mathematics

MDPI

Article

Simplifying Implications with Positive and Negative Attributes: A Logic-Based Approach

Francisco Pérez-Gámez, Domingo López-Rodríguez, Pablo Cordero, Ángel Mora and Manuel Ojeda-Aciego *

Departamento Matemática Aplicada, Universidad de Málaga, 29071 Málaga, Spain; franciscoperezgamez@uma.es (F.P.-G.); dominlopez@uma.es (D.-L.R.); pcordero@uma.es (P.C.); amora@uma.es (Á.M.)

* Correspondence: aciego@uma.es

Abstract: Concepts and implications are two facets of the knowledge contained within a binary relation between objects and attributes. Simplification logic (SL) has proved to be valuable for the study of attribute implications in a concept lattice, a topic of interest in the more general framework of formal concept analysis (FCA). Specifically, SL has become the kernel of automated methods to remove redundancy or obtain different types of bases of implications. Although originally FCA used only the positive information contained in the dataset, negative information (explicitly stating that an attribute does not hold) has been proposed by several authors, but without an adequate set of equivalence-preserving rules for simplification. In this work, we propose a mixed simplification logic and a method to automatically remove redundancy in implications, which will serve as a foundational standpoint for the automated reasoning methods for this extended framework.

Keywords: formal concept analysis; implicational systems; negative attributes

Citation: Pérez-Gámez, F.; López-Rodríguez, D.; Cordero, P.; Mora, Á.; Ojeda-Aciego, M. Simplifying Implications with Positive and Negative Attributes: A Logic-Based Approach. *Mathematics* **2022**, *10*, 607. https://doi.org/10.3390/math10040607

Received: 15 December 2021
Accepted: 12 February 2022
Published: 16 February 2022

1. Introduction

Since the 1980s, formal concept analysis (FCA) has been a solid framework to analyze data and extract hidden knowledge, comparable to other well-known techniques in terms of cost. Given a binary table, called *formal context*, FCA can build ontologies similar to AI-based knowledge representation methods [1] but with a solid *algebraic* structure, in which order theory and logic are the main tools: given a formal context, FCA builds a hierarchical structure of *concepts*, the so-called *concept lattice* (indeed a complete lattice), which encompasses all the information in the formal context. Moreover, in the same process, FCA returns sets of implications and/or association rules (well-known in other areas such as data mining, machine learning, and rough set theories) with a rich algebraic framework in which we can also compute closed sets and their minimal generators, pseudointents, different types of bases, etc. This knowledge can reveal interesting patterns and solve significant problems in modern areas as *social network analysis* [2,3] or *recommender systems* [4,5].

In the classical framework of FCA, the formal context is a binary relation between the elements of two sets, that is, a relationship between a set of *objects* and a set of *attributes*, establishing the properties that each object does satisfy (namely, *positive information*). Notwithstanding, sometimes the properties that are not satisfied for each object (*negative* information) are also relevant. For instance, in a table in which objects are birds, the object "ostrich" shall not have the property (attribute) "fly". That is, for "ostrich", not only the positive information is relevant ("large", "heavy", "fast"); the negative information ("does not fly") is also relevant.

The management of negative attributes in association rules already appeared in [6,7]. Mining *concise* sets of association rules (also called *bases*) is of particular importance in the machine learning community, and, to this end, several works have been proposed that, on top of the classical *minimum support–minimum confidence* strategy, impose other measures of

interestingness [8] or *informativeness* [9] based on statistical parameters, which help in the pruning of frequent item sets when determining a representative set of association rules.

In our work, we focus on *exact* association rules (also called *implications*); hence, we resort to the framework of FCA, where the first occurrences of negative information appear in Missaoui et al. [10,11], in which the authors computed mixed implications (with positive and negative attributes) from a double context formed by the initial context together with its opposite. This last approach can generate a huge number of redundant rules and algorithms executed with increased execution time. Moreover, and more importantly, the relationship between positive and negative information (mixed attributes) is not considered.

In [12], Rodríguez et al. proposed a generalization of classical FCA to consider both positive and negative attributes together with the relationship between them. New derivation operators and a Galois connection was defined to extend the classical framework. In addition, algorithms to compute the *mixed concept lattice* and *mixed implications* were proposed; furthermore, an axiomatic system based on the simplification paradigm was proposed to manage mixed implications [13] .

In different areas, such as artificial intelligence, database theory, data mining, and machine learning, one of the main problems is the huge degree of redundancy contained in the rules extracted from a dataset. The focus of this work is the study of redundancy elimination in implications with positive and negative attributes. We must recall that several authors have worked to remove redundancy in association rules in data mining. Zaki [14] used the notion of closed itemsets to reduce the set of rules, and also stated "the number of redundant rules is exponential in the length of the longest frequent itemsets". Other works study the relaxation of the notion of closure and closed itemsets [15] to describe a compact set of association rules which is approximately informative in the sense that the support and confidence of the remaining association rules can be derived from this compact set with high accuracy. Very recent works [16,17] prove that this is still an open problem.

When dealing with implications, our team has worked on this problem, providing the axiomatic system of simplification logic [18], on which it is possible to develop automated methods to remove redundancy in formal concept analysis which constitute the core to obtain bases of implications [19]. However, all these works consider the positive information in the dataset. To the best of our knowledge, no methods have been developed to eliminate redundancy in the rules when considering positive and negative attributes within the rules.

In this paper, we develop an automated logic-based method to remove redundancy in a set of mixed implications, that is, implications relating positive and negative attributes. For this, we propose the notion of a *simplified mixed implicational system* and an axiomatic system equivalent to that given in [12], but with rules better suited for implementation. The main idea is a set of logical equivalences oriented to detect redundancy and contradictions between positive and negative attributes, and therefore to simplify the set of implications, by removing attributes inside a single implication or even removing the whole implication.

The rest of this work is structured as follows: in Section 2, we present some preliminary notions about FCA and implicational systems with positive and mixed attributes. In Section 3, the idea of a simplified mixed implicational system is presented together with its motivation. In Section 4, we present logical equivalence rules especially suited for implications with mixed attributes, with the purpose of simplifying the set of implications. The algorithm to build simplified systems of implications is presented in Section 5, and a thorough experimental evaluation of the simplification achieved by the proposed algorithm is given in Section 6, as well as a discussion of the obtained results. The conclusions and future research lines are presented in Section 7. For the sake of readability, the proofs of the technical results of this work are collected in Appendix A.

2. Preliminary Notions and Results

Formal concept analysis (FCA) [20,21] is a mathematical theory based on lattice theory which analyses the information given in a *formal context*, i.e., a relationship between a set of objects and a set of attributes stored in a table. Usually, a formal context is given by a triple

$\mathbb{K} = (G, M, I)$, where G is the set with all the objects, M is the set with all the attributes and I is called the incidence and is defined by $I\colon G \times M \to \{0, 1\}$ and $I(g, m) = 1$ if the object g has the attribute m and $I(g, m) = 0$ otherwise.

FCA extracts the information stored in a formal context by using the derivation operators, namely, the pair of mappings $(^{\uparrow}, ^{\downarrow})$ defined as follows: given $X \subseteq G$, we have that $X^{\uparrow} = \{m \in M \mid I(g, m) = 1 \text{ for all } g \in X\}$, that is, the attributes shared by all the objects in X, and, given $Y \subseteq M$, we have $Y^{\downarrow} = \{g \in G \mid I(g, m) = 1 \text{ for all } m \in Y\}$, that is, all the objects that share all the attributes in Y. Using the derivation operators, a *formal concept* is defined as a pair (X, Y) satisfying $X^{\uparrow} = Y$ and $X = Y^{\downarrow}$; the set of formal concepts can be ordered by inclusion in the first component, and this ordering provides the structure of a complete lattice.

Another type of information provided by a formal context is given by the so-called *attribute implications*, which are any expression of the form $A \to B$ where A and B are sets of attributes, i.e., $A, B \subseteq M$ for a given a formal context $\mathbb{K} = (G, M, I)$. We are interested in implications that hold in the formal context $\mathbb{K}$: an implication $A \to B$ holds in $\mathbb{K}$ if $A^{\downarrow} \subseteq B^{\downarrow}$, i.e., if all the objects that have all the attributes from A satisfy the attributes from B as well. We denote by $\mathcal{L}_M$ the set of all the implications in M.

Simplification logic ($\mathcal{S}$) [18] was introduced as a means to remove redundancies in a given set of implications that hold in a formal context. The axiom system for $\mathcal{S}$ consists of the axiom schema `[Ref]` and the three inference rules `[Frag]`, `[Comp]`, `[Simp]` given below for all sets $A, B, C \subseteq M$:

`[Ref]` Reflexivity: $\vdash_{\mathcal{S}} A \to A$.

`[Frag]` Fragmentation: (We will follow the usual convention in this research area of omitting the symbol $\cup$ whenever necessary. For instance, in this rule BC means $B \cup C$.) $A \to BC \vdash_{\mathcal{S}} A \to B$.

`[Comp]` Composition: $A \to B, C \to D \vdash_{\mathcal{S}} AC \to BD$.

`[Simp]` Simplification: $A \to B, C \to D \vdash_{\mathcal{S}} A(C \setminus B) \to D$.

The notion of inference in $\mathcal{S}$ is defined as usual: let ϕ be an implication in $\mathcal{L}_M$ and let Σ be a set of implications in $\mathcal{L}_M$. We say that ϕ *is a syntactic consequence of* Σ *in* $\mathcal{S}$ (and we denote it by $\Sigma \vdash_{\mathcal{S}} \phi$, if there exists a sequence of implications $\phi_1, \ldots, \phi_n$ such that $\phi_n = \phi$) and, for all ϕ_i with $1 \leq i \leq n$ we have either $\phi_i \in \Sigma$ or we can obtain ϕ_i by applying one of the rules of $\mathcal{S}$ to the implications in the set $\{\phi_j \mid j < i\}$.

We recall below some derived rules from simplification logic, which we will use in this paper. The proof is straightforward using the definition of derivation. Given a set of attributes M, the following inference rules hold for all $A, B, C, D \in M$:

`[GenRef]` Generalized reflexivity: $\vdash_{\mathcal{S}} A \to C$ if $C \subseteq A$.

`[Augm]` Augmentation: $A \to B \vdash_{\mathcal{S}} AC \to BC$.

It is worth mentioning that some rules of simplification logic are in fact logical equivalences. The main equivalence rules that we have in the $\mathcal{S}$ are the following:

`[FragEq]` $\{A \to B\} \equiv \{A \to B \setminus A\}$.

`[UnEq]` $\{A \to B,\ A \to C\} \equiv \{A \to BC\}$.

`[GenEq]` $\{A \to B,\ C \to D\} \equiv \{A \to BD\}$ when $A \subseteq C \subseteq AB$.

`[∅-Eq]` $\{A \to \varnothing\} \equiv \varnothing$.

`[SimpEq]` $\{A \to B,\ C \to D\} \equiv \{A \to B,\ C \setminus B \to D \setminus B\}$ when $A \subseteq C \setminus B$.

Thus far, we have used the information explicitly given by the formal context, but have said nothing about pairs satisfying $I(g, m) = 0$. The point is that, in principle, $I(g, m) = 0$ does not mean that object g does not have the attribute m; it means that we do not have any evidence either in favour or to the contrary. If $I(g, m) = 0$, then we affirm that the

object g does not have the attribute m. Some authors [11,12,22,23] assume that $I(g,m) = 0$ means that the object g does not have the attribute m. To the best of our knowledge, the paper [11] was the first of these approaches which, given $\mathbb{K} = (G, M, I)$, built a new formal context $(\mathbb{K} \mid \overline{\mathbb{K}}) = (G, M \cup \overline{M}, I^*)$ where $I^*(g,m) = I(g,m)$ for all $m \in M$ and $I^*(g,\overline{m}) = \min(1, 1 - I(g,m))$; that is, $I^*(g,\overline{m}) = 0$ if $I(g,m) = 1$ and $I^*(g,\overline{m}) = 1$ otherwise. In this approach, the attributes in M are called *positive* and those in $\overline{M}$ are called *negative*. Note that, with this view, we duplicate the number of columns, and, as a consequence, the method is not the most efficient possible.

In this work, we adopt the approach considered in [12] in which $I(g,m) = 0$ means that the object g does not have the attribute m and, instead of duplicating the number of columns, the derivation operators are changed without changing the formal context.

The extended operators considered here are $^{\Uparrow} : 2^G \to 2^{M \cup \overline{M}}$ and $^{\Downarrow} : 2^{M \cup \overline{M}} \to 2^G$, and are defined as follows:

$$X^{\Uparrow} = \{m \in M \mid (g,m) \in I \ \ \forall\, g \in X\} \cup \{\overline{m} \in \overline{M} \mid (g,m) \notin I \ \ \forall\, g \in X\}$$

$$Y^{\Downarrow} = \{g \in G \mid (g,m) \in I \ \ \forall\, m \in Y\} \cap \{g \in G \mid (g,m) \notin I \ \ \forall\, \overline{m} \in Y\}$$

When considering both positive and negative attributes, we will say that we are using a *mixed context*; when working with $\mathbb{K}$ (resp. $\overline{\mathbb{K}}$) with the classical view, we say that we are working with a *positive* (resp. *negative*) context.

As in the classical case, we can define when an implication holds in a mixed context in terms of $\Uparrow$ and $\Downarrow$ as follows: $A \to B$ *is valid in a mixed context* (denoted by $\mathbb{K} \models A \to B$) if and only if $A^{\Downarrow} \subseteq B^{\Downarrow}$.

The axiom system of simplification logic was extended to this new approach with mixed attributes in [12,24].

The new axiomatic system contains one axiom schema `[Ref]` and four inference rules, `[Simp]`, `[Key]`, `[InKey]` and `[Red]`:

`[Ref]` Reflexivity: $\vdash_{\mathcal{S}} A \to A$.

`[Simp]` Simplification: $A \to B, C \to D \vdash_{\mathcal{S}} A(C \smallsetminus B) \to D$.

`[Key]` Key: (Following the convention, hereonwards b represents the singleton $\{b\}$, and $A\overline{b}$ means $A \cup \{\overline{b}\}$). $A \to b \vdash_{\mathcal{S}} A\overline{b} \to M\overline{M}$.

`[InKey]` Inverse key: $Ab \to M\overline{M} \vdash_{\mathcal{S}} A \to \overline{b}$.

`[Red]` Reduction: $Ab \to C, A\overline{b} \to C \vdash_{\mathcal{S}} A \to C$.

This new system is a proper extension of that of simplification logic in that rules `[Frag]` and `[Comp]` can be derived from the new axioms [12]. Our set of derived rules can be further extended by a version of the *ex contradictione quodlibet* and the *contraposition rule*:

`[Cont]` Contradiction: $\vdash a\overline{a} \to M\overline{M}$.

`[Rft]` Reflection: $Aa \to b \vdash A\overline{b} \to \overline{a}$.

Notice that `[Key]` is in fact the converse of `[InKey]`, and they provide an equivalence between the implications $Ab \to M\overline{M}$ and $A \to \overline{b}$, which reflects the fact that whenever the set $A \cup \{b\}$ is inconsistent, then $A \to \overline{b}$ should hold. Moreover, a version of the well-known *cut rule* arises as the equivalence between $A \to C$ and the set $\{Ab \to C, A\overline{b} \to C\}$.

3. Simplified Mixed Implicational Systems

In order to formally define the notion of a*simplified mixed implicational system*, we will use the following notation:

Notation 1. *Hereafter, we will use* $\overline{X} := \{\overline{x} : x \in X\}$ *and notice that* $\overline{\overline{x}} = x$ *for any* $x \in M\overline{M}$. *Thus,* $x \in \overline{X}$ *if, and only if,* $\overline{x} \in X$. *For any* $A, B \subseteq M\overline{M}$, *we have* $\overline{AB} = \overline{A}\,\overline{B}$ *and also* $\overline{A \cap B} = \overline{A} \cap \overline{B}$.

Definition 1. *Let M be a set of attributes and Σ be an implicational system with attributes in $M\overline{M}$. Σ is said to be a* simplified mixed implicational system *(or* sm-implicational system *in short) if the following conditions hold for all $A \to B, C \to D \in \Sigma$:*

(i) $B \neq \varnothing$ *and* $A \cap B = \varnothing$;
(ii) $A = C$ *implies* $B = D$;
(iii) $A \subsetneq C$ *implies* $C \cap B = \varnothing = D \cap B$;
(iv) $B \neq M\overline{M}$ *and* $A \cap \overline{A} = \varnothing$;
(v) *If* $x \in A \cap \overline{C}$ *and* $A \smallsetminus x = C \smallsetminus \overline{x}$, *then* $D \not\subseteq B$.

The first three properties are inherited from the definition of a simplified implicational system by [19] and express the idea that the size of the problem cannot be reduced by applying the equivalence rules for positive attributes. We comment below on the ideas underlying conditions *(iv)* and *(v)*, which are specific to the case of mixed attributes.

For *(iv)*, the condition $B \neq M\overline{M}$ refers to the fact that, on the one hand, $\varnothing \to M\overline{M}$ is not admissible and, on the other hand, if $A \neq \varnothing$, we would have that $A \to M\overline{M}$ is equivalent to $A \smallsetminus x \to \overline{x}$ for any $x \in A$ (hence, the system would be reduced by considering the latter). Furthermore, by the derived rule `[Cont]`, any implication of the form $A \to B$ with a contradiction in the antecedent ($A \cap \overline{A} \neq \varnothing$) is valid; therefore, it can be safely removed from the implicational system.

Finally, *(v)* expresses syntactically a situation in which rule `[Red]` would apply. If $A \smallsetminus x = C \smallsetminus \overline{x}$, there exists a set of attributes S such that $A = Sx$ and $C = S\overline{x}$. In addition, if $D \subseteq B$, we would have

$$\begin{aligned} \{A \to B, C \to D\} &= \{Sx \to B, S\overline{x} \to D\} \\ &\equiv \{Sx \to B \smallsetminus D, Sx \to D, S\overline{x} \to D\} && \texttt{[FragEq]} \\ &\equiv \{Sx \to B \smallsetminus D, S \to D\} && \texttt{[Red]}; \end{aligned}$$

hence, the size of the system can be reduced.

The next theorem provides a set of equivalence rules oriented to obtaining an automated method to build an *sm-implicational system* from a given one. For the sake of readability, the proofs of all the technical results are collected in Appendix A.

Theorem 1. *Let $\mathbb{K} = (G, M, I)$ be a formal context and $A, B, C, D \subseteq M\overline{M}$. Then, the following equivalences hold:*

`[KeyEq]` *If $A \subseteq C$ and $B \cap \overline{C} \neq \varnothing$, then*

$$\{A \to B, C \to D\} \equiv \{A \to B\}.$$

`[InKeyEq]` *If there exists $x \in A \smallsetminus C$, such that $A \smallsetminus x \subseteq C$, then*

$$\{A \to M\overline{M}, C \to D\} \equiv \{A \to M\overline{M}, C \to D\overline{x}\}.$$

`[RedEq]` *If $D \subseteq B$ and there exists $x \in A \cap \overline{C}$, with $A \smallsetminus x \subseteq C \smallsetminus \overline{x}$, then*

$$\{A \to B, C \to D\} \equiv \{A \to B, C \smallsetminus \overline{x} \to D\}.$$

By using the previous theorem, we propose the following inference rules, which are more convenient to be implemented algorithmically.

`[Ref]` Reflexivity: $\vdash A \to A$.

`[Simp]` Simplification: $A \to B, C \to D \vdash A(C \smallsetminus B) \to D$.

`[Key']` Key: $A \to B, C \to D \vdash C \to M\overline{M}$ if $A \subseteq C$ and $B \cap \overline{C} \neq \varnothing$.

[InKey′] Inverse key: $A \to M\overline{M}, C \to D \vdash C \to D\overline{x}$ if there exists $x \in A \setminus C$ and $A \setminus x \subseteq C$.

[Red′] Reduction: $A \to B, C \to D \vdash C \setminus \overline{x} \to D$ if $D \subseteq B$ and there exists $x \in A \cap \overline{C}$ with $A \setminus x \subseteq C \setminus \overline{x}$.

Of course, this new axiomatic system is equivalent to the initial one, as shown in the theorem below.

Theorem 2. *The system formed by rules* [Ref], [Simp], [Key′], [InKey′] *and* [Red′] *is equivalent to that formed by* [Ref], [Simp], [Key], [InKey] *and* [Red].

4. Simplification via Equivalence Rules

In this section, we introduce some other equivalence rules that will be useful for removing redundant attributes. The starting point will use the following result:

Lemma 1. *For all $A, B, C, D \subseteq M\overline{M}$, we have:*

[ContEq] *If $A \cap \overline{A} \neq \varnothing$, then $\{A \to B\} \equiv \varnothing$.*

[ContEq′] *If $A \neq \varnothing$ and $AB \cap \overline{B} \neq \varnothing$, then $\{A \to B\} \equiv \{A \to M\overline{M}\} \equiv \{A \setminus x \to \overline{x}\}$ for any $x \in A$.*

[ContEq″] *If $C \neq \varnothing$, $A \subseteq CD$ and $B \cap \overline{CD} \neq \varnothing$, then, for any $x \in C$*

$$\{A \to B, C \to D\} \equiv \{A \to B, C \to M\overline{M}\} \equiv \{A \to B, C \setminus x \to \overline{x}\}.$$

The equivalences above allow us to detect contradictions and, hence, reduce the size of the set of implications. Below, we propose some other equivalence rules that take into account the possible relationship between different implications in order to reduce their size.

Theorem 3. *Consider $A, B, C, D \subseteq M\overline{M}$:*

[KeyEq′] *If there exist $x \in A \cap D$, $y \in B \cap \overline{C}$ with $A \setminus x = C \setminus \overline{y}$, then*

$$\{A \to B, C \to D\} \equiv \{A \to B \setminus y, C \setminus \overline{y} \to y\} \equiv \{A \to B \setminus y, C \to M\overline{M}\}.$$

[KeyEq″] *If $A \subseteq C \neq \varnothing$ and $B \cap \overline{D} \neq \varnothing$, for any $x \in C$ we have that then*

$$\{A \to B, C \to D\} \equiv \{A \to B, C \setminus x \to \overline{x}\}.$$

[RedEq′] *If $D \subseteq B$ and there exists $x \in A \cap \overline{C}$ such that $A \setminus x = C \setminus \overline{x}$, then*

$$\{A \to B, C \to D\} \equiv \{A \to B \setminus D, C \setminus \overline{x} \to D\}.$$

[RftEq] *If there exist $x \in A$, $y \in B \cap \overline{C}$ and $A \setminus x = C \setminus \overline{y}$, then*

$$\{A \to B, C \to D\} \equiv \{A \to B \setminus y, C \to D\overline{x}\}.$$

[RftEq′] *If there exist $x \in A \cap \overline{D}$, $y \in B \cap \overline{C}$ and $A \setminus x \subseteq C \setminus \overline{y}$, then*

$$\{A \to B, C \to D\} \equiv \{A \to B, C \to D \setminus \overline{x}\}.$$

[MixUnEq] *If there exist $x \in A$, $y \in C$ such that $A \setminus x = C \setminus y$ and $b \in D$, then*

$$\{A \to b, C \to D\} \equiv \{(A \setminus x)\overline{b} \to \overline{xy}, C \to D \setminus b\}.$$

Among all the rules above, there are two which, in principle, do not help to reduce the size of the system(`[RftEq]` and `[MixUnEq]`), but guarantee to keep the number of attributes fixed. However, there is a particular case where the size is reduced as a result of removing implications: the application of either `[RftEq]` or `[MixUnUnitEq]` when one of the implications is a unit implication (its right-hand side has only one element) allows to remove that implication. The result is formally stated in the following corollary:

Corollary 1. *For all $A, C, D \subseteq M\overline{M}$ and $b, y \in M\overline{M}$, the following equivalences hold:*

`[RftUnitEq]` *If $y \in \overline{C}$ and there exist $x \in A$ such that $A \smallsetminus x = C \smallsetminus \overline{y}$, then*

$$\{A \to y, C \to D\} \equiv \{C \to D\overline{x}\}.$$

`[MixUnUnitEq]` *If there exist $x \in A$, $y \in C$ such that $A \smallsetminus x = C \smallsetminus y$, then*

$$\{A \to b, C \to b\} \equiv \{(A \smallsetminus x)\overline{b} \to \overline{xy}\}.$$

5. Automatic Computation of sm-Implicational Systems

In this section, we propose an algorithm to simplify a system of implications with positive and negative attributes. The algorithm is based on the equivalence rules obtained in Section 3 and Section 4. For the sake of readability, the main algorithm is decomposed into subroutines that check the conditions required to apply certain equivalence rules.

Following the same strategy as in Section 4, we start by defining the algorithms to simplify the implications that contain contradictions. In Algorithm 1, we translate `[ContEq']` into pseudo-code, whereas in Algorithm 2 we do the same with `[ContEq'']`.

Algorithm 1: `Simplify-Cont`$(A \to B)$

```
if A ≠ ∅ and B ∩ A̅B̅ ≠ ∅ then                    // Conditions for [ContEq']
    Select x ∈ A
    A := A ∖ x;   B := x̅
return A → B
```

In Algorithm 2 below, we work with two implications, $A \to B$ and $C \to D$, and check the conditions of `[ContEq'']` in both orders by changing the roles of $A \to B$ and $C \to D$ in the equivalence rule, so we can simplify both implications in a single execution of `Simplify-Cont2`. We will make use of this strategy in the rest of the algorithms.

Algorithm 2: `Simplify-Cont2`$(A \to B, C \to D)$

```
if A ≠ ∅ and C ⊆ AB and D ∩ A̅B̅ ≠ ∅ then     // Conditions for [ContEq'']
    Select x ∈ A
    A := A ∖ x;   B := x̅
else if C ≠ ∅ and A ⊆ CD and B ∩ C̅D̅ ≠ ∅ then     // Converse conditions
    Select x ∈ C
    C := C ∖ x;   D := x̅
return (A → B, C → D)
```

The equivalence rules `[KeyEq]` and `[KeyEq']` are presented in Algorithm 3. Observe that the conditions of those equivalence rules are nested since they had some common requirements so the algorithm can be written more compactly. Additionally, notice that when an implication $X \to Y$ can be removed from the implicational system due to the use of an equivalence rule, in the algorithms, it is indicated as $X, Y := \varnothing$.

Algorithm 3: `Simplify-Key`$(A \to B, C \to D)$

```
if B ∩ C̄ ≠ ∅ then
    if A ⊆ C then C, D := ∅                                  // Conditions for [KeyEq]
    else if A ∩ D ≠ ∅ then                                   // [KeyEq′] branch
        foreach x ∈ A ∩ D, y ∈ B ∩ C̄ do
            if A ∖ x = C ∖ ȳ then
                B := B ∖ y;   C := C ∖ ȳ;   D := y
if A ∩ D̄ ≠ ∅ then
    if C ⊆ A then A, B := ∅                                  // Converse of [KeyEq]
    else if B ∩ C ≠ ∅ then                                   // Converse of [KeyEq′]
        foreach x ∈ A ∩ D̄, y ∈ B ∩ C do
            if A ∖ x = C ∖ y then
                D := D ∖ x̄;   A := A ∖ x;   B := x̄
return (A → B, C → D)
```

Algorithm 4 presents the pseudocode for `[KeyEq″]`. It has not been incorporated into the same algorithm as `[KeyEq]` and `[KeyEq′]` for the sake of readability. The equivalence rules derived from the inference rule `[Red]` are condensed in Algorithm 5. Then, in Algorithm 6, we present the equivalence rules `[RftEq′]` and `[RftUnitEq]`.

Algorithm 4: `Simplify-Key2`$(A \to B, C \to D)$

```
if B ∩ D̄ ≠ ∅ then
    if C ≠ ∅ and A ⊆ C then                                  // Conditions for [KeyEq″]
        Select x ∈ C
        C := C ∖ x;   D := x̄
    else if A ≠ ∅ and C ⊆ A then                             // Converse for [KeyEq″]
        Select x ∈ A
        A := A ∖ x;   B := x̄
return (A → B, C → D)
```

Algorithm 5: `Simplify-Red`$(A \to B, C \to D)$

```
if A ∩ C̄ ≠ ∅ then
    if D ⊆ B then                                            // Conditions for [Red]
        foreach x ∈ A ∩ C̄ do
            if A ∖ x ⊆ C ∖ x̄ then
                C := C ∖ x̄
                if A ∖ x = C then B := B ∖ D                 // Condition for [RedEq′]
    else if B ⊆ D then                                       // Converse conditions
        foreach x ∈ A ∩ C̄ do
            if C ∖ x̄ ⊆ A ∖ x then
                A := A ∖ x
                if A = C ∖ x̄ then D := D ∖ B
return (A → B, C → D)
```

Algorithm 6: `Simplify-Rft`$(A \to B, C \to D)$

```
1  if B ∩ C̄ ≠ ∅ then
2      if |B| = 1 and A ≠ ∅ then                          // [RftUnitEq] branch
3          Let y be the only element in B ∩ C̄
4          foreach x ∈ A do
5              if A ∖ x = C ∖ ȳ then
6                  D := Dx̄;   A, B := ∅
7      else if A ∩ D̄ ≠ ∅ then                             // [RftEq'] branch
8          foreach x ∈ A ∩ D̄, y ∈ B ∩ C̄ do
9              if A ∖ x ⊆ C ∖ ȳ then                      // Direct
10                 D := D ∖ x̄
11             else if C ∖ ȳ ⊆ A ∖ x then                 // Converse
12                 B := B ∖ y
13 else if |D| = 1 and C ≠ ∅ and A ∩ D̄ ≠ ∅ then   // Converse [RftUnitEq]
14     Let y be the only element in A ∩ D̄
15     foreach x ∈ C do
16         if C ∖ x = A ∖ y then
17             B := Bx̄;   C, D := ∅
18 return (A → B, C → D)
```

Algorithm 7 presents the code for the equivalence rule `[MixUnUnitEq]`. It is a simple translation of the conditions into pseudocode. Note that, in this case, there is no need to include the converse conditions due to the equality required.

Algorithm 7: `Simplify-MixUnUnit`$(A \to B, C \to D)$

```
1  if |B| = 1 and B = D and A, C ≠ ∅ then
2      foreach x ∈ A do
3          foreach y ∈ C do
4              if A ∖ x = C ∖ y then
5                  Let b be the only element in B
6                  A := (A ∖ x)b̄;   B := xȳ;   C, D := ∅
7  return (A → B, C → D)
```

The complete method is presented in Algorithm 8, which incorporates the other algorithms in order to simplify the implications with all the studied equivalence rules. We can explain its procedure as follows: firstly, the set Σ' of proper implications (i.e., the antecedent and consequent are disjoint) without contradictions is built from Σ. We initialize $\Sigma_s := \Sigma'$ and reset Σ'. We will use Σ' as a list to store the simplified implications in each iteration.

For each implication $A \to B \in \Sigma_s$, we try to simplify it with the other implications already stored in Σ'. If the result after simplification is a proper implication (nonempty consequent), it is added to Σ'. By this procedure, all implications are compared with each other, and both $A \to B \in \Sigma_s$ and all $C \to D \in \Sigma'$ are simplified in a single step. The algorithm checks whether the implications have any contradiction; in that case, rules `[ContEq]` and `[ContEq']` are used (lines 5 and 8). Lines 9–10 express the conditions to use `[GenEq]`, that is, where $A \to B$ is more general than $C \to D$ or vice versa, keeping only the more general. If it is not the case, then the algorithm proceeds to check the conditions of the simplification rules described in Sections 3 and 4. The first simplification to be considered is `[SimpEq]` (lines 12–15), since it allows us to remove attributes from both the left-hand and right-hand sides of the implications, whenever applicable. Later, the algorithm checks all the simplifications that are specific for mixed attributes and that have already been described in Algorithms 2 to 7. Line 22 is used to add the implications $C \to D$ that have

not been removed after all the simplification steps. Lines 23–26 add the implication $A \to B$ (if it has not been removed) to Σ'.

Algorithm 8: `Simplify-Mixed`(Σ)

Input: Σ: a set of implications.
Output: Σ_s: an sm-implicational system that is equivalent to Σ.
`// Consider only proper implications (`$A \cap B = \varnothing$`) without contradictions`

1 $\Sigma' := \{A \to B \setminus A | A \to B \in \Sigma, B \not\subseteq A, A \cap \overline{A} = \varnothing\}$
2 **repeat**
3 $\quad \Sigma_s := \Sigma'$; $\Sigma' := \varnothing$
4 $\quad$ **foreach** $A \to B \in \Sigma_s$ **do**
5 $\quad\quad A \to B :=$ `Simplify-Cont`$(A \to B)$
6 $\quad\quad \Gamma := \varnothing$
7 $\quad\quad$ **foreach** $C \to D \in \Sigma'$ **do**
8 $\quad\quad\quad C \to D :=$ `Simplify-Cont`$(C \to D)$
9 $\quad\quad\quad$ **if** $C \subseteq A \subseteq CD$ **or** $A \subseteq C \subseteq AB$ **then** `// [GenEq]`
10 $\quad\quad\quad\quad A := A \cap C$; $B := BD$
11 $\quad\quad\quad$ **else** `// Apply simplifications sequentially`
$\quad\quad\quad\quad$ `// First, general simplifications`
12 $\quad\quad\quad\quad$ **if** $A \subsetneq C$ **and** $D \not\subseteq B$ **then** `// [SimpEq]`
13 $\quad\quad\quad\quad\quad C := C \setminus B$; $D := D \setminus B$
14 $\quad\quad\quad\quad$ **else if** $C \subsetneq A$ **then**
15 $\quad\quad\quad\quad\quad A := A \setminus D$; $B := B \setminus D$
$\quad\quad\quad\quad$ `// Simplifications specific for mixed attributes`
16 $\quad\quad\quad\quad (A \to B, C \to D) :=$ `Simplify-Red`$(A \to B, C \to D)$
17 $\quad\quad\quad\quad (A \to B, C \to D) :=$ `Simplify-Key`$(A \to B, C \to D)$
18 $\quad\quad\quad\quad (A \to B, C \to D) :=$ `Simplify-Key2`$(A \to B, C \to D)$
19 $\quad\quad\quad\quad (A \to B, C \to D) :=$ `Simplify-Rft`$(A \to B, C \to D)$
20 $\quad\quad\quad\quad (A \to B, C \to D) :=$ `Simplify-Cont2`$(A \to B, C \to D)$
21 $\quad\quad\quad\quad (A \to B, C \to D) :=$ `Simplify-MixUnUnit`$(A \to B, C \to D)$
22 $\quad\quad\quad\quad$ **if** $D \neq \varnothing$ **then** $\Gamma := \Gamma \cup \{C \to D\}$
23 $\quad\quad$ **if** $B = \varnothing$ **then** $\Sigma' := \Gamma$
24 $\quad\quad$ **else**
25 $\quad\quad\quad A \to B :=$ `Simplify-Cont`$(A \to B)$
26 $\quad\quad\quad \Sigma' := \Gamma \cup \{A \to B \setminus A\}$
27 **until** $\Sigma_s = \Sigma'$
28 **return** Σ_s

The theorem below proves the termination and correctness of Algorithm 8 in the sense that the ouput is an implicational system equivalent to the input and, in addition, is an sm-implicational system. The notion of size and the following notations will be useful for the proofs of the results hereafter.

The *size* of an implicational system Σ is defined as $\|\Sigma\| = \sum_{A \to B \in \Sigma}(|A| + |B|)$, where $|X|$ represents the cardinality of the set X. Thus, $\|\Sigma\|$ is the amount of attributes that appear in the antecedents and consequents in the implications in Σ.

Theorem 4. *The function* `Simplify-Mixed` *given in Algorithm 8 reaches a fixed point* Σ' *given any finite implicational system* Σ. *In addition,* $\Sigma' \equiv \Sigma$ *and* Σ' *is an sm-implicational system.*

Proof. First, note that, after the comparison of a pair of implications (lines 7–22), the size of $\{A \to B, C \to D\}$ does not increase, since there are only three possibilities: (1) no simplification is made; (2) one or more attributes are removed from the left-hand side or the

right-hand side of any of the two implications; and (3) one implication is removed at the cost of adding, at most, one attribute to the other implication (this happens in `[RftUnitEq]` and `[MixUnUnitEq]`). Since the minimum size of a proper implication is one, in all these three cases, we can guarantee that the size of the system does not increase.

Moreover, if no simplification is made in an iteration over all implications $A \to B \in \Sigma_s$, i.e., the system is not modified in that iteration, then the algorithm stops. Thus, in an iteration of the repeat loop in lines 2–27, the only two possibilities are (1) the system is reduced by removing implications or by removing attributes, and (2) the system is not modified in the iteration. There cannot be infinitely many iterations since the size of the system is finite and decreases in each iteration. Hence, the algorithm will stop after finitely many steps.

Let Σ be an implicational system and let Σ' be the output of Algorithm 8 on Σ. Since Σ' is built iteratively by applying equivalence rules, it is obvious that $\Sigma' \equiv \Sigma$. Let us now prove that Σ' is an sm-implicational system. Assume $A \to B, C \to D \in \Sigma'$:

(i) $B \neq \varnothing$ and $A \cap B = \varnothing$ follow from the construction of Σ' in lines 1 and 23–26;

(ii) Let us prove that in Σ' there cannot be two *different* implications $A \to B$ and $C \to D$ with $A = C$. At a given iteration of the algorithm, if two implications verify that their left-hand sides are equal, they would meet the conditions in line 9, so they will be *merged* into a single implication, and, therefore, the algorithm would not reach a fixed point at that iteration. Hence, in the fixed point Σ', there cannot be such duplicity since the implications would have been previously merged;

(iii) $A \subsetneq C$ implies $C \cap B = \varnothing = D \cap B$: it is a consequence of the application of `[SimpEq]` in lines 12–15;

(iv) $B \neq M\overline{M}$ and $A \cap \overline{A} = \varnothing$ due to rules `[ContEq]` and `[ContEq']`;

(v) If $x \in A \cap \overline{C}$ and $A \smallsetminus x = C \smallsetminus \overline{x}$, then $D \not\subseteq B$. Otherwise, `[RedEq]` could be applied, and therefore, Σ' would not be the fixed point.

Hence, Σ' is an sm-implicational system. □

Note that the only equivalence rules for mixed attributes needed to obtain an sm-implicational system are `[ContEq]`, `[ContEq']` and `[RedEq]`. The purpose of the rest of equivalence rules is to further minimize and get an even simpler implicational system.

Let us now show a brief example of the application of Algorithm 8, omitting minor details for an easier reading.

Example 1. *Given* $\Sigma = \{a\overline{d}b \to c, \overline{b}\overline{c} \to ad, ac\overline{b} \to d, a\overline{c} \to bd\}$, *the application of* `Simplify-Mixed`(Σ) *is as follows:*

- *Firstly, the pair* $\{a\overline{d}b \to c, \overline{b}\overline{c} \to ad\}$ *is studied. Using* `[ContEq'']`, *we simplify* $\overline{b}\overline{c} \to ad$ *into* $\overline{c} \to b$. *Thus,*

$$\{a\overline{d}b \to c, \overline{b}\overline{c} \to ad, ac\overline{b} \to d, a\overline{c} \to bd\} \equiv \{a\overline{d}b \to c, \overline{c} \to b, ac\overline{b} \to d, a\overline{c} \to bd\};$$

- *Later,* $\{a\overline{d}b \to c, a\overline{c} \to bd\}$ *satisfies the conditions of* `[RftUnitEq]`, *so* $a\overline{d}b \to c$ *is removed.*

$$\{a\overline{d}b \to c, \overline{c} \to b, ac\overline{b} \to d, a\overline{c} \to bd\} \equiv \{\overline{c} \to b, ac\overline{b} \to d, a\overline{c} \to bd\};$$

- *For* $\{\overline{c} \to b, a\overline{c} \to bd\}$, `[SimpEq]` *can be applied, changing* $a\overline{c} \to bd$ *into* $a\overline{c} \to d$:

$$\{\overline{c} \to b, ac\overline{b} \to d, a\overline{c} \to bd\} \equiv \{\overline{c} \to b, ac\overline{b} \to d, a\overline{c} \to d\};$$

- *When comparing* $\{ac\overline{b} \to d, a\overline{c} \to d\}$, `[RedEq]` *can be applied, transforming* $ac\overline{b} \to d$ *into* $a\overline{b} \to d$:

$$\{\overline{c} \to b, ac\overline{b} \to d, a\overline{c} \to d\} \equiv \{\overline{c} \to b, a\overline{b} \to d, a\overline{c} \to d\};$$

- *Now, for* $\{a\overline{b} \to d, a\overline{c} \to d\}$, `[MixUnUnitEq]` *can also be applied, changing* $a\overline{c} \to d$ *to* $a\overline{d} \to bc$ *and removing* $a\overline{b} \to d$*:*

$$\{\overline{c} \to b, a\overline{b} \to d, a\overline{c} \to d\} \equiv \{\overline{c} \to b, a\overline{d} \to bc\};$$

- *At this point, the algorithm has reached a fixed point and returns* $\Sigma' = \{\overline{c} \to b, a\overline{d} \to bc\}$.

We can check that the size of Σ *is 16, and the size of the sm-implicational system obtained by Algorithm 8 is 6; that is, we have obtained a size reduction of* $\approx$*62.5%.*

This example shows that the algorithm can be highly effective in reducing the size of a system with mixed attributes. In the next section, we will present a thorough experimental evaluation of the size reduction achieved. We conclude this section by presenting a worst-case complexity analysis of the algorithm, showing that it has polynomial complexity.

Theorem 5. *Let* Σ *be an implicational system with mixed attributes. The worst-case time complexity of Algorithm 8 on* Σ *is* $\mathcal{O}(|\Sigma|^2\|\Sigma\|)$.

Proof. In every iteration of the algorithm, we remove at least one attribute or one implication. Therefore, the maximum number of iterations of the **repeat** loop is $\|\Sigma\| = \max\{|\Sigma|, \|\Sigma\|\}$, since $\|\Sigma\| \geq |\Sigma|$, for we are considering only proper implications. Furthermore, in each iteration, all possible pairs of implications are studied in order to apply the equivalence rules, so a maximum of $\mathcal{O}(|\Sigma|^2)$ steps are made in every iteration. In aggregate, the maximum number of steps is $\mathcal{O}(|\Sigma|^2\|\Sigma\|)$. □

It is worth noting that, as $\|\Sigma\| \geq |\Sigma|$, the complexity of Algorithm 8 is polynomial, $\mathcal{O}(\|\Sigma\|^3)$, in the size of the input.

As a related issue, it is important to note that the complexity of the construction of the canonical basis of implications, as well as of the enumeration of the rules belonging to this basis, has been studied before by other authors [25–27]. However, such a problem is different from the one we study in the present work, where we consider a *previously computed* system of implications as the input for our method, which, with polynomial complexity, returns an equivalent simplified system. Moreover, the input system in our proposal does not have to represent a basis associated with a given formal context.

6. Experimental Results

In order to evaluate the capability of Algorithm 8 for simplifying an implicational system, a number of random mixed contexts have been generated with different numbers of attributes in M and different densities (proportion of non-zero elements in the table of the relation I), and their Duquenne–Guigues bases of implications [28] has been computed. As these bases are built without using the logic of mixed attributes, it makes sense to use them as a benchmark for the proposed algorithm.

Contexts were constructed for $|M| \in \{4, 5, \ldots, 10\}$ (that is, $|M\overline{M}| \in \{8, 10, \ldots, 20\}$) and density $\delta \in \{0.1, 0.25\}$. This choice of values for δ is due to two main reasons: on the one hand, (purely positive) formal contexts in real situations are very sparse, with a very low proportion of non-zero elements in the table of the relation I; on the other hand, the apposition of a context $\mathbb{K}$ and its associated *negative* context, $\overline{\mathbb{K}}$, by construction, always has density 0.5. Therefore, the choice of δ has no relevance to the complexity of the problem since, as we have seen in Theorem 5, the main indicators of the complexity of the problem are the size and the cardinality of the implication basis. Therefore, particular values of δ typically representative of low densities have been selected. For each combination of these parameters, 50 contexts have been randomly generated. All experiments have been performed using the R programming language and the library `fcaR` [29], particularly for the generation of the datasets as well as for the computation of the implication bases.

As stated in Section 5, in order to obtain an sm-implicational system, it is enough to consider `[ContEq]`, `[ContEq']`and `[RedEq]`. For this reason, in the experiment, we have

compared two different versions of the algorithm: (v1) where only the simplifications related to the positive attributes, `[GenEq]`and `[SimpEq]`, as well as the aforementioned `[ContEq]`, `[ContEq′]` and `[RedEq]`, are performed; (v2) the simplification algorithm as described in Algorithm 8. Thus, (v1) is obtained by removing lines 6 and 10 from Algorithm 5 and lines 17–21 from Algorithm 8. The idea behind this comparison is to check the improvement obtained by adding the rest of the equivalence rules.

Hence, for each experiment carried out, the size and the cardinal of the sm-implicational system returned by the simplification algorithm have been measured, both for version (v1) and (v2). The number of iterations performed by each of the two versions before reaching the fixed point was also taken into account, as well as the number of simplifications performed and the calculation time.

In Table 1, we present the averaged results of the execution of the two versions on the problems generated according to their number of attributes and density. As expected, whereas the reduction produced by the algorithm version (v1) to obtain sm-implicational systems ranges from 19% to 74% with respect to the system size, and 2% to 47% with respect tocardinality, version (v2) provides much more reduced outputs, with a range of reduction between 73% and 85% with respect size, and between 61% to 79% with respect to cardinality. Furthermore, it is worth mentioning that, despite the fact that (v2) performs a greater number of checks, the computation time is, in many occasions, smaller than that of (v1); the reason is that in (v2), it is possible to perform many more simplifications in each iteration. This shows that the extra cost of applying the extra equivalence rules pays off, since we obtain a notably smaller version of the implicational system with shorter computation times.

Table 1. Data from the experimental evaluation. Columns 3 to 10 express the average value obtained in the experiments for each of the studied parameters (cardinality, size and execution time).

				(v1)			(v2)		
$\|M\|$	δ	$\|\Sigma\|$	$\|\|\Sigma\|\|$	$\|\Sigma'\|$	$\|\|\Sigma'\|\|$	t	$\|\Sigma'\|$	$\|\|\Sigma'\|\|$	t
4	0.1	8.34	45.68	4.40	11.66	0.105	2.68	6.64	0.089
	0.25	11.48	55.92	7.52	21.44	0.261	2.96	8.94	0.121
5	0.1	11.50	75.92	6.68	21.86	0.186	3.92	12.14	0.167
	0.25	22.86	120.38	17.90	57.02	1.694	6.16	21.08	0.852
6	0.1	15.60	117.22	9.78	34.78	0.435	5.66	19.16	0.335
	0.25	38.48	220.18	32.50	111.58	6.299	10.78	40.94	2.771
7	0.1	20.40	171.04	13.64	51.88	0.956	7.36	26.24	0.563
	0.25	61.96	379.92	54.96	203.40	20.356	18.68	75.16	9.439
8	0.1	28.04	254.36	20.32	78.56	2.388	10.04	37.56	1.642
	0.25	102.84	652.44	94.92	368.56	67.209	33.68	146.44	34.22
9	0.1	35.80	351.10	27.20	108.80	3.319	12.90	51.20	2.596
	0.25	149.10	992.20	140.10	571.40	144.114	50.50	232.00	74.970
10	0.1	46.21	486.50	36.40	149.90	7.921	17.60	70.10	5.220
	0.25	218.92	1494.20	208.90	891.10	393.850	81.40	393.10	188.461

The study of the relationship between the size and cardinality of the input system and the size and cardinality of the simplified systems by versions (v1) and (v2) provides the following results: in Figure 1, we can see the practically linear relationship in the reduction of both factors. Again, we observe that (v2) presents a lower slope, corroborating the trend indicated in Table 1. A linear regression has been performed on the data, obtaining

(v1) $\|\Sigma'\| \approx 0.629\|\Sigma\|$ (with $R^2 = 0.956$), and $|\Sigma'| \approx 0.975|\Sigma|$ (with $R^2 = 0.9992$).
(v2) $\|\Sigma'\| \approx 0.249\|\Sigma\|$ (with $R^2 = 0.967$), and $|\Sigma'| \approx 0.331|\Sigma|$ (with $R^2 = 0.961$).

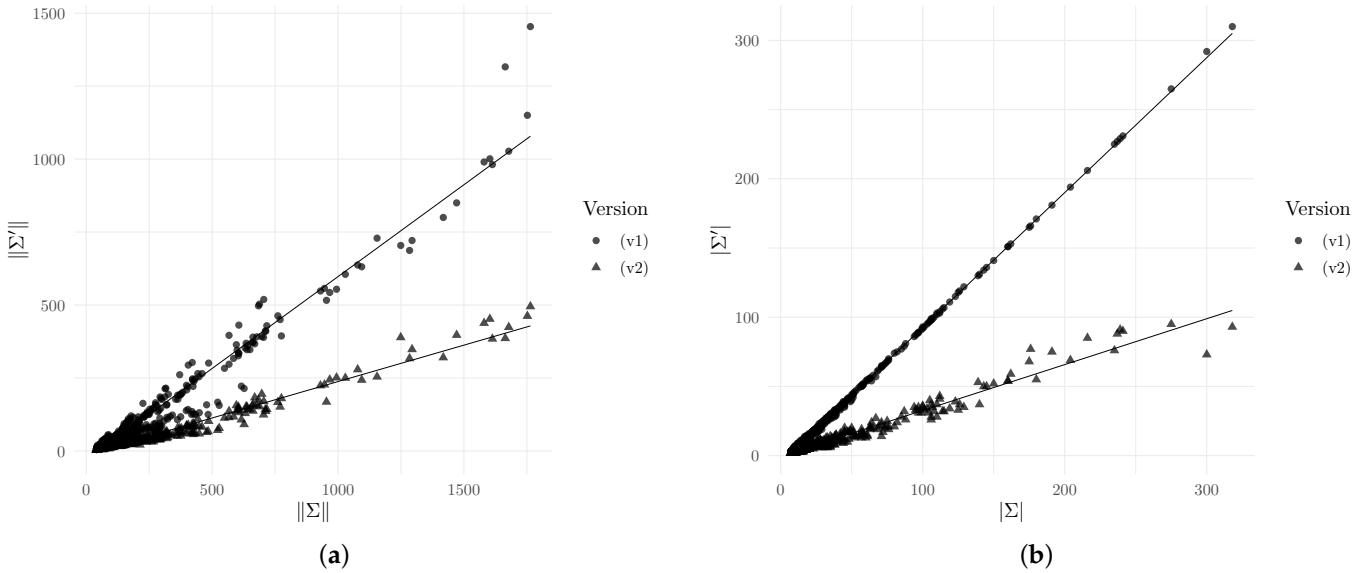

Figure 1. Comparison of versions (v1) and (v2) with respect to a reduction ratio of $\|\Sigma\|$ (**a**) and of $|\Sigma|$ (**b**) for all the datasets used in the experiments.

Finally, in Figure 2, we show the execution time of the algorithm for each of the problems, depending on the size of the input. On the scatterplot, the cubic fit to the data has been plotted using $\|\Sigma\|$ as the predictor. It can be seen how the theoretical polynomial fit mentioned in Theorem 5 is also reflected in the experimental evaluation.

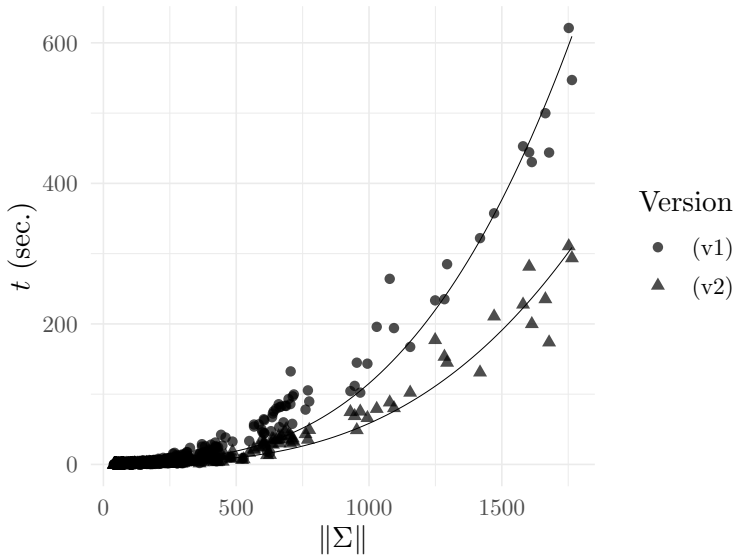

Figure 2. Execution time of versions (v1) and (v2), with cubic adjustment.

Discussion

The topic of attribute reduction and simplification of implicational systems is important insofar as it can be considered a preliminary step to other algorithms (computation of logical closure, generation of direct bases) whose computational complexity depends directly on the size of the implicational system; however, to the best of our knowledge, there is no previous algorithm for simplifying implications with mixed attributes and, thus, our proposal is the first one in this extended framework.

We have presented experimental evidence of the algorithm's ability to reduce the size of an implicational system with mixed attributes. Specifically, we have presented two versions of the algorithm: the first one (v1) simply guarantees the obtainment of an

sm-implicational system, and the most complete one (v2) obtains more simplified results, achieving an average reduction of about 75% of the initial system size (see Figure 1), even with a lower execution time (see Figure 2).

7. Conclusions and Future Work

We have proposed a novel logic-based method able to face the problem of simplifying implicational systems consisting of positive and negative attributes. This method is based on the *simplification logic for mixed attributes*, which provides a sound and complete framework to deal with this type of formal context. Among the contributions of this paper, we have found a set of logical equivalence rules for mixed attribute implications suitable for computational implementation and an algorithm to simplify an implicational system with a polynomial time-complexity. The experimental results show very promising results with respect to the reduction ratio of the algorithm.

Concerning future work, a thorough study of the minimality properties of sm-implicational systems is needed in order to pursue definitions similar to the different types of bases for classical implicational systems [19]. This will certainly be a first step on which to build automated reasoning methods based on *simplification logic*.

Another interesting direction for future work is the extension of these results with unknown information, which may be due to the existence of partial information in the data, or because the context has been obtained by compacting the data, as proposed in [30]. In this line, we will follow the research line initiated in [31] and consider the formal contexts as trivalued relations in which a third value representing the unknown is added apart from the standard positive and negative values.

Extending the work presented here to the case of association rules will require adaptation of the axiomatic system and equivalence properties in order to ensure the correctness of the scheme and the preservation of the *informativeness* and completeness properties of a basis of rules, and this task will be undertaken in future work.

Finally, in this work, we have focused on rules with conjunctive antecedents and consequents, where both positive and negative attributes appear. However, this is not the only point of view through which to model the problem of negation. In particular, the use of disjunctive rules, such as in [32], is of particular interest because of their generality. As part of our future work, we will explore the relationship of the axiomatic system and the equivalence rules described under this construction.

Author Contributions: Conceptualization, F.P.-G., D.L.-R., P.C., Á.M. and M.O.-A.; Investigation, F.P.-G., D.L.-R., P.C., Á.M. and M.O.-A.; Validation, F.P.-G., D.L.-R., P.C., Á.M. and M.O.-A. All authors have read and agreed to the published version of the manuscript.

Funding: The authors were partially supported by the Spanish Ministry of Science, Innovation, and Universities (MCIU), State Agency of Research (AEI), Junta de Andalucía (JA), Universidad de Málaga (UMA) and European Regional Development Fund (FEDER) through the projects PGC2018-095869-B-I00 (MCIU/AEI/FEDER), TIN2017-89023-P (MCIU/AEI/FEDER), PRE2018-085199 and UMA2018-FEDERJA-001 (JA/UMA/FEDER).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Proofs of the Technical Results

Proof of Theorem 1. [KeyEq] Clearly, $\{A \to B, C \to D\} \vdash \{A \to B\}$. Let us prove $\{A \to B\} \vdash \{C \to D\}$ under the hypotheses. Assume $A \subseteq C$ and consider $x \in B \cap \overline{C}$. Then:

(1) $A \to B$ (by hypothesis);
(2) $\overline{x} \to \overline{x}$ (by [Ref]);
(3) $A\overline{x} \to B\overline{x}$ (by [Comp] on (1) and (2));
(4) $C \to A$ (by [GenRef], since $A \subseteq C$);
(5) $C \to A\overline{x}$ (by [Comp] with (2) and (4), since $\overline{x} \in C$, by hypothesis);
(6) $C \to B\overline{x}$ (by [Simp] on (5) and (3));
(7) $B \to x$ (by [GenRef], since $x \in B$);
(8) $B\overline{x} \to M\overline{M}$ (by [Key]on (7));
(9) $C \to M\overline{M}$ (by [Simp] on (6) and (9));
(10) $C \to D$ (by [Frag]).

[InKeyEq] Since $\{A \to M\overline{M}, C \to D\overline{x}\} \vdash_S \{A \to M\overline{M}, C \to D\}$ by [Frag], let us show that, under the hypotheses, we can infer $C \to D\overline{x}$ from $A \to M\overline{M}$ and $C \to D$:

(1) $(A \setminus x)x \to M\overline{M}$ (by hypothesis);
(2) $A \setminus x \to \overline{x}$ (by [InKey] on (1));
(3) $C \to A \setminus x$ (by [GenRef] and $A \setminus x \subseteq C$);
(4) $C \to \overline{x}$ (by [Simp] on (2) and (3));
(5) $C \to D$ (by hypothesis;)
(6) $C \to D\overline{x}$ (by [Comp] on(4) and (5)).

[RedEq] $\{A \to B, C \setminus \overline{x} \to D\} \vdash_S \{A \to B, C \to D\}$ holds since:

(1) $C \to C \setminus \overline{x}$ (by [GenRef]);
(2) $C \setminus \overline{x} \to D$ (by premise);
(3) $C \to D$ (by [Simp] on (1) and (2)).

Let us show now that we have $\{A \to B, C \to D\} \vdash_S \{C \setminus \overline{x} \to D\}$:

(1) $C \setminus \overline{x} \to A \setminus x$ (by hypothesis and [GenRef]);
(2) $x \to x$ (by [Ref]);
(3) $(C \setminus \overline{x})x \to (A \setminus x)x$ (by [Comp] on (1) and (2));
(4) $A \to B$ (by premise);
(5) $(C \setminus \overline{x})x \to B$ (by [Simp] on (3) and (4));
(6) $B \to D$ (by hypothesis and [GenRef]);
(7) $(C \setminus \overline{x})x \to D$ (by [Simp] on (5) and (6));
(8) $C = (C \setminus \overline{x})\overline{x} \to D$ (by premise);
(9) $C \setminus \overline{x} \to D$ (by [Red] on (7) and (8)).

□

Proof of Theorem 2. It suffices to show that [Key], [InKey] and [Red] hold in the new system, since the converse is true by Theorem 1.

[Key] Let us suppose $A \to b$ and show that $A\overline{b} \to M\overline{M}$:

(1) $A \to b$ (premise);
(2) $A\overline{b} \to A\overline{b}$ (by [Ref]);
(3) $A\overline{b} \to M\overline{M}$ (by [Key′] on (1) and (2)).

[InKey] Let is suppose $Ab \to M\overline{M}$ and prove that $A \to \overline{b}$:

(1) $Ab \to M\overline{M}$ (premise);
(2) $A \to A$ (by [Ref]);
(3) $A \to A\overline{b}$ (by [InKey′] on (1) and (2));
(4) $A \to \overline{b}$ (by [Frag]).

[Red] Assuming $Ab \to C$ and $A\overline{b} \to C$, then, by [Red′], we can infer $A \to C$.

□

Proof of Lemma 1. [ContEq] We only need to show how to derive $A \to B$ from the axioms. Suppose there is $x \in A \cap \overline{A}$, i.e., $x, \overline{x} \in A$:

(1) $A \to x\overline{x}$ (by [GenRef]);
(2) $x\overline{x} \to M\overline{M}$ (by [Cont]);
(3) $A \to M\overline{M}$ (by [Simp] on (2) and (3));
(4) $A \to B$ (by [Frag]).

[ContEq′] Firstly note that by [Key] and [InKey] we have $\{A \to M\overline{M}\} \equiv \{A \setminus x \to \overline{x}\}$ for all $x \in A$. In addition, $\{A \to M\overline{M}\} \vdash \{A \to B\}$, so it suffices to show that we can infer $A \to B \vdash A \to M\overline{M}$. Consider $y \in AB \cap \overline{B}$; note that, in particular, $\overline{y} \in \overline{B} \subseteq \overline{AB}$. Then:

(1) $A \to B$ (premise);
(2) $A \to AB$ (by [Augm] on (1));
(3) $A \to y\overline{y}$ (by [Frag] on (2));
(4) $y\overline{y} \to M\overline{M}$ (by [Cont]);
(5) $A \to M\overline{M}$ (by [Simp] on (3) and (4)).

[ContEq″] As with [ContEq′], it now suffices to prove $\{A \to B, C \to D\} \vdash C \to M\overline{M}$ under the hypotheses. Thus:

(1) $CD \to A$ (by [GenRef]);
(2) $C \to D$ (premise);
(3) $C \to A$ (by [Simp] on (1) and (2));
(4) $A \to B$ (premise);
(5) $C \to B$ (by [Simp] on (3) and (4));
(6) $C \to BCD$ (by [Comp] on (2) and (5) and $C \to C$);
(7) $C \to M\overline{M}$ (by [ContEq′]).

□

Proof of Theorem 3.

[KeyEq′] We will prove just the first equivalence, since the second one is obtained by the application of [Key].

⇒ Assume $A \to B$ and $C \to D$. Since $A \to B \setminus y$ can be obtained by [Frag], we just have to show how to infer $C \setminus \overline{y} \to y$:

(1) $(A \setminus x)x \to y$ (by [Frag] on the premise $A \to B$);
(2) $(C \setminus \overline{y})x \to y$ (by hypothesis $A \setminus x = C \setminus \overline{y}$);
(3) $(C \setminus \overline{y})\overline{y} \to x$ (by [Frag] on the premise $C \to D$);
(4) $(C \setminus \overline{y})\overline{x} \to y$ (by [Rft]);
(5) $C \setminus \overline{y} \to y$ (by [Red] on (2) and (4)).

⇐ Under the hypotheses, suppose $A \to B \setminus y$ and $C \setminus \overline{y} \to y$; let us prove that $A \to B$ and $C \to D$:

(1) $A \setminus x \to y$ (by premise and $C \setminus \overline{y} = A \setminus x$);
(2) $A \to A \setminus x$ (by [GenRef]);
(3) $A \to y$ (by [Simp] on (1) and (2));
(4) $A \to B$ (by [Comp] of (3) with premise $A \to B \setminus y$);
(5) $(C \setminus \overline{y})\overline{y} \to M\overline{M}$ (by [Key] on the second premise);
(6) $C \to D$ (by [Frag]).

[KeyEq″] ⇒ Consider $x \in C$ and $y \in B \cap \overline{D}$ and assume as premises $A \to B$ and $C \to D$.

(1) $C \to A$ (by [GenRef]);
(2) $A \to B$ (premise);
(3) $C \to B$ (by [Simp] on (1) and (2));
(4) $C \to BD$ (by [Comp] on (3) and premise $C \to D$);
(5) $C \setminus x \to \overline{x}$ (by Lemma 1 [ContEq″], since $y, \overline{y} \in BD$).

⇐ Now, assuming $A \to B$ and $C \setminus x \to \overline{x}$, let us prove $C \to D$:

(1) $C \to M\overline{M}$ (by [Key] of second premise);
(2) $C \to D$ (by [Frag]).

[RedEq′] Let us denote $S = A \setminus x = C \setminus \overline{x}$, so $A = Sx$ and $C = S\overline{x}$. Then:

$$\begin{aligned}\{A \to B, C \to D\} &\equiv \{A \to B \setminus D, A \to D, C \to D\} && \text{(by [Frag] and [UnEq])}\\ &\equiv \{A \to B \setminus D, Sx \to D, S\overline{x} \to D\} && \text{(since } A = Sx \text{ and } C = S\overline{x}\text{)}\\ &\equiv \{A \to B \setminus D, S = C \setminus \overline{x} \to D\} && \text{(by [RedEq])}\end{aligned}$$

[RftEq] $\Rightarrow$ Let us show that $\{A \to B, C \to D\} \vdash \{A \to B \setminus y, C \to D\overline{x}\}$:

(1) $A \to B$ (premise);
(2) $C \to D$ (premise);
(3) $A \to B \setminus y$ (by [Frag] on (1));
(4) $(A \setminus x)x \to y$ (by [Frag] on (1));
(5) $(A \setminus x)\overline{y} \to \overline{x}$ (by [Rft] on (4));
(6) $(C \setminus \overline{y})\overline{y} \to \overline{x}$ (by hypothesis $A \setminus x = C \setminus \overline{y}$);
(7) $C \to D\overline{x}$ (by [UnEq] on (2) and (6)).

$\Leftarrow$ For the converse:

(1) $A \to B \setminus y$ (premise);
(2) $(C \setminus \overline{y})\overline{y} \to \overline{x}$ (by [Frag] on the premise $C \to D\overline{x}$);
(3) $(C \setminus \overline{y})x \to y$ (by [Rft] on (2);
(4) $(A \setminus x)x \to y$ (by hypothesis $A \setminus x = C \setminus \overline{y}$);
(5) $A \to B$ (by [Comp] on (1) and (4));
(1) $C \to D$ (by [Frag] on $C \to D\overline{x}$).

[RftEq′] $\Rightarrow$ $\{A \to B, C \to D\} \vdash_S \{A \to B, C \to D \setminus \overline{x}\}$ by [Frag].

$\Leftarrow$ It suffices to show that we can infer $C \to D$ from $\{A \to B, C \to D \setminus \overline{x}\}$:

(1) $A \to B$ (premise);
(2) $C \to D \setminus \overline{x}$ (premise);
(3) $(A \setminus x)x \to y$ (by [Frag] on (1));
(4) $(A \setminus x)\overline{y} \to \overline{x}$ (by [Rft] on (3));
(5) $C \setminus \overline{y} \to A \setminus x$ (by hypothesis and [GenRef]);
(6) $(C \setminus \overline{y})\overline{y} \to (A \setminus x)\overline{y}$ (by [Augm] on (5));
(7) $C \to \overline{x}$ (by [Simp] on (4) and (6));
(8) $C \to D$ (by [UnEq] on (2) and (7)).

[MixUnEq] Let us write $S = A \setminus x = C \setminus y$. Then:

$\Rightarrow$ Since $C \to D \setminus b$ can be obtained by using [Frag], we just need to prove $S\overline{b} \to \overline{xy}$:

(1) $Sx \to b$ (premise);
(2) $Sy \to b$ (by [Frag] on $C \to D$);
(3) $S\overline{b} \to \overline{x}$ (by [Rft] on (1));
(4) $S\overline{b} \to \overline{y}$ (by [Rft] on (2));
(5) $S\overline{b} \to \overline{xy}$ (by [UnEq] on (3) and (4)).

$\Leftarrow$ Assume premises $\{S\overline{b} \to \overline{xy}, C \to D \setminus b\}$ and let us prove $A \to b$ and $C \to D$:

(1) $S\overline{b} \to \overline{x}$ (by [Frag] on first premise);
(2) $A \to b$ (by [Rft] on (1) and $Sx = A$);
(3) $S\overline{b} \to \overline{y}$ (by [Frag] on first hypothesis);
(4) $C \to b$ (by [Rft] on (3) and $Sy = B$);
(5) $C \to D \setminus b$ (second premise);
(1) $C \to D$ (by [UnEq] on (4) and (5)).

□

References

1. Staab, S.; Studer, R. *Handbook on Ontologies*, 2nd ed.; Springer Publishing Company, Incorporated: New York, NY, USA, 2009.
2. Messaoudi, A.; Missaoui, R.; Ibrahim, M.H. Detecting Overlapping Communities in Two-mode Data Networks using Formal Concept Analysis. *Revue des Nouvelles Technologies de l'Information* **2019**, *RNTI-E-35*, 189–200.
3. Ibrahim, M.H.; Missaoui, R.; Vaillancourt, J. Identifying Influential Nodes in Two-Mode Data Networks Using Formal Concept Analysis. *IEEE Access* **2021**, *9*, 159549–159565.

4. Cordero, P.; Enciso, M.; López, D.; Mora, A. A conversational recommender system for diagnosis using fuzzy rules. *Expert Syst. Appl.* **2020**, *154*, 113449.
5. Cordero, P.; Enciso, M.; Ángel Mora.; Ojeda-Aciego, M.; Rossi, C. A Formal Concept Analysis Approach to Cooperative Conversational Recommendation. *Int. J. Comput. Intell. Syst.* **2020**, *13*, 1243–1252.
6. Agrawal, R.; Srikant, R. Fast Algorithms for Mining Association Rules in Large Databases. In Proceedings of the 20th International Conference on Very Large Data Bases, Santiago de Chile, Chile, 12–15 September 1994; VLDB '94, pp. 487–499.
7. Boulicaut, J.F.; Bykowski, A.; Jeudy, B. Towards the Tractable Discovery of Association Rules with Negations. In *Flexible Query Answering Systems*; Larsen, H.L., Andreasen, T., Christiansen, H., Kacprzyk, J., Zadrożny, S., Eds.; Springer Publishing Company: New York, NY, USA, 2001; pp. 425–434.
8. Wu, X.; Zhang, C.; Zhang, S. Efficient mining of both positive and negative association rules. *ACM Trans. Inf. Syst. (TOIS)* **2004**, *22*, 381–405.
9. Ben Yahia, S.; Gasmi, G.; Mephu Nguifo, E. A new generic basis of "factual" and "implicative" association rules. *Intell. Data Anal.* **2009**, *13*, 633–656.
10. Missaoui, R.; Nourine, L.; Renaud, Y. An Inference System for Exhaustive Generation of Mixed and Purely Negative Implications from Purely Positive Ones. In Proceedings of the 7th International Conference on Concept Lattices and Their Applications, Sevilla, Spain, 19–21 October 2010; CEUR Workshop Proceedings, Volume 672, pp. 271–282.
11. Missaoui, R.; Nourine, L.; Renaud, Y. Computing Implications with Negation from a Formal Context. *Fundam. Informaticae* **2012**, *115*, 357–375.
12. Rodríguez-Jiménez, J.M.; Cordero, P.; Enciso, M.; Mora, A. Data mining algorithms to compute mixed concepts with negative attributes: an application to breast cancer data analysis. *Math. Methods Appl. Sci.* **2016**, *39*, 4829–4845.
13. Cordero, P.; Enciso, M.; Mora-Bonilla, A.; Rodríguez-Jiménez, J., Inference of Mixed Information in Formal Concept Analysis. In *Trends in Mathematics and Computational Intelligence*; Studies in Computational Intelligence; Springer: Cham, Switzerland, 2019; pp. 81–87.
14. Zaki, M.J. Generating Non-Redundant Association Rules. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, USA, 20–23 August 2000; pp. 34–43.
15. Cheng, J.; Ke, Y.; Ng, W. Effective elimination of redundant association rules. *Data Min. Knowl. Discov.* **2008**, *16*, 221–249.
16. Díaz Vera, J.; Negrín Ortiz, G.; Molina, C.; Amparo Vila, M. Knowledge redundancy approach to reduce size in association rules. *Informatica* **2020**, *44*, 167–181.
17. Jin, M.; Wang, H.; Zhang, Q. Association rules redundancy processing algorithm based on hypergraph in data mining. *Clust. Comput.* **2019**, *22*, 8089–8098.
18. Mora, A.; Cordero, P.; Enciso, M.; Fortes, I.; Aguilera, G. Closure via functional dependence simplification. *Int. J. Comput. Math.* **2012**, *89*, 510–526.
19. Rodríguez Lorenzo, E.; Bertet, K.; Cordero, P.; Enciso, M.; Mora, A. Direct-optimal basis computation by means of the fusion of simplification rules. *Discret. Appl. Math.* **2018**, *249*, 106–119.
20. Ganter, B.; Wille, R. *'Formal Concept Analysis' Mathematical Foundations*; Springer: Berlin, Germany, 1996.
21. Wille, R. Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts. *Ordered Sets* **1982**, *83*, 445–470.
22. Konecny, J. Attribute implications in L-concept analysis with positive and negative attributes: Validity and properties of models. *Int. J. Approx. Reason.* **2020**, *120*, 203–215.
23. Rodríguez-Jiménez, J.M.; Cordero, P.; Enciso, M.; Rudolph, S. Concept lattices with negative information: A characterization theorem. *Inf. Sci.* **2016**, *369*, 51–62.
24. Rodríguez-Jiménez, J.M. Extracción de Conocimiento Usando Atributos Negativos en el Análisis de Conceptos Formales Aplicaciones en la Ingeniería. Ph.D. Thesis, Universidad de Málaga, Málaga, Spain, 2017.
25. Kuznetsov, S.O. On the Intractability of Computing the Duquenne-Guigues Base. *J. Univers. Comput. Sci.* **2004**, *10*, 927–933.
26. Distel, F.; Sertkaya, B. On the complexity of enumerating pseudo-intents. *Discret. Appl. Math.* **2011**, *159*, 450–466.
27. Babin, M.A.; Kuznetsov, S.O. Computing premises of a minimal cover of functional dependencies is intractable. *Discret. Appl. Math.* **2013**, *161*, 742–749.
28. Guigues, J.L.; Duquenne, V. Familles Minimales d'Implications Informatives Résultant d'un Tableau de Données Binaires. *Mathématiques Sci. Hum.* **1986**, *95*, 5–18.
29. López-Rodríguez, D.; Mora, A.; Domínguez, J.; Villalón, A.; Johnson, I. fcaR: Formal Concept Analysis. R Package Version 1.1.0. 2020. Available online: https://cran.r-project.org/web/packages/fcaR/index.html (accessed on 14 December 2021).
30. Ganter, B.; Meschke, C. A Formal Concept Analysis Approach to Rough Data Tables. In *Transactions on Rough Sets XIV*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 37–61.
31. Pérez-Gámez, F.; Cordero, P.; Enciso, M.; Mora, A. A New Kind of Implication to Reason with Unknown Information. *Lect. Notes Comput. Sci.* **2021**, *12733*, 74–90.
32. Hamrouni, T.; Ben Yahia, S.; Mephu Nguifo, E. Sweeping the disjunctive search space towards mining new exact concise representations of frequent itemsets. *Data Knowl. Eng.* **2009**, *68*, 1091–1111.