Computing Proper Implications

Rafik Taouil[1] and Yves Bastide[2]

[1] INRIA Lorraine, F-54506 Vandœuvre-lès-Nancy Cedex, France
rafik.taouil@loria.fr
[2] LIMOS, université Blaise Pascal, F-63177 Aubière Cedex, France
yves.bastide@libd2.univ-bpclermont.fr

Abstract. This paper presents the proper implications: all implications holding on a set with a minimal left-hand side and a one-item right-hand side. Although not the smallest representation, they are easily readable and allow for some efficient selection and projection (embedding) operations.

The proposed algorithm, Impec, is designed to efficiently find proper implications given a set and a closure operator on this set. Additionally, it can be easily extended with a weight function or to compute embedded implications.

1 Introduction

Given a set and some closure operator on it, finding valid and—more to the point—interesting implications constitute long-standing research subjects, with immediate applications in such domains as database design and formal concept analysis.

Generating implications is not a problem; the problems come when one wants to use them. It is a well-known fact that except for toy examples, the set of implications is far too big to be easily manageable, both technically and from the point of view of an analyst. Researchers have devised lots of ways to overcome this difficulty: from ignoring provably redundant ones [18, 11], to assigning some measure of "usefulness" to each and pruning uninteresting ones—the data mining

approach [1, 3]—, to using rough sets techniques [15, 17], and so on. Most of these techniques are usually combined [16].

One lossless reduction strategy is the use of covers, that is, smaller sets of implications that still embody the whole information.

In this paper, we will present a particular cover for implications: the *proper implications*. They have a minimal left-hand side and a one-item right-hand side. As said above, a cover represents all the implications that hold, discarding no information. Although this particulary one is neither new nor the smallest possible one, we think that it offers a number of advantages that cannot be overlooked. Some of these are that the proper implications are informative, that they can quickly be put in a form used in database design [19], and that they are easy to project onto a subset of the attributes (this is commonly called *embedding* implications). We propose the Impec algorithm, which computes efficiently the proper implications.

1.1 Related Works

The most popular covers for implications are non-redundant ones. Among them, one well studied is the Duquenne-Guigues basis [11], as computed by Ganter's Next-Closure algorithm [8]. This algorithm is also independent of the closure operator. Another non-redundant cover corresponds in fact to the proper implications after removing the redundancy, and was proposed in the context of functional dependencies [18, 20]. Carpineto and Romano also proposed an algorithm to compute it [5]. These algorithms implicitely depend on a closure associated with their particular input.

Proper implications are discussed by e. g. Ganter and Wille [9] and Carpineto et al. [5, 6].

1.2 Contribution

The aim of this paper is twofold. First, we wish to emphasize the usefulness of proper implications. Secondly, we propose the Impec algorithm to compute them. This algorithm works for any closure operator; it can trivially be extended to use weight functions; and it can compute either all the proper implications or only those embedded in a subset.

The rest of the paper is organized as follows. The next section presents the motivations which lead to using the proper implications. Section 3 recalls some definitions on implications in terms of closure operators; Section 4 then formally describes the proper implications. Section 5 shows how they fit the goals given previously. Section 6 describes the Impec algorithm. Finally, Section 7 concludes and discusses some further developments in this area.

2 Motivations

Some motivations in the choice of a particular subset of the implications are presented heres. Firstly, this subset should be small, as compared to the whole set of implications.

Secondly, the chosen implications must be useful. This term has several meanings; here, we want all non-trivial implications in an easily readable form (an end-user should not need to apply Armstrong's axioms [2]).[1]

A third factor of choice would be the applicability of a *measure of interestingness* (and its use in the algorithm). In the context of data mining, such a measure could be the support [1]. More generally, we can use any weight function compatible with the closure operator [21].[2]

[1] An alternative is to read the implications on the Hasse representation of the concept lattice [22]. However, this requires the user to get accustomed, and is hard to scale.

[2] A weight function w compatible with a closure operator φ is a mapping from the power-set of X into $\mathbb{N}$, anti-monotonous w. r. t. φ.

Finally, here are two important use cases. Assuming the implications are stored in a database, and that they are expressed as $A \to c$ where c is a single attribute, one of them is "Given the conclusion c, which are the rules $A \to c$?" (This is only the simplest form of query on the implications; an example of selection language is the KESO system, by Klemettinen et al. [13].) Another useful case is projecting the implications on a subset of the initial set on which they hold. The embedded implications found should keep their other properties.

Well, not surprisingly, proper implications are interesting with respect to the factors above; their only drawback is their number.

3 Concepts and Notations

This section briefly restates some definitions for closure operators and implications. A thorough treatment can be found in Ganter and Wille's book [9]. The notation $\mathfrak{P}(X)$ means the power-set of X.

A *closure operator* φ on a set X is a map from $\mathfrak{P}(X)$ into $\mathfrak{P}(X)$ that is monotonous, extensive and idempotent. The set of closed subsets of X forms a lattice.

Let X be a finite set. An *implication* in X is a pair of subsets $(A, B) \subseteq X$ noted $A \to B$. Given a closure operator φ on X, an implication $A \to B$ *holds in* X *according to* φ if and only if $B \subseteq \varphi(A)$. The set $\mathfrak{L}_\varphi$ of all implications holding in X according to φ is closed, and follows Armstrong's axioms [2]: reflexivity, augmentation, and pseudo-transitivity. Conversely, given a set of implications $\mathfrak{L}$ in X, there exists a unique minimal complete system $\mathfrak{L}^+$ of $\mathfrak{L}$ that verifies Armstrong's axioms, and thus holds in X according to some closure operator.

A set of implications $\mathfrak{G}$ is called a *cover* of $\mathfrak{L}$ if $\mathfrak{G}^+ = \mathfrak{L}^+$. Finally, $\mathfrak{L}$ is *redundant* if it contains r such that $(\mathfrak{L} \setminus \{r\})^+ = \mathfrak{L}^+$.

4 Proper Implications

In the framework given previously, our goal is of course to find a small cover for the implications. The smallest possible covers, or bases, are non-redundant; they are also non-unique. As a matter of fact, the classical algorithm for computing a basis from a set of implications [4] is order-dependent. In contrast, proper implications are wel behaved but redundant.

Definition 1. *Given a finite set X and a closure operator φ on $\mathfrak{P}(X)$, the set of implications:*

$$\mathcal{L}_{\mathrm{p}}(\varphi) = \{\, A \to b \mid A \subseteq X \text{ and } b \in X \setminus A \text{ and } \forall Z \subset A, Z \to b \notin \mathcal{L}_{\varphi} \,\} \tag{1}$$

is called the set of proper implications *associated with φ.*

5 Appropriateness of the Proper Implications

We will use an example in the sequel. Table 1 represents an *objects* × *attributes* relation; and we will apply the Galois closure [9] on it.

Table 1. Example of relation

1	a c
2	a b
3	b c d
4	d e

As stated previously, and as noted by other authors, proper implications can be far more numerous than a basis' implications (see e. g. [5] for hard numbers). In fact, the ratio between these sizes can be exponential [7]. In our example,

proper implications are given in Table 2; Table 3 is their reduction to non-redundant implications; and Table 4 represents the Duquenne-Guigues basis. (In Table 4, implications with a common left-hand side are grouped together.)

Table 2. Proper implication holding on the example

$bc \rightarrow d$	$abc \rightarrow e$	$ad \rightarrow b$	$ad \rightarrow c$	$ad \rightarrow e$
$bd \rightarrow c$	$cd \rightarrow b$	$e \rightarrow d$	$ae \rightarrow b$	$ae \rightarrow c$
$be \rightarrow a$	$be \rightarrow c$	$ce \rightarrow a$	$ce \rightarrow b$	

Table 3. Derived basis for the example

$bc \rightarrow d$	$abc \rightarrow e$	$ad \rightarrow b$	$bd \rightarrow c$
$cd \rightarrow b$	$e \rightarrow d$	$be \rightarrow a$	

Table 4. Duquenne-Guigues basis for the example

$bc \rightarrow d$	$bd \rightarrow c$	$cd \rightarrow b$
$ad \rightarrow bce$	$e \rightarrow d$	$bcde \rightarrow a$

In spite of their number, proper implications have a number of advantages.

– They are very readable. For a given right-hand side item, the corresponding left-hand sides are all present in a minimal form. In contrast, the use of a basis will mean applying at least pseudo-transitivity. (It can be argued that the rules with a minimal left-hand side are not necessarily the most informative ones, particularly in the presence of measures of interestingness like the confidence; see, for example, the "A-Maximal" implications by Bayardo

and Agrawal [3], which have a maximal left-hand side with respect to their confidence.)

On the example, the answer to the query "What imply a?" is the set $\{be, ce\}$. Using the derived basis, only be is directly available; ce would be obtained by using pseudo-transitivity. With the Duquenne-Guigues basis the answer given is $bcde$: this set is non-minimal.

– Proper implications allow for polynomial retrieval of embedded implications. Given Y, a subset of the set X on which the proper implications were computed, the proper implications embedded in Y are simply those which contain only items from Y. By contrast, the general case is exponential [10]. Besides their immediate usefulness in an exploration framework, embedded implications are used for instance in data-warehousing [23, 14]. (The authors are aware of two algorithms for computing embedded implications: RBR by Gottlob [10], and Impec.)

  Here, the set of proper implications embedded in $\{a, d, e\}$ is $ad \to e, e \to d$, not directly visible on either bases.

– In the context of functional dependencies, a *canonical basis* is a cover which is non-redundant and whose left-hand sides are minimal and right-hand sides contain one attribute [18].[3] One of their uses is in designing third normal form database relations [4, 19]. Getting a canonical basis from the proper implications is polynomial [4, 5]. This is the basis presented in Table 3.

6 The Impec Algorithm

We will now present the Impec algorithm: Given a set X and a closure operator φ on X, it computes $\mathfrak{L}_{\mathrm{p}}(\varphi)$, the set of proper implications corresponding to φ.

[3] This is different from the Duquenne-Guigues basis, which is also sometimes called by the same name.

For $A \subset X$, let $\mathrm{rhs}(A)$ be the set of all its right-hand sides in $\mathcal{L}_{\mathrm{p}}(\varphi)$:

$$\mathrm{rhs}(A) = \{\, x \in X \mid A \to x \in \mathcal{L}_{\mathrm{p}}(\varphi) \,\} \ . \tag{2}$$

The Impec algorithm computes the family of implications:

$$\mathcal{L} = \{\, A \to \mathrm{rhs}(A) \mid \mathrm{rhs}(A) \neq \emptyset \,\} \ . \tag{3}$$

It is not difficult to see that any set A such as $\mathrm{rhs}(A) \neq \emptyset$ is minimal among the sets having the same closure.

The following proposition is used by the algorithm to ensure that it only uses such minimal sets:

Proposition 1. *Let $A, B \subset X$ such that $A \subset B$. Then we have:*

$$\varphi(B) = \varphi(A) \text{ if and only if } B \setminus A \subseteq \varphi(A) \setminus A \ . \tag{4}$$

Proof. We have: $\varphi(B) = \varphi(A \cup (B \setminus A)) = \varphi(\varphi(A) \cup (B \setminus A))$. Thus, $B \setminus A \subseteq \varphi(A)$ is equivalent to $\varphi(B) = \varphi(A)$. □

To compute $\mathrm{rhs}(A)$, we will use this proposition:

Proposition 2. *Let $A \subset X$. Then we have:*

$$\mathrm{rhs}(A) = \varphi(A) \setminus (A \cup \bigcup \{\, \mathrm{rhs}(B) \mid B \subset A \,\}) \ . \tag{5}$$

Proof. This proposal is presented, with a proof, in a slightly different form in Ganter and Wille's book [9, Proposition 22]. □

The Algorithm The pseudo-code of Impec is given in Algorithm 1. Line 1 initializes the set of pairs $(A, \mathrm{rhs}(A))$ with the empty set. The following loop looks at

each attribute in the set X. We initially suppose that all pairs $(A, \text{rhs}(A))$ computed so far will be used (line 3). For each of them, we compute a putative right-hand side Z. Assuming Z is not empty, the two loops in lines 7–11 and 12–14 apply Propositions 1 and 2, respectively. The resulting pair $(A \cup \{x\}, \text{rhs}(A \cup \{x\}))$ is eventually added to the set $\mathcal{L}$ (line 16). (Note that this is needed even if $\text{rhs}(A \cup \{x\})$ is empty, as we will see in the example.) In the end, all pairs with a non-empty right-hand side are returned.

Algorithm 1 Impec

Input: X: Set.
φ: Closure operator on X.
Output: $\mathcal{L}$: (Combined) proper implications corresponding to φ.

```
 1: L := {(∅, φ(∅))}
 2: for all x ∈ X do
 3:    J := L
 4:    for all (A, rhs(A)) ∈ J do
 5:       Z := φ(A ∪ {x}) \ (A ∪ {x})
 6:       if Z ≠ ∅ then
 7:          for all (B, rhs(B)) ∈ J with B ⊃ A do
 8:             if B \ A ⊆ Z then
 9:                J := J \ {(B, rhs(B))}
10:             end if
11:          end for
12:          for all (B, rhs(B)) ∈ J with B ⊂ A ∪ {x} do
13:             Z := Z \ rhs(B)
14:          end for
15:       end if
16:       L := L ∪ {(A ∪ {x}, Z)}
17:    end for
18: end for
19: return ⋃{ (A, rhs(A)) ∈ L | rhs(A) ≠ ∅ ∧ A ≠ ∅ }
```

Example 1. Table 5 shows the different steps of the algorithm on the example from Table 1, using the Galois closure. $\mathcal{L}$ contains initially $(\emptyset, \emptyset)$ as $\varphi(\emptyset) = \emptyset$. $\emptyset \cup \{a\}$ is $\{a\}$ and $\varphi(\{a\} \setminus \{a\}$ is empty, so the pair $(\{a\}, \emptyset)$ is added to $\mathcal{L}$ (we see here why $(\emptyset, \emptyset)$ was put in $\mathcal{L}$; otherwise, the loop would yield nothing). The

same holds for b, ab c and ac. The first nonempty right-hand side occurs with bc: there, $\varphi(\{b,c\}) = \{b,c,d\}$, and $(\{b,c\},\{d\})$ is added to $\mathcal{L}$. We can see pruning in action with $x = d$ and $A = \{a\}$ ($\varphi(\{a,d\}) = \{a,b,c,d,e\}$): For $B = \{a,b\}$, $\{b\}$ is a subset of $\{b,c,e\}$; thus $\{a,b,d\}$ will not be considered. The same thing happens for $B = \{a,c\}$ and $B = \{a,b,c\}$.

Table 5. Impec steps on the example

x	$(A, \mathrm{rhs}(A))$	Z	$(A \cup \{x\}, Z)$
a	$(\emptyset,\emptyset)$	$\emptyset$	$(\{a\},\emptyset)$
b	$(\emptyset,\emptyset)$	$\emptyset$	$(\{b\},\emptyset)$
	$(\{a\},\emptyset)$	$\emptyset$	$(\{a,b\},\emptyset)$
c	$(\emptyset,\emptyset)$	$\emptyset$	$(\{c\},\emptyset)$
	$(\{a\},\emptyset)$	$\emptyset$	$(\{a,c\},\emptyset)$
	$(\{b\},\emptyset)$	$\{d\}$	$(\{b,c\},\{d\})$
	$(\{a,b\},\emptyset)$	$\{d,e\}$	$(\{a,b,c\},\{e\})$
d	$(\emptyset,\emptyset)$	$\emptyset$	$(\{d\},\emptyset)$
	$(\{a\},\emptyset)$	$\{b,c,e\}$	$(\{a,d\},\{b,c,e\})$
	$(\{b\},\emptyset)$	$\{c\}$	$(\{b,d\},\{c\})$
	$(\{c\},\emptyset)$	$\{b\}$	$(\{c,d\},\{b\})$
e	$(\emptyset,\emptyset)$	$\{d\}$	$(\{e\},\{d\})$
	$(\{a\},\emptyset)$	$\{b,c,d\}$	$(\{a,e\},\{b,c\})$
	$(\{b\},\emptyset)$	$\{a,c,d\}$	$(\{b,e\},\{a,c\})$
	$(\{c\},\emptyset)$	$\{a,b,d\}$	$(\{c,e\},\{a,b\})$

Complexity Let τ_φ be the maximal cost of calling the closure operator, this cost being assumed greater than the cost of operations in line 5. Let us assume that the loops lines 7 and 12 both cost at most $|\mathcal{J}| \times |X|$ basic operations.[4] Lines 3, 9 and 16 are either done in constant time or negligible compared with their neighborhood. Then, the time complexity of each iteration over one attribute x from X is in $O(|\mathcal{L}| \times (\tau_\varphi + |\mathcal{L}| \times |X|))$. We know that $|\mathcal{L}|$ is bounded by $2^{|X|}$ (and

[4] Their actual cost is of course far lower when using e. g. a *trie* with transversal links [12].

hopefully much lower). The whole algorithm has thus a complexity bounded by

$$O(|X| \times 2^{|X|}(\tau_\varphi + 2^{|X|} \times |X|)) \ . \tag{6}$$

Extensions We will consider here two small extensions to the Impec algorithm: restricting the generated implications to those where the left-hand side satisfies some condition, and generating embedded implications.

- A predicate can be applied to $A \cup \{x\}$ at each pass, such as a minimum support value in a data mining context. It must be compatible with the closure operator (that is, if the predicate is false for a given left-hand side, it is also false for its supersets). This how to use a weight function. See also Luxenburger [16].
- We saw above (in Section 5) how to select embedded implications from the set of all proper implications. An alternative would be to use the Impec algorithm to compute them: for this, the only modifications are to replace X by the subset Y in line 2 and the assignment to Z by $Z := \varphi(A \cup \{x\}) \setminus (A \cup \{x\}) \cap Y$ in line 5.

7 Conclusion

In this paper, we presented the proper implications: they are implications with a minimal left-hand side and a 1-item right-hand side. They are a redundant cover of the whole set of rules, sometime much bigger than a non-redundant one; they have also some significant advantages over them, in terms of readability and of efficiency of applying transformations on them (such as finding a basis or embedding them).

The Impec algorithm proposed here is designed to compute the proper implications given a set and a closure operator on this set. It can be easily extended

with a weight function, or (among its uses) applied to find embedded proper implications.

Future work includes developing a framework in which an end-user can use the proper implications for knowledge exploration.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, pages 207–216, May 1993.
2. W. W. Armstrong. Dependency structures of data base relationships. *IFIP Congress*, pages 580–583, 1974.
3. R. Bayardo and R. Agrawal. Mining the most interesting rules. In *Proc. of the 5th Int'l Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 145–154, August 1999.
4. P. A. Bernstein. Synthesizing third normal form relations from functional dependencies. *TODS*, 1(4):277–298, 1976.
5. C. Carpineto and G. Romano. Inferring minimal rule covers from relations. In *Proc. of the 5th Congress of the Italian Association for Artificial Intelligence (AI*IA)*, volume 1321 of *LNAI*, pages 147–158, September 1997.
6. C. Carpineto, G. Romano, and P. D'Adamo. Inferring dependencies from relations: a conceptual clustering approach. *Computational Intelligence*, 15(4), 1999.
7. P. C. Fischer, J. H. Jou, and D. M. Tsou. Succinctness in dependency systems. *TCS*, 24:323–329, 1983.
8. B. Ganter. Two basic algorithms in concept analysis. Technical report, Technische Hochschule Darmstadt, 1984.
9. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations.* Springer, 1999.
10. G. Gottlob. Computing covers for embedded functional dependencies. In *Proc. 6th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 58–69. ACM, March 1987.
11. J.-L. Guigues and V. Duquenne. Famille minimale d'implication informatives résultant d'un tableau de données binaires. *Math. Sci. Hum.*, 24(95):5–18, 1986.
12. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, pages 1–12, May 2000.
13. M. Klemettinen, H. Mannila, and A. Inkeri Verkamo. Association rules selection in a data mining environment. In *Proc. of the 3rd European PKDD Conf.*, number 1704 in LNAI, pages 372–377, September 1999.
14. D. Laurent, J. Lechtenbörger, N. Spyratos, and G. Vossen. Complements for data warehouses. In *Proc. of the 15th Int'l Conf. on Data Engineering (ICDE)*, pages 490–499, 1999.
15. T. Y. Lin and N. Cercone, editors. *Rough sets and data mining.* Analysis of Imprecise Data. Kluwer Academic Publishers, 1996.

16. M. Luxenburger. Implications partielles dans un contexte. *Math. Inf. Sci. Hum.*, 29(113):35–55, 1991.
17. M. M. Kryszkiewicz. Representative association rules. In *Proc. 2nd Pacific-Asia Conf. on Research and Development in Knowledge Discovery and Data Mining (PAKDD)*, pages 198–209, April 1998.
18. D. Maier. Minimum covers in the relational database model. *JACM*, 27:664–674, 1980.
19. H. Mannila and K.-J. Räihä. Algorithms for inferring functional dependencies from relations. *Data and Knowledge Engineering*, 1994.
20. H. Mannila and K.-J. Räihä. *The design of relational databases.* Addison-Wesley, 1994.
21. G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Fast computation of concept lattices using data mining techniques. In *Proc. of the 7th Int'l Workshop on "Knowledge Representation meets Databases" (KRDB)*, pages 129–139, August 2000.
22. F. Vogt and R. Wille. TOSCANA — a graphical tool for analyzing and exploring data. In *Proc. GD '94*, volume 894 of *LNCS*, pages 226–233. Springer, October 1995.
23. Y. Zhuge, H. Garcia-Molina, J. Hammer, and J. Widom. View maintenance in a warehousing environment. In *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, pages 316–327, 1995.