$\mathcal{R}$-generalization of application forms to limit data collection

Nicolas Anciaux[1,3]
[1]INRIA Rocquencourt
Domaine de Voluceau
78153 Le Chesnay, FR
nicolas.anciaux@inria.fr

Marouane Fazouane[1,2]
[2]ENSTA-Paristech
828, bd des Maréchaux
91762 Palaiseau, FR
fazouane@ensta.fr

Benjamin Nguyen[1,3]
[3]Univ. de Versailles St-Quentin
45, av. des Etats-Unis
78035 Versailles, FR
benjamin.nguyen@inria.fr

ABSTRACT

Online services such as e-admin, e-banking, etc. use complex decision processes (fed by forms) to calibrate the offer (benefits) they make to each applicant. These decision processes require many personal data items, which are subsequently processed and stored. Removing from users' application forms the personal data items which are not strictly useful for its subsequent evaluation by a service provider is imposed by privacy laws enacted worldwide, and is useful for both service providers and users. We show in this paper that an algorithm reducing the information exposed in an application form is not sufficient. Indeed, missing information can be revealed by exploiting the set of benefits obtained, the decision rules, and any kind of background knowledge, including knowledge about the algorithm used. The contributions of this article are fourfold: (a) we formally define the data reduction problem in the context of logical inference attacks and introduce the concept of $\mathcal{R}$-generalization, (b) we propose algorithms to compute these logical inference attacks and study their complexity, (c) we modify existing data reduction algorithms and propose new algorithms to defeat these attacks, and (d) demonstrate their efficiency on real and synthetic data.

Categories and Subject Descriptors

H.2.0 [**Information Systems**]: Database Management – *General.* K.4.1 [**Computing Milieux**]: Comuters and Society – *Public Policy Issues.*

General Terms

Logic, Privacy, Algorithms.

Keywords

Privacy, Limited Data Collection, 3-valued Logic, Inference on Lattices, R-Generalization.

1. INTRODUCTION

Disclosing personal data when applying to online services has become unavoidable. In this article, we consider services such as banking, social care, income tax, etc. for which providers request users to transmit their personal data via application forms. This data is used to construct a personalized answer (in general a commercial proposition), subsequently returned to the applicant. In order to customize this proposition regarding the specific situation of each applicant, more and more complex decision making systems are used. These systems need to be fed with an increasing amount of data, harvested through large forms. Indeed, it is common today for a user to fill in forms with hundreds of personal data items, e.g. like in the domain of bank loans[1] and social care[2]. This causes serious issues for both users (privacy problems) and service providers (storing and processing cost, responsibility and financial cost in the case of data leaks)

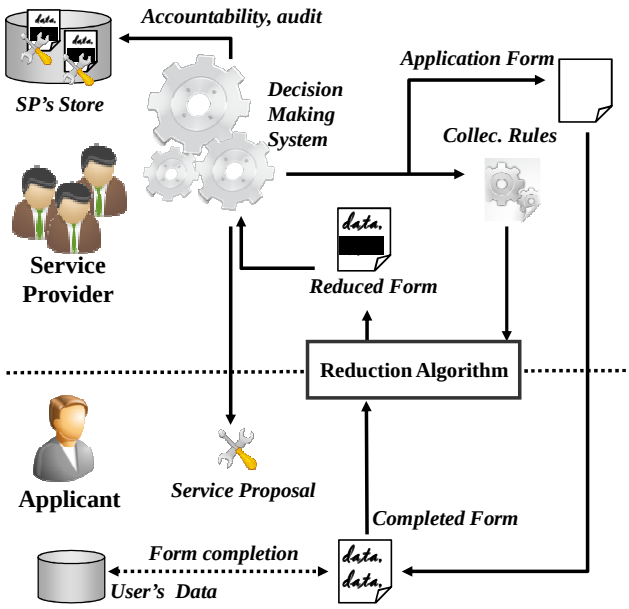

Figure 1. Users' application form content reduction.

Limited Data Collection and Retention (LDC), a privacy principle enacted worldwide [8, 15] is the current answer to minimize the impact of these issues. This principle states that service providers must not collect any data items that have no consequence on the offer made to the applicant. LDC is currently implemented by placing as prerequisite to data treatments, a reduction module in charge of cleansing forms of any data item which would not change the final decision (see Figure 1), a process that we will call henceforth *data reduction* (an abstraction of the *Minimum Exposure* concept we proposed in [3, 4]). Data reduction is performed using (i) the set of user data values or predicates provided in its application form, and (ii) a set of logical

[1] The mortgage application form of Nationwide Building Society (the largest building society in the world) contains a large number of data items. See http://www.nationwide.co.uk/nr/rdonlyres/a48ffc87-7e29-4ea6-b24d-2720746c5d9e/0/m1inov06.pdf

[2] The GEVA application form is provided in France by General Councils to calibrate financial and material assistance for dependant people, and requests more than 600 data items. See http://www.cnsa.fr/IMG/pdf/GEVA_graphique-080529-2.pdf

formulas built by the service provider, called *collection rules*, associating each potential benefit (e.g. lower interest rate, length of loan) with a set of predicates on users' data items.

Example *(collection rules).* Social care services propose different benefits to dependent people: Financial support for home aid is granted to an applicant if: (i) her pension is under €30.000 and her age is above 80, or (ii) she has more than two lost abilities (e.g., dressing and bathing independently). Support for simple home adaptation is provided in the following cases: having (i) a pension under €30.000 and no nursing bed, or (ii) living alone and having no relatives in the neighborhood. The corresponding collection rules can be expressed by a Boolean DNF formula :

$$(\ pension \leq 30.000 \wedge age \geq 80\ ) \vee (\ lost\_abilities \geq 3\ ) \rightarrow home\_aid$$
$$(\ pension \leq 30.000 \wedge nursing\_bed = \text{'}no\text{'}\ ) \vee (\ alone = \text{'}yes\text{'} \wedge relatives = 0\ ) \rightarrow home\_adpt$$

Collection rules can be built by experts working for the service providers, directly derived from laws and directives, (e.g. social care application presented in [3]), or even generated automatically (e.g., if the decision making system involves data mining tools, such as neural networks or support vector machines, algorithms can be used [5] to transform them into collection rules).

The (form) reduction operation computes the benefits of the applicant given her (complete) application form, then reduces the amount of data to be exposed by constructing a reduced form that yields the same benefits. Data reduction can be performed using a state-of-the-art Binary Integer Program solver as proposed in [4], or using simple heuristics as proposed in the context of a smartcard implementation used to hide the collection rules and the user's complete form [3]. In practice, even very simple algorithms lead to an important reduction (around 50% [3]) of the number of exposed data items.

Example *(reduced application form).* Considering the collection rules presented above, assume an applicant who satisfies the predicates $pension \leq 30.000$, $age \geq 80$ and $alone = yes$, but not the others. The benefit *home_aid* can be obtained, but not *home_adpt*. A reduction algorithm may remove from that user's application form the attributes *lost_abilities*, *nursing_bed* and *relatives* and keep *pension* and *age*, since the publication of these two predicates will yield the same benefits as publishing the whole set.

We argue in this paper that providing a reduced form only makes sense if the set of removed data items cannot be derived from the result of that reduction. We give below three simple attacks leading to infer information about unexposed data items, and then illustrate those attacks using an example.

Attack 1. A first attack can be conducted considering that (i) the reduction algorithms preserve the service offer, i.e. any benefit which is obtained using the complete application must also be obtained after the reduction, and (ii) the collection rules are known by the attacker. Indeed, collection rules are public in many cases (e.g. tax return or social care), and are known by the service provider who generated them. Confronting the reduced form and the collection rules easily leads to infer valuations of predicates involving users attributes that are not exposed.

Attack 2. A second attack can be conducted if some unrevealed data items or predicates are known by the attacker. This will typically be the case by using background knowledge (e.g. obtained by successive publication of forms) to infer the values of predicates involving other attributes.

Attack 3. A third attack can be made if the algorithm used to perform the reduction is public or if the algorithm is based on well known heuristics. An attacker could use the algorithm to identify predicates which would have been selected in the output had their valuation been true, but were not provided, and thus infer that their valuation is false.

Example *(attacks).* Consider the collection rules and the reduced form presented in the above examples. As shown in Fig. 2, according to attack 1, the reduced form does lead to the benefit *home_adpt*, thus we can conclude that *nursing_bed* = '*no*' is false (i.e. *nursing_bed*='*yes*'). Let us suppose that the attacker has the background knowledge that *alone* = '*yes*', attack 2 leads to conclude that *relatives=1*. Finally, suppose that we know that a simple heuristic reduction algorithm which chooses the shortest conjunction of predicates for each obtained benefit was used (attack 3). We can infer that *lost_abilities* ≤ *2*, unveiling the complete application form.

Complete form

pension	*age*	*lost_abilities*	*nursing_bed*	*alone*	*relatives*
≤ 30.000	*≥ 80*	*≤ 2*	*='yes'*	*='yes'*	*≥ 1*

↓ *Reduction algorithm*

Reduced form

≤ 30.000	*≥ 80*	?	?	?	?

↓ *Attack 1*

≤ 30.000	*≥ 80*	?	*= 'yes'*	?	?

↓ *Attack 2 (knowing that alone='yes')*

≤ 30.000	*≥ 80*	?	*= 'yes'*	*='yes'*	*= 1*

↓ *Attack 3*

Inferred form

≤ 30.000	*≥ 80*	*≤ 2*	*= 'yes'*	*='yes'*	*= 1*

Figure 2. Example of a (fully successful) attack.

These simple attack examples show that a reduction algorithm which would not integrate such simple knowledge would produce erroneous results.

In this paper, we propose a new model of the reduction problem which helps identifying the set of unrevealed

predicates that can be inferred from a reduced form, considering different kinds of background knowledge. Such a model is a primary requirement to quantify the information *really* exposed by applicants publishing application forms obtained using existing reduction algorithms. This is also the foundation to propose a new family of reduction algorithms, which would be based *by design* on the computation of their own inference. The precise contributions of this paper are the following:

(i) we formally define the data reduction problem in the context of logical inference attacks (Section 2);

(ii) we propose algorithms to compute the maximum inference given different background knowledge (Section 3);

(iii) we modify existing algorithm to take into account inference and propose a new genetic algorithm resistant to the aforementioned attacks to compute the reduction of an application form (Section 4);

(iv) we show by a set of experiments on real and synthetic datasets the quantity of the information that can be inferred using existing reduction algorithms, and we confront these results with the inference-based reduction algorithm that we propose (Section 6).

We also provide a overview of related works in Section 5, and a conclusion in Section 7.

2. NOTATIONS AND PROBLEM STATEMENTS

In this section, we define the concepts and notations used in the rest of the paper. We conclude this section by formalizing the reduction problem in the context of data inference.

2.1. Prerequisites

3-valued logic

In the context of forms, data reduction is performed by removing information. This can be seen as changing the value of a predicate to *unknown*. In order to capture this, we use 3-valued logic in the rest of the paper. The semantics of the third value are *unknown* or *undefined*. We call ternary and note $\mathcal{T}$={0, 1, ?} the set where 1 is *true*, 0 is *false* and ? is *undefined* or *unknown*. Analogously, we note $\mathcal{B}$={0, 1} the set of Boolean values. We introduce a partial ordering over $\mathcal{T}$ which is semantically equivalent to an increase in knowledge: the following inequalities hold: $? \leq 0$ and $? \leq 1$; 0 and 1 are incomparable. This ordering can be represented as a lattice in which each node is greater than its direct child (Fig. 3).

The connectors used in this article are the classical Boolean connectors extended to 3-valued logic so that DeMorgan's theorem and other properties (commutative, distributive, idempotent …) still hold.

We assume that, in all the following, 3-valued logic is fully defined by $(\mathcal{T} = \{0; 1; ?\} \wedge \vee \neg \leq)$ and that any logical expression written using Boolean algebra $(\mathcal{B} = \{0; 1\} \wedge \vee \neg)$ is written by extension in 3-valued logic.

Table 1. Basic operations truth table

A	B	$A \vee B$	$A \wedge B$	$\neg A$
1	1	1	1	0
1	?	1	?	0
1	0	1	0	0
?	?	?	?	?
?	0	?	0	?
0	0	0	0	1

Notation convention

We note with upper case letters of the beginning of the alphabet (*A*, *B*,…) tuples of Boolean predicates and of the end of the alphabet (*X*, *Y*,…) ternary predicates. We note $A.p_i$ the *i*-th predicate of *A*. We write as a shorthand, e.g. *A*=(10?) to indicate that $A.p_1$=1, $A.p_2$=0 and $A.p_3$=?. Letters written in upper case scripts' style are sets, e.g. $\mathcal{B}$, $\mathcal{T}$, $\mathcal{R}$. Functions and propositional formulas are noted in bold, e.g. $\boldsymbol{\mathcal{F}}$ or $\boldsymbol{\Psi}$.

The lattice vision

We assume that an individual is fully defined by its own predicates, and therefore model an individual as a tuple of Boolean predicates. Running a data reduction process on an individual means reducing the knowledge in a tuple by *generalizing* some values to unknown.

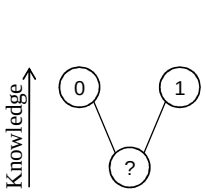

Figure 3. The Lattice $(\mathcal{T}, \leq)$

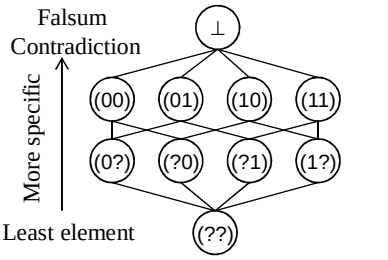

Figure 4. Lattice representation of $(\mathcal{T}^2, \leq)$

More formally, let Alice be an individual $(A = (A.p_1,\ A.p_2,\ \ldots,\ A.p_n) \in \mathcal{B}^n)$.). For some $X = (X.p_1,\ X.p_2,\ \ldots, X.p_n) \in \mathcal{T}^n$, X is said to be a *generalization* of A iff. $\forall i, X.p_i \leq A.p_i$. X is said to be a *proper generalization* of *A* if it's also different from *A*. One can prove easily that the relation "is a generalization of" is actually a partial ordering on $\mathcal{T}^n$ which we will denote unambiguously by $\leq$. For instance, consider Fig. 4, (?1) is both a generalization of (01) and (11) and a specialization of (??). We see that (?1) is the greatest lower bound of (01) and (11). We note the greatest lower bound $(?1) = inf_{(\mathcal{T}^2, \leq)}\{(01), (11)\}$. For the sake of readability, in the rest of this article, we will omit the subscript.

More generally, $\leq$ is a well-founded ordering in which its least element is *X*=(??...?), its greatest element (falsum, representing contradiction) is noted $\perp$, and the number of elements in $\mathcal{T}^n$ that have exactly *k* undefined predicates is $\binom{n}{k}.2^{n-k}$. Thus, the cardinal of this lattice (without counting

the falsum) is 3^n. The main benefit from having a falsum element is the ease of using a supremum in all cases. For instance, $(11) = sup\{(?1), (1?)\}$ and $\perp = sup\{(01), (1?)\}$. Thus, semantically, $inf(J)$represents the greatest element that generalizes all elements of J and $sup(J)$ is the least element that specializes all elements of J.

The lattice representation will be used to conduct inferences and quantify using the greatest lower bound of possible individuals, or a set of lower bounds, the efficiency of the data reduction applied to a form.

Proposition 1

Let $\boldsymbol{\mathcal{F}}$ be a propositional formula. The three following equivalent statements hold:
1. $\boldsymbol{\mathcal{F}}$ is monotonically increasing in $\mathcal{T}^n$. 2. $\forall X, X' \in \mathcal{T}^n, X' \geq X \Longrightarrow \boldsymbol{\mathcal{F}}(X') \geq \boldsymbol{\mathcal{F}}(X)$ 3. $\forall X \in \mathcal{T}^n, \boldsymbol{\mathcal{F}}(X) \in \mathcal{B} \Longrightarrow \forall Y \geq X, \boldsymbol{\mathcal{F}}(Y) = \boldsymbol{\mathcal{F}}(X)$

Proof. 1 is a definition, and the equivalence between 2 and 3 is straightforward.

2.2. Problem description (Definitions/Axioms)

In this subsection, we formalize the classical concepts used in data reduction (Individual, Form, Rule Set, Exposure Function, Exposure Reduction Algorithm). We include additional concepts ($\mathcal{R}$-generalization, Maximum Inference, Background Knowledge) and tools (Indicator functions), to cover inference attacks. In closing, we state the problem to be solved. Rule Set, Exposure Reduction Algorithm, $\mathcal{R}$-generalization are considered as **axioms**. Note that the running example used is voluntarily very simple. Real scenarios are obviously much more complex.

Definition (Individual)

An individual *A* is an element of $\mathcal{B}^n$.

Definition (Form)

A form *X* is an element of $\mathcal{T}^n$.

Definition (Rule set and atoms)

We denote by $\mathcal{R}$ a set of propositional formulas (called "collection rules" in [4]) written in *Disjunctive Normal Form*; each conjunct (conjunctive formula) is called an *atom*. These rules semantically represent the "benefits" an applicant can obtain. We say that a form *X satisfies* a given benefit $\boldsymbol{r}$ if this benefit is *true*, when evaluated using *X* as a (ternary) model. Given a rule set $\mathcal{R}$, we introduce a new function $\boldsymbol{BF}_{\mathcal{R}}: {}^{\mathcal{T}^n \to \mathcal{P}(\mathcal{T}^n)^3}_{X \mapsto (T_{\mathcal{R}}(X), F_{\mathcal{R}}(X), U_{\mathcal{R}}(X))}$ such that:

$$\begin{cases} T_{\mathcal{R}}(X) = \{\boldsymbol{r} \in \mathcal{R} \mid \boldsymbol{r}(X) = 1\} \\ F_{\mathcal{R}}(X) = \{\boldsymbol{r} \in \mathcal{R} \mid \boldsymbol{r}(X) = 0\} \\ U_{\mathcal{R}}(X) = \{\boldsymbol{r} \in \mathcal{R} \mid \boldsymbol{r}(X) = ?\} \end{cases}$$

Where $\mathcal{P}(\mathcal{T}^n)$ is the powerset of $\mathcal{T}^n$ (also written $2^{\mathcal{T}^n}$).

Note: $\forall X \in \mathcal{T}^n, BF_R(X)$ is a partition of $\mathcal{R}$: given a form *X*, $\boldsymbol{BF}_{\mathcal{R}}$ partitions the collection rules into three sets: the set of rules that are *true* for *X* (satisfied by *X*), the set of rules that are *false* for *X* (discarded by *X*), and the set of rules that are *unknown* for *X*.

Example (Rule Set). We consider Rule Set $\mathcal{R}$, presented in the introduction defined by:

$r_1 = a_1 \vee a_2$ and $r_2 = a_3 \vee a_4$ where the atoms are defined by:

$a_1 = p_1 \wedge p_2;\ a_2 = p_3;\ a_3 = p_1 \wedge p_4; a_4 = p_5 \wedge p_6$

The semantics of the predicates are:

p_1=(*pension* ≤ *30.000*); p_2=(*age* ≥ *80*); p_3=(*lost_abilities* ≥ *2*); p_4=(*nursing_bed* = '*no*'); p_5=(*alone* = '*yes*'); p_6=(*relatives* = *0*)

and the semantics of the benefits are :

r_1= *home_aid*; r_2= *home_adpt*

Consider a form $X = (11????)$. $T_{\mathcal{R}}(X) = \{\boldsymbol{r}_1\}$, $F_{\mathcal{R}}(X) = \emptyset$, $U_{\mathcal{R}}(X) = \{\boldsymbol{r}_2\}$.

Definition ($\mathcal{R}$-generalization)

Given *X*, *Y* in $\mathcal{T}^n$ we say that *X* is a $\mathcal{R}$-generalization of *Y*, noted $X \leq_{\mathcal{R}} Y$ iff $T_{\mathcal{R}}(X) = T_{\mathcal{R}}(Y)$ and $X \leq Y$. We say conversely that *Y* is a $\mathcal{R}$-specialization of *X*.

Proposition 2

$\mathcal{R}$-generalization is a partial ordering on $\mathcal{T}^n$.

Proof is straightforward.

Definition (Indicator functions)

Given a rule set $\mathcal{R}$ and a ternary tuple *X*, we define three indicator functions on a ternary tuple *Y*:

$$\begin{cases} \boldsymbol{\mathcal{F}}_X^+\langle\mathcal{R}\rangle(Y) = \bigwedge_{r \in T_{\mathcal{R}}(X)} \boldsymbol{r}(Y) \\ \boldsymbol{\mathcal{F}}_X^-\langle\mathcal{R}\rangle(Y) = \bigwedge_{r \notin T_{\mathcal{R}}(X)} \overline{\boldsymbol{r}(Y)} \\ \boldsymbol{\mathcal{F}}_X\langle\mathcal{R}\rangle(Y) = \boldsymbol{\mathcal{F}}_X^+\langle\mathcal{R}\rangle(Y) \wedge \boldsymbol{\mathcal{F}}_X^-\langle\mathcal{R}\rangle(Y) \end{cases}$$

Note: We use $\langle\mathcal{R}\rangle$ to indicate that $\boldsymbol{\mathcal{F}}_X$ has an explicit dependence on $\mathcal{R}$.
$\boldsymbol{\mathcal{F}}_X^+\langle\mathcal{R}\rangle(Y)$ is true if *Y* satisfies *at least all* the benefits that *X* satisfies. $\boldsymbol{\mathcal{F}}_X^+\langle\mathcal{R}\rangle(Y)$ is false if *at least one* of the benefits satisfied by *X* is discarded by *Y*. $\boldsymbol{\mathcal{F}}_X^+\langle\mathcal{R}\rangle(Y)$ is unknown otherwise.
$\boldsymbol{\mathcal{F}}_X^-\langle\mathcal{R}\rangle(Y)$ is true if *Y* satisfies *none of the* benefits that *X* either *discards*, or that are unknown for *X*. $\boldsymbol{\mathcal{F}}_X^-\langle\mathcal{R}\rangle(Y)$is false if *Y* satisfies at least one benefit *discarded* by *X* or unknown for *X*. $\boldsymbol{\mathcal{F}}_X^-\langle\mathcal{R}\rangle(Y)$ is unknown otherwise.
$\boldsymbol{\mathcal{F}}_X\langle\mathcal{R}\rangle(Y)$ is true if *Y* satisfies *exactly the same* benefits as *X* (and satisfies no benefit that is unknown or discarded by *X*). $\boldsymbol{\mathcal{F}}_X\langle\mathcal{R}\rangle(Y)$ is false if *Y* satisfies at least one benefit discarded or unknown for *X*, or if *Y* does not satisfy at least one benefit satisfied by *X*. $\boldsymbol{\mathcal{F}}_X\langle\mathcal{R}\rangle(Y)$ is unknown otherwise.

In general *X* and *Y* do not need to be comparable. However, in the rest of this article, we will use these indicator

functions on tuples such that $X \leq Y$ or $Y \leq X$. Note that we do not necessarily have $\boldsymbol{\mathcal{F}}_X\langle\mathcal{R}\rangle(X) = 1$ (it is indeed possible to have for a given X, $\boldsymbol{\mathcal{F}}_X\langle\mathcal{R}\rangle(X) =?$). Note that if $A \in \mathcal{B}^n$ then $\forall X, \boldsymbol{\mathcal{F}}_X\langle\mathcal{R}\rangle(A) \in \mathcal{B}^n$.

Example (Indicator Functions).

Consider $X_1 = (111???)$ and $Y_1 = (111000)$. Then $\boldsymbol{\mathcal{F}}^+_{X_1}\langle\mathcal{R}\rangle(Y_1) = 1, \boldsymbol{\mathcal{F}}^-_{X_1}\langle\mathcal{R}\rangle(Y_1) = 1, \boldsymbol{\mathcal{F}}_{X_1}\langle\mathcal{R}\rangle(Y_1) = 1.$

Consider $X_2 = (11????)$ and $Y_2 = (11000?)$. Then $\boldsymbol{\mathcal{F}}^+_{X_2}\langle\mathcal{R}\rangle(Y_2) = 1, \boldsymbol{\mathcal{F}}^-_{X_2}\langle\mathcal{R}\rangle(Y_2) =?, \boldsymbol{\mathcal{F}}_{X_2}\langle\mathcal{R}\rangle(Y_2) =?.$

Consider $X_3 = (11????)$ and $Y_3 = (111111)$. Then $\boldsymbol{\mathcal{F}}^+_{X_3}\langle\mathcal{R}\rangle(Y_3) = 1, \boldsymbol{\mathcal{F}}^-_{X_3}\langle\mathcal{R}\rangle(Y_3) = 0, \boldsymbol{\mathcal{F}}_{X_3}\langle\mathcal{R}\rangle(Y_3) = 0.$

Note that only the output of $\boldsymbol{BF}_{\mathcal{R}}(X)$ is needed to compute these indicators, therefore they can be computed even if X is not given. This is in general the case, since the output X of a reduction algorithm will not necessarily be known, but the benefits satisfied will be.

Characterizing $\mathcal{R}$-generalization

In the rest of the paper, we want to characterize $\mathcal{R}$-generalization through functional means. We introduce the following theorem, which gives two equivalent characterizations of $\mathcal{R}$-generalization, which will be used in many proofs.

Theorem 1 (Functional characterization of $\mathcal{R}$-generalization)

Let X, Y in $\mathcal{T}^n$ then

(a) $X \leq_{\mathcal{R}} Y \Leftrightarrow \boldsymbol{\mathcal{F}}^+_Y\langle\mathcal{R}\rangle(X) = 1$ and $X \leq Y$.

(b) $X \leq_{\mathcal{R}} Y \Leftrightarrow \boldsymbol{\mathcal{F}}^-_X\langle\mathcal{R}\rangle(Y) \neq 0$ and $X \leq Y$.

Two remarks will be used in the proof:

(*) $\forall X, Y \in \mathcal{T}^n, X \leq Y \Longrightarrow T_{\mathcal{R}}(X) \subseteq T_{\mathcal{R}}(Y)$

(**) $\forall X, Y \in \mathcal{T}^n, T_{\mathcal{R}}(X) \subseteq T_{\mathcal{R}}(Y) \Leftrightarrow \boldsymbol{\mathcal{F}}^+_X\langle\mathcal{R}\rangle(Y) = 1$

Proof:

$$(a) X \leq_{\mathcal{R}} Y \Leftrightarrow \begin{cases} X \leq Y \\ T_{\mathcal{R}}(Y) = T_{\mathcal{R}}(X) \end{cases} \overset{(*)}{\Leftrightarrow} \begin{cases} X \leq Y \\ T_{\mathcal{R}}(Y) \subseteq T_{\mathcal{R}}(X) \end{cases} \overset{(**)}{\Leftrightarrow} \begin{cases} X \leq Y \\ \boldsymbol{\mathcal{F}}^+_Y\langle\mathcal{R}\rangle(X) = 1 \end{cases}$$

$$(b) X \leq_{\mathcal{R}} Y \Leftrightarrow \begin{cases} X \leq Y \\ T_{\mathcal{R}}(Y) = T_{\mathcal{R}}(X) \end{cases} \overset{(*)}{\Leftrightarrow} \begin{cases} X \leq Y \\ T_{\mathcal{R}}(Y) \subseteq T_{\mathcal{R}}(X) \end{cases} \Leftrightarrow \begin{cases} X \leq Y \\ \forall \boldsymbol{r} \in \mathcal{R}, \boldsymbol{r} \notin T_{\mathcal{R}}(X) \Longrightarrow \boldsymbol{r} \notin T_{\mathcal{R}}(Y) \end{cases} \Leftrightarrow \begin{cases} X \leq Y \\ \forall \boldsymbol{r} \notin T_{\mathcal{R}}(X), \boldsymbol{r}(Y) \neq 1 \end{cases} \Leftrightarrow \begin{cases} X \leq Y \\ \boldsymbol{\mathcal{F}}^-_X\langle\mathcal{R}\rangle(Y) \neq 0 \end{cases}$$ ■

The problem with equivalence (a), is that in our case the value of X is given, and we want to explore the domain of corresponding Y values. This will lead to manipulate different functions (one per value of Y), which will be awkward. Therefore we use the equivalent characterization (b) which exchanges the position of X and Y, which results in using only one function: $\boldsymbol{\mathcal{F}}^-_X\langle\mathcal{R}\rangle$. Moreover, note that when Y is an individual (i.e. $Y \in \mathcal{B}^n$, which is the typical case), we have $\mathcal{F}^-_X\langle\mathcal{R}\rangle(Y) \neq?$. Therefore (b) can also be written like this: $X \leq_{\mathcal{R}} Y \Leftrightarrow \begin{cases} X \leq Y \\ \mathcal{F}^-_X\langle\mathcal{R}\rangle(Y) = 1 \end{cases}$.

Example ($\mathcal{R}$-generalization). Using $\mathcal{R}$, X_1 and Y_1 as defined in previous examples, we have $X_1 \leq_{\mathcal{R}} Y_1$. Using Th.1 (a), by computing $\boldsymbol{\mathcal{F}}^+_{Y_2}\langle\mathcal{R}\rangle(X_2) = 1$, it is easy to see that $X_2 \leq_{\mathcal{R}} Y_2$. Using Th.1 (b), since $\boldsymbol{\mathcal{F}}^-_{X_3}\langle\mathcal{R}\rangle(Y_3) = 0$, we can conclude that $X_3 \not\leq_{\mathcal{R}} Y_3$. Note that by using Th.1 (b), there is no need to compute $\boldsymbol{\mathcal{F}}^+_{Y_3}\langle\mathcal{R}\rangle(X_3)$ and see that its value is *unknown*.

Definition (Exposure function)

A function $\mathbb{E}: \mathcal{T}^n \longrightarrow \mathbb{Q}^+$ is said to be an *exposure function* iff $\mathbb{E}$ is monotonically increasing and $\mathbb{E}\big((???...?)\big) = 0$.

Example (Exposure Function). In the rest of this article, we consider the simple exposure function $\mathbb{E}$ such that forall X in $\mathcal{T}^n$, $\mathbb{E}(X)$ = the number of known (true or false) predicates. It is easy to see that $\mathbb{E}$ verifies the definition.

Definition (Exposure reduction algorithm)

An algorithm α is said to be an *exposure reduction algorithm* iff every instantiation for a rule set the following holds: $\forall A \in \mathcal{B}^n, \alpha\langle\mathcal{R}\rangle(A)$ is a random variable whose values are in $\mathcal{T}^n$ and $\mathcal{R}$-generalize A.

Note: Let us stress that an exposure reduction algorithm is defined *regardless* of any exposure function.

Example (Exposure reduction algorithm). The following algorithm is a simple example :

Simple Exposure Reduction Algorithm

Input: A in $\mathcal{B}^n$, $\mathcal{R}$ a rule set
Ouput: X in $\mathcal{T}^n$

1. $X \leftarrow (??...?)$
2. *forall r in* $T_{\mathcal{R}}(A)$ *do*
3. *Chose a random atom a in r that is a true for A*
4. *Set to true in X the predicates corresponding to those in a*
5. *return X*

Note that the algorithms proposed in Section 3 are also exposure reduction algorithms.

Definition (Background knowledge)

Some background knowledge $\mathcal{K}$ is a satisfiable set of logical formulas. Minimal background knowledge is:

$$\mathcal{K}_0 = \{r \textit{ is a rule } \leftrightarrow \mathrm{r} \in \mathrm{R}\} \cup \{\textit{the axioms are true}\}$$

We assume that all background knowledge should at least contain $\mathcal{K}_0$. Note that for the sake of simplicity, we do not consider probabilistic background knowledge and reasoning. We believe that similar results could also be achieved.

Example (Background Knowledge).

$$\mathcal{K} = \mathcal{K}_0 \cup \{A.p_1 = 1 \textit{ and } A.p_3 = 0\}$$

Using these definitions, we can now define the core concept of this article, which is the *maximum inference* given some knowledge $\mathcal{K}$.

Definition and Notation (Maximum Inference)

Maximum inference, given some background knowledge $\mathcal{K}$ is defined by :

$$I_{\mathcal{K}}(X) = \inf_{\leq}\{A \in \mathcal{B}^n \mid X \leq_{\mathcal{R}} A \text{ and } \mathcal{K} \text{ is true}\}$$

Maximum Inference of a tuple *X* means computing the set of $\mathcal{R}$-specializations in $\mathcal{B}^n$ of *X*, and taking its greatest lower bound. This represents the *unique* maximum consistent deduction possible in three values logic.

Example (Maximum Inference).

Consider $X_1 = (111???)$, $\mathcal{R}$ as defined above and $\mathcal{K}_0$. The set of all $\mathcal{R}$-specializations of X_1 in $\mathcal{B}^n$ is the following: {(111000), (111001), (111010), (111011)} and therefore $I_{\mathcal{K}_0}(X_1) = (1110??)$.

Maximum Inference is a good metric to measure confidentiality, since it corresponds to the form containing the least information from which no predicate value may be inferred, which is a definition often used in law [8, 15]. A naïve computation of maximum inference would be to proceed by brute force on the output *X* of an exposure reduction algorithm and test all individuals *A* such that $X \leq_{\mathcal{R}} A$, and find their lower bound.

Another way to measure confidentiality of the solution is to evaluate its Shanon entropy. This metric simply counts the number of different individuals leading to the form, and takes its logarithm.

Definition (Entropy Based Metric)

For $X \in \mathcal{T}^n$, we define the entropy based metric
$\mathcal{H}\langle\mathcal{R}\rangle(X) = log_2\#\{A \in \mathcal{B}^n \mid X \leq_{\mathcal{R}} A\}$

In general, this computation is exponential in the number of predicates. In this paper we focus on the computation of Maximum Inference only, since it is simpler. We show in Appendix C that it is possible to efficiently compute entropy for our new algorithm introduced in Section 4.

2.3. Problem statement

Comparing the maximum inference or the entropy based metric of the output of two data reduction algorithms are two *quantitative* and *sensible* measurement of the confidentiality gained by each algorithm by removing information. The objective in the rest of the paper revolves around computing these metrics. More precisely, we will investigate the following problems :

1. Identify and define $\mathcal{K}$ in the context of attacks on user data collected via application forms.
2. Find efficient algorithms to compute $I_{\mathcal{K}}(\alpha(A))$ for these types of $\mathcal{K}$.
3. Find a new family of exposure reduction algorithms defined by its own inferences and compare its exposure reduction with mainstream algorithms, using *maximum inference*.

3. COMPUTING MAXIMUM INFERENCE

Computing the maximum inference of a form *X*, given $\mathcal{R}$ through brute force has a complexity of 2^p where *p* is the number of unknown predicates in *X*. In this section, we characterize some situations where it is possible to improve this complexity, and propose new algorithms to compute maximum inference.

The three cases covered in this paper are minimal knowledge (attack 1, see Section 3.1), background knowledge on predicates (attack 2, see Section 3.2) and knowledge of the exposure reduction algorithm used (attack 3, see Section 3.3). They illustrate respectively the case of (i) a service provider, (ii) a service provider with multiple forms of a same user and (iii) a process in which the exposure reduction algorithm is public. Indeed, by definition the service provider is the one proposing the service, so it knows the rule set ($\mathcal{R}$) the axioms. If the service provider has more than one service (i.e. more than one rule set) or receives successive applications of the same individual, it can join information from both forms to infer additional information. The third hypothesis must also be studied since in certain scenarios, the implementation of the reduction algorithm is public, based on open source code (e.g., using open source solvers) or using very simple heuristics.

3.1 Minimal knowledge $\mathcal{K} = \mathcal{K}_0$

In this case, maximum inference is, by definition:

$I_{\mathcal{K}_0}(X) = \inf_{\leq}\{A \in \mathcal{B}^n \mid X \leq_{\mathcal{R}} A \text{ and } \mathcal{K}_0 \text{ is true}\}$.

Theorem 2 (Functional characterization of maximum inference using $\mathcal{K}_0$)

$I_{\mathcal{K}_0}(X) = \inf_{\leq}\{Y \in \mathcal{T}^n \mid \boldsymbol{\mathcal{F}}_X^-\langle\mathcal{R}\rangle(Y) = 1\ ;\ X \leq Y$ and $\mathcal{R}$ is in the DNF form $\}$

In order to prove this theorem, we introduce a simple lemma, stating the equality on the lower bounds of $\mathcal{B}^n$and $\mathcal{T}^n$ given a propositional formula.

Lemma 1

Given a propositional formula $\boldsymbol{\mathcal{F}}$ and $X \in \mathcal{T}^n$, the two following elements are equal:

$\inf\{A \in \mathcal{B}^n;\ X \leq A \text{ and } \boldsymbol{\mathcal{F}}(A) = 1\}$,
$\inf\{Y \in \mathcal{T}^n;\ X \leq Y \text{ and } \boldsymbol{\mathcal{F}}(Y) = 1\}$.

Proof (Lemma 1). Straightforward using Proposition 1. ■

Proof (Theorem 2) .

From Theorem 1(b), we can rewrite the definition of I_{K0} as follows:

$I_{\mathcal{K}_0}(X) = \inf_{\leq}\{A \in \mathcal{B}^n \mid \mathcal{F}_X^-\langle\mathcal{R}\rangle(A) = 1\ ;\ X \leq A$ and $\mathcal{R}$ is in the DNF form $\}$

Using Lemma 1, we can write:

$I_{\mathcal{K}_0}(X) = \inf_{\leq}\{Y \in \mathcal{T}^n \mid \mathcal{F}_X^-\langle\mathcal{R}\rangle(Y) = 1\ ;\ X \leq Y$ and $\mathcal{R}$ is in the DNF form $\}$ ■

This theorem shows that in order to compute $I_{\mathcal{K}_0}$ we can propose a smarter algorithm than brute force, since we can start from a form X and explore the lattice, stopping whenever we find a tuple such that $\mathcal{F}_X^-\langle\mathcal{R}\rangle(Y)$=1.

Algorithm 1 (Maximum inference)

***Inference* ( )**

Input: X in $\mathcal{T}^n$
Ouput: I in $\mathcal{T}^n$

1. $I \leftarrow \bot$;
2. if ($\mathcal{F}_X^-\langle\mathcal{R}\rangle(X) = 1$)
3. $I \leftarrow X$
4. return I;
5. else if ($\mathcal{F}_X^-\langle\mathcal{R}\rangle(X) = ?$)
6. $X_1 \leftarrow X$
7. $X_0 \leftarrow X_1$
8. Choose p such that $X.p = ?$
9. $X_0.p \leftarrow 0$
10. $X_1.p \leftarrow 1$
11. if ($\mathcal{F}_X^-\langle\mathcal{R}\rangle(X_0) \neq 0$)
12. $I \leftarrow$ Inference(X_0)
13. if ($\mathcal{F}_X^-\langle\mathcal{R}\rangle(X_1) \neq 0$)
14. $I \leftarrow inf($ I, Inference(X_1))
15. return I

Complexity. In the worst case, this algorithm has a complexity of 3^p, where p is the number of unknown predicates in X. In the worst case, this can be worse than the brute force attack of 2^n. We show next that we achieve even better results by exploiting the fact that rules are in DNF form. We start by considering that the rule set contains no negations (Section 3.1.1) to propose a polynomial algorithm to compute maximum inference, then we extend this to the case with negations by proposing an algorithm exponential in the number of predicates that appear in positive and negative form (Section 3.1.2).

3.1.1 No negations in $\mathcal{R}$: Fix-point attack

Let us next consider the case where $r \in U_{\mathcal{R}}(X)$ is written in *DNF* form as a disjunction of atoms and *contains no negation* : $r(Y) = 0 \Leftrightarrow \forall_{i\in[\![1;k]\!]} a_i(Y) = 0$ where $r = \bigvee_{i=1}^k a_i$. Note that $\mathcal{F}_X^-\langle\mathcal{R}\rangle(Y) = 1 \Leftrightarrow \forall r \in U_R(X), r(Y) = 0$ if $X \leq Y$, which can be done by computing the fixed point of a specific operator $\boldsymbol{\Psi}$ we will define next.

Let the universe of our models be $\mathcal{T}^n \times \{a; a$ is an atom in a rule of $\mathcal{R}\} \times \mathcal{R}$.

A model is an interpretation of rules, atoms and predicates. Rules can only be interpreted as *true* or *false* but atoms and predicates can also be *unknown*. For the sake of simplicity, we overload X to refer both to the model associated to X, $\mathcal{R}$ and atoms and to the form as in the previous section. In other words:

For a rule $\boldsymbol{r}$: $X.r = \begin{cases} 1; if\ \boldsymbol{r}(X) = 1 \\ 0\ otherwise \end{cases}$

For an atom a :

$$X.a = \begin{cases} 1; if\ \boldsymbol{a}(X) = 1 \\ 0; if\ \boldsymbol{a}(X) = 0\ or\ \exists r = a \bigvee a_i\ ,\ X.r = 0 \\ ?\ otherwise \end{cases}$$

If p is a predicate, we keep the same interpretation for $X.p$ as previously.

Let $\boldsymbol{\Psi}$ represent an inference operator on these models such that, for every model X, $\boldsymbol{\Psi}(X)$ is also a model:

$\boldsymbol{\Psi}(X).r = X.r$ and $\boldsymbol{\Psi}(X).a = X.a$

$$\boldsymbol{\Psi}(\mathrm{X}).p = \begin{cases} 1; if\ X.p = 1 \\ 0;\ if\ X.p = 0\ or\ \exists_{a=p\wedge\bigwedge_{i=1}^k p_i;\ X.a=0} \forall_{i\in[\![1;k]\!]} X.p_i = 1 \\ ?\,; otherwise \end{cases}$$

In other words, $\boldsymbol{\Psi}$ changes the value of an *unknown* predicate to *false* if it is the only unknown predicate in an atom that must be *false*, for which all other predicates are *true*.

In what follows, when using the $\leq$ and $\leq_{\mathcal{R}}$ comparators, we do not consider the rules and atoms, only the *form* (predicates) part of the model.

Proposition 3

Let X represent a form such that $\mathcal{F}_X\langle\mathcal{R}\rangle$ is satisfiable, and Y a form. The following inequalities hold:

(a) $X \leq_{\mathcal{R}} \boldsymbol{\Psi}(X)$

(b) $X \leq_{\mathcal{R}} Y \implies \boldsymbol{\Psi}(X) \leq_{\mathcal{R}} \boldsymbol{\Psi}(Y)$

Proof. Since $\boldsymbol{\Psi}$ can only transform ? values into 0 values, and since there is no negations in the rule set (a) is straightforward.

If $X \leq_{\mathcal{R}} Y$, then $X.a = 0 \Longrightarrow Y.a = 0$ and $X.p = 1 \Longrightarrow Y.p = 1$. Therefore any value changed from ? to 0 in $\boldsymbol{\Psi}(X)$ will also be changed from ? to 0 in $\boldsymbol{\Psi}(Y)$. In consequence, $\boldsymbol{\Psi}(X) \leq_{\mathcal{R}} \boldsymbol{\Psi}(Y)$. ■

Proposition 3 shows in particular that, given X, $(\boldsymbol{\Psi}^n(X))_n$ is an increasing sequence over $\mathcal{T}^n$. And since, $\mathcal{T}^n$ is finite the fixed point $\boldsymbol{\Psi}^\omega(X)$ of $\boldsymbol{\Psi}$ and is reached within a finite number of steps (in the worst case, n).

This is a simple reasoning on the generalization and the background knowledge. And since this knowledge is sound, all that we infer when infinitely applying this operator on X is coherent with all the solutions to $\mathcal{F}_X^-\langle\mathcal{R}\rangle$ that are bigger than X. Thus, by definition of the infimum, $\boldsymbol{\Psi}^\omega(X) \leq I_{\mathcal{K}_0}(X)$.

Theorem 3 (Maximum inference with no negations)

If $\mathcal{R}$ contains no negations, then $I_{\mathcal{K}_0}(X) = \boldsymbol{\Psi}^\omega(X)$

Proof

See Appendix A.

We now propose an algorithm based on the computation of $\boldsymbol{\Psi}$ to compute the maximum inference of any form X. The *simplify* side effect function eliminates atoms of S that are *false* and predicates that are *true*. $\boldsymbol{\Psi}$ is simply to choose one of the possible predicates and set it to *false*.

Algorithm 2 (Maximum inference on DNF/no negations)

***Inference* ( )**

Input: X in $\mathcal{T}^n$
Ouput: I in $\mathcal{T}^n$

```
1.  I ← X
2.  S ← atoms of rules in U_R(X)
3.  if ( S ≠ Ø )
4.     simplify ( S, X )
5.     while ( S ≠ Ø and one least elements of S is of size 1 )
6.        // PSI(X)
7.        p ← the corresponding predicate of that least element
8.        I.p ← 0
9.        simplify ( S, X )
10. return I

11. Function: simplify ( S, X )
12.    foreach a: atom of S
13.    if ( X.a ≠ ? )
14.       erase a from S
15.    else
16.       foreach p: predicate of a
17.          if ( X.p = 1 )
18.             erase p from a
```

Note : S is a partially ordered set of atoms in which the size of an atom is expressed by the number of predicates inside this atom. The smaller is this size, the smaller is the atom.

Complexity

Since at each step we infer at least one predicate, the complexity of the algorithm is linear in the number of unknown predicates of X (which is less than n, the dimension of the lattice) times the complexity of the decision in each step (which can be considered linear in n_a, the number of atoms in rules in $U_{\mathcal{R}}(X)$ which is bounded by the total number of atoms a. In general (see [3]), we have $n_a \propto n$. Therefore the complexity of this algorithm is $O(n^2)$ in the worst case.

Use. Since it is fast, this algorithm can be included in any exposure reduction algorithm to construct the maximum inference of the proposed solution. The exposure function used should then be applied to $I_{\mathcal{K}_0}(\alpha(A))$ instead of $\alpha(A)$, as we discuss in Section 4.1.

Example

We use the same notations as the example in Section 2.2. Consider an individual $A = (110010)$. Suppose some reduction algorithm $\alpha(A)$ produces the following result; $X = \alpha(A) = (11????)$. We can apply Th. 3.

Recall that the atoms are $a_1 = (p_1 \wedge p_2)$; $a_2 = p_3$; $a_3 = (p_1 \wedge p_4)$; $a_4 = (p_5 \wedge p_6)$. The model associated to X is: $X = (p(11????); a(1?00); r(10))$.

The fixed point $\boldsymbol{\Psi}^2(X) = \boldsymbol{\Psi}(X) = \{p(11?0??); a(1?00); r(10)\}$ is therefore the maximum inference possible under $\mathcal{K}_0$ for a rule set with no negations.

3.1.2 General case: rules with negations

In the general case, we can transform the operator the take into account negations:

$$\widetilde{\boldsymbol{\Psi}}(\mathrm{X}).p = \begin{cases} 1; if\ X.p = 1\ or\ \exists a = \neg p \wedge \boldsymbol{rest}\ and\ X.a = 0\ such\ that \\ \qquad \boldsymbol{rest}(X) = 1 \\ 0;\ if\ X.p = 0\ or\ \exists a = p \wedge \boldsymbol{rest}\ and\ X.a\ such\ that \\ \qquad \boldsymbol{rest}(X) = 1 \\ ?; otherwise \end{cases}$$

In this case, Theorem 3 no longer holds, but we still have $\widetilde{\Psi}^{\omega}(X) \leq I_{\mathcal{K}_0}(X)$. To compute maximum inference, a simple idea is to use a recursive algorithm combined with the previous approach.

Algorithm 3 (Maximum inference on DNF)

***Inference* ( )**

Input: X in $\mathcal{T}^n$
Ouput: I in $\mathcal{T}^n$

```
1.  I ← X
2.  S ← atoms of U_R(X)
3.  if ( S ≠ Ø )
4.     simplification ( S, X )
5.     while ( S ≠ Ø and one least elements of S is of size 1 )
6.        p ← the predicate of that least element // p can be a negation or
       not
7.        if ( p is not a negation )
8.           I.p ← 0;
9.        else // p is a negation
10.          I.p ← 1;
11.       if ( simplification( S, X ) = false )
12.          return ⊥; // inconsistent
13.    S2 ← set of all conflicting predicates;
14.    if ( S2 ≠ Ø )
15.       p ← element of S2
16.       X2 ← I
17.       X1 ← I
18.       X1.p ← 1
19.       X2.p ← 0
20.       I ← inf ( Inference(X1), Inference(X2) ); // infimum
21. return I

22. Function: simplification ( S, X )
23.    foreach a: atom of S
24.    if ( X.a = 0 )
25.       erase a from S
26.    else if ( X.a = 1 )
27.       return false
28.    else
29.       foreach p: predicate of a // can be a negation or not
30.          if ( ( p is not a negation and X.p = 1 )
31.                or ( p is a negation and X.p=0 ) )
32.             erase p from a
33. return true
```

The recursion decreases the number of conflicts (the presence of a predicate and its opposite in the reduced equations) and the reasoning is complete (every solution is kept in some way). Conversely, contradictions are not kept in the computation of the infimum since $inf(\perp, X) = X$.

Complexity

In the worst case, the recursion is simple, and the complexity is clearly exponential. Take for example these

null atoms: $a_{2k} = p_{2k} \wedge p_{2k+1}$ and $a_{2k+1} = p_{2k} \wedge \neg p_{2k+1}$ with $k \in [\![1; q]\!]$. The complexity is at least 2^qwhich is exponential in the number of unknown predicates (usually less than n though). Each individual reasoning is $O(1)$ for an overall cost of $O(2^n)$ in the worst case.

3.2 $\mathcal{K}$ includes information on some predicates

If we consider the case where some information on the truth value of an individual's predicates has been leaked (e.g. logical formulas or exact values), we can enrich our rule models' universe with new atoms and generate their interpretations in the model associated with *X*. The maximum inference is the output of the above algorithm with the new universe. All reasoning is conducted similarly as in Section 3.1, both theorems obviously can be applied, and any algorithm useable for the previous case will also work here.

3.2.1. Statistical attacks

We are not interested in statistical attacks on the individuals in this paper, but inferring more information based on a database of exposure reductions, integrating them using the same attack formalism by including the probabilistic rules in the background knowledge, and performing probabilistic reasoning remains future work.

3.2.2. Random Algorithms

Random algorithms provide an interesting approach, since they are fast and retrieve reasonably good results, but a bad side effect is that if the same random algorithm is run multiple times for the same individual, we will disclose more information than expected (since background knowledge is going to increase each run). Conversely if we take into consideration the information disclosed by each execution as background knowledge of the random algorithm, this problem can easily be avoided.

3.3 Case3: The algorithm is deterministic and fully known

In general, it is realistic to consider that the algorithm used to perform exposure reduction is known. Different attacks can be conducted, based on the characteristics of these algorithms, for instance, minimality [17] or how the algorithm is applied in the case of deterministic heuristics. This can be modeled by adding background knowledge in the form of logical expressions that use atoms.

Definition (Algorithm Based Attack)

An algorithm based attack can be conducted if, using the knowledge of the algorithm, it is possible to add logical expressions (of any order) on *rules*, *atoms* and *predicates* to the background knowledge.

Obviously efficient results will be achieved when it is possible to write 0-order expressions.

Example.

Consider the HME algorithm which is a simplified version of an exposure reduction algorithm proposed in the context of social care offered by French General Councils to reduce users' application forms [3]. It is a greedy algorithm that processes each rule one by one and selects the smallest atom in terms of predicates. Its code is given in Appendix B.

Consider the following rule set :

$$\boldsymbol{r}_1 = (p_1 \wedge p_2) \vee p_3 = \boldsymbol{a}_1 \vee \boldsymbol{a}_2$$
$$\boldsymbol{r}_2 = (p_1 \wedge p_4) \vee (p_5 \wedge p_6 \wedge p_7) = \boldsymbol{a}_3 \vee \boldsymbol{a}_4$$

Since the algorithm selects the smallest atom, the background knowledge added is :

$$\mathcal{K} = \mathcal{K}_0 \cup \{\, a_1 \Longrightarrow \neg a_2, a_4 \Longrightarrow \neg a_3\}$$

Let $A = (1100111)$, *HME(A)* $= (11??111)$

By executing algorithm 2 extended to consider atoms, we deduce that a_2=*false*, a_3=*false* and therefore A=(1100111).

A *good* algorithm must therefore either (i) not let the attacker write *any* logical formula, or (ii) ensure that the number of satisfiable models is sufficiently large (exponential). Case (i) provides complete protection i.e. only the basic $\mathcal{K}_0$ can be used. Case (ii) provides computational protection. Obviously, a good algorithm must also be able to evaluate its privacy guarantees for an attack using $\mathcal{K}_0$.

4. TOWARDS NEW EXPOSURE REDUCTION ALGORITHMS

In this section, we propose two directions to improve the quality of minimum exposure algorithms, while remaining tractable. The first direction (a) is to use existing exposure reduction algorithms, but to include inference in the computation of the exposure metric. This approach involves computing the maximal inference, and can therefore be costly. The second direction (b) is to directly compute reduced forms from which nothing more can be inferred. We propose a genetic algorithm that can produce its first results quickly and improve them if given more processing time. We compare their performance in Section 6.

4.1. Adapting Existing Exposure Reduction Algorithms

We have shown in Section 3 that it is possible to compute maximal inference in some common cases. We can now quantify by a more realistic value the data exposed by algorithms. In previous work, this quantity was a comparison of the exposures of the individual A and its reduced exposure, $\alpha(A)$. The exposure is computed using an exposure function $\mathbb{E}$, and exposure gain is computed by $\mathcal{G}(A, \alpha(A)) = 1 - \frac{\mathbb{E}(\alpha(A))}{\mathbb{E}(A)}$ (the greater the gain, the better). We argue that since maximum inference can be computed, the following formula for exposure gain should be used given some background knowledge $\mathcal{K}$:

$$\mathcal{G}_{\mathcal{K}}(A, \alpha(A)) = 1 - \frac{\mathbb{E}(I_{\mathcal{K}}(\alpha(A)))}{\mathbb{E}(I_{\mathcal{K}}(A))} = 1 - \frac{\mathbb{E}(I_{\mathcal{K}}(\alpha(A)))}{\mathbb{E}(A)}$$

Previous, existing algorithms optimize $\mathcal{G}$, thus trying to minimize $\mathbb{E}$ over the set of all $\mathcal{R}$-generalizations of A.

Taking $\mathcal{G}_{\mathcal{K}}$ into account instead of $\mathcal{G}$, would intuitively lead to minimize $\mathbb{E}(I_{\mathcal{K}})$. The problem is that $\mathbb{E}(I_{\mathcal{K}})$ is not an exposure function : it cannot be computed on all the elements of the lattice. However, $\mathbb{E}(I_{\mathcal{K}_0})$ does have the same behavior as an exposure function, if restricted to the set of $\mathcal{R}$-generalizations of A. Therefore, existing algorithms can optimize: $\mathcal{G}_{\mathcal{K}_0}(A, \alpha(A)) = 1 - \frac{\mathbb{E}(I_{\mathcal{K}_0}(\alpha(A)))}{\mathbb{E}(A)}$. This is interesting in practice, because the solution obtained is the best solution for existing algorithms (i.e. no modification other than changing the computation of exposure gain is necessary, which is a plug in of most algorithms) that protects against attacks 1 and 2. Moreover, in many practical cases, the algorithms used are random, and are by nature protected against attack 3 (ex: the RAND* algorithm implanted in smartcards used by General Council presented in [3]). In consequence, since the use of maximum inference provides a better quantification of the data leaked by algorithms (around 30% for real cases as shown in Section 6) and that its integration is seamless, all existing algorithms will benefit from using it.

However, this approach offers no additional protection (it is neither better nor worse) against attack 3, which can be conducted against many algorithms including deterministic and minimal algorithms (e.g., heuristic algorithms like HME, or exact resolutions using a solver).

4.2. Providing resilience to inference by construction

In order to provide resilience against attack 3, we propose a new algorithm, termed *Inference Resistant (InfRes) Algorithm*. This algorithm provides solutions that are equal to their maximum inference. It is therefore resistant to attacks 1, 2 and 3 *by construction*.

Algorithm intuition

Some simple but important results on $\mathcal{F}_X\langle\mathcal{R}\rangle$ are '*If* $\mathcal{F}_X\langle\mathcal{R}\rangle(Y) = 1$ *then* $I_{\mathcal{K}_0}(Y) = Y$' and '*If* $X \leq_{\mathcal{R}} Y$ *then* $\mathcal{F}_X\langle\mathcal{R}\rangle = \mathcal{F}_Y\langle\mathcal{R}\rangle$'. That means that if we start from a given individual A and chose a $\mathcal{R}$-generalization that satisfies $\mathcal{F}_A\langle\mathcal{R}\rangle$ then the attacker cannot infer anything more than the information already revealed by the $\mathcal{R}$-generalization. In order to improve the quality (exposure) of the solution proposed, we can progress by computing the lower bound of several points that satisfy $\mathcal{F}_A\langle\mathcal{R}\rangle$, until we find a point X for which $\mathcal{F}_A\langle\mathcal{R}\rangle(X) =?$. The more points we compute, the smaller their infimum will be. A first idea is therefore to build the "best" points such that $\mathcal{F}_A\langle\mathcal{R}\rangle = 1$ and to take their infimum X. However it is crucial that at least one point used be *not* a $\mathcal{R}$-generalization of A. Indeed, if this were the case, the following (albeit very costly) attack would be possible : given X, find all points M such that $X \leq_{\mathcal{R}} M$. In turn, $B = Sup(M) \geq_{\mathcal{R}} X$ provides a better approximation of A for the attacker. Obviously, to be resistant to attack 3, it would be possible to simply return *a single* point such that $\mathcal{F}_A\langle\mathcal{R}\rangle = 1$, but its exposure reduction would clearly be suboptimal.

This leads us to state two propositions that will be used by a genetic algorithm, that we call *InfRes* algorithm which will compute "good" points such that $\mathcal{F}_A\langle\mathcal{R}\rangle = 1$, while assuring that these points *could be* generated by elements Y that *do not* $\mathcal{R}$-generalize A and compute their infimum, which will be equal to its own maximum inference.

Proposition 4

1. Given A an individual, let J be a subset of forms that satisfy $\mathcal{F}_A\langle\mathcal{R}\rangle$. If $Inf(J) \leq_{\mathcal{R}} A$ then $I_{\mathcal{K}_0}(Inf(J)) = Inf(J)$.
2. Let X be a $\mathcal{R}$-generalization of A and Y a form such that $X \leq_{\mathcal{R}} Y$. If $\mathcal{F}_A\langle\mathcal{R}\rangle(Y) = 1$ then $I_{\mathcal{K}_0}(X) \leq Y$.

Proof. (1) If $Inf(J) \leq_{\mathcal{R}} A$ means that $I_{\mathcal{K}_0}(Inf(J)) \leq Y$ for every $Y \in \mathcal{T}^n$ such that $\mathcal{F}_A\langle\mathcal{R}\rangle(Y) = 1$.

And by definition of J, the following holds :

$\forall Y \in J, I_{\mathcal{K}_0}(Inf(J)) \leq Y$. Thus $I_{\mathcal{K}_0}(Inf(J)) \leq Inf(J)$.

(2) From (1) and since *inf*(*A*)=*A* we have $I_{\mathcal{K}_0}(Y) = Y$ if $\mathcal{F}_A\langle\mathcal{R}\rangle(Y) = 1$. It's easy to see that $I_{\mathcal{K}_0}$ is monotonically increasing in $(\mathcal{T}^n, \leq_{\mathcal{R}})$. Thus $I_{\mathcal{K}_0}(X) \leq Y$.■

The *InfRes* algorithm

Proposition 4 gives a method to compute forms that will be resistant to attacks 1 and 2. This property needs to be combined with an algorithm introducing (ideally randomly) some points which are not $\mathcal{R}$-generalizations of the individual, in order to be resistant to attack 3. In what follows, we propose a genetic algorithm with these properties.

Definition (Population)

A population is defined inductively as:

1. A set of forms that satisfy $\mathcal{F}_A\langle\mathcal{R}\rangle$;
2. The infimum of a population if this infimum is a $\mathcal{R}$-generalization of *A*;
3. A union of populations .

Every genetic algorithm has 6 main notions: the population, the initialization step, crossover, selection, mutation and the fitness function. We will now define those notions that fully define the algorithm.

Initialization: start with an individual A and randomly generate a population of forms that satisfy $\mathcal{F}_A\langle\mathcal{R}\rangle$.

Crossover: As suggested by Proposition 4, the most appropriate crossover of two elements is their infimum. In order to remain equal to their maximum inference, we only build the infimum of two elements that both satisfy $\mathcal{F}_A\langle\mathcal{R}\rangle$.

Selection: First of all, we eliminate elements that do not $\mathcal{R}$ -generalize *A*. In order to maintain constant population size, we then perform a tournament selection.

Mutation: the basic idea behind mutation is that it must introduce forms that are not $\mathcal{R}$-generalizations of A. We simply introduce new elements like in the initialization phase, while keeping the size of the population constant. While the randomness does not guarantee that we introduce elements that do not not $\mathcal{R}$-generalize A, it is a possibility, therefore this forbids any inference by the attacker using the sub-lattice of $\mathcal{R}$-generalizations.

Fitness function: the fitness function (or the function to optimize) is $\mathbb{E} \circ I_{\mathcal{K}_0}$.

The output of the algorithm is ideally the form that minimizes the exposure $\mathbb{E} \circ I_{\mathcal{K}_0}$ among all solutions constructed using all elements of the population in the i-th iteration. This can be costly. However, given Proposition 4, by using only pairwise combinations of elements of the population, $\mathbb{E} \circ I_{\mathcal{K}_0} = \mathbb{E}$, and therefore no inference needs to be conducted.

The *InfRes* algorithm
Create population For(n_step<Max_iterations) Random Crossovers on the population Perform a simple selection to reduce the output of the crossover step Merge the current population with the crossovers and perform tournaments selection Add mutant to the population Return best candidate(population,A)
Best candidate: The element that minimizes $\mathbb{E} \circ I_{\mathcal{K}_0}(X)$

Section 6 shows that this algorithm, using pairwise combinations to compute the final candidate, provides better exposure than existing algorithms, runs within seconds, and grants protection against attacks 1, 2 and 3 by construction.

We show in Appendix C that it is also possible to compute the entropy of the proposed solution.

5. RELATED WORKS

5.1. Useful Concepts

This article is based on the concept of Minimum Exposure introduced in [4], which is a technique to implement Limited Data Collection, a world know privacy principle [8, 15]. Our approach using the computation of maximum inference via a fix-point is inspired by the computation of stable models in logic programming using 3-values [10, 11, 13, 14, 16]. However the main difference in semantics is that in logic programming negation derives from failure to deduce a true value, and that the order used in the bilattice truth/knowledge is the *truth* axis. Our approach uses the *knowledge* axis, and exploits the fact that if some benefits are *false* then the premises are also *false*. Our results therefore cannot be directly derived from 3-valued logic programming.

5.2. Similar Approaches

Privacy-preserving data management has been a hot topic for over ten years. Several works in this field are close to our proposal in this article, in particular privacy-preserving data publishing (see [7, 9] for a survey on the topic of PPDP). However, PPDP is different from the Limited Data Collection approach. Indeed PPDP seeks to *publish* data for other purposes (e.g. data mining), and achieves privacy through reduction of data utility (e.g. through generalization/suppression, or introduction of noise). PPDP can be seen as *setting* a given level of privacy, and by trying to *maximize* data utility. On the contrary, the essence of LDC is to *maintain* utility, and *maximize* privacy (in the current case, via generalization/suppression). PPDP and LDC are therefore orthogonal. This is why concepts such as *minimality attack* introduced in [17] are not tailored for LDC, and cannot be applied. Other interesting works appeared in privacy preserving data mining (see [2] for a survey), and propose attacks to infer information from data published [1, 12], however, the models used are those of data mining (e.g. association rules) and are probabilistic, and are not really interested background knowledge. The techniques involved are therefore completely different.

6. EXPERIMENTAL RESULTS

In this section, we present an experimental validation of our approach on real and synthetic data. All the experiments were conducted on a HP workstation with 3.1GHz Intel CPU and 8GB RAM. Our algorithms are implemented in C language (link to the code: http://project.inria.fr/minexp/).

The real datasets used are GEVA application forms and the corresponding set of collection rules previously built by experts as provided by the General Council of Yvelines District in France. We have also used a synthetic data generator to simulate smaller and larger forms and rules using a similar rule set topology (same number of atoms per collection rules, same number of atoms per predicate). On our graphs, the point corresponding to n=440 corresponds to the results observed using the real GEVA application forms.

For all the experiments we have chosen a set of 10000 application forms for which 50% of the collection rules are satisfied. The plots are obtained by computing the average result obtained for these forms. *HME* and *RAND* as proposed in [3] were chosen as representatives of existing algorithms. *RAND* picks a true atom in each true collection rule and is only sensitive to attacks 1 and 2. *HME* is based on an ad-hoc heuristic, and is thus sensitive to attacks 1, 2 and 3. In the experiments, we consider background knowledge $\mathcal{K}_0$ and in the case of HME, the knowledge of the deterministic algorithm heuristic.

Our first measurements pictured in Fig. 5 plot (for *HME* and *RAND*) the ratio of the average number of predicates discovered by inference divided by the average number of predicates exposed in the reduced output form (on the left), and the gain $\mathcal{G}_{\mathcal{K}}$ of the *HME*, *RAND* and *InfRes* algorithms (on the right). On the left curve, we see that the ratio of predicates obtained using inference is rather high (around

20% on average). It shows also that with HME, the inference ratio is higher than with RAND. This is due to a larger background knowledge, since with HME the knowledge of the algorithm can be used to infer more elements, using attack 3 (which is not the case with RAND). The right curve shows that the *InfRes* algorithm converges to a solution which is better than those produced by the algorithms *HME* and *RAND*. We can also see that the exposure gain $\mathcal{G}_{\mathcal{K}}$ obtained with *HME* or *RAND* is good since over half of the predicates remain hidden while still satisfying all the possible benefits for the applicant, and is best with *InfRes*. In previous experiments (see [4]) $\mathcal{G}$ was computed and not $\mathcal{G}_{\mathcal{K}}$. In this context, HME clearly outperformed RAND. When taking inference into account, HME and RAND produce similar results in terms of exposure reduction. All this demonstrates the importance of taking inference into account, and its interest in practice.

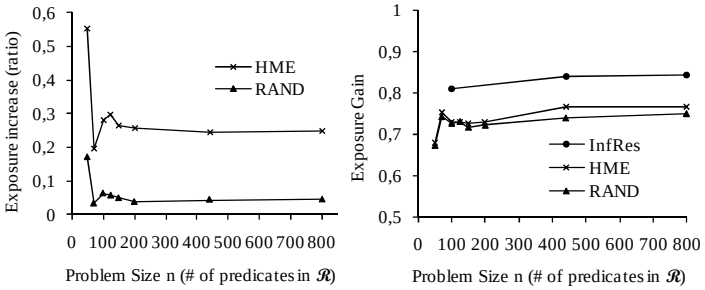

Figure 5. Exposure analysis of reduction algorithms.

Figure 6 shows the performance of the *InfRes* algorithm in function of the execution time allocated to it. The population used to initialize the algorithm is of size 10000 (random individuals were added to the real GEVA dataset). This algorithm is fast, since the solution can be obtained in a few seconds, which is interesting in practice.

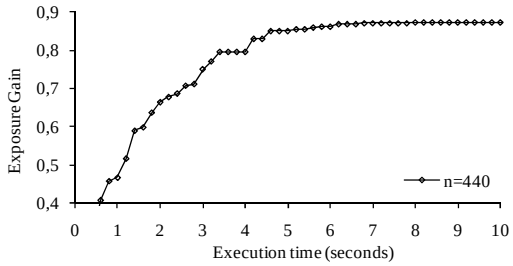

Figure 6. Exposure of *InfRes* algorithm.

7. CONCLUSION

Removing from users' application forms the personal data items which are not strictly useful for its subsequent evaluation by a service provider is imposed by privacy laws enacted worldwide, and is useful for both service providers and users. We have shown that data items removed by form reduction processes can be inferred in different manners: by exploiting unrequested benefits (attack 1), using sets of previously disclosed data items (attack 2) and/or by the nature of the reduction process itself (attack 3). We have formalized the data reduction problem and have proposed algorithms to compute inference and to perform data reduction resistant to inference attacks. We have validated through experiments the relevance of the inference problem and the efficiency of our algorithms. In our future works, we plan to adapt this work to entropy based metrics, and study statistical attacks.

8. ACKNOWLEDGEMENTS

This work is partially supported by KISS ANR-11-INSE-005 and INRIA CAPPRIS grants.

9. REFERENCES

[1] Aggarwal, C., C., Pei, J. and Zhang, B. On privacy preservation against adversarial data mining. In *ACM SIGKDD*, 2006.

[2] Aggarwal, C., C., Yu, P., S. A general survey of privacy-preserving data mining models and algorithms. *Advances in Database Systems*, 2008.

[3] Anciaux, N., Bezza, W., Nguyen, B., and Vazirgiannis, M. MinExp-card: limiting data collection using a smart card. In *EDBT*, 2013.

[4] Anciaux, N., Nguyen, B., and Vazirgiannis, M. Limiting data collection in application forms : A real case application of a founding privacy principle. In *IEEE PST*, 2012.

[5] Baesens, B., Setiono, R., Mues, C., and Vanthienen, J. Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science*, 49(3), 2003.

[6] Calders, T., and Goethals, B. Quick inclusion-exclusion. In *KDID*, 2005.

[7] Chen, B.-C., Kifer, D., LeFevre, K., and Machanavajjhala, A. Privacy-Preserving Data Publishing. *Found. Trends databases* 2, 1–2 (Jan. 2009), 1-167, 2009.

[8] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data. *Official Journal of the EC, 23*, 1995.

[9] Fung, B., C., M., Wang, K., Chen, R., and Yu, P., S. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.* 42(4), 53 pages, 2010.

[10] Gelfond, M. and Lifschitz, V. The stable model semantics for logic programming. In *Int. Conf. on Logic Programming*. R. A. Kowalski and K. Bowen, Eds. The MIT Press, Cambridge, Massachusetts, p1070–1080, 1988.

[11] Gelfond, M. and Lifschitz, V. Classical negation in logic programs and disjunctive databases. *New Generation Computing 9*, 3/4, p365–386, 1991.

[12] Kifer, D. Attacks on privacy and deFinetti's theorem. In *ACM SIGMOD*, 2009.

[13] Loyer, Y., Straccia, U. Any-world assumptions in logic programming. *Theoretical Computer Science*, 342(2–3), p351-381, 2005.

[14] Loyer, Y., and Straccia, U. Epistemic foundation of stable model semantics. *Theory Pract. Log. Program.*, 6(4), p355-393, 2006.

[15] OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data, 23rd Sept. 1980.

[16] Przymusinski, T. Well-founded semantics coincides with three-valued stable semantics. *Fundam. Inf.* 13(4), p445-463, 1990.

[17] Wong, R. C. W., Fu, A. W. C., Wang, K., and Pei, J. Minimality attack in privacy preserving data publishing. In *VLDB*, 2007.

APPENDIX A - PROOFS

Proof of Theorem 3 (Maximum inference with no negations)

We have $\forall_a \exists_{J_a \subseteq [\![1;n]\!]} a = \bigwedge_{i \in J_a} p_i$

Let $Y = \Psi^\omega(X)$ and $\Lambda = \{a | \, Y.a = 0 \text{ and } a(Y) = ?\}$.

The present proof make use of the two following remarks and lemma:

(*) $\begin{cases} a(Y) = ? \\ Y.a = 0 \end{cases} \Longrightarrow (\exists_{i,j \in J_a} i \neq j \text{ and } Y.p_i = Y.p_j = ?)$

(**) $\begin{cases} Y \leq A \\ \forall_{a \in \Lambda} a(A) = 0 \end{cases} \Rightarrow \begin{cases} Y \leq A \\ \mathcal{F}_Y^-\langle \mathcal{R} \rangle(A) = 1 \end{cases} \Rightarrow Y \leq_{\mathcal{R}} A$

Remark (*) is true since $\underline{Y}$ is a fixed point of Ψ. (**) results from the characterization of the $\mathcal{R}$-generalization.

Lemma 2

Let $\Omega \subseteq \{A \in \mathcal{B}^n | X \leq_{\mathcal{R}} A\}$. The following holds: $\inf(\Omega) = \mathrm{Y} \Longrightarrow I_{\mathcal{K}_0}(X) = Y$

Proof of lemma 2

Since $X \leq_{\mathcal{R}} \Psi^\omega(X)$ and $\Psi^\omega(X) \leq_{\mathcal{R}} I_{\mathcal{K}_0}(X)$ we have the following:

$I_{\mathcal{K}_0}(X) \leq_{\mathcal{R}} I_{\mathcal{K}_0}(\Psi^\omega(X)) \leq_{\mathcal{R}} I_{\mathcal{K}_0}(I_{\mathcal{K}_0}(X))$ because $I_{\mathcal{K}_0}$ is monotonically increasing in $(\mathcal{T}^n, \leq_{\mathcal{R}})$.

On the other hand, $I_{\mathcal{K}_0}$ is idempotent so:

$I_{\mathcal{K}_0}(X) \leq_{\mathcal{R}} I_{\mathcal{K}_0}\big(\Psi^\omega(X)\big) \leq_{\mathcal{R}} I_{\mathcal{K}_0}\Big(I_{\mathcal{K}_0}(X)\Big) = I_{\mathcal{K}_0}(X)$

Then $I_{\mathcal{K}_0}(X) = I_{\mathcal{K}_0}\big(\Psi^\omega(X)\big)$. ■

The idea behind the proof is to exhibit some individuals that can be $\mathcal{R}$-generalized into Y such that the infimum of these solutions is exactly Y.

Case1: $\Lambda = \emptyset$

if $\forall_a Y.a = 0 \Longrightarrow a(Y) = 0$

In this case, we have $\mathcal{F}_Y\langle \mathcal{R} \rangle(Y) = 1$ then: $\forall_{A \in \mathcal{B}^n} Y \leq_{\mathcal{R}} A \Leftrightarrow Y \leq A$.

We choose $\Omega = \{A \in \mathcal{B}^n | Y \leq A\} = \{A \in \mathcal{B}^n | Y \leq_{\mathcal{R}} A\}$

So we have $\begin{cases} \inf(\Omega) = \inf\{A \in \mathcal{B}^n | Y \leq A\} = Y \\ \Omega \subseteq \{A \in \mathcal{B}^n | X \leq_{\mathcal{R}} A\} \end{cases}$

So $I_{\mathcal{K}_0}(X) = Y$.

Case2: otherwise ($\Lambda \neq \emptyset$)

Let $i \in [\![1; n]\!]$ such that $Y.p_i = ?$.

We introduce two individuals A_i^0 and A_i^1 such that:

$\forall_{j \in [\![1;n]\!]} j \neq i \Longrightarrow A_i^0.p_j = A_i^1.p_j = \begin{cases} Y.p_j; \; if \; Y.p_j \neq ? \\ 0; otherwise \end{cases}$

and $\begin{cases} A_i^0.p_i = 0 \\ A_i^1.p_i = 1 \end{cases}$.

It's easy to see that $Y \leq A_i^0$ and $Y \leq A_i^1$.

On the other hand, let $a \in \Lambda$.

If $i \in J_a$, from remark (*) $\exists_{j \in J_a} j \neq i$ and $Y.p_j = ?$

Otherwise, since $a \in \Lambda$ and $i \notin J_a$ then $\exists_{j \in J_a} j \neq i$ such that $Y.p_j = ?$. So in both cases such j always exists.

Then $a(A_i^0) = a(A_i^1) = 0$ since $A_i^0.p_j = A_i^1.p_j = 0$ (by definition of A_i^0 and A_i^1).

Thus from remark (**) $\begin{cases} Y \leq_{\mathcal{R}} A_i^0 \\ Y \leq_{\mathcal{R}} A_i^1 \end{cases}$.

The choice of i is arbitrary and $A_i^0.p_i = 0$ $A_i^1.p_i = 1$, the infimum of those exhibited solutions pairwise are forms such that the predicate p_i has the unknown value:

That means if we let $\Omega = \bigcup_{\substack{i \in [\![1;n]\!] \\ Y.p_i = ?}} \{A_i^0, A_i^1\}$ then $\Omega \subseteq \{A \in \mathcal{B}^n | X \leq_{\mathcal{R}} A\}$ and $\inf(\Omega) = \mathrm{Y}$.

Thus $I_{\mathcal{K}_0}(X) = Y$. ■

APPENDIX B - ALGORITHM CODE

Algorithm 4 (Simplified HME)

Input: A an individual

```
HME(A)
{
  X=(??...?);
  foreach(r: rule in T_R(A))
    if(r(X)==?)
      a_least=first element of r;
      foreach(a: atom in r)
        if(a.length<a_least.length)
          a_least=a;
      foreach(p: predicate in a_least)
        X.p=1;
  return X;
}
```

APPENDIX C - ENTROPY

In this article, we have used maximum inference to quantify exposure increase. Entropy, noted $\mathcal{H}\langle \mathcal{R} \rangle(X)$, as defined in Section 2, is a more precise metric, but is in general much more expensive to compute. We show that for specific cases of forms respecting Proposition 4, it can be computed much faster.

Proposition C$_1$.

Let $\mathcal{R}$ be a rule set, and M a form such that $\mathcal{F}_M^-\langle \mathcal{R} \rangle(M) = 1$. Let q represent the number of unknown values in M, then $\mathcal{H}\langle \mathcal{R} \rangle(M) = q$.

Proof. By construction, all elements $A \in \mathcal{B}^n$ such that $M \leq A$ are such that $M \leq_{\mathcal{R}} A$ since $\mathcal{F}_M^-\langle\mathcal{R}\rangle(M) = 1$ and their cardinality is 2^q. ■

Proposition C_2.

Let $\mathcal{M} = \{M_i; i \in [\![1;k]\!]\}$ such that $\mathcal{F}_A\langle\mathcal{R}\rangle(M_i) = 1$. $\forall i \in [\![1;k]\!]$ let $\Omega_i = \{A' \in \mathcal{B}^n \mid M_i \leq_{\mathcal{R}} A'\}$|. Let $X = inf\ (\mathcal{M})$), if $X \leq_{\mathcal{R}} A$ then $\mathcal{H}\langle\mathcal{R}\rangle(X) \geq log_2(\#\cup_{i=1}^k \Omega_i)$ and this lower bound can be computed in $O(2^{\#M})$ in the worst case using the inclusion-exclusion principle [6] or approximated using Bonferoni inequalities.

Proof.

1. Let $A \in \bigcup_{i=1}^k \Omega_i$ then $\exists i$, $M_i \leq_{\mathcal{R}} A$. Thus $X \leq_{\mathcal{R}} A$.

$$\bigcup_{i=1}^{k} \Omega_i \subseteq \{A \in \mathcal{B}^n;\ X \leq_{\mathcal{R}} A\}$$

2. The inclusion-Exclusion principle (given as follows) :

$$\#\bigcup_{i\in I} \Omega_i = \sum_{J\subseteq I} (-1)^{\#J}.\#\bigcap_{i\in J} \Omega_i$$

$$\#\bigcup_{i\in I} \Omega_i = \sum_{k=1}^{\#I} (-1)^{k+1}.\Big(\sum_{1\leq i_1<\cdots<i_k\leq \#I} \#\bigcap_{j=1}^{k} \Omega_{i_j}\Big)$$

is a formula to compute the cardinality of the union of a finite number of sets. Different algorithms exist to compute it ([6]) and are exponential in the number of sets (#*M* in our proposition). However in our case, we also need to be able to compute the intersection sets for all elements of *P*(*M*). We show next that the intersection can be computed in constant time.

We have:

$$\#\bigcap_{j=1}^{k} \Omega_{i_j} = \#\inf\{Y \in \mathcal{T}^n \mid \forall j \in [\![1;k]\!] M_{i_j} \leq Y\ ;\ \mathcal{F}_{M_{i_j}}\langle\mathcal{R}\rangle(Y) = 1\}$$

And since $\exists i, M_i \leq Y \Longrightarrow \mathcal{F}_{M_i}\langle\mathcal{R}\rangle(Y) = 1$:

$$\#\bigcap_{j=1}^{k} \Omega_{i_j} = \#\inf\{Y \in \mathcal{T}^n \mid \forall j \in [\![1;k]\!] M_{i_j} \leq Y\}$$

The right side is exactly the size of the least element of $\{M_{i_j};\ j \in [\![1;k]\!]\}$.

Thus $\#\bigcap_{j=1}^k \Omega_{i_j} = \#sup\{M_{i_j};\ j \in [\![1;k]\!]\}$. ■

Given M_1 and M_2, $\sup(M_1, M_2)$ can thus be computed by :

$\sup(M_1, M_2).p_i$=1 if $M_1.p_i$=1 and ($M_2.p_i$=1 or $M_2.p_{i}$=?) or $M_2.p_i$=1 and ($M_1.p_i$=1 or $M_1.p_i$=?))

$\sup(M_1, M_2).p_i$=0 if $M_1.p_i$=0 and ($M_2.p_i$=0 or $M_2.p_{i}$=?) or $M_2.p_i$=0 and ($M_1.p_i$=0 or $M_1.p_i$=?))

$\sup(M_1, M_2).p_i$=? if $M_1.p_i$=? and $M_2.p_{i}$=?).

Note : If $\exists i$ such that $M_1.p_i = 1\ and\ M_2.p_i = 0$ (or the contrary) then $\sup(M_1, M_2) = \perp$ and in consequence $\#(\Omega_1 \cap \Omega_2)$=0.

Example.

Consider A=(111000). $\mathcal{M}$={(11?00?), (11?0?0), (0?1?0?), (0?1??0), (??100?), (??10?0)}, and therefore A can be protected by publishing any of these solutions, in which case the entropy can be computed directly given Proposition E. e.g. $\mathcal{H}\langle\mathcal{R}\rangle(11?\,00?) = 2$, $\mathcal{H}\langle\mathcal{R}\rangle(0?1?0?) = 3$.

It is possible to combine several elements of *M* to obtain a better solution in terms of entropy : (11?0??) is a *R*-generalization of M_1=(11?00?) and M_2=(11?0?0). To compute its entropy, we compute the upper bound of M_1 and M_2 which is *I*=(11?000). We deduce $\mathcal{H}\langle\mathcal{R}\rangle(11?\,0?\,?) = \log_2(2^2 + 2^2 - 2^1) = \log_2(6) \approx 2.6$

Using these propositions.

The consequence is that since the *InfRes* Algorithm computes points such that $\mathcal{F}_A\langle\mathcal{R}\rangle = 1$, computing the entropy of their infimum can be done in time exponential to the number of points computed. In practice, only a few points need to be computed, and therefore computing $\mathcal{H}\langle\mathcal{R}\rangle(X)$ can be done in a reasonable time.