proper implications of the context NumE2de. It should be taken into account that these lists of implications are (minimal) bases or just "generating subsets", respectively, for the set of all implications between the attributes from M_{2d} with regard to all positive integers, as will be seen in the next section. The list of non-empty intents, which are independent subsets of the attribute set M_{2d}, must be composed "by hand" using print-outs of the concept list and of the list of implications with an independent premise to obtain the concept numbers. These intents are shown in Table 10 together with the numbers of the corresponding concepts. In the concept lattice, these subsets generate so-called Boolean sub-lattices (those with three elements for example a cubic structure), which often facilitate the structuring of the line diagram.

One can deduce from the line diagram of the concept lattice NumE2de in Figure 5 that the intents

$$I_1 := \{\ \textit{no-square, not-prime, not-cubic, odd}\ \}$$

and

$$I_2 := \{\ \textit{no-square, not-prime, not-cubic, even}\ \}$$

(and all their subsets) constitute independent intents. From Table 9 — and in particular from those concept numbers of the context NumE2de not occurring in this list (the number of the concept generated by the (premise of the) implication is shown in parentheses) — one can infer that the independent intents for the context NumE2de are precisely the intents of the concepts no. 1 to 8 (and these are all the subsets of the intersection $I_1 \cap I_2$ of the two intents I_1 and I_2, and this intersection is the intent belonging to the concept no. 8), no. 15 to 22 (which are further subsets of I_1, which is the intent of the concept no. 22) and no. 29 to 36 (which are further subsets of I_2, which is the intent to the concept no. 36), since these concepts are missing in this table.

This means, however, that we should be interested in the maximal sets among them, which in our case belong to the concepts no. 22 and no. 36 and are just the intents I_1 and I_2.

```
NumE2de       nnn   Minimal base of the implications of the context
              ooo   NumE2de
              t-t
            s -s-    1. <4>:      cubic  ===>  not-prime
           pqcpqc    2. <4>:     square  ===>  not-prime
         e ruuruu    3. <4>:      prime  ===>  no-square not-cubic
         voiabiab    4. <0>:      cubic not-cubic  ===>       even
         edmrimri                                              odd
         ndeeceec    5. <0>:     square no-square  ===>       even
         --------                                              odd
       1!.x.xxx..!   6. <0>:      prime not-prime  ===>       even
       2!x.x...xx!                                             odd
       3!.xx...xx!   7. <0>:       even       odd  ===>      prime
       4!x..x.x.x!                                          square
       6!x....xxx!                                           cubic
       8!x...xxx.!
       9!.x.x.x.x!
      15!.x...xxx!
      27!.x..xxx.!
      64!x..xxx..!
        ----------
```

Table 8: Formal Context NumE2de (reduced)
and a minimal base of implications

When saving any of the lists of implications which can be computed or entered, neither the numbers possibly assigned to the concepts nor the numbers of "realizations" (i.e. the numbers in angular brackets (like $< n >$)) will be saved. The "number of realizations" will be computed anew for each output on the screen or on the printer (unless this option was deactivated in the implication menu: **I R** . . . , where " . . . " stands for the toggles corresponding to the different lists of implications: **D**, **M**, **E**, **H** and **B**). These numbers can be useful for example in connection with contexts with empirical (for example medical) data, in order to determine how many objects (for example patients) actually support the respective implication.

In the implication editor (**I H** . . .) (see below) one also has the **search mode** (. . . **S**): after marking all the attributes of the premise of the respective implication with a "+" (or even non-wanted attributes with a "−") one may press . . . **L** in order to get the list — however, only on the screen — of all those objects in the context having all those positively marked attributes in their intent (and not having the negatively marked ones), if there are any such objects in the context.

The lists of implications will not be saved (**I S** . . .) in the way they are usually printed, but with a list of the attribute names, a list of the truth values "T" or "U" (if the corresponding implication has been accepted as true or uncertain, respectively,

```
 1. ( 9) < 4> :      cubic    ===>   not-prime
 2. (10) < 2> :      cubic no-square    ===>   not-prime
 3. (11) < 4> :     square    ===>   not-prime
 4. (12) < 2> :     square not-cubic    ===>   not-prime
 5. (13) < 2> :     square     cubic    ===>   not-prime
 6. (14) < 4> :      prime    ===>   no-square not-cubic
 7. (23) < 2> :        odd     cubic    ===>   not-prime
 8. (24) < 1> :        odd     cubic no-square    ===>   not-prime
 9. (25) < 2> :        odd    square    ===>   not-prime
10. (26) < 1> :        odd    square not-cubic    ===>   not-prime
11. (27) < 1> :        odd    square     cubic    ===>   not-prime
12. (28) < 3> :        odd     prime    ===>   no-square not-cubic
13. (37) < 2> :       even     cubic    ===>   not-prime
14. (38) < 1> :       even     cubic no-square    ===>   not-prime
15. (39) < 2> :       even    square    ===>   not-prime
16. (40) < 1> :       even    square not-cubic    ===>   not-prime
17. (41) < 1> :       even    square     cubic    ===>   not-prime
18. (42) < 1> :       even     prime    ===>   no-square not-cubic
19. (43) < 0> :      prime    square    ===>        M_2d
20. (43) < 0> :      cubic not-cubic    ===>        M_2d
21. (43) < 0> :     square no-square    ===>        M_2d
22. (43) < 0> :      prime not-prime    ===>        M_2d
23. (43) < 0> :      prime     cubic    ===>        M_2d
24. (43) < 0> :       even       odd    ===>        M_2d
```

Table 9: List of implications with an independent premise for the context NumE2de (the premise and the conclusion together always constitute an intent. If all attributes occur, "M_2d" is briefly given as conclusion)

in connection with "attribute exploration" — observe that implications entered by the implication editor or computed directly from the context are always marked "T") and with an additional table, where an attribute of a premise is marked by "P" and an attribute of a conclusion is marked by "C", respectively, and the n-th line corresponds to the n-th implication and the m-th column corresponds to the m-th attribute —observe that some lists of implications can be printed into a file in a similar form (called "block form" in the printing menu). — When printed into a file or by a printer in this way, the implications are marked according to whether they were regarded as certain (T) or uncertain, which is only important, however, in connection with the results of an "attribute exploration" (see section 6). In this connection it should also be noted that, apart from the concept, predecessor and successor lists, all data which can be saved by the program ConImp will be saved as "almost uncoded" texts, which means that one can always look at them with a text editor, even if this is rather awkward in the case of the lists of implications. Concept, predecessor and successor lists are coded by means of a binary code "in rows" and can also be viewed at in the text editor. Direct decoding is possible but rather laborious.

```
 (2) { not-cubic }
 (3) { no-square }
 (4) { no-square, not-cubic }
 (5) { not-prime }
 (6) { not-prime, not-cubic }
 (7) { not-prime, no-square }
 (8) { not-prime, no-square, not-cubic }
(15) {       odd }
(16) {       odd, not-cubic }
(17) {       odd, no-square }
(18) {       odd, no-square, not-cubic }
(19) {       odd, not-prime }
(20) {       odd, not-prime, not-cubic }
(21) {       odd, not-prime, no-square }
(22) {       odd, not-prime, no-square, not-cubic }
(29) {      even }
(30) {      even, not-cubic }
(31) {      even, no-square }
(32) {      even, no-square, not-cubic }
(33) {      even, not-prime }
(34) {      even, not-prime, not-cubic }
(35) {      even, not-prime, no-square }
(36) {      even, not-prime, no-square, not-cubic }
```

Table 10: List of the non-empty intents for the context NumE2de, which are not occurring in Table 9, and which hence are independent

6 Attribute exploration and (background-) implication editor

If one now looks at the implications listed in Table 7, one discovers — as one may have found out already in the previous section — that the first and the second implication are true for the context NumE1, but not for the set of all positive integers, since 64 is square and cubic but not odd, whereas 27 is odd and cubic but not square. This means that with regard to the set of attributes which we have chosen the numbers from 1 to 9 are "not typical of all positive integers".

The problem of finding (sets of) "typical objects and attributes" is an important one e.g. in knowledge processing but also in many other areas. In Formal Concept Analysis this question also arises frequently, for example, if one intends to determine the concept lattice of a very large or even infinite context — which may mostly only be known in principle, but not in full detail —, having either a comparatively small set of attributes or of objects. The concept lattice of a context with n attributes or objects

```
 1. ( 9) < 4> :      cubic    ===>   not-prime
 2. (11) < 4> :     square    ===>   not-prime
 3. (14) < 4> :      prime    ===>   no-square not-cubic
 4. (43) < 0> :      prime    square    ===>        even        odd     cubic
 5. (43) < 0> :      cubic not-cubic    ===>        even        odd
                                                   prime     square no-square
 6. (43) < 0> :     square no-square    ===>        even        odd
                                                   prime      cubic not-cubic
 7. (43) < 0> :      prime not-prime    ===>        even        odd
                                                  square      cubic
 8. (43) < 0> :      prime     cubic    ===>        even        odd    square
 9. (43) < 0> :       even       odd    ===>       prime     square
                                                   cubic  not-prime
                                               no-square  not-cubic
```

Table 11: List of all proper implications for the context NumE2de

can maximally comprise 2^n concepts (usually much less). In Formal Concept Analysis such a large (still unknown) context is frequently called a **conceptual universe**.

If $\mathbb{U} := (G_{\mathbb{U}}, M, I_{\mathbb{U}})$ is a conceptual universe with a fixed set M of attributes and a generally very large set $G_{\mathbb{U}}$ of objects, a subset G of $G_{\mathbb{U}}$ is called **typical** of the conceptual universe (or a **representative set of objects** for $\mathbb{U}$), if the concept lattices $(\mathfrak{B}(G_{\mathbb{U}}, M, I_{\mathbb{U}}), \leq)$ and $(\mathfrak{B}(G, M, I_{\mathbb{U}} \cap (G \times M)), \leq)$[34] are isomorphic in such a way that related concepts have the same intents: $(A, B) \mapsto (\tilde{A}, B)$. In particular attribute concepts belonging to the same attribute will then be assigned to each other.

With the procedure of **attribute exploration**, sometimes also called "interactive implication program", Formal Concept Analysis offers a procedure for determining a typical set of objects (if the expert who is questioned has sufficient knowledge; compare the general remarks on implications at the beginning of the previous section). If one changes the roles of objects and attributes (**A V**) — one also says that one **transposes** the context or the table, respectively, and this can be carried out by means of the alteration menu (**A** . . . , described in the main menu by: "Change context data") — it is also possible to determine a typical set of attributes with ConImp , i.e. to carry out a so-called **object exploration**.[35] If neither the set of objects nor the set of attributes is fixed from the beginning, they can be extended by alternately using attribute and object exploration — however then the procedure

[34]This means that we are dealing with the concept lattice of the context being formed by the subset G of $G_{\mathbb{U}}$ as the object set, the fixed attribute set M and the restriction of the incidence relation $I_{\mathbb{U}}$ to this setting.

[35]The question which then has to be asked in the case of a suggested implication is no longer *"Do all objects in the conceptual universe to which all the attributes of the premise apply also have all the attributes of the conclusion?"*, but *"Do all attributes in the conceptual universe which apply to all objects of the premise also apply to all objects of the conclusion?"*. This type of question is for example relevant in connection with a subject catalogue for a library program, in which the books are regarded as objects and the (classifying) keywords as attributes (cf. [WaW92]).

might never stop and last infinitely long.

In this procedure of attribute exploration, the program ConImp suggests implications to the expert which he/she can either accept or disprove and in the latter case one has to enter a counterexample with all its respective attributes.[36] With the help of the implications possibly entered in advance by means of the implication editor — the so-called **background implications** — and with the implications already accepted, a "candidate P for a pseudo-intent" is first examined by ConImp as to whether the gap to the closure P'' of P w.r.t. the context can already be closed and whether the suggested implication $P \Longrightarrow P'' \setminus P$ can therefore be accepted automatically. Otherwise, it will be examined which parts of the premise and and which parts of the conclusion are "indispensable", and this part will be highlighted (inverted) on the screen in the next suggestion of an implication to the expert.[37] The expert then has to decide whether he can prove the implication or whether he can refute it by a counterexample. If both is impossible for the moment, the implication can also be **accepted as uncertain** (... **U**)[38], or one can automatically generate **fictitious objects** by pressing **O** for the so-called "optimal strategy" found by Richard Holzer (see [H01] or in a short form in [BH00]) and implemented starting from version 4.16.[39]

After having carried through an attribute exploration, during which only correct and complete answers were given to all questions of ConImp (and in the case of

[36]If one chooses a so-called three-valued logic as discussed at the beginning of this text, one has to disprove only part of the conclusion (at least one attribute) and one can leave the answers to the other questions open by entering question marks; such examples can possibly be treated further on the occasion of another suggested implication.

[37]However, this highlighting depends on the order in which the attributes are listed in the program; therefore it is in general not canonical.

[38]This option is no longer recommended. One should use the "optimal strategy" (... **O**) instead. However, for those used to the old versions it is still supported. ConImp usually works with an implication accepted as uncertain, as if it had been "accepted normally". If, however, a candidate for an implication could only be accepted automatically by using at least one uncertain implication (besides using accepted implications and background implications), then the expert is asked explicitly, whether he is able to prove or disprove this one.

[39]If you choose **O**, then you will be asked for each attribute in the conclusion, whether it follows from the premise. If you can answer **Y**, then it is added to a list, which — if it is non-empty at the end of this procedure — will become the conclusion of a new accepted implication. Otherwise, if it is one of the attributes which caused your uncertainty, answer **O** again, and the unknown (sub)implication will be automatically encoded as a fictitious object. The advantage of this strategy, which includes that during the procedure of attribute exploration the fictitious objects — whose names start with a question mark, which therefore should not be used at the beginning "normal objects" — are not updated w.r.t. newly accepted implications, is the following one: At the end of the attribute exploration with correct answers the fictitious objects encode all remaining open questions: If you can fully answer the implications encoded in the fictitious objects at some later time, then you have full information about your universe after having added the new counterexamples to your context and the new accepted implications to your old list. You will not have to run the interactive subprogram again. (Using the option **U** or producing your fictitious counterexamples "in some arbitrary way" will usually suppress some implications, which should be asked to get optimal information about your universe, and you will have to run the subprogram over and over again.) If you then want to produce the Duquenne-Guigues-base, you then only have to run the automatic procedure: **I I** <**Ret**> again.

three-valued logic: without any question marks remaining in the resulting context), one obtains on the one hand a Duquenne-Guigues-base for the implications valid in the conceptual universe, and on the other hand a context with a "typical set of objects". This means, in particular, that for every implication not valid in the conceptual universe one has an example (in the set of objects obtained by the attribute exploration) the intent of which does not respect this implication.

*We now add two further objects to the list of objects of the context NumE1, i.e. by means of the alteration menu (***A M** *. . .) in* ConImp *the number of objects is increased from nine to eleven, with the sub-program for naming these new objects (***N M***) the names "27" resp. "64" are assigned to them, and then the corresponding attributes are marked with a cross in the context editor (***B** *. . .). In this way, we obtain a context NumE2. We load (append) the list of Duquenne-Guiges-implications of NumE1 by means of the special loading option* **I L S** *into the implication editor.*

Observe that, starting from version 4.16 one can also add or delete objects and attributes one by one within the **context editor**:

Ins (or **Ctrl V**) allows to insert before or after the actual position of the cursor within the context a new object or attribute and give it a name immediately.

Del (or **Ctrl G**) allows to delete an object or attribute in the row or column of the cursor, respectively.[40]

While the use of the change menu allows to recover the original context after the deletion of an object (**A S**) or attribute (**A R**) by the option **A A**— as long as the new context has not been reduced or clarified (**R** . . .) —, the corresponding option in the context editor can only be reverted by inserting a new object or attribute for the deleted ones and entering the corresponding data.[41]

Observation: The option (**I L S**) enables the user of ConImp to load lists of implications possibly computed or entered for a different context into the implication editor or to add them to an existing list, as long as there are attribute names, which occur in the actual context as well as in the list which one intends to load. If a premise is fully represented in the current context and if additionally at least one attribute from the conclusion makes sense in the context, the "meaningful part" of the implication will be loaded. If necessary, the program even singles out implications which do not respect the intent of some object in the current context. Moreover, implications accepted "as uncertain" are also singled out at this stage, should they occur in the list to be loaded.

[40]Observe that **Ins** (the insertion key) or **Del** (the delete key) may not work on every computer in this connection, as we know by experience.

[41]If one wants to empty the list of objects, this can only be done via the option **A S** (i.e. via the change menu). Observe that the numeric keys may be used instead of the arrow keys in order to move in the context editor; however this may not work on every computer.

For instance in our example NumE2 the first two implications of NumE1 do not respect the intents of the numbers (objects) 64 resp. 27.[42]

If one now starts the attribute exploration for the context NumE2, using the three remaining implications in the background, all implications will be accepted automatically, i.e. $G := \{1, 2, 3, 4, 5, 6, 7, 8, 9, 27, 64\}$ *is a typical set of objects with regard to* M *in the conceptual universe of positive natural numbers.*

If one does not want to confirm each of the automatically accepted implications at each occurrence, one can deactivate this option as well as a number of other — sometimes troublesome — inquiries of the program in other parts of it (**A F**). In particular, this option provides the user of ConImp with a simple possibility to examine a given set of implications entered for a context $\mathbb{K}$ as to whether it is a generating set of implications for $\mathbb{K}$.

It is always checked in the implication editor (**I H** ...) whether an entered implication is valid in the context. If it is valid, one can leave the current line (e.g. by ... **V** or ... **Z**) or the editor (by ... <**ESC**>), otherwise one cannot do it before one has deleted (... **Y**)) the wrong implication.

Having at hand a typical set of objects, one can now search the context — and thus the conceptual universe — by means of the search mode (... **S**) in the editor of (background) implications (**I H** ...), for a list of all those objects which have or do not have particular attributes. However, if one also asks for attributes which should not apply to such an object, and if one wants to be sure to find at least one such example, whenever there is any at all in the conceptual universe, then one has to have applied attribute exploration to the **dichotomized** context (i.e. the set M of attributes has to contain with each attribute also its negation) — and, symbolically, to the dichotomized conceptual universe — before applying attribute exploration to it.

In order to dichotomize e.g. the given context NumE2 with ConImp one has to use a sub-program of the alteration menu (**A D**) which automatically adds as many attributes to the old context (e.g. NumE2) as the old one already contains and fills the new columns in such a way that the new attribute with (consecutive) number, say, $m + r$ applies to an object if and only if the corresponding original one (with number r, if M has m attributes) does not apply to it (**negation of the attribute**). Now the user of ConImp — by means of the naming menu (**N** ... , or ... ^**N** ... , if he/she is in the context editor) — assigns names to the new attributes which make clear that this attribute is the negation of the older attribute. If the original context $\mathbb{K}$ has m attributes, then the $(m + r)$-th attribute in the dichotomized context $\mathbb{K}_d$ is the negation of the r-th attribute of the context $\mathbb{K}$ (for $1 \leq r \leq m$), as already

[42]When transforming the context NumE1 into NumE2 it is also possible to retain the Duquenne-Guigues-base of NumE1, if the corresponding toggle (**I F**) is set to "TRUE" — what is the default — (or to load it beforehand into the implication editor (**I U** ...) and to retain this list, when questioned), load it into the implication editor and delete the two implications not valid in our new conceptual universe. You will detect them there quite easily when passing through the implications, since you cannot leave an implication or even the editor, when the actual implication does not hold in the actual context.

indicated above.

*In NumE2d we introduce "not even", "not odd", "not-prime", "no-square" and "not-cubic" as new attribute names. Since, however, the attributes "not even" and "odd" resp. "not odd" and "even" are equivalent, we delete the corresponding new attributes by means of the sub-program "change the sequence and number of attributes" (***A R***) in the alteration menu (***A** ...). The context is named NumE2d (***N K***) and has the following attribute set, which consists of 8 elements:*

$$M_{2d} = \{ \textit{even, odd, prime, square, cubic,} \\ \textit{not-prime, no-square, not-cubic} \} .$$

*Using the option "***I L S***" as described above, we now load the Duqenne-Guiges-base of NumE2 into the implication editor and add there all new attributes to the conclusions of these three implications (***I H A** ... **H***), since each of them has an inconsistent premise. Then we start the attribute exploration for NumE2d (***I I I** ...). If necessary, one can obtain an output of the minutes (referred to in the dialogue as "offprint") of the session in a file (***I I I Y** ...). The attributes which have not been presented in inverted mode on the screen by the program in connection with a proposal of an implication (i.e. which might be neglected at a first glance in connection with the decision about acceptance or refutation of the proposed implication — yet in the case of a refutation of the implication one has to consider all occurring attributes) will be put in parentheses below:*

1. *Suggested implication:*

   not-prime, no-square, not-cubic $\Longrightarrow$ even?

   *The answer is "***No***", because 15 is an* odd *number, which is neither prime nor square nor cubic.*

2. *Suggested implication:*

   cubic $\Longrightarrow$ not-prime?

   To be accepted.

3. *Suggested implication (comprises all attributes):*

   cubic, (not-prime), not-cubic $\Longrightarrow$
   $\Longrightarrow$ even, odd, (prime), (square), (no-square)?

   Here ConImp *points out that the implication comprises all attributes, in order to make the user or expert aware of the fact that the premise might be inconsistent.*
   Since here the "essential" attributes of the premise negate each other, this implication must also be accepted.

4. *Suggested implication:*

   square $\Longrightarrow$ not-prime?

   To be accepted.

5. *Suggested implication (comprises all attributes):*

   square, (not-prime), no-square $\Longrightarrow$

$\Longrightarrow$ **even, odd, (prime), (cubic), (not-cubic)?**

This implication also has to be accepted, because the premise contains attributes which negate each other.

6. *Suggested implication:*

   prime $\Longrightarrow$ no-square, not-cubic?

   Evidently, this implication also applies to our conceptual universe.

7. *Suggested implication (comprises all attributes):*

   prime, not-prime, (no-square), (not-cubic) $\Longrightarrow$
   $\Longrightarrow$ even, odd, (square), cubic?

   This implication also comprises all attributes and has to be accepted, because the premise contains attributes which negate each other.

8. *Suggested implication (comprises all attributes):*

   even, odd $\Longrightarrow$
   $\Longrightarrow$ prime, square, cubic, not-prime, not-square, not-cubic?

   This implication has been accepted automatically.

The attribute exploration ends at this stage. Seven implications have been accepted as base and one further counterexample has been generated. We name the context NumE2de. $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 15, 27, 64\}$ *is then an object set typical for the current conceptual universe. However, the object reduced (***R G***) object set* $\{1, 2, 3, 4, 6, 8, 9, 15, 27, 64\}$ *of the reduced context NumE2de is still typical for our universe, too. Furthermore, as mentioned earlier, these object sets have the property that every inquiry in the search mode yields at least one object (i.e. a number) as output, if and only if the conceptual universe contains a number which has or does not have the attributes marked accordingly (in this connection we could now even omit the attributes newly introduced in comparison to NumE2 (i.e. we can omit the negations of attributes from NumE2)). The reduced context for NumE2 and some lists of implications connected with it have already been represented in the previous section in Tables 8, 9 and 11. The concept lattice with fourty three concepts can be seen in Figure 5.*

In order to give a short introduction into one possible treatment of incomplete knowledge in connection with ConImp *we proceed as follows:*

- *We remove the attribute "cubic" from the context NumE2 (by ...* **Del** *... or ...* $^\wedge$**G** *... from within the context editor, or by means of the sub-program "Change the sequence and number of attributes" (***A R** *...)), and add three new attributes (by ...* **Ins** *... or* $^\wedge$**V** *... from within the context editor, or by means of the sub-program "Increase the number of attributes" (***A M** *...)):*

sum2even	*i.e.*	*"is the sum of two even numbers"*
sum2sqare	*i.e.*	*"is the sum of two squares"*
sum2prime	*i.e.*	*"is the sum of two primes"*

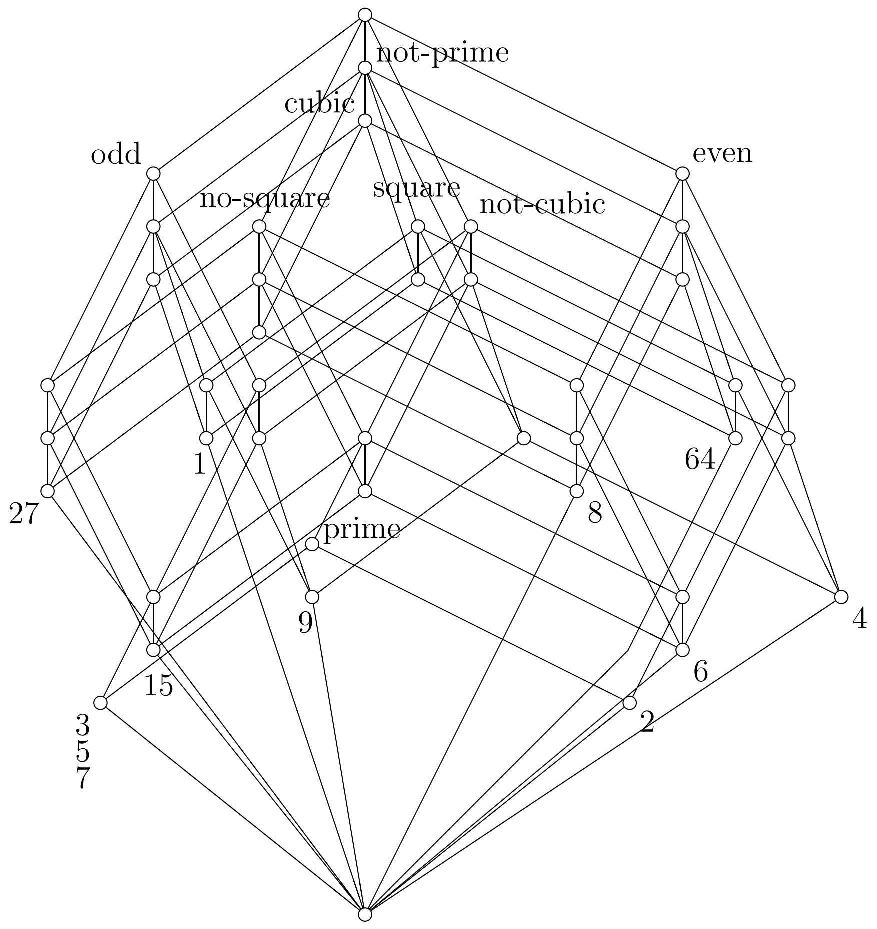

Figure 5: Line diagram for the context NumE2de

- *After completing the entries and subsequently reducing the context with respect to the set of objects (***R G***), we obtain the context NumE3*:= (G_3, M_3, I_3) *with the object set* $G_3 = \{1, 2, 3, 4, 5, 7, 8, 9\}$ *and the attribute set* $M_3 = \{$ *even, odd, prime, square, sum2even, sum2prime, sum2square*$\}$ *(cf. Table 13). In order to work on this new context, it has to be chosen as main context (***A H***). In this connection we have not introduced the attribute "*sum of two odd numbers*" since it is equivalent to the attribute "*even*" and we do not want the list of questions in the following attribute exploration to be too long).*

- *We shorten the description of the attribute exploration procedure by giving a short representation of the questions and answers in Table 12. In this context "***O***pt.strat." stands for "*The implication is accepted as uncertain by using the optimal strategy (**O**has been pressed)", *"aut." for "*accepted automatically". *On the left side, we state the consecutive number of an accepted implication, in the middle the answer, and on the right a possible counterexample (here, for some entries we give hints concerning the decision that the corresponding number is a counterexample in the context or we just observe that a proposed implication has been accepted automatically).*

- *If we do not want to enter all the implications we already know by hand, we can for example use the option "***I L S***" to load the implications computed for the context NumE2 into the implication editor (in as far as they are relevant) and add the new attributes to the conclusion of the (two) implications with inconsistent premises. Furthermore, we enter the following evident implications:*

  $$\textit{sum2even} \Longrightarrow \textit{even}$$
  $$\textit{sum2even, prime} \Longrightarrow M_3$$

  (in this connection we observe that 2 is the only even prime and the only even number which is not the sum of two even numbers).

- *Note that already the first question in the attribute exploration corresponds to the so-called Goldbach-Conjecture of number theory, which says that every even number which is unequal to 2 is the sum of two primes. This conjecture has neither been proved nor refuted, so that we must accept the proposal as uncertain. Variations of this question with larger premises come later.*

- *Having the explored context at hand, one can now try to eliminate question marks (***U A***). This yields the context NumE2eu.*

- *This is the best result we can get at the moment. The theory says*[43] *that part of the fictitious objects correspond to actual counterexamples in the universe and form together with the other objects a typical system of objects for the universe; and part of the remaining ones (mostly all of them) correspond to encoded implications such that those together with the accepted implications form an implicational base of the universe - not necessarily the Duquenne-Guigues-base,*

[43]Cf. [H01].

No.	*Proposal of an implication*	*Answer*	*Kind/Counterex.*
1.	sum2even $\Longrightarrow$ (even), sum2prime sum2even $\Longrightarrow$ even sum2even $\Longrightarrow$ sum2prime	**O**pt. strat. yes **O**pt. strat.	 ?Fict1
	square, sum2sqare $\Longrightarrow$ M_3	no	25=23+2 $=3^2+6^2$
	square, sum2sqare $\Longrightarrow$ odd, sum2prime	no	100=97+3 $=6^2+8^2$
	square, sum2sqare $\Longrightarrow$ sum2prime	no	289=7*41+2 $=15^2+8^2$
2.	prime, sum2prime $\Longrightarrow$ odd	yes	
3.	prime, square $\Longrightarrow$ M_3	yes	aut.
	odd, prime, sum2sqare $\Longrightarrow$ sum2prime	no	17=3*5+2 $=4^2+1^2$
4.	even, sum2prime $\Longrightarrow$ sum2even	yes	
	(even,) sum2even $\Longrightarrow$ sum2prime	**O**pt. strat.	?Fict2
	(even,) sum2even, sum2sqare $\Longrightarrow$ sum2prime	**O**pt. strat.	?Fict3
5.	even, square $\Longrightarrow$ (sum2even), sum2prime even, square $\Longrightarrow$ sum2even even, square $\Longrightarrow$ sum2prime	**O**pt. strat. yes **O**pt. strat.	 ?Fict4
	(even,) square, sum2even $\Longrightarrow$ sum2prime	**O**pt. strat.	?Fict5
	(even,) square, sum2even, sum2squar $\Longrightarrow$ $\Longrightarrow$ sum2prime	 **O**pt. strat.	 ?Fict6
6.	even, prime $\Longrightarrow$ sum2sqare	yes	
7.	(even,) prime, sum2even, sum2squar $\Longrightarrow$ M_3	yes	
8.	even, odd $\Longrightarrow$ M_3	yes	aut.
	End of program: 8 implications computed *6 open implications encoded in fictitious examples*		

Table 12: Attribute exploration for the context NumE3

but that can then easily be computed, once all the open questions have been solved.

- *The last observation shows that at the end of such an exploration —possibly with open questions encoded by fictitious examples— the results provide optimal information in as far as also all open questions are known (cf. also footnote 39).*

- *It may however happen that in some cases the exploration ends with a message (or has it in between) that some fictitious object contradicts some of the accepted implications or some of its consequences. Then one should erase that fictitious example at the end of the exploration (if it is done during the exploration, this may cause additional but unnecessary questions).*

- *Observe that before updating also the fictitious examples w.r.t. the accepted implications (***U A***) one should save the context with the fictitious examples directly resulting from the finished exploration, since after the updating the open questions are no longer fully recoverable from the fictitious examples — since then there may be more than one blank, and the conclusion is no longer seen witout a protocol file.*

It has been proved in [H01] that it is not possible to obtain more information from our expert by means of ConImp than just described.

```
NumE3     ss      NumE3e     ss      NumE3eu    ss
         suu                suu                suu
         umm                umm                umm
        sm22               sm22               sm22
       pq2ps              pq2ps              pq2ps
     e ruerq            e ruerq            e ruerq
     voiaviu            voiaviu            voiaviu
     edmrema            edmrema            edmrema
     ndeener            ndeener            ndeener
     -------            -------            -------
   1!.x.x...!         1!.x.x...!         1!.x.x...!
   2!x.x...x!         2!x.x...x!         2!x.x...x!
   3!.xx....!         3!.xx....!         3!.xx....!
   4!x..xxx.!         4!x..xxx.!         4!x..xxx.!
   5!.xx..xx!         5!.xx..xx!         5!.xx..xx!
   7!.xx..x.!         7!.xx..x.!         7!.xx..x.!
   8!x...xxx!         8!x...xxx!         8!x...xxx!
   9!.x.x.x.!         9!.x.x.x.!         9!.x.x.x.!
    ---------    ?Fict1!????x?.!    ?Fict1!x..?x?.!
                     25!.x.x.xx!        25!.x.x.xx!
                    100!x..xxxx!       100!x..xxxx!
                    289!.x.x..x!       289!.x.x..x!
                     17!.xx...x!        17!.xx...x!
                 ?Fict2!x???x?.!    ?Fict2!x..?x?.!
                 ?Fict3!x???xx.!    ?Fict3!x..?xx.!
                 ?Fict4!x??x??.!    ?Fict4!x..x??.!
                 ?Fict5!x??xx?.!    ?Fict5!x..xx?.!
                 ?Fict6!x??xxx.!    ?Fict6!x..xxx.!
                       ---------          ---------
```

Table 13: Formal context NumE3 and the explored one NumE3e and the updated context NumE3eu

So far we do not know about an attribute exploration with three-valued logic which has been been carried out with authentic examples — except for the case of incomplete knowledge; moreover an authentic example would be too extensive for this introduction —, therefore we just sketch a short "constructed" example in the following tables, where Table 14 shows the questions and answers. The subsequent Table 15 then shows the resulting context. Recall that one can choose a three-valued logic, when the start context is edited (**B** ...); or, if one has already edited a two-valued context or loaded one without question marks — which therefore is automatically treated as two-valued —, one may change to a three-valued logic in the change menu (**A L**).

The set M of attributes is given by

$$M := \{\, \textit{even, odd, <>2, pr, sq, sq+sq} \,\}\,,$$

where — in order to fit into the tables —

- *"<>2" means "unequal to 2",*
- *"pr", "sq" and "sq+sq" are abbreviations for "prime", "square" and "sum of two squares", respectively.*

When question marks can be used, it is possible to change counterexamples already entered earlier. As a support one is first asked, whether one wants to inspect the counterexamples entered so far, yet at this stage one cannot make any changes. Then one has to enter the name of the example which one really wants to change. Each time a new implication is accepted, ConImp tests all objects entered so far, as to whether question marks can be eliminated because of the new information — however, this does not apply to fictitious examples in order not to lose information.[44]

In our example this has been done twice with "100" and once with "25" — the intermediate entry lines have been put in parenthesis.

If one carries out this kind of attribute exploration and if all suggested implications which are valid in the conceptual universe have been accepted as certain by the user, one obtains a lattice which is isomorphic to the concept lattice of the conceptual universe and in which all attributes are correctly assigned to their attribute concepts. On the other hand, those objects in the table which still have a question mark in their "line", although all accepted implications have been taken into account (**U C**), can be omitted according to [H01] without losing information about the structure of the concept lattice of the universe; however, if one does not omit them but converts there the remaining question marks into blancs (or rather periods) (**U T**) or crosses (**U P**), — this is indicated in ConImp by adding a question mark to the beginning of the respective object name (this may be changed in later versions such that "?" then is appended) —, then such object names might not be assigned correctly to the object concepts generated by them in the universe.

In the above example "169" and "45" are such objects, where question marks remain at the end of the exploration. When all question marks are replaced by periods:

[44]Cf. [H01].

No.	Premise		Conclusion	True?	Comment:
1	$\underline{\text{sq}}$	⇒	$\underline{\text{<>2}}$	yes	**Implication No. 1**
2*	<>2, $\underline{\text{sq}}$, $\underline{\text{sq+sq}}$	⇒	$\underline{\text{even}}$, $\underline{\text{odd}}$, $\underline{\text{pr}}$	no	10th c.-ex. := **169**
3	<>2, $\underline{\text{sq}}$, $\underline{\text{sq+sq}}$	⇒	$\underline{\text{odd}}$, $\underline{\text{pr}}$	no	11th c.-ex. := 100
4	<>2, $\underline{\text{sq}}$, $\underline{\text{sq+sq}}$	⇒	$\underline{\text{pr}}$	no	1st modific. of 100
5	$\underline{\text{<>2}}$, $\underline{\text{pr}}$	⇒	$\underline{\text{odd}}$	yes	**Implication No. 2**
6	$\underline{\text{odd}}$	⇒	$\underline{\text{<>2}}$	yes	**Implication No. 3**
7	$\underline{\text{odd}}$, <>2, $\underline{\text{sq+sq}}$	⇒	$\underline{\text{pr}}$	no	12th c.-ex. := **45**
8*	$\underline{\text{odd}}$, <>2, $\underline{\text{sq}}$, $\underline{\text{sq+sq}}$	⇒	$\underline{\text{even}}$, $\underline{\text{pr}}$	no	13th c.-ex. := 25
9	$\underline{\text{odd}}$, <>2, $\underline{\text{sq}}$, $\underline{\text{sq+sq}}$	⇒	$\underline{\text{pr}}$	no	last modific. of **25**
10*	odd, <>2, $\underline{\text{pr}}$, $\underline{\text{sq}}$	⇒	$\underline{\text{even}}$, $\underline{\text{sq+sq}}$	yes	**Implication No. 4**
11	$\underline{\text{even}}$, $\underline{\text{pr}}$	⇒	$\underline{\text{sq+sq}}$	yes	**Implication No. 5**
12*	$\underline{\text{even}}$, <>2, $\underline{\text{sq}}$, $\underline{\text{sq+sq}}$	⇒	$\underline{\text{odd}}$, $\underline{\text{pr}}$	no	last modific. of **100**
13*	$\underline{\text{even}}$, $\underline{\text{odd}}$, <>2	⇒	$\underline{\text{pr}}$, $\underline{\text{sq}}$, sq+sq	yes	**Implication No. 6**

Table 14: An attribute exploration using three-valued logic

"?45" — the object of the corresponding one-valued context — gets by chance into the correct position in the concept lattice (see Figure 6), yet "?169", which should have the same properties as "25", and which therefore should generate the same concept in the final concept lattice, will not be assigned the attribute "odd", and therefore it will be assigned to another concept (the question mark added to the name indicates that one should be careful).

7 Further features of ConImp

In addition to the features explained in connection with the above examples, ConImp offers further options, which we will only briefly mention in this section.

For example one can modify an existing context in the following ways:

- One can add an attribute to the set of attributes which constitutes the "Or-combination" of a number of existing attributes (**A O**). ConImp automatically computes the respective column, one only has to enter the new name.
- One can replace the present context by its "complementary" context (**A K**), i.e. all existing attributes are negated by replacing crosses by dots and dots by crosses.
- In the case of certain modifications of the context (like **A R** or **A S**) changing the order and/or number of attributes or objects, respectively) it is possible to restore the original main context (**A A**) without having to reload the original context from the disk (yet this only works, if the "intermediate" context has neither been reduced nor clarified).

$\mathbb{K}$	even	odd	<>2	prime	square	sq+sq	Comment: entered
1	.	×	×	.	×	.	in advance
2	×	.	.	×	.	×	in advance
3	.	×	×	×	.	.	in advance
4	×	.	×	.	×	.	in advance
5	.	×	×	×	.	×	in advance
6	×	.	×	.	.	.	in advance
7	.	×	×	×	.	.	in advance
8	×	.	×	.	.	×	in advance
9	.	×	×	.	×	.	in advance
169	.	?	×	?	×	×	in 2nd proposal
(100	?	.	×	?	×	×	) in 3rd proposal
(100	?	.	×	.	×	×	) in 4th p.; 1st modif.
45	?	×	×	.	?	×	in 7th proposal
(25	.	×	×	?	×	×	) in 8th proposal
25	.	×	×	.	×	×	in 9th p.; 1st modif.
100	×	.	×	.	×	×	in 12th p.; 2nd modif.

Table 15: Attribute exploration using a three-valued context:
The resulting context still contains some question marks.

- In the case of a context with the same number of objects as it has attributes, it is possible to compute the "reflexive and transitive closure" of the incidence relation I of the given context (i.e. the "quasi-order" generated by the present context; however this only makes sense, when objects and attributes with the same (consecutive) numbers can be assumed to represent the same element) (**A Q**).

- Having entered some part of a context (**B** ... – or at least a list of attributes and a possibly empty list of objects) and a list of implications valid in that context (**I H E** ...), one can generate a formal context containing the given one and having the list of implications as generating set of all the implications valid in the generated context (**I I K**). In the case when the resulting context becomes too large (i.e. more than 255 objects), then it is reduced w.r.t. the objects during the run of the subprogram, and this might have the effect that some of the original objects might also be deleted. At the end of the procedure the user is asked, whether he or she wants to reduce the context and choose the result as main context. Any key other than <**ESC**> has this effect.

  This subroutine is useful in particular, whenever one has some "implicational logic" and wants to find the corresponding concept lattice or to find out which objects would be missing to have a context having the entered implications

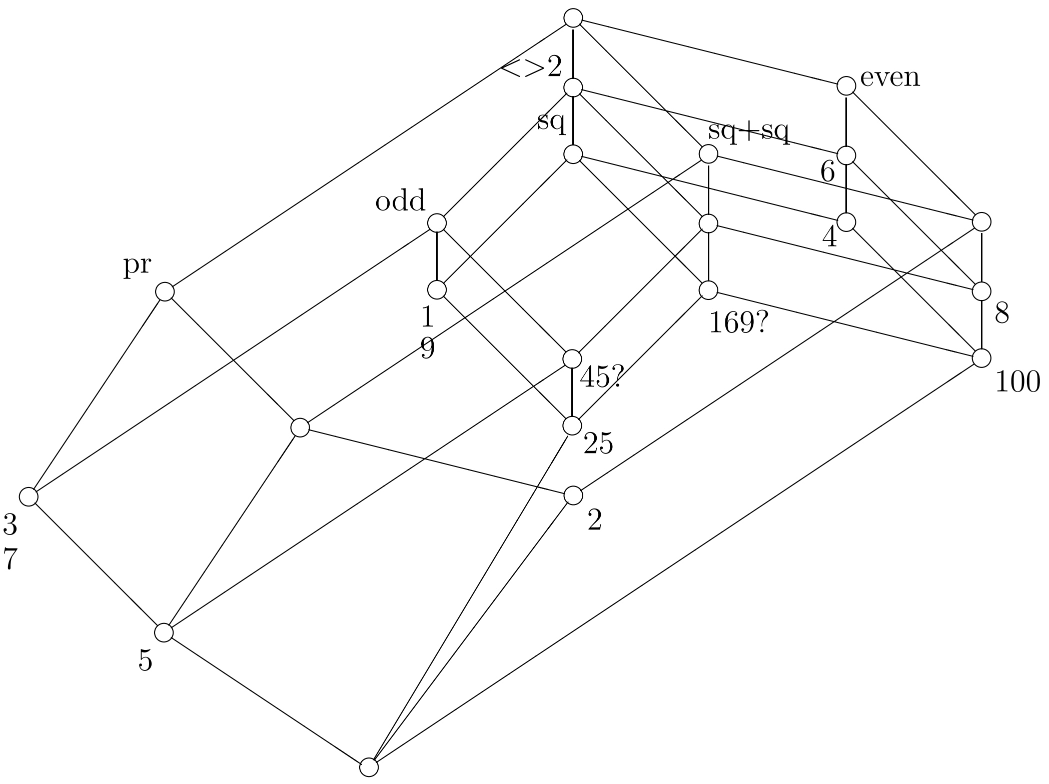

Figure 6: Concept lattice belonging to the three-valued context from Table 15

generating its valid implications.

By means of the **implication editor** (**I H** . . .) one can enter a candidate for an implication and then one can check whether or not it is valid in the present context (. . . **H**): one gets all those attributes from the entered conclusion (marked with “U”) which do not follow from the entered premise as well as those attributes (then marked by an “M”) which have not yet been entered but which can be inferred from the premise (w.r.t. the context), i.e. which lie in the closure of the premise computed w.r.t. the context.

Furthermore,

- one can modify the “**scroll parameters**” of the context editor (**A E**), i.e. the distance of the cursor from the margin of the context sector at which at new sector of the context is shown;
- in connection with the **loading** (**L** . . . or **I L** . . .) or **saving** (**S** . . . or **I S** . . .) of contexts or context data one can enter by ^**N** (= **Ctrl N**) into a submenue, where one can display all or parts of the contents of the folder chosen just before. One can do this while being in the subprogram for loading or saving contexts or implications at the place at which one has to enter the name of the file — or to accept the default);

- one can change the number of characters in a printed line (default ...: 80 characters); this is not only possible via the print menu (**D D** ...), but also via the alteration menu (**A Z**); the latter option is particularly important in case one has deactivated some check-backs of the program (**A F**), which also deactivates the question for the length of the printed line when printing computed data);
- one can look at some parameters of the program ConImp (the maximal number of attributes or objects and the maximal length of their names) and check which computations have already been carried out for the current context (**P**). If possible, the size of the (reduced) context, the length of the computed lists or the form of the effected reduction are also being shown.

At some places ConImp offers the possibility to call help screens (... ^**I**), by which one can get informations about what the subprograms under consideration are supposed to do or about the effect of some key strokes allowed in the submenu you are in.

Actually, one can use ConImp for quite a lot of further tasks. However, this short introduction stops here.

References

[B91] P. Burmeister. *Merkmalimplikationen bei unvollständigem Wissen.* In: W. Lex (Ed.), Arbeitstagung Begriffsanalyse und Künstliche Intelligenz, Informatik-Bericht 89/3. TU Clausthal, 1991, pp. 15–46.

[B00] P. Burmeister. ConImp — *Ein Programm zur Formalen Begriffsanalyse.* In: G.Stumme, R.Wille (Eds.): *Begriffliche Wissensverarbeitung: Methoden und Anwendungen.* Springer, 2000, pp. 25–56.

[BH00] P.Burmeister, R.Holzer. *On the treatment of incomplete knowledge in Formal Concept Analysis.* In: B.Ganter, G.W.Mineau (Eds.): *Conceptual Structures: Logical, Linguistic, and Computational Issues* (Proceedings of the ICCS 2000 at Darmstadt, August 2000), LNAI 1867, Springer, 2000, pp. 385–389.

[DGu86] V. Duquenne and J.-L. Guigues. *Familles minimales d'implications informatives résultant d'un tableau de données binaires.* Math. Sci. hum. 95, 1986, pp. 5–18.

[G87] B. Ganter. *Algorithmen zur Formalen Begriffsanalyse.* In: B. Ganter, R. Wille and K. E. Wolff (Eds.), Beiträge zur Begriffsanalyse. B.I.-Wissenschaftsverlag, Mannheim, 1987, p. 241-254.

[GW96] B. Ganter and R. Wille. *Formale Begriffsanalyse: Mathematische Grundlagen.* Springer-Verlag, Berlin Heidelberg, 1996.
English translation: *Formal Concept Analysis: Mathematical foundations.* Springer-Verlag, Berlin Heidelberg, 1999.

[H01] R.Holzer. *Methoden der formalen Begriffsanalyse bei der Behandlung unvollständigen Wissens.* Doctoral thesis, Darmstadt, February 2001, published at Shaker Verlag, Aachen 2001.

[M83] D. Maier. *The Theory of Relational Databases.* Computer Science Press, 1983.

[S89] M. Skorsky. *How to draw concept lattices with parallelograms.* In: R. Wille (Ed.), Klassifikation und Ordnung. Indeks-Verlag, Frankfurt, 1989, pp. 191–196.

[SpWo] N. Spangenberg, K. E. Wolff. Conceptual structures as indicators of change in the treatment of an anorectic patient. Manuscript.

[WaW92] C. Wachter and R. Wille. *Formale Begriffsanalyse von Literaturdaten.* In: DGD (Ed.), Deutscher Dokumentartag 1991 — Information und Dokumentation in den 90er Jahren: Neue Herausforderung, neue Technologien. Frankfurt, 1992, pp. 203–224.

[W82] R. Wille. *Restructuring lattice theory: an approach based on hierarchies of concepts.* In: I. Rival (Ed.), Ordered sets. Reidel, Dordrecht-Boston, 1982, pp. 445–470.

[W84] R. Wille. *Liniendiagramme hierarchischer Begriffssysteme.* In: H. H. Bock (Ed.), Anwendungen der Klassifikation: Datenanalyse und numerische Klassifikation. Indeks-Verlag, Frankfurt, 1984, pp. 32–51.

Index