Noname manuscript No.
(will be inserted by the editor)

Representing Uncertain Data: Models, Properties, and Algorithms

Anish Das Sarma[1], Omar Benjelloun[2], Alon Halevy[2], Shubha Nabar[3], Jennifer Widom[1]

Abstract In general terms, an uncertain relation encodes a set of possible certain relations. There are many ways to represent uncertainty, ranging from alternative values for attributes to rich constraint languages. Among the possible models for uncertain data, there is a tension between simple and intuitive models, which tend to be *incomplete*, and complete models, which tend to be nonintuitive and more complex than necessary for many applications. We present a space of models for representing uncertain data based on a variety of uncertainty constructs and tuple-existence constraints. We explore a number of properties and results for these models.

We study completeness of the models, as well as closure under relational operations, and we give results relating closure and completeness. We then examine whether different models guarantee unique representations of uncertain data, and for those models that do not, we provide complexity results and algorithms for testing equivalence of representations. The next problem we consider is that of minimizing the size of representation of models, showing that minimizing the number of tuples also minimizes the size of constraints. We show that minimization is intractable in general and study the more restricted problem of maintaining minimality incrementally when performing operations. Finally, we present several results on the problem of approximating uncertain data in an insufficiently expressive model.

1 Introduction

The field of *uncertain databases* has attracted considerable attention over the last few decades (e.g., [2,3,31,38,43]), and is experiencing revived interest [6,13,16,19,37,40,56, 59] due to the increasing popularity of applications such as data cleaning and integration, information extraction, scientific and sensor databases, and others. We observe that data models for uncertainty either tend to have limited expressive power, or gain expressiveness at the cost of being nonintuitive. This paper presents a space of data models for uncertainty. A wide spectrum of expressiveness is covered by varying the allowed tuple-level constructs and existence constraints across tuples. We explore this space through several important properties of uncertain data models.

A relation in an uncertain database represents a set of *possible instances* (sometimes called *possible worlds*) for the relation. A variety of constructs can be used to represent possible instances [2]: A set of alternative values can be used to indicate uncertainty about the value of a particular attribute in a tuple [10,25,27,30,31,39,45] or the entire tuple can be comprised of several alternative tuples [12, 25,37,53]. Annotations can be used to indicate uncertainty about whether or not a tuple is present, and constraints on variables or tuple identifiers can be used to correlate uncertainties such as tuple presence or alternative values [3,21, 23,34,35,38,43].

In this paper, we consider combinations of tuple-level constructs as well as constraints across tuples, giving us a *space of models* for uncertain data comprised of a finite set of possible instances. We identify an inherent tension: Intuitive models that appear to capture the most common types of uncertainty in data typically are not *complete*—they cannot represent every finite set of possible instances. Furthermore, often such models are not even *closed* under some standard relational operators—the result of performing operations on the uncertain data may not be representable in the models. Complete models, on the other hand, can be complex and their representations can be difficult to understand and reason about. After enumerating the space of models to

This work was supported by the National Science Foundation under grants IIS-0324431 and IIS-0414762, by grants from the Boeing and Hewlett-Packard Corporations, by a Microsoft Graduate Fellowship, and by a Stanford Graduate Fellowship from Sequoia Capital.

[1]Stanford University, [2]Google Inc., [3]Microsoft Corp.

be considered, we identify and study several important properties of the models:

Closure and Completeness

We first analyze the closure and completeness properties of the models, as defined in the previous paragraph. (Intuitively, closure studies whether a model can represent the result of an operation, and completeness studies whether a model can represent any set of possible instances.) For models that are not closed under some operations, we determine which other models can represent results of performing these operations. We also show a new result connecting closure and completeness: Any model that can represent a certain minimal form of uncertainty, and that is closed under a small subset of relational operations, is also complete.

Expressiveness

We study the expressive power of each model. A model $\mathcal{M}$ is strictly more expressive than a model $\mathcal{M}'$ if $\mathcal{M}$ can represent all uncertain relations that can be represented in $\mathcal{M}'$, and at least one uncertain relation that cannot be represented in $\mathcal{M}'$. By identifying fundamental properties of uncertain data that the various models do or do not satisfy, we are able to establish a hierarchy of expressive power among the models we study.

Membership Problems

In uncertain databases it is natural to consider *membership tests*: (1) Is a given tuple t in some instance of an uncertain relation R? (2) Is a given tuple t in *every* instance of an uncertain relation R? (3) Is a given certain relation I an instance of an uncertain relation R? (4) Is a given certain relation I *the only* instance of an uncertain relation R? A considerable amount of past work has studied these problems, e.g., [2,3,34,35,38]. We give complexity results for these problems with respect to the new models we introduce, showing that some of the simpler models permit more efficient membership testing.

Uniqueness and Equivalence

A model $\mathcal{M}$ is said to be *unique* if every representable set of relation instances has a unique representation in $\mathcal{M}$; otherwise $\mathcal{M}$ is *non-unique*. We analyze uniqueness properties of the different models we study. Previous models for uncertain databases, e.g., [3,31,34,38,43], are generally very expressive and easily seen to be non-unique. The problem of uniqueness becomes significantly more interesting when we consider simpler models. For the non-unique models, we address the problem of testing whether two uncertain relations represent the same set of instances. We analyze the complexity of equivalence testing and give algorithms for the polynomial cases.

Minimization

Since non-unique models may have many equivalent representations, we are interested in defining and identifying *minimal representations* for sets of possible instances. The non-unique models that we consider are comprised of tuples and constraints. An important result we show is that minimizing the number of tuples in a representation also minimizes the size of constraints required to represent the uncertain relation. This result simplifies the minimization problem, but minimizing arbitrary uncertain relations is still NP-hard. We therefore study the problem of preserving minimality incrementally when performing operations on minimal representations.

Approximation

The last problem we address is that of approximating an uncertain relation when we wish to use a simple model 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 in some of the simpler models we consider. If not, then approximation is required (i.e., representing the set of possible instances "as well as possible"). We show that there are "bad cases" of sets of possible instances for which there is no constant-factor approximation in the simpler models we consider. We also define a "best" approximation, show that it is NP-hard to find, and give a polynomial-time algorithm to find an approximate best approximation.

1.1 Outline of Paper

The paper proceeds as follows. In Section 2 we motivate our constructs for uncertainty, then in Section 3 we enumerate the models we consider. The properties and problems highlighted above are studied in Sections 4–9. Related work is discussed in Section 10, and we conclude with future work in Section 11.

2 Constructs for Uncertainty

We introduce a sequence of examples on a running sample application to motivate the space of models we consider (Sections 2.1 and 2.2). We then review and define the necessary fundamentals for the remainder of the paper (Section 2.3).

2.1 A Running Example

As a running example for the paper, we consider data management for the *Christmas Bird Count* (CBC) [1,60]. Each year, volunteers and professionals worldwide observe birds for a fixed period of time, recording their observations. The data from year to year is used to understand trends in bird populations, and to correlate bird life with short-term and long-term environmental conditions. Individual bird sightings may not always be precise in terms of species, location, or time, and the observations of professionals may provide more reliable data than those of amateurs. Thus, there is inherent uncertainty in the data.

In this paper we considerably simplify and hypothesize some of the CBC data and functionality to keep the examples relevant and simple. We use the following schema; for the actual CBC schema see [1].

```
BirdInfo(birdname, color, size)
Sightings(observer, when, where, birdname)
```

2.2 Motivation for Uncertainty Constructs

Let us begin with an observer, Amy, who definitely saw a jay, and may or may not have seen another bird that was either a crow or a raven. These observations can be represented in the `Sightings` relation as follows:

Observer	When	Where	Birdname
Amy	12/23/06	Stanford	jay
Amy	12/23/06	Stanford	{crow, raven}

?

"{`crow, raven`}" is an *attribute-or*, indicating uncertainty between the two values, and "?" denotes a *maybe-tuple*, i.e., uncertainty whether the tuple is in the relation. (Previous work has used similar constructs to represent uncertainty in the value of an attribute [10,25,27,30,31,39,45] and uncertainty in the presence of a tuple [3,21,23,34,35,38,43].) Intuitively, this uncertain relation represents the following set of three possible relation instances, the first containing only a single tuple, and the other two containing two tuples and differing on `birdname`:

Observer	When	Where	Birdname
Amy	12/23/06	Stanford	jay

Observer	When	Where	Birdname
Amy	12/23/06	Stanford	jay
Amy	12/23/06	Stanford	crow

Observer	When	Where	Birdname
Amy	12/23/06	Stanford	jay
Amy	12/23/06	Stanford	raven

2.2.1 Completeness and Closure

A first natural question to ask is whether every possible set of relation instances can be captured by a model $\mathcal{M}$ that allows only attribute-ors and maybe-tuples. The answer is no, meaning $\mathcal{M}$ is *incomplete*.

Example 1 Consider for example the following three instances, in which Amy saw either a crow, a raven, or both.

Observer	When	Where	Birdname
Amy	12/23/06	Stanford	crow

Observer	When	Where	Birdname
Amy	12/23/06	Stanford	raven

Observer	When	Where	Birdname
Amy	12/23/06	Stanford	crow
Amy	12/23/06	Stanford	raven

To see why this set of instances cannot be represented, note that we would need two separate tuples for `crow` and `raven`, otherwise we would not get the third instance, which has two tuples. Both of these tuples would have to be marked "?" (to get the two instances with one tuple each). However, then the empty relation (no birds sighted) would also be a possible instance, which we did not intend to include. □

Now let us explore what happens when we perform operations on the data. Informally, the operation is performed on all possible input instances to generate its result (formally defined in Section 2.3).

Example 2 Suppose we have the following sighting of either a dove or a sparrow:

Observer	When	Where	Birdname
Bill	12/27/06	Palo Alto	{dove,sparrow}

and the following relevant tuples in the `BirdInfo` relation:

Birdname	Color	Size
dove	gray	medium
sparrow	brown	small

If we perform a natural join of these two relations, there are two possible instances in the result:

Observer	When	Where	Birdname	Color	Size
Bill	12/27/06	Palo Alto	dove	gray	medium

Observer	When	Where	Birdname	Color	Size
Bill	12/27/06	Palo Alto	sparrow	brown	small

Using the types of uncertainty we have looked at so far—attribute-ors and maybe-tuples—there is no way to represent that exactly one of these two tuples exists. □

Example 3 Consider the same sighting tuple from the previous example, but now as a contrived example for illustrative purposes, suppose the `BirdInfo` relation contains:

Birdname	Color	Size
dove	gray	medium
dove	white	small

Now the natural join produces the following two instances (the empty instance is shown by displaying just the schema and no data):

Observer	When	Where	Birdname	Color	Size

Observer	When	Where	Birdname	Color	Size
Bill	12/27/06	Palo Alto	dove	gray	medium
Bill	12/27/06	Palo Alto	dove	white	small

Again, using the types of uncertainty we have looked at so far, there is no way to represent that either both of the tuples exist or neither do. □

2.2.2 Adding Constraints to the Model

Examples 2 and 3 show that a model with only attribute-ors and maybe-tuples is not closed under the natural join operation. Specifically, Example 2 shows that for closure we need some form of *mutual exclusion* over tuples (exclusive-or, denoted $\oplus$), while Example 3 shows we need *mutual inclusion* (iff, denoted $\equiv$).

These examples suggest adding *constraints* over the existence of tuples, and in fact later we will see that by allowing arbitrary existence constraints, we obtain a complete (and therefore closed) model. The next example shows that constraints involving only $\oplus$ and $\equiv$ are not sufficient for completeness.

Example 4 Consider the following set of instances representing zero, one, or two sightings, but the later sighting cannot be recorded without the earlier one:

Observer	When	Where	Birdname

Observer	When	Where	Birdname
Carol	12/25/06	Los Altos	bluebird

Observer	When	Where	Birdname
Carol	12/25/06	Los Altos	bluebird
Carol	12/26/06	Los Altos	bluebird

The reader may verify that this set of instances also cannot be represented using attribute-ors and maybe-tuples. Intuitively, this example requires an *implication constraint* between two tuples. □

For complexity reasons it is natural to consider constraints that are restricted to binary clauses: 2-satisfiability is polynomially solvable, whereas 3-satisfiability is NP-hard [33]. It turns out 2-clauses are not sufficient for completeness either, as seen in the next example, which also explores the effect of selection predicates on uncertain attribute values.

Example 5 Suppose we have the following sighting of either a crow, sparrow, or dove:

Observer	When	Where	Birdname
Dave	12/25/06	Menlo Park	{crow,sparrow,dove}

and the following tuples in the `BirdInfo` relation:

Birdname	Color	Size
crow	black	medium
sparrow	brown	small
dove	gray	medium

If we perform a natural semijoin of `BirdInfo` with `Sightings`, the result has the following set of three possible instances:

Birdname	Color	Size
crow	black	medium

Birdname	Color	Size
sparrow	brown	small

Birdname	Color	Size
dove	gray	medium

Representing this set of instances requires an exclusive-or among three tuples, so it cannot be modeled with only 2-clauses. □

2.2.3 Tuple-Ors

Let us go back to Example 2: The join result has two possible instances with one tuple each and is not representable using only attribute-ors and maybe-tuples. The possible instances in this result can, however, be represented using *tuple-ors*, a set of distinct alternative (mutually exclusive) values for tuples, as follows:

(Observer, When, Where, Birdname, Color, Size)
(Bill, 12/27/06, Palo Alto, dove, gray, medium) ||
(Bill, 12/27/06, Palo Alto, sparrow, brown, small)

Note that tuple-ors are strictly more expressive than attribute-ors: a tuple with multiple attribute-ors represents all combinations of possible attribute values, whereas a tuple-or can specify all combinations or only specific ones. With tuple-ors and "?"s we can also represent the three-way mutual-exclusion in Example 5 above, but we still do not achieve closure: the mutual inclusion in Example 3 is still not representable. Previous work [12,25,37,53] has used similar constructs to represent possible values for a tuple.

2.2.4 C-Tables

Most early work in uncertain databases has been devoted to defining and analyzing data models, and in particular, complete models. The first and foremost complete model is *c-tables*, introduced originally in [38]. A c-table is comprised of tuples, possibly containing some variables in place of values. Each tuple has an associated *condition*, specified by a conjunction of (in)equality constraints. A c-table may also contain a global condition (introduced in [34]). Each assignment of values to variables that satisfies the global condition represents one possible relation instance: the relation

containing all tuples whose conditions are satisfied with the given assignment.

The presence of free variables in c-tables allows them to represent an infinite set of possible instances, while in this paper we are considering only models that represent finite sets of possible instances. Further, the models we consider in this paper represent sets of possible instances for each uncertain relation in isolation; i.e., correlations across relations aren't captured. Under this setting, we shall see that like c-tables, $\mathcal{M}^A_{prop}$, the most expressive model in our space, is *complete*. Therefore, in our setting $\mathcal{M}^A_{prop}$ and c-tables are equally expressive.

Example 6 Recall the original example uncertain relation representing a sighting of a jay and a possible sighting of either a crow or a raven. This example is represented in a c-table as:

Observer	When	Where	Birdname	
Amy	12/23/06	Stanford	jay	
Amy	12/23/06	Stanford	crow	$(x = 0) \wedge (y = 0)$
Amy	12/23/06	Stanford	raven	$(x \neq 0) \wedge (y = 0)$

The conditions involving x and y together impose the constraint that at most one of the tuples `(Amy,12/23/06,Stanford,crow)` and `(Amy,12/23/06,Stanford,raven)` is present. If $(y \neq 0)$ none of the two tuples is present and if $(y = 0)$ exactly one of the two tuples is present depending on the value of x: the former tuple is present if $(x = 0)$ and the latter tuple is present if $(x \neq 0)$. □

While c-tables are a very elegant formalism, the above example illustrates one of its disadvantages: even fairly simple cases of uncertainty can be hard for users to read and reason with intuitively. Some studies [51,55] suggest that the use of free variables is what makes it nonintuitive. The uncertainty models we consider in this paper—both the incomplete and complete models—do not have free variables.

Several restrictions of c-tables (such as v-tables, Codd tables, etc.) were also studied around the same time as c-tables and subsequently [3]. These models put together can also be thought of as a *space* of models. In this paper, we focus on data models containing uncertainty constructs and various forms of tuple-existence constraints mentioned earlier.

2.2.5 Summary

Before proceeding we make a few observations:

1. Simple, intuitive models for uncertainty may be sufficient for the purposes of some applications. If the initial uncertainty in the data is representable in a simple model $\mathcal{M}$, and the application requires only a restricted set of operations under which $\mathcal{M}$ is closed, then $\mathcal{M}$ is sufficient for the application. Thus, it is worthwhile to study closure and expressiveness properties of various models.
2. When relational operations are performed on simple models, more complex forms of uncertainty may be generated, as seen in examples above. We are therefore interested in analyzing the progression in complexity of uncertainty as different operations are performed.
3. Complex types of uncertainty may take many forms, but in general uncertainty can be captured by adding existence constraints among tuples, describing what combinations of tuples may together in a possible instance. Different forms of constraints capture different types of uncertainty.

We will soon (Section 3) introduce a specific space of models motivated by these observations. Then in Sections 4–9 we will study several properties of the models as motivated in Section 1. However, we first need some preliminary definitions and fundamentals.

2.3 Fundamentals of Uncertain Databases

We review and define a number of basic formal concepts needed in the remainder of the paper. Note that varied terminology for many of these concepts has been used in previous related work.

A *relation schema* is defined as a set of attributes $\{A_1, A_2, \ldots, A_n\}$. A *tuple* is an assignment of one value to each of the attributes in the schema. A *relation instance* is a multiset of tuples. Set semantics are obtained through explicit duplicate-elimination operations. We use the term *ordinary relation* to refer to conventional relations with no uncertainty.

Definition 1 (Uncertain Relation) An *uncertain relation* R defines a set of *possible instances* (or *instances* for short), $I(R) = \{R_1, R_2, \ldots, R_n\}$, where each R_i is an ordinary relation. □

A *data model* (or simply *model*) defines a method for representing uncertain relations R, i.e., a way of representing sets of instances $I(R)$. Section 2.2 motivated some possible data models for uncertain relations by introducing various extensions to the basic relational model: sets of possible attribute values instead of single values, sets of alternative tuples instead of regular tuples, maybe-tuples, and existence constraints among tuples.

Definition 2 (Completeness) A data model $\mathcal{M}$ is *complete* if any finite set of relation instances corresponding to a given schema can be modeled by an uncertain relation represented in $\mathcal{M}$. □

In Section 2.2 several examples illustrated incompleteness by showing a model $\mathcal{M}$ and a set of instances S such that no R in $\mathcal{M}$ could represent S, i.e., there was no R expressible using $\mathcal{M}$ such that $I(R) = S$.

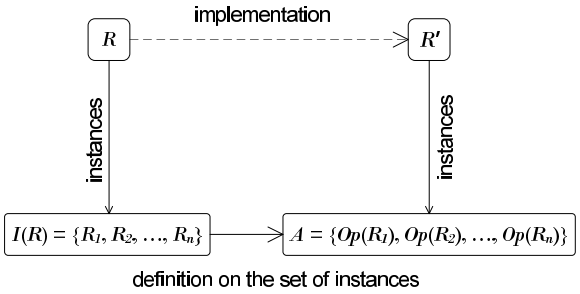


Fig. 1 Example illustrating Definition 3 for unary operation *Op* on uncertain relation R.

Next we formalize operations on uncertain relations and the closure property.

Definition 3 (Operations on Uncertain Relations) 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 $I(R) = \{I \mid I = Op\,(I_1, I_2, \ldots, I_n),\ I_1 \in I(R_1),\ I_2 \in I(R_2), \ldots, I_n \in I(R_n)\}$. □

Note this definition can be applied to any data model for uncertain relations according to Definition 1. Figure 1 illustrates the definition for a unary operator *Op*. The definition considers the set of instances $I(R)$ of an uncertain relation R (the downward arrow on the left), and applying *Op* on each instance of $I(R)$ to obtain the set of resulting instances A (the lower arrow). The first question raised by this definition is whether an uncertain relation R' exists in the model such that $I(R')$ is the set of instances in A (the downward arrow on the right). *Closure* is the condition that formalizes the existence of R' in a model $\mathcal{M}$:

Definition 4 (Closure) A model $\mathcal{M}$ is said to be *closed* under an operation *Op* if performing *Op* on any set of uncertain relations in $\mathcal{M}$ results in an uncertain relation that can be represented in $\mathcal{M}$. □

Naturally, when $\mathcal{M}$ is closed under *Op*, a reasonable implementation would compute *Op* directly on R and not through the set of possible instances, as depicted by the upper dashed arrow in Figure 1.

3 Space of Models

In this section we describe formally a space of models and their interrelationships, capturing and combining the types of uncertainty we saw in Section 2.2. As illustrated in Section 2.2, we consider two fundamental types of uncertainty:

1. uncertainty within a tuple, about the value of the tuple itself
2. uncertainty across tuples, where the existence of a tuple may be uncertain, or may depend on the existence of other tuples

We begin by defining $\mathcal{M}^A_{prop}$, a complete model that captures intuitively both types of uncertainty. We will define the other models in terms of different *restrictions* on $\mathcal{M}^A_{prop}$. In our notation, the superscript on $\mathcal{M}$ specifies the first type of uncertainty, and we use A to indicate that attribute-ors are permitted. The subscript on $\mathcal{M}$ specifies the second type of uncertainty, and we use $prop$ to indicate that full propositional logic constraints across tuples are permitted. We now formalize A-tuples, the $\mathcal{M}^A_{prop}$ model, and then the other models.

Definition 5 (A-tuple) An A-tuple is a tuple consisting of either a single value or an attribute-or for each of its attributes. An *instance* of an A-tuple is a regular tuple in which we choose a single value for each attribute-or. □

Definition 6 ($\mathcal{M}^A_{prop}$) An $\mathcal{M}^A_{prop}$ relation $R = (T, f)$ consists of:

1. a multiset of A-tuples whose identifiers are $T = t_1, \ldots, t_n$, and
2. a boolean formula $f(T)$. □

We assume unique tuple identifiers, i.e., $t_i \neq t_j$ for $i \neq j$, but the values of the A-tuples referred to by t_i and t_j could be the same. In $f(T)$ the tuple identifiers are used as propositional variables with the meaning that t_i is `True` if and only if t_i is in the relation instance. A *satisfying assignment* for $f(T)$ is an assignment of either `True` or `False` to each $t_i \in T$ such that $f(T)$ evaluates to `True`. Note that identifiers that do not appear in $f(T)$ may be set to either `True` or `False` in such an assignment. The following defines the instances of an $\mathcal{M}^A_{prop}$ relation.

Definition 7 (Instances of $\mathcal{M}^A_{prop}$) The set of instances $I(R)$ of an $\mathcal{M}^A_{prop}$ relation $R = (T, f)$ is the set of all possible ordinary relations obtained as follows:

1. Pick a satisfying assignment σ of $f(T)$. Let T' be the set of A-tuples in T whose identifiers are set to `True` in σ.
2. Pick one instance for every A-tuple in T' to obtain an instance $I(R)$ of R. □

Example 7 The set of instances from Example 4 can be represented as an $\mathcal{M}^A_{prop}$ relation as follows:

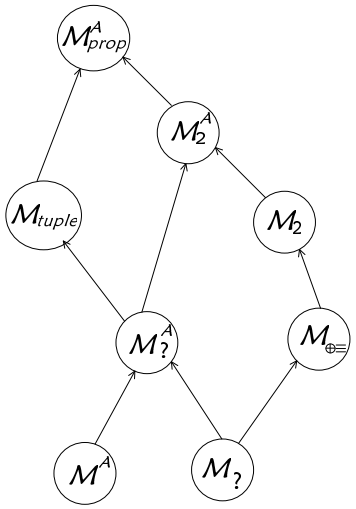
	Observer	When	Where	Birdname
t_1	Carol	12/25/06	Los Altos	bluebird
t_2	Carol	12/26/06	Los Altos	bluebird

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

The formula $f(T)$ ensures that t_2 is present only if t_1 is present as well. □

Model	Data	Constraints
$\mathcal{M}^A$	A-tuples	none
$\mathcal{M}_?$	tuples	?
$\mathcal{M}^A_?$	A-tuples	?
$\mathcal{M}_{\oplus\equiv}$	tuples	binary $\oplus$, $\equiv$
$\mathcal{M}_2$	tuples	2-clause
$\mathcal{M}^A_2$	A-tuples	2-clause
$\mathcal{M}_{tuple}$	tuples	n-way mutual exclusion, ?
$\mathcal{M}^A_{prop}$	A-tuples	propositional

(a) Models



(b) Expressiveness Hierarchy

Fig. 2 Space of Models

We now define a space of models as restrictions on $\mathcal{M}^A_{prop}$. Restrictions are obtained by limiting the kind of constraints specified in formula f, denoted in the subscript of the model name, and/or limiting the model to use ordinary tuples and not A-tuples, denoted in the superscript of the name. Specifically, we use the notation shown in Table 2(a). The superscript A on $\mathcal{R}$ means the model allows A-tuples, otherwise only ordinary tuples. The subscript specifies the type of constraints allowed: An empty subscript means that each A-tuple must be present in the relation. A subscript "?" means that maybe-tuples are allowed. A subscript "2" means that f is a conjunction of clauses with at most two literals. The subscript "$\oplus\ \equiv$" means that f is a conjunction of formulas each of the form $(t_k \oplus t_l)$ or $(t_i \equiv t_j)$. $\mathcal{M}_{tuple}$ consists of tuple-ors (Section 2.2.3) and "?"s. We shall see shortly that $\mathcal{M}_{tuple}$ can be represented as a restriction of $\mathcal{M}^A_{prop}$ using regular tuples and constraints encoding n-way mutual exclusion.

Figure 2(b) gives the interrelationship of our models in terms of their expressive power: An edge from model $\mathcal{M}_1$ up to $\mathcal{M}_2$ means $\mathcal{M}_2$ is strictly more expressive than $\mathcal{M}_1$. We will study and prove the relative expressive power of these models in detail in Section 5.

Example 8 In Example 3 we joined an A-tuple with two ordinary tuples, but the result was not expressible with A-tuples. The result of the join can be expressed with an $\mathcal{M}_{\oplus\equiv}$ relation:

	Observer	When	Where	Birdname	Color	Size
t_1	Bill	12/27/06	Palo Alto	dove	gray	medium
t_2	Bill	12/27/06	Palo Alto	dove	white	small

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

Example 9 The result of the semijoin in Example 5 was not representable with A-tuples. We can represent it in $\mathcal{M}_{tuple}$ with a single tuple-or:

(Birdname, Color, Size)
(crow, black, medium) || (sparrow, brown, small) || (dove, gray, medium)

This tuple-or is equivalent to encoding the constraint

$$f = (\neg t_1 \vee \neg t_2) \wedge (\neg t_2 \vee \neg t_3) \wedge (\neg t_1 \vee \neg t_3) \wedge (t_1 \vee t_2 \vee t_3)$$

over tuples t_1, t_2, t_3 given by:

	Birdname	Color	Size
t_1	crow	black	medium
t_2	sparrow	brown	small
t_3	dove	gray	medium

□

In general, any $\mathcal{M}_{tuple}$ relation can be represented in $\mathcal{M}^A_{prop}$ by constraints $f(t_1, t_2, \ldots, t_n)$ that take the following special form. For an $\mathcal{M}_{tuple}$ relation with m tuple-ors, we partition T into m parts $r_1, \ldots, r_m$ corresponding to the tuple-ors. For each r_j containing say $\{t_{j_1}, t_{j_2}, \ldots, t_{j_n}\}$, f contains the constraint $\neg t_{j_l} \vee \neg t_{j_m}$ for every pair $1 \leq l < m \leq n$. If the tuple-or r_j does not contain a "?", then we also add the constraint $t_{j_1} \vee t_{j_2} \vee \ldots \vee t_{j_n}$.

Example 10 The $\mathcal{M}_{tuple}$ relation from Example 9 can be converted to a $\mathcal{M}^A_{prop}$ relation with three tuples t_1, t_2, t_3 corresponding to the three choices for the tuple-or. The constraints are $((\neg t_1 \vee \neg t_2) \wedge (\neg t_2 \vee \neg t_3) \wedge (\neg t_1 \vee \neg t_3) \wedge (t_1 \vee t_2 \vee t_3))$. □

Other models could also have been picked in our space. We picked this set of models because, as will be seen, they are distinct from each other in many respects, and together they capture a variety of interesting properties.

4 Closure and Completeness

Now that we have defined the space of models, we study their closure (Section 4.1) and completeness (Section 4.2) properties. We then present an interesting result connecting closure and completeness (Section 4.3).

4.1 Closure

In Section 2.2 we saw several examples of models and operations under which they were not closed: we were not able to "complete the square" in Figure 1 by finding an R' within the model that represented the resulting set-of-instances A. In this section we study the closure properties of the models we introduced in Section 3.

We consider the multiset relational operators *Union*, *Intersection*, *Cross-Product*, *Join*, *Difference*, and *Projection*, in addition to *Duplicate-Elimination* and *Aggregation*. As it turns out, closure under *Selection* subtly depends on the exact form of selection. We analyze closure for three different selection predicate cases, based on whether or not attribute-ors are involved:

1. In *Select(ee)* both operands have exact values.
2. In *Select(ea)* one operand has an exact value and the other is an attribute-or.
3. In *Select(aa)* both operands are attribute-or.

As an example of the subtlety, $\mathcal{M}^A$ is closed only under *Select(ee)*, while $\mathcal{M}^A_?$ is closed only under *Select(ee)* and *Select(ea)*.

Theorem 1 The closure properties of our models are specified in Table 1. If "Y" appears in the row for operation Op and column for model $\mathcal{M}$, then $\mathcal{M}$ is closed under the operation Op, otherwise it is not. □

An exhaustive proof of this theorem appears in the appendix. Let us look at an example showing the difference between *Select(ee)* and *Select(ea)*. $\mathcal{M}^A$ is closed under *Select(ee)*. Consider a relation with one A-tuple `[{crow,raven}]`, and predicate `bird='crow'` which falls under *Select(ea)* since `'crow'` is exact and `bird` is an attribute-or. The result is the maybe-tuple `[crow]?`, which is not representable in $\mathcal{M}^A$. Thus $\mathcal{M}^A$ is not closed under *Select(ea)*.

It is interesting to note that while $\mathcal{M}_{\oplus\equiv}$, $\mathcal{M}_{tuple}$, and $\mathcal{M}^A_2$ all have differing expressive powers, they have exactly the same closure properties. Note that $\mathcal{M}^A_{prop}$ is closed under all relational operations. We shall see shortly that $\mathcal{M}^A_{prop}$ is in fact a complete model.

Model Closure	$\mathcal{M}^A$	$\mathcal{M}_?$	$\mathcal{M}^A_?$	$\mathcal{M}_2$	$\mathcal{M}_{\oplus\equiv}$,$\mathcal{M}^A_2$, $\mathcal{M}_{tuple}$	$\mathcal{M}^A_{prop}$
Union	Y	Y	Y	Y	Y	Y
Select(ee)	Y	Y	Y	Y	Y	Y
Select(ea)	N	Y	Y	Y	Y	Y
Select(aa)	N	Y	N	Y	Y	Y
Intersection	N	Y	N	N	N	Y
Cross-Product	Y	N	N	N	N	Y
Join	N	N	N	N	N	Y
Difference	N	Y	N	N	N	Y
Projection	Y	Y	Y	Y	Y	Y
Duplicate Elimination	N	Y	N	N	N	Y
Aggregation	N	N	N	N	N	Y

Table 1 Closure of the Space of Models

4.2 Completeness

Every complete model is closed under all relational operations, since every operation generates a finite set of instances, and any set of instances is representable by a complete model. The converse is not true however: models may be closed under many operations even though they are not complete. An extreme example is a model that permits only ordinary relations. This model is closed under all relational operations but certainly is not complete for uncertain data. In our space of models, obviously only $\mathcal{M}^A_{prop}$ could be complete, since all the other models are not closed under some operation. The following theorem shows that $\mathcal{M}^A_{prop}$ is indeed complete.

Theorem 2 $\mathcal{M}^A_{prop}$ is a complete model. □

Proof Given a set of instances $\{I_1, I_2, \ldots, I_n\}$, we show how to construct an $\mathcal{M}^A_{prop}$ relation $R = (T, f)$ such that $I(R) = \{I_1, I_2, \ldots, I_n\}$. We associate with each tuple appearing in any I_i a tuple identifier $t \in T$. For each distinct tuple value in any I_i, the number of tuples in T having that value is equal to the maximum number of times it appears in any instance. For any i let $T_i \subseteq T$ be the identifiers of tuples appearing in I_i; $T_i = \{ti_1, ti_2, \ldots, ti_{n_i}\}$. Let the identifiers in $T - T_i$ be $\{si_1, si_2, \ldots, si_{m_i}\}$. We then set f to be:

$$\bigvee_{i=1}^{n}(ti_1 \wedge ti_2 \wedge \ldots \wedge ti_{n_i} \wedge \neg si_1 \wedge \neg si_2 \wedge \ldots \wedge \neg si_{m_i})$$

Each satisfying assignment of f now corresponds to one instance: For f to be satisfied, one of the disjunctive clauses above needs to be true. If the ith disjunct is true, the instance I_i is obtained. □

Now we show that no restriction of $\mathcal{M}^A_{prop}$ bounding the size of the clauses in the propositional formula can be complete.

Theorem 3 Let model $\mathcal{M}$ be the restriction of $\mathcal{M}^A_{prop}$ where the size of each clause in the CNF representation of *prop* is less than k, a constant or function of the size of the schema (but independent of the data). $\mathcal{M}$ is an incomplete model. □

Proof We show a set of instances not representable in $\mathcal{M}$. Consider an uncertain relation $R(A, B)$ and k tuples $t_1, t_2, \ldots, t_k$ where the value of tuple t_i is (i, i). Let R have all possible instances except the empty instance, i.e., each instance has at least one t_j. R cannot be represented in $\mathcal{M}$: It would need the k-clause $(t_1 \vee t_2 \vee \ldots \vee t_k)$. Any other propositional formula involving clauses with fewer than k literals, or a different k-clause, rules out some possible instance of R: Specifically, it rules out possible instances corresponding to assignments that set the literals in the clause to *false*. (Note that the encoding of an arbitrary CNF formula into 3CNF needs to add new variables and hence is not applicable above.) □

4.3 Closure versus Completeness

In our space of models, many of them are closed under many operations (Table 1), but only one of them is complete. We show an interesting result that any model $\mathcal{M}$ that is expressive enough to represent a certain basic form of uncertainty, and that is closed under a certain small set of operations, can represent all finite sets of possible instances of an uncertain relation, i.e., $\mathcal{M}$ is complete. Note that a more general similar result was subsequently obtained in [37].

Theorem 4 (Closure ⇒ Completeness)
Consider a model $\mathcal{M}$ with the following properties:

- **Basic Uncertainty:** $\mathcal{M}$ can represent all ordinary (certain) relations as well as independent copies of a unary uncertain relation R with two possible instances $\{\texttt{[0]}\}$ and $\{\texttt{[1]}\}$.
- **Minimal Closure:** $\mathcal{M}$ is closed under one of the following sets of operations: (a) $\{\Pi, \bowtie\}$, or (b) $\{\sigma, \Pi, \times\}$.[1]

Then $\mathcal{M}$ is a complete model. □

Proof We show how any arbitrary set of possible instances $P = \{I_1, I_2, \ldots, I_n\}$ can be represented in $\mathcal{M}$. We shall first construct an uncertain relation T_1 with n possible instances: $\{\texttt{[1]}\}$, $\{\texttt{[2]}\}$, $\cdots$, $\{\texttt{[n]}\}$. We will then construct an ordinary relation T_2 such that joining T_1 and T_2 yields the set of possible instances in P.

Although T_1 is conceptually simple, creating it from our building blocks is a bit complex. Start with l uncertain relations R_1, R_2, …, R_l, each with two possible instances, $\{\texttt{[0]}\}$ and $\{\texttt{[1]}\}$, where l is the smallest integer such that $2^l \geq n$. Successively perform $(l-1)$ joins with empty join conditions to obtain S_1, i.e., $S_1 = (\cdots((R_1 \times R_2) \times R_3) \times \cdots \times R_l)$. If n is not a power of 2 we get $m = 2^l$, the next power of 2, possible instances in S_1: Each instance has one l-length tuple with one of the possible combinations of 0's and 1's, i.e., binary representations of numbers from 0 to $2^l - 1$. Now consider an ordinary relation S_2 that contains all attributes in S_1 and an additional attribute *PW*. S_2 has m tuples corresponding to the tuples in the m possible instances in S_1. The *PW* attribute in S_2 is assigned such that all values from 1 to n appear in some tuple, and 1 to n are the only values that appear. Define T_1 to be the natural join of S_1 and S_2 followed by a projection onto *PW*. T_1 has n possible instances, the ith instance having a single tuple with value i.

We now construct T_2 as follows. Let X be the set of attributes in P. The schema of T_2 is the composition $X \circ PW$. The tuples of T_2 are obtained by taking, for each j, $1 \leq j \leq n$, all tuples of I_j concatenated with the tuple in the instance P_j of T_1. Since T_2 is an ordinary relation, it can be represented in $\mathcal{M}$.

Finally, since T_1 and T_2 are representable in $\mathcal{M}$, so is $\Pi_X(T_1 \bowtie T_2)$. $\Pi_X(T_1 \bowtie T_2)$ has exactly the possible instances of P, so P is representable in $\mathcal{M}$.

Our result also holds for the minimal closure set $\{\Pi, \times, \sigma\}$ since a join can be performed as a cross-product followed by selection and projection. □

Note that the same result holds when the uncertain relation can only represent two possible instances $\{\}$ and $\{\texttt{[1]}\}$, i.e., presence or absence of one tuple.

5 Expressiveness and Transition Diagram

Next, we present and analyze several distinguishing properties in our space of models, and we use the properties to establish the strict hierarchy of expressive power we saw in Figure 2(b). We then consider a generalized form of closure: we present a transition diagram showing which of our models can represent results of performing operations on other models.

The following definition formalizes relative expressive power of models.

Definition 8 (Relative Expressive Power) A model $\mathcal{M}_2$ is *as expressive* as a model $\mathcal{M}_1$ if every set of instances that can be represented by a relation in $\mathcal{M}_1$ can also be represented by a relation in $\mathcal{M}_2$. $\mathcal{M}_2$ is *more expressive* than $\mathcal{M}_1$ if $\mathcal{M}_2$ is as expressive as $\mathcal{M}_1$ and there is some set of instances representable in $\mathcal{M}_2$ but not $\mathcal{M}_1$. □

As a simple example, $\mathcal{M}^A_?$ is more expressive than $\mathcal{M}^A$: $\mathcal{M}^A_?$ can represent the set of two possible instances $\{\texttt{[a]}\}$ and $\{\}$, while $\mathcal{M}^A$ cannot represent these instances. The following theorem establishes the main result of this section.

Theorem 5 (Expressiveness Hierarchy) Recall the expressiveness hierarchy depicted in Figure 2(b). If there is an arrow from model $\mathcal{M}_1$ to model $\mathcal{M}_2$, then $\mathcal{M}_2$ is

[1] σ, Π, $\bowtie$, and $\times$ denote arbitrary selections, projections, joins, and cross-product respectively.

more expressive than $\mathcal{M}_1$. If $\mathcal{M}_1$ and $\mathcal{M}_2$ are not ancestor/descendant pairs, then neither $\mathcal{M}_1$ nor $\mathcal{M}_2$ is as expressive as the other. □

The proof of Theorem 5 will be based on a set of properties that distinguish the expressive power of the different models. (These properties also seem interesting in their own right.) We will determine for which models each property holds for all relations in that model. The properties are essentially restrictions on the possible instances for uncertain relations. Thus, if a model $\mathcal{M}_1$ satisfies a property that $\mathcal{M}_2$ does not, then there is some set of possible instances representable in $\mathcal{M}_2$ but not $\mathcal{M}_1$. The properties we use are:

- **Constant Cardinality:** All instances of R have the same number of tuples.
- **Path Connectedness:** Define a *path* between two instances I_1 and I_2 of R as a sequence of ordinary relations beginning with I_1 and ending with I_2 where each relation adds, deletes, or replaces a single tuple in the previous relation in the sequence. We say that a model $\mathcal{M}$ is *path connected* for R if for any pair of instances I_1 and I_2 of R, there is a path between I_1 and I_2 such that every relation I along the path is also an instance of R.
- **Unique Minimum:** R has a unique instance with a minimum number of tuples.
- **Complement:** Every instance I of R has a complement $I^c \in R$ such that: (1) $I \cup I^c$ contains the entire set of tuples that appear in *any* instance of R, and (2) $I \cap I^c$ contains the set of tuples that appear in every instance of R.
- **3-Tuple Exclusion:** A model $\mathcal{M}$ satisfies the *3-tuple exclusion* property if there is no relation R expressible in $\mathcal{M}$ such that there exist three ordinary tuples t_1, t_2 and t_3 satisfying the following property: every instance $I \in R$ contains exactly one t_i, and $(I - t_i + t_j) \in R$ for $1 \leq i \neq j \leq 3$. That is, a model satisfies the property if it cannot express 3-way mutual exclusion between ordinary tuples.

The following theorem specifies exactly which of our models satisfy which of these properties. The proof of the Expressiveness Hierarchy theorem (5) is based on this theorem.[2]

Theorem 6 (Properties of Models) Table 2 establishes the properties of our models. If "Y" appears in the row for property P and column for model $\mathcal{M}$, then $\mathcal{M}$ satisfies the property P, and otherwise it does not. □

Proof

[2] Note that two pairs of columns in Table 2 are identical. In most cases the properties delineate the models, but in this case a more subtle delineation is needed and is seen in the proof.

- **Constant Cardinality:** The constant cardinality property is satisfied by $\mathcal{M}^A$ since every A-tuple contributes exactly one tuple to any instance. For every other model, one can construct instances either using maybe-tuples or constraints so that there are instances with unequal cardinalities: A "?" denotes possible absence of a tuple and hence can give instances with different cardinalities. In a model with constraints, absence of any constraint on a tuple implies it may or may not be present, and hence again gives different cardinalities.
- **Path Connectedness:** Path connectedness of an $\mathcal{M}^A$ relation R between two of its instances I_1 and I_2 follows from a series of swaps of the distinct tuple instances of the A-tuples in I_1 and I_2: Since picking any instance of each A-tuple results in an instance of R, each relation in the path is also an instance of R. Path connectedness of $\mathcal{M}_?$ follows from the fact that I_1 and I_2 may only differ on maybe-tuples. These differences can be eliminated by a series of insertions/deletions. Path connectedness of $\mathcal{M}_?^A$ can be similarly observed by first looking at the tuples generated from maybe-tuples in I_1 and I_2 and eliminating/adding these one by one like $\mathcal{M}_?$. The difference between I_1 and I_2 now only originates from the other A-tuples, which can be connected just like for $\mathcal{M}^A$. Note that each of the atomic changes above keeps us within the set of instances. The path connectedness property of $\mathcal{M}_{tuple}$ follows the same idea as that of $\mathcal{M}_?^A$. Finally, the set of instances in Example 8 can be represented in all of $\mathcal{M}_{\oplus\equiv}$, $\mathcal{M}_2$, and $\mathcal{M}_2^A$, but this set of instances does not satisfy the path connectedness property as the only two instances are not neighbors.
- **Unique Minimum:** $\mathcal{M}_?$ satisfies this property since the unique minimum instance is obtained by not choosing any of the maybe-tuples. In the models allowing attribute-ors or tuple-ors, we can create a relation with at least two minimum cardinality instances by introducing an attribute-or (respectively tuple-or) in any tuple present in the minimum instance. Finally, $\mathcal{M}_{\oplus\equiv}$ and $\mathcal{M}_2$ do not satisfy the unique minimum property since both of them can represent the set of instances in Example 2 containing two minimums.
- **Complement:** The complement of an $\mathcal{M}_?$ instance is obtained by inverting the selections of each of the maybe-tuples. The $\mathcal{M}_{\oplus\equiv}$ relation only allows constraints of the form $t_i \oplus t_j$ and $t_i \equiv t_j$. For these constraints, any satisfying assignment gives another satisfying assignment by flipping the value of every variable. Hence every instance of an $\mathcal{M}_{\oplus\equiv}$ relation has a complement (the certain tuples correspond to pairs of variables whose tuple values are the same but assignments are flipped in the satisfying assignments). Note that such a complement satisfying assignment does not necessarily exist for $\mathcal{M}_2$ or $\mathcal{M}_2^A$ with arbitrary 2-clauses in f. The

Property-Model	$\mathcal{M}^A$	$\mathcal{M}_?$	$\mathcal{M}_?^A$	$\mathcal{M}_{\oplus\equiv}$	$\mathcal{M}_2$	$\mathcal{M}_2^A$	$\mathcal{M}_{tuple}$	$\mathcal{M}_{prop}^A$
Constant Cardinality	Y	N	N	N	N	N	N	N
Path Connectedness	Y	Y	Y	N	N	N	Y	N
Unique Minimum	N	Y	N	N	N	N	N	N
Complement	N	Y	N	Y	N	N	N	N
3-Tuple Exclusion	N	Y	N	Y	Y	N	N	N

Table 2 Properties Satisfied by Models

property also is not satisfied by either of $\mathcal{M}^A$or $\mathcal{M}_?^A$ because one could construct a relation with an A-tuple having, say, three instances, and at most one of them appears in any instance of the relation. Similarly, an $\mathcal{M}_{tuple}$ relation having a tuple-or with three tuples does not satisfy the property.

– **3-Tuple Exclusion:** This property is not as natural and intuitive as the other properties. However, it brings out the distinction between $\mathcal{M}_2$ and $\mathcal{M}_2^A$. In the general case, for modeling a 3-way exclusive-or constraint, a 3-clause is necessary. However, an A-tuple, for example {`finch`, `nightingale`, `dove`}, can have an implicit 3-way exclusive-or constraint across its instances. Therefore, models that have A-tuples—$\mathcal{M}^A$, $\mathcal{M}_?^A$, $\mathcal{M}_2^A$, and $\mathcal{M}_{prop}^A$—do not satisfy this property. Similarly, $\mathcal{M}_{tuple}$ also does not satisfy this property. $\mathcal{M}_?$ satisfies this property since all tuples in any $\mathcal{M}_?$ relation are independent of each other. $\mathcal{M}_{\oplus\equiv}$ can express only binary exclusive-or, and since 2-clauses cannot be used to express 3-way exclusion, $\mathcal{M}_{\oplus\equiv}$ and $\mathcal{M}_2$ also satisfy the property.

Finally, $\mathcal{M}_{prop}^A$ does not satisfy any of the properties as it is a complete model: Since it can represent any finite set of instances, it can represent the ones that violate the properties. □

We are now in a position to prove Theorem 5.

Proof of Theorem 5: There are two parts in the proof. (1) Subsumption: If there is an edge from $\mathcal{M}_2$ to $\mathcal{M}_1$, then $\mathcal{M}_1$ is more expressive than $\mathcal{M}_2$. (2) Incomparability: If $\mathcal{M}_1$ and $\mathcal{M}_2$ are not ancestor/descendant pairs, then neither $\mathcal{M}_1$ nor $\mathcal{M}_2$ is as expressive as the other.

– **Subsumption:** We must show for every $\mathcal{M}_2 \rightarrow \mathcal{M}_1$ that every uncertain relation in $\mathcal{M}_2$ can be converted to an equivalent relation in $\mathcal{M}_1$, and that there exists an uncertain relation in $\mathcal{M}_1$ not representable in $\mathcal{M}_2$.
The first part—showing $\mathcal{M}_2$ can be converted to an equivalent representation in $\mathcal{M}_1$—is easier than showing strict subsumption, i.e., $\mathcal{M}_1$ is more expressive than $\mathcal{M}_2$. Clearly every $\mathcal{M}^A$ and $\mathcal{M}_?$ relation is also an $\mathcal{M}_?^A$ relation and every $\mathcal{M}_{\oplus\equiv}$ relation is also an $\mathcal{M}_2$ relation. Also, every $\mathcal{M}_?$ relation can be converted to an equivalent $\mathcal{M}_{\oplus\equiv}$ relation by imposing no constraints on the maybe-tuples. For each other tuple (with no "?"), two identifiers with the same value are involved in an xor constraint in $\mathcal{M}_2$. Every $\mathcal{M}_?^A$ relation can be converted to an $\mathcal{M}_2^A$ relation with absence/presence of positive 1-clause constraints: For every tuple t in $\mathcal{M}_?^A$ with no "?" the constraint t is added in $\mathcal{M}_2^A$, and no constraint is added involving the other tuples. Since every tuple is also an A-tuple, every $\mathcal{M}_2$ relation is also a $\mathcal{M}_2^A$ relation. Every $\mathcal{M}_?^A$ relation can be converted to an equivalent $\mathcal{M}_{tuple}$ relation by replacing every A-tuple with a tuple-or containing all the instances of the A-tuple and retaining the "?" labels.
Proving that $\mathcal{M}_1$ is more expressive than $\mathcal{M}_2$ involves giving explicit examples of sets of possible instances representable in $\mathcal{M}_1$ and not $\mathcal{M}_2$. We rely on the properties introduced earlier in this section. Consider the $\mathcal{M}_?^A$ relation:

Observer	When	Where	Birdname
Amy	12/23/06	Stanford	jay
Amy	12/23/06	Stanford	{crow, raven}

?

The set of instances represented by the above relation neither satisfies the constant cardinality property (satisfied by $\mathcal{M}^A$) nor the complement property (satisfied by $\mathcal{M}_?$), and hence $\mathcal{M}_?^A$ is more expressive than both $\mathcal{M}^A$ and $\mathcal{M}_?$. The above relation can also be represented in $\mathcal{M}_2$, and since $\mathcal{M}_{\oplus\equiv}$ also satisfies the complement property, $\mathcal{M}_2$ is more expressive than $\mathcal{M}_{\oplus\equiv}$. Consider the set of instances in Example 2 representable in $\mathcal{M}_{\oplus\equiv}$: Since this set of instances does not satisfy the unique minimum property (satisfied by $\mathcal{M}_?$), $\mathcal{M}_{\oplus\equiv}$ is more expressive than $\mathcal{M}_?$. To see that $\mathcal{M}_2^A$ is more expressive than $\mathcal{M}_?^A$, recall the following set of two instances from Example 3:

Observer	When	Where	Birdname	Color	Size

Observer	When	Where	Birdname	Color	Size
Bill	12/27/06	Palo Alto	dove	gray	medium
Bill	12/27/06	Palo Alto	dove	white	small

Since $\mathcal{M}_?^A$ satisfies the path connectedness property, violated by the instances above, no $\mathcal{M}_?^A$ relation can represent it. An $\mathcal{M}_2^A$ relation can however represent it by explicitly encoding the mutual inclusion constraint. Since $\mathcal{M}_2$ satisfies 3-tuple exclusion, and an attribute-or in $\mathcal{M}_2^A$ can express an instance not satisfying the property, $\mathcal{M}_2^A$ is more expressive than $\mathcal{M}_2$. The fact that $\mathcal{M}_{tuple}$

is more expressive than $\mathcal{M}_?^A$ follows from Example 9 showing an $\mathcal{M}_{tuple}$ relation not representable in $\mathcal{M}_?^A$. Finally $\mathcal{M}_{prop}^A$ is clearly more expressive than all other models since no other model is complete.

– **Incomparability:** We must show that for every $\mathcal{M}_1$ and $\mathcal{M}_2$ that are not ancestor/descendant pairs, there exist sets of possible instances representable by $\mathcal{M}_1$ but not $\mathcal{M}_2$ and vice-versa.
We need to show incomparability for each of the following pairs of models: ($\mathcal{M}^A$,$\mathcal{M}_?$), ($\mathcal{M}^A$,$\mathcal{M}_{\oplus\equiv}$), ($\mathcal{M}^A$,$\mathcal{M}_2$), ($\mathcal{M}_{\oplus\equiv}$,$\mathcal{M}_?^A$), ($\mathcal{M}_{\oplus\equiv}$,$\mathcal{M}_{tuple}$), ($\mathcal{M}_?^A$,$\mathcal{M}_2$), ($\mathcal{M}_2$,$\mathcal{M}_{tuple}$), and ($\mathcal{M}_{tuple}$,$\mathcal{M}_2^A$). To establish incomparability of a pair, it is sufficient to show a property satisfied by the first model and not the second, and a property satisfied by the second model and not the first. It can be seen from Table 2 that for each of the pairs listed above except ($\mathcal{M}_{tuple}$,$\mathcal{M}_2^A$), in the corresponding two columns there exists a row with a "Y" entry in the first column and "N" in the second, and a row with a "N" entry in the first column and "Y" in the second, thus proving the incomparability of each of these pairs. For the pair ($\mathcal{M}_{tuple}$,$\mathcal{M}_2^A$), path connectedness is satisfied by $\mathcal{M}_{tuple}$ and not by $\mathcal{M}_2^A$. Therefore it remains to show that there exists a set of instances representable by $\mathcal{M}_{tuple}$ and not by $\mathcal{M}_2^A$. A simple example is an $\mathcal{M}_{tuple}$ relation containing just one tuple-or with three tuples not representable by attribute-ors. Representing this instance with A-tuples would require a 3-clause and so cannot be done by $\mathcal{M}_2^A$. □

Transition Diagram

Now that we have analyzed relative expressiveness in our space of models, to complete the picture on closure and expressiveness we consider the problem of determining which other (more expressive) models can represent results of performing operations on relations in a model, for operations under which the model is not closed. The following theorem, proved in the appendix, shows how we transition among models upon performing these operations.

Theorem 7 Consider Figure 3. In this figure, an arrow from model $\mathcal{M}_1$ to model $\mathcal{M}_2$ with label Op means that the result of performing Op on relations in $\mathcal{M}_1$ is expressible in $\mathcal{M}_2$. □

Roughly, the arrows in Figure 3 are obtained by combining the expressiveness hierarchy in Figure 2 and the closure properties from Table 1. *Intersection*, *Cross-Product*, *Join*, and *Difference* are denoted by standard symbols in Figure 3. $Select(ea)$ and $Select(aa)$ are denoted by σ_1 and σ_2 respectively, and α stands for *Aggregation*. All the "Y" entries in Table 1 could have been shown as self-arcs in Figure 3

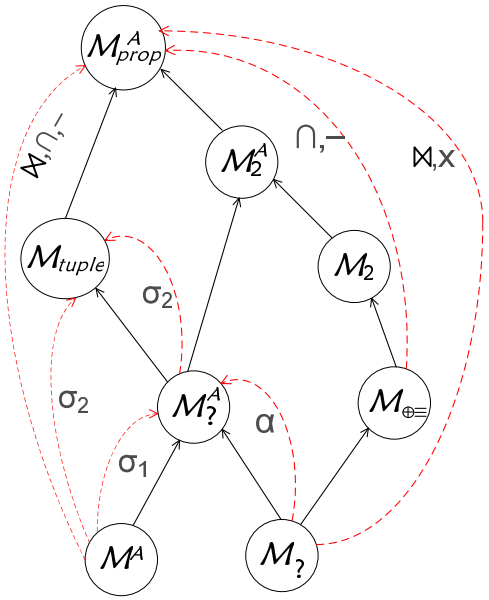


Fig. 3 Closure Transition Diagram

but are omitted for readability. In addition, when there is a transition from a model $\mathcal{M}$ to the complete model $\mathcal{M}_{prop}^A$ on a particular operation, then the same transition holds for all models that are more expressive than $\mathcal{M}$; these arcs are also omitted for readability. Finally, for all models except $\mathcal{M}_?$, aggregation and duplicate elimination transition to $\mathcal{M}_{prop}^A$, and these arcs are not shown either.

In summary, we have shown the relative expressive power of our space of models by identifying a set of interesting properties that delineate the various models in the space. We then saw how we transition from a model $\mathcal{M}$ to other (more expressive) models upon performing operations under which $\mathcal{M}$ is not closed.

6 Membership Problems

In uncertain databases it is natural to consider *membership problems* [2,3,25]. We analyze the complexity of these problems in our models, presenting algorithms and hardness results. Consider an uncertain relation R whose possible instances are $I(R)$. The problems we are interested in are:

– **Instance Membership:** Given a relation instance I, determine whether I is an instance of R, i.e., whether $I \in I(R)$.
– **Instance Certainty:** Given a relation instance I, determine whether I is the only instance of R, i.e., whether $I(R) = \{I\}$.
– **Tuple Membership:** Given a tuple identifier t, determine if there exists some instance $I \in I(R)$ such that $t \in I$.

– **Tuple Certainty:** Given a tuple identifier t, determine if t appears in every instance of $I(R)$, i.e., $\forall I \in I(R)$, $t \in I$.

Unlike most previous work, we are more interested in identifier-based tuple membership and certainty, rather than value-based membership and certainty. That is, we want to know whether the tuple identified by t appears in any/all instance(s), and not whether any tuple with the same value appears in any/all instance(s). Identifier-based membership is motivated by systems that show the representation of uncertain data to the user. Once an uncertain relation is shown to the user (in any of the models), the user may "click" on a tuple and ask for its membership or certainty. For the same reason, we are more interested in membership of materialized relations, rather than membership of query answers.

Next we consider each of the four membership problems for each of the eight models in our space. The following theorem shows which of the simpler models permit more efficient membership testing than more expressive ones.

Theorem 8 (Membership Problems)

– The tuple membership and certainty problems can be solved in polynomial time for all models in our space except $\mathcal{M}^A_{prop}$.
– The instance membership problem can be solved in polynomial time for $\mathcal{M}^A$, $\mathcal{M}_?$, $\mathcal{M}^A_?$, and $\mathcal{M}_{tuple}$, but is NP-complete for $\mathcal{M}_{\oplus\equiv}$, $\mathcal{M}_2$, $\mathcal{M}^A_2$, and $\mathcal{M}^A_{prop}$.
– The instance certainty problem can be solved in polynomial time for $\mathcal{M}^A$, $\mathcal{M}_?$, $\mathcal{M}^A_?$, $\mathcal{M}_{tuple}$, and $\mathcal{M}_{\oplus\equiv}$, but is Co-NP-complete for $\mathcal{M}^A_{prop}$. □

Proof The intractability results for all the four problems for the complete model from [3] can be adapted for $\mathcal{M}^A_{prop}$ as well. We prove the remaining results.

Tuple Membership and Certainty: Tuple membership and certainty for $\mathcal{M}^A$, $\mathcal{M}_?$, $\mathcal{M}^A_?$, and $\mathcal{M}_{tuple}$ can be solved simply by scanning the list of tuples (or the tuple-ors in the case of $\mathcal{M}_{tuple}$), checking if t is a possible tuple, and looking at "?" labels if any: For each of the models, the answer to membership of t is "yes" if t is in the list of tuples. The answer to tuple certainty is "yes" if and only if t appears alone, with no "?" or other tuples. For $\mathcal{M}_{\oplus\equiv}$, $\mathcal{M}_2$, and $\mathcal{M}^A_2$, we can again look for t, and then solve the 2-SAT problem by either setting the corresponding variable to 1 (for tuple membership) or 0 (for tuple certainty): Setting t to 1, if the 2-SAT formula is satisfiable then t appears in some instance, otherwise not. Setting t to 0, if the 2-SAT formula is satisfiable then t is not a certain tuple, otherwise it is.

Instance Membership: Instance membership for $\mathcal{M}^A$, $\mathcal{M}_?$, $\mathcal{M}^A_?$, and $\mathcal{M}_{tuple}$ can be reduced to polynomially-solvable maximum bipartite matching problems as follows. We construct a bipartite graph G with vertices in the first partition A corresponding to the tuples in instance I, and vertices in the second partition B corresponding to A-tuples in the $\mathcal{M}^A$, $\mathcal{M}_?$, $\mathcal{M}^A_?$, or $\mathcal{M}_{tuple}$ representation for R. There is an edge (v_1, v_2) in G, $v_1 \in A$ and $v_2 \in B$, if the tuple corresponding to v_1 is an instance of the A-tuple corresponding to v_2. Every matching now corresponds to an instance of R. I is an instance of R if and only if the maximum bipartite matching has cardinality $|I|$, i.e., every tuple of I is matched.

Next we show the instance membership hardness results. The corresponding completeness results can be seen easily. First we show that the instance membership problem is NP-hard for $\mathcal{M}_{\oplus\equiv}$ by a reduction from the subset-sum problem: given a set of positive integers $S = \{x_1, x_2, \ldots, x_n\}$ and a positive integer k, determine if the numbers in some subset of S add up to k. Given an arbitrary instance of the subset-sum problem, we construct an $\mathcal{M}_{\oplus\equiv}$ relation R with $\sum x_i$ tuple identifiers $\{t^1_1, t^2_1, \ldots, t^{x_1}_1, t^1_2, \ldots, t^{x_2}_2, \ldots, t^{x_n}_n\}$, all corresponding to the same tuple value `[1]`. We add equivalence constraints $t^j_i \equiv t^k_i$ for each $1 \leq i \leq n$, for each $1 \leq j < k \leq x_i$. Intuitively, for the first group of x_1 tuples, either all of them are present or none of them are; and similarly for each successive group of x_i tuples. Hence, the possible instances of R are relations where the tuple `[1]` appears with multiplicity equal to a subset-sum of S. Now the subset-sum problem reduces to instance membership of a relation where `[1]` appears k times.

The instance membership problem is NP-hard for $\mathcal{M}_2$ and $\mathcal{M}^A_2$. We show hardness by a reduction from the vertex-cover problem: given a graph $G = (V, E)$, determine if G has a vertex cover of size k. Given any arbitrary instance of the vertex-cover problem, we construct an instance of the instance membership problem as follows: For every $v_i \in V$, we add a tuple $t_i =$ `[x]` to the relation R. For every $e_{ij} \in E$, we add a constraint $t_i \vee t_j$. Now the size of the smallest instance in R is equal to the size of the minimum vertex cover in G, and an instance containing the tuple `[x]` k times is an instance of R if and only if G has a vertex cover of size k.

Instance Certainty: Instance certainty for $\mathcal{M}^A$, $\mathcal{M}_?$, $\mathcal{M}^A_?$, and $\mathcal{M}_{tuple}$ can be solved by checking if there is any "?" entry or any A-tuple with more than one instance—if so there is more than one possible instance. If there is no "?" entry and all A-tuples have no or-sets, then the relation has just one certain instance.

In the case of $\mathcal{M}_{\oplus\equiv}$ the instance certainty problem can be solved in polynomial time by partitioning the tuple identifiers as follows. Every tuple identifier is treated as a node in a graph. Every pair of identifiers that is related through an $\equiv$ according to the constraints is collapsed into a single node. Then for every pair of identifiers that is related

through an $\oplus$, an edge is created between the corresponding nodes. If the resulting graph contains any cycles with an odd number of nodes (or self-loops) the formula is unsatisfiable, and the relation does not have any instances at all. Otherwise the formula is satisfiable and in fact the tuple identifiers in each connected component of the graph will have exactly two satisfying assignments. Take an arbitrary node in a connected component. All the tuple identifiers appearing at nodes at an even distance from this node will be called the "even" identifiers, while all the rest will be called "odd". Then the two satisfying assignments are as follows: either all the even identifiers are set to *true* and all the odd identifiers are set to *false*, or vice-versa. If in any connected component the two assignments correspond to different sets of tuple values, then there is no unique instance. Otherwise the unique instance corresponds to any one of the satisfying assignments and can be found in polynomial time. □

Algorithms or hardness results for the instance certainty problem for $\mathcal{M}_2$ and $\mathcal{M}_2^A$ remain open problems and are a clear avenue for future work.

7 Uniqueness and Equivalence

A natural question that arises in our various models of uncertainty is whether every set of possible instances is guaranteed to have a unique representation. Furthermore, when uniqueness is not guaranteed, we would like to know whether two given uncertain relations in a model are equivalent, i.e., represent the same set of instances. We study uniqueness in Section 7.1 and equivalence in Section 7.2.

7.1 Uniqueness

We show in this section that $\mathcal{M}_{tuple}$ and $\mathcal{M}_?^A$ guarantee unique representations, which in turn implies that $\mathcal{M}^A$ and $\mathcal{M}_?$ also guarantee unique representations. The remaining models do not guarantee unique representations.

Definition 9 (Uniqueness) Let $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$ be two relations in an uncertainty model $\mathcal{M}$. $\mathcal{M}$ is said to guarantee unique representations if $I(R_1) = I(R_2)$ implies $T_1 = T_2$ and $f_1 \equiv f_2$. □

Theorem 9 Every set of possible instances representable in $\mathcal{M}_{tuple}$, $\mathcal{M}_?^A$, $\mathcal{M}^A$, or $\mathcal{M}_?$ has a unique representation. □

The full proof for this theorem is presented in the appendix. The result is first shown for $\mathcal{M}_{tuple}$. Given any two distinct relations R_1 and R_2 in $\mathcal{M}_{tuple}$ ($T_1 \neq T_2$), we explicitly construct an instance of one of the relations that is not an instance of the other. Thus any two relations in $\mathcal{M}_{tuple}$ that represent the same set of instances must be identical. We then show that distinct $\mathcal{M}_?^A$ relations map to distinct $\mathcal{M}_{tuple}$ relations, thus proving the uniqueness of $\mathcal{M}_?^A$. Since $\mathcal{M}^A$ and $\mathcal{M}_?$ are special cases of $\mathcal{M}_?^A$, then $\mathcal{M}^A$ and $\mathcal{M}_?$ guarantee unique representations as well.

Theorem 10 There exist sets of possible instances representable in $\mathcal{M}_{\oplus\equiv}$, $\mathcal{M}_2$, $\mathcal{M}_2^A$, and $\mathcal{M}_{prop}^A$ that do not have unique representations. □

Once again, the full proof is presented in the appendix. We show that for each of the models, there are representations $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$ such that $I(R_1) = I(R_2)$ and $T_1 \neq T_2$ or $f_1 \not\equiv f_2$. 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 model have different sets of A-tuples but the same set of instances?

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

We show that for each of the models, the answer to at least one of the questions is "yes". Hence none of the models guarantee unique representations.

Our uniqueness results divide our hierarchy into two pieces as shown in Figure 4: The models to the left of the divide all guarantee unique representations, while models to the right of the divide do not. In most of the remainder of the paper, instead of considering all eight of our models, we have selected two models from each side to focus on: $\mathcal{M}_?^A$ and $\mathcal{M}_{tuple}$ from the left of the divide, and $\mathcal{M}_2$ and $\mathcal{M}_{prop}^A$ from the right.

7.2 Equivalence

The next natural problem to consider is testing whether two relations in a model are equivalent.

Definition 10 (Equivalence) Two relations $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$ in an uncertainty model $\mathcal{M}$ are equivalent if they represent the same set of possible instances, i.e., $I(R_1) = I(R_2)$. □

All models to the left of the divide in Figure 4 guarantee unique representations for any set of possible instances, so we need not consider the problem of equivalence testing for them. Thus we focus on our two representative models from the right: $\mathcal{M}_2$ and $\mathcal{M}_{prop}^A$. The following three theorems show our equivalence results: the first theorem gives a PTIME result and the next two two theorems give hardness results, for which matching upper bounds can be obtained easily.

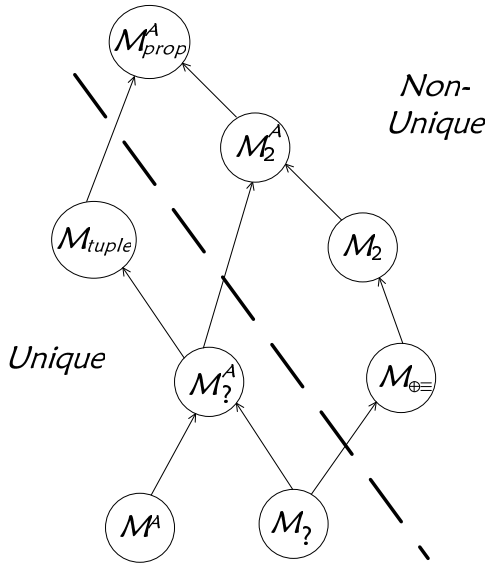


Fig. 4 Division of our Space of Models.

Theorem 11 The equivalence of two relations in $\mathcal{M}_2$ can be tested in polynomial time if duplicate tuples are not allowed in the representation. □

Theorem 12 If duplicate tuples are allowed in the representation, equivalence testing of two relations in $\mathcal{M}_2$ is NP-hard. □

Theorem 13 Equivalence testing of two relations in $\mathcal{M}^A_{prop}$ is Co-NP-hard (with or without duplicates in the representation). □

Proof of Theorem 11: We first present an equivalence testing algorithm for $\mathcal{M}_2$, then show its correctness and polynomial time complexity.

Algorithm: Given two $\mathcal{M}_2$ relations $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$, we test equivalence as follows.

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. If $T'_1 \neq T'_2$ return "not equivalent".
3. Construct f'_i from f_i by setting all variables in $T_i - T'_i$ to *false*.
4. If $f'_1 \not\equiv f'_2$ return "not equivalent".
5. Return "equivalent".

Informally, the algorithm eliminates tuples that are not present in any of the possible instances of the uncertain relations and modifies the formulas accordingly. If either the resulting tuple sets are not the same or the formulas are not equivalent at the end of this process, the relations are not equivalent. Otherwise, they are equivalent.

Correctness: For $i = 1, 2$, the algorithm first modifies T_i to retain only those tuples that are set to *true* in at least one satisfying assignment of f_i, thus obtaining minimal T'_i and the corresponding f'_i such that $I(T'_i, f'_i) = I(T_i, f_i)$. At this point, if $T'_1 \neq T'_2$, then $R'_1 \neq R'_2$: There is some $t \in T_1$, $t \notin T_2$ (or vice-versa) such that setting t to *true* keeps f_1 satisfiable, and hence there exists an instance of R_1 with t that is not in R_2. If however $T'_1 = T'_2$, the problem reduces to testing for the equivalence of f'_1 and f'_2. If the formulas are equivalent, so are R_1 and R_2. If the formulas are not equivalent, we use a result from the proof of Theorem 10: two duplicate-free $\mathcal{M}_2$ relations with the same set of tuples but non-equivalent formulas cannot represent the same set of instances.

Complexity: Each step of the algorithm can be run in polynomial time: Since testing for satisfiability of 2-CNF is polynomial [33], we can successively set each tuple in T_i to *true* and test for satisfiability to obtain T'_i (Step 1). Clearly Steps 2 and 3 of the algorithm are polynomial. Finally, since testing the equivalence of 2-CNF formulas can be done in polynomial time, Step 4 is also polynomial. □

Proof of Theorem 12: We prove that testing equivalence of two relations in $\mathcal{M}_2$ when the relations may contain duplicates is NP-hard via a reduction from the NP-hard minimum vertex cover problem [33]: Given a graph, $G = (V, E)$, determine if the size of the minimum vertex cover in G is k. We reduce the vertex-cover problem to that of testing equivalence in $\mathcal{M}_2$ as follows. Given G, we construct a relation R_1 in $\mathcal{M}_2$ 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 $(t_i \vee t_j)$, effectively encoding the condition that for every edge, at least one vertex is in the cover. Therefore, the size of the smallest instance in R_1 is equal to the size of the minimum vertex cover in G. We construct another relation R_2 in $\mathcal{M}_2$ with $|V|$ tuples $t_1 = t_2 = \ldots = t_{|V|} = [x]$ and formula $f_2 = t_1 \wedge t_2 \wedge \ldots \wedge t_k$. Therefore, G contains a vertex cover of size k if and only if R_1 and R_2 are equivalent. □

Proof of Theorem 13: For $\mathcal{M}^A_{prop}$, we show that equivalence testing is Co-NP-hard via a reduction from SAT [33]. Given a SAT formula σ, we construct an $\mathcal{M}^A_{prop}$ instance R with a distinct tuple for every variable and σ as the formula over the tuples. Now σ is satisfiable if and only if R is not equivalent to an $\mathcal{M}^A_{prop}$ relation with an empty set of A-tuples. □

8 Minimization of $\mathcal{M}^A_{prop}$

As seen in Section 7, $\mathcal{M}^A_{prop}$ and $\mathcal{M}_2$ do not guarantee unique representations for sets of possible instances. In this

section we consider the problem of finding a "minimal" representation for a set of possible instances. We present results for $\mathcal{M}^A_{prop}$ only, since all of them apply to $\mathcal{M}_2$ as well.

Two equivalent $\mathcal{M}^A_{prop}$ representations may in fact have arbitrarily different sizes. As an extreme example, consider the empty set of possible instances. This database can be represented in $\mathcal{M}^A_{prop}$ by a large unsatisfiable formula over many tuples, or by an empty set of tuples and an empty formula. Recall that an $\mathcal{M}^A_{prop}$ relation consists of A-tuples T and a boolean formula $f(T)$. We define minimality in terms of the combination of the sizes of T and f.

Example 11 Consider the possible instances:

Observer	When	Where	Birdname
Amy	12/23/06	Stanford	Crow

Observer	When	Where	Birdname
Amy	12/23/06	Stanford	Raven

One $\mathcal{M}^A_{prop}$ representation is to encode the mutual exclusion using a constraint:

	Observer	When	Where	Birdname
t_1	Amy	12/23/06	Stanford	Crow
t_2	Amy	12/23/06	Stanford	Raven

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

The constraint would require two clauses in CNF form (i.e., $f(T) = (t_1 \vee t_2) \wedge (\neg t_1 \vee \neg t_2)$). However, the minimal representation is to have one A-tuple with an attribute-or and constraint with one clause (i.e., $f = t_1$):

	Observer	When	Where	Birdname
t_1	Amy	12/23/06	Stanford	{Crow, Raven}

□

In an $\mathcal{M}^A_{prop}$ relation (T, f), let $|T|$ denote the number of A-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 11 (Minimal $\mathcal{M}^A_{prop}$) An $\mathcal{M}^A_{prop}$ relation $R = (T, f)$ is *minimal* if there does not exist another $\mathcal{M}^A_{prop}$ 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., [41,52,47]), but in the worst case minimizing arbitrary formulas (in terms of number of literals or minimal CNF form) is NP-hard [33]. 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 $\mathcal{M}^A_{prop}$ representation.

At first glance it may seem that minimizing 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 $\mathcal{M}^A_{prop}$ relation, by minimizing $|T|$ we also minimize $size(f)$. This result also holds for any of our models with a restricted set of propositional constraints, for example $\mathcal{M}_2$. We subsequently show how this result can be used to maintain minimality while performing certain operations in $\mathcal{M}^A_{prop}$.

Section 8.1 presents our main result on minimization for $\mathcal{M}^A_{prop}$ relations without duplicates and Section 8.2 briefly considers maintaining incremental minimality. Considering other definitions of minimal $\mathcal{M}^A_{prop}$ representations (such as minimizing the "number of ORs" in A-tuples, or looking at other combinations of $|T|$ and f) as well as $\mathcal{M}^A_{prop}$ relations with duplicates are interesting directions of future work.

8.1 Tuple-Minimality versus Constraint-Minimality

The following result applies to $\mathcal{M}^A_{prop}$ relations without duplicates, i.e., no two A-tuples can have the same possible tuple as an instance.

Theorem 14 (Tuple vs. Constraint Minimality) Given $\mathcal{M}^A_{prop}$ relation $R_1 = (T_1, f_1)$, if there exists $R_2 = (T_2, f_2)$, $R_2 \equiv R_1$ and $|T_2| < |T_1|$, then there exists an equivalent $\mathcal{M}^A_{prop}$ relation $R = (T, f)$ with $|T| < |T_1|$ and $size(f) \leq size(f_1)$. In other words, minimizing the number of A-tuples required to represent any set of possible instances also minimizes the size of the minimal constraint. □

Recall that $size(f)$ denotes the size in the minimal representation of f. The theorem is proved in the appendix. We show that any $\mathcal{M}^A_{prop}$ relation that uses more than the minimum required number of A-tuples can be converted to an equivalent $\mathcal{M}^A_{prop}$ relation over the minimum number of A-tuples without an increase in the size of the formula required to maintain equivalence. The translation is done through 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 the formula based on the A-tuples that are split or combined. We then use a counting argument to show that the translation cannot increase the size of the formula.

Suppose $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$ are two $\mathcal{M}^A_{prop}$ relations that represent the same set of possible instances. The theorem says that if R_1 has a minimal T then it also has minimal f_1. However, it does not imply that if $|T_1| < |T_2|$, then f_1 can be made smaller than f_2. In other words, if neither R_1 nor R_2 contains a minimal set of A-tuples, then we cannot infer the relative sizes of $size(f_1)$ and $size(f_2)$ from $|T_1|$ and $|T_2|$. The following counterexample refutes the possibility that fewer A-tuples always implies fewer constraints:

Example 12 We illustrate $\mathcal{M}^A_{prop}$ 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	

$f_1 = t_1 \wedge t_2 \wedge (\neg t_3 \vee \neg t_4) \wedge (t_3 \Rightarrow l_i) \wedge (t_4 \Rightarrow l_i)$, for all i
$f_2 = (t_1 \oplus t_2) \wedge (t_3 \oplus t_4) \wedge (t_5 \Rightarrow l_i)$, 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 A-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 result coupling minimality of A-tuples and constraints unfortunately does not hold when duplicate A-tuples occur in the representation.

Lemma 1 Theorem 14 does not hold for $\mathcal{M}^A_{prop}$ relations with duplicates in the representation. □

Proof Consider $\mathcal{M}^A_{prop}$ relation R where $T = \{t_1$:[a], t_2:[a], t_3:[a], 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 $\mathcal{M}^A_{prop}$ 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 $\mathcal{M}^A_{prop}$ representation for the same set of instances with the number of A-tuples and size of constraints being smaller than T and f respectively. □

8.2 Incremental Minimality

It can easily be shown that minimizing an arbitrary $\mathcal{M}^A_{prop}$ relation is intractable in general. (The proof reduces an instance of the SAT problem to minimization of an $\mathcal{M}^A_{prop}$ relation, with variables of the SAT instance mapped to distinct tuples. The SAT instance is unsatisfiable iff the minimal $\mathcal{M}^A_{prop}$ relation has an empty set of tuples and $f \equiv false$. Therefore, if we can minimize an arbitrary $\mathcal{M}^A_{prop}$ relation, we can test the satisfiability of an arbitrary SAT instance.) However, we can still make use of the result of Theorem 14: If we perform an operation on minimal $\mathcal{M}^A_{prop}$ relations, then minimizing the number of A-tuples in the result also minimizes the size of constraints. In general, even maintaining a minimal number of A-tuples while performing operations is a hard problem, but for some operations we have efficient algorithms. The following results are proven in the appendix.

Theorem 15 (Incremental Minimality of $\mathcal{M}^A_{prop}$) Given $\mathcal{M}^A_{prop}$ relations with minimal number of A-tuples, there exist polynomial time algorithms to compute their union or cross-product so that the resulting $\mathcal{M}^A_{prop}$ relation is minimal with respect to the number of A-tuples. □

Proposition 1 Given $\mathcal{M}^A_{prop}$ relations with minimal number of A-tuples, the most natural algorithms (operating in a "A-tupleby A-tuple fashion") to perform selection, projection, or natural join on them may fail to maintain incremental minimality. □

Theorem 16 Given an $\mathcal{M}^A_{prop}$ relation with minimal number of A-tuples, maintaining minimal number of A-tuples after performing a selection is NP-hard. □

A more detailed study of maintaining incremental minimality while performing operations is an interesting direction for future work.

9 Approximate Representations

Lastly, we study the problem of approximate representations. Approximating an uncertain relation is required when we use a model that is not expressive enough for a particular set of instances. For example, users may prefer a simpler representation than $\mathcal{M}^A_{prop}$—one that does not include full propositional formulas—and are willing to compromise with a not fully accurate representation of the possible instances. (After all, we are operating on uncertainty to begin with!)

Example 13 Consider the following three possible instances of a relation R:

Observer	When	Where	Birdname
Amy	12/23/06	Stanford	Crow

Observer	When	Where	Birdname
Amy	12/23/06	Stanford	Raven

Observer	When	Where	Birdname
Amy	12/23/06	Stanford	Crow
Amy	12/23/06	Stanford	Raven

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

	(Observer, When, Where, Birdname)	
t_1	(Amy, 12/23/06, Stanford, Crow)	?
t_2	(Amy, 12/23/06, Stanford, Raven)	?

□

In this section we consider three classes of problems:

1. Determining whether a set of possible instances P can be represented exactly in a model $\mathcal{M}$ (Section 9.1).
2. Approximating a set of possible instances P in an insufficiently expressive model $\mathcal{M}$ (Section 9.2).
3. Approximating an uncertain relation represented in model $\mathcal{M}_1$ in a less expressive model $\mathcal{M}_2$ (Section 9.3).

Note that although Sections 9.1 and 9.2 consider explicit representation of possible instances, we do not advocate using sets of possible instances as a model for representing uncertain data. The goal of this section is to explore whether a model in our space can represent a set of possible instances exactly/approximately. Thereafter, all computation (query evaluation and other studied properties) are performed over the model and not the original set of possible instances.

9.1 Does a Model $\mathcal{M}$ Suffice?

We seek to answer the following question: Given a set P of possible instances for a relation, can P be represented in a model $\mathcal{M}$? To study this problem we consider models $\mathcal{M}_{tuple}$ and $\mathcal{M}_?^A$. We first answer the question for $\mathcal{M}_{tuple}$, and then extend our result to $\mathcal{M}_?^A$.

Theorem 17 Let P be a set of possible instances. Algorithm 1, $\mathcal{M}_{tuple}$-REP, outlines a polynomial (in the size of P) algorithm that returns an exact representation of P in $\mathcal{M}_{tuple}$ whenever one exists. □

More details of the algorithm, and the proof of the theorem, appear in the appendix. Intuitively, the algorithm looks at co-occurrences of tuples in the possible instances and from them attempts to construct an $\mathcal{M}_{tuple}$ relation. If at any point a tuple cannot be added to the $\mathcal{M}_{tuple}$ relation being constructed, then no $\mathcal{M}_{tuple}$ representation is possible and the algorithm exits. If all tuples are added successfully, then the result is an $\mathcal{M}_{tuple}$ representation of P if it correctly represents the possible instances. If not, there is no representation.

Theorem 18 Algorithm $\mathcal{M}_{tuple}$-REP for finding exact $\mathcal{M}_{tuple}$ representations can be extended to return $\mathcal{M}_?^A$ representations whenever they exist. □

1: **Input:** Set of Possible Instances P
Output: $\mathcal{M}_{tuple}$ representation R for P, if one exists.
2: Pad every instance in P with zero or more special tuples "*" so the number of tuples in each instance is equal to the maximum cardinality of any instance in P.
3: 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.
4: Consider t_1. Add m_1 tuple-ors to R, each containing t_1.
5: 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 $\mathcal{M}_{tuple}$ representation, then there are $m_1 - n$ tuple-ors to which t_2 can be added. Additionally, create $m_2 - m_1 + n$ new tuple-ors to accommodate the remaining t_2's.
6: 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 t_i cannot be successfully added in this manner, then exit returning "no $\mathcal{M}_{tuple}$ representation."
7: Remove "*" from all tuple-ors in R containing it, and annotate these tuple-ors with '?'.
8: Check to see if the possible instances of R exactly match the input P. If yes, return R, otherwise return "no $\mathcal{M}_{tuple}$ representation".

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

Proof If there is no $\mathcal{M}_{tuple}$ representation, clearly there is no $\mathcal{M}_?^A$ representation either. If there is an $\mathcal{M}_{tuple}$ representation R, we check if each tuple-or in R can be represented using attribute-ors. If we can convert all tuple-ors to attribute-ors, we have our $\mathcal{M}_?^A$ representation. If there are tuple-ors in R that cannot be converted to attribute-ors, there exists no $\mathcal{M}_?^A$ representation. This result follows from the uniqueness of $\mathcal{M}_{tuple}$ (Theorem 9): If there is an $\mathcal{M}_?^A$ representation not obtained by conversion of tuple-ors in R to attribute-ors, it can be mapped to a different $\mathcal{M}_{tuple}$ representation R', violating uniqueness. □

9.2 Approximating a Set of Possible Instances

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

Consider a set of instances P and an $\mathcal{M}_{tuple}$ relation R with instances $I(R)$. A "best" approximation R of P could be defined in several ways:

- *Closest-Set:* R such that $I(R)$ is "as close to" P as possible
- *Maximal-Subset:* maximal R such that $I(R) \subseteq P$
- *Minimal-Superset:* minimal R such that $I(R) \supseteq P$

For the closest-set definition we consider the Jaccard measure of similarity between two sets S_1 and S_2, given by

$\frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$. Under this measure of similarity, we shortly state a result showing that even the best approximations can be very bad.

For the maximal-subset and minimal-superset definitions, we use $|R|$, the number of A-tuples, as our metric for maximality/minimality of R. We restrict ourselves to R where for each t, the number of A-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 tuple-or of R. (Without this condition, a trivial best minimal-superset approximation is obtained by including n tuple-ors, each having all possible tuples in R, where n is the maximum cardinality of any instance in P. Such a representation is likely to have many spurious instances, but $|R|$ is small.)

Lemma 2 There exists set of instances P for which there is no constant-factor closest-set $\mathcal{M}_{tuple}$-approximation R under the Jaccard measure of similarity between P and $I(R)$. □

The proof appears in the appendix. Here we give the intuition. Consider n instances $\{P_1, P_2, \ldots, P_n\}$ for a relation with two attributes. Let the ith instance have two tuples, $[i, 1]$ and $[i, 2]$. This set of instances does not admit any constant-factor approximation R in $\mathcal{M}_{tuple}$: If $I(R)$ contains all n of the possible instances, then R would need to have $O(n^2)$ instances in total.

Given the negative result under the closest set definition and to preserve the possible instances, in the following, we use the minimal superset definition.

Example 14 Recall the approximation in Example 13. It contains all instances of the desired uncertain relation (and one additional instance, the empty instance). Clearly any $\mathcal{M}^A_?$ or $\mathcal{M}_{tuple}$ approximation with fewer than two tuples cannot contain all instances of the uncertain relation. Thus the given representation is a best minimal-superset approximation. □

Lemma 3 There always exists some R in $\mathcal{M}_{tuple}$ such that $I(R) \supseteq P$. □

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 the proof above may give a poor approximation, with $|R|$ being larger than the minimum possible.

Lemma 4 The best approximation R for a set of instances P is not unique. □

Proof Consider P having a single instance. The corresponding deterministic relation R is a best approximation. However, annotating each tuple with '?' is also a best approximation. □

We have the following result for finding best approximations, according to the minimal superset definition.

Theorem 19 (Best $\mathcal{M}_{tuple}$ Approximation for P)

1. Finding a best $\mathcal{M}_{tuple}$ approximation for an arbitrary set of possible instances P is NP-hard.
2. $\mathcal{M}_{tuple}$ approximation can be reduced to the minimum graph coloring problem, which admits a 5/7-differential approximation. □

The proof of hardness, presented in the appendix, is shown by a reduction from the NP-hard minimum graph coloring problem. In the appendix we then show the inverse reduction to the graph coloring problem.

9.3 Approximating One Model in Another

We next consider the problem of directly approximating $\mathcal{M}^A_{prop}$ relations in $\mathcal{M}_{tuple}$ and in $\mathcal{M}_2$. Given the negative result (Lemma 2) for the closest-set definition of approximation, let us continue with the minimal-superset definition.

Approximating in $\mathcal{M}_{tuple}$

Given an $\mathcal{M}^A_{prop}$ relation R, we seek a minimal $\mathcal{M}_{tuple}$ relation S such that $I(S) \supseteq I(R)$. We can obtain an $\mathcal{M}_{tuple}$ relation S by eliminating the constraints f in R, expanding out the attribute-ors to form tuple-ors, and adding "?"s to each of them. Since all satisfying assignments of f still correspond to instances in S, S is a superset, i.e., $I(S) \supseteq I(R)$. However, S may contain many spurious instances that are not in $I(R)$. As a simple example, consider R having n distinct A-tuples $\{t_1, \ldots, t_n\}$, each with two tuple instances, and the constraint $f = (t_1 \wedge \neg t_2 \wedge \ldots \wedge \neg t_n)$. R has exactly two possible instances, but the approximation S obtained as described above contains 2^n instances.

The following lemma shows that finding the best (closest-set or maximal-subset or minimal-superset) $\mathcal{M}_{tuple}$ approximation is a hard problem.

Lemma 5 Finding the best $\mathcal{M}_{tuple}$ representation for an arbitrary $\mathcal{M}^A_{prop}$ relation is NP-hard. □

Proof We show a simple reduction from SAT. Given an instance f of SAT, we construct an $\mathcal{M}^A_{prop}$ relation R with the set of tuples being the set of variables in f, and the constraints in R being f itself. We now wish to find the best $\mathcal{M}_{tuple}$ approximation of R. Since R has at least one

possible instance if and only f is satisfiable, f is unsatisfiable if and only if the best approximation of R is the empty $\mathcal{M}_{tuple}$ instance (i.e., no tuple-ors). Therefore, we have a polynomial-time reduction of SAT to finding the best $\mathcal{M}_{tuple}$ approximation of an $\mathcal{M}^A_{prop}$ relation. □

For this NP-hard problem, we can apply Theorem 19: We convert the $\mathcal{M}^A_{prop}$ relation to a set of possible instances (in a potentially exponential process) and then using Theorem 19 we obtain a 5/7-differentially approximate $\mathcal{M}_{tuple}$ relation S.

Approximating in $\mathcal{M}_2$

Finally we look briefly at the problem of approximating $\mathcal{M}^A_{prop}$ relations in $\mathcal{M}_2$. Consider an $\mathcal{M}^A_{prop}$ relation R. First we transform the A-tuples with attribute-ors in R into ordinary tuples and constraints over them, as follows: For every A-tuple t in R, perform the cross-product of the attribute-ors to obtain regular tuples $\{t_1, t_2, \ldots, t_n\}$. Then, to the set of constraints f, we do the following:

1. Add constraints $\neg t_i \vee \neg t_j$ for every $1 \leq i < j \leq n$
2. Replace every occurrence of t with $t_1 \vee t_2 \vee \ldots \vee t_n$
3. For every clause C containing $\neg t$, replace C with the set of n clauses $\{C_1, \ldots, C_n\}$, where C_i is obtained by replacing $\neg t$ with $\neg t_i$

We then have an equivalent $\mathcal{M}^A_{prop}$ relation R' but with only ordinary tuples. We are now interested in approximating the general propositional formula in R' into a 2-CNF formula. Although in general finding best approximations for propositional theories into tractable classes is known to be a hard problem, approximation techniques have been proposed in the past. Specifically, [42] describes techniques for finding 2-CNF lower and upper bounds f_1 and f_2 for a general propositional formula f, i.e., the satisfying assignments of f_1 (f_2 respectively) are a subset (superset respectively) of the satisfying assignments of f. These techniques try to minimize the number of differing satisfying assignments (which correspond to possible instances in our case), and not the number of variables (which would correspond to the number of tuples). The algorithms in [42] proceed in an online fashion, progressively giving better approximations until finding the best. We can employ these techniques to approximate the set of constraints f in R', and thus obtain subset and superset approximate $\mathcal{M}_2$ relations R_1 and R_2 of R, i.e., $I(R_1) \subseteq I(R)$ and $I(R_2) \supseteq I(R)$.

10 Related Work

The study of uncertain databases has a long history, dating back to a series of initial papers from the early 1980's, e.g., [2,11,15,38,58], and a great deal of follow-on work, e.g., [10,12,16,23,30,37,38,43,44,46,48,56]. Much of this previous work lays theoretical foundations and considers query answering, e.g., [2,3,35,58]. Systems based around uncertain data are discussed in [6,13,19,40,43,60,59].

In this paper, we introduced a space of uncertain-data models based on tuple-level uncertainty constructs and tuple-existence constraints. We then studied the problems of closure and relative expressive power, completeness, uniqueness, equivalence, minimization, and approximation. There have been few papers addressing some of these problems in isolation for specific data models, which we describe next. However, we are not aware of any previous work that attempts to understand a space of models (or even a single model for that matter) by studying all of the above problems, which is the goal of this paper.

Closure and Expressiveness: Most previous work has focused on complete models, and does not explore different incomplete models in terms of relative expressiveness and closure properties, one of the contributions of our paper. Several incomplete data models (such as v-tables, Codd tables, i-tables) studied along with c-tables [3,38] can be thought of as comprising a "space of models." However, these models were studied primarily in isolation and past work does not relate them in terms of closure and expressiveness.

Recently, [37] studied closure and completeness of incomplete models in detail, and independently obtained a generalization of our Theorem 4 that connects closure to completeness. Interestingly, their notion of "algebraic completion" makes incomplete models become complete by taking their closure under a minimal set of operations, which is in the same spirit as the transition diagram of Section 5.

Equivalence, Minimization, and Approximation: In general, most complete models are easily seen to be non-unique, hence, equivalence testing, minimization, and approximation are the most interesting problems for complete models. Reference [3] studies the complexity of *containment* checking for c-tables, i.e., determining whether the set of possible worlds represented by one uncertain relation is contained in that of another. The problem of equivalence testing can be solved using containment. However, minimization and approximation are not discussed in [3].

Reference [7] studies the problem of finding *maximal decompositions* of an uncertain relation represented as a "world-set decomposition" (called the *gWSD* data model). Maximal decompositions in their setting can be thought as minimizing the representation. The paper gives a PTIME maximal decomposition algorithm. In contrast, we show that when uncertain relations are represented using arbitrary propositional constraints, minimization is intractable. Then, we show that reducing the number of tuples also reduces

the size of the minimal constraint, and we study incremental minimality, which are not considered in [7].

Reference [53] studies a limited form of approximation in uncertain databases. They present a new data model, called *BID tables*, and address the problem of approximating uncertain relations as BID tables. They show that determining whether BID tables are sufficiently expressive for a given uncertain relation is intractable. The paper also proposes a *partial BID* representation to further approximate a set of possible worlds, and studies when a partial BID view represents a unique probability distribution. In this paper we consider the problem of approximation in more generality and depth: We consider the problem of approximating uncertain relations (represented as a set of possible instances or in any data model) into a less expressive data model. We define the notion of a *best approximation* and address the problem of finding a best approximation. Since in general finding the best approximation is intractable, we give PTIME algorithms to approximate the best approximation.

Trio: 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 [60]. The enumeration of the space of models, their expressiveness hierarchy, and related membership problems were studied in an initial Trio paper on models for uncertainty [25]. Although the approximation problem was suggested in [25], it was not solved, and the other problems we consider here—uniqueness, equivalence, and minimization—were not discussed at all. Note that the material on uniqueness, equivalence, and minimization first appeared as a technical report [26], but is unpublished.

Other Work: Like uncertain databases, the related area of *probabilistic databases* has been experiencing revived interest, especially in query answering [22–24]. Work in probabilistic databases also has not, to the best of our knowledge, considered the problems we address in this paper. The models we consider in this paper do not allow for 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., [4, 8,9,14,20,28,36,61], in which the possible "minimal repairs" [9,17,61] 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 also related, e.g., [29,50,54]. Again, our work in this paper focuses on a specific set of problems associated with models for representing 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., [5,30,32,57]. This body of work differs from ours in that we look at modeling uncertainty and querying it exactly, as opposed to modeling exact data and querying it approximately.

11 Conclusions and Future Work

This paper addressed a number of problems that arise in the representation of uncertain data. We defined a space of models obtained by combining basic constructs and studied in detail their closure and completeness properties and relative expressiveness. We also gave a result connecting closure and completeness for a broad class of models. For a representative set of four models in this space, we further explored the problems of uniqueness, equivalence, minimization, and approximation, obtaining complexity results and providing algorithms for tractable cases.

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

- As mentioned in Section 10, the area of probabilistic databases is of great interest. Uncertain databases can be extended to include probabilities [10,13,18,32,43], 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 models and operations (Section 8). The problem remains open for other operations and models, as well as for representations with duplicates.
- For the problem of approximating an uncertain relation in an incomplete model, we considered primarily $\mathcal{M}_{tuple}$ (Section 9). We have not yet considered approximations in $\mathcal{M}_?^A$ in any detail, nor other incomplete models in our space. Similarly, we have not considered, for all combinations of models, the problem of directly converting uncertain relations in a more expressive model into approximations in weaker ones.
- We have not addressed the case of representing correlations across uncertain relations. Extending our work to arbitrary correlations is an important direction of further work.

Acknowledgments

We are very grateful to the anonymous reviewers of a previous draft of this manuscript, and the editors Dan Suciu and Peter Haas, for their insightful comments. We also thank all the members of the Trio project for numerous useful discussions.

References

1. Christmas Bird Count Homepage. *http://www.audobon.org/bird/cbc/*.
2. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
3. S. Abiteboul, P. Kanellakis, and G. Grahne. On the Representation and Querying of Sets of Possible Worlds. *Theoretical Computer Science*, 78(1), 1991.
4. S. Agarwal, A. M. Keller, G. Wiederhold, and K. Saraswat. Flexible Relation: An Approach for Integrating Data from Multiple, Possibly Inconsistent Databases. In *Proc. of ICDE*, 1995.
5. S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated Ranking of Database Query Results. In *Proc. of CIDR*, 2003.
6. L. Antova, C. Koch, and D. Olteanu. MayBMS: Managing Incomplete Information with Probabilistic World-Set Decompositions. In *Proc. of ICDE*, 2007.
7. L. Antova, C. Koch, and D. Olteanu. World-set decompositions: Expressiveness and efficient algorithms. In *Proc. of ICDT*, 2007.
8. M. Arenas, L. Bertossi, and J. Chomicki. Answer sets for consistent query answering in inconsistent databases. *TPLP*, 3(4), 2003.
9. M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent Query Answers in Inconsistent Databases. In *Proc. of ACM PODS*, 1999.
10. D. Barbará, H. Garcia-Molina, and D. Porter. The Management of Probabilistic Data. *TKDE*, 4(5), 1992.
11. R. S. Barga and C. Pu. Accessing Imprecise Data: An Approach Based on Intervals. *IEEE Data Engineering Bulletin*, 16(2), 1993.
12. O. Benjelloun, A. Das Sarma, A. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In *Proc. of VLDB*, 2006.
13. 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.
14. 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.
15. B. P. Buckles and F. E. Petry. A Fuzzy Model for Relational Databases. *International Journal of Fuzzy Sets and Systems*, 7, 1982.
16. D. Burdick, P. M. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. OLAP over uncertain and imprecise data. *J. VLDB*, 16(1):123–144, 2007.
17. 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.
18. R. Cavallo and M. Pittarelli. The theory of probabilistic databases. In *Proc. of VLDB*, 1987.
19. R. Cheng, S. Singh, and S. Prabhakar. U-DBMS: A database system for managing constantly-evolving data. In *Proc. of VLDB*, 2005.
20. J. Chomicki and J. Marcinkowski. Minimal-change integrity maintenance using tuple deletions.
21. E. F. Codd. Extending the Database Relational Model to Capture More Meaning. *ACM Transactions on Database Systems*, 4(4), 1979.
22. N. Dalvi, G. Miklau, and D. Suciu. Asymptotic Conditional Probabilities for Conjunctive Queries. In *Proc. of ICDT*, 2005.
23. N. Dalvi and D. Suciu. Efficient Query Evaluation on Probabilistic Databases. In *Proc. of VLDB*, 2004.
24. N. Dalvi and D. Suciu. Answering Queries from Statistics and Probabilistic Views. In *Proc. of VLDB*, 2005.
25. A. Das Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working Models for Uncertain Data. In *Proc. of ICDE*, 2006.
26. A. Das Sarma, S. Nabar, and J. Widom. Representing uncertain data: Uniqueness, equivalence, minimization, and approximation. Technical report, Stanford InfoLab, 2005. Available at http://dbpubs.stanford.edu/pub/2005-38.
27. L. G. DeMichiel. Resolving Database Incompatibility: An Approach to Performing Relational Operations over Mismatched Domains. *IEEE Transactions on Knowledge and Data Engineering*, 1(4), 1989.
28. P. M. Dung. Integrating data from possibly inconsistent databases. In *COOPIS '96: Proceedings of the First IFCIS International Conference on Cooperative Information Systems*, 1996.
29. N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proc. of IJCAI*, 1999.
30. N. Fuhr. A Probabilistic Framework for Vague Queries and Imprecise Information in Databases. In *Proc. of VLDB*, 1990.
31. N. Fuhr and T. Rölleke. A Probabilistic NF2 Relational Algebra for Imprecision in Databases. *Unpublished Manuscript*, 1997.
32. N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM TOIS*, 14(1), 1997.
33. M. R. Garey and D. S. Johnson. Computers and Intractability. *W. H. Freeman and Company*, 1979.
34. G. Grahne. Dependency Satisfaction in Databases with Incomplete Information. In *Proc. of VLDB*, 1984.
35. G. Grahne. Horn Tables - An Efficient Tool for Handling Incomplete Information in Databases. In *Proc. of ACM PODS*, 1989.
36. G. Greco, S. Greco, and E. Zumpano. A logical framework for querying and repairing inconsistent databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(6).
37. T. J. Green and V. Tannen. Models for incomplete and probabilistic information. In *Proc. of IIDB Workshop*, 2006.
38. T. Imielinski and W. Lipski. Incomplete Information in Relational Databases. *Journal of the ACM*, 31(4), 1984.
39. T. Imielinski, S. Naqvi, and K. Vadaparty. Incomplete objects - data model for design and planning applications. In *Proc. of ACM SIGMOD*, 1991.
40. R. Jampani, L. Perez, M. Wu, F. Xu, C. Jermaine, and P. J. Haas. Mcdb: A monte carlo approach to managing uncertain data. In *Proc. of ACM SIGMOD*, 2008.
41. M. Karnaugh. The map method for synthesis of combinational logic circuits. *Trans. AIEE. pt I*, 1953.
42. H. Kautz and B. Selman. Knowledge Compilation and Theory Approximation. *Journal of the ACM*, 1996.
43. L. V. S. Lakshmanan, N. Leone, R. Ross, and V.S. Subrahmanian. ProbView: A Flexible Probabilistic Database System. *ACM TODS*, 22(3), 1997.
44. S. K. Lee. An extended Relational Database Model for Uncertain and Imprecise Information. In *Proc. of VLDB*, 1992.
45. L. Libkin and L. Wong. Semantic representations and query languages for or-sets. In *Proc. of ACM PODS*, 1993.
46. K. Liu and R. Sunderraman. Indefinite and Maybe Information in Relational Databases. *ACM TODS*, 1990.
47. E. J. McCluskey. Minimization of boolean functions. *The Bell System Technical Journal*, 1956.
48. A. Motro. Management of Uncertainty in Database Systems. *Modern Database Systems: The Object Model, Interoperability, and Beyond*, 1994.
49. V. Th. Paschos. Polynomial approximation and graph-coloring. *Computing*, 70(1), 2003.
50. J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.
51. William Purdy. A Logic for Natural Language. *Journal of Formal Logic*, 32(1), 1991.
52. W. Quine. The problem of simplifying truth functions. *American Math Monthly*, 59(1), 1952.
53. C. Re and D. Suciu. Materialized views in probabilistic databases for information exchange and query optimization. In *Proc. of VLDB*, 2007.
54. S. Sanghai, P. Domingos, and D. Weld. Dynamic probabilistic relational models. In *Proc. of IJCAI*, 2003.

55. Renate A. Schmidt. Relational Grammars for Knowledge Representation. In M. Böttner and W. Thümmel, editors, *Variable-Free Semantics*, volume 3 of *Artikulation und Sprache*, pages 162–180. secolo Verlag, Osnabrück, Germany, 2000.
56. P. Sen and A. Deshpande. Representing and Querying Correlated Tuples in Probabilistic Databases. In *Proc. of ICDE*, 2007.
57. A. Theobald and G. Weikum. The XXL Search Engine: Ranked Retrieval of XML Data Using Indexes and Ontologies. In *Proc. of ACM SIGMOD*, 2002.
58. M. Y. Vardi. Querying logical databases. In *Proc. of ACM PODS*, 1985.
59. Daisy Zhe Wang, Eirinaios Michelakis, Minos Garofalakis, and Joseph M. Hellerstein. Bayesstore: Managing large, uncertain data repositories with probabilistic graphical models. In *Proc. of VLDB*, 2008.
60. J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In *Proc. of CIDR*, 2005.
61. J. Wijsen. Condensed representation of database repairs for consistent query answering. In *Proc. of ICDT*, 2003.

A Closure and Expressiveness Proofs

Theorem 1: The closure properties of our models are specified in Table 1. If "Y" appears in the row for operation Op and column for model $\mathcal{M}$, then $\mathcal{M}$ is closed under the operation Op, otherwise it is not. □

Proof We prove the closure properties of our models under each of the operations. Table 3 shows the closure properties of each of the models; we have excluded the complete model $\mathcal{M}^A_{prop}$ and expanded out the last column from Table 1 as different examples are required to show non-closure for the different models.

An entry of "N" signifying non-closure is shown by the corresponding counter-example number as follows. For proofs of some of the cases we shall use the properties from Table 2 (Section 5).

1. We first show non-closure of duplicate elimination for $\mathcal{M}^A$, $\mathcal{M}^A_?$, $\mathcal{M}_{tuple}$, and $\mathcal{M}_{\oplus\equiv}$, and then for $\mathcal{M}^A_2$.

   $\mathcal{M}^A$, $\mathcal{M}^A_?$, $\mathcal{M}_{tuple}$, $\mathcal{M}_{\oplus\equiv}$: Consider the following relation R with two A-tuples in any of the models:
   `R: [{a,b}], [{a,c}]`
   In case of $\mathcal{M}^A$ and $\mathcal{M}^A_?$ R has two A-tuples, in case of $\mathcal{M}_{tuple}$ two tuple-ors, and in case of $\mathcal{M}_{\oplus\equiv}$ R is represented using mutual exclusion. Duplicate elimination in R produces the following set of possible instances:

   ```
   I1: [a]
   I2: [a],[c]
   I3: [a],[b]
   I4: [b],[c]
   ```

   The above set of instances does not satisfy the complementation property, which is satisfied by all uncertain relations representable in $\mathcal{M}_{\oplus\equiv}$. Hence the above is not representable in $\mathcal{M}_{\oplus\equiv}$. Next we show that the set of instances is not representable in $\mathcal{M}_{tuple}$, and since $\mathcal{M}_{tuple}$ is more expressive than each of $\mathcal{M}^A$, $\mathcal{M}_?$, and $\mathcal{M}^A_?$, the result is also not representable in each of $\mathcal{M}^A$, $\mathcal{M}_?$, and $\mathcal{M}^A_?$. Suppose that the above result is representable in $\mathcal{M}_{tuple}$ relation R. Since the only cardinality of instances are 1 and 2, there are two tuple-ors in R one with a "?" (call it t_1) and one without (call it t_2). Now since I1 is the only single tuple instance, t_2 is the regular tuple `[a]`. This is a contradiction because I4 does not have the tuple `[a]`.

   $\mathcal{M}^A_2$: Consider the following $\mathcal{M}^A_2$ relation R with two A-tuples and no constraints:
   `R: [{a,b,c}], [{a,b,c}]`
   Duplicate elimination of R produces the following possible instances:

   ```
   I1: [a]
   I2: [b]
   I3: [c]
   I4: [b],[c]
   I5: [a],[c]
   I6: [a],[b]
   ```

   To represent the above set of instances, we effectively need to encode the constraints $(\neg a \vee \neg b \vee \neg c)$ and $(a \vee b \vee c)$. These constraints cannot be represented using two clauses because disjoint set of 3-clauses (that cannot be resolved) cannot be expressed using sets of 2-clauses. And in the above set of instances, using attribute-ors also does not help, because adding attribute-ors to any of the tuples also results in additional instances.
2. Let $R_1(A)$ have two tuples—t_1:`[x]` and t_2:`[z]`—and constraint $f = t_2$. Let $R_2(A)$ have two tuples—s_1:`[y]` and s_2:`[w]`—and constraint $g = s_2$. On taking the cross-product of R_1 and R_2, we have the following set of instances:

   ```
   I1: [w,z]
   I2: [w,x],[w,z]
   I3: [y,z],[w,z]
   I4: [y,x],[y,z],[w,x],[w,z]
   ```

   Since the set of instances above do not satisfy the path connectedness property satisfied by $\mathcal{M}_?$ and $\mathcal{M}^A_?$, it is not representable in either of $\mathcal{M}_?$ and $\mathcal{M}^A_?$.
3. Consider relations $R_1(X, Y)$ and $R_2(X, Y)$ with the following possible instances, both of which can be represented in $\mathcal{M}_2$ and $\mathcal{M}^A_2$:

   ```
   R1:
   I1: [a,a],[b,b]
   I2: [c,c]
   ```

Closure-Model	$\mathcal{M}^A$	$\mathcal{M}_?$	$\mathcal{M}_?^A$	$\mathcal{M}_{\oplus\equiv}$	$\mathcal{M}_2$	$\mathcal{M}_{tuple}$	$\mathcal{M}_2^A$
Union	Y	Y	Y	Y	Y	Y	Y
Select(ee)	Y	Y	Y	Y	Y	Y	Y
Select(ea)	N[10]	Y	Y	Y	Y	Y	Y
Select(aa)	N[9]	Y	N[9]	Y	Y	Y	Y
Intersection	N[7]	Y	N[7]	N[8]	N[3]	N[13]	N[3]
Cross-Product	Y	N[2]	N[2]	N[14]	N[4]	N[13]	N[4]
Join	N[6]	N[11]	N[6]	N[8]	N[3]	N[13]	N[3]
Difference	N[15]	Y	N[15]	N[15]	N[15]	N[15]	N[15]
Projection	Y	Y	Y	Y	Y	Y	Y
Duplicate Elimination	N[1]	Y	N[1]	N[1]	Y	N[1]	N[1]
Aggregation	N[5]	N[12]	N[5]	N[5]	N[5]	N[5]	N[5]

Table 3 Closure and Expressiveness of Models

`R2:`
`I1: [b,b],[c,c]`
`I2: [a,a],[c,c]`

The intersection or natural join of R_1 and R_2 produces the following set of instances:

`I1: [a,a]`
`I2: [b,b]`
`I3: [c,c]`

We know that the above set of instances cannot be represented using 2-clauses as it is a 3-way xor. Also, attribute-ors cannot be used, as then additional tuples would be added to instances. Hence $\mathcal{M}_2$ and $\mathcal{M}_2^A$ are not closed under join or intersection.

4. Since a natural join can be performed as a cross-product followed by a selection, and $\mathcal{M}_2$ and $\mathcal{M}_2^A$ are closed under selection but not join, they are both not closed under cross-product.
5. Let $R(X, Y)$ be:
   `[{a,b},1]`
   `[{b,c},1]`
   The result of performing the aggregation "*sum Y group by X*" gives the following set of possible instances:

   `I1: [a,1],[b,1]`
   `I2: [b,1],[c,1]`
   `I3: [a,1],[c,1]`
   `I4: [b,2]`

   These set of instances is not representable in $\mathcal{M}^A$, $\mathcal{M}_?$, $\mathcal{M}_?^A$, and $\mathcal{M}_{tuple}$ as it does not satisfy the path connectedness property satisfied by all the models. Further, since the possible instances also do not satisfy the complement property, it is not representable in $\mathcal{M}_{\oplus\equiv}$.
   Now consider a relation $R(X, Y)$ which has all 7 non-empty sets of possible instances consisting of tuples `[a,a]`, `[b,b]`, and `[c,c]`. On performing the aggregation "*select min*" gives the instances:

   `I1: [a,a]`
   `I2: [b,b]`
   `I3: [c,c]`

   As mentioned in Case 3 above, these are not representable in $\mathcal{M}_2$ and $\mathcal{M}_2^A$.
6. Let $R_1(X)$ have one A-tuple `[{a,b}]`, and $R_2(X, Y)$ have two tuples `[a,1]` and `[a,2]`. The natural join of R_1 and R_2 gives two possible instances—`{[a,1],[a,2]}` and `{}`—that do not satisfy the path connectedness property and hence is not representable in $\mathcal{M}^A$ or $\mathcal{M}_?^A$.
7. Let $R_1(X, Y)$ have one A-tuple: `[{a,b},{a,b}]` and let $R_2(X, Y)$ have two tuples: `[a,a]` and `[b,b]`. $R_1 \cap R_2$ has the following possible instances clearly not representable in $\mathcal{M}^A$ or $\mathcal{M}_?^A$:

   `I1: [a,a]`
   `I2: [b,b]`

8. Let $R_1(X)$ have tuples `[a]`, `[b]`, and `[c]`, with constraint: $(b \equiv c)$ (mutual inclusion) and $R_2(X)$ have the same three tuples `[a]`, `[b]`, and `[c]`, but with constraint: $(b \oplus c)$ (mutual exclusion). The join or intersection of R_1 and R_2 produces the following set of instances:

   `I1: empty`
   `I2: [a]`
   `I3: [b]`
   `I4: [c]`
   `I5: [a],[b]`
   `I6: [a],[c]`

   Since this set of instances does not satisfy the complement property satisfied by $\mathcal{M}_{\oplus\equiv}$, the result is not representable in $\mathcal{M}_{\oplus\equiv}$.
9. Let $R(X, Y)$ have the single A-tuple: `[{a,b},{a,b}]`. Performing selection on R with the predicate $(X = Y)$ gives exactly one of `[a,a]` and `[b,b]` in the result, which cannot be represented in $\mathcal{M}^A$ or $\mathcal{M}_?^A$.
10. Let $R(A)$ have one A-tuple: `[{a,b}]`. Selection from R with the predicate $(A = a)$ gives a maybe-tuple not representable in $\mathcal{M}^A$.
11. Let $R_1(A)$ contain one maybe-tuple: `[x]`**?**. Let $R_2(A)$ contain two tuples: `[x]` and `[x]`. The natural join of

R_1 and R_2 has two possible instances—`I1:empty, I2:[x],[x]`—that do not satisfy the path connectedness property and hence are not representable in $\mathcal{M}_?$.

12. Let $R_1(A)$ contain one maybe-tuple `[1]`**?** and one regular tuple `[1]`. Performing the "*sum*" aggregation, we obtain two possible instances each with one tuple violating the unique minimum property. Hence the result is not representable in $\mathcal{M}_?$.
13. Let $R_1(X)$ have one tuple-or $\{$`[a]` $\|$ `[b]`$\}$ and $R_2(X,Y)$ have two tuples `[a,1]` and `[a,2]`. The join of R_1 and R_2 gives possible instances:

    ```
    I1: empty
    I2: [a,1],[a,2]
    ```

    This set of instances does not satisfy the path connectedness property, and hence is not representable in $\mathcal{M}_{tuple}$. If we instead do the cross-product, instance I1 becomes `[b,a,1],[b,a,2]` and I2 `[a,a,1],[a,a,2]` still violating the path connectedness property.
    For intersection consider $R_1(A)$:
    $\{$`[a]` $\|$ `[c]`$\}$
    $\{$`[b]` $\|$ `[d]`$\}$
    and $R_2(A)$:
    $\{$`[a]` $\|$ `[b]`$\}$
    $\{$`[c]` $\|$ `[d]`$\}$
    The intersection has the set of instances:

    ```
    I1: empty
    I2: [a]
    I3: [b]
    I4: [c]
    I5: [d]
    I6: [a],[d]
    I7: [b],[c]
    ```

    The above set of instance is not representable in $\mathcal{M}_{tuple}$ as we need at least two tuple-ors, both maybe-tuples and two alternatives, which gives at least four instances with two tuples.
14. Since a join can be performed as a cross-product followed by selection and $\mathcal{M}_{\oplus\equiv}$ is closed under selection but not join, therefore $\mathcal{M}_{\oplus\equiv}$ is also not closed under cross-product.
15. Since the set intersection of R_1 and R_2 can be expressed as $R_1 - (R_1 - R_2)$, any model that is not closed under set intersection is also not closed under set (and hence bag) difference.

We now justify the *Y* entries in the table. Let us first consider $\mathcal{M}^A$. The bag union is obtained by forming the union of the A-tuples in the input relations. Selection with both operands being exact is performed by eliminating A-tuples not satisfying the select condition. Cross-product is performed by taking every pair of A-tuples from the input relations and concatenating them. Bag projection is done by simply projecting out the attributes from each of the A-tuples.

Let us now consider the $\mathcal{M}_?$ column. Bag union is performed by just unioning the tuples from the input relations, retaining the **?** annotations on each tuple. Set union can be done by performing the closed operation of duplicate elimination after this. Duplicate elimination can be performed by retaining only one copy of each tuple appearing multiple times. A label of **?** is applied if all its copies had the **?** label, and otherwise there is not such label. Since $\mathcal{M}_?$ is closed under duplicate elimination, if a bag operation displays closure, so does the corresponding set operation. Intersection of two $\mathcal{M}_?$ relations can be done in a similar way by considering only the tuples that appear in both the input relations (with or without **?** labels). All kinds of selection are equivalent for $\mathcal{M}_?$ relations as there are no A-tuples, and these can be done by eliminating the tuples that do not satisfy the selection predicate. Difference is performed in a way similar to intersection; look at all the tuples in the first relation and add multiple copies of them in the result depending on the number of times this tuple appears in the second relation without a **?** annotation. And by looking at the **?** annotations of the inputs, decide how many of the resulting tuples should be labeled with **?**. Projection is performed by simply projecting out the attributes from the result.

Let us now look at $\mathcal{M}_?^A$. Bag projection and bag union can be performed easily by projecting out the attributes, or taking the union of the set of A-tuples with their annotations respectively. Selection with one of the operands being approximate is performed by applying the predicate to each A-tuple. The key point is that after application of this predicate, the result can also be represented as an A-tuple, possible with the addition of a **?** label in case there are instances of the A-tuple that do not satisfy the predicate. Since selection with one operand being approximate can be done, selection with both operands being exact also displays closure.

Let us now look at $\mathcal{M}_{\oplus\equiv}$. Once again, bag projection and bag union can be performed easily: The union of the tuple variables is taken (with renaming if required to avoid same names for variables from different relations), and the constraints are the conjunction of the input constraints. Since $\mathcal{M}_{\oplus\equiv}$ does not have A-tuples, the three kinds of selection can be treated equivalently, just assuming that each tuple in the input either satisfies the selection predicate or not. We need to show that the result of applying such a predicate and eliminating some tuples can be performed resulting in another $\mathcal{M}_{\oplus\equiv}$ relation. This is done by eliminating variables corresponding to removed tuples from the constraints one by one. To eliminate a variable, first add the transitive closure of all equivalence constraints and then just drop the clauses containing the variable to be eliminated.

We now consider $\mathcal{M}_2$. Just like $\mathcal{M}_{\oplus\equiv}$, closure under bag projection and union can be seen. Closure under selection is also very similar to $\mathcal{M}_{\oplus\equiv}$. First the tuples that do not satisfy the selection predicate are identified, and then the corresponding variables are eliminated using the standard resolution variable elimination technique. Note that eliminating variables from 2-CNF using resolution keeps the resulting formula also in 2-CNF. Next we show closure under duplicate elimination, implying closure under set union and projection as well. For every set of tuples $t_1, t_2, \ldots, t_m$ with the same value, we can consider two tuples at a time. Since we can merge two tuples, say t_1 and t_2 eliminating duplicates, we can eliminate all the m possible duplicates one after the other. Doing this for every value, we get the desired result.

Let us not look at the column for $\mathcal{M}_{tuple}$. Closure under bag union and bag projection can again be seen easily. For selection, we look at each entry (set of tuples) in the input $\mathcal{M}_{tuple}$ relation and retain the tuples that satisfy the selection predicate for the result. A **?** label is added to the entry (if not already present) in case there are tuples in the entry that do not satisfy the selection predicate.

Finally, consider the column for $\mathcal{M}_2^A$. $\mathcal{M}_2^A$is closed under bag projection and bag union because the results are obtained by applying the projection on each tuple individually, and taking the union of the A-tuples and conjunction of the constraints respectively. We now show closure under selection with both operands possibly being approximate. For A-tuples with no instance satisfying the selection predicate, the corresponding variables are eliminated as in the case for $\mathcal{M}_2$. For A-tuples with every instance satisfying the selection predicate, no change is made to the variable. Finally, we need to look at A-tuples some of whose instances satisfy the selection predicate, and some do not. Let one such A-tuplebe denoted by t. The first step is to split this A-tupleinto its instances satisfying the selection predicate, and call them say $t_1, t_2, \ldots, t_n$. The next step is to replace all occurrences of t in the constraints as follows. Drop the clauses in which t appears positively, and for each clause where t appears negatively, make n copies and replace $\neg t$ with $\neg t_i$ for $1 \leq i \leq n$. Lastly, add constraints of the form $\neg t_j \vee \neg t_k$ where $1 \leq j < k \leq n$. □

Theorem 7: Recall the expressiveness hierarchy depicted in Figure 2(b). If there is an arrow from model $\mathcal{M}_1$ to model $\mathcal{M}_2$, then $\mathcal{M}_2$ is more expressive than $\mathcal{M}_1$. If $\mathcal{M}_1$ and $\mathcal{M}_2$ are not ancestor/descendant pairs, then neither $\mathcal{M}_1$ nor $\mathcal{M}_2$ is as expressive as the other. □

Proof We first give justifications for transitions to a state other than $\mathcal{M}_{prop}^A$, and then give examples to justify the other transitions. Performing $Select(aa)$ on either an $\mathcal{M}^A$ relation or $\mathcal{M}_?^A$ relation gives results not expressible in $\mathcal{M}^A$ as shown by Example 9 above. $\mathcal{M}_{tuple}$ is however closed on $Select(aa)$, and so these transition to $\mathcal{M}_{tuple}$. $Select(ea)$ on $\mathcal{M}^A$ raises the need for maybe-tuples, and so transition to $\mathcal{M}_?^A$, which is closed under $Select(ea)$. Aggregation on $\mathcal{M}_?$ gives one of several possible values for each group which can be represented by or-sets, and thus in $\mathcal{M}_?^A$.

We now give examples to justify each of the transitions to $\mathcal{M}_{prop}^A$:

1. Transition from $\mathcal{M}^A$ to $\mathcal{M}_{prop}^A$ on join and intersection. The natural join and intersection of R_1 and R_2 are not expressible in $\mathcal{M}_2^A$, where R_1 and R_2 are as follows. R_1 has one A-tuple: `[{a,b,c},{a,b,c}]`, and R_2 has three tuples: `[a,a]`, `[b,b]`, and `[c,c]`, giving three possible instances in the result: `I1:[a,a]`, `I2:[b,b]`, and `I3:[c,c]`.
   Further, the join and intersection cannot be expressed in $\mathcal{M}_{tuple}$, from the construction in Example 13 from the non-closure proofs, and so the only more expressive model remaining is $\mathcal{M}_{prop}^A$.
2. Transition from $\mathcal{M}^A$ to $\mathcal{M}_{prop}^A$ on difference. Since the intersection can be expressed by a sequence of differences and under intersection $\mathcal{M}^A$ transitions to $\mathcal{M}_{prop}^A$, even under difference we have the same transition.
3. Transition from $\mathcal{M}_?$ to $\mathcal{M}_{prop}^A$ on cross-product and join. The set of instances in Example 11 from non-closure above does not satisfy the path connectedness property and hence cannot be represented in any of $\mathcal{M}_?^A$ and $\mathcal{M}_{tuple}$. Also, the join of two $\mathcal{M}_?$ relations, each with two maybe-tuples, with the same value on the joining attribute, is not representable in any of $\mathcal{M}_{\oplus\equiv}$, $\mathcal{M}_2$, and $\mathcal{M}_2^A$.
   Since a natural join can be rewritten as a cross-product followed by selection and $\mathcal{M}_?$ is closed under selection, the transition for cross-product is also to $\mathcal{M}_{prop}^A$.
4. Transition from $\mathcal{M}_{\oplus\equiv}$ to $\mathcal{M}_{prop}^A$ on difference or intersection. The difference $R_2 - R_1$ is not expressible in any model in the path to $\mathcal{M}_{prop}^A$, where R_1 and R_2 are as follows.
   R_1 has tuples:
   `[a]` *certain* (certain tuple obtained by xor-ing with itself)
   `[a]` *certain*
   R_2 has six tuples: t_1`:[a]`, t_2`:[a]`, t_3`:[a]`, t_4`:[x]`, t_5`:[y]`, and t_6`:[z]`. Constraints on R_2: $t_1 \equiv t_4$ and $t_2 \equiv t_5$ and $t_3 \equiv t_6$.
5. Transition from $\mathcal{M}_{\oplus\equiv}$ to $\mathcal{M}_{prop}^A$ on intersection. Using an idea similar to that of the join from Example 3 above, we construct R_1 and R_2 as follows so that the intersection of R_1 and R_2 is not expressible in any model in the path to $\mathcal{M}_{prop}^A$. The tuples of R_1 and R_2 are as follows. R_1 has six tuples: `[y]`, `[w]`, `[x]`, `[z]`, `[1]`, and `[1]`, where `[y]` and `[w]` are certain tuples and one of the `[1]` tuples is equivalent to `[x]` while the other

is equivalent to [z]. Similarly R_2 has six tuples: [x], [z], [y], [w], [1], and [1], where [x] and [z] are certain tuples and one of the [1] tuples is equivalent to [y] while the other is equivalent to [w].

□

B Uniqueness and Equivalence Testing Proofs

B.1 Proof of Uniqueness of $\mathcal{M}^A_?$ and $\mathcal{M}_{tuple}$

Theorem 9: Every set of possible instances representable in $\mathcal{M}_{tuple}$, $\mathcal{M}^A_?$, $\mathcal{M}^A$, or $\mathcal{M}_?$ has a unique representation. □

Proof We consider each of the models separately.

$\mathcal{M}_{tuple}$: Consider two relations R_1 and R_2 in $\mathcal{M}_{tuple}$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 be the number of tuple-ors in R_1 containing 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 $\mathcal{I}$. (If not, we directly have an instance of R_1 that is not an instance of R_2.) 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 tuple-or in T_1 is set to t, then the corresponding tuple-or that is equal to it in S_1 must also be set to t.

At the end of this iteration, S_2 contains at least one tuple-or from which no tuple has been chosen (specifically, r'_j does not contain t). So we next look at such an 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 T_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.

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

B.2 Proof of Non-Uniqueness of $\mathcal{M}_{\oplus\equiv}$, $\mathcal{M}_2$, $\mathcal{M}^A_2$ and $\mathcal{M}^A_{prop}$

Theorem 10: There exist sets of possible instances representable in $\mathcal{M}_{\oplus\equiv}$, $\mathcal{M}_2$, $\mathcal{M}^A_2$, and $\mathcal{M}^A_{prop}$ that do not have unique representations. □

Proof We prove non-uniqueness of the models by showing that answer to one of the following questions for each of the schemes is yes:

Q1: Can two relations under a model have different sets of A-tuples but the same set of instances?

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

Table 4 below summarizes the answers to the two questions for $\mathcal{M}_2$ and $\mathcal{M}^A_{prop}$. We consider both cases where the relations contain duplicates and where they don't. The superscript in the table points to the proof for that entry.

a: 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

	Are there duplicates?	$\mathcal{M}_{\oplus\equiv}$	$\mathcal{M}_2$	$\mathcal{M}_2^A$	$\mathcal{M}_{prop}^A$
Q1	No	No^a	Yes^b	Yes^b	Yes^b
Q1	Yes	No^a	Yes^b	Yes^b	Yes^b
Q2	No	No^c	No^c	Yes^e	Yes^e
Q2	Yes	Yes^d	Yes^d	Yes^d	Yes^d

Table 4 Non-Uniqueness Questions

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.

b: 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 $I(R_1) = I(R_2)$.

c: 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 $I(R_1) \neq I(R_2)$.

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 $f_1 = (t_1 \oplus t_2) \wedge (t_2 \oplus t_3) \wedge (t_3 \oplus t_4)$ and $f_2 = (t_1 \oplus t_2) \wedge (t_3 \oplus t_4)$. f_1 and f_2 are not equivalent, but $I(R_1) = I(R_2)$.

e: Consider tuple sets $T_1 = T_2$ with three A-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 $R_2 = (T_2, f_2)$.

□

C Minimization Proofs

C.1 Tuple Versus Constraint Minimality

Theorem 14: For $\mathcal{M}_{prop}^A$ relations with no duplicates in the representation, minimizing the number of A-tuples required to represent a set of possible instances also minimizes the size of the minimal constraint. □

Proof Consider an $\mathcal{M}_{prop}^A$ relation $R = (T, f)$ that represents $I(R)$ using the fewest possible number of A-tuples. Consider another $\mathcal{M}_{prop}^A$ relation $R' = (T', f')$ that also represents $I(R)$ but with $|T'| > |T|$. If f and f' are minimal constraints for T and T' respectively, we show that $size(f) \leq size(f')$.

To show this, starting from R', we construct an equivalent relation $R'' = (T, f'')$ with $size(f'') \leq size(f')$. Since f is the minimal size of constraints required to represent $I(R)$ using T, $size(f) \leq size(f'') \leq size(f')$.

To begin with, we first convert every A-tuple in T and T' to a tuple-or and view f and f' as constraints over these tuple-ors. We will here onwards operate in this world of $\mathcal{M}_{tuple}$ relations with constraints and provide a translation from the $\mathcal{M}_{tuple}$ representation of R' to the $\mathcal{M}_{tuple}$ representation of R''. Now note that if T' contains no redundant or duplicate tuple-ors, then the tuple-ors in T can be obtained from T' through a series of *splits* and *combines*: a single tuple-or may be split to give two tuple-ors or two tuple-ors may be combined to give a single tuple-or.

In order to construct f'', we view the translation of T' to T as follows: all necessary splits of tuple-ors in T' are first performed, and then the resulting tuple-ors are combined to obtain T. Since T has fewer tuple-ors than T', the number of combines will be more than the number of splits. Further, every split of a tuple-or necessarily gives rise to a combine (otherwise T cannot be minimal). Each time either a split or combine is performed, we modify f' so that the new relation with the modified tuple-ors and modified formula remains equivalent to R'. We now describe how the formula is modified during splits and combines. It can be seen that more clauses are taken away during combines than are added during splits. Hence we ultimately get a formula with size at most that of f'.

Splits: Consider relations $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$ where $T_2 = (T_1 - t) \cup t_1 \cup t_2$ and t_1 and t_2 are obtained by splitting tuple-or $t \in T_1$. We would like to modify f_1 to obtain f_2 such that $R_2 \equiv R_1$. We start off by adding the constraint $\neg t_1 \vee \neg t_2$ to f_1. Then every occurrence of t in f_1 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_1 and R_2 while adding $k+1$ clauses, where k is the number of clauses containing $\neg t$.

Combinations: Consider relations $R_1 = (T_1, f_1)$ and $R_2 = (T_2, f_2)$ where $T_2 = (T_1 - t_1 - t_2) \cup t$ and t is obtained by combining t_1 and t_2. We modify f_1 to obtain f_2 such that $R_2 \equiv R_1$. 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_1 and R_2 while reducing the number of clauses in the formula by l where l is the number of clauses containing $\neg t_1$. □

C.2 Incremental Minimality of Operations

Theorem 15: Given $\mathcal{M}^A_{prop}$ relations with minimal number of A-tuples, there exist polynomial time algorithms to compute their union or cross-product so that the resulting $\mathcal{M}^A_{prop}$ relation is minimal with respect to the number of A-tuples. □

Proof We show here that the algorithms presented in [25] maintain incremental minimality under union and cross product. We just need to show that no A-tuples in the result of union or cross product can be merged. That is, if we start with two $\mathcal{M}^A_{prop}$ relations R_1 and R_2 with the minimum number of A-tuples, the algorithms in [25] give $S = R_1 \cup R_2$ and $T = R_1 \times R_2$, where S and T have minimal number of A-tuples.

Union: The result $S = R_1 \cup R_2$ has the union of the A-tuples in R_1 and R_2. Now note that no instances of A-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 A-tuple in them appears in some instance. Finally, if we can merge A-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 an A-tuple for every combination of A-tuples from R_1 and R_2. Here again, merging any two instances of A-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 A-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. □

Proposition 1: Given $\mathcal{M}^A_{prop}$ relations with minimal number of A-tuples, the most natural algorithms to perform selection, projection, or natural join on them may fail to maintain incremental minimality. □

Proof We show that after performing the most natural algorithms for these operations, it may become necessary to merge two A-tuples in the result. Consider an $\mathcal{M}^A_{prop}$ relation R with one attribute X having two possible instances:

```
I1: [1], [3]
I2: [2]
```

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

```
I1': [3]
I2': [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 A-tuples does not give the minimal result.

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

```
I1:[p,1]
I2:[q,2]
```

The minimal $\mathcal{M}^A_{prop}$ relation for S would need two A-tuples t_1:[p, 1] and t_3:[q, 2]. However, on performing $S' = \Pi_X(S)$, the result is two possible instances `I1':[p]` and `I2':[q]`. In this result, t'_1 and t'_2 can be merged to give a single A-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 single possible instance [b]. The result of $R \bowtie T$ is:

```
I1': [b,3]
I2': [b,2]
```

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

Theorem 16: Given an $\mathcal{M}^A_{prop}$ relation with minimal number of A-tuples, maintaining minimal number of A-tuples after performing a selection is NP-hard.

Proof Finally, we show that in general, maintaining minimal number of A-tuples while performing operations is a hard problem. This can be seen by a direct reduction from the NP-complete bi-clique cover problem. Given an instance of the bi-clique cover problem, we first create a table with two attributes. The table contains one A-tuple with all the

left hand side nodes forming an or-set for the first attribute and right hand side nodes forming an or-set for the second attribute. The table thus represents a complete bipartite graph on the nodes of the input bi-clique cover problem. Note that this is the minimal way to represent such a complete graph in a $\mathcal{M}^A_{prop}$ relation. A selection operation is then performed on this table to select only the edges in the input of the bi-clique cover problem. The minimal set of A-tuples required to represent this output would correspond to the solution to the bi-clique cover problem. □

D Approximate Representation Proofs

Theorem 17: Let P be a set of possible instances. Algorithm 1, $\mathcal{M}_{tuple}$-REP, outlines a polynomial (in the size of P) algorithm that returns an exact representation of P in $\mathcal{M}_{tuple}$ whenever one exists. □

Proof First note that Algorithm $\mathcal{M}_{tuple}$-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 $\mathcal{M}_{tuple}$ 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 $\mathcal{M}_{tuple}$ representation of P; in other words, we show that every step in the algorithm adds tuples to tuple-ors of the partially generated R in the only way possible. When we look for the addition of t_i in step 5, we look at its co-occurrences 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 $\mathcal{M}_{tuple}$ 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 $\mathcal{M}_{tuple}$ representation of P. □

Theorem 2: There exists set of instances P for which there is no constant-factor $\mathcal{M}_{tuple}$-approximation R under the Jaccard measure of similarity between P and $I(R)$. □

Proof Consider n instances $\{P_1, P_2, \ldots, P_n\}$ for a relation with two attributes. Let the ith instance have two tuples: t^1_i:[i, 1] and t^2_i:[i, 2]. We show that this set of n possible instances 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 $\mathcal{M}_{tuple}$ representation of R. Further, whenever tuples for a particular instances, say P_j are chosen, the rest of the tuples are not present in the instance. So either the tuple-ors for all $[k, 1]$ and $[k, 2]$, $j \neq k$ have '?', or are in the same tuple-ors as $[j, 1]$ and $[j, 2]$ respectively. It can be seen that under these conditions, for m to be smallest we have just two tuple-ors 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. □

Theorem 19:

1. Finding the best $\mathcal{M}_{tuple}$ approximation for an arbitrary set of possible instances P is NP-hard.
2. $\mathcal{M}_{tuple}$ approximation can be reduced to the minimum graph coloring problem, which admits a 5/7-differential approximation. □

Proof 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 A-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 A-tuple in R. Therefore, each coloring gives an approximation with the number of A-tuples being the number of colors, and vice-versa. The best approximation gives the minimum number of A-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 [49]. 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, with the number of A-tuples in the approximation equal to the number of colors in the coloring: tuples corresponding to vertices with the same color are assigned to the same A-tuple. □