Some variations on Alan Day's algorithm for calculating canonical basis of implications

Vincent Duquenne

CNRS-ECP6, Université Pierre et Marie Curie,
175 rue du Chevaleret, 75013 Paris, France
duquenne@math.jussieu.fr

Abstract. Two variations of an algorithm by Alan Day for reducing a list of implications regarding redundancy are given, with a new simple justification. All three algorithms have the property that the list can be reduced *in place* -at no extra memory cost- that will be useful for large applications and databases.

Keywords: basis of implications, closure operator, reduction in place, redundancy.

Introduction

Many years after having "introduced implication basis into FCA" (as noted in [GW99 p.94]) which dates back to the fall of 1983 (see [GD84-86], [G84-87], [D84-87]), it is surprising if not hard to acknowledge that we didn't learn a lot more on "their intimacy" in the meantime, despite many interesting papers using or revisiting them. In that respect, a special attention should be paid to one of the last papers by Alan Day ([Day92]), who spent several months of his spare time to clarifying the interest of *Lattice Theory* for putting *databases* into *canonical forms*, by decomposing them through *functional dependencies* and *relation schemes*. This was linked with other works (among which [W95]) that made precise the connections with functional dependencies ([MA83]) -classical in databases and AI-, to the FCA community. More recently we had ([D&A01], [O&D03-07], [V&D03-07]) to device some variations around Alan's algorithm and some pain to explain how they work which we now do.

Let A be a (finite) set of *attributes*, $L:=\{X_i \to Y_i \mid i \in I, X_i, Y_i \subseteq A\}$ be a *list of attribute implications*, and let consider the two closure operators:

(1) *L-closure*, $X \mid\!\to L(X)$ for all $X \subseteq A$, that is defined by reiteration of $X^L := X \cup \cup \{Y_i \mid X_i \to Y_i \in L, X_i \subseteq X\}$ up to reaching a fixpoint (hence finiteness…), namely $L(X) := X^{L \ldots L} = (X^{L \ldots L})^L$ which is the *consequence* of X, and similarly,

(2) *L-saturation* $X \mid\!\to L^\circ(X)$ for all $X \subseteq A$, $L^\circ(X) := X^{L^\circ \ldots L^\circ} = (X^{L^\circ \ldots L^\circ})^{L^\circ}$ defined by reiteration of $X^{L^\circ} := X \cup \cup \{L(X_i) \mid X_i \to Y_i \in L, X_i \subseteq X, L(X_i) \neq L(X)\}$.

The L-saturation $L^\circ(X)$ is a *restricted consequence* of X for "what is already known for smaller premises than X out of X's L-closed class". Notice how it is a bit harder to handle since dependent of the L-closure due to the condition $L(X_i) \neq L(X)$ that requires to have L(X) at hand when calculating $L^\circ(X)$ and will thus slow down algorithms.

An implication is *full* whenever its conclusion is L-closed, L itself is said to be *full* when $L:=\{X_i \rightarrow L(X_i) \mid i \in I, X_i \subseteq A\}$, which could be taken as a strong hypothesis, but is very natural when the input is a context. To avoid repeating $X_i \rightarrow Y_i \cup X_i$ all the time in the sequel, we always suppose that $L:=\{X_i \rightarrow Y_i \mid i \in I, X_i, Y_i \subseteq A\}$ is such that $X_i \subset Y_i$ ($i \in I$) and sometimes that L is full, specifying wherever results and algorithms can be extended to families of non-full implications. We freely mix together our original terminology and denotations with more established ones in FCA [GW99].

Now, $X=L^\circ(X)$ is called *L-saturated* for short. A saturated subset $L^\circ(X)$ is called *L-quasi-closed* when not closed $L^\circ(X) \neq L(X)$ –some authors did differently…-, in which case L(X) is called *essential L-closed* (or *meet-essential element* of the $\cap$-semi-lattice of L-closed subset, see [D84-87, D91]). $L^\circ(X)$ is called *L-pseudo-closed* when $\subseteq$-minimal L-quasi-closed in $\{Y \subseteq A \mid L(Y)=L(X)\}$. The set $B_L:=\{X \rightarrow L(X) \mid X$ L-pseudo-closed$\}$ is called the *canonical basis* of L (*saturated* moreover, if necessary, sometimes *Guigues-Duquenne basis* or – a new comer- *stem basis* in FCA's folklore).

The main result in [GD84-86] states that B_L is a minimal set of implications inferring L -for usual propositional calculus, or so-called *Armstrong rules*- and that moreover any such minimal family is in one-one correspondence with B_L through a natural construction, hence the name *canonical saturated basis* [D84-87 p.225] that was first chosen. Many authors revisited these notions introducing their own denotations. Here, we will not come back on that, but focus on some properties of *quasi / pseudo-closed* subsets that should be extended for bettering the algorithms.

It is now part of the folklore (see the above references) that:

Lemma 1. For a list L, a subset $H \subseteq L$ defines the same closure operator (has the same set of consequences) as L iff H has the same canonical basis as L iff for every L-pseudo-closed X there is at least one $(X_i \rightarrow Y_i) \in H$ for which $X_i \subseteq X \subset L(X)=H(X_i)$.

It provided the characterizations formulated in early drafts of [D84-87]:

Lemma 2. For a closure operator L() on A (or a list L of implications...):
1. $X=L^\circ(X)$ iff $L(X_i) \subset X$ for all $X_i \subseteq A$ s. t. $X_i \subset X$ and $L(X_i) \subset L(X)$.
2. $X=L^\circ(X)$ iff $L(X_i) \subset X$ for all L-pseudo-closed X_i s. t. $X_i \subset X$ and $L(X_i) \subset L(X)$.
3. X is L-pseudo-closed or L-closed iff $L(X_i) \subseteq X$ for all L-pseudo-closed $X_i \subset X$.

Remark. (2) reduces (1) to L-pseudo-closed. (3) gives a recursive definition of L-pseudo-closed subsets, that [G84-87], [GW99] took as a starting definition instead.

Now when starting from a list of implications, it will be enough to rewrite (1) as:

Lemma 3. For a list $L:=\{X_i \rightarrow Y_i \mid i \in I, X_i \subset Y_i \subseteq A\}$, $X=L^\circ(X)$
iff $L(X_i) \subset X$ for all $(X_i \rightarrow Y_i) \in L$ such that $X_i \subset X$ and $L(X_i) \subset L(X)$.

Our aim is to complete these basic properties for deriving either the canonical or some arbitrary basis of a list L, by dropping out redundant implications one at a time.

Reducing lists of full implications

Lists of full implications will be somehow easier to deal with thanks to the simple:

Lemma 4. For a full list of implications $L:=\{X_i \to Y_i = L(X_i) \mid i \in I,\ X_i \subset Y_i \subseteq A\}$, $(X_i \to Y_i) \in L$ and $H:=L \setminus \{X_i \to Y_i\}$,
1. $H(X_i) \neq L(X_i)$ iff
2. $L^\circ(X_i)$ is L-pseudo-closed and there is no other $(X_k \to Y_k) \in L$ with $L^\circ(X_k)=L^\circ(X_i)$.

Proof. Since $\{X_i \to Y_i \in L,\ X_i \subseteq X,\ L(X_i) \neq L(X)\} \subseteq H \subset L$, notice that

(*) for $X \subseteq A$ with $L(X)=L(X_i)$, $X \subseteq L^\circ(X) \subseteq H(X) \subseteq L(X)$ holds.

Suppose (2) fails. Case 1: $L^\circ(X_i)$ not-quasi-closed hence L-closed $L^\circ(X_i)=L(X_i)$ implies $H(X_i)=L(X_i)$ by (*). Case 2: $L^\circ(X_i)$ L-quasi-closed not pseudo-closed implies the existence of a L-pseudo-closed Z with $Z \subset L^\circ(X_i) \subset L(Z)=L(X_i)$, and by Lemma 1 the existence of at least one $(X_k \to Y_k) \in L$ with $L^\circ(X_k)=Z$ so that $H(X_i) \supseteq L(X_k)=L(X_i)$ hence $H(X_i)=L(X_i)$. Case 3: $L^\circ(X_i)$ pseudo-closed and the existence of some $(X_k \to Y_k) \in L$ with $X_k \neq X_i$ and $L^\circ(X_k)=L^\circ(X_i)$ implies $H(X_i)=H(X_k)=L(X_i)$. Conversely, suppose that (2) holds: since $L^\circ(X_i)$ is L-quasi-closed and minimal in its L-closed class for this property, for any $(X_k \to Y_k) \in L$ such that $X_k \subset L^\circ(X_i)$, $L(X_k) \subset L^\circ(X_i)$ must hold by Lemma 3, which implies $L^\circ(X_i)=H(L^\circ(X_i))$, so that by (*) it comes that $X_i \subseteq L^\circ(X_i)=H(L^\circ(X_i)) \subseteq H(H(X_i))=H(X_i)$, hence $H(X_i)=L^\circ(X_i) \neq L(X_i)$, by isotony of H() and $L^\circ(X_i)$ is L-pseudo-closed.

Algorithm 1.

Input a full family $L:=\{X_i \to Y_i = L(X_i) \mid i \in I,\ X_i \subset Y_i \subseteq A\}$.
Output: canonical basis $B_L:=\{X_k \to Y_k \mid k \in K,\ X_k$ L-pseudo-closed$\}$ (or an arbitrary one).

```
1. For i∈I
2.    L=L\{Xi→Yi}   /drop it out /
3.    X=L(Xi)          /see below Amendment 1/
4.    If X≠Yi Then
5.      L=L∪{X→Yi}  /restore it when Xi was the (last)/
6.    Endif          /generator of a pseudo-closed/
7. Endfor
```

We have used this algorithm in GLAD [D83-96] for years, specially in preparing [D&Al01], [O&D03-07], [V&D03-07]. It has been independently conceived in [R07].

Remarks. The negation of Lemma 4 (1) provides a simple criteria for dropping out redundant implications $X_i \to Y_i$ in L: when $H(X_i)=L(X_i)$. Notice that otherwise, $H(X_i)=L^\circ(X_i)$ is so to say automatically delivered L-pseudo-closed thanks to fullness. Moreover, to get a basis with smaller premises replace 5 by (5' $L=L\cup\{X_i \to Y_i\}$).

The morality of this procedure is that L-pseudo-closed are generated by either their *single* generator or *last examined* generator whenever several are existing in L, which is another explanation -out of their recursive nature- for their difficulty to be reached.

A main feature of this algorithm is that the reduction can be done in place. The price is to suppose the list L full, which takes … full benefit of transitivity and keeps tracks of implication consequences by isotony after they have been dropped out. The bonus are that the painful part (statement 3) reduces in time as implications are dropped out. There is no post-processing to get the pseudo-closed and this can be used to extract a basis made of the original implications, while preserving the redundant ones by permutation of L, separating L in two areas basis / redundant implications.

Amendment 1. Statements 3-4 in Algorithm 1 can be replaced by the following:

3'. X=L-conditional(X_i,Y_i,L,Restore)

4'. If (Restore=true) Then

where L-conditional is a function that reiteratively calculates $L(X_i)$ but cancels the calculus as soon as (if ever) the criteria $X=Y_i$ is reached within the iterative loop (returning Restore=false), and returns $L(X_i)$ otherwise (with Restore=true). As many implications are redundant in practice, this usually will save time.

Hence, in any circumstances where L is naturally full or can be made full at small price –which is quite often the case in FCA when the input is a context- this will avoid to calculate the L-saturation of the X_is by applying the definition and dealing continuously with both the painful restrictions $L(X_k)\neq L(X_i)$ and deadly reiteration. Fullness provides the clarity and efficiency of this simple algorithm.

Reducing non-full implications

Now, in the context of databases and AI, it may be the case that dependencies are expressed by non-full implications. For instance, see the new developments in [B06] that promote *canonical direct basis* for which the L-closure / saturation do not require reiteration, but are always reached with a single scan of the basis in construction.

For reducing such lists of implications some specific properties are required:

Lemma 5. Let $L:=\{X_i \to Y_i \mid i\in I,\ X_i \subset Y_i \subseteq A\}$ be a list, $(X_i \to Y_i)\in L$ and $H:=L\setminus\{X_i \to Y_i\}$, suppose moreover that $H(X)\subset L(X)$ for some $X\subseteq A$, then:

1. $X\subseteq X\cup X_i \subset X\cup Y_i \subseteq L(X)$ and $L(X)\supseteq L(X_i)$, and
2. $H(X)\supseteq X_i$, and
3. $H(X).\text{not.}\supseteq Y_i$ holds.

Proof. (1): $H(X)\subset L(X)$ implies that $X_i\to Y_i$ must take part in the iterative calculus of $L(X)$, so that $X\subseteq X\cup X_i\subset X\cup Y_i\subseteq L(X)$. By isotony and idempotence of $L()$, this implies $L(X)\subseteq L(X\cup X_i)\subseteq L(L(X))=L(X)$, hence $L(X)=L(X\cup X_i)\supseteq L(X_i)$ must hold. (2): By contraposition. Since $X\subseteq H(X)\subset L(X)$ and $L\backslash H=\{X_i\to Y_i\}$ hold, $H(X).\text{not}.\supseteq X_i$ implies $H(X)=L(X)$, a contradiction. (3): Similarly, $H(X)\supseteq Y_i\supset X_i$ implies $H(X)\cup X_i=H(X)\cup Y_i$, so that by contraposition of (1), $H(H(X))=L(H(X))$, but $H(H(X))=H(X)$ and $L(H(X))=L(X)$, hence $H(X)=L(X)$ should hold, a contradiction, so that $H(X).\text{not}.\supseteq Y_i$ holds as asserted.

This should be made a little bit more precise and gives some indications where the premises $X\subset A$ of non-redundant implications for which $H(X)\subset L(X)$ are, a potentiality that will be made clearer by the following:

Lemma 6. For a list $L:=\{X_i\to Y_i \mid i\in I,\ X_i\subset Y_i\subseteq A\}$, $(X_i\to Y_i)\in L$ and $H:=L\backslash\{X_i\to Y_i\}$,
1. either $H(X_i)\supseteq Y_i$, in which case $H(X_i)=L(X_i)$, $H()=L()$ and $X_i\to Y_i$ is L-redundant,
2. or $H(X_i).\text{not}.\supseteq Y_i$, in which case $H(X_i)\subset L(X_i)$ and $H(X_i)=L°(H(X_i))$ is L-quasi-closed and $X_i\subseteq L°(X_i)\subseteq H(X_i)=L°(H(X_i))\subset L(X_i)$ holds, that moreover collapses $L°(X_i)=H(X_i)$ whenever $L°(X_i)$ is L-pseudo-closed and there is no other $(X_k\to Y_k)\in L$ with $L°(X_k)=L°(X_i)$ and $Y_k.\text{not}.\subseteq L°(X_i)$.

Proof. (1): $H(X_i)\supseteq Y_i$ implies $H(X_i)=L(X_i)$ by contraposition of Lemma 5.3. Suppose that there exists some $X\subset A$ for which $H(X)\subset L(X)$, this implies $H(X)\supseteq X_i$ by Lemma 5.2, hence $H(X)\supseteq H(X_i)=L(X_i)\supseteq Y_i$ holds by isotony of $H()$, hence $H(X)=L(X)$ by contraposition of Lemma 5.3, a contradiction, so that $H()=L()$, and $X_i\to Y_i$ is therefore redundant in L. (2): Let $H(X_i).\text{not}.\supseteq Y_i$ holds. $X_i\subset Y_i\subseteq L(X_i)$ and $X_i\subseteq H(X_i)\subseteq L(X_i)$ imply that $H(X_i)=L(X_i)$ implies $H(X_i)\supseteq Y_i$, a contradiction, hence $H(X_i)\subset L(X_i)$ must hold. Suppose now that $H(X_i)$ is not L-quasi-closed. By Lemma 3 there must exist some $(X_k\to Y_k)\in L$ for which $X_k\subseteq H(X_i)$, $L(X_k).\text{not}.\subseteq H(X_i)$ and such that $L(X_k)\subset L(H(X_i))=L(X_i)$. $L(X_k)\subset L(X_i)$ implies $L(X_k).\text{not}.\supseteq L(X_i)$, hence $H(X_k)=L(X_k)$, by contraposition of Lemma 5.1. By isotony and idempotence of $H()$, $X_k\subseteq H(X_i)$ implies $X_k\subseteq H(X_k)\subseteq H(H(X_i))=H(X_i)$, so that $L(X_k)\subseteq H(X_i)$, a contradiction, $H(X_i)$ is thus L-quasi-closed. $X_i\subseteq L°(X_i)\subseteq H(X_i)\subset L(X_i)$ follows from (*) in Lemma 4's proof. Last, $L°(X_i)$ L-pseudo-closed and there is no other $(X_k\to Y_k)\in L$ with $L°(X_k)=L°(X_i)$ and $Y_k.\text{not}.\subseteq L°(X_i)$ implies $L°(X_i)=H(L°(X_i))$, the above inequalities imply that $X_i\subseteq L°(X_i)=H(L°(X_i))\subseteq H(X_i)\subset L(X_i)$ hold, but $L°(X_i)\supseteq X_i$ implies $H(L°(X_i))\supseteq H(X_i)$ by isotony, so that $H(X_i)=L°(X_i)$ must hold.

Remark. Lemma 6 (1) provides an efficient criteria for dropping out most redundant implications in L, and in particular all those $(X_i \to Y_i) \in L$ with $L(X_i)$ not-essential L-closed (of which the L-closed classes contain no L-quasi-closed subset). Lemma 6 (2) guaranties that replacing $X_i \to Y_i$ in L by $H(X_i) \to H(H(X_i) \cup Y_i) = L(X_i)$ gives an equivalent set of implications with a new premise $H(X_i)$ that is made automatically L-quasi-closed, without having to calculate any saturation actually.

Together with Lemma 1 and reiteration, this can be used to produce a superset of the canonical basis, and provides a new simple justification of the algorithm that is given at the end of [Day92 p. 426], of which the original setting –although focused on functional dependencies- is somehow complex and quite algebraic in nature, since it is expressed with sophisticated constructions on semi-lattice congruence relations:

Algorithm 2. [Alan Day 1992 p.426]

Input a family $L:=\{X_i \to Y_i \mid i \in I,\ X_i \subset Y_i \subseteq A\}$.

Output: an equivalent set of implications
$L:=\{X_k \to Y_k \mid k \in K,\ X_k$ L-quasi-closed$\}$.

```
1. For i∈I
2.   L=L\{Xi→Yi}     /drop it out /
3.   Xi=L(Xi)              /see above Amendment 1/
4.   If Yi.not.⊆Xi Then
5.     Yi=L(Xi∪Yi)   /restore non-redundant/
6.     L=L∪{Xi→Yi}   /remade full implications/
7.   Endif
8. Endfor
```

Remark. As observed by Alan Day, getting the canonical basis ("*critical*" basis in his own terms) out of this new amended list requires a post-processing to sort it and check premises' minimal property in their L-closed classes. This is due to the fact that a non-minimal quasi-closed can be obtained and kept in the list while a smaller pseudo-closed in the same L-closure class makes it becoming redundant afterwards. Hence, the reduction process is dependent of the order taken for scanning through L, and there is no way to insure that quasi-closed are tested in a $\subseteq$-compatible order.

However, a main advantage of this algorithm is to drop out a lot of redundant implications (statements 2-4), although it cannot detect the L-pseudo-closed sets on the fly, that could be a handicap for real scale applications and huge databases.

This can be avoided by using their recursive characterization in Lemma 2.3:

Algorithm 3.

Input a family $L:=\{X_i \to Y_i \mid i \in I,\ X_i \subset Y_i \subseteq A\}$.

Output: basis $B_L:=\{X_k \to Y_k \mid k \in K,\ X_k$ L-pseudo-closed$\}$.

```
1.  For i∈I
2.    L=L\{Xi→Yi}     /drop it out /
3.    Xi=L(Xi)              /see above Amendment 1/
4.    If Yi.not.⊆Xi Then
5.      Yi=Xi∪Yi      /restore non-redundant/
6.      L=L∪{Xi→Yi}  /possibly non-full implications/
7.    Endif
8.  Endfor
9.  SORT L by lexicographic order on Xi(i∈I)
10. Basis=∅           /namely: 00<01<10<11.../
11. For i∈I
12.   For (Xk→Yk)∈Basis    /is Xi L-pseudo-closed?/
13.     If Xk⊆Xi.and.Xi.not.⊃Yk Then
14.       L=L\{Xi→Yi}     /no: drop it out /
15.       Goto 20               /i.e. Endfor i/
16.     Endif
17.   Endfor
18. Yi=L(Yi)              /yes: make Xi→Yi full/
19. Basis=Basis∪{Xi→Yi}
20 Endfor
```

Remarks. As compared with Alan Day's Algorithm 2, the only new idea is to separate saturating the premises although keeping enough information from the original implications (first loop), from detecting the L-pseudo-closed -by using their recursive characterization (Lemma 2.3, which requires sorting L before, statement 9)- and closing their conclusions (second loop). Even if this sorting and detection (statements 12-17) have a cost, a major benefit is to L-close *only* the L-pseudo-closed (statement 18), which will be more efficient when L comprises far more implications than L-pseudo-closed, by saving the deadly price of reiteration in calculating L(). This algorithm can also be done in place by permuting L to avoid the extra table "Basis".

Discussion.

Starting from an (a priori) non-full list of implications $L:=\{X_i \to Y_i \mid i \in I,\ X_i \subset Y_i \subseteq A\}$ for calculating its canonical basis $B_L:=\{X_k \to L(X_k) \mid k \in K,\ X_k$ L-pseudo-closed$\}$, any algorithm will so far require $|I|+|K|$ closures, unless a sparing-closure-test is elaborated in the future to detect non-L-pseudo-closed premises at first sight...

Algorithm 1 requires $|I|$ closures, but half the work is already embedded and was supposed done by the fullness hypothesis, to be honest. Notice that in this note it is the only procedure giving optionally at no memory cost a basis with the original smaller premises -that may be crucial in some applications either for semantical reasons (because the user care them) or for optimization (cost, minimal generators…). L should usually shrink quickly during the reduction process since -in addition to redundant implications- L-quasi-non-pseudo-closed are dropped out at first sight thanks to fullness, which so doing provides a non order-dependent algorithm.

The best feature of Alan Day's Algorithm 2 is to start with over-saturating premises that drops out redundant implications of which the premise L-closure is not essential, leading to $|I|+|K'|$ closures, $|K'| \in [|K|, |I|]$, but is order dependent.

Algorithm 3 is a variation which requires exactly the optimal $|I|+|K|$ closures, after a pre-sorting -instead of post-sorting as for Alan's-, with an extra $|I| \times$ loops in the growing list to detect L-pseudo-closed premises, replacing iterative closures by simple loops, which follows the popular advice: "better clean before you close"!

All three algorithms share two nice properties: they don't calculate L-saturation actually -which would involve reiteration and presuppose the L-closure at hand- but do a test leading "automatically" to pseudo / quasi-closed premises, and secondly the reduction processes can be done in place, which is nice for building general programs.

In that respect, since the beginning of the development of our computer program GLAD (General Lattice Analysis & Design, see [D83-96]), a main concern has not been so much a pernickety fight against complexity, but to try understanding the interplay between the representations of algebraic / structural objects in programs, methodological questions ([D99]), and a search for simple / clearer algorithms. Alan's algorithm is very elegant and doesn't require sophisticated (often valued although exponential…) constructions, even if Alan (or his hidden editor) was perhaps a bit unfair when claiming that another alternative algorithm [NextClosure for implications that starts from a "*formal*" *context*] "provides an excellent (though *necessarily exponential time*) algorithm" [Day92 p.410]. Unfair, because … the input are not the same! If one reduces something (a list) which is close to our current goal (a non-redundant list), one is for sure in a better situation than wandering in the exponential powerset of all potential premises even with clever shortcuts… In practice, both types of algorithms are absolutely necessary depending on the context and applications.

Remaining questions for the future will be: Are there other ideas than fullness / pre-sorting to optimize Alan Day's algorithm? Combining different approaches and tricks, will there be other properties for identifying pseudo-closed premises quickly? Is there any hidden algebraic property or tool that could drive their detection?

So far, the three algorithms that we focused on in this note give another discrete clue on why / how pseudo-closed premises are hard to catch: as the ultimate survivors among the original list of candidates, may be they are just "the veracious core of what is left after throwing away the unnecessary gangue", in an Hegelian spirit...

Acknowledgements
The writing of this note was postponed for many years due to a special emotion. Its first motivation came out of discussions with Alan Day, along with his friends Christian Herrmann, Douglas Pickering and Michael Roddy who brought me to Thunder Bay -so scared and feeling such an amateur- in April 1990, a few months before Alan's untimely death. After open talks, accompanying us to his front door, Alan told me with a grave smile "The only thing which is wrong is me: so, we don't meet again". I could not swallow and until today have not been able to utter a word.

Years afterwards, a first draft was at last prepared for the workshop "*Le treillis Rochelais*", La Rochelle, May 2007, for which many thanks are due to the organizers. Thanks are also due to the referees for their remarks aiming at clarifying this paper.

References

[B06] Bertet K., Some algorithmic aspects using the canonical direct implicational basis, in *CLA'06: Concept Lattices and their Applications* (S. Ben Yahia and E. Mephu Nguifo eds), Hammamet (2006) 101-114.

[Day92] Day A., The lattice theory of functional dependencies and Normal decompositions, *Intern. J. of Algebra and Computations* 2 (4) (1992) 409-431.

[D84-87] Duquenne V., Contextual implications between attributes and some representation properties for finite lattices. in *Beitrage zur Begriffsanalyse*, (B. Ganter, R. Wille and K.E. Wolf eds), 1987, Mannheim: 213-239.

[D83-96] Duquenne V., GLAD (General Lattice Analysis & Design): a program for a glad user, *ORDAL 96*: *Order and decision-making* (I. Rival ed.), Ottawa, www.csi.uottawa.ca.

[D99] Duquenne V., Latticial structures in Data Analysis, *ORDAL 96*: *Order and decision-making* (I. Rival ed.), Ottawa,www.csi.uottawa.ca, *Theo. Comp. Sci. 217* (1999) 407-436.

[D&Al01] Duquenne V., Chabert C., Cherfouh A, Delabar J.-M., Doyen A.-L, Pickering D., Structuration of phenotypes / genotypes through Galois lattices and implications, in *Proc. of ICCS2001-CLKDD'01* (Stanford 07/2001, E. Mephu Nguifo et *al*. eds) 21-32, and *Applied Artificial Intelligence 17* (2003) 243-256.

[G84-87] Ganter B., Algorithmen zur Formalen Begriffsanalyse in *Beitrage zur Begriffsanalyse*, (B. Ganter, R. Wille and K.E. Wolf ed.), 1987, Mannheim: 213-239 (preprint TH Darmstadt, 1984).

[GW99] Ganter B., Wille R., *Formal Concept Analysis, Mathematical Foundations.* Springer Verlag, Berlin, 1999.

[GD84-86] Guigues J.L., Duquenne.V., Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques & Sciences Humaines 95* (1986) 5-18, (preprint Groupe Mathématiques et Psychologie, Université Paris V, 1984).

[Ma83] Maier D., *The theory of relational databases*, Computer Science Press, 1983.

[O&D03-07] Obiedkov S., Duquenne V., Incremental construction of the canonical basis, in *JIM'2003* (M. Nadif, A. Napoli, E. SanJuan, A. Sigayret eds) 15-23, and accepted in *Annals of Mathematics and Artificial Intelligence 49* (2007) 77-99.

[R07] Rudolf S., Some notes on pseudo-closed sets, in ICFCA'07: Formal Concept Analysis (S.O. Kuznetsov and S. Schmidt eds), Clermont-Ferrand (2007), LNAI 4390, 151-165.

[V&D03-07] Valtchev P., Duquenne V. Towards scalable divide-and-conquer methods for computing concepts and implications, in *JIM'2003* (M. Nadif, A. Napoli, E. SanJuan, A. Sigayret eds) 2-12.

[W95] Wild M., Computations with finite closure systems and implications. In *Proceedings of the 1st Annual International Conference on Computing and Combinatorics,* volume 959 of *LNCS,* pages 111-120. Springer, 1995.