An Information-Theoretic Approach to Normal Forms for Relational and XML Data

Marcelo Arenas
University of Toronto
marenas@cs.toronto.edu

Leonid Libkin
University of Toronto
libkin@cs.toronto.edu

ABSTRACT

Normalization as a way of producing good database designs is a well-understood topic. However, the same problem of distinguishing well-designed databases from poorly designed ones arises in other data models, in particular, XML. While in the relational world the criteria for being well-designed are usually very intuitive and clear to state, they become more obscure when one moves to more complex data models.

Our goal is to provide a set of tools for testing when a condition on a database design, specified by a *normal form*, corresponds to a good design. We use techniques of information theory, and define a measure of information content of elements in a database with respect to a set of constraints. We first test this measure in the relational context, providing information-theoretic justification for familiar normal forms such as BCNF, 4NF, PJ/NF, 5NFR, DK/NF. We then show that the same measure applies in the XML context, which gives us a characterization of a recently introduced XML normal form called XNF. Finally, we look at information-theoretic criteria for justifying normalization algorithms.

1. Introduction

What constitutes a good database design? This question has been studied extensively, with well-known solutions presented in practically all database texts. But what is it that makes a database design good? This question is usually addressed at a much less formal level. For instance, we know that BCNF is an example of a good design, and we usually say that this is because BCNF eliminates update anomalies. Most of the time this is sufficient, given the simplicity of the relational model and our good intuition about it.

PODS 2003, June 9-12, 2003, San Diego, CA.

Several papers [13, 28, 18] attempted a more formal evaluation of normal forms, by relating it to the elimination of update anomalies. Another criterion is the existence of algorithms that produce good designs: for example, we know that every database scheme can be losslessly decomposed into one in BCNF, but some constraints may be lost along the way.

The previous work was specific for the relational model. As new data formats such as XML are becoming critically important, classical database theory problems have to be revisited in the new context [26, 24]. However, there is as yet no consensus on how to address the problem of well-designed data in the XML setting [10, 3].

It is problematic to evaluate XML normal forms based on update anomalies; while some proposals for update languages exist [25], no XML update language has been standardized. Likewise, using the existence of good decomposition algorithms as a criterion is problematic: for example, to formulate losslessness, one needs to fix a small set of operations in some language, that would play the same role for XML as relational algebra for relations. Stating dependency preservation and testing normal forms is even more problematic: while in the relational world, we have well-understood procedures for doing this, for XML we do not even know if implication of functional dependencies is decidable.

This suggests that one needs a different approach to the justification of normal forms and good designs. Such an approach must be applicable to new data models *before* the issues of query/update/constraint languages for them are completely understood and resolved. Therefore, such an approach must be based on some intrinsic characteristics of the data, as opposed to query/update languages for a particular data model.

In this paper we suggest such an approach based on information-theoretic concepts, more specifically, on measuring the information content of the data. Our goal here is twofold. First, we present information-theoretic measures of "goodness" of a design, and test them in the relational world. To be applicable in other contexts, we expect these measures to characterize familiar normal forms. Second, we apply them in the XML context, and show that they justify a normal form XNF proposed in [3]. We also use our measures to reason about normalization algorithms, by showing that standard decomposition algorithms never decrease the information content of any piece of data in a database/document.

The rest of the paper is organized as follows. In Section 2 we give the notations, and review the basics of information theory (entropy and conditional entropy).

Section 3 is an "appetizer" for the main part of the paper: we present a particularly simple information-theoretic way of measuring the information content of a database, and show how it characterizes BCNF and 4NF. The measure, however, is too coarse, and, furthermore, cannot be used to reason about normalization algorithms.

In Section 4 we present our main information-theoretic measure of the information content of a database. Unlike the measure studied before [16, 6, 8, 17], our measure takes into account both database instance and schema constraints, and defines the content with respect to a set of constraints. A well-designed database is one in which the content of each datum is the maximum possible. We use this measure to characterize BCNF and 4NF as the best way to design schemas under FDs and MVDs, and to justify normal forms involving JDs (PJ/NF, 5NFR) and other types of integrity constraints (DK/NF).

In Section 5, we show that the main measure of Section 4 straightforwardly extends to the XML setting, giving us a definition of well-designed XML specifications. We prove that for constraints given by FDs, well-designed XML specifications are precisely those in XNF.

In Section 6, we use the measures of Sections 4 and 5 to reason about normalization algorithms, by showing that good normalization algorithms do not decrease the information content of each datum at every step.

Finally, Section 7 presents the conclusions. All proofs can be found in the full version of this paper.

2. Notations

Schemas and instances. A database schema S is a finite set of relation names, with a set of attributes, denoted by $sort(R)$, associated with each $R \in S$. We shall identify $sort(R)$ of cardinality m with $\{1, \ldots, m\}$. Throughout the paper, we assume that the domain of each attribute is $\mathbb{N}^+$, the set of positive integers. An instance I of schema S assigns to each symbol $R \in S$ with $m = |sort(R)|$ a relation $I(R)$ which is a finite set of m-tuples over $\mathbb{N}^+$. By $adom(I)$ we mean the active domain of I, that is, the set of all elements of $\mathbb{N}^+$ that occur in I. The size of $I(R)$ is defined as $\|I(R)\| = |sort(R)| \cdot |I(R)|$, and the size of I is $\|I\| = \sum_{R \in S} \|I(R)\|$.

If I is an instance of S, the set of *positions* in I, denoted by $Pos(I)$, is the set $\{(R, t, A) \mid R \in S,\ t \in I(R)$ and $A \in sort(R)\}$. Note that $|Pos(I)| = \|I\|$.

We shall deal with *integrity constraints* which are first-order sentences over S. Given a set Σ of integrity constraints, Σ^+ denotes the set of all constraints implied by it, that is, constraints φ such that for every instance I, $I \models \Sigma$ implies $I \models \varphi$. We define $inst(S, \Sigma)$ as the set of all database instances of S satisfying Σ and $inst_k(S, \Sigma)$ as $\{I \in inst(S, \Sigma) \mid adom(I) \subseteq [1, k]\}$, where $[1, k] = \{1, \ldots, k\}$.

Constraints and normal forms. Here we briefly review the most common normal forms BCNF, 4NF, PJ/NF, 5NFR, and DK/NF. For more information, the reader is referred to [4, 15, 1, 5].

The most widely used among those are BCNF and 4NF, defined in terms of functional dependencies (FD) and multivalued dependencies (MVD), respectively. We shall use the standard notations $X \to Y$ and $X \to\!\!\!\to Y$ for FDs and MVDs. Given a set Σ of FDs over S, (S, Σ) is in BCNF if for every nontrivial FD $X \to Y \in \Sigma^+$, X is a key (that is, if $X \to Y$ is defined over R, then $X \to sort(R) \in \Sigma^+$). If Σ is a set of FDs and MVDs over S, then 4NF is defined analogously [11]: for every nontrivial MVD $X \to\!\!\!\to Y \in \Sigma^+$, X must be a key. Recall that in the case of FDs nontrivial means $Y \not\subseteq X$, and in the case of MVDs nontrivial means $Y \not\subseteq X$ and $X \cup Y \subsetneq sort(R)$.

The normal forms PJ/NF (projection-join normal form) [12] and 5NFR [27] deal with FDs and join dependencies (JDs). Recall that a JD over $R \in S$ is an expression of the form $\bowtie[X_1, \ldots, X_n]$, where $X_1 \cup \cdots \cup X_n = sort(R)$. A database instance I of S satisfies $\bowtie[X_1, \ldots, X_n]$, if $I(R) = \pi_{X_1}(I(R)) \bowtie \cdots \bowtie \pi_{X_n}(I(R))$. Given a set Σ of FDs and JDs over S, (S, Σ) is in PJ/NF if $\Delta \models \Sigma$, where Δ is the set of key dependencies in Σ^+ (that is, dependencies of the form $X \to sort(R)$ for $X \subseteq sort(R)$). In other words, every instance of S that satisfies all the keys in Σ^+ must satisfy Σ as well.

PJ/NF is an extension of both 4NF and BCNF. Since an MVD $X \to\!\!\!\to Y$ over R is a JD $\bowtie[XY, X(sort(R) - Y)]$, when only FDs and MVDs are present in Σ, the definition of PJ/NF coincides with 4NF. If no JDs are present at all, it reduces to the definition of BCNF [12].

An alternative normal form for FDs and JDs was introduced in [27]. Given a set of FDs and JDs Σ over S, a JD $\varphi = \bowtie[X_1, \ldots, X_n]$ in Σ is strong-reduced if for every $i \in [1, n]$, $\bowtie[X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n]$ is not in Σ^+ or $X_1 \cup \cdots \cup X_{i-1} \cup X_{i+1} \cup \cdots \cup X_n \subsetneq sort(R)$. (S, Σ) is in 5NFR (reduced 5th normal form) if for every nontrivial, strong-reduced join dependency $\bowtie[X_1, \ldots, X_n] \in \Sigma^+$ and every $i \in [1, n]$, X_i is a key. PJ/NF is strictly stronger than 5NFR.

The "ultimate" normal form for relational databases was introduced in [13]. In our setting[1], it says the following. Given *any* set of integrity constraints Σ over S, (S, Σ) is in DK/NF (domain-key normal form) if Σ is implied by the set of key dependencies in Σ^+.

[1] When domain dependencies [13] are not considered.

2.1 Basics of Information Theory

The main concept of information theory is that of entropy, which measures the amount of information provided by a certain event. Assume that an event can have n different outcomes $s_1, \ldots, s_n$, each with probability p_i, $i \leq n$. How much information is gained by knowing that s_i occurred? This is clearly a function of p_i. Suppose g measures this information; then it must be continuous and decreasing function with domain $(0, 1]$ (the higher the probability, the less information gained) and $g(1) = 0$ (no information is gained if the outcome is known in advance). Furthermore, g is additive: if outcomes are independent, the amount of information gained by knowing two successive outcomes must be the sum of the two individuals amounts, that is, $g(p_i \cdot p_j) = g(p_i) + g(p_j)$. The only function satisfying these conditions is $g(x) = -c \ln x$, where c is an arbitrary positive constant [23]. It is customary to use base 2 logarithms: $g(x) = -\log x$.

The *entropy* of a probability distribution represents the average amount of information gained by knowing that a particular event occurred. Let $\mathcal{A} = (\{s_1, \ldots, s_n\}, P_\mathcal{A})$ be a probability space. If $p_i = P_\mathcal{A}(s_i)$, then the entropy of $\mathcal{A}$, denoted by $H(\mathcal{A})$, is defined to be

$$H(\mathcal{A}) \;=\; \sum_{i=1}^{n} p_i \log \frac{1}{p_i} \;=\; -\sum_{i=1}^{n} p_i \log p_i.$$

Observe that some of the probabilities in the space $\mathcal{A}$ can be zero. For that case, we adopt the convention that $0 \log \frac{1}{0} = 0$, since $\lim_{x \to 0} x \log \frac{1}{x} = 0$. It is known that $0 \leq H(\mathcal{A}) \leq \log n$, with $H(\mathcal{A}) = \log n$ only for the uniform distribution $P_\mathcal{A}(s_i) = 1/n$ [7].

We shall also use *conditional entropy*. Assume that we are given two probability spaces $\mathcal{A} = (\{s_1, \ldots, s_n\}, P_\mathcal{A})$, $\mathcal{B} = (\{s'_1, \ldots, s'_m\}, P_\mathcal{B})$ and, furthermore, we know probabilities $P(s'_j, s_i)$ of all the events (s'_j, s_i) (that is, $P_\mathcal{A}$ and $P_\mathcal{B}$ need not be independent). Then the conditional entropy of $\mathcal{B}$ given $\mathcal{A}$, denoted by $H(\mathcal{B} \mid \mathcal{A})$, gives the average amount of information provided by $\mathcal{B}$ if $\mathcal{A}$ is known [7]. It is defined using conditional probabilities $P(s'_j \mid s_i) = P(s'_j, s_i)/P_\mathcal{A}(s_i)$:

$$H(\mathcal{B} \mid \mathcal{A}) = \sum_{i=1}^{n} \left(P_\mathcal{A}(s_i) \sum_{j=1}^{m} P(s'_j \mid s_i) \log \frac{1}{P(s'_j \mid s_i)} \right).$$

3. Information theory and normal forms: an appetizer

We will now see a particularly simple way to provide information-theoretic characterization of normal forms. Although it is very easy to present, it has a number of shortcomings, and a more elaborate measure will be presented in the next section.

Violating a normal form, e.g., BCNF, implies having redundancies. For example, if $S = \{R(A, B, C)\}$ and $\Sigma = \{A \to B\}$, then (S, Σ) is not in BCNF (A is not a key) and some instances can contain redundant information: in Figure 1 (a), the value of the gray cell must be equal to the value below it. We do not need to store this value as it can be inferred from the remaining values and the constraints.

We now use the concept of entropy to measure the information content of every position in an instance of S. The basic idea is as follows: we measure how much information we gain if we lose the value in a given position, and then someone restores it (either to the original, or to some other value, not necessarily from the active domain). For instance, if we lose the value in the gray cell in figure 1 (a), we gain zero information if it gets restored, since we know from the rest of the instance and the constraints that it equals 2.

Formally, let $I \in inst_k(S, \Sigma)$ (that is, $adom(I) \subseteq [1, k]$) and let $p \in Pos(I)$ be a position in I. For any value a, let $I_{p \leftarrow a}$ be a database instance constructed from I by replacing the value in position p by a. We define a probability space $\mathcal{E}^k_\Sigma(I, p) = ([1, k+1], P)$ and use its entropy as the measure of information in p (we define it on $[1, k+1]$ to guarantee that there is at least one value outside of the active domain). The function P is:

$$P(a) = \begin{cases} 0 & I_{p \leftarrow a} \not\models \Sigma, \\ 1/|\{b \mid I_{p \leftarrow b} \models \Sigma\}| & \text{otherwise.} \end{cases}$$

In other words, let m be the number of $b \in [1, k+1]$ such that $I_{p \leftarrow b} \models \Sigma$ (note that $m > 0$ since $I \models \Sigma$). For each such b, $P(b) = 1/m$, and elsewhere $P = 0$.

For the instance in figure 1, (a) if p is the position of the gray cell, then the probability distribution is as follows: $P(2) = 1$ and $P(a) = 0$, for all other $a \in [1, k+1]$. Thus, the entropy of $\mathcal{E}^k_\Sigma(I, p)$ for position p is zero, as we expect. More generally, we can show the following.

Theorem 1. *Let Σ be a set of FDs (or FDs and MVDs) over a schema S. Then (S, Σ) is in BCNF (or 4NF, resp.) if and only if for every $k > 1$, $I \in inst_k(S, \Sigma)$ and $p \in Pos(I)$,*

$$H(\mathcal{E}^k_\Sigma(I, p)) > 0.$$

Proof: We give the proof for the case of FDs; for FDs and MVDs the proof is almost identical.

($\Rightarrow$) Assume that (S, Σ) is in BCNF. Fix $k > 0$, $I \in inst_k(S, \Sigma)$ and $p \in Pos(I)$. Assume that a is the pth element in I. We show that $I_{p \leftarrow k+1} \models \Sigma$, from which we conclude that $H(\mathcal{E}^k_\Sigma(I, p)) > 0$, since $\mathcal{E}^k_\Sigma(I, p)$ is uniformly distributed, and $P(a), P(k+1) \neq 0$.

Towards a contradiction, assume that $I_{p \leftarrow k+1} \not\models \Sigma$. Then, there exist $R \in S$, $t'_1, t'_2 \in I_{p \leftarrow k+1}(R)$ and $X \to A \in \Sigma^+$ such that $t'_1[X] = t'_2[X]$ and $t'_1[A] \neq t'_2[A]$. Assume that t'_1, t'_2 were generated from tuples $t_1, t_2 \in I(R)$, respectively. Note that $t'_1[X] = t_1[X]$ (if $t_1[X] \neq t'_1[X]$, then $t'_1[B] = k+1$ for some $B \in X$; given that $k+1 \notin adom(I)$, only one position in $I_{p \leftarrow k+1}$ mentions

A	B	C
1	2	3
1	2	4

(a)

A	B	C
1	1	2
2	3	4

(b)

A	B	C
1	2	3
1	2	4
1	2	5

(c)

Figure 1: Database instances.

this value and, therefore, $t'_1[X] \neq t'_2[X]$, a contradiction). Similarly, $t'_2[X] = t_2[X]$ and, therefore, $t_1[X] = t_2[X]$. Given that (S, Σ) is in BCNF, X must be a key in R. Hence, $t_1 = t_2$, since $I \models \Sigma$. We conclude that $t'_1 = t'_2$ and, thus, $t'_1[A] = t'_2[A]$, a contradiction.

($\Leftarrow$) Assume that (S, Σ) is not in BCNF. We show that there exists $k > 0$, $I \in inst_k(S, \Sigma)$ and $p \in Pos(I)$ such that $H(\mathcal{E}^k_\Sigma(I, p)) = 0$.

Since (S, Σ) is not in BCNF, there exist $R \in S$ and $X \to A \in \Sigma^+$ such that $A \notin X$, $X \cup \{A\} \subsetneq sort(R)$ and X is not a key in R. Thus, there exists a database instance I of S such that $I \models \Sigma$ and $I \not\models X \to sort(R)$. We can assume that $I(R)$ contains only two tuples, say t_1, t_2. Let k be the greatest value in I, $i = t_1[A]$ and p be the position of $t_1[A]$ in I. It is easy to see that $I \in inst_k(S, \Sigma)$ and $P(j) = 0$, for every $j \neq i$ in $[1, k+1]$, since $t_1[A]$ must be equal to $t_2[A] = i$. Therefore, $H(\mathcal{E}^k_\Sigma(I, p)) = 0$. $\square$

That is, a schema is in BCNF or 4NF iff for every instance, each position carries non-zero amount of information.

This is a clean characterization of BCNF and 4NF, but the measure $H(\mathcal{E}^k_\Sigma(I, p))$ is not accurate enough for a number of reasons.

For example, let $\Sigma_1 = \{A \to B\}$ and $\Sigma_2 = \{A \to\to B\}$. The instance I in figure 1 (a) satisfies Σ_1 and Σ_2. Let p be the position of the gray cell in I. Then $H(\mathcal{E}^k_{\Sigma_1}(I, p)) = H(\mathcal{E}^k_{\Sigma_2}(I, p)) = 0$. But intuitively, the information content of p must be higher under Σ_2 than Σ_1, since Σ_1 says that the value in p must be equal to the value below it, and Σ_2 says that this should only happen if the values of the C-attribute are distinct.

Next, consider I_1 and I_2 shown in figures 1 (a) and (c), respectively. Let $\Sigma = \{A \to B\}$, and let p_1 and p_2 denote the positions of the gray cells in I_1 and I_2. Then, $H(\mathcal{E}^k_\Sigma(I_1, p_1)) = H(\mathcal{E}^k_\Sigma(I_2, p_2)) = 0$. But again we would like them to have different values, as the amount of redundancy is higher in I_2 than in I_1.

Finally, let $S = R(A, B)$, $\Sigma = \{\emptyset \to\to A\}$, and $I = \{1, 2\} \times \{1, 2\} \in inst(S, \Sigma)$. For each position, the entropy would be zero. However, consider both positions in attribute A corresponding to the value 1. If they both disappear, then we know that no matter how they are restored, the values must be the same. The measure presented in this section cannot possibly talk about interdependencies of this kind.

In the next section we will present a measure that overcomes these problems.

4. A general definition of well-designed data

Let S be a schema, Σ a set of constraints, and let $I \in inst(S, \Sigma)$ be an instance with $\|I\| = n$. Recall that $Pos(I)$ is the set of positions in I, that is, $\{(R, t, A) \mid R \in S,\ t \in I(R) \text{ and } A \in sort(R)\}$. Our goal is to define a function $\text{INF}_I(p \mid \Sigma)$, the information content of a position $p \in Pos(I)$ with respect to the set of constraints Σ. For a general definition of well-designed data, we want to say that this measure has the maximum possible value. This is a bit problematic for the case of an infinite domain ($\mathbb{N}^+$), since we only know what the maximum value of entropy is for a discrete distribution over k elements: $\log k$.

To overcome this, we define, for each $k > 0$, a function $\text{INF}^k_I(p \mid \Sigma)$ that would only apply to instances whose active domain is contained in $[1, k]$, and then consider the ratio $\frac{\text{INF}^k_I(p|\Sigma)}{\log k}$. This ratio tells us how close the given position p is to having the maximum possible information content, for databases with active domain in $[1, k]$. As our final measure $\text{INF}_I(p \mid \Sigma)$ we then take the limit of this sequence as k goes to infinity.

Informally, $\text{INF}^k_I(p \mid \Sigma)$ is defined as follows. Let $X \subseteq Pos(I) - \{p\}$. Suppose the values in those positions X are lost, and then someone restores them from the set $[1, k]$; we measure how much information about the value in p this gives us. This measure is defined as the entropy of a suitably chosen distribution. Then $\text{INF}^k_I(p \mid \Sigma)$ is the average such entropy over all sets $X \subseteq Pos(I) - \{p\}$. Note that this is much more involved than the definition of the previous section, as it takes into account all possible interactions between different positions in an instance and the constraints.

We now present this measure formally. An *enumeration* of I with $\|I\| = n$, $n > 0$, is a bijection f_I between $Pos(I)$ and $[1, n]$. From now on, we assume that every instance has an associated enumeration[2]. We say that the position of $(R, t, A) \in Pos(I)$ is p in I if the enumeration of I assigns p to (R, t, A), and if R is clear from the context, we say that the position of $t[A]$ is p. We normally associate positions with their rank in the enumeration f_I.

[2]The choice of a particular enumeration will not affect the measures we define.

A	B	C
6	5	4
3	2	1

(a) An enumeration of I

A	B	C
1	7	3
1	2	4

(b) $I_{(7,\bar{a}_1)} = \sigma_1(I_{(7,\bar{a}_1)})$

A	B	C
v_6	7	3
1	2	v_1

(c) $I_{(7,\bar{a}_2)}$

A	B	C
8	7	3
1	2	4

(d) $\sigma_2(I_{(7,\bar{a}_2)})$

Figure 2: Defining $\text{INF}_I^k(p \mid \Sigma)$.

Fix a position $p \in Pos(I)$. As the first step, we need to describe all possible ways of removing values in a set of positions X, different from p. To do this, we shall be placing variables from a set $\{v_i \mid i \geq 1\}$ in positions where values are to be removed. Furthermore, we assume that each set of positions is equally likely to be removed. To model this, let $\Omega(I, p)$ be the set of all 2^{n-1} vectors $(a_1, \ldots, a_{p-1}, a_{p+1}, \ldots, a_n)$ such that for every $i \in [1, n] - \{p\}$, a_i is either v_i or the value in the i-th position of I. A probability space $\mathcal{A}(I, p) = (\Omega(I, p), P)$ is defined by taking P to be the uniform distribution.

Example 1: Let I be the database instance shown in Figure 1 (a). An enumeration of the positions in I is shown in Figure 2 (a). Assume that p is the position of the gray cell shown in Figure 1 (a), that is, $p = 5$. Then, $\bar{a}_1 = (4, 2, 1, 3, 1)$ and $\bar{a}_2 = (v_1, 2, 1, 3, v_6)$ are among the 32 vectors in $\Omega(I, p)$. For each of these vectors, we define P as $\frac{1}{32}$. □

Our measure $\text{INF}_I^k(p \mid \Sigma)$, for $I \in inst_k(S, \Sigma)$, will be defined as the conditional entropy of a distribution on $[1, k]$, given the above distribution on $\Omega(I, p)$. For that, we define conditional probabilities $P(a \mid \bar{a})$ that characterize how likely a is to occur in position p, if some values are removed from I according to the tuple $\bar{a}$ from $\Omega(I, p)$[3]. We need a couple of technical definitions first.

If $\bar{a} = (a_i)_{i \neq p}$ is a vector in $\Omega(I, p)$ and $a > 0$, then $I_{(a,\bar{a})}$ is a table obtained from I by putting a in position p, and a_i in position $i, i \neq p$. If $k > 0$, then a *substitution* $\sigma : \bar{a} \to [1, k]$ assigns a value from $[1, k]$ to each a_i which is a variable, and leaves other a_is intact. We can extend σ to $I_{(a,\bar{a})}$ and thus talk about $\sigma(I_{(a,\bar{a})})$.

Example 2: (example 1 continued) Let $k = 8$ and σ_1 be an arbitrary substitution from $\bar{a}_1$ to $[1, 8]$. Note that σ_1 is the identity substitution, since $\bar{a}_1$ contains no variables. Figure 2 (b) shows $I_{(7,\bar{a}_1)}$, which is equal to $\sigma_1(I_{(7,\bar{a}_1)})$.

Let σ_2 be a substitution from $\bar{a}_2$ to $[1, 8]$ defined as follows: $\sigma(v_1) = 4$ and $\sigma(v_6) = 8$. Figure 2 (c) shows $I_{(7,\bar{a}_2)}$ and Figure 2 (d) shows the database instance generated by applying σ_2 to $I_{(7,\bar{a}_2)}$. □

If Σ is a set of constraints over S, then $SAT_\Sigma^k(I_{(a,\bar{a})})$ is defined as the set of all substitutions $\sigma : \bar{a} \to [1, k]$ such that $\sigma(I_{(a,\bar{a})}) \models \Sigma$ and $\|\sigma(I_{(a,\bar{a})})\| = \|I\|$ (the latter ensures that no two tuples collapse as the result of applying σ). With this, we define $P(a \mid \bar{a})$ as:

$$P(a \mid \bar{a}) \quad = \quad \frac{|SAT_\Sigma^k(I_{(a,\bar{a})})|}{\displaystyle\sum_{b \in [1,k]} |SAT_\Sigma^k(I_{(b,\bar{a})})|}.$$

We remark that this corresponds to conditional probabilities with respect to a distribution P' on $[1, k] \times \Omega(I, p)$ defined by $P'(a, \bar{a}) = P(a \mid \bar{a}) \cdot (1/2^{n-1})$, and that P' is indeed a probability distribution for every $I \in inst_k(S, \Sigma)$ and $p \in Pos(I)$.

Example 3: (example 2 continued) Assume that $\Sigma = \{A \to B\}$. Given that the only substitution σ from $\bar{a}_1$ to $[1, 8]$ is the identity, for every $a \in [1, 8]$, $a \neq 2$, $\sigma(I_{(a,\bar{a}_1)}) \not\models \Sigma$, and, therefore, $SAT_\Sigma^8(I_{(a,\bar{a}_1)}) = \emptyset$. Thus, $P(2 \mid \bar{a}_1) = 1$ since $\sigma(I_{(2,\bar{a}_1)}) \models \Sigma$. This value reflects the intuition that if the value in the gray cell of the instance shown in Figure 1 (a) is removed, then it can be inferred from the remaining values and the FD $A \to B$.

There are 64 substitutions with domain $\bar{a}_2$ and range $[1, 8]$. A substitution σ is in $SAT_\Sigma^8(I_{(7,\bar{a}_2)})$ if and only if $\sigma(v_6) \neq 1$, and, therefore, $|SAT_\Sigma^8(I_{(7,\bar{a}_2)})| = 56$. The same can be proved for every $a \in [1, 8]$, $a \neq 2$. On the other hand, the only substitution that is not in $SAT_\Sigma^8(I_{(2,\bar{a}_2)})$ is $\sigma(v_1) = 3$ and $\sigma(v_6) = 1$, since $\sigma(I_{(2,\bar{a}_2)})$ contains only one tuple. Thus, $|SAT_\Sigma^8(I_{(2,\bar{a}_2)})| = 63$ and, therefore,

$$P(a \mid \bar{a}_2) = \begin{cases} \frac{63}{455} & \text{if } a = 2, \\ \\ \frac{56}{455} & \text{otherwise.} \end{cases}$$ □

We define a probability space $\mathcal{B}_\Sigma^k(I, p) = ([1, k], P)$ where

$$P(a) \quad = \quad \frac{1}{2^{n-1}} \sum_{\bar{a} \in \Omega(I,p)} P(a \mid \bar{a}) \ ,$$

and, again, omit I, p, k and Σ as parameters, and overload the letter P since this will never lead to confusion.

The measure of the amount of information in position p, $\text{INF}_I^k(p \mid \Sigma)$, is the conditional entropy of $\mathcal{B}_\Sigma^k(I, p)$ given $\mathcal{A}(I, p)$, that is, the average information provided by p, given all possible ways of removing values in the instance

[3]We use the same letter P here, but this will never lead to confusion. Furthermore, all probability distributions depend on I, p, k and Σ, but we omit them as parameters of P since they will always be clear from the context.

I:

$$\text{INF}^k_I(p \mid \Sigma) \stackrel{\text{def}}{=} H(\mathcal{B}^k_\Sigma(I,p) \mid \mathcal{A}(I,p)) = \sum_{\bar{a}\in\Omega(I,p)} \Big(P(\bar{a}) \sum_{a\in[1,k]} P(a \mid \bar{a}) \log \frac{1}{P(a \mid \bar{a})} \Big).$$

Note that for $\bar{a} \in \Omega(I,p)$, $\sum_{a\in[1,k]} P(a \mid \bar{a}) \log \frac{1}{P(a|\bar{a})}$ measures the amount of information in position p, given a set of constraints Σ and some missing values in I, represented by the variables in $\bar{a}$. Thus, $\text{INF}^k_I(p \mid \Sigma)$ is the average such amount over all $\bar{a} \in \Omega(I,p)$.

From the definition of conditional entropy, $0 \leq \text{INF}^k_I(p \mid \Sigma) \leq \log k$, and the measure $\text{INF}^k_I(p \mid \Sigma)$ depends on the domain size k. We now consider the ratio of $\text{INF}^k_I(p \mid \Sigma)$ and the maximum entropy $\log k$. It turns out that this sequence converges:

Lemma 1. *If Σ is a set of first-order constraints over a schema S, then for every $I \in inst(S,\Sigma)$ and $p \in Pos(I)$, $\lim_{k\to\infty} \text{INF}^k_I(p \mid \Sigma)/\log k$ exists.*

In fact, Lemma 1 shows that such a limit exists for any set of *generic* constraints, that is, constraints that do not depend on the domain. This finally gives us the definition of $\text{INF}_I(p \mid \Sigma)$.

Definition 1. *For $I \in inst(S,\Sigma)$ and $p \in Pos(I)$, the measure $\text{INF}_I(p \mid \Sigma)$ is defined as*

$$\lim_{k\to\infty} \frac{\text{INF}^k_I(p \mid \Sigma)}{\log k}.$$

$\text{INF}_I(p \mid \Sigma)$ measures how much information is contained in position p, and $0 \leq \text{INF}_I(p \mid \Sigma) \leq 1$. A well-designed schema should not have an instance with a position that has less than maximum information:

Definition 2. *A database specification (S,Σ) is* well-designed *if for every $I \in inst(S,\Sigma)$ and every $p \in Pos(I)$, $\text{INF}_I(p \mid \Sigma) = 1$.*

Example 4: Let S be a database schema $\{R(A,B,C)\}$. Let $\Sigma_1 = \{A \to BC\}$. Figure 1 (b) shows an instance I of S satisfying Σ_1 and figure 3 (a) shows the value of $\text{INF}^k_I(p \mid \Sigma_1)$ for $k = 5,6,7$, where p is the position of the gray cell. As expected, the value of $\text{INF}^k_I(p \mid \Sigma_1)$ is maximal, since (S,Σ_1) is in BCNF.

The next two examples show that the measure $\text{INF}^k_I(p \mid \Sigma)$ can distinguish cases that were indistinguishable with the measure of Section 3. Let $\Sigma_2 = \{A \to B\}$ and $\Sigma'_2 = \{A \to\to B\}$. Figure 1 (a) shows an instance I of S satisfying both Σ_2 and Σ'_2. Figure 3 (b) shows the value of $\text{INF}^k_I(p \mid \Sigma_2)$ and $\text{INF}^k_I(p \mid \Sigma'_2)$ for $k = 5,6,7$. As expected, the values are smaller for Σ_2.

Finally, let $\Sigma_3 = \{A \to B\}$. Figures 1 (a) and 1 (c) show instances I_1, I_2 of S satisfying Σ_3. We expect the information content of the gray cell to be smaller in I_2 than in I_1, but the measure used in Section 3 could not distinguish them. Figure 3 (c) shows the values of $\text{INF}^k_{I_1}(p \mid \Sigma_3)$ and $\text{INF}^k_{I_2}(p \mid \Sigma_3)$ for $k = 5,6,7$. As expected, the values are smaller for I_2. Furthermore, $\text{INF}_{I_1}(p \mid \Sigma_3) = 0.875$ and $\text{INF}_{I_2}(p \mid \Sigma_3) = 0.78125$. □

4.1 Basic properties

It is clear from the definitions that $\text{INF}_I(p \mid \Sigma)$ does not depend on a particular enumeration of positions. Two other basic properties that we can expect from the measure of information content are as follows: first, it should not depend on a particular representation of constraints, and second, a schema without constraints must be well-designed (as there is nothing to tell us that it is not). Both are indeed true.

Proposition 1.

(1) Let Σ_1 and Σ_2 be two sets of constraints over a schema S. If they are equivalent (that is, $\Sigma_1^+ = \Sigma_2^+$), then for any instance I satisfying Σ_1 and any $p \in Pos(I)$, $\text{INF}_I(p \mid \Sigma_1) = \text{INF}_I(p \mid \Sigma_2)$.

(2) If $\Sigma = \emptyset$, then (S,Σ) is well-designed.

The definition of being well-designed states that $\lim_{k\to\infty}(\text{INF}^k_I(p \mid \Sigma)/\log k) = 1$. This leaves open the possibility that $\text{INF}^k_I(p \mid \Sigma)$ exhibits sub-logarithmic growth, e.g., $\log k - \log\log k$ which results in almost, but not completely perfect information in position p. It turns out that such behavior is impossible: if $\lim_{k\to\infty}(\text{INF}^k_I(p \mid \Sigma)/\log k) = 1$, then $\text{INF}^k_I(p \mid \Sigma)$ cannot grow sub-logarithmically. We show this by establishing a structural criterion for $\text{INF}_I(p \mid \Sigma) = 1$.

Proposition 2. *Let S be a schema and Σ a set of constraints over S. Then the following are equivalent.*

(1) (S,Σ) is well-designed.

(2) For every $I \in inst(S,\Sigma)$ and $p \in Pos(I)$, $\lim_{k\to\infty}[\log k - \text{INF}^k_I(p \mid \Sigma)] = 0$.

(3) For every $I \in inst(S,\Sigma)$, $p \in Pos(I)$ and $a \in \mathbb{N}^+ - adom(I)$, $I_{p\leftarrow a} \models \Sigma$.

A natural question at this point is whether the problem of checking if a relational schema is well-designed is decidable. It is not surprising that for arbitrary first-order constraints, the problem is undecidable:

Proposition 3. *The problem of verifying whether a relational schema containing first-order constraints is well-designed is undecidable.*

However, integrity constraints used in database schema design are most commonly *universal*, that is, of the form $\forall\bar{x}\,\psi(\bar{x})$, where $\psi(\bar{x})$ is a quantifier-free formula. FDs, MVDs and JDs are universal constraints as well as more

k	$A \to BC$	$\log k$
5	2.3219	2.3219
6	2.5850	2.5850
7	2.8074	2.8074

(a)

k	$A \to B$	$A \to\to B$
5	2.0299	2.2180
6	2.2608	2.4637
7	2.4558	2.6708

(b)

k	I_1	I_2
5	2.0299	1.8092
6	2.2608	2.0167
7	2.4558	2.1914

(c)

Figure 3: Value of conditional entropy.

elaborated dependencies such as equality generating dependencies and full tuple generating dependencies [1]. For universal constraints, the problem of testing if a relational schema is well-designed can be reduced to the problem of verifying whether a Schönfinkel-Bernays sentence is inconsistent. Using complexity bounds for the latter [22], we obtain the following result.

Proposition 4. *The problem of deciding whether a schema containing only universal constraints is well-designed is in co-NEXPTIME. Furthermore, if for a fixed m, each relation in S has at most m attributes, then the problem is in PSPACE.*

For specific kinds of constraints, e.g., FDs, MVDs, lower complexity bounds will follow from the results in the next section.

4.2 Justification of relational normal forms

We now apply the criterion of being well-designed to various relational normal forms. We show that all of them lead to well-designed specifications, and some precisely characterize the well-designed specifications that can be obtained with a class of constraints.

We start by finding constraints that always give rise to well-designed schemas. An *extended-key* over a relational schema S is a constraint φ of the form:

$$\forall \, (R(\bar{x}_1) \wedge \cdots \wedge R(\bar{x}_m) \to \bar{x}_i = \bar{x}_j),$$

where $i, j \in [1, m]$, $\forall$ represents the universal closure of a formula, and there is an assignment of variables to columns such that each variable occurs only in one column (that is, an extended-key is a typed constraint [1]). Note that every key is an extended-key.

Proposition 5. *If S is a schema and Σ a set of extended-keys over S, then (S, Σ) is well-designed.*

Corollary 1. *A relational specification (S, Σ) in DK/NF is well-designed.*

Next, we characterize well-designed schemas with FDs and JDs.

Theorem 2. *Let Σ be a set of FDs and JDs over a relational schema S. (S, Σ) is well-designed if and only if for every $R \in S$ and every nontrivial join dependency $\forall(R(\bar{x}_1) \wedge \cdots \wedge R(\bar{x}_m) \to R(\bar{x}))$ in Σ^+, there exists $M \subseteq \{1, \ldots, m\}$ such that:*

1. *$\bar{x} \subseteq \bigcup_{i \in M} \bar{x}_i$.*
2. *For every $i, j \in M$, $\forall(R(\bar{x}_1) \wedge \cdots \wedge R(\bar{x}_m) \to \bar{x}_i = \bar{x}_j) \in \Sigma^+$.*

This justifies various normal forms proposed for JDs and FDs [12, 27].

Corollary 2. *Let Σ be a set of FDs and JDs over a relational schema S. If (S, Σ) is in PJ/NF or 5NFR, then it is well-designed.*

However, neither of these normal forms characterizes precisely the notion of being well-defined:

Proposition 6. *There exists a schema S and a set of JDs and FDs Σ such that (S, Σ) is well-designed, but it violates all of the following: DK/NF, PJ/NF, 5NFR.*

Proof: Let $S = \{R(A, B, C)\}$ and $\Sigma = \{AB \to C, AC \to B, \bowtie[AB, AC, BC]\}$. This specification is not in DK/NF and PJ/NF since the set of keys implied by Σ is $\{AB \to ABC, AC \to ABC, ABC \to ABC\}$ and this set does not imply $\bowtie[AB, AC, BC]$. Furthermore, this specification is not in 5NFR since $\bowtie[AB, AC, BC]$ is a strong-reduced join dependency and BC is not a key in Σ.

Join dependency $\bowtie[AB, AC, BC]$ corresponds to the following first order sentence:

$$\forall x \forall y \forall z \forall u_1 \forall u_2 \forall u_3 \, (R(x, y, u_1) \wedge R(x, u_2, z) \wedge R(u_3, y, z) \to R(x, y, z)).$$

From Theorem 2, we conclude that (S, Σ) is well designed since Σ implies the sentence

$$\forall x \forall y \forall z \forall u_1 \forall u_2 \forall u_3 (R(x, y, u_1) \wedge R(x, u_2, z) \wedge R(u_3, y, z) \to y = u_2 \wedge z = u_1).$$

and $(x, y, z) \subseteq (x, y, u_1) \cup (x, u_2, z)$. $\square$

By restricting Theorem 2 to the case of specifications containing only FDs and MVDs (or only FDs), we obtain the equivalence between well-designed databases and 4NF (respectively, BCNF).

Theorem 3. *Let Σ be a set of integrity constraints over a relational schema S.*

1. *If Σ contains only FDs and MVDs, then (S, Σ) is well-designed if and only if it is in 4NF.*
2. *If Σ contains only FDs, then (S, Σ) is well-designed if and only if it is in BCNF.*

5. Normalizing XML data

In this section we give a quick overview of the XML normal form called XNF, and show that the notion of being well-designed straightforwardly extends from relations to XML. Furthermore, if all constraints are specified as functional dependencies, this notion precisely characterizes XNF.

5.1 Overview of XML constraints and normalization

DTDs and XML trees. We shall use a somewhat simplified model of XML trees in order to keep the notation simple. We assume a countably infinite set of labels L, a countably infinite set of attributes A (we shall use the notation $@l_1, @l_2$, etc for attributes to distinguish them from labels), and a countably infinite set V of values of attributes. Furthermore, we do not consider PCDATA elements in XML trees since they can always be represented by attributes.

A DTD (Document Type Definition) D is a 4-tuple (L_0, P, R, r) where L_0 is a finite subset of L, P is a set of rules $a \to P_a$ for each $a \in L_0$, where P_a is a regular expression over $L_0 - \{r\}$, R assigns to each $a \in L_0$ a finite subset of A (possibly empty; $R(a)$ is the set of attributes of a), and $r \in L_0$ (the root).

Example 5: The DTD below is a part of DBLP [9] that stores conference data.

```
<!ELEMENT db (conf*)>
<!ELEMENT conf (issue+)>
  <!ATTLIST conf
            title CDATA #REQUIRED>
<!ELEMENT issue (inproceedings+)>
<!ELEMENT inproceedings EMPTY>
  <!ATTLIST inproceedings
            author CDATA #REQUIRED
            title CDATA #REQUIRED
            pages CDATA #REQUIRED
            year CDATA #REQUIRED>
```

This DTD is represented as $(L_0,\ P,\ R,\ r)$, where $r = db$, $L_0 = \{db,\ conf,\ issue,\ inproceedings\}$, $P = \{db \to conf^*,\ conf \to issue^+,\ issue \to inproceedings^+,\ inproceedings \to \epsilon\}$, $R(conf) = \{@title\}$, $R(inproceedings) = \{@author,\ @title,\ @pages, @year\}$ and $R(db) = R(issue) = \emptyset$. $\square$

An XML tree is a finite rooted directed tree $T = (N, E)$ where N is the set of nodes and E is the set of edges, together with the labeling function $\lambda : N \to L$ and partial attribute value functions $\rho_{@l} : N \to V$ for each $@l \in A$. We furthermore assume that for every node x in N, its children $x_1, \ldots, x_n$ are ordered and $\rho_{@l}(x)$ is defined for a finite set of attributes $@l$. We say that T conforms to DTD D, written as $T \models D$, if the root of T is labeled r, for every $x \in N$ with $\lambda(x) = a$, the word $\lambda(x_1) \cdots \lambda(x_n)$ that consists of the labels of its children belongs to the language denoted by P_a, and for every $x \in N$ with $\lambda(x) = a$, $@l \in R(a)$ if and only if the function $\rho_{@l}$ is defined on x (and thus provides the value of attribute $@l$).

FDs for XML. An *element path* q is a word in L^*, and an *attribute path* is a word of the form $q.@l$, where $q \in L^*$ and $@l \in A$. An element path q is consistent with a DTD D if there is a tree $T \models D$ that contains a node reachable by q (in particular, all such paths must have r as the first letter); if in addition the nodes reachable by q have attribute $@l$, then the attribute path $q.@l$ is consistent with D. The set of all paths (element or attribute) consistent with D is denoted by $paths(D)$. This set is finite for a non-recursive D and infinite if D is recursive.

A *functional dependency* over DTD D [3] is an expression of the form $\{q_1, \ldots, q_n\} \to q$, where $q, q_1, \ldots, q_n \in paths(D)$.

To define the notion of satisfaction for FDs, we use a relational representation of XML trees from [3]. Given $T \models D$, a *tree tuple* in D is a mapping $t : paths(D) \to N \cup V \cup \{\perp\}$ such that if q is an element path whose last letter is a and $t(q) \neq \perp$, then

- $t(q) \in N$ and its label, $\lambda(t(q))$, is a;
- if q' is a prefix of q, then $t(q') \neq \perp$ and the node $t(q')$ lies on the path from the root to $t(q)$ in T;
- if $@l$ is defined for $t(q)$ and its value is $v \in V$, then $t(q.@l) = v$.

Intuitively, a tree tuple assigns nodes or attribute values or nulls ($\perp$) to paths in a consistent manner. A tree tuple is maximal if it cannot be extended to another one by changing some nulls to values from $N \cup V$. The set of maximal tree tuples is denoted by $tuples_D(T)$.

Now we say that FD $\{q_1, \ldots, q_n\} \to q$ is true in T if for any $t_1, t_2 \in tuples_D(T)$, whenever $t_1(q_i) = t_2(q_i) \neq \perp$ for all $i \leq n$, then $t_1(q) = t_2(q)$ holds.

Example 6: Let D be the DTD from Example 5. Among the set Σ of FDs over this DTD are:

$$db.conf.@title \to db.conf,$$
$$db.conf.issue \to db.conf.issue.inproceedings.@year.$$

The first FD specifies that two distinct conferences must have distinct titles. The second one specifies that any two *inproceedings* children of the same *issue* must have the same value of *@year*. $\square$

XML normal form. Suppose we are given a DTD D and a set Σ of FDs over D. The set of all FDs implied by (D, Σ) is denoted by $(D, \Sigma)^+$, and an FD is called *trivial* if it belongs to $(D, \emptyset)^+$, that is, implied by the DTD alone. For example, $q \to r$, where r is the root, or $q \to q.@l$, are trivial FDs.

We say that (D, Σ) is in the *normal form XNF* [3] if for any nontrivial FD $X \to q.@l$ in $(D,\Sigma)^+$, where X is a set of paths, the FD $X \to q$ is in $(D,\Sigma)^+$ as well.

Intuitively, a violation of XNF means that there is some redundancy in the document: we may have many nodes reachable by path q but all of them will have the same value of attribute $@l$ (provided they agree on X).

Example 7: The DBLP example 5 seen earlier may contain redundant information: year is stored multiple times for a conference. It is *not* in XNF since

$$db.conf.issue \;\to\; db.conf.issue.inproceedings$$

is not in $(D,\Sigma)^+$. This suggests making $@year$ an attribute of $issue$, and indeed, such a revised specification can easily be shown to be in XNF. $\square$

5.2 Well-designed XML data

We do not need to introduce a new notion of being well-designed specifically for XML: the definition that we formulated in Section 4 for relational data will apply. We only have to define the notion of positions in a tree, and then reuse the relational definition.

For relational databases, positions correspond to the "shape" of relations, and each position contains a value. Likewise, for XML, positions will correspond to the shape (that is more complex, since documents are modeled as trees), and they must have values associated with them. Consequently, we formally define the set of positions $Pos(T)$ in a tree $T = (N, E)$ as

$$\{(x, @l) \mid x \in N,\ @l \in R(\lambda(x))\}.$$

As before, we assume that there is an enumeration of positions (a bijection between $Pos(T)$ and $\{1, \ldots, n\}$ where $n = |Pos(T)|$) and we shall associate positions with their numbers in the enumeration. We define $adom(T)$ as the set of all values of attributes in T.

As in the relational case, we take the domain of values V to be $\mathbb{N}^+$. Let Σ be a set of FDs over a DTD D and $k > 0$. Define $inst(D,\Sigma)$ as the set of all XML trees that conform to D and satisfy Σ and $inst_k(D,\Sigma)$ as its restriction to trees T with $adom(T) \subseteq [1,k]$.

Now fix $T \in inst_k(D,\Sigma)$ and $p \in Pos(T)$. With the above definitions, we define the probability spaces $\mathcal{A}(T,p)$ and $\mathcal{B}^k_\Sigma(T,p)$ exactly as we defined $\mathcal{A}(I,p)$ and $\mathcal{B}^k_\Sigma(I,p)$ for a relational instance I. That is, $\Omega(T,p)$ is the set of all tuples $\bar{a}$ of the form $(a_1, \ldots, a_{p-1}, a_{p+1}, \ldots, a_n)$ such that every a_i is either a variable, or the value T has in the corresponding position, $SAT^k_\Sigma(T_{(a,\bar{a})})$ as the set of all possible ways to assign values from $[1,k]$ to variables in $\bar{a}$ that result in a tree satisfying Σ, and the rest of the definition repeats the relational case one verbatim, substituting T for I.

We use these to define $\text{INF}^k_T(p \mid \Sigma)$ as the entropy of $\mathcal{B}^k_\Sigma(T,p)$ given $\mathcal{A}(T,p)$:

$$\text{INF}^k_T(p \mid \Sigma) \stackrel{\text{def}}{=} H(\mathcal{B}^k_\Sigma(T,p) \mid \mathcal{A}(T,p)) .$$

As in the relational case, we can show that the limit

$$\lim_{k\to\infty} \frac{\text{INF}^k_T(p \mid \Sigma)}{\log k}$$

exists, and we denote it by $\text{INF}_T(p \mid \Sigma)$. Following the relational case, we introduce

Definition 3. *An XML specification (D,Σ) is* well-designed *if for every $T \in inst(D,\Sigma)$ and every $p \in Pos(T)$, $\text{INF}_T(p \mid \Sigma) = 1$.*

Note that the information-theoretic definition of well-designed schema presented in Section 4 for relational data proved to be extremely robust, as it extended straightforwardly to a different data model: we only needed a new definition of $Pos(T)$ to use in place of $Pos(I)$, and $Pos(T)$ is simply an enumeration of all the places in a document where attribute values occur.

Now we show the connection between well-designed XML and XNF:

Theorem 4. *An XML specification (D,Σ), where Σ is a set of FDs, is well-designed iff it is in XNF.*

The theory of XML constraints and normal forms is not nearly as advanced as its relational counterparts, but we demonstrated here that the definition of well-designed schemas works well for the existing normal form based on FDs; thus, it can be used to test other design criteria for XML when they are proposed.

6. Normalization algorithms

We now show how the information-theoretic measure of Section 4 can be used for reasoning about normalization algorithms at the instance level. For this section, we assume that Σ is a set of FDs, both for the relational and the XML cases. The results shown here state that after each step of a decomposition algorithm, the amount of information in each position does not decrease.

6.1 Relational Databases

Let I' be the result of applying one step of a normalization algorithm to I; we need to show how to associate positions in I and I'. Since Σ contains FDs, we deal with BCNF, and standard BCNF decomposition algorithms use the steps of the following kind: pick a relation R with the set of attributes W, and let W be the disjoint union of X, Y, Z, such that $X \to Y \in \Sigma^+$. Then an instance $I = I(R)$ of R gets decomposed into $I_{XY} = \pi_{XY}(I)$ and $I_{XZ} = \pi_{XZ}(I)$, with the sets of FDs Σ_{XY} and Σ_{XZ}, where Σ_U stands for $\{C \to D \in \Sigma^+ \mid CD \subseteq U \subseteq W\}$.

This decomposition gives rise to two partial maps π_{XY} : $Pos(I) \to Pos(I_{XY})$ and $\pi_{XZ} : Pos(I) \to Pos(I_{XZ})$. If p is the position of $t[A]$ for some $A \in XY$, then $\pi_{XY}(p)$ is defined, and equals the position of $\pi_{XY}(t)[A]$ in I_{XY}; the mapping π_{XZ} is defined analogously. Note that π_{XY} and π_{XZ} can map different positions in I to the same position of I_{XY} or I_{XZ}.

We now show that the amount of information in each position does not decrease in the normalization process.

Theorem 5. *Let (X, Y, Z) partition the attributes of R, and let $X \to Y \in \Sigma^+$. Let $I \in inst(R, \Sigma)$ and $p \in Pos(I)$. If U is either XY or XZ and π_U is defined on p, then* $\text{INF}_I(p \mid \Sigma) \leq \text{INF}_{I_U}(\pi_U(p) \mid \Sigma_U)$.

A decomposition algorithm is *effective* in I if for one of its basic steps, and for some p, the inequality in Theorem 5 is strict: that is, the amount of information increases. This notion leads to another characterization of BCNF.

Proposition 7. *(R, Σ) is in BCNF if and only if no decomposition algorithm is effective in (R, Σ).*

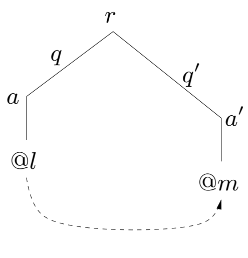
6.2 XML data

We now treat the XML case. We shall prove a result similar to Theorem 5. However, to state the result, we first need to review the normalization algorithm for XML data, and explain how each step of the algorithm induces a mapping between positions in two XML trees.

Throughout the section, we assume that the DTDs are non-recursive and that all FDs contain at most one element path on the left-hand side.

We start with an XNF normalization algorithm proposed in [3]. To present this algorithm we need to introduce some terminology. Given a DTD D and a set of FDs Σ, a non-trivial FD $X \to q.@l$ is called *anomalous*, over (D, Σ), if it violates XNF; that is, $X \to q.@l \in (D, \Sigma)^+$ but $X \to q \notin (D, \Sigma)^+$. The algorithm eliminates anomalous functional dependencies by using two basic steps: moving an attribute, and creating a new element type.

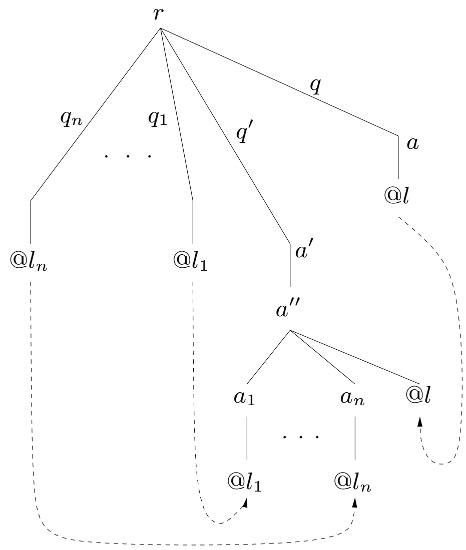
Moving attributes. Let $D = (L_0, P, R, r)$ be a DTD, $q.@l$ an attribute path in D, q' an element path in D and $@m$ an attribute not used in D. Assume that a, a' are the last elements of q and q', respectively. The DTD $D[q.@l := q'.@m]$ is constructed by moving the attribute $@l$ from the set of attributes of a to the set of attributes of a', and changing its name to $@m$, as shown in the following figure.



Formally, $D[p.@l := q.@m]$ is (L_0, P, R', r), where $R'(a') = R(a') \cup \{@m\}$, $R'(a) = R(a) - \{@l\}$ and $R'(b) = R(b)$ for each $b \in L_0 - \{a, a'\}$.

Given a set of FDs Σ over D, a set of FDs $\Sigma[q.@l := q'.@m]$ over $D[q.@l := q'.@m]$ consists of all FDs $X \to Y \in (D, \Sigma)^+$ with $X \cup Y \subseteq paths(D[q.@l := q'.@m])$.

Creating new element types. Let $D = (L_0, P, R, r)$ be a DTD, $q.@l \in paths(D)$, $S = \{q', q_1.@l_1, \ldots, q_n.@l_n\} \subseteq paths(D)$ such that $n \geq 1$ and q' is an element path in D. Assume that a, a' are the last elements of q and q', respectively. We construct a new DTD D' by creating a new element type a'' as a child of a', making $a_1, \ldots, a_n$ its children, $@l$ its attribute, and $@l_1, \ldots, @l_n$ attributes of $a_1, \ldots, a_n$, respectively. Furthermore, we remove $@l$ from the set of attributes of a, as shown in the following figure.



Formally, if $\{a'', a_1, \ldots, a_n\}$ are element types which are not in L_0, the new DTD, denoted by $D[q.@l := q'.a''[a_1.@l_1, \ldots, a_n.@l_n, @l]]$, is (L_0', P', R', r), where $L_0' = L_0 \cup \{a'', a_1, \ldots, a_n\}$ and P', R' are defined as follows.

1. Assume that $a' \to P_{a'} \in P$. Then, $P' = (P - \{a' \to P_{a'}\}) \cup \{a' \to (a'')^* P_{a'}, a'' \to a_1^* \cdots a_n^*, a_1 \to \epsilon, \ldots, a_n \to \epsilon\}$.
2. $R'(a'') = \{@l\}$, $R'(a_i) = \{@l_i\}$, for each $i \in [1, n]$, $R'(a) = R(a) - \{@l\}$ and $R'(b) = R(b)$ for each $b \in L_0 - \{a\}$.

Given $D' = D[q.@l := q'.a''[a_1.@l_1, \ldots, a_n.@l_n, @l]]$ and a set Σ of FDs over D, we define a set $\Sigma[q.@l := q'.a''[a_1.@l_1, \ldots, a_n.@l_n, @l]]$ of FDs over D' as the set that contains the following:

1. $X \to Y \in (D, \Sigma)^+$ with $X \cup Y \subseteq paths(D')$;

2. For each FD $X \to Y \in (D, \Sigma)^+$ with $X \cup Y \subseteq \{q', q_1, \ldots, q_n, q_1.@l_1, \ldots, q_n.@l_n, q.@l\}$, we include an FD obtained from it by changing q_i to $q'.a''.a_i$, $q_i.@l_i$ to $q.a''.a_i.@l_i$, and $q.@l$ to $q.a''.@l$;

3. $\{q',\ q'.a''.a_1.@l_1,\ \ldots,\ q'.a''.a_n.@l_n\} \to q'.a''$, and $\{q'.a'', q'.a''.a_i.@l_i\} \to q'.a''.a_i$ for $i \in [1, n]$.

The algorithm does not apply this transformation to an arbitrary FD, but rather to a *minimal* one. In the relational context, a minimal FD is $X \to A$ such that $X' \not\to A$ for any $X' \subsetneq X$. In the XML context the definition is a bit more complex to account for paths used in FDs. We say that $\{q, q_1.@l_1, \ldots, q_n.@l_n\} \to q_0.@l_0$ is (D, Σ)-minimal if there is no anomalous FD $X \to q_i.@l_i \in (D, \Sigma)^+$ such that $i \in [0, n]$ and X is a subset of $\{q, q_1, \ldots, q_n, q_0.@l_0, \ldots, q_n.@l_n\}$ such that $| X | \leq n$ and X contains at most one element path.

Given an XML specification (D, Σ), the normalization algorithm applies the two transformations until the schema is in XNF. The algorithm always terminates and produces a tree in XNF [3].

Let (D, Σ) be an XML specification and $T \in inst(D, \Sigma)$. Assume that (D, Σ) is not in XNF. Let (D', Σ') be an XML specification obtained by executing one step of the normalization algorithm. Every step of this algorithm induces a natural transformation on XML documents. One of the properties of the algorithm is that for each normalization step that transforms $T \in inst(D, \Sigma)$ into $T' \in inst(D', \Sigma')$, one can find a map $\pi_{T',T} : Pos(T') \to 2^{Pos(T)}$ that associates each position in the new tree T' with one or more positions in the old tree T, as shown below.

1. Assume that $D' = D[q.@l := q'.@m]$ and, therefore, $q' \to q.@l$ is an anomalous FD in (D, Σ). In this case, an XML tree T' is constructed from T as follows. For every $t \in tuples_D(T)$, define a tree tuple t' by using the following rule: $t'(q'.@m) = t(q.@l)$ and for every $q'' \in paths(D) - \{q.@l\}$, $t'(q'') = t(q'')$. Then, T' is an XML tree whose tree tuples are $\{t' \mid t \in tuples_D(T)\}$. Furthermore, positions in t' are associated to positions in t as follows: if $p' = (t'(q'), @m)$, then $\pi_{T',T}(p') = \{(t(q), @l)\}$; otherwise, $\pi_{T',T}(p') = \{p'\}$.

2. Assume that (D', Σ') was generated by considering a (D, Σ)-minimal anomalous FD $\{q',\ q_1.@l_1, \ldots,\ q_n.@l_n\} \to q.@l$. Thus, $D' = D[q.@l := q'.a[a_1.@l_1, \ldots, a_n.@l_n, @l]]$. In this case, an XML tree T' is constructed from T as follows. For every $t \in tuples_D(T)$, define a tree tuple t' by using the following rule: $t'(q'.a)$ is a fresh node identifier, $t'(q'.a.@l) = t(q.@l)$, $t'(q'.a.a_i)$ is a fresh node identifier ($i \in [1, n]$), $t'(q.a.q_i.@l_i) = t(q_i.@l_i)$ and for every $q'' \in paths(D) - \{q.@l\}$, $t'(q'') = t(q'')$. Then, T' is an XML tree whose tree tuples are $\{t' \mid t \in tuples_D(T)\}$. Furthermore, positions in t' are associated to positions in t as follows. If $p' = (t'(q'.a), @l)$, then $\pi_{T',T}(p') = \{(t(q), @l)\}$. If $p' = (t'(q'.a.a_i), @l_i)$, then $(t(q_i), @l_i) \in \pi_{T',T}(p')$ (note that in this case $\pi_{T',T}(p)$ may contain more than one position). For any other position p' in t', $\pi_{T',T}(p') = \{p'\}$.

Similarly to the relational case, we can now show the following.

Theorem 6. *Let T be a tree that conforms to a DTD D and satisfies a set of FDs Σ, and let $T' \in inst(D', \Sigma')$ result from T by applying one step of the normalization algorithm. Let $p' \in Pos(T')$. Then*

$$\textsc{Inf}_{T'}(p' \mid \Sigma') \quad \geq \quad \max_{p \in \pi_{T',T}(p')} \textsc{Inf}_T(p \mid \Sigma).$$

Just like in the relational case, one can define effective steps of the algorithm as those in which the above inequality is strict for at least one position, and show that (D, Σ) is in XNF iff no decomposition algorithm is effective in (D, Σ).

7. Conclusion

Our goal was to find criteria for good data design, based on the intrinsic properties of a data model rather than tools built on top of it, such as query and update languages. We were motivated by the justification of normal forms for XML, where usual criteria based on update anomalies or existence of lossless decompositions are not applicable until we have standard and universally acceptable query and update languages.

We proposed to use techniques from information theory, and measure the information content of elements in a database with respect to a set of constraints. We tested this approach in the relational case and showed that it works: that is, it characterizes the familiar normal forms such as BCNF and 4NF as precisely those corresponding to good designs, and justifies others, more complicated ones, involving join dependencies. We then showed that the approach straightforwardly extends to the XML setting, and for the case of constraints given by functional dependencies, equates the normal form XNF of [3] with good designs. In general, the approach is very robust: although we do not show it here due to space limitations, it can be easily adapted to the nested relational model, where it justifies a normal form NNF [20, 21].

Future work. It would be interesting to characterize 3NF by using the measure developed in this paper. So far, a little bit is known about 3NF. For example, as in the case of BCNF, it is possible to prove that the synthesis approach for generating 3NF databases does not decrease the amount of information in each position. Furthermore, given that 3NF does not necessarily eliminate all redundancies, one can find 3NF databases where the amount of information in some positions is not maximal.

We would like to consider more complex XML constraints and characterize good designs they give rise to. We also would like to connect this approach with that of [14], where information capacities of two schemas can be compared based on the existence of queries in some standard language that translate between them. For two classes of well-designed schemas (those with no constraints, and with keys only), being information-capacity equivalent means being isomorphic [2, 14], and we would like to see if this connection extends beyond the classes of schemas studied in [2, 14].

Acknowledgment We thank Pablo Barceló and Michael Benedikt for helpful comments.

8. References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] J. Albert, Y. Ioannidis, and R. Ramakrishnan. Equivalence of keyed relational schemas by conjunctive queries. *JCSS*, 58(3):512–534, 1999.

[3] M. Arenas and L. Libkin. A normal form for XML documents. In *PODS'02*, pages 85–96.

[4] C. Beeri, P. Bernstein, and N. Goodman. A sophisticate's introduction to database normalization theory. In *VLDB'78*, pages 113–124.

[5] J. Biskup. Achievements of relational database schema design theory revisited. In *Semantics in Databases*, LNCS 1358, pages 29–54. Springer-Verlag, 1995.

[6] R. Cavallo and M. Pittarelli. The theory of probabilistic databases. In *VLDB'87*, pages 71–81.

[7] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[8] M. Dalkilic and E. Robertson. Information dependencies. In *PODS'00*, pages 245–253.

[9] DBLP. `http://www.informatik.uni-trier.de/~ley/db/`.

[10] D. W. Embley and W. Y. Mok. Developing XML documents with guaranteed "good" properties. In *ER'01*, pages 426–441.

[11] R. Fagin. Multivalued dependencies and a new normal form for relational databases. *ACM TODS*, 2(3):262–278, 1977.

[12] R. Fagin. Normal forms and relational database operators. In *SIGMOD'79*, pages 153–160.

[13] R. Fagin. A normal form for relational databases that is based on domains and keys. *ACM TODS*, 6(3):387–415, 1981.

[14] R. Hull. Relative information capacity of simple relational database schemata. *SIAM J. Comput.*, 15(3):856–886, 1986.

[15] P. Kanellakis. *Elements of Relational Database Theory*, In *Handbook of TCS, vol. B*, pages 1075–1144. 1990.

[16] T. T. Lee. An information-theoretic analysis of relational databases - Part I: Data dependencies and information metric. *IEEE Trans. on Software Engineering*, 13(10):1049–1061, 1987.

[17] M. Levene and G. Loizou. Why is the snowflake schema a good data warehouse design? *Information Systems*, to appear.

[18] M. Levene and M. W. Vincent. Justification for inclusion dependency normal form. *IEEE TKDE*, 12(2):281–291, 2000.

[19] D. Maier, A. O. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. *ACM TODS*, 4(4):455–469, 1979.

[20] W.Y. Mok, Y. K. Ng, D. Embley. A normal form for precisely characterizing redundancy in nested relations. *ACM TODS* 21 (1996), 77–106.

[21] Z. M. Özsoyoglu, L.-Y. Yuan. A new normal form for nested relations. *ACM TODS* 12(1): 111–136, 1987.

[22] C. H. Papadimitriou. *Computational Complexity* Addison-Wesley, 1994.

[23] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 (Part I), 623–656 (Part II), 1948.

[24] D. Suciu. On database theory and XML. *SIGMOD Record*, 30(3):39–45, 2001.

[25] I. Tatarinov, Z. Ives, A. Halevy, and D. Weld. Updating XML. In *SIGMOD'01*, pages 413–424.

[26] V. Vianu. A Web Odyssey: from Codd to XML. In *PODS'01*, pages 1–15.

[27] M. W. Vincent. A corrected 5NF definition for relational database design. *TCS*, 185(2):379–391, 1997.

[28] M. W. Vincent. Semantic foundations of 4NF in relational database design. *Acta Informatica*, 36(3):173–213, 1999.