Representing Uncertain Data: Uniqueness, Equivalence, Minimization, and Approximation*

Anish Das Sarma, Shubha U. Nabar, Jennifer Widom
Stanford University
{anish,sunabar,widom}@cs.stanford.edu

Abstract

Many schemes have been proposed for representing uncertain data, where an uncertain database is defined as a set of *possible instances* for the database. Some important properties of representation schemes, such as *completeness* and *closure*, have already been considered. In this paper we identify several other interesting properties and obtain results for them with respect to a variety of different representation schemes. We first consider the *uniqueness* of representations, i.e., whether there is at most one representation for a set of possible instances in a particular representation scheme. For schemes that do not guarantee unique representations, we provide algorithms and complexity results for testing *equivalence* of representations. We also consider the problem of *minimizing* the size of the representation for a set of possible instances, obtaining a strong result for representation schemes comprised of tuples and constraints: minimizing the number of tuples also minimizes the size of constraints. We show the NP-hardness of minimizing a representation and study the more restricted problem of maintaining minimality when performing operations on the data. We present several results on the problem of *approximating* an uncertain database when we wish to use a simple (incomplete) representation scheme. Finally, we give an interesting result relating closure and completeness for uncertain databases.

1 Introduction

The field of *uncertain databases* is experiencing revived interest, e.g., [10, 17, 18, 19, 20, 48], with new perspectives revealing a number of interesting open problems. In general, a relation in an uncertain database represents a set of *possible instances* (sometimes called *possible worlds*) for the relation. Numerous representation schemes have been proposed to model such uncertain relations, e.g., [1, 2, 7, 20, 25, 26, 29, 30, 32, 36]. In this paper we identify and study several properties of representation schemes for uncertainty that have not been studied before. Suppose $\mathcal{S}$ is a representation scheme. The properties we study are:

- **Uniqueness:** Is there a unique representation in $\mathcal{S}$ for every representable set of possible instances?
- **Equivalence:** If not, how do we determine when two uncertain relations in $\mathcal{S}$ represent the same set of relation instances?
- **Minimization:** When we may have many equivalent representations, how do we define and find a "minimal" one?
- **Approximation:** When a scheme $\mathcal{S}$ is not expressive enough to represent a set of possible instances, how best can we "approximate" the instances using a representation in $\mathcal{S}$?

We will elaborate on these problems momentarily. First we briefly introduce some constructs that are used in the representation schemes we study.

Constructs for Uncertainty

An uncertain relation R is comprised of a set of *uncertain tuples*, or *u-tuples* for short. We consider two different constructs for u-tuples:

- *attribute-ors:* An attribute-or in a u-tuple specifies a set of alternative values for an attribute. For example:

{Thomas,Tom}	Poplar Ave.

  contains an attribute-or in its first field and represents one of two possible tuples: `(Thomas, Poplar Ave.)` or `(Tom, Poplar Ave.)`.

- *tuple-ors:* A tuple-or in a u-tuple specifies a set of possible tuples.[1] For example, the uncertainty in the previous example can also be represented by:

(Thomas, Poplar Ave.) || (Tom, Poplar Ave.)

Note that tuple-ors are strictly more expressive than attribute-ors: A u-tuple with multiple attribute-ors represents all possible combinations of attribute values, while a tuple-or can specify all combinations or only specific ones.

*This work was supported by the National Science Foundation under grant IIS-0324431, by a grant from the Boeing Corporation, and by a Stanford Graduate Fellowship from Sequoia Capital.

[1] Note: u-tuples with tuple-ors are equivalent to the *x-tuples* in [9].

In addition to attribute-ors and tuple-ors, we consider '?' annotations that denote the possible absence of a u-tuple, and more general *constraints* that relate the existence of u-tuples to other u-tuples. We assume the ordinary relations in possible instances may have duplicates, but we primarily assume no duplicate u-tuples in representations. We do present some interesting differences that arise in the presence of duplicate u-tuples.

Uniqueness and Equivalence

We will define a few representation schemes that are used throughout the paper, by combining the constructs above. We first analyze whether the different schemes guarantee unique representations for sets of relation instances. Hereafter we use the terms *unique* and *non-unique* representation schemes to distinguish the classes. Previous representation schemes for uncertain databases, e.g., [2, 26, 29, 32, 36], are generally very expressive and easily seen to be non-unique. The problem of uniqueness becomes significantly more interesting when we consider simpler representation schemes, which are relevant from a usability perspective [20].

For the non-unique schemes, we next address the problem of whether two uncertain relations represent the same set of instances. We analyze the time complexity of equivalence testing and give algorithms for the polynomial cases.

Minimization and Approximation

Since non-unique schemes may have many equivalent representations, we are interested in defining and identifying minimal representations for sets of possible instances. The non-unique schemes that we consider have u-tuples (of both types) and constraints. An important result we show is that minimizing the number of u-tuples in a representation also minimizes the size of the minimal constraint. This result simplifies the minimization problem, but minimizing arbitrary uncertain relations is NP-hard. We then study the problem of preserving minimality incrementally when performing operations on minimal representations.

The next problem we address is that of approximating an uncertain relation when we wish to use a simple representation scheme that cannot represent all sets of possible instances (i.e., that is *incomplete*). We first give an algorithm to determine whether a set of possible instances can be represented accurately in some of the simpler schemes we consider. If not, then approximation is required. We show that there are "bad cases" of sets of possible instances for which there is no constant-factor approximation (under the Jaccard similarity measure) in the simpler schemes we consider. We also define a "best" approximation, show that finding it is NP-hard, and give a polynomial time algorithm to find an approximate to the best approximation.

Closure and Completeness

A representation scheme is *complete* if it can represent all finite sets of possible instances. It is *closed* under an operation *Op* if it can represent the result of performing *Op* over any representation. Closure is weaker than completeness, and closure may be sufficient for some applications since they may control the initial data and which operations are performed over it. We show a new result connecting closure and completeness: Any representation scheme that can represent a certain minimal form of uncertainty (defined later) and that is closed under a small subset of relational operations (also defined later) is also complete.

Summary and Outline

The main results of the paper proceed as follows.

- In Section 3 we analyze uniqueness for a set of representation schemes and provide complexity results and algorithms for equivalence testing in the non-unique schemes.
- In Section 4 we show the useful result that in our non-unique schemes, minimizing the number of u-tuples minimizes the size of the minimal constraint. Though the general problem of minimization is NP-hard, we show how to preserve minimality incrementally when performing certain operations.
- In Section 5 we present several results on the approximation of uncertainty, such as an algorithm for determining exact representability in a particular scheme, hardness of finding a "best" approximation, and an algorithm for approximating the best approximation.
- In Section 6 we present a new result that connects closure of any representation scheme to completeness.

Some preliminaries are given in Section 2, related work is presented in Section 7, and we conclude with future work in Section 8. Due to space constraints, all theorems in the paper body are followed by proof sketches; full proofs can be found in the appendix.

2 Preliminaries

Definition 2.1 (Uncertain Relation)**.** An *uncertain relation* R defines a set of *possible instances*, $I(R) = \{I_1, I_2, \ldots, I_n\}$, where each I_j is a relation instance in the conventional sense. □

Operations on uncertain relations are defined based on their possible instances:

Definition 2.2 (Operations)**.** Consider uncertain relations $R_1, R_2, \ldots, R_n$, and an n-ary relational operator *Op*. The result of $Op\,(R_1, R_2, \ldots, R_n)$ is an uncertain relation R such that $I(R) = \{I \mid I = Op\,(I_1, I_2, \ldots, I_n),\ I_1 \in I(R_1), \ldots, I_n \in I(R_n)\}$. □

A *representation scheme* provides a way to represent uncertain relations, i.e., sets of possible instances, and we will see examples of representation schemes shortly. Two important properties of representation schemes are completeness and closure:

Definition 2.3 (Completeness)**.** A representation scheme $\mathcal{S}$ is said to be *complete* if any finite set of possible instances can be represented in $\mathcal{S}$. □

Definition 2.4 (Closure)**.** A representation scheme $\mathcal{S}$ is said to be *closed* under an operation *Op* if performing *Op* on uncertain relations in $\mathcal{S}$ always results in a set of possible instances that can be represented in $\mathcal{S}$. □

We now define a set of four representation schemes, $\mathcal{S}_{prop}$, $\mathcal{S}_{attr}$, $\mathcal{S}_{tuple}$, and $\mathcal{S}_2$, that we study in this paper. $\mathcal{S}_{prop}$ is complete, while $\mathcal{S}_{attr}$, $\mathcal{S}_{tuple}$, and $\mathcal{S}_2$ are incomplete. While many other schemes can be assembled out of the constructs that we consider (attribute-ors, tuple-ors, ?'s, and constraints; recall Section 1), we chose these four as a representative sample of the space of construct combinations, and because they serve to delineate different results for the problems we consider. Other complete schemes, such as the well-known *c-tables* [32], are similar to $\mathcal{S}_{prop}$ with respect to the problems considered in this paper.

2.1 $\mathcal{S}_{prop}$

Our first representation scheme, $\mathcal{S}_{prop}$, consists of u-tuples and constraints. The u-tuples in $\mathcal{S}_{prop}$ contain attribute-ors (recall Section 1; see also [22, 33, 38]) allowing a finite set of alternative values to be specified for individual attributes. The *existential constraints* in $\mathcal{S}_{prop}$ (considered earlier in the context of relations with nulls [16]) enable us to specify uncertainty about whether a u-tuple exists in a relation, or how its existence depends on the existence of other u-tuples. A relation in $\mathcal{S}_{prop}$ is represented formally by:

1. a multiset of u-tuples, $T = t_1, \ldots, t_n$
2. a boolean formula $f(T)$

The name $\mathcal{S}_{prop}$ is derived from the propositional formula comprising $f(T)$.

Definition 2.5 (Instances in $\mathcal{S}_{prop}$)**.** The possible instances $I(R)$ of an $\mathcal{S}_{prop}$ relation $R = (T, f)$ is a set of relations corresponding to all satisfying assignments for $f(T)$ and all possible choices of attribute-ors. Let σ be a satisfying assignment for $f(T)$. Then an instance in $I(R)$ is obtained by taking the set of u-tuples set to *true* in σ and picking one attribute value for each attribute-or in these u-tuples. □

For example, consider this $\mathcal{S}_{prop}$ relation:

	Name	Address
t_1	{Thomas, Tom}	Poplar Ave.
t_2	Alice	{Maine St., Main St.}

$$f(T) = t_1 \wedge t_2$$

Tuple t_1's Name is either `Thomas` or `Tom`, and tuple t_2 indicates that `Alice` lives on either `Maine` or `Main St`. The constraint f indicates that both t_1 and t_2 must be present. The possible instances of this relation are thus:

Name	Address
Thomas	Poplar Ave.
Alice	Maine St.

Name	Address
Thomas	Poplar Ave.
Alice	Main St.

Name	Address
Tom	Poplar Ave.
Alice	Maine St.

Name	Address
Tom	Poplar Ave.
Alice	Main St.

2.2 $\mathcal{S}_{attr}$

Like $\mathcal{S}_{prop}$, $\mathcal{S}_{attr}$ uses attribute-ors in its u-tuples, but it does not allow the specification of arbitrary constraints over the u-tuples. Instead, it allows us to specify uncertainty about whether a u-tuple is present: A '?' on a u-tuple denotes possible absence of the tuple. The possible instances of an $\mathcal{S}_{attr}$ relation correspond to all combinations of attribute values as in $\mathcal{S}_{prop}$. However, in place of considering all satisfying assignments as in $\mathcal{S}_{prop}$, we consider all combinations of absences of tuples labeled '?'.

Consider the example in the previous section. Suppose that instead of the formula f, tuple t_1 is labeled with a '?' and tuple t_2 has no label. Then the possible instances of this $\mathcal{S}_{attr}$ relation are:

Name	Address
Thomas	Poplar Ave.
Alice	Maine St.

Name	Address
Thomas	Poplar Ave.
Alice	Main St.

Name	Address
Tom	Poplar Ave.
Alice	Maine St.

Name	Address
Tom	Poplar Ave.
Alice	Main St.

Name	Address
Alice	Maine St.

Name	Address
Alice	Main St.

Notice that we can represent this same set of instances in $\mathcal{S}_{prop}$ by simply changing the constraint to $f(T) = t_2$.

2.3 $\mathcal{S}_{tuple}$

The next scheme we consider uses '?' labels as in $\mathcal{S}_{attr}$ but uses tuple-ors instead of attribute-ors (thus the names $\mathcal{S}_{tuple}$ versus $\mathcal{S}_{attr}$). The possible instances of an $\mathcal{S}_{tuple}$ relation are obtained by choosing at most one tuple alternate from each u-tuple labeled with a '?' and exactly one alternate from each unlabeled u-tuple. Consider the following $\mathcal{S}_{tuple}$ relation:

	Name, Address	
t_1	(Thomas, Poplar Ave.) || (Tom, Poplar Ave.)	?
t_2	(Alice, Maine St.) || (Alice, Main St.)	

This uncertain relation generates the same six instances as the example for $\mathcal{S}_{attr}$. As in this example, $\mathcal{S}_{attr}$ can often yield more compact representations than $\mathcal{S}_{tuple}$, but $\mathcal{S}_{tuple}$ is more expressive. We will later highlight some interesting results and differences for the two.

2.4 $\mathcal{S}_2$

$\mathcal{S}_2$ differs from the other schemes in that it uses ordinary tuples instead of u-tuples. Like $\mathcal{S}_{prop}$ it includes a boolean formula f over the tuples, however the formula must be in 2-CNF form (hence the name $\mathcal{S}_2$). As in Definition 2.5 for $\mathcal{S}_{prop}$, possible instances are determined by considering all satisfying assignments of f and collecting all tuples set to *true* in the assignment. For example, consider the $\mathcal{S}_2$ relation:

	Name	Address
t_1	Alice	Main St.
t_2	Alice	Maine St.

$f(T) = t_1 \oplus t_2$

where $\oplus$ denotes an exclusive-or. This relation has two possible instances:

Name	Address
Alice	Main St.

Name	Address
Alice	Maine St.

$\mathcal{S}_2$ is an interesting scheme to study for several reasons. $\mathcal{S}_2$ is closed under union, selection, and projection, and sometimes allows for better approximations than $\mathcal{S}_{attr}$ or $\mathcal{S}_{tuple}$. The primary reason is that $\mathcal{S}_2$ can encode tuple dependencies, whereas u-tuples in $\mathcal{S}_{attr}$ or $\mathcal{S}_{tuple}$ are independent of each other. Furthermore, $\mathcal{S}_2$ has tractable 2-CNF constraints as opposed to arbitrary constraints in $\mathcal{S}_{prop}$, making certain problems such as membership questions easier to answer.

3 Uniqueness and Equivalence

A natural question that arises in the context of our representation schemes is whether sets of possible instances have unique representations in the different schemes. We show in this section that while $\mathcal{S}_{attr}$ and $\mathcal{S}_{tuple}$ do guarantee unique representations, $\mathcal{S}_{prop}$ and $\mathcal{S}_2$ do not.

Theorem 3.1. Every set of possible instances representable in $\mathcal{S}_{tuple}$ or $\mathcal{S}_{attr}$ has a unique representation.

Proof Sketch: Recall that full proofs for all theorems in this paper are given in the online technical report [21]. This result is first shown for $\mathcal{S}_{tuple}$. Given any two distinct relations R_1 and R_2 in $\mathcal{S}_{tuple}$, we explicitly construct an instance of one of the relations that is not an instance of the other. Thus any two relations in $\mathcal{S}_{tuple}$ that represent the same set of instances must be identical. We then show that distinct $\mathcal{S}_{attr}$ relations map to distinct $\mathcal{S}_{tuple}$ relations, thus proving the uniqueness of $\mathcal{S}_{attr}$ in turn. □

Theorem 3.2. There exist sets of possible instances representable in $\mathcal{S}_2$ or $\mathcal{S}_{prop}$ that do not have unique representations.

Proof Sketch: For both representation schemes, we analyze the two possible sources of non-uniqueness: differences in tuple sets of relations that represent the same set of instances, and differences in constraints. In particular, we consider the following two questions:

Q1: Can two relations under a representation scheme have different sets of u-tuples but the same set of instances?

Q2: Consider two relations under a representation scheme having the same sets of u-tuples. Can these relations have non-equivalent constraints over the u-tuples, yet the same set of instances?

We show for $\mathcal{S}_2$ that the answer to Q1 is "yes" and to Q2 is "no". For $\mathcal{S}_{prop}$ the answer to both the questions is "yes". Hence neither scheme guarantees unique representations. □

We also consider variations on Theorem 3.2 such as allowing duplicate u-tuples in the representation, and a restricted form of $\mathcal{S}_2$ allowing only mutual-exclusion ($\oplus$) and mutual-inclusion ($\equiv$). We present results for all of these cases with respect to questions Q1 and Q2 in the technical report.

3.1 Equivalence Testing

The results in Section 3 showing non-uniqueness of $\mathcal{S}_{prop}$ and $\mathcal{S}_2$ lead next to the problem of testing, in these schemes, whether two relations are equivalent, i.e., whether they represent the same possible instances. ($\mathcal{S}_{attr}$ and $\mathcal{S}_{tuple}$ are unique so we need not consider the problem of equivalence testing for them.)

Theorem 3.3. The equivalence of two relations in $\mathcal{S}_2$ can be tested in polynomial time, while testing for equivalence in $\mathcal{S}_{prop}$ is NP-hard.

Proof Sketch: The hardness of equivalence testing in $\mathcal{S}_{prop}$ follows directly from the hardness of SAT. For equivalence testing in $\mathcal{S}_2$, we present a polynomial time algorithm. Given two $\mathcal{S}_2$ relations $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$:

1. Construct T_i' from T_i by eliminating all $t \in T_i$ such that setting t to *true* makes f_i unsatisfiable.
2. Construct f_i' from f_i by setting all variables in $T_i - T_i'$ to *false*.
3. If $T_1' \neq T_2'$ return $R_1 \not\equiv R_2$.

4. If $f_1' \not\equiv f_2'$ return $R_1 \not\equiv R_2$.

5. Return $R_1 \equiv R_2$.

The algorithm works by first eliminating tuples that are not present in any of the possible instances of the uncertain relations and modifying the formulas accordingly. We are thus left with minimal tuple sets for both relations. If the tuple sets are not the same at the end of this process then there is some tuple that is present in some instance of one relation that is not present in any instance of the other relation. If the formulas are not equivalent, then we use the Q2 "no" answer for $\mathcal{S}_2$ from the proof sketch of Theorem 3.2: two duplicate-free $\mathcal{S}_2$ relations with the same tuple sets but non-equivalent formulas cannot represent the same set of instances. □

We show in the technical report that if duplicate tuples are allowed in the representation, equivalence testing in $\mathcal{S}_2$ also becomes NP-hard.

4 Minimization of $\mathcal{S}_{prop}$

As seen in Section 3, $\mathcal{S}_{prop}$ and $\mathcal{S}_2$ do not guarantee unique representations for sets of possible instances. In this section we present results for $\mathcal{S}_{prop}$ only, since all of them apply to $\mathcal{S}_2$ as well. Two equivalent $\mathcal{S}_{prop}$ representations may in fact have arbitrarily different sizes. As an extremely simple example, consider the empty set of possible instances. This scenario could be represented in $\mathcal{S}_{prop}$ by a large unsatisfiable formula over many tuples, or by an empty set of tuples and an empty formula. In this section we study the *minimization* of $\mathcal{S}_{prop}$ relations. Recall that an $\mathcal{S}_{prop}$ relation consists of u-tuples T and a boolean formula $f(T)$. We define minimal in terms of the sum of the sizes of T and f.

Example 4.1. Consider the possible instances:

Name	Address
Alice	Maine St.

Name	Address
Alice	Main St.

One $\mathcal{S}_{prop}$ representation is to encode the mutual exclusion using a constraint:

	Name	Address
t_1	Alice	Main St.
t_2	Alice	Maine St.

$$f(T) = (t_1 \oplus t_2)$$

However, the minimal representation is to have one u-tuple with an attribute-or, and no constraint (i.e., $f = true$):

	Name	Address
t_1	Alice	{Main St., Maine St.}

□

Let (T, f) denote $\mathcal{S}_{prop}$ relations, $|T|$ denote the number of u-tuples, and let $size(f)$ denote the size of the formula measured as the number of clauses in the minimal CNF form. (Our results also hold if $size(f)$ denotes the number of literals.) Also, we write $R_1 \equiv R_2$ if R_1 and R_2 represent the same set of instances.

Definition 4.2 (Minimal $\mathcal{S}_{prop}$)**.** An $\mathcal{S}_{prop}$ relation $R = (T, f)$ is *minimal* if there does not exist another $\mathcal{S}_{prop}$ relation $R' = (T', f')$ with $R \equiv R'$ and $(|T'| + size(f')) < (|T| + size(f))$.

In terms of f, there exist practical techniques for minimization of propositional formulas (e.g., [34, 44, 40]), but in the worst case minimizing arbitrary formulas is NP-hard [28]. Moreover, we need to simultaneously minimize the sum of sizes of T and f: minimizing f for a given T need not give a minimal $\mathcal{S}_{prop}$ representation.

At first glance it may seem that finding the smallest representation requires a search over all possible sizes of T and f, and not just a search for the minimal $|T|$ or minimal $size(f)$. However, we will give an interesting result showing that for any $\mathcal{S}_{prop}$ relation, by minimizing $|T|$, we also minimize $size(f)$. This result holds for any representation scheme with a restricted set of propositional constraints, for example $\mathcal{S}_2$. We subsequently show how this result may be used to maintain minimality while performing certain operations in $\mathcal{S}_{prop}$.

4.1 Tuple-Minimality versus Constraint-Minimality

Theorem 4.3 (Tuple vs. Constraint Minimality)**.** Given $\mathcal{S}_{prop}$ relations $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$, if $R_1 \equiv R_2$ and $|T_1| > |T_2|$, then there exists an equivalent $\mathcal{S}_{prop}$ relation $R = (T, f)$ with $|T| < |T_1|$ and $size(f) \leq size(f_1)$. In other words, minimizing the number of u-tuples minimizes the size of the minimal constraint. □

Proof Sketch: The proof of this theorem is constructive: given R_1 and R_2, we give an algorithm to construct R satisfying the theorem. We use R_1 and R_2 to construct an independent set of u-tuples T, where $|T| < |T_1|$. We then show that there is an equivalent representation R with u-tuples T, and that there is a translation from R_1 to R involving a series of atomic operations that we call "splits" and "combines". We give bounds on the number of clauses these operations add or remove from f_1 based on the u-tuples that are split or combined. We also give the number of combines and splits required to translate from R_1 to R, and use a counting argument to show $size(f) \leq size(f_1)$. □

The theorem implies that if R has a minimal set of tuples T it also has minimal f, but it does not imply that if R_1 has fewer tuples than R_2, then f_1 can be made smaller than f_2. The following counterexample refutes the possibility that fewer tuples always implies fewer constraints:

Example 4.4. We illustrate $\mathcal{S}_{prop}$ relations $R_1 \equiv R_2$ with $|T_1| < |T_2|$ and in the minimal form $size(f_1) > size(f_2)$. Let T_1, T_2, f_1, and f_2 be:

	T_1
t_1	{a,b}
t_2	{c,d}
t_3	p
t_4	q
l_1	
$\vdots$	$\vdots$
l_m	

	T_2
t_1	a
t_2	b
t_3	c
t_4	d
t_5	{p,q}
l_1	
$\vdots$	$\vdots$
l_m	

$$f_1 = t_1 \wedge t_2 \wedge (\neg t_3 \vee \neg t_4) \wedge (t_3 \Rightarrow l_i) \wedge (t_4 \Rightarrow l_i), \text{ for all } i$$
$$f_2 = (t_1 \oplus t_2) \wedge (t_3 \oplus t_4) \wedge (t_5 \Rightarrow l_i), \text{ for all } i$$

$l_1, \ldots, l_m$ can be any set of distinct tuples. The constraints state that the presence of either p or q implies the presence of all of $l_1, \ldots, l_m$, in addition to other constraints on the t tuples. Choosing a sufficiently large m, we get $size(f_1) > size(f_2)$, and $R_1 \equiv R_2$ but $|T_1| < |T_2|$. □

Our nice result coupling minimality of tuples and constraints unfortunately does not hold when we allow duplicate u-tuples.

Lemma 4.5. Theorem 4.3 does not hold for $\mathcal{S}_{prop}$ relations with duplicates in the representation.

Full Proof: Consider $\mathcal{S}_{prop}$ relation R where $T = \{t_1$:(a), t_2:(b), t_3:(c), t_4:(x), t_5:(y), t_6:(z)$\}$ and $f = (t_1 \Rightarrow t_4) \wedge (t_2 \Rightarrow t_5) \wedge (t_3 \Rightarrow t_6) \wedge (\neg t_4 \vee \neg t_5 \vee \neg t_6)$. Intuitively, the presence of one (or two) a tuples in an instance implies the presence of at least one (or two respectively) x, y, or z tuples. Since there is a constraint that not all three of x, y, and z can appear in the same instance, no instance can have three a's. Therefore, there is an equivalent $\mathcal{S}_{prop}$ relation R' with $T' = \{s_1$:(a), s_2:(a), s_3:(x), s_4:(y), s_5:(z)$\}$ but f' requires more than 4 clauses. Therefore, we have $R \equiv R'$, $|T| > |T'|$, and $size(f) < size(f')$, yet there is no other $\mathcal{S}_{prop}$ representation for the same set of instances with the number of u-tuples and size of constraints being smaller than T and f respectively. □

4.2 Incremental Minimality

It can easily be shown that minimizing an arbitrary $\mathcal{S}_{prop}$ relation is NP-hard in general.[2] However, we can still use the result of Theorem 4.3 to our advantage: If we perform an operation on minimal $\mathcal{S}_{prop}$ relations, then minimizing the number of u-tuples in the result also minimizes the size of constraints. Thus, if we start with minimal relations, we can maintain minimality by always minimizing the number of u-tuples in the results of operations.

In general, even maintaining a minimal number of u-tuples while performing operations is a hard problem, but for certain operations we have efficient algorithms.

Theorem 4.6 (Incremental Minimality of $\mathcal{S}_{prop}$)**.** Given $\mathcal{S}_{prop}$ relations in minimal form, there exist polynomial time algorithms to compute their union or cross-product so that the resulting $\mathcal{S}_{prop}$ relation is minimal with respect to the number of tuples (and hence also constraints). □

We prove this theorem in the technical report, and also show why the most natural algorithms for selection, projection, and natural join fail to maintain minimality. A more detailed study of maintaining incremental minimality while performing operations is an interesting direction for future work.

5 Approximate Representations

Now that we have addressed uniqueness, equivalence, and minimality of representation schemes, let us consider approximation. Our primary motivation is that users may wish to view a simpler representation than $\mathcal{S}_{prop}$—one that does not include full propositional formulas—and furthermore they may not require a fully accurate representation of the possible instances. (After all, we are operating uncertainty to begin with!)

Example 5.1. Suppose we have three possible instances of R:

Name	Address
Thomas	Poplar Ave.

Name	Address
Alice	Main St.

Name	Address
Thomas	Poplar Ave.
Alice	Main St.

Suppose we would like to represent this uncertain relation in $\mathcal{S}_{tuple}$. There is no exact representation of R in $\mathcal{S}_{tuple}$, but the following is a reasonable approximation: It has all the instances of R and also the empty relation as an additional instance:

	Name, Address	
t_1	(Thomas, Poplar Ave.)	?
t_2	(Alice, Main St.)	?

□

In this section, we first explore approximations our using constraint-free representation schemes $\mathcal{S}_{attr}$ and $\mathcal{S}_{tuple}$, then we study approximation in $\mathcal{S}_2$.

5.1 Approximating in $\mathcal{S}_{attr}$ or $\mathcal{S}_{tuple}$

Since $\mathcal{S}_{attr}$ and $\mathcal{S}_{tuple}$ are incomplete, the first question to ask is: Given a set P of possible instances for a relation, how do we determine whether P can be represented in $\mathcal{S}_{attr}$ or $\mathcal{S}_{tuple}$? We first answer this question for $\mathcal{S}_{tuple}$, and then extend our result to $\mathcal{S}_{attr}$.

[2] The proof reduces an instance of the SAT problem to minimization of an $\mathcal{S}_{prop}$ relation, with variables of the SAT instance mapped to distinct tuples. The SAT instance is unsatisfiable iff the minimal $\mathcal{S}_{prop}$ relation has an empty set of tuples and $f \equiv false$.

Input: Set of Possible Instances P
Output: $\mathcal{S}_{tuple}$ representation R for P, if one exists.

1: Pad every instance in P with zero or more special tuples "*" so the number of tuples in each instance is equal to the maximum original instance cardinality.
2: Consider distinct tuples in P, in any order, say $t_1, \ldots, t_m$. Let the maximum multiplicity of t_i in any instance in P be m_i.
3: Consider t_1. Add m_1 tuple-ors to R, each containing t_1.
4: Look at the maximum number n of instances of t_1 that co-occur with m_2 instances of t_2, in some instance of P. If there is an $\mathcal{S}_{tuple}$ representation, then there are n tuple-ors to which t_2 can be added. Additionally, create $m_2 - n$ new tuple-ors to accommodate the remaining t_2's.
5: Continuing, for each t_i look at the maximum number of instances of t_1 through t_{i-1} that co-occur with m_i instances of t_i in some instance of P and determine the pre-existing tuple-ors to which t_i must be added, as well as the new tuple-ors that must be created. If no such addition of t_i is possible, exit returning "no $\mathcal{S}_{tuple}$ representation."
6: Remove "*" from all tuple-ors in R containing it, and annotate these tuple-ors with '?'.
7: Check to see if the possible instances of R exactly match the input P. If yes, return R, otherwise return "no representation".

Algorithm 1: $\mathcal{S}_{tuple}$-REP Algorithm

Theorem 5.2. Algorithm 1, $\mathcal{S}_{tuple}$-REP, gives the basic steps of a polynomial (in the size of P) algorithm that returns an exact representation of P in $\mathcal{S}_{tuple}$ whenever one exists. □

Proof Sketch: The algorithm looks at co-occurrences of tuples in the possible instances and from them attempts to construct an $\mathcal{S}_{tuple}$ relation. If at any point a tuple cannot be added to the $\mathcal{S}_{tuple}$ relation being constructed, then no $\mathcal{S}_{tuple}$ representation is possible and the algorithm exits. If all tuples are added successfully, then the result is an $\mathcal{S}_{tuple}$ representation of P if it correctly represents the possible worlds. If not, there is no representation. □

Theorem 5.3. Algorithm $\mathcal{S}_{tuple}$-REP for finding exact $\mathcal{S}_{tuple}$ representations can be modified (easily) to obtain $\mathcal{S}_{attr}$ representations. □

Full Proof: If there is no $\mathcal{S}_{tuple}$ representation, there is clearly no $\mathcal{S}_{attr}$ representation either. If there is an $\mathcal{S}_{tuple}$ representation R, we check if each u-tuple in R can be represented using attribute-ors (instead of tuple-ors). If we can convert all tuple-ors to attribute-ors, we have our $\mathcal{S}_{attr}$ representation. If there are u-tuples in R that cannot be converted to attribute-ors, there exists no $\mathcal{S}_{attr}$ representation. This result follows from the uniqueness of $\mathcal{S}_{tuple}$: If there is an $\mathcal{S}_{attr}$ representation not obtained by conversion of sets in R to u-tuples comprised only of attribute-ors, it can be mapped to a different $\mathcal{S}_{tuple}$ representation R', violating uniqueness. □

Now that we have seen how to find exact representations in $\mathcal{S}_{tuple}$, let us consider approximating a set of possible instances in $\mathcal{S}_{tuple}$ when no exact representation exists. Note that approximating in $\mathcal{S}_{tuple}$ is usually easier than in $\mathcal{S}_{attr}$, since $\mathcal{S}_{tuple}$ is more expressive than $\mathcal{S}_{attr}$. In the remainder of this subsection we consider only approximation in $\mathcal{S}_{tuple}$; extension of the results to $\mathcal{S}_{attr}$ is left as future work.

Consider a set of instances P and an $\mathcal{S}_{tuple}$ relation R with instances $I(R)$. We would like $I(R)$ to be as "close" to P as possible, for some notion of closeness between sets of possible instances. First we state a result showing that it is not always possible to find good approximations.

Lemma 5.4. There exists an $\mathcal{S}_{prop}$ relation R for which there is no constant factor $\mathcal{S}_{tuple}$-approximation R' under the Jaccard measure of similarity between $I(R)$ and $I(R')$. □

Example 5.5. We give the intuition behind this result with an example set of possible instances that have no constant factor approximation in $\mathcal{S}_{tuple}$. Consider n instances $\{P_1, P_2, \ldots, P_n\}$ for a relation with two attributes. Let the ith instance have two tuples: t_i^1:(i, 1) and t_i^2:(i, 2).

This set of instances does not admit any constant factor approximation R in $\mathcal{S}_{tuple}$: If $I(R)$ contains all n of the possible instances, then R would need to have $O(n^2)$ instances in total. □

Though we see that there are cases when no good approximations exist, we would still like to do as well as possible. A "best" approximation R of P can be defined in several ways:

- *Maximal Subset:* maximal R such that $I(R) \subseteq P$
- *Minimal Superset:* minimal R such that $I(R) \supseteq P$
- *Closest Set:* R such that $I(R)$ is as close to P as possible (using a measure of closeness like Jaccard similarity).

For this paper, we consider the minimal superset definition so as to preserve the possible instances. Results analogous to ours are likely to hold for other notions of best approximation as well, and we plan to explore this topic as future work. We use $|R|$, the number of u-tuples, as our metric for minimality of R. Further, we restrict ourselves to R where for each t, the number of u-tuples in R containing t is equal to the maximum multiplicity of t in any possible instance of P. We impose this condition to restrict the "width" (i.e., number of possibilities) in each u-tuple of R.[3]

Lemma 5.6. There always exists some R in $\mathcal{S}_{tuple}$ such that $I(R) \supseteq P$.

[3]Without this condition, a trivial best approximation is obtained by including n u-tuples, each having all possible tuples in R, where n is the maximum cardinality of any instance in P.

Full Proof: R is constructed by including in R a tuple-or s for every tuple t appearing in some instance of P, with multiplicity equal to the maximum number of times it appears in any possible instance. Annotating each tuple-or with '?', we get $I(R) \supseteq P$. □

Of course the construction from this proof may give a very poor approximation, with $I(R)$ being much larger than P. We have the following result for finding better approximations:

Theorem 5.7 (Best $\mathcal{S}_{tuple}$ Approximation for P)**.**

1. Finding the best $\mathcal{S}_{tuple}$ approximation for an arbitrary set of possible instances P is NP-hard.
2. There exists an algorithm, that given P, constructs a 5/7-differential approximation in polynomial time. □

Proof Sketch: The proof of hardness is shown by a reduction from the NP-hard minimum graph coloring problem. We then show that there is an inverse reduction to the graph coloring problem that can be 5/7-differentially approximated. □

Approximating an $\mathcal{S}_{prop}$ Representation

Given an arbitrary $\mathcal{S}_{prop}$ relation, the trivial approximation obtained by simply eliminating all constraints works, but may not give a good solution. Finding the best $\mathcal{S}_{tuple}$ representation for an arbitrary $\mathcal{S}_{prop}$ relation is intractable, but we can use Theorem 5.7: A brute-force approach is to construct all possible instances from the representation and then proceed as above. However, if we are able to construct all pairwise conflicts among u-tuples in the $\mathcal{S}_{prop}$ relation, then we can use the algorithm of Theorem 5.7 without actually constructing the possible instances.

The next section considers approximating $\mathcal{S}_{prop}$ in $\mathcal{S}_2$. More algorithms for constructing approximations of one representation scheme in another in general suggests an interesting direction for future work.

5.2 Approximating in $\mathcal{S}_2$

We briefly look at the problem of approximating $\mathcal{S}_{prop}$ relations in using $\mathcal{S}_2$. as an approximate representation for an uncertain relation. An arbitrary $\mathcal{S}_{prop}$ relation R can be approximated as $\mathcal{S}_2$ relations R_1 and R_2 such that $I(R_1) \subseteq I(R)$ and $I(R_2) \supseteq I(R)$, using techniques developed for theory approximation [35]. In general finding best approximations for propositional theories into tractable classes can be hard, but [35] gives algorithms for finding lower and upper bounds on a general theory in an online fashion. Their techniques are applicable for a class of tractable theories, including 2-CNF. We use these algorithms for approximation of $\mathcal{S}_{prop}$ representations. Applying the algorithms directly gives representations with 2-clauses over u-tuples. However, we can first translate an $\mathcal{S}_{prop}$ relation S into an equivalent $\mathcal{S}_{prop}$ relation S' with ordinary tuples and constraints over them. We then use the algorithms from [35] on S' to progressively get better approximations R_1 and R_2 for S' (and hence S).

6 Closure versus Completeness

Recall from Section 2 that a representation scheme $\mathcal{S}$ is complete if it can represent all finite sets of possible instances, and it is closed under an operation *Op* if it can represent the result of performing *Op* on any relations represented in $\mathcal{S}$.

Clearly, every complete representation scheme $\mathcal{S}$ is also closed under all relational algebra operations, since every operation generates a finite set of possible instances, which by completeness can be represented in $\mathcal{S}$. The converse, however, is not true in general: for example, $\mathcal{S}_{tuple}$ and $\mathcal{S}_{attr}$ are incomplete, yet they are closed under some operations such as projection and cross-product. One can certainly argue that closure is more important than completeness for real applications. We show an interesting result that any scheme $\mathcal{S}$ that is expressive enough to represent a very basic form of uncertainty, and that is closed under a small set of operations, can represent all finite sets of possible instances of an uncertain relation, i.e., $\mathcal{S}$ is complete.

Theorem 6.1 (Closure $\Rightarrow$ Completeness)**.**
Consider a representation scheme $\mathcal{S}$ with the following properties:

- **Basic Uncertainty:** $\mathcal{S}$ can represent all regular (certain) databases, and also can represent a unary uncertain relation R with two possible instances `P1:(1)` and `P2:(2)`.
- **Minimal Closure:** $\mathcal{S}$ is closed under one of the following sets of operations: (a) $\{\Pi, \bowtie\}$, or (b) $\{\sigma, \Pi, \times\}$.

Then $\mathcal{S}$ is a complete representation scheme. □

Proof Sketch: We show that any set of possible instances P can be constructed by applying the minimal sets of operations on regular databases and R. We use the uncertainty in `P1` versus `P2` to construct the number of possible instances in P, and then combine this uncertainty with regular relations in $\mathcal{S}$ to obtain P. □

7 Related Work

The study of uncertain databases has a long history, dating back to a series of initial papers in the early 1980's, e.g., [8, 12, 32, 47], and a great deal of follow-on work, e.g., [7, 18, 25, 32, 39, 36, 37, 41]. Much of this previous work lays theoretical foundations and considers query answering, e.g., [1, 2, 30, 47]. Systems based around uncertain data are discussed in [10, 36, 48].

The specific properties and problems addressed in most of this previous work are centered around completeness, closure, and membership questions. We are not aware of any previous work that focuses on the problems we consider in this paper: uniqueness, equivalence, minimization, and approximation.

Like uncertain databases, the related area of *probabilistic databases* has been experiencing revived interest, especially in query answering [17, 18, 19]. Work in probabilistic databases also has not, to the best of our knowledge, considered the problems we address in this paper. The representation schemes we consider in this paper do not include probability distributions; revisiting and extending our results to the probabilistic case is an important direction of future work.

Another related area is *inconsistent databases*, e.g., [3, 5, 6, 11, 15, 23, 31, 49], in which the possible "minimal repairs" [6, 13, 49] to an inconsistent database result in a set of possible instances (i.e., an uncertain database). Reasoning with uncertainty in the Artificial Intelligence context is similarly related, e.g., [24, 43, 45]. Again, our work in this paper focuses on specific problems associated with representation schemes for uncertainty, and we have not seen these problems addressed in the related areas.

Approximate query answering, and obtaining ranked results to imprecisely defined queries, is also an active area of research, e.g., [4, 27, 25, 46]. This body of work should not be confused with ours: we look at modeling uncertainty and querying it exactly, as opposed to modeling exact data and querying it approximately.

The problems addressed in this paper arose in the context of the *Trio* project at Stanford, whose objective is to develop a system that fully integrates data, uncertainty, and lineage [48]. In an initial Trio paper on models for uncertainty [20], we introduced numerous representation schemes and considered their completeness, closure, and relative expressiveness. Although the approximation problem was suggested in that paper, it was not solved, and the other problems we consider here—uniqueness, equivalence, and minimization—were not discussed at all. In a separate paper, we consider several aspects of integrating data lineage together with uncertainty [9], which is the ultimate goal of the Trio project.

8 Conclusions and Future Work

This paper addressed a number of problems arising from representation schemes for uncertain data. These problems have not been studied before to the best of our knowledge. We introduced four different schemes by assembling basic constructs for uncertainty. Using these schemes, we explored uniqueness, equivalence, minimization, and approximation, obtaining complexity results and providing algorithms for tractable cases. We also gave a new result connecting closure and completeness for a broad class of representation schemes.

The results in this paper suggest a number of areas for future work:

- As mentioned in Section 7, the area of probabilistic databases is of great interest. Uncertain databases can be extended to include probabilities [7, 10, 14, 27, 36], so extending the definitions and results in this paper poses a set of interesting open problems.
- We considered the problem of maintaining minimality incrementally for only certain schemes and operations (Section 4). The problem remains open for other operations and schemes, as well as for representations with duplicates.
- Likewise, for the problem of approximating an uncertain relation in a simple representation scheme, we considered primarily $\mathcal{S}_{tuple}$ (Section 5). We have not yet considered approximations in $\mathcal{S}_{attr}$ in any detail, nor other simple representation schemes that may be appealing to users [20].
- In this paper, we considered primarily the problem of approximating a set of possible instances when a representation scheme is not expressive enough. A practical problem for future work is that of developing algorithms that directly convert uncertain relations in an expressive representation scheme (such as $\mathcal{S}_{prop}$) into approximations in weaker schemes.
- We concluded with an interesting result showing that closure under a small set of operations implies completeness. An interesting open problem is to fully characterize the gap between closure and completeness, i.e., to come up with all minimal sets of operations for which closure implies completeness.

Acknowledgments

We are grateful to Omar Benjelloun, Alon Halevy, and the entire Trio group for inspiring discussions leading to the results in this paper.

References

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

[2] S. Abiteboul, P. Kanellakis, and G. Grahne. On the Representation and Querying of Sets of Possible Worlds. *Theoretical Computer Science*, 1991.

[3] S. Agarwal, A. M. Keller, G. Wiederhold, and K. Saraswat. Flexible Relation: An Approach for Integrating Data from Multiple, Possibly Inconsistent Databases. In *ICDE*, 1995.

[4] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated Ranking of Database Query Results. In *Proc. of CIDR*, 2003.

[5] M. Arenas, L. Bertossi, and J. Chomicki. Answer sets for consistent query answering in inconsistent databases. *Theory and Practice of Logic Programming*, 2003.

[6] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent Query Answers in Inconsistent Databases. In *Proc. of ACM PODS*, 1999.

[7] D. Barbará, H. Garcia-Molina, and D. Porter. The Management of Probabilistic Data. *IEEE Trans. Knowl. Data Eng.*, 1992.

[8] R. S. Barga and C. Pu. Accessing Imprecise Data: An Approach Based on Intervals. *IEEE Data Engineering Bulletin*, 1993.

[9] O. Benjelloun, A. Das Sarma, A. Halevy, and J. Widom. The Symbiosis of Lineage and Uncertainty, http://dbpubs.stanford.edu/pub/2005-39. *Under Submission to PODS 2006*.

[10] J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu. MYSTIQ: a system for finding more answers by using probabilities. In *Proc. of ACM SIGMOD*, 2005.

[11] F. Bry. Query answering in information systems with integrity constraints. In *Proceedings of the IFIP TC11 Working Group 11.5, First Working Conference on Integrity and Internal Control in Information Systems*, 1997.

[12] B. P. Buckles and F. E. Petry. A Fuzzy Model for Relational Databases. *International Journal of Fuzzy Sets and Systems*, 1982.

[13] A. Cali, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of ACM PODS*, 2003.

[14] R. Cavallo and M. Pittarelli. The theory of probabilistic databases. In *Proc. of VLDB*, 1987.

[15] J. Chomicki and J. Marcinkowski. Minimal-change integrity maintenance using tuple deletions.

[16] E. F. Codd. Extending the Database Relational Model to Capture More Meaning. *ACM Transactions on Database Systems*, 1979.

[17] N. Dalvi, G. Miklau, and D. Suciu. Asymptotic Conditional Probabilities for Conjunctive Queries. In *Proc. of ICDT*, 2005.

[18] N. Dalvi and D. Suciu. Efficient Query Evaluation on Probabilistic Databases. In *Proc. of VLDB*, 2004.

[19] N. Dalvi and D. Suciu. Answering Queries from Statistics and Probabilistic Views. In *Proc. of VLDB*, 2005.

[20] A. Das Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working Models for Uncertain Data (to appear). In *Proc. of ICDE*, April 2006.

[21] A. Das Sarma, S. U. Nabar, and J. Widom. Representing Uncertain Data: Uniqueness, Equivalence, Minimization, and Approximation. http://dbpubs.stanford.edu/pub/2005-38.

[22] L. G. DeMichiel. Resolving Database Incompatibility: An Approach to Performing Relational Operations over Mismatched Domains. *IEEE Transactions on Knowledge and Data Engineering*, 1989.

[23] P. M. Dung. Integrating data from possibly inconsistent databases. In *COOPIS '96: Proceedings of the First IFCIS International Conference on Cooperative Information Systems*, 1996.

[24] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proc. of IJCAI*, 1999.

[25] N. Fuhr. A Probabilistic Framework for Vague Queries and Imprecise Information in Databases. In *Proc. of VLDB*, 1990.

[26] N. Fuhr and T. Rölleke. A Probabilistic NF2 Relational Algebra for Imprecision in Databases. *Unpublished Manuscript*, 1997.

[27] N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM TOIS*, 1997.

[28] M. R. Garey and D. S. Johnson. Computers and Intractability. *W. H. Freeman and Company*, 1979.

[29] G. Grahne. Dependency Satisfaction in Databases with Incomplete Information. In *Proc. of VLDB*, 1984.

[30] G. Grahne. Horn Tables - An Efficient Tool for Handling Incomplete Information in Databases. In *Proc. of ACM PODS*, 1989.

[31] G. Greco, S. Greco, and E. Zumpano. A logical framework for querying and repairing inconsistent databases. *IEEE Transactions on Knowledge and Data Engineering*, 2003.

[32] T. Imielinski and W. Lipski Jr. Incomplete Information in Relational Databases. *Journal of the ACM*, 1984.

[33] T. Imielinski, S. Naqvi, and K. Vadaparty. Incomplete objects - data model for design and planning applications. In *Proc. of ACM SIGMOD*, 1991.

[34] M. Karnaugh. The map method for synthesis of combinational logic circuits. *Trans. AIEE. pt I*, 1953.

[35] H. Kautz and B. Selman. Knowledge Compilation and Theory Approximation. *Journal of the ACM*, 1996.

[36] L. V. S. Lakshmanan, N. Leone, R. Ross, and V.S. Subrahmanian. ProbView: A Flexible Probabilistic Database System. *ACM TODS*, 1997.

[37] S. K. Lee. An extended Relational Database Model for Uncertain and Imprecise Information. In *Proc. of VLDB*, 1992.

[38] L. Libkin and L. Wong. Semantic representations and query languages for or-sets. In *Proc. of ACM PODS*, 1993.

[39] K. Liu and R. Sunderraman. Indefinite and Maybe Information in Relational Databases. *ACM TODS*, 1990.

[40] E. J. McCluskey. Minimization of boolean functions. *The Bell System Technical Journal*, 1956.

[41] A. Motro. Management of Uncertainty in Database Systems. *Modern Database Systems: The Object Model, Interoperability, and Beyond*, 1994.

[42] V. Th. Paschos. Polynomial approximation and graph-coloring. *Computing*, 2003.

[43] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[44] W. Quine. The problem of simplifying truth functions. *American Math Monthly*, 1952.

[45] S. Sanghai, P. Domingos, and D. Weld. Dynamic probabilistic relational models. In *Proc. of IJCAI*, 2003.

[46] A. Theobald and G. Weikum. The XXL Search Engine: Ranked Retrieval of XML Data Using Indexes and Ontologies. In *Proc. of ACM SIGMOD*, 2002.

[47] M. Y. Vardi. Querying logical databases. In *Proc. of ACM PODS*, 1985.

[48] J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In *Proc. of CIDR*, 2005.

[49] J. Wijsen. Condensed representation of database repairs for consistent query answering. In *Proc. of ICDT*, 2002.

A Uniqueness and Equivalence Testing Proofs

A.1 Proof of Uniqueness of $\mathcal{S}_{attr}$ and $\mathcal{S}_{tuple}$

Proof. We consider each of the models separately.

$\mathcal{S}_{tuple}$**:** Consider two relations R_1 and R_2 in $\mathcal{S}_{tuple}$with $I(R_1) = I(R_2)$. Let $\{r_1, \ldots, r_m\}$ and $\{r'_1, \ldots, r'_n\}$ be their tuple-ors. Note that $m = n$. This is because, if m were less than n, there would be some instance in $I(R_2)$ with n tuples, while the maximum number of tuples in any instance of $I(R_1)$ is m, leading to a contradiction. Similarly, m cannot be greater than n.

We now need to show that $\forall i, \exists j \; s.t. \; r_i = r'_j$. Suppose this is not true - then we can construct an instance, $\mathcal{I}$, in $I(R_1)$ that is not in $I(R_2)$, arriving at a contradiction. Let T_1 be the set of all r_i for which there is some r'_j such that $r_i = r'_j$ and let T_2 be the remaining tuple-ors in R_1. Similarly, define S_1 and S_2 over the tuple-ors in R_2. Thus $T_1 = S_1$ and $|T_2| = |S_2|$.

Choose $r_i \in T_2$ and $r'_j \in S_2$. Without loss of generality, assume that there is some tuple t in r_i that is not in r'_j. Let x tuple-ors in R_1 contain t. Then for each of these x tuple-ors, include t in the instance $\mathcal{I}$. There must be x corresponding tuple-ors in R_2 that must be set to t in order to produce the instance $\mathcal{I}$. Note that an equal number of tuple-ors from T_1 and S_1 would be set to t and so also, an equal number of tuple-ors from T_2 and S_2. Moreover, if a set in T_1 is set to t, then the corresponding set that is equal to it in S_1 must also be set to t.

At the end of this process no tuple from r'_j has been chosen since r'_j does not contain t. So we next look at r'_j and another tuple-or r_k from T_2 such that $r'_j \neq r_k$ and no tuple from r_k has yet been chosen to be included in the instance. We include some tuple u that occurs in r'_j but not in r_k (or vice versa) in the instance $\mathcal{I}$, and set all other tuple-ors that contain u in R_1 and R_2 to u. We carry on in this manner until finally we are left with only one tuple-or in each of T_2 and S_2 for which tuples have not yet been chosen. Let v be a tuple that is present in the remaining tuple-or from T_2 but not in the tuple-or from S_2. Let there be y tuple-ors remaining in R_1 that contain v. Every tuple-or in T_1 that contains v has a corresponding tuple-or in S_1 that contains v. Thus from here on, it would be possible to construct $\mathcal{I} \in I(R_1)$ that contains y instances of the tuple v whereas only $y-1$ tuple-ors in R_2 can be set to v. Thus $\mathcal{I} \in I(R_1)$ and $\mathcal{I} \notin I(R_2)$.

This shows that R_1 and R_2 must have the same set of tuple-ors. Moreover, the '?' annotations on the tuple-ors must be the same and to show this we can repeat the above proof by removing the labels and instead adding '?' to every labeled tuple-or as a special alternate tuple. If the resulting instance $\mathcal{I}$ contains any '?' tuples then we just discard these tuples and the resulting instance is the one that can be found in one relation but not the other.

$\mathcal{S}_{attr}$**:** Every $\mathcal{S}_{attr}$ relation can easily be converted to an equivalent $\mathcal{S}_{tuple}$ relation. Each u-tuple t in the $\mathcal{S}_{attr}$ relation is converted to a tuple-or r in the $\mathcal{S}_{tuple}$ relation by choosing all possible combinations of attributes from the attribute-ors in t as alternate tuples for r. A '?' tag for a u-tuple in $\mathcal{S}_{attr}$ becomes a '?' tag for the corresponding tuple-or. Under this transformation, two syntactically different $\mathcal{S}_{attr}$ relations give two distinct $\mathcal{S}_{tuple}$ relations. Hence by the uniqueness result in $\mathcal{S}_{tuple}$, $\mathcal{S}_{attr}$ also has a unique representation for every representable set of possible instances. □

A.2 Proof of Non-Uniqueness of $\mathcal{S}_2$, $\mathcal{S}_{prop}$ and $\mathcal{S}_{\oplus\equiv}$

Recall that we prove non-uniqueness of the representation schemes by showing that answer to one of the following questions for each of the schemes is yes:

Q1: Can two relations under a representation scheme have different sets of u-tuples but the same set of instances?

Q2: Consider two relations under a representation scheme having the same sets of u-tuples. Can these relations have non-equivalent constraints over the u-tuples, yet the same set of instances?

The table below summarizes the answers to the two questions for $\mathcal{S}_2$ and $\mathcal{S}_{prop}$. We consider both cases where the relations contain duplicates and where they don't. In addition, we consider a restricted version of $\mathcal{S}_2$ where only mutual-exclusion ($\oplus$) or mutual inclusion ($\equiv$) constraints are allowed between tuples. We call this restricted representation scheme $\mathcal{S}_{\oplus\equiv}$.

	Are there duplicates?	$\mathcal{S}_2$	$\mathcal{S}_{prop}$	$\mathcal{S}_{\oplus\equiv}$
Q1	No	Yes[a]	Yes[c]	No[e]
Q1	Yes	Yes[a]	Yes[c]	No[e]
Q2	No	No[b]	Yes[g]	No[b]
Q2	Yes	Yes[d]	Yes[d]	Yes[f]

Table 1: Uniqueness

The superscript in the table above points to the proof for that entry.

Proof. a: Consider two relations $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$. Let T_1 contain $t_1 = (a)$ and $t_2 = (b)$ and T_2 contain $s_1 = (a)$. Now if $f_1 = t_1 \wedge \neg t_2$ and $f_2 = s_1$, then $I(R_1) = I(R_2)$.

b: Consider two relations $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$. Let $T_1 = T_2$. If $f_1 \neq f_2$ then there must be some assignment, σ, of 1s and 0s to the tuples of the relations such that $f_1(\sigma) = 1$ and $f_2(\sigma) = 0$ or vice-versa. Since T_1 and T_2 do not contain duplicates, the instance represented by σ in $I(R_1)$ cannot be obtained by any other assignment of truth

values to variables in T_1 and hence in T_2. Thus $I(R_1) \neq I(R_2)$.

c: Consider two relations $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$. Let T_1 contain one tuple, $t_1 = (\{a, b\})$ with $f_1 = t_1$. Let T_2 contain two tuples, $s_1 = (a)$, $s_2 = (b)$ with $f_2 = (s_1 \vee s_2) \wedge (\neg s_1 \vee \neg s_2)$. Then $I(R_1) = I(R_2)$.

d: Consider two relations, $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$ with $T_1 = T_2$ containing the tuples $t_1 = (a)$, $t_2 = (a)$, $t_3 = (a)$ and $t_4 = (b)$. Let $f_1 = (t_1 \vee t_2) \wedge (\neg t_1 \vee \neg t_2) \wedge (t_2 \vee t_3) \wedge (\neg t_2 \vee \neg t_3) \wedge (t_3 \vee t_4) \wedge (\neg t_3 \vee \neg t_4)$ and $f_2 = (t_1 \vee t_2) \wedge (\neg t_1 \vee \neg t_2) \wedge (t_3 \vee t_4) \wedge (\neg t_3 \vee \neg t_4)$. f_1 and f_2 are not equivalent, but $I(R_1) = I(R_2)$.

e: Let T_1 and T_2 be the tuple sets of two relations R_1 and R_2 that have the same set of possible instances. We show that every tuple t that belongs to T_1 occurs with the same multiplicity in T_2 and vice versa. So in order to get a contradiction, let us assume that t occur m times in T_1 and n times in T_2 and let m be greater than n. Now look at the instance, $\mathcal{I}$, in $I(R_1)$ that contains t the most number of times. In particular, let t occur r times in this instance. This means that some instance in $I(R_2)$ also contains r ts. Consider the assignment of 1s or 0s to the boolean variables corresponding to the tuples in R_2 that produced this instance. Inverting this assignment would also satisfy the constraints of the relation and would produce an instance containing $n - r$ ts. So there must be an instance in $I(R_1)$ that also contains $n - r$ ts. Now consider the assignment of 1s and 0s to the boolean variables corresponding to the tuples in R_1 that produced this instance. Once again, inverting this assignment would satisfy the constraints of R_1 and would produce an instance containing $m - n + r$ ts. But $m - n + r > r$ since $m > n$ and hence $\mathcal{I}$ cannot be the instance in $I(R_1)$ containing the most number of ts resulting in a contradiction.

f: Consider two relations, $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$ with $T_1 = T_2$ containing the tuples $t_1 = (a)$, $t_2 = (a)$, $t_3 = (a)$ and $t_4 = (b)$. Let $f_1 = (t_1 \oplus t_2) \wedge (t_2 \oplus t_3) \wedge (t_3 \oplus t_4)$ and $f_2 = (t_1 \oplus t_2) \wedge (t_3 \oplus t_4)$. f_1 and f_2 are not equivalent, but $I(R_1) = I(R_2)$.

g: Consider tuple sets $T_1 = T_2$ with three u-tuples $t_1 = (\{a, b\})$, $t_2 = (\{b, c\})$ and $t_3 = (\{c, a\})$. Let $f_1 = ((t_1 \oplus t_2) \wedge \neg t_3)$ and $f_2 = ((t_1 \oplus t_3) \wedge \neg t_2)$. It can then be seen that $I(R_1) = I(R_2)$ where $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$.

□

A.3 Proof of Theorem 3.3

Proof. We consider each of the models in turn:

$\mathcal{S}_2$ **without duplicates:** Consider two relations $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$ in $\mathcal{S}_2$. We first modify T_1, T_2, f_1 and f_2 to create T_1', T_2', f_1' and f_2' respectively such that $T_1' = T_2'$, $I(T_1', f_1') = I(T_1, f_1)$ and $I(T_2', f_2') = I(T_2, f_2)$. The problem then reduces to testing for the equivalence of f_1' and f_2'. In order to make the two tuple sets equal, we remove tuples and make changes to the formulas as follows: If a tuple t is present in T_1, but not not in T_2, we check to see if there is any instance of R_1 that contains t. This is easily accomplished by setting the variable corresponding to t to `True` in f_1 and checking to see if f_1 is satisfiable. Since f_1 is in 2-CNF form - this can be checked in polynomial time. If the formula remains satisfiable, then R_1 and R_2 cannot be equivalent since no instance of R_2 can contain the tuple t. On the other hand if the formula is no longer satisfiable, it means that no instance of R_1 contains t and t can be removed from T_1. f_1 is modified by removing the variable corresponding to t from it. Similarly tuples in T_2 that are not present in T_1 can also be removed. In the end, we are left with two equal tuple sets and two 2-CNF formulas f_1' and f_2' whose equivalence can be tested in polynomial time.

$\mathcal{S}_2$ **with duplicates:** We prove that testing for the equivalence of two relations in $\mathcal{S}_2$ when the relations contain duplicates is NP-hard via a reduction from Vertex Cover. Given a graph, $G = (V, E)$, we would like to determine if the graph contains a vertex cover of size k. For any such graph, we construct a relation R_1 in $\mathcal{S}_2$ by creating a tuple $t_i = (x)$ for each vertex $v_i \in V$. For each edge $e_{ij} \in E$, we add a constraint to the formula of the form $(t_i \vee t_j)$. We then construct another relation R_2 in $\mathcal{S}_2$ with $|V|$ tuples $t_1 = t_2 = \ldots = t_{|V|} = (x)$ and formula $f_2 = t_1 \wedge t_2 \wedge \ldots \wedge t_k$. Now G contains a vertex cover of size k if and only if R_1 and R_2 are equivalent.

$\mathcal{S}_{prop}$**:** We show that testing for the equivalence of two $\mathcal{S}_{prop}$ relations is NP-hard via a reduction from SAT. Given a SAT formula, σ, we construct an $\mathcal{S}_{prop}$ instance, R, with a distinct tuple for every variable and with σ as the formula over the tuples. Now σ is satisfiable if and only if R is not equivalent to an $\mathcal{S}_{prop}$ relation with an empty set of u-tuples.

□

B Minimization Proofs

B.1 Tuple Versus Constraint Minimality

Proof of Theorem 4.3: Given R_1 and R_2, we give an algorithm to construct $R = (T, f)$ such that $|T| < |T_1|$ and $size(f) \leq size(f_1)$. First we construct an $\mathcal{S}_{tuple}$ relation R_{12} with every u-tuple being collection of mutually exclusive tuples. Two tuples, t_1 and t_2, of a relation are said

to be mutually exclusive if (1) no instance of the relation contains both t_1 and t_2 and (2) any instance of the relation containing t_1 can be converted to another instance of the relation by replacing t_1 with t_2 (and vice versa). The u-tuples in R_{12} are obtained by first converting the u-tuples with attribute-ors in R_1 and R_2 into u-tuples with tuple-ors. Each such u-tuple is a set of mutually exclusive tuples. We can now look at combinations of u-tuples from R_1 and R_2 to form the smallest number of u-tuples, T, each of which can be represented as attribute-ors. This number of u-tuples is less than $|T_1|$ as R_2 itself has fewer u-tuples that R_1. Let us assume for this proof we have constraints over u-tuples with tuple-ors (which are equivalent to the constraints over attribute-ors).

We will now show how to transform f_1 to constraints over T. The u-tuples in T can be obtained from those in T_1 by a series of *splits* and *combines*: sets of tuples in the u-tuples of R_1 are either split into two sets, or two sets of tuples are combined into one.

All necessary splits of u-tuples in R_1 are first performed, and then the u-tuples are combined to obtain T. Since R_{12} has fewer u-tuples than T_1, the number of combines is more than the number of splits. Further, every split of a u-tuple necessarily gives rise to a combine (otherwise the split would not have been performed itself). Each time either a split or combine is performed, we modify the formula so that the new relation with the modified u-tuples and formula remains equivalent to R_1. We now describe how the formula is modified during splits and combines, and it can be seen that the number clauses added during splits are reduced back during combines. Hence we ultimately get a formula with size at most that of f_1.

Splits: Consider relations $R = (T, f)$ and $R' = (T', f')$ where $T' = (T - t) \cup t_1 \cup t_2$ and t_1 and t_2 are obtained by splitting set $t \in T$. We would like to modify f to obtain f' such that $R' \equiv R$. We start off by adding the constraint $\neg t_1 \vee \neg t_2$ to f. Then every occurrence of t in f is replaced by $t_1 \vee t_2$. And every clause of the form $\neg t \vee C$ is replaced by two clauses, $\neg t_1 \vee C$ and $\neg t_2 \vee C$. This transformation preserves the equivalence of R and R' while adding $k + 1$ clauses, where k is the number of clauses containing $\neg t$.

Combinations: Consider relations $R = (T, f)$ and $R' = (T', f')$ where $T' = (T - t_1 - t_2) \cup t$ and t is obtained by combining t_1 and t_2. We modify f to obtain f' such that $R' \equiv R$. Without loss of generality, assume that the number of clauses containing $\neg t_1$ is at least as large as the number of clauses containing $\neg t_2$. Then delete all clauses containing $\neg t_1$ and replace every occurrence of t_2 and $\neg t_2$ with t and $\neg t$ respectively. Also replace every clause of the form $t_1 \vee C$ with C. This transformation preserves the equivalence of R and R' while reducing the number of clauses in the formula by l where l is the number of clauses containing $\neg t_1$. □

B.2 Incremental Minimality of Operations

Proof of Theorem 4.6:

We show here that the algorithms presented in [20] maintain incremental under union and cross product. We just need to show that no u-tuples in the result of union or cross product can be merged. That is, if we start with two $\mathcal{S}_{prop}$ relations R_1 and R_2 with the minimum number of u-tuples, the algorithms in [20] give $S = R_1 \cup R_2$ and $T = R_1 \times R_2$, where S and T have minimal number of u-tuples.

Union: The result $S = R_1 \cup R_2$ has the union of the u-tuples in R_1 and R_2. Now note that no instances of u-tuples from R_1 can be merged from those of R_2. This follows from the fact that we use multiset semantics, and merging tuples, say t_1 and t_2 would eliminate the possible instance in S which contained both t_1 and t_2; there has to be such an instance containing both t_1 and t_2 as R_1 and R_2 being minimal, every u-tuple in them appears in some instance. Finally, if we can merge u-tuples in R_1 (or R_2 resp.) itself, then this violates minimality of R_1 (R_2 resp.).

Cross Product: Now consider $S = R_1 \times R_2$. The result S contains a u-tuple for every combination of u-tuples from R_1 and R_2. Here again, merging any two instances of u-tuples from R_1 and R_2 would change the possible instances of S: Suppose we merged two tuples t_1 and t_2, both these were in distinct u-tuples in a least one of R_1 and R_2, and there must have been an instance containing both t_1 and t_2, which is ruled out after merging. □

Failure of Selection, Projection and Natural Join:

We show that the most natural algorithms of performing these operations fail to maintain incremental minimality. In other words, we show that after performing these operations, it may become necessary to merge two u-tuples in the result. Consider an $\mathcal{S}_{prop}$ relation R with one attribute X having two possible instances:

```
P1: (1), (3)
P2: (2)
```

It can be seen that any $\mathcal{S}_{prop}$ relation that represents exactly the possible instances above would need to have three u-tuples $\{t_1$:(1),t_2:(2),t_3:(3)$\}$. Now performing the selection $R' = \sigma_{X>1}(R)$ we get the possible instances:

```
P1': (3)
P2': (2)
```

As can be seen, now the tuples (2) and (3) can be merged in the result to give the representation: $\{(2), (3)\}$, and just retaining t'_2:(2) and t'_3:(3) as u-tuples does not give the minimal result.

We now similarly show a case where u-tuples can be merged after applying projection. Consider the relation $S(X, Y)$ with two attributes and just one possible world:

```
P1:(p 1), (q 2)
```

The minimal $\mathcal{S}_{prop}$ relation for S would need two u-tuples $\{t_1$:(p 1) and t_3:(q 2). However, on performing $S' = \Pi_X(S)$, the result is a single possible world with two tuples (p) and (q). In this result, t'_1 and t'_2 can be merged to give a single u-tuple $\{p, q\}$.

For natural join, we use an idea similar to selection: replace (1), (2) and (3) in R with (a, 1), (b, 2) and (b, 3), and then perform the join of R with T having a simple possible instance (b). The result of $R \bowtie T$ is:

```
P1': (b 3)
P2': (b 2)
```

Here again the tuples (b 2) and (b 3) can be merged in the result to form one u-tuple {(b 2), (b 3)}. Therefore, for these operations if we want to maintain incremental minimality, we need to detect possible merges of the resulting u-tuples, and just the natural algorithm without merging does not work.

Finally, we show that in general, maintaining minimal number of u-tuples while performing operations is a hard problem. This can be seen by a direct reduction from the NP-complete bi-clique cover problem to the minimality of number of u-tuples for two attributes. An instance of the bi-clique cover problem is reduced to a set of tuples over two attributes with (a, b) added if and only if there is an edge between a and b. And any arbitrary set of tuples can be obtained from a minimal $\mathcal{S}_{prop}$ relation with exactly one u-tuple containing the cross product of all possible vertex combinations: We perform a selection only to retain the edges in the input of the bi-clique cover problem.

C Approximate Representation Proofs

Proof of Theorem 5.2: First note that Algorithm $\mathcal{S}_{tuple}$-REP runs in polynomial time in P: Steps 1-6 require at most a pass of the instances in P. In the final step we check whether the constructed R represents exactly P. Even a brute force implementation of this would take at most quadratic time in P; we can enumerate the instances of R and check if they appear in P, each with a scan of P.

We now show that the algorithm correctly returns R whenever an $\mathcal{S}_{tuple}$ representation of P exists. Clearly if it returns R, it is a representation of P (Step 7). Finally we show if no R is returned, there does not exist any $\mathcal{S}_{tuple}$ representation of P; in other words, we show that every step in the algorithm adds tuples to u-tuples of the partialy generated R in the only way possible. When we look for the addition of t_i in step 5, we look at its co-occurences with other added tuples. We then find a representation for the restriction of P to tuples $t_1, \ldots, t_i$, and there is a unique $\mathcal{S}_{tuple}$ representation for this (if any). So if we can add t_i, this is the unique possible way of adding t_i, and if there is no representation for the restriction of P to $t_1, \ldots, t_i$, there is no $\mathcal{S}_{tuple}$ representation of P. □

Proof of Lemma 5.4: We show that the set of n possible instances from Example 5.5 has no constant factor approximation under the Jaccard measure of similarity between possible instances. Consider some approximation R which agrees with $P = \{P_1, \ldots, P_n\}$ on exactly k instances, $0 \leq k \leq n$. Let the number of possible instances of R be m; the approximation is then given by $|I(R) \cap P|/|I(R) \cup P| = k/(n + m - k)$. Note that for R to have a constant factor approximation, $k = \Omega(n)$.

Now since R has k possible instances from P, the tuples of at least k of the possible instances must be in the $\mathcal{S}_{tuple}$ representation of R. Further, whenever tuples for a particular instances, say P_j is chosen, the rest of the tuples are not present in the instance. So either the u-tuples for all $(k, 1)$ and $(k, 2)$, $j \neq k$ have '?', or are in the same u-tuples as $(j, 1)$ and $(j, 2)$ respectively. It can be seen that under these conditions, for m to be smallest we have just two u-tuples in R, the first one being `(` $(i_1$`, 1)` || $(i_2$`, 1)` || ... || $(i_k$`, 1)` `)` and the second being `(` $(i_1$`, 2)` || $(i_2$`, 2)` || ... || $(i_k$`, 2)` `)`. Even in this case we have k^2 possible instances, and so we do not get a constant factor approximation. □

Proof of Theorem 5.7: We first show the NP-hardness of finding the best approximation by a reduction from the NP-complete minimum graph coloring problem. Consider an input graph $G(V, E)$ to the minimum graph coloring problem. We construct a set P of $|E|$ possible instances over a schema with one attribute. For all $v_i, v_j \in V$, the instance containing the two tuples (v_i) and (v_j) is in P if and only if $(v_i, v_j) \in E$. Intuitively, a possible instance being covered by an approximation R imposes the condition that the endpoints of that edge are colored differently in P. Any approximation R under the conditions mentioned in the paper is a coloring of G with each u-tuple being a different color. Firstly, since the maximum multiplicity of any tuple in a possible instance is 1, each tuple appears in at most one u-tuple in R. Therefore, each coloring gives an approximation with the number of u-tuples being the number of colors, and vice versa. The best approximation gives the minimum number of u-tuples required and hence the minimum number of distinct colors required to color G.

We now show an inverse mapping: reducing an instance of the best approximation problem to that of minimum graph coloring, which is known to admit a 5/7-differential approximation [42]. Let us say we are given a set P of possible instances. As a first step, we make the maximum multiplicity of any tuple to be 1. For example, let us say there is a tuple t that appears a maximum of 2 times in instances of P; each first instance is replaced by t_1, and second instance by t_2. We now construct a graph $G(V, E)$ with each distinct tuple t_k in any instance of P being a vertex, and adding an edge (t_i, t_j) if and only if they appear together in some instance of P. It can again be seen that every coloring of G gives an approximation of P, and every approximation of P gives a coloring of G. Thus,

the best approximation of P is obtained by assigning every tuple corresponding to a vertex with the same color to a u-tuple.

□

D Closure versus Completeness Proof

Proof of Theorem 6.1:

We show the result for the set $\{\Pi, \bowtie\}$, and this implies that the result holds for $\{\Pi, \times, \sigma\}$, because a join can be performed as a cross product, followed by selection and projection. We show how any arbitrary set of possible instances $P = \{I_1, I_2, \ldots, I_n\}$ can be represented in $\mathcal{S}$. Start with an uncertain relation R with two possible instances, P_1:(1) and P_2:(2) that is representable in $\mathcal{S}$. Perform $R' = R \bowtie_\theta R$, where θ is the empty condition. Since we look at uncertain relations, the two uncertain relations R in the expression for R' are independent of each other and the result of the join gives four possible instances: P_1:(11), P_2:(12), P_3:(21), and P_4:(22). We keep performing such joins till we get n possible instance. (If n is not a power of 2, we get the next power of 2 possible instances and then join with a regular relation with n tuples to get exactly n possible instances.) Let us call this uncertain relation T_1, that is representable in $\mathcal{S}$.

We now construct a regular relation T_2 as follows. The schema of T_2 has all attributes in the schema of the instances of P, and also all attributes in T_1. The tuples of T_2 are obtained by taking all tuples of I_j, padding them with the tuples in P_j of T_1 and including them in T_2. Since T_2 is a regular relation, it can be represented in $\mathcal{S}$.

Finally, since T_1 and T_2 are representable in $\mathcal{S}$, so is $\Pi_X(T_1 \bowtie T_2)$ where X is the set of attributes in P. And notice that $\Pi_X(T_1 \bowtie T_2)$ has exactly the possible worlds of P, and so P is representable in $\mathcal{S}$. □