Materialized Views in Probabilistic Databases
For Information Exchange and Query Optimization
(Full Version)
University of Washington Technical Report
#TR2007-03-02

Christopher Ré
University of Washington
chrisre@cs.washington.edu

Dan Suciu
University of Washington
suciu@cs.washington.edu

Abstract

Views over probabilistic data contain correlations between tuples, and the current approach is to capture these correlations using explicit lineage*. In this paper we propose an alternative approach to materializing probabilistic views, by giving conditions under which a view can be represented by a block-independent disjoint (BID) table. Not all views can be represented as BID tables and so we propose a novel partial representation that can represent all views but may not define a unique probability distribution. We then give conditions on when a query's value on a partial representation will be uniquely defined. We apply our theory to two applications: query processing using views and information exchange using views. In query processing on probabilistic data, we can ignore the lineage and use materialized views to more efficiently answer queries. By contrast, if the view has explicit lineage, the query evaluation must reprocess the lineage to compute the query resulting in dramatically slower execution. The second application is information exchange when we do not wish to disclose the entire lineage, which otherwise may result in shipping the entire database. The paper contains several theoretical results that completely solve the problem of deciding whether a conjunctive view can be represented as a BID and whether a query on a partial representation is uniquely determined. We validate our approach experimentally showing that representable views exist in real and synthetic workloads and show over three magnitudes of improvement in query processing versus a lineage based approach.*

1 Introduction

A view over probabilistic data contains correlations between tuples which make views expensive to represent. Currently, materialized views are based on a complete approach (e.g. [20, 37, 38]) which can represent any conjunctive view but requires storing auxiliary information (e.g. lineage [20]). Lineage (or provenance [18]) has many advantages, especially for scientific applications [7]. However, there are important drawbacks with lineage for information exchange and query optimization using views. In query optimization using views, to compute probabilities correctly we must determine how tuples are correlated. Inspecting the lineage of tuples in a view at query execution time is expensive since the lineage of a single tuple can be as large as the database. In contrast, if we can prove that all tuples in the view are independent, we do not need to examine the lineage at execution time. Further, we can optimize our query to use much more efficient query processing techniques (e.g. safe plans [19, 36, 37]). In some applications of exchanging views of probabilistic data, using lineage to describe the data semantics is not an option. For example, we may not want to disclose any lineage (e.g. B2B applications) or the size of the lineage may be prohibitive. We study if a view can be understood without the full lineage.

In this paper, we consider the block-independent disjoint (**BID**) formalism. Informally, each relation is partitioned into blocks that are disjoint but, across blocks, tuples are independent. The **BID** formalism captures many representations previously discussed in the literature (e.g. tuple independent [19, 31], p-or tables [26], ?-tables and x-relations [20]). We study the **view representation problem** (Problem 1) which is to decide: Given a conjunctive view V, is the output of V **representable** as a **BID** table? Not all views can be represented as **BID** tables and so we propose a **partial representation** which can represent any view but may not specify how all tuples in a view correlate. Because all correlations among tuples are not defined, many probability distributions can agree with a single partially representable view which can cause a query's value to fail to be uniquely defined. This motivates us to study the **partial view answering problem** (Problem 2) which is: Given a query Q using partially representable views, is Q's value uniquely defined?

We validate our solutions using data given to us by iLike.com [13], a service provided by the music site GarageBand [12] which provides users with recommendations for music and friends. iLike has three characteristics which make it a good candidate for materialized views. First, the data are uncertain because the recommendations are based on similarity functions. Second, the database is large (e.g. tens of gigabytes) and backs an interactive website with many users; hence, query performance is important. Third, the database hosts more than one service, implying that there are integration issues. Probabilistic materialized views are a tool that addresses all three of these issues. Interestingly, materialized views are present in iLike. Since there is no support for uncertain views, the materialized views are constructed in an *ad-hoc* manner and require care to use correctly. Our experiments show that 80% of the materialized views in iLike are representable by BID tables and more than 98.5% of their workload could benefit from our optimizations. We also experiment with synthetic workloads and find that the majority of queries can benefit from our techniques. Using the techniques described in this paper, we are able

Chef	Restaurant	**P**	
TD	D. Lounge	0.9	(w_1)
TD	P. Kitchen	0.7	(w_2)
MS	C. Bistro	0.8	(w_3)

W(Chef,Restaurant;;**P**) (WorksAt)

Restaurant	Dish
D. Lounge	Crab Cakes
P. Kitchen	Crab Cakes
P. Kitchen	Lamb
C. Bistro	Fish

S(Restaurant,Dish) (Serves)

Chef	Dish	Rating	**P**	
TD	Crab Cakes	High	0.8	(r_{11})
		Med	0.1	(r_{12})
		Low	0.1	(r_{13})
TD	Lamb	High	0.3	(r_{21})
		Low	0.7	(r_{22})
MS	Fish	High	0.6	(r_{31})
		Low	0.3	(r_{32})

R(Chef,Dish;Rating;**P**) (Rated)

Figure 1. Sample Restaurant Data in Alice's Database. In `WorksAt` each tuple is *independent*. There is no uncertainty about `Serves`. In `Rated`, each (Chef,Dish) pair has one true rating. The syntax is described in def. 2.1.

to achieve speed ups of more than three orders of magnitude on large probabilistic data sets (e.g. TPC 1G with additional probabilities). A summary of our contributions:

- We solve the *view representation problem* for the case of conjunctive views over BID databases (def. 2.2) which capture several representations in the literature [10, 19, 37, 39, 41]. Specifically:
  - We give a sound and complete algorithm for finding a representation when one exists and prove that the decision is Π_2^p Complete in general. (Sec. 4.1)
  - We give a polynomial time approximation for the *view representation problem* which is sound, but not complete. We show that it is complete for all queries without self-joins. (Sec. 4.2)
- We solve the *probabilistic materialized view answering problem* for the case of conjunctive queries and conjunctive views.
  - We propose a partial representation system to handle views that are not representable. (Sec. 5.1)
  - We give a sound and complete algorithm to decide if Q correctly uses a partially represented view and prove this decision is Π_2^p-Complete. (Sec. 5.2)
  - We give a polynomial time approximation for the *probabilistic materialized view answering problem* that is sound, but not complete. We show that it is complete for queries without repeated probabilistic views. (Sec. 5.3)
- We validate our techniques experimentally (Sec. 7), showing that representable views exist in practice, that our techniques yield several orders magnitude improvement and that our practical algorithms are almost always complete.

2 Problem Definition

Our running example is a scenario in which a user, Alice, maintains a restaurant database that is extracted from web data and she wishes to send data to Bob. Alice's data are uncertain because they are the result of *information extraction* [34, 27, 29] and *sentiment analysis* [23]. A natural way for her to send data to Bob is to write a view, materialize its result and send the result to Bob. We begin with some preliminaries about probabilistic databases based on the *possible worlds model* [5] and will discuss the data in Fig. 1.

2.1 Representation Formalism

Definition 2.1 (Syntax for Representations)**.** *A **block independent disjoint table description** (BID table description) is a relational schema with the attributes partitioned into three classes separated by semicolons:*

$$\mathtt{R}(K_1, \ldots, K_n; A_1, \ldots, A_m; \mathbf{P})$$

*where $K = \{K_1, \ldots, K_n\}$ is called the **possible worlds key**, $A = \{A_1, \ldots, A_n\}$ is called the **value attribute set** and $\mathbf{P}$ is a single distinguished attribute called the **probability attribute**; its type is a value in the half-open interval* $(0, 1]$.

Semantics. Representations contain probability attributes ($\mathbf{P}$) but the worlds they represent do not. For example, a BID table description $\mathtt{R}(K; A; \mathbf{P})$ with distinguished attribute $\mathbf{P}$ corresponds to a **BID symbol** $\mathtt{R}^p(K; A)$ with no attribute $\mathbf{P}$[1]. A representation **yields** a distribution on *possible worlds* over *BID symbols*.

Definition 2.2. *Given an instance of a BID table description $\mathtt{R}(K; A; \mathbf{P}) = \{t_1^r, \ldots, t_n^r\}$, a **possible world** is a subset of tuples, I, without the attribute $\mathbf{P}$ that satisfies the key constraint $K \to A$. Let $\mathbf{AK}(I) = \{k \mid \exists i\ t_i^r[K] = k \wedge \forall t \in I\ t_i^r[K] \neq t[K]\}$. The probability of possible world I, denoted $\mu(I)$, is defined as:*

$$\mu(I) = \overbrace{\Big(\prod_{i=1:t_i^r[KA]\in I}^{n} t_i^r[\mathbf{P}]\Big)}^{\textit{Present Tuples}} \underbrace{\Big(\prod_{k\in\mathbf{AK}(I)} \Big(1 - \sum_{j=1:t_j^r[K]=k}^{n} t_j^r[\mathbf{P}]\Big)\Big)}_{\textit{Absent Tuples}}$$

Informally, Def. 2.2 says three things: The marginal probability of each tuple, $\mu(t)$, satisfies $\mu(t) = t^r[\mathbf{P}]$, any set of tuples, $t_1, \ldots, t_m$ with distinct keys are independent (i.e. $\mu(t_1 \wedge \cdots \wedge t_m) = \prod_{i=1}^m \mu(t_i)$) and any two distinct tuples s, t that share the same *possible worlds key* are disjoint, $\mu(s \wedge t) = 0$.

We shall refer to a schema that contains both deterministic and BID table descriptions as a **BID schema** and an instance of a BID schema as a **BID instance**.

[1] To emphasize this distinction, we will use a typewriter faced symbol with a superscripted p (e.g. $\mathtt{R}^p(K; A)$) to denote the BID symbol corresponding to the BID table description $\mathtt{R}(K; A; \mathbf{P})$. For deterministic tables, this distinction is immaterial.

Definition 2.3. *A possible world I for a BID instance $\mathtt{R}_1, \ldots \mathtt{R}_n$ consists of a n possible worlds $I_1, \ldots, I_n$ with measures $\mu_1, \ldots, \mu_n$, one for each $\mathtt{R}_i$, and its probability is defined as $\mu(I) = \prod_{i=1}^{n} \mu_i(I_i)$.*

Definition 2.4 (Conjunctive View)**.** *A view is a conjunctive query of the form:*

$$V^p(\vec{h}) \coloneq g_1, \ldots, g_n \tag{1}$$

*where each g_i is a **subgoal**, that is either a BID symbol (e.g. $\mathtt{R}^p$) in which case we say g_i is probabilistic or a deterministic table (e.g. $\mathtt{S}$). The set of variables in V^p is denoted $\mathbf{var}(V^p)$ and the set of head variables $\vec{h} \subseteq \mathbf{var}(V^p)$. The natural schema associated with V^p is the list of head variables denoted by H.*

Definition 2.5 (View Semantics)**.** *Given a view $V^p(H)$, the marginal probability of a tuple t in the output of V^p is denoted $\mu(V^p(t))$ and satisfies:*

$$\mu(V^p(t)) = \sum_{I : I \models V^p(t)} \mu(I)$$

We will denote by $V^p(\mathfrak{I})$, the output of the view on the representation which is the set $\{(t_1, p_1), \ldots, (t_m, p_m)\}$ such that for each $i \in \{1, \ldots m\}$, $\mu(V^p(t_i)) = p_i > 0$. p_i is the marginal probability that $V^p(t_i)$ is satisfied.

2.2 Running Example

Sample data is shown in Fig. 1 for Alice's schema that contains three relations described in BID syntax: `W` (`WorksAt`), `S` (`Serves`) and `R` (`Rating`). The relation `W` records chefs, who may work at multiple restaurants in multiple cities. The tuples of `W` are extracted from text and so are uncertain. For example, ('TD', 'D. Lounge') (w_1) in `W` signifies that we extracted that 'TD' works at 'D.Lounge' with probability 0.9. Our syntax tells us that all tuples are independent because they all have different possible worlds keys. The relation `R` records the rating of a chef's dish (e.g. 'High' or 'Low'). Each (Chef,Dish) pair has only one true rating. Thus, r_{11} and r_{13} are *disjoint* because they rate the pair ('TD', 'Crab Cakes') as both 'High' and 'Low'. Because distinct (Chef,Dish) pair ratings are extracted independently, they are associated to ratings independently.

Semantics. In `W`, there are 2^3 *possible worlds*. For example, the probability of the singleton subset {('TD', 'D. Lounge')} is $0.9 * (1 - 0.7) * (1 - 0.8) = 0.054$. This representation is called a p-?-table [26], ?-table [20] or tuple independent [19]. The BID table `R` yields $3 * 2 * 3 = 18$ possible worlds since each (Chef,Dish) pair is associated with at most one rating. When the probabilities shown sum to 1, there is at least one rating for each pair. For example, the probability of the world $\{r_{11}, r_{21}, r_{31}\}$ is $0.8 * 0.3 * 0.6 = 0.144$. `R` is a slight generalization of p-or-set-table [26] or x-table [20].

2.2.1 Representable Views

Alice wants to ship her data to Bob, who wants a view with all chefs and restaurant pairs that make a highly rated dish. Alice obliges by computing and sending the following view:

$$V_1^p(c, r) \coloneq \mathtt{W}^p(c, r), \mathtt{S}(r, d), \mathtt{R}^p(c, d; \text{'High'}) \tag{2}$$

In the following example data, we calculate the probability of a tuple appearing in the output both numerically in the **P** column and symbolically in terms of other tuple probabilities of Fig. 1. We calculate the probabilities symbolically only for exposition; the output of a query or view is a set of tuples matching the head with associated probability scores.

Example 2.1 (Output of V_1^p from Eq. (2))**.**

	C	*R*	**P**	*(Symbolic Probability)*
t_1^o	*TD*	*D. Lounge*	0.72	$w_1 r_{11}$
t_2^o	*TD*	*P.Kitchen*	0.602	$w_2(1-(1-r_{11})(1-r_{21}))$
t_3^o	*MS*	*C.Bistro*	0.32	$w_3 r_{31}$

The output tuples, t_1^o and t_2^o, are *not* independent because both depend on r_{11}. This lack of independence is problematic for Bob because a BID instance cannot represent this type of correlation. Hence we say V_1^p is not a **representable view**. For Bob to understand the data, Alice must ship the lineage of each tuple. For example, it would be sufficient to ship the symbolic probability polynomials in Ex. 2.1.

Consider a second view where we can be much smarter about the amount of information necessary to understand the view. In V_2^p, Bob wants to know which working chefs make and serve a highly rated dish:

$$V_2^p(c) \coloneq \mathtt{W}^p(c, r), \mathtt{S}(r, d), \mathtt{R}^p(c, d; \text{'High'}) \tag{3}$$

Example 2.2 (Output of V_2^p from Eq. (3))**.**

c	**P**	*(Symbolic Probability)*
TD	0.818	$r_{11}(1-(1-w_1)(1-w_2)) + (1-r_{11})(w_2 r_{21})$
MS	0.48	$w_3 r_{31}$

Importantly, Bob can understand the data in Ex. 2.2 with no auxiliary information because the set of events contributing to 'TD' and those contributing to 'MS' are *independent*. It can be shown that over any instance, all tuples contributing to distinct c values are independent. Thus, V_1^p is a **representable view**. This motivates us to understand the following fundamental question:

Problem 1 (View Representabilty)**.** *Given a view V^p, can the output of V^p be represented as a BID table?*

It is interesting to note that efficiency and representability are distinct concepts: Computing the output of V_1^p can be done in polynomial time but its result is not representable. On the other hand, computing V_2^p can be shown to be #P-Hard [19, 36], but V_2^p is *representable*. To process V_2^p, we must use Monte Carlo procedures (e.g. [37]), which are orders of magnitude more expensive than traditional SQL processing. We do not discuss evaluation further because our focus is not on efficiently evaluating views, but on representing the output of views.

2.2.2 Partially Representable Views

To optimize and share a larger class of views, we would like to use the results of views that are not *representable*. The output of a non-representable view still has meaning: It contains the marginal probabilities that each tuple appears in the view but may not describe how all tuples correlate. Consider the output of V_1^p in Ex. 2.1: Tuples that agree on c are correlated, but tuples with different c values are independent. To capture this, we define a **partially representable view** (def. 5.3) which has schema $V^p(K_I; D; A)$. The intuition is that tuples that differ on K_I are guaranteed to be independent, and distinct tuples which agree on $K_I D$ are guaranteed to be disjoint. Tuples that agree on K_I but not on D may be correlated.

Example 2.3. *Recall the view V_1^p in Eq. (2), whose natural schema is $V^p(C, R)$. It is partially representable as $V_1^p(C; R; \emptyset)$ (i.e. $K_I = \{C\}$, $D = \{R\}$ and $A = \emptyset$). Tuples that differ on C are independent, but those that agree on C but differ on R may be correlated in unknown ways. Thus, the materialized view does have some meaning for Bob, but does not contain sufficient information to completely determine a probability distribution on its possible worlds.*

Trivially, any view $V^p(H)$ can be partially represented by letting $K_I = A = \emptyset$ and $D = H$. Thus the interesting question is: What is the complexity to decide, for a given K_I, D, A, if a view V^p is *partially representable*?

2.2.3 Using Views to Answer Queries

In an information exchange setting, we do not have access to the base data and so for partially representable views may not know how all tuples are correlated. Thus, to answer a query Q, we need to ensure that the value of Q does not depend on correlations that are not captured by the partial representation. To illustrate, we attempt to use the output of V_1^p, a partially representable view, to answer two queries.

Example 2.4. *Suppose Bob has a local relation, $\mathrm{L}^p(d; r)$, where d is a possible worlds key for* L*. Bob wants to answer the following queries:*

$$Q_u(d) \coloneq \mathrm{L}^p(d; r), V_1^p(c, r) \text{ and } Q_n(c) \coloneq V_1^p(c, r)$$

Since the materialized view V_1^p does not uniquely determine a probability distribution on its possible worlds, it is not immediately clear that Bob can answer these queries without access to Alice's database and to V_1^p's definition. However, the query

Q_u is uniquely defined. For any fixed d_0, the set of r_i values such that $\mathtt{L}^p(d_0; r_i)$ partition the possible worlds:

$$\mu(Q_u(d_0)) = \sum_{r_i : I \models \mathtt{L}^p(d_0; r_i)} \mu(\mathtt{L}^p(d_0; r_i) \wedge \exists c\ V_1^p(c, r_i))$$

The partial representation $V_1^p(C; R; \emptyset)$, tells us that distinct values for c in V_1^p are independent. Thus, each term in the summation is uniquely defined implying Q_u is uniquely defined. In contrast, Q_n is not uniquely defined because Q_n('TD') is true when either of t_1^o or t_2^o are present; our partial representation does not capture the correlation of the pair (t_1^o, t_2^o).

This example motivates the following problem:

Problem 2 (View Answering)**.** *Let V^p be a partially representable view. Given a query Q using V^p, is the value of Q uniquely defined?*

If V^p is representable, then this problem is trivial because V^p uniquely defines a probability distribution.

3 Preliminaries

We define notations we need in the remainder of the paper.

Definition 3.1. *A **valuation** v for a view V^p is a mapping from variables and constants in V^p to constants which is identity on constants ([2]). Given a valuation we let* $\mathbf{im}(v)$ *denote the image of v.*

We define a subset of valuations called **disjoint aware valuations** which associate each possible worlds key value to exactly one attribute value. Recall the definition of a view in Eq. (1); if g_i is probabilistic, we let $\vec{k}_i$ (resp. $\vec{a}_i$) denote the list of variables or constants in the possible world key attribute positions (resp. value attribute positions)[2]. For a subgoal g_i, $\mathbf{pred}(g_i)$ denotes the predicate associated to g_i.

Definition 3.2 (Disjoint Aware Valuation)**.** *A valuation, v, for a conjunctive view is **disjoint aware** if*

$$\forall i, j\ \mathbf{pred}(g_i) = \mathbf{pred}(g_j)\ \textit{and}\ v(\vec{k}_i) = v(\vec{k}_j) \implies v(\vec{a}_i) = v(\vec{a}_j)$$

Example 3.1 (Disjoint Aware Valuations)**.**

$$V_4^p() \text{ :- } \mathtt{R}^p(x, y; \textit{'High'}), \mathtt{R}^p(x, z; \textit{'Low'}) \tag{4}$$

Any valuation v such that $v(y) \neq v(z)$ is disjoint aware but if $v(y) = v(z)$, it is not.

[2]We assume deterministic tables do not have keys.

We need to consider pairs of valuations that do not use disjoint events, called **compatible valuations**.

Definition 3.3. *A pair of disjoint aware valuations (v, w) for a view V^p is called **compatible** if the pair satisfies:*

$$\forall i, j \; \mathbf{pred}(g_i) = \mathbf{pred}(g_j) \text{ and } v(\vec{k}_i) = w(\vec{k}_j) \implies v(\vec{a}_i) = w(\vec{a}_j)$$

3.1 Generalizing Functional Dependencies

There are three distinct notions of functional dependencies over variables in the body of a view that we need to consider: standard functional dependencies, denoted $V^p \models \vec{a} \to \vec{b}$, representation functional dependencies, denoted $V^p \models \vec{a} \to_r \vec{b}$, and possible worlds dependencies, denoted $V^p \models \vec{a} \to_p \vec{b}$.

Definition 3.4. *For a pair of valuations (v, w) for V^p, let $\dagger(v, w)$ denote the property:*

$$v(\vec{a}) = w(\vec{a}) \implies v(\vec{b}) = w(\vec{b})$$

Then we write:

- $V^p \models \vec{a} \to \vec{b}$ *if* $\dagger(v, w)$ *for any valuations.*
- $V^p \models \vec{a} \to_r \vec{b}$ *if* $\dagger(v, w)$ *for any disjoint aware valuations.*
- $V^p \models \vec{a} \to_p \vec{b}$ *if* $\dagger(v, w)$ *for any compatible valuations.*

$V^p \models \vec{a} \to \vec{b}$ if and only if $\vec{b} \subseteq \vec{a}$ as sets of variables[3]. To see how these definitions relate, it is immediate that:

$$V^p \models \vec{a} \to \vec{b} \implies V^p \models \vec{a} \to_r \vec{b} \implies V^p \models \vec{a} \to_p \vec{b}$$

We show by example that both reverse implications fail:

Example 3.2. *Consider a BID table* $\mathtt{U}(K; A; \mathbf{P})$ *and a deterministic binary table* $\mathtt{D}(B, C)$.

$$V_5^p(x, y, z) \coloneq \mathtt{U}^p(x; y), \mathtt{U}^p(x; z), \mathtt{D}(y, z) \tag{5}$$

Here $V_5^p \not\models xy \to z$ but $V_5^p \models xy \to_r z$ since, any disjoint aware valuation for V_5 must satisfy $v(y) = v(z)$.

$$V_6^p() \coloneq \mathtt{U}^p(x; y) \tag{6}$$

[3]This fails when we add dependencies in Sec. 4.3.

Algorithm 1 Decision Procedure for $V^p \models \vec{a} \rightarrow_p \vec{b}$

Input: $V^p(\vec{h}) :- g_1, \ldots, g_m$ and
$\vec{a}, \vec{b} \subseteq \mathbf{var}(V^p) \cup \mathbf{const}(V^p)$

Output: `'Yes'` iff $V^p \models \vec{a} \rightarrow_p \vec{b}$ else `'No'`

1: η is a function that is identity on $\vec{a}$ and $\mathbf{const}(V^p)$ and maps all other variables to distinct fresh variables.
2: $VV(\vec{b'}, \eta(\vec{b})) :- g_1, \ldots, g_m, \eta(g_1), \ldots, \eta(g_m)$
3: Let $VV'(\vec{b}, \vec{b'}) = \mathbf{DJA}(\ VV(\vec{b}, \eta(\vec{b}))\ )$
4: **if** Chase Succeeds **and** $\vec{b} \neq \vec{b'}$ **then**
5: **return** `'No'` (* $V^p \not\models \vec{a} \rightarrow_p \vec{b}$ *)
6: **else**
7: **return** `'Yes'` (* $V^p \models \vec{a} \rightarrow_p \vec{b}$ *)

In this case, $V_6^p \not\models x \rightarrow_r y$ *but* $V_6^p \models x \rightarrow_p y$.

3.2 A Chase Procedure

The important property we need of the chase is that it constructs a *universal plan* ([22]), which equates exactly those variables that must be equated by any disjoint aware valuation.

Proposition 3.1 (Chase [22])**.** *There is a polynomial time procedure that takes as input a conjunctive view* V^p *and produces as output a view* $\mathbf{DJA}(V^p)$ *and surjective homomorphism* θ *from* V^p *to* $\mathbf{DJA}(V^p)$ *such that* $\mathbf{DJA}(V^p)$ *is equivalent to* V^p *over all possible worlds and the identity homomorphism on* $\mathbf{DJA}(V^p)$ *is disjoint aware. The Chase may fail if no such view exists.*

Example 3.3. *Consider* V_5^p *from Ex. 3.2, then* [4]

$$\mathbf{DJA}(V_5^p) = W(x, y, y) :- \mathtt{U}^p(x; y), \mathtt{D}(y, y)$$

The chase equates y and z because both have possible worlds key x. In contrast, the chase will fail on

$$V(c, r) :- \mathtt{R}^p(c, r; \text{'High'}), \mathtt{R}^p(c, r; \text{'Low'})$$

The chase cannot unify the constants 'High' and 'Low'.

We show how to decide $V^p \models \vec{a} \rightarrow_r \vec{b}$ and $V^p \models \vec{a} \rightarrow_p \vec{b}$.

Proposition 3.2. *If* $\mathbf{DJA}(V^p)$ *exists, let* θ *be the chase homomorphism, then the following holds:*

$$\mathbf{DJA}(V^p) \models \theta(\vec{a}) \rightarrow \theta(\vec{b}) \iff V \models \vec{a} \rightarrow_r \vec{b}$$

[4] We are *not* doing minimization, only removing duplicates.

We use Prop. 3.2 to decide $V^p \models \vec{a} \rightarrow_r \vec{b}$. For example, $V_5^p \models xy \rightarrow_r z$ Eq. (5) because, the chase homomorphism θ is given by $\theta(x, y, z) = (x, y, y)$. Thus, $\theta(z) \subseteq \theta(xy)$.

We use Alg. 1 to efficiently decide $V^p \models \vec{a} \rightarrow_p \vec{b}$.

Proposition 3.3. *Algorithm 1 is a polynomial time sound and complete algorithm to decide if* $V^p \models \vec{a} \rightarrow_p \vec{b}$.

Sketch. Observe that if the chase fails there must be some contradiction in the view so there are no disjoint aware valuations for V^p implying $V^p \models \vec{a} \rightarrow_p \vec{b}$ trivially holds. If the chase succeeds, then by Prop. 3.1, VV' has the property that the identity valuation is disjoint aware. Thus, if our test outputs `'No'`, there is a disjoint aware valuation vv for VV^p such that $vv(\vec{a}) = vv(\vec{a})$ but $vv(\vec{b}) \neq vv(\vec{b'})$. Let v (resp. w) be the restriction of vv to $g_1, \ldots, g_m$ (resp. $\eta(g_1), \ldots, \eta(g_n)$). More precisely, (v, w) is a *compatible* pair of valuations for V^p such that $v(\vec{a}) = w(\vec{a})$ but $v(\vec{b}) \neq w(\vec{b})$. Thus, $V^p \not\models \vec{a} \rightarrow_p \vec{b}$. The reverse direction and efficiency of the procedure follow directly from the Chase and Prop. 3.1. □

Example 3.4. *Consider the view:*

$$V_7^p() :- \mathtt{R}^p(x, y; u), \mathtt{U}^p(x; z), \mathtt{T}^p(x, z; u) \tag{7}$$

We want to check $V_7^p \models x \rightarrow_p u$. *Alg. 1 forms VV by making copies of* V_7^p *and equating x in the copies as follows:*

$$VV_7^p(u, u') :- \quad \mathtt{R}^p(x, y\ ; u), \mathtt{U}^p(x; z), \mathtt{T}^p(x, z\ ; y),$$
$$\mathtt{R}^p(x, y'; u'), \mathtt{U}^p(x; z'), \mathtt{T}^p(x, z'; y')$$

The Chase first uses $K_{\mathtt{U}} \rightarrow A_{\mathtt{U}}$ *to derive that* $z = z'$, *then uses* $K_{\mathtt{T}} \rightarrow A_{\mathtt{T}}$ *to force* $y = y'$ *and finally,* $K_{\mathtt{T}} \rightarrow A_{\mathtt{T}}$ *to make* $u = u'$. *Thus, the algorithm says* `'Yes'`. *If we drop any subgoal, we can no longer derive* $u = u'$ *and so the algorithm will say* `'No'`.

4 Problem 1: Representability

The goal of this section is to give a solution to Problem 1, deciding if a view is representable, when the views are described by conjunctive queries and the representation formalism is BID. Since representability is a property of a view on an infinite family of representations, it is not immediately clear that the property is decidable. Our main result is that testing representability is decidable and is Π_2^p-Complete in the size of the view definition. The high complexity motivates us to give an efficient sound (but not complete) test in Sec. 4.2. For the important special case when all probabilistic symbols used in the view definition are distinct, we show that this test is complete as well. We then discuss how our technical result relates to prior art in Sec. 4.3.

4.1 Statement of Main Results

There are two key properties of BID representations: Tuples that differ on a possible worlds key are independent, which we will call **block independent**, and distinct tuples that share a possible worlds key must be disjoint, which we call **disjoint**

in blocks.

Definition 4.1. *Given a view with schema $V^p(H)$ defined in terms of BID symbols. For $K \subseteq H$, we say V^p is K-**block independent** if and only if for any BID instance $\Im$ and $I \subseteq V^p(\Im)$ (def. 2.5) satisfying $\forall s, t \in I\ s[K] = t[K] \implies s = t$, the following equation holds:*

$$\mu(\bigwedge_{s \in I} V^p(s[H])) = \prod_{s \in I} s[\mathbf{P}]$$

*For $K' \subseteq H$, we say $V^p(K', H - K')$ is K'-**disjoint in blocks** if $s, t \in V^p(\Im)$ such that $s[K'] = t[K']$ and $s \neq t$ then:*

$$\mu(V^p(s[H]) \wedge V^p(t[H])) = 0$$

*We say a view V^p is **representable** if there is some K such that V^p is K-block independent and K-disjoint in blocks.*

In other words, V^p is *representable*, i.e. K-block independent and K-disjoint in blocks, if and only if we can represent the output of V^p as a BID table with BID table description $\mathtt{V}(K; H - K; \mathbf{P})$. Deciding if the preceding definition holds for a view is a formal definition of the *view representability* problem. We will first consider $V^p(H)$ and $K \subseteq H$ as given and return to the problem of deducing K from the view definition in Sec. 4.1.3.

4.1.1 Block Independence

Intuitively, two tuples in a view are *not* independent if their value depends on two tuples with the same possible worlds key value.

Definition 4.2. *A tuple t is **disjoint critical** for a Boolean view $V^p()$ if and only if there exists a possible world I such that $V^p(I) \neq V^p(I - \{t\})$. A pair of tuples (s, t) each with the same arity as a probabilistic BID symbol $\mathtt{R}_i^p(K_i; A_i)$ such that $s[K_i] = t[K_i]$ is K-**doubly critical** for a view V^p if $\exists s^o, t^o$ such that $s^o[K] \neq t^o[K]$ and s (resp. t) is **disjoint critical** for $V^p(s^o)$ (resp. $V^p(t^o)$).*

In the above definition, it is important to note that that we do not require that s and t be different tuples, only that they agree on the possible worlds key of some probabilistic relation.

Example 4.1. *Recall from Ex. 2.1 that the view V_1^p returns three tuples $\{t_1^o, t_2^o, t_3^o$. For each $t \in \{t_1^o, t_2^o, t_3^o\}$, the tuples referenced by the symbolic probability for $V_1^p(t)$ are* disjoint critical *for $V_1^p(t)$. For example, r_{11} is disjoint critical for $V_1^p(t_2^o)$ because we can take $I = \{w_2, r_{11}, \mathsf{S}(\text{'D.Lounge'}, \text{'Crab Cakes'})\}$ and $I \models V_1^p(t_2^o)$ but $I - \{r_{11}\} \not\models V_1^p(t_2^o)$. Further, r_{11} is also critical for $V_1^p(t_1^o)$. Since $t_1^o[CR] \neq t_2^o[CR]$, the pair (r_{11}, r_{11}) is an example of a CR-*doubly critical tuple. *Interestingly, there are no C-*doubly critical tuples.

Lemma 4.1. *Given a view $V^p(H)$ and $K \subseteq H$, there are no K-doubly critical tuples if and only if V^p is K-block independent.*

Algorithm 2 K-Block Independence

Input: A conjunctive view $V^p(H)$ and $K \subseteq H$
Output: `'Yes'` iff V^p is K-Block Independent

1: Let $n = |\mathbf{var}(V^p)|$, $C = \{c_1, \ldots, c_{n^2}\}$ be fresh constants
2: Let $\vec{h}$ denote head variables, $\vec{k}$ variables at positions K
3: $\mathbf{D} = \{u | u$ disjoint aware valuation for V^p s.t.
$\forall x \in \mathbf{var}(V^p)\ u(x) \in C \cup \mathbf{const}(V^p)\}$.
4: **if** $\forall v, w \in \mathbf{D}, \forall s \in \mathbf{im}(v), \forall t \in \mathbf{im}(w)$.
$v(\vec{k}) \neq w(\vec{k})$, $s, t \in \mathtt{R}_i^p$ **and** $s[K_i] = t[K_i]$ **implies**
$\mathbf{im}(v) - \{s\} \models V^p(v(\vec{h}))$ **and** $\mathbf{im}(w) - \{t\} \models V^p(w(\vec{h}))$
5: **then return** `'Yes'` **else** `'No'`

The proof of this lemma requires a detailed examination of the multilinear polynomials produced by a view on a probabilistic instance and we leave it for the appendix (Sec. 10). Lem. 4.1 is the basis for Alg. 2, which decides K-block independence by looking for K-doubly critical tuples.

Example 4.2 (Ex. 4.1 continued)**.** *In Ex. 4.1, we observed that there are no C-doubly critical tuples for V_1^p, which implies that V_1^p is C-block independent. Also, we observed that V_1^p is not CR-block independent, because of r_{11}, a CR-doubly critical tuple.*

Complexity. Alg. 2 is in exponential time. However, it is also a Π_2^P algorithm because it consists of nested $\forall\exists$ quantifiers which range over polynomially sized choices[5]. We show in appendix (Sec. 10), that the decision is Π_2^P hard as well, hence complete for Π_2^P. The proof showing Π_2^P-Hardness is a lengthy direct reduction from $\forall\exists$`3-CNF`. Although deciding K-block independence is in general hard, our approximation algorithm in Sec. 4.2 is almost always complete in practice.

Main Result. Summarizing our discussion, we have the following theorem:

Theorem 4.1. *Algorithm 2 is sound and complete. Further, checking that no K-doubly critical exists for a conjunctive view is Π_2^P-Complete.*

4.1.2 Disjoint in Blocks

Having established a test for block independence, we now state how to decide if a query is disjoint within blocks. The idea here is simple: A view fails to be K-disjoint within blocks if and only if there exist distinct tuples which agree on K but can occur in some possible world together. We give a polynomial time algorithm based on a Chase (Sec. 3.2) and the following lemma:

Lemma 4.2. *Given a conjunctive view $V^p(H)$ and $K \subseteq H$ then $V^p \models K \rightarrow_p H$[6] if and only if V^p is K-disjoint in blocks.*

To see the forward direction, consider any two tuples s, t which disagree on K. It must be the case that every valuation such that $v(\vec{h}) = s[H]$ and $w(\vec{h}) = t[H]$ use at least one tuple that is disjoint else $V^p \not\models K \rightarrow_p H$. To see the reverse direction,

[5]Line 4 begins with $\forall$ quantified variables. The $\models$ statements are equivalent to the existence of homomorphisms.
[6]We use $V^p \models K \rightarrow_p H$ to mean $V^p \models \vec{k} \rightarrow_p \vec{h}$ where $\vec{k}$ ($\vec{h}$) is the list of variables and constants at K (resp. H).

observe that if (v, w) is compatible then $\mathbf{im}(v) \cup \mathbf{im}(v) = I$ satisfies the constraints and is a possible world. Hence, s and t are both answers to V^p on I, which is a contradiction to our assumption that V^p is disjoint in blocks.

Algorithm 3 K-Disjoint in Blocks

Input: $V^p(H)$ and $K \subseteq H$

1: **return** $V^p \models K \rightarrow_p H$ (* See Alg. 1 *)

Theorem 4.2. *Algorithm 3 is a sound and complete* PTIME *algorithm to decide given V^p, K and H, if $V^p \models K \rightarrow_p H$ and hence if V^p is K-disjoint in blocks*

Example 4.3. *Consider the following view:*

$$V_8^p(d; r) :- \mathtt{L}^p(d; r), V_2^p(c, r) \tag{8}$$

*where $K = \{D\}$ and $A = \{R\}$. Any compatible pair of disjoint aware valuations that agree on d must agree on r, else they would be inconsistent. Thus, V_8^p is D-*disjoint in blocks*. To see a negative example, observe that V_1^p Eq. (2) is not C-disjoint in blocks because the pair of valuations, $v(c, r, d) =$ ('TD', 'D.Lounge', 'Crab Cakes') and $w(c, r, d) =$ ('TD', 'P.Kitchen', 'Crab Cakes'), is* compatible *and $v(c) = w(c)$ but $v(r) \neq w(r)$.*

4.1.3 Finding Possible Worlds Keys

In previous sections, we assumed that the BID schema for V^p was part of the input; we now consider how to infer the schema for V^p from its definition. Interestingly, we can efficiently find K such that if $V^p(K'; H - K')$ is representable for *any* K' then $V^p(K; H - K)$ is representable. Formally, we efficiently find a **candidate key** K for V^p.

Definition 4.3. *K is a **candidate key** for V^p if V^p is representable if and only if V^p is K-block independent.*

The central observation to find a candidate K for a fixed V^p is the following:

Proposition 4.1. *If V^p is K-disjoint in blocks and K'-block independent then $V^p \models K \rightarrow_r K'$.*

Sketch. Suppose that $V^p \not\models K \twoheadrightarrow_r K'$ then there is a representation on which the output of V^p contains tuples s, t such that $s[K] = t[K]$ but $s[K'] \neq t[K']$, $s[\mathbf{P}] > 0$ and $t[\mathbf{P}] > 0$. Since s, t agree on K but, $s \neq t$ and V^p is K-disjoint in blocks this implies s, t are disjoint. On the other hand, they disagree on K' which since V^p is K'-block independent implies s, t are independent. Since a pair of events with positive probability cannot be both independent and disjoint; we reach a contradiction. □

This proposition says something interesting: Informally, up to $\rightarrow_r$ equivalence, there is a *unique* choice of K for which $V^p(K; H - K)$ can be representable. Since we can infer these dependencies in PTIME (Alg. 3), Prop. 4.1 suggests the efficient algorithm in Alg. 4.

Algorithm 4 Finding a candidate key for V^p

Input: V^p, a conjunctive view
Output: Candidate key K for V^p

```
W^p(H_W) ← DJA(V^p)
K ← H_W
for each A ∈ H do
    if V^p ⊨ K − {A} →_p H then (* see Alg. 1 *)
        K ← K − {A}
return K (* K is a minimal possible worlds key *)
```

Theorem 4.3. *When there are no functional dependencies in the representation, Algorithm 4 correctly finds a candidate key K.*

To get an intuition for Thm. 4.3, we observe that the returned $K \subseteq H$ satisfies $V^p \models K \rightarrow_r K'$ for any representable $V^p(K'; H)$. Prop. 3.2 implies that the chase homomorphism, θ satisfies $\theta(K') \subseteq \theta(K)$. If $V^p(K; H - K)$ is not representable, we show that, $\theta(K') \subset \theta(K)$. This allows us to construct a strict subset of K, call it K_0, such that $V \models K_0 \rightarrow_p H$, which is a contradiction to K's minimality. In particular, take $K_0 = \theta^{-1}(K') \cap K$, which is valid because θ is surjective (Prop. 3.1).

4.1.4 A Solution for Problem 1

We have now established all the necessary ingredients to solve problem 1 for conjunctive views, which we summarize in the following theorem:

Theorem 4.4. *Given a conjunctive view $V^p(H)$, deciding if there is some K such that output of V^p can be represented as a single BID relation $V(K; H - K; \mathbf{P})$ is decidable. Further, it is Π_2^p Complete.*

The algorithm first runs Alg. 4 which returns a candidate key K, which we use as input to Alg. 2.

Remark 4.1. *One may suspect that the hardness is the result of the strong independence requirement. However, in the appendix (Sec. 10.7), we show that checking even much weaker probabilistic requirements remains Π_2^p Complete.*

4.2 Practical Algorithm for Representability

Since the intractable portion of the representability check is deciding K-block independence, we give a polynomial time approximation for K-block independence that is sound, i.e. it says a view is representable only if it is representable. However, it may not be complete, declaring that a view is not representable, when in fact it is. The central notion is a $\vec{k}$-*collision*, which intuitively says there are two output tuples which may depend on input tuples that are not independent (i.e. the same tuple or disjoint).

Definition 4.4. *A $\vec{k}$-**collision** for a view*

$$V^p(\vec{k}, \vec{a}) :- g_1, \ldots, g_n$$

Algorithm 5 Finding a K-Collision for V^p

Input: $V^p(H) :- g_1, \ldots, g_n$ and K
Output: `'Yes'` iff V^p has a collision

1: **for each** $i, j \in 1, \ldots, n$ **do**
2: (* Make a fresh copy of V^p *)
$V_1^p(K,\ H-K) :-\ g_1, \ldots, g_n$
$V_2^p(K', H-K') :-\ g'_1, \ldots, g'_n$
3: **if** g_i is probabilistic **and pred**(g_i) = **pred**(g'_j) **then**
4: Unify $g_i[K_i] = g'_j[K_j]$.
5: Let $W_1 \leftarrow$ **DJA**$(V_1), W_2 \leftarrow$ **DJA**(V_2)
6: **if** Chase Succeeds **and** $W_1[K] \neq W_2[K']$ **then**
7: **return** `'Yes'` (* There is a Collision. *)
8: **return** `'No'` (* There is no Collision. *)

Algorithm 6 Practical K-Block Independence

Input: $V^p(H)$ a conjunctive view and $K \subseteq H$
Output: `'Yes'` only if V^p is K-Block Independent
return `'Yes'` if $V^p(K; H-K)$ has no K-Collision.

is a pair of disjoint aware valuations (v, w) *such that* $v(\vec{k}) \neq w(\vec{k})$ *but there exists* i, j *such that* g_i *that is probabilistic,* **pred**(g_i) = **pred**(g_j) *and* $v(\vec{k_i}) = w(\vec{k_j})$.

Theorem 4.5. *For a view* $V^p(H)$ *and* $K \subseteq H$*, if algorithm 6 outputs* `'Yes'` *then* V^p *is guaranteed to be K-block independent. Further, if* V^p *does not contain repeated probabilistic subgoals then algorithm 6 is complete. The algorithm is* `PTIME`.

When V^p does not contain repeated probabilistic subgoals the algorithm is complete because every probabilistic tuple in the image of a valuation must be critical. In particular, the image of g_i and g_j in the definition of collision are critical.

Example 4.4. *Consider* $V_2^p(C)$ *in Eq. (3), if we unify any pair of probabilistic subgoals, we are forced to unify the head, c. This means that a collision is never possible and we conclude that* V_2^p *is C-block independent. Notice that we can unify the* `S` *subgoal for distinct values of c, since* `S` *is deterministic, this is not a collision. In* $V_1^p(c, r)$ *Eq. (2), the following pair* (v, w)*,* $v(c, r, d) =$ (*'TD'*, *'D.Lounge'*,*'Crab Cakes'*) *and* $w(c, r, d) =$ (*'TD'*, *'P.Kitchen'*, *'Crab Cakes'*)*, is a collision because* $v(c, r) \neq w(c, r)$ *and we have unified the keys of the* $\mathtt{R}^p$ *subgoal. Since there are no repeated probabilistic subgoals, we are sure that* V_1^p *is not CR-block independent.*

4.3 Extensions and Discussion

Extending to Many Views. In a BID instance, tuples in distinct views must be *independent*. The following pair of views illustrates the problem:

$$V_x^p(x) :- \mathtt{T}^p(x, y, z;) \text{ and } V_y^p(y) :- \mathtt{T}^p(x, y, z;)$$

Each view is representable by itself. However, all tuples in `T` contribute to each view, so the pair of views is not representable.

It is straightforward to extend our test to handle independence of tuples in distinct views and is left for the full paper.

Dependencies. We can extend each of the algorithms to handle dependencies in a straightforward way with the exception of the search for candidate keys. To deduce the appropriate K from the definition of V^p such that $V^p(K; H - K)$ is representable, the algorithm of Sec. 4.1.3 relies on only trivial functional dependencies holding in the representation. The naive adaptation of this algorithm requires time polynomial in the number of functional dependencies which can be exponential in the number of head variables. In the appendix (Sec.10.4), we give an algorithm to handle finding candidate keys when functional dependencies are present.

Relation to Query Evaluation. We have observed that efficient query evaluation for a view and representability are distinct concepts. To see this, observe that Thm. 4.1 shows that any single Boolean view is *representable*. Some Boolean queries have high complexity (#P) [19, 36]. When a query has a PTIME algorithm, it is called **safe**. This implies that not every representable view is safe. On the other hand, Ex. 2.1 gives an example of a non-representable view that has a safe plan. However, not all queries have safe plans, but for conjunctive queries there are efficient schemes to approximate probabilities to essentially any desired precision [37]. Using the result of approximation schemes for materialized view optimizations and providing error guarantees is an interesting open question.

Complex Correlations. The problem of K-Block Independence is to decide: For tuples s, t, is it the case that $\mu(V_1^p(s) \wedge V_2^p(t)) = \mu(V_1^p(s))\mu(V_2^p(t))$? In [33], a similar problem was studied where V_1^p is a secret query and V_2^p is a public view and our goal is to determine if the secret query and public view are independent. It was shown that this problem is Π_2^P Complete[7]. That work used a more restrictive tuple independent model in which the FKG inequality [3] $\mu(V_1^p(s) \wedge V_2^p(t)) \geq \mu(V_1^p(s))\mu(V_2^p(t))$ holds. Fig. 2 shows that this inequality no longer holds in our setting by showing a view V_9^p and family of representations such that tuples in V_9^p are positively correlated, negatively correlated or even independent depending on how we set the probabilities in the representations. This technical difference is significant because the proof in [33] is an inductive argument that relies on the FKG inequality. Since the FKG inequality does not hold, we must use a completely different technique.

Example 4.5. *Consider the family of representations given in Fig. 2. Consider the query:*

$$V_9^p(k_2) :- \mathtt{M}_1^p(k_1; x), \mathtt{M}_2^p(k_2; x) \tag{9}$$

The probabilities are described symbolically in the figure. Thus, $\mu(V_9^p(a) \wedge V_9^p(b)) = a_H b_H c_T + a_T b_T c_H$. *In case* (I)*, the tuples appear to be pairwise independent, but this does not hold for every distribution. For example in case* (P) *the two tuples are positively correlated, while in* (N) *they are negatively correlated. These correlations are possible even though* V_9^p *is a very simple conjunctive view. They are the result of the more sophisticated BID representation system, which allows disjoint events.*

[7] However, there seems to be no direct reduction to our problem in the single view case.

K_1	A_1	**P**
c	H	c_H
	T	c_T

$\mathtt{M}_1(K_1; A_1; \mathbf{P})$

K_2	A_2	**P**
a	H	a_H
	T	a_T
b	H	b_H
	T	b_T

$\mathtt{M}_2(K_2; A_2; \mathbf{P})$

K_2	**P**
a	$a_H c_T + a_T c_H$
b	$b_H c_T + b_T c_H$

$V_9^p(k_2) :\!- \mathtt{M}_1^p(k_1; x), \mathtt{M}_2^p(k_2; x)$

	a_H	b_H	c_H	$\mu(V_9^p(a) \wedge V_9^p(b))$
(I)	0.5	0.5	0.5	0.25
(P)	0.9	0.9	0.5	0.41
(N)	0.9	0.1	0.5	0.09

On all three representations, $\mu(V_9^p(a))\mu(V_9^p(b)) = 0.25$.

Figure 2. Sample Data for Discussion. If all probabilities are 0.5**,** $V_9^p(a)$ **and** $V_9^p(b)$ **appear to be independent.** $l \in \{a, b, c\}$ $l_H = 1 - l_T$**.**

5 Problem 2: Querying using Views

In this section, we study problem 2: Given a conjunctive query Q written using a materialized view V^p is the value of $\mu(Q)$ uniquely defined? Of course, if V^p is representable this problem is trivial: Q's value is always uniquely defined.

5.1 Partially Representable Views

In contrast to an ordinary probabilistic materialized view that represents a unique probability distribution, a partially represented view represents *many* probability distributions, each of which we call *agreeable*.

Definition 5.1. *A **partial BID view description** is a relational schema with the attributes partitioned into four classes:*

$$V(K_I; D; A; \mathbf{P})$$

*where K_I is called the **independence key**, $K_I D$ is called the **disjointness key**, A is called the **value attribute set** and* **P** *is called the **probability attribute**, a distinguished attribute taking values in the half-open interval* $(0, 1]$.

Example 5.1. *Recall that $V_1^p(C, R)$ from Eq. (2) is not representable. We will show that it is partially representable with syntax: $V_1(C; R; \emptyset; \mathbf{P})$.*

The intuition is that a partial representation preserves marginal probabilities but may not specify all correlations: If a set of tuples differ on K_I, they are independent. If two distinct tuples agree on $K_I D$, they are disjoint. However, if two tuples agree on K_I but disagree on D, they may be correlated in complicated ways.

Definition 5.2 (Semantics)**.** *Given a view $V^p(H)$ and a partition K_I, D, A of H, we say V^p is **partially representable** if V^p is K_I-block independent and $K_I D$-disjoint in blocks.*

Definition 5.3. *A possible world is a set of tuples, I, that satisfies $K_I D \to A$. We say a distribution on possible worlds, μ,* ***agrees*** *with $V^p(K_I; D; A)$ if for any set of tuples $I \subseteq V^p(\mathfrak{I})$ without* $\mathbf{P}$ *that satisfy $s[K_I] = t[K_I] \implies s = t$ then*

$$\mu(\bigwedge_{t \in I} V^p(t)) = \prod_{t \in I} \mu(V^p(t))$$

In particular, if $D = \emptyset$, then Def. 5.3 coincides with Def 2.2 and so the partial representation uniquely defines a probability distribution. Any view has a trivial partial representation with $K_I = A = \emptyset$ and $D = H$. In the previous section, we showed that checking K-block independence is Π_2^P-Complete. Thus, the following is immediate:

Theorem 5.1. *Given a conjunctive view V^p with head H and K, D, A satisfying $K \oplus D \oplus A = H$, deciding if the output of V^p is partially representable as $V(K; D; A; \mathbf{P})$ is Π_2^P-Complete.*

5.2 Statement of Main Results

Intuitively, a query's value fails to be uniquely defined if it depends on two tuples whose correlation is not specified by the partial representation. Due to space constraints, we present queries that use a single partially representable view.

Definition 5.4. *A* ***critical pair*** *for a Boolean query $Q()$ is a pair of distinct tuples (s, t) such that there exists a possible world I satisfying*

$$Q(I - \{s, t\}) \neq Q(I) \text{ and } Q(I - \{s\}) = Q(I - \{t\})$$

*Given a partially representable view $V^p(K_I; D; A)$, a pair of tuples (s, t) is called V^p-****intertwined*** *if $s, t \in V^p$ and $s[K_I] = t[K_I]$ but $s[D] \neq t[D]$.*

In contrast to K-doubly critical tuples, the possible world I must be the same for s, t.

Example 5.2 (Running Example)**.** *Consider the partial representation in Ex. 5.1 and the queries:*

$$Q_1() :- V_1^p(c, r) \text{ and } Q_2(c) :- V_1^p(c, \text{`D.Lounge'})$$

t_1^o and t_2^o are a critical pair of tuples for Q_1 and are V_1^p-intertwined. For any fixed c_0, there is no critical pair of tuples of V_1^p-intertwined tuples for $Q_2(c_0)$.

We state the link between intertwined tuples and distributions that agree with a view.

Proposition 5.1. *Given a partially representable view $V^p(K_I; D; A)$, μ be a distribution that agrees with V^p and $s, t \in V^p$ that are V^p-intertwined such that $\mu(s) \neq 1$ and $\mu(t) \neq 1$ then there exists a distribution ν that agrees with V^p, such that $\mu(s \wedge t) \neq \nu(s \wedge t)$ and $\mu(s \vee t) \neq \nu(s \vee t)$.*

5.2.1 Critical Intertwined Captures Uniqueness

A query $Q()$ is **uniquely defined** if for any two agreeable distributions,μ, ν, we have $\mu(Q()) = \nu(Q())$. We establish that the existence of a critical pair of intertwined tuples captures when a query fails to be *uniquely defined*.

Lemma 5.1. *There exist a critical pair of intertwined tuples for a conjunctive query $Q()$ if and only if $Q()$ is not uniquely defined.*

To see the forward direction consider a conjunctive query Q. If there is a pair of critical tuples (s, t) for $Q()$, then there are two cases: $I - \{s\} \models Q()$, in which case Q is satisfied when either of s, t are present, or $I - \{s\} \not\models Q()$, in which case $Q()$ is satisfied only when s and t are both present. Since I is a possible world, we can create a representation $\Im$ such that s, t are the only tuples with $\mu \neq 1$. For a possible world J of $\Im$, $J \models Q() \iff J \models s \wedge t$ or$J \models Q() \iff J \models s \vee t$, by Prop. 5.1 neither is uniquely defined. The reverse direction is an inductive proof that gives less information and is in the appendix (Sec. 12.1).

Example 5.3 (Continuing Ex. 5.2)**.** *A distribution, μ, that always agrees with V_1^p is the result of inlining of V_1^p in Q_1. Here $\mu(Q_1()) \approx 0.905$. A second distribution that agrees with V_1^p, ν, is to assume independence. Thus, $\nu(Q_1) = 1 - (1 - 0.72)(1 - 0.602)(1 - 0.32) \approx 0.924$. As we saw in Ex. 5.2, Q_1 does have a critical pair of intertwined tuples. On the other hand, for each c value, the query Q_2 is uniquely defined, in the example its value is* 0.72*.*

Theorem 5.2. *Given a query Q using a partially representable view V^p, deciding if Q's value is uniquely defined is Π_2^p Complete.*

Let $n = |\mathbf{var}(Q)|$ and C be a set of n^2 fresh constants; a complete algorithm checks that for all possible worlds with domains in $\mathbf{const}(Q) \cup C$, there is not a critical pair of intertwined tuples; this algorithm is in Π_2^p.

5.3 Practical Test for Uniqueness

Definition 5.5. *Given a schema with a single partially representable view V^p, an **intertwined collision** for a query $Q(H)$ is a pair of compatible valuations (v, w) such that $v(\vec{h}) = w(\vec{h})$ and there exists a pair of subgoals, (g_i, g_j), such that $\mathbf{pred}(g_i) = \mathbf{pred}(g_j) = V^p$, $v(\vec{ki}_i) = w(\vec{ki}_j)$ and $v(\vec{d}_i) \neq w(\vec{d}_j)$ where $\vec{ki}_i$ ($\vec{ki}_j$) is the list of variables at K_I in g_i (resp. g_j) and $\vec{d}_i$ ($\vec{d}_j$) is the list of variables at D in g_i (resp. g_j).*

The algorithm to find an intertwined collision is a straightforward extension of finding a K-collision. The key difference is that we use the Chase to ensure that the valuations we find are compatible, not individually disjoint aware.

Theorem 5.3. *If no intertwined collisions exist for a conjunctive query Q, then its value is uniquely defined. If the partially representable view symbol V^p is not repeated, this test is complete. The test can be implemented in* `PTIME`*.*

Sketch. We sketch the soundness argument in the special case of a Boolean query $Q()$. We show that if there exists a critical intertwined pair (s, t) for Q, then there must be an intertwined collision. Let I be the instance provided by Def. 5.4. Suppose, $I - \{s\} \models Q()$. Since $I - \{s, t\} \not\models Q()$, the image of any valuation v that witnesses $I - \{s\} \models Q()$ must contain t. By symmetry, the image of any valuation that witnesses $I - \{t\} \models Q()$ must contain w. It is easy to see that (v, w) is compatible and hence (v, w) is an intertwined collision. If $I - \{s\} \not\models Q()$ then there is a single valuation v which uses both s, t. Thus, (v, v) is an intertwined collision. □

Example 5.4. *In V_1^p, $K_I = \{C\}$ and $D = \{R\}$. An intertwined collision for Q_1 is $v(c, r) =$ ('TD', 'D.Lounge') and $w(c, r) =$ ('TD', 'P.Kitchen'), thus Q's value is not uniquely defined. On the other hand, in Q_2, trivially there is no intertwined collision and so Q_2's value is uniquely defined.*

5.4 Extensions and Discussion

Optimization. In an optimizer, we would like *syntactic independence* [9], which is the ability to rewrite a query Q that does not use a materialized view V into an equivalent Q' that does use V. The same theory applies, but we must additionally check that Q' correctly uses a view as described in Sec. 5.2. A key difference in query optimization is that we usually have access to the view definitions. When the view definitions are present, a partial representation for a view essentially strips the view's lineage. If a query's value is uniquely defined, its value is the same as inlining the view definition. In the appendix (Sec. 12.2), we show that deciding uniqueness in this setting is Π_2^p complete and give `PTIME` approximations.

View Selection. Informally, the view selection problem [11] is to select given a set of queries $\mathcal{Q}$, the workload and a space budget B, choose a set of views $\mathcal{V}$ to materialize within the space budget B to minimize the cost of $\mathcal{Q}$. In the probabilistic setting, we now also check each $q \in \mathcal{Q}$ is uniquely defined using $\mathcal{V}$. The new twist is that the cost function has a large step: If a query Q can be executed using a *safe plan* [19, 36] over a view V, the cost of executing Q is dramatically lower.

6 Related Work

Materialized views are a fundamental technique used to optimize queries [1, 9, 25, 28] and as a means to share, protect and integrate data [33, 40] that are currently implemented by all major database vendors. Because the complexity of deciding when a query can use a view is high, there has been a considerable amount of work on making query answering using views algorithms scalable [25, 35]. In the same spirit, we provide efficient practical algorithms for our representability problems.

Recently, probabilistic databases have received attention because of their ability to deal with uncertainty resulting from data cleaning tasks [4, 37], information extraction [8, 27] and sensor data [21, 30]. This has resulted in several systems [6, 21, 38, 41] with accompanying work on probabilistic query processing [10, 19, 36, 37, 39]. Prior art has considered using a representation system [20, 37, 38] that can represent every conjunctive view. Typically, these systems use base tables

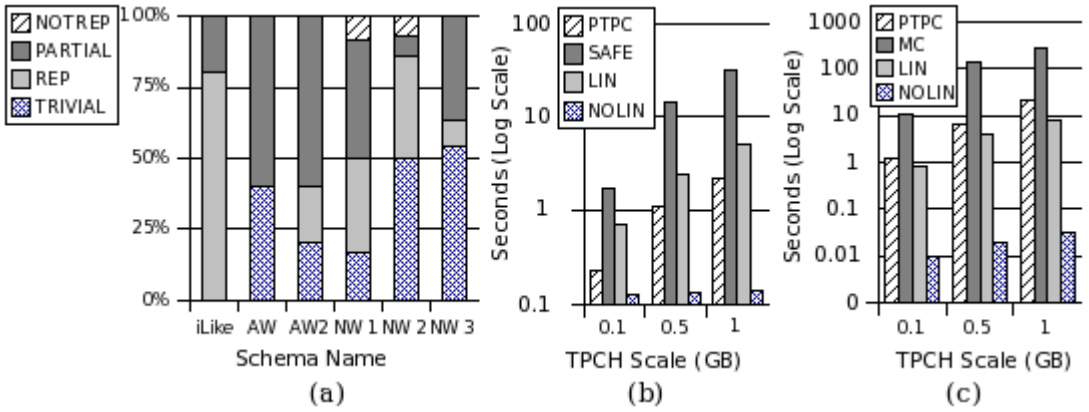

Figure 3. (a) Percentage by workload that are representable, non-trivially partially representable or not representable. We see that almost all views have some non-trivial partial representation. (b) Running times for Query 10 which is safe. (c) Retrieval times for Query 5 which is not safe. Performance data is TPC-H (0.1, 0.5, 1G) data sets. All running times in seconds and on logarithmic scale.

that are similar to BID representations but then introduce auxiliary information (e.g. lineage [41] or factors [38]) to track correlations introduced by query processing.

In prior art [20], the following question is studied: Given a class of queries $\mathcal{Q}$ is a particular representation formalism closed for all $Q \in \mathcal{Q}$? In contrast, our test is more fine-grained: For any fixed conjunctive Q, is the *BID* formalism closed under Q? Also relevant for expanding the class of practical algorithm is the recent work in [32].

7 Experiments

In this section we answer three main questions: To what extent do representable and partially representable views occur in real and synthetic data sets? How much do probabilistic materialized views help query processing? How expensive are our proposed algorithms for finding representable views?

7.1 Experimental Setup

Data Description. We experimented with a variety of real and synthetic data sets including: a database from iLike.com [13], the Northwind database (NW) [14], the Adventure Works Database from SQL Server 2005 (AW)[15] and the TPC-H/R benchmark (TPCH) [16, 17]. We manually created several probabilistic schemata based on the Adventure Works [15], Northwind [14] and TPC-H data which are described in Fig. 4.

Queries and Views. We interpreted all queries and views with scalar aggregation as probabilistic existence operators (i.e. computing the probability a tuple is present). iLike, Northwind and Adventure Works had predefined views as part of the schema. We created materialized views for TPC-H using an exhaustive procedure to find all subqueries that were

Schema	Tables (w/**P**)
AW	18 (6)
AW2	18 (3)
NW1	16 (2)
NW2	16 (5)
NW3	16 (4)
TPC-H	8 (5)

(a)

Size (w/**P**)	Tuples (w/**P**)
0.1 (440M)	3.3M (2.4M)
0.5 (2.1G)	16M (11.6M)
1.0 (4.2G)	32M (23.2M)

(b)

Figure 4. Schema and TPC Data statistics. (a) Number of tables referenced by at least one view and number of probabilistic tables (i.e. with attribute **P**). (b) Size and (w/**P**) are in Gb. The number of deterministic and probabilistic tuples is in millions.

representable, did not contain cross products and joined at least two probabilistic relations.

Real data: iLike.com We were given query logs and the relational schema of iLike.com, which is interesting for three reasons: It is a real company, a core activity of iLike is manipulating uncertain data (e.g. similarity scores) and the schema contains materialized views. iLike's data, though not natively probabilistic, is easily mapped to a BID representation. The schema contains over 200 tables of which a handful contain uncertain information. The workload trace contains over 7 million queries of which more than 100,000 manipulated uncertain information contained in 5 views. Of these 100,000 queries, we identified less than 10 query types which ranged from simple selections to complicated many way joins.

Performance Data All performance experiments use the TPC-H data set with a probabilistic schema containing uncertainty in the `part`, `orders`, `customer`, `supplier` and `lineitem` tables. We used the TPC-H tool `dbgen` to generate relational data. The data in each table marked as probabilistic was then transformed by uniformly at random injecting additional tuples such that each key value was expected to occur 2.5 times. We allowed for *entity uncertainty*, that is, the sum of probabilities for a possible worlds key may be less than 1.

System Details. Our experimental machine was a Windows Server 2003 machine running SQL Server 2005 with 4GB of RAM, 700G Ultra ATA drive and dual Xeon (3GHz) processors. The Mystiq engine is a middleware system that functions as a preprocessor and uses a complete approach [6, 37]. The materialized view tools are implemented using approximately 5000 lines of OCaml. After importing all probabilistic materialized views, we tuned the database using only the SQL Server Database Engine Tuning Advisor.

Execution Time Reporting Method. We reduced query time variance by executing each query seven times, dropping the highest and lowest times and averaging the remaining five times. In all reported numbers, the variance of the five runs was less than 5% of query execution time.

7.2 Question 1: Do Representable and Partially Representable views exist?

In Fig. 3(a), we show the percentage of views in each workload that is trivially representable because there are no probabilities in the view *(TRIVIAL)*, representable *(REP)*, non-trivially partially representable *(PARTIAL)* or only trivially partially representable *(NOTREP)*. In iLike's workload, 4 of the 5 views (80%) are representable. Further, 98.5% of the over 100k queries that manipulate uncertain data use the representable views. In synthetic data sets, representable views exist as well. In fact, 50% or more of the views in each data set except for AW are representable. Overall, 63% of views are representable. 45% of the representable views are non-trivially representable. Additionally, almost all views we examined have a non-trivial partial representations (over 95%). We conclude that that representable and partially representable views exist and can be used in practice.

7.3 Question 2: Do our techniques make query processing more efficient?

The TPC data set is the basis for our performance experiments because it is reproducible and the data can be scaled arbitrarily. We present queries 5 and 10, because they both have many joins (6 and 4) and they are contrasting: Query 10 is *safe* [19, 36], and so can be efficiently evaluated by a modified SQL query. Query 5 is *unsafe* and so requires expensive Monte Carlo techniques. Graphs 3(b) and 3(c) report the time taken to execute the query and retrieve the results. For query 10, this is the total time for execution because it is safe. In contrast, query 5 requires additional Monte Carlo techniques to compute output probabilities.

Graph Discussion. In Fig. 3(b), we see running times of query 10 without probabilistic semantics *(PTPC)*, as a safe plan *(SAFE)*, with a subview materialized and retaining lineage *(LIN)* and the same subview without lineage *(NOLIN)*. *LIN* is equivalent to a standard materialized view optimization; the lineage information is computed and stored as a table. In *NOLIN*, we discard the lineage and retain only the probability that a tuple appears in the view. The graph confirms that materializing the lineage yields an order of magnitude improvement for safe queries because we do not need to compute three of the four joins at query execution time. Interestingly, the bars for *NOLIN* show that precomputing the probabilities and ignoring the lineage yields an additional order of magnitude improvement. This optimization is correct because the materialized view is *representable*. This is interesting because it shows that being aware of when we can remove lineage is helpful even for safe plans.

As a baseline, Fig. 3(c) shows the query execution times for query 5 without probabilistic semantics but using the enlarged probabilistic tables *(PTPC)*. Fig. 3(c) also shows the cost of retrieving the tuples necessary for Monte Carlo simulation *(MC)*. Similarly, we also see the cost when materializing a view and retaining lineage *(LIN)* and when we precompute the probabilities and discard the lineage *(NOLIN)*. For *(MC)* and *(LIN)*, the extra step of Monte Carlo Simulation is necessary which for TPC 0.1 (resp. TPC 0.5, TPC 1) requires an additional 13.62 seconds (resp. 62.32s, 138.21s). Interestingly, query

5 using the materialized view does *not* require Monte Carlo Simulation because the rewritten query is safe. Thus, the time for *NOLIN* is an end-to-end running time and so we conclude that our techniques offer four order of magnitude improvement over materializing the lineage alone ($8.2s + 138.21s$ with lineage v. 0.03s without).

7.4 Question 3: How costly are our algorithms?

All views listed in this paper were correctly classified by our practical algorithm (Alg. 6), which always executes in well under 1 second. Finding all representable or partially representable sub-views for all but two queries completed in under 145 seconds; the other two queries completed in under an hour. Materializing views for unsafe queries completed under 1.5 hours for all results reported in the paper. However, this is an offline process and can be parallelized because it can utilize multiple separate Monte Carlo processes.

8 Conclusion

We have formalized and solved the problems of representability and using partial representable views to answer queries in the case of conjunctive views and queries. We have shown that representable and partially representable views exist in real and synthetic data sets and demonstrated that understanding representability is a large optimization win even in complete approaches.

References

[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. pages 254–263, 1998.

[2] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley Publishing Co, 1995.

[3] N. Alon and J. Spencer. *The Probabilistic Method*. John Wiley, 1992.

[4] P. Andritsos, A. Fuxman, and R. J. Miller. Clean answers over dirty databases. In *ICDE*, 2006.

[5] F. Bacchus, A.J. Grove, J.Y. Halpern, and D.Koller. Generating new beliefs from old. In *Proceedings of UAI*, pages 37–45, 1994.

[6] J. Boulos, N .Dalvi, B. Mandhani, S. Mathur, C. Ré, and D. Suciu. Mystiq: A system for finding more answers by using probabilities. In *SIGMOD*, 2005. system demo.

[7] P. Buneman, A. Chapman, and J. Cheney. Provenance management in curated databases. In S. Chaudhuri, V. Hristidis, and N. Polyzotis, editors, *SIGMOD Conference*, pages 539–550. ACM, 2006.

[8] M.J. Cafarella, C. Ré, D. Suciu, and O. Etzioni. Structured querying of web text data: A technical challenge. In *CIDR*, pages 225–234. www.crdrdb.org, 2007.

[9] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim. Optimizing queries with materialized views. *icde*, 00:190, 1995.

[10] R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD*, pages 551–562, 2003.

[11] R. Chirkova, A. Halevy, and D. Suciu. A formal perspective on the view selection problem. In *Proceedings of VLDB*, Rome, Italy, September 2001.

[12] Garage Band Corp. `http://www.garageband.com/`.

[13] Garage Band Corp. `www.ilike.com`.

[14] Microsoft Corp. Northwind for sql server 2000.

[15] Microsoft Corp. Sql server 2005 samples (feb. 2007).

[16] Transaction Processing Performance Council. Tpc-h (ad-hoc, decision support) benchmark. http://www.tpc.org/.

[17] Transaction Processing Performance Council. Tpc-r (decision support) benchmark (obsolete). http://www.tpc.org/.

[18] Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. *VLDBJ*, 12(1):41–58, 2003.

[19] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, Toronto, Canada, 2004.

[20] A. Das Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working models for uncertain data. In *ICDE*, 2006.

[21] A. Deshpande, C. Guestrin, S. Madden, J.M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, pages 588–599, 2004.

[22] A. Deutsch, L. Popa, and V. Tannen. Physical data independence, constraints and optimization with universal plans. In *VLDB*, 1999.

[23] O. Etzioni, M .Banko, and M.J. Cafarella. Machine reading. In *AAAI*. AAAI Press, 2006.

[24] G.Miklau and Dan Suciu. Preprint: A formal analysis of information disclosure in data exchange. *JCSS*, To appear. `http://www.cs.washington.edu/homes/suciu/miklau_suciu_jcsspreprint.pdf`.

[25] J. Goldstein and P. Larson. Optimizing queries using materialized views: a practical, scalable solution. In *SIGMOD 2001*, pages 331–342, New York, NY, USA, 2001. ACM Press.

[26] T.J. Green and V. Tannen. Models for incomplete and probabilistic information. *IEEE Data Engineering Bulletin*, 29(1):17–24, March 2006.

[27] R. Gupta and S. Sarawagi. Curating probabilistic databases from information extraction models. In *Proc. of the 32nd Int'l Conference on Very Large Databases (VLDB)*, 2006.

[28] A. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.

[29] T.S. Jayram, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. Zhu. Avatar information extraction system. *IEEE Data Engineering Bulletin*, 29(1), 2006.

[30] N. Khoussainova, M. Balazinska, and D. Suciu. Probabilistic rfid data management. Technical Reprot TR2007-03-01, University of Washington, Seattle, Washington, March 2007.

[31] L. Lakshmanan, N. Leone, R. Ross, and V.S. Subrahmanian. Probview: A flexible probabilistic database system. *ACM Trans. Database Syst.*, 22(3), 1997.

[32] A. Machanavajjhala and J .Gehrke. On the efficiency of checking perfect privacy. In Stijn Vansummeren, editor, *PODS*, pages 163–172. ACM, 2006.

[33] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. In *SIGMOD*, 2004.

[34] O.Etzioni, M.J. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D.S. Weld, and A. Yates. Web-scale information extraction in knowitall: (preliminary results). In S.I. Feldman, M. Uretsky, M. Najork, and C.E. Wills, editors, *WWW*, pages 100–110. ACM, 2004.

[35] R. Pottinger and A. Halevy. Minicon: A scalable algorithm for answering queries using views. *The VLDB Journal*, 10(2-3):182–198, 2001.

[36] C. Ré, N. Dalvi, and D. Suciu. Query evaluation on probabilistic databases. *IEEE Data Engineering Bulletin*, 29(1):25–31, 2006.

[37] C. Ré, N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *Proceedings of ICDE*, 2007.

[38] P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *Proceedings of ICDE*, 2007.

[39] M. Soliman, I.F. Ilyas, and K. Chen-Chaun Chang. Top-k query processing in uncertain databases. In *Proceedings of ICDE*, 2007.

[40] J. D. Ullman. Information integration using logical views. In *ICDT*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 1997.

[41] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, pages 262–276, 2005.

9 Appendix A: Proofs for Preliminaries

9.1 Proof of Prop. 3.2

Proposition 9.1. *Let θ be the chase homomorphism, then if v is a disjoint aware valuation for V then if $\theta(x) = \theta(y)$ implies $v(x) = v(y)$.*

Sketch. To start, θ is identity and so the claim holds vacuously. Since during execution, we only equate two variables if any mapping that did not equate them would violate the key constraint, it holds after each chase step. □

Proposition 9.2 (Restatement of Prop. 3.2)**.** *Let* $\mathbf{DJA}(V)$ *exist then* $V \models \vec{a} \rightarrow_r \vec{b}$ *iff* $\mathbf{DJA}(V) \models \theta(\vec{a}) \rightarrow \theta(\vec{b})$

Proof. Let $W = \mathbf{DJA}(V)$, which exists by assumption. By the chase, there is a chase homomorphism $\theta : V \rightarrow W$.

Forward direction Let v', w' be valuations for W. We assume for contradiction that $v'(A) = w'(A)$ but $v'(B) \neq w'(B)$. If v, w are disjoint aware, then we immediately get a contradiction to $V \models \vec{a} \rightarrow \vec{b}$, because we find, by composition $v = v' \circ \theta$, a v and w such that $v(A) = w(A)$ and $v(B) \neq w(B)$ and are disjoint aware.

We now how to transform (v', w') into a new valuation v^* for W such that $v^*(A) = w^*(A)$, $v^*(B) \neq w^*(B)$ with v^*, w^* disjoint aware. By our previous argument, this results in a contradiction. At each step we reduce the number of potentially violated keys: where a key is potentially violated if it has the same value as another key. We may create new violations but once we fix a pair of subgoals, it will be shown the two subgoals can never be in conflict in the execution of the algorithm. Thus, the algorithm terminates in at most $O(n^2)$ steps.

Since v' is not disjoint aware, there is some (K_i, A_i) and (K_j, A_j) such that $v'(K_i) = v'(K_j)$ but $v'(A_i) \neq v(A_j)$. We observe that $K_i \neq K_j$, because W is the result of the chase and this would imply $A_i = A_j$. Thus, there are is a variable $x \in K_i \,\Delta\, K_j$ (symmetric difference) that differs in a position. Suppose $x \notin B - A$, then consider new valuations v_0, w_0 such that $v_0(x) = w_0(x) = c$ where c is a fresh constant. Then it is the case that $v_0(A) = w_0(A)$ and $v_0(B) \neq v_0(B)$. If $x \in B - A$, if x is the only variable in B which they differ x to the same constant could make $v_0(B) = w_0(B)$. So we may assume they differ on only x. However, since the $K_i \neq K_j$ the variable in the same position in K_j is not x, call it y. Thus, we can map y to a fresh constant as above. Since each time we fix a pair of subgoals, one set of keys contains a fresh constant in a given position, which is never added again, this shows that the subgoals will never again be in conflict.

Reverse Direction If v is a valuation for V then v restricted to $\mathbf{var}(\mathbf{DJA}(V))$ is a valuation for $\mathbf{DJA}(V)$. This is because the chase only equates variables. Then, it must be that $\theta(B)$ has equated these variables, but this contradicts our proposition above.

□

10 Appendix B: Representability

The goal of this section is to give a solution to problem 1 in the case where the views are described by conjunctive queries and the representation formalism is **BID**. Our main result is that testing representability is Π_2^P-Complete in the size of the view definition. In sec. 10.2, we extend our test to handle sets of views with no increase in complexity. The high complexity motivates us to give an efficient sound (but not complete) test in sec. 4.2. For the important special case when all symbols used in the view definition are distinct, we show that this test is complete as well.

10.1 Proof of Lemma. 4.1

Lemma 10.1 (Restatement of 4.1)**.** *Given a view $V^p(H)$ and $K \subseteq H$, there are no K-doubly critical tuples if and only if V^p is K-block independent.*

In this section, we give an overview of the proof of Lemma 4.1. Our proof requires an examination of the polynomials produced by queries, which we relate to a special kind of Boolean functions related to DNFs on possible worlds.

Overview of Section. In sec. 10.1.1, we prove our main technical lemma that two Boolean functions are independent as long as they do not share an **influential variable** (def. 10.3). This is similar to the result obtained in [33], but there is a major technical difference, the proof relies on the FKG inequality but that inequality does not hold here. We then translate this technical lemma into a database style test, showing that queries are not independent if there exist **doubly critical tuples** (def. 10.5). We then establish testing that doubly critical tuples do not exist is Π_2^P-Complete.

10.1.1 EDNF Formula

In this section we define **EDNF formula** which are a generalization of Boolean formula well-suited for our application.

Definition 10.1. *An **Equality Disjunctive Normal Form (EDNF) formula** on variables $X = \{x_1, \ldots, x_n\}$ with value space $\Xi = \xi_1 \times \cdots \times \xi_n$ is a syntactic formula defined by the following grammar:*

$$\texttt{EQTerm} ::- \ \texttt{EQTerm} \wedge \texttt{EQTerm} \mid x_i = a_j \text{ and } a_j \in \xi_i - \{\bot_i\}$$
$$\texttt{EDNF} \quad ::- \ \texttt{EDNF} \vee \texttt{EDNF} \mid \texttt{EQTerm}.$$

where $\bot_i$ is a distinct symbol for each x_i.

An **assignment** is a tuple $\theta \in \Xi$. A formula is satisfied by an assignment in the obvious way.

Relation to Query Answering. The existence of a tuple in the output of a query can be viewed as a DNF ([37]), EDNFs are just a convenient way to think about them. To each distinct key value (e.g. k_i) in the representation, we associate a variable (e.g. x_i). The possible values of the value attributes associated with k_i is represented by ξ_i. Thus, an assignment

corresponds to a possible world, specifically $\theta_{x_i} = a$ implies that the value associated with k_i is a. Further, $\theta_{x_i} = \perp_i$ implies that no tuple with key value k_i is present under θ, which corresponds to *entity uncertainty*. Each disjoint aware valuation v for Q on a representation $\Im$ correspond to a single monomial in the EDNF. In

Example 10.1. *The EDNF formula corresponding to V_2^p('TD') in example 2.2 is*

$$(w_1 = 1 \ \wedge \ r_1 = \text{'High'}) \vee (w_1 = 1 \ \wedge \ r_2 = \text{'High'}) \vee \ (w_2 = 1 \ \wedge \ r_2 = \text{'High'})$$

where $\theta_{w_1} = 1$, implies w_1 is present.

Distributions on EDNFs. For any i, let $m_i = |\xi_i|$. A **valid value distribution** distribution on ξ_i can be described by a list of non-negative numbers $p_{i,1}, \ldots, p_{i,m_i}$ satisfying $\sum_{j=1}^{m_i-1} p_{i,j} \leq 1$. Thus, our convention is that p_{i,m_i} does not need to be specified and satisfies $p_{m_i} = \sum_{j=1}^{m_i-1} p_{i,j}$. Since we will consider only product distributions, a distribution on all of Ξ can be described by a list $\vec{p} = \{p_{1,1}, \ldots p_{1,m_1-1}, p_{2,1}, \ldots p_{n,m_n-1}\}$. We say $\vec{p}$ is **valid value space distribution** for Ξ if for each $i \in 1, \ldots n$, $\{p_{i,1}, \ldots, p_{i,m_i-1}\}$ is a *valid value distribution*. Thus the probability of an assignment θ can be computed as $\prod_{i \in 1,\ldots,n} p_{i,\theta_i}$. Given an EDNF formula ϕ and a *valid value space distribution* $\vec{p}$ we denote the probability of a satisfying assignment with $\Pr_{\vec{p}}[\phi]$.

We recall a useful fact from functional analysis.

Proposition 10.1. *Let h be a polynomial and $q \in \mathbb{N}$ in $\mathbb{R}^q$ if $h = 0$ on any open ball in then h is the zero polynomial.*

The next proposition is the motivation behind choosing our specification of probability distributions. Specifically, it is easy to see using this specification that the space of valid probability distributions contains an open ball of $\mathbb{R}^q$ for some q.

Proposition 10.2. *For a fixed value space $\Xi = \times_{i=1}^n \xi_i$, the space of valid value space distributions can be viewed as as subset of $\mathbb{R}^q$ where $q = \sum_{i=1}^n (m_i - 1)$ that contains an open ball.*

For an EDNF formula ϕ, we now describe the probability that ϕ is satisfied under distribution described by $\vec{p}$ denoted $\Pr_{\vec{p}}[\phi]$ as a polynomial in $\vec{p}$.

Proposition 10.3. *Let f be a polynomial such that $f(\vec{p}) = \Pr_{\vec{p}}[\phi]$ then three facts hold: First, f is a polynomial in $\vec{p}$ and is unique. Second, f has degree 1 in every variable. Third, $\forall i, j \neq j'$ f contains no term containing $p_{i,j} * p_{i,j'}$*

Proof. $\Pr[\phi]$ is a summation of possible world probabilities, hence a polynomial. Our fact above gives that f is unique since the space of all probabilities contains an open ball. The probability associated with each world is of degree 1 in each variable. In addition, $p_{i,j}$ and $p_{i,j'}$ for $j \neq j'$ represent disjoint events thus no probability for a possible world can contain both of these either. □

An immediate corollary is that we can write the polynomials in a particularly simply form.

Corollary 10.1. *Let f be a polynomial corresponding to $\Pr_{\vec{p}}[\phi]$ then for any variable x_i, we can write $f = f_1 p_{i,1} + f_2 p_{i,2} + \ldots f_{m-1} p_{i,m-1} + g$ where $f_1, \ldots, f_{m-1}, g$ are polynomials that do not contain $p_{i,j}$ for any j. We call this the **decomposition with respect to** x_i. Further, the decomposition is unique.*

Definition 10.2. *We say that f is **independent of** x_i if and only if its decomposition with respect to x_i the polynomials $f_1, \ldots, f_{m-1}$ satisfy $f_1 = f_2 = \cdots = f_{m-1} = 0$.*

We now show that independence in polynomials is equivalent to probabilistic independence for *every* valid value distribution.

Lemma 10.2. *Given two formulas ϕ and ψ on variables X. $\Pr_{\vec{p}}[\phi \wedge \psi] = \Pr_{\vec{p}}[\psi] \Pr_{\vec{p}}[\phi]$ for every valid value space distribution $\vec{p}$ if and only if for each x_i at least one of the polynomials corresponding to $\Pr_{\vec{p}}[\phi]$ or $\Pr_{\vec{p}}[\psi]$ is independent of x_i.*

Proof. We first observe that if $\Pr[\phi \wedge \psi] = \Pr[\psi] \Pr[\phi]$ for every distribution, then they are the same polynomial. This follows because they agree on P which contains an open ball.

Forward Direction Assume for contradiction that $\Pr_{\vec{p}}[\phi \wedge \psi] = \Pr_{\vec{p}}[\phi] \Pr_{\vec{p}}[\psi]$ for every distribution but $\Pr_{\vec{p}}[\phi]$ and $\Pr_{\vec{p}}[\psi]$ are not independent of some x_i. Consider their factorization with respect to x_i. Notice that their product will contain a non-zero term either of of degree 2 in some $p_{i,j}$ or the term $p_{i,j} p_{i,j'}$. By our previous proposition $\Pr_{\vec{p}}[\phi \wedge \psi]$ can contain no such term with a non-zero coefficient. Since their coefficients are equal, this is a contradiction.

Reverse Direction If at least one of $\Pr_{\vec{p}}[\phi]$ and $\Pr_{\vec{p}}[\psi]$ are independent of each x_i implies they can both be written entirely in terms not sharing a single variable. Thus every assignment $\vec{x}$ can be written as $(\vec{y}, \vec{z})$ where $\vec{y}$ contains all variables not independent of $\Pr_{\vec{p}}[\phi]$ and $\vec{z}$ all those variables independent of $\vec{z}$.

We break the computation in to pieces $\Pr_{\vec{p}}[\phi \wedge \psi] = \sum_{\vec{x}} \phi(x) \wedge \psi(x) = \sum_{\vec{y}} \sum_{\vec{z}} \phi(\vec{y}) * \psi(\vec{z})$. The last step follows since if the polynomials do not contain same terms, multiplication is always valid. Thus, the value of $\phi(\vec{y})$ is independent of $\vec{z}$ thus, we can rewrite this as $\sum_{\vec{y}} \phi(\vec{y})(\sum_{\vec{z}} \phi(\vec{z}))$. Since $\sum_{\vec{z}} \phi(\vec{z}) = \Pr_{\vec{p}}[\phi]$ we have $\Pr_{\vec{p}}[\phi](\sum_{\vec{y}} \phi(\vec{y})) = \Pr_{\vec{p}}[\phi] \Pr_{\vec{p}}[\psi]$ as desired. □

We give an example to show that a stronger statement, that independence with respect to a single distribution implies independence in polynomials is false. In particular, we give a single distribution such that $\Pr_{\vec{p}}[\phi \wedge \psi] = \Pr_{\vec{p}}[\psi] \Pr_{\vec{p}}[\phi]$ but the polynomials are *not* independent.

Example 10.2. *There are three random variables x, y, z taking values in $\{H, T\}$. We denote the probability that x takes value h (resp. t) as x_h (resp. x_h).*

$$\Pr[\phi] = x_h y_t + x_t y_h \text{ and } \Pr[\psi] = y_h z_t + y_t z_h$$

Thus, $\Pr[\phi \wedge \psi] = x_h y_t z_h + x_t y_h z_t$. Consider when all probabilities are equal to 0.5. Then we have: $\Pr[\psi] \Pr[\phi] = 0.25 = \Pr[\phi \wedge \psi]$ even though ϕ and ψ are both influenced by y. However, this is not a counter example to our theorem which requires

the stronger hypothessis that they should be independent with respect to every *distribution. Consider* $x_h = y_h = z_h = \frac{2}{3}$ *and* $x_t = y_t = z_t = \frac{1}{3}$*, in this case it is straightforward to check that* $\Pr[\psi]\Pr[\phi] = \frac{20}{81} \neq \frac{18}{81} = \Pr[\psi \wedge \phi]$.

10.1.2 Influential Variables

Definition 10.3. *Given an EDNF formula* ϕ *on variables* X*, we say a variable* $x \in X$ *is* ***influential*** *for* ϕ *if there exists a pair of assignments* (θ, θ') *such that* $\forall y \in Y\ y \neq x \implies \theta_y = \theta'_y$ *and* $\phi(\theta) \neq \phi(\theta')$.

We now relate the notion of independent polynomials to influential variables:

Theorem 10.1. *Given two EDNF formula* ϕ, ψ *on a common set of variables* X*.* $\Pr_{\vec{p}}[\phi \wedge \psi] = \Pr_{\vec{p}}[\phi]\Pr_{\vec{p}}[\psi]$ *for every probability distribution* $\vec{p}$ *if and only if there does not exist a variable influential for both* ψ *and* ϕ*.*

Proof. By our previous theorem, we need only show that there exists an influential variable if and only if $\Pr[\phi]$ and $\Pr[\psi]$ is not independent on x_i. We first show that a variable is influential for an arbitrary single formula ϕ if and only if $\Pr[\phi]$ is not independent of x_i. This then establishes the theorem because there is an influential variable for both ϕ and ψ if and only if there is a shared variable on which they are not independent.

Suppose x is influential for a single formula ϕ, then there are θ and θ' such that $\theta(\phi) \neq \theta'(\phi)$ but agree everywhere but x. This implies that there is at least one term in $\Pr[\phi]$ involving $p_{x,1}, \ldots, p_{x,m}$. In other words, ϕ is not independent of x. □

Example 10.3. *Consider the following EDNF formula, shown with their associated probabilistic polynomials.*

$$\begin{aligned} \phi = &\quad ((X = H) \wedge (Y = T)) \vee ((X = T) \wedge (Y = H)) &\quad x_h y_t + x_t y_h \\ \psi = &\quad ((Y = H) \wedge (Z = T)) \vee ((Y = T) \wedge (Z = H)) &\quad y_h z_t + y_t z_h \end{aligned}$$

It is straightforward that Y is influential for both formula, which correctly implies that these formula are not independent for some setting of probabilities. Since these polynomials are the same as in ex. 10.2, we have already confirmed this fact.

10.1.3 Block Independence

The main result of this section says that if two output tuples share a single tuple (t) that affect their output value then they cannot be independent.

Definition 10.4. *A tuple* t *is a* ***disjoint critical tuple*** *for* $V^p()$ *if and only if* t *is in a* probabilistic relation *and there exists possible worlds* I *satisfying* $I \models V^p()$ *and* $I - \{t\} \not\models V^p()$*.*

Definition 10.5. *A pair of tuples* (s, t) *is* K***-doubly critical*** *for a view* $V^p(H)$ *if and only if* $\exists s^o, t^o$ *both agreeing with* K *such that* $s^o \neq t^o$ *and such that* s *is* disjoint critical *for* $Q(s^o)$ *and* t *is* disjoint critical *for* $Q(t^o)$*,* $\mathbf{pred}(s) = \mathbf{pred}(t)$ *and* $key(s) = key(t)$*.*

We have shown:

Theorem 10.2. *The output of a conjunctive view $V^p(H)$ is guaranteed to be* block independent *if and only if for all possible worlds I there do not exist a doubly critical tuples s, t.*

10.2 Extension: Sets of Representable Views

In the previous section, we established a test for a view, in isolation, to be representable. However, the BID model makes the assumption that tuples in different relations/views are *independent*. In this section, we give an extension to handle when a set of views is **representable**.

Definition 10.6. *A set of views $\{V_1^p \ldots, V_m^p\}$ is **representable** if for $i \in 1, \ldots, n$ V_i^p is representable and for any set of tuples T such that each $t \in T$ appears in at most one V_i^p we have independence for every instance:*

$$\Pr[\bigwedge_{t \in T} t] = \prod_{t \in T} \Pr[t]$$

In the spirit of the previous definitions, we give the following generalization of critical tuples:

Definition 10.7. *A tuple is t **doubly critical** for a pair of views V_1^p, V_2^p if exists s^o, t^o such t is disjoint critical for $V_1^p(s^o)$ and $V_2^p(t^o)$.*

We state the following theorem for completeness, its proof is straightforward.

Theorem 10.3. *A set of views $\{V_1^p, \ldots, V_n^p\}$ is **representable** if and only if there does not exist a views V_i^p and V_j^p and a tuple t that is critical for V_i^p and V_j^p. Further, this test is Π_2^P-Complete.*

The previous Π_2^P-algorithm works in this more general setting. Additionally, the special case when the views are Boolean and contain only independent tuples has already been shown Π_2-Hard in [33].

The following corollary is immediate from Thm. 10.3.

Corollary 10.2. *The output of $V_1(H)$ and $V_2(H)$ is guaranteed to be independent if and only if for all representations $\Im$ there does not exist a tuple s, t that is disjoint critical for some h and h' $V_1(h)$ and $V_2(h')$.*

10.3 Representability Completeness: Thm. 4.1

10.3.1 Membership in Π_2^P

Lemma 10.3. *Algorithm 2 is correct*

Proof. Using Lem. 4.1, the algorithm is sound because the image of the homomorphisms is exactly the required witness. Consider I such that $I \models Q()$ and $I - \{t\} \not\models Q()$, we construct $I' = \mathbf{im}(f)$, for some homomorphism f which shows completeness. We can take f to be the homomorphism that witnesses $I \models Q()$. Since $\mathbf{im}(f) \subseteq I$, if there where a homomorphism from Q to $\mathbf{im}(f) - \{t\}$, since $\mathbf{im}(f) - \{t\} \subseteq I - \{t\}$, we would be one to $I - \{t\}$ as well, a contradiction. □

10.3.2 Hardness

We prove the hardness, the second half of Thm. 4.1 in its own section of the appendix, Sec. 11, because of its length.

10.4 Finding Possible Worlds Keys: R-Equivalence Version

In this section, we prove that the complexity of finding candidate keys does not dominate the costs of checking representability. Our algorithm is very likely to be suboptimal. In this section, we will use the attribute notation because it is slightly more natural.

Proposition 10.4. *If $V^p \models K \rightarrow_r K'$ and $V^p \models K' \rightarrow_r K$ then $V^p(K; H - K)$ and $V^p(K'; H - K')$.*

Proof. If $V^p \models K \rightarrow_r K'$ then any disjoint aware valuation that agrees on K agrees on K'. So suppose that there is some s, t such that $s[K] = t[K]$ but $s[K'] \neq t[K']$ this implies there is a disjoint aware valuation with the same property, contradicting the hypothesis that $K \equiv_r K'$. Thus, block independence follows. Also by transitivity, disjoint in blocks follows. □

Theorem 10.4. *If K is minimal as a set such that $V^p \models K \rightarrow_p H$ then K is a candidate key.*

Proof. Suppose not, assume that $V^p(K; H - K)$ is not representable but $V^p(K'; H - K')$ is. Then by proposition in the paper $V^p \models K \rightarrow_r K'$. Since only trivial functional dependencies hold in the representation, we have $\theta(K') \subseteq \theta(K)$. Since K is not representable, it must be that $V \not\models K' \rightarrow_r K$ else $K \equiv_r K'$ and the previous proposition shows that $V^p(K; H - K)$ is representable, a contradiction. Thus, the inclusion is strict, that is $\theta(K') \subset \theta(K)$. Now consider $K_0 = \theta^{-1}(K') \cap K$. Clearly $\theta(K') \subseteq \theta(K_0)$. Thus, $K_0 \rightarrow_r K'$ which implies that $K_0 \rightarrow_p H$. However, $K_0 \subset K$, since $\theta^{-1}(K') \neq K$, which contradicts K's minimality. □

Proposition 10.5. *Fix a view $V^p(H)$. If $K \subseteq H$ is block disjoint and $K' \subseteq H$ is block independent then $K \rightarrow_R K'$.*

Proof. Suppose not, then there are two tuples $s[K] = t[K]$ such that $s[K'] \neq t[K']$. This implies that s, t are disjoint since $s[K] = t[K]$ and K is block disjoint. On the other hand, s and t are not disjoint, since $s[K'] \neq t[K']$ and K' is block independent (positive suffices). □

Say that $K \equiv_R K'$ if $V^p \models K \rightarrow_R K' \wedge K' \rightarrow_R K$.

Corollary 10.3. *If $V^p(K, A)$ and $V^p(K', A')$ are representable then $K \equiv_R K'$.*

Definition 10.8. *Given $K_1, \ldots, K_n$ sets of attributes then the greatest common dependency is a set of attributes G such that $\forall i K_i \rightarrow_R G$ and if $G' \subseteq G$ with the same property $G' = G$. We denote this as $\mathtt{GCD}_R(K_1, \ldots, K_n)$*

This can be seen to be defined as $G = \bigcap_{i=1}^{n} K_i^*$.

Proposition 10.6. *If $K_1, \ldots, K_n$ are block disjoint then $K' \subseteq H$ is block independent implies that $K' \subseteq \mathtt{GCD}_R(K_1, \ldots, K_n)$*

Proposition 10.7 (Completness of GCD)**.** *For a fixed $V^p(H)$ let $\mathcal{K} = \{K \mid K \rightarrow_p H\}$ and $K_\infty = \mathtt{GCD}_R(\mathcal{K})$ then $\exists K$ such that $V^p(K, H - K)$ is representable iff $V^p(K_\infty, H - K)$ is.*

Proof. The reverse direction is trivial $K = K_\infty$. Suppose that $V^p(K, H - K)$ is representable then $K \rightarrow_R K_\infty$ by definition. If $K_\infty \rightarrow_R K$ then we're done, so suppose not. Suppose $K_\infty \nrightarrow_R K$, but $\forall i K_i \rightarrow_R K$, then $K \subseteq K_\infty$, a contradiction. So we may assume that for some $K' \in \mathcal{K}$ we have $K' \nrightarrow_R K$ and $K' \rightarrow_p H$. However, since K is assumed to be independent (positive), this is a contradiction to prop. 10.5 □

10.5 A Δ_2^p Algorithm for GCD

We give an $P^{\mathtt{NP}}$ algorithm for GCD. The problem is that $\mathcal{K}$, can contain exponentially many possible world keys. We want a running time that does not depend on the number of $|\mathcal{K}|$ but only on V. Specifically, we get a Δ_2^P algorithm in terms of the number of the attributes in the head of V denoted H. The algorithm makes the following call:

Oracle Question. Given a set of attributes $\vec{C} = \{C_1, \ldots, C_m\}$ and a query $V^p(H)$, return a subset $\vec{K}$ satisfying:

1. $V^p \not\models \vec{K} \nrightarrow_R \vec{C}$
2. $V^p \models \vec{K} \rightarrow_p H$

The guess is simply $\vec{K}$. Both properties can be verified in PTIME, which implies that the guess is in NP. However, if no such B exists, the algorithm returns "NO". This subroutine is in $P^{\mathtt{NP}} = \Delta_2^P$.

Algorithm. Let $C_0 = H$. Let K_{i+1} be the result of calling the above routine on input C_i. Let $C_{i+1} = K_i^*$ (the R-closure). Return K_i such that the above routine returns "No" on C_{i+1}. We will denote this step by K_∞ and show that it converges.

Convergence. We observe that $C_{i+1} \subset C_i$ so the process can only run for a finite number of steps. Since, $C_{i+1} = K_i^*$, this implies it is a strict subset of C_i (property 1 above). Thus, the algorithm runs for at most n of these steps.

Correctness. Clearly $\mathtt{GCD}_R(K_1, \ldots, K_n) \subseteq K_\infty$ since K_∞ is the intersection of a subset of the K_i. Suppose that K_∞ is a proper super set, this implies there is some K such that $K \nrightarrow_R K_\infty$. This K would be returned by the subroutine, hence $K_\infty = \mathtt{GCD}_R(K_1, \ldots, K_n)$

Relationship. This shows that our algorithm is in Π_2^P because we can run this algorithm to get K which is runnable by our more powerful $\mathtt{coNP}^{\mathtt{NP}}$ machine to get K, then run the standard machine. Alternately, it can be seen that this is closure under intersection as Π_2^P languages.

10.6 Practical Algorithm (Proof of Thm. 4.5)

Theorem 10.5 (Restatement of Thm. 4.5)**.** *For a view $V^p(H)$ and $K \subseteq H$, if algorithm 6 outputs* `'Yes'` *then V^p is guaranteed to be K-block independent. Further, if V^p does not contain repeated probabilistic subgoals then algorithm 6 is complete. The algorithm is* `PTIME`*.*

Proof. Suppose there is no collision, but must be tuples s, t that agree on some K and are critical, but since they are critical there must disjoint aware be valuations whose image contains each of s and t, these valuations are collisions. If there are no repeated subgoals, then every tuple in the image of a valuation is critical, which shows completeness. Since the algorithm must only check each pair of goals, it is bounded by n^2 calls to the **DJA** algorithm and is efficient. □

10.7 Remark 4.1: Weaker probabilistic remain hard

Checking that all tuples that disagree on a set of attributes, seems like a strong condition. One may expect that a weaker probabilistic requirement would be easier to check. For example, if our representation is block independent on K, we know that if $s[K] \neq t[K]$ then s and t must be independent. We could imagine checking something much weaker: If $s[K] \neq t[K]$, can we promise that s, t are *not* disjoint? We state this problem formally and show that even checking this much weaker property remains Π_2^P Complete.

Definition 10.9. *We say that V is K-positive given a view $V^p(H)$ and $K \subseteq H$, $\forall s, t \in V$ such that $s[K] \neq t[K]$ and any BID representation $\Im$ does there exist a possible world I such that $I \models_D Q(s) \wedge Q(t)$? The* positivity problem *is given a view $V^p(H)$ and $K \subseteq H$ decide if V is K-positive.*

Proposition 10.8. *The* positivity problem *is Π_2^P-Hard.*

Sketch. We omit some formal details but explain the gist of the reduction. Consider a $\forall\exists$ 3CNF formula: $\forall x_1, \ldots, x_n \exists y_1, \ldots, y_m \phi(\vec{x}, \vec{y})$.

Let $X_1, \ldots, X_n$ be query variables corresponding to the variables $x_1, \ldots, x_n$. Let $Y_1, \ldots, Y_m$ be query variables corresponding to $y_1, \ldots, y_m$. The schema consists of four BID relations $\mathtt{R}_0(\emptyset; A_1, A_2, A_3; \mathbf{P}), \mathtt{R}_1(A_1; A_2, A_3; \mathbf{P}), \mathtt{R}_2(A_1, A_2; A_3; \mathbf{P}), \mathtt{R}_3(A_1A_2, A_3; \emptyset; \mathbf{P})$. The cardinality of the possible worlds key and the subscript of a relation is the same. The head of the query is $V^p(X_1, \ldots, X_n, Z)$ where Z is a fresh variable and $K = \{X_1, \ldots, X_n, Z\}$. The dummy variable ensures will ensure that all assignments are considered.

For each clause (say i) e.g. $X_1 \vee \bar{Y}_2 \vee Y_3$ create a term R_i where i is the number of positive terms in the clause. The possible worlds key are all positive terms and the value attributes are all negative terms (e.g. $R_2(X_1, Y_3 \ ; \ Y_2)$.

Consider any input tuple pair, if $s[X_1] = t[X_1]$ then consider X_1 false and true otherwise. Abusing notation, I will denote the valuations and the tuples identically. Thus, it makes sense to write $t[Y_1]$ even though t does not appear in the head. With this notation, we will say that Y_1 is true if $s[Y_1] = t[Y_1]$.

The claim is that are consistent assignments to the Y for any set of X iff and only if the formula is satisfied. Consider any term: $R_2(X_1, Y_3 \ ; \ Y_2)$ if $s[X_1] = t[X_1]$ and $s[Y_3] = t[Y_3]$ but $s[Y_2] \neq t[Y_2]$ then the valuations are not compatible and hence cannot form a possible world. Further in any consistent world both terms are satisfied. Thus, they can be compatible if and only if the valuations satisfy each term and hence if the formula is satisfied. □

Remark 10.1. *This proposition is interesting because we need to use the disjoint events in the BID representation, else the problem is trivial. This is in contrast to the hardness discussed in the next section.*

Remark 10.2. *In Prop. 10.5, we only need positivity - not independence.*

11 Appendix C: Hardness Proof

We prove the second half of Thm. 4.1, the Π_2^P Hardness. We consider a tuple independent representation (i.e. all the attributes of each relation form a possible worlds key) and conjunctive queries, from which hardness of the BID situation follows. If all tables contain only independent tuples, it is immediate from Def. 4.2 that there must be a *single* tuple that is doubly critical for two distinct output tuples.

We reduce from ∀∃3CNF using a construction similar to [24]. Given a ∀∃3CNF formula, $\forall \vec{x}\ \forall \vec{y}\ \phi(\vec{x}, \vec{y})$, we produce a pair $(V(h), T)$ where $V(h)$ is a view and T is a subset of subgoals of V. We then show that ϕ holds if and only if for all homomorphisms f for $V(h)$ and t satisfying $f^{-1}(t) \subseteq T$, $\mathbf{im}(f) - \{t\} \models V(f(h))$. To complete the proof, we then show that the view $V(h)$ produced in the pair $(V(h), T)$ has a doubly critical tuple t if and only if there is a homomorphism f such that $f^{-1}(t) \subseteq T$ and $\mathbf{im}(f) - \{t\} \models V(f(h))$. Thus, finding a doubly critical is at least as hard as deciding satisfaction for ∀∃3CNF.

11.1 Setups

We shall construct the view V in stages below and select the set T.

Definition 11.1. *A triple (f, T, t) where f is a homomorphism for V, T is a set of subgoals and t is a tuple satisfying $f^{-1}(t) \subseteq T$ is called a **setup**. We say that a setup is **unsat** if $\mathbf{im}(f) - \{t\} \not\models V(f(h))$ and **sat** otherwise.*

We show that we can restrict the class of setups we need to consider, to **fine** setups. Essentially, in a fine setup variables not in $\mathbf{var}(f^{-1}(t))$ are mapped to distinct constants.

Definition 11.2. *Given a setup (f, T, t), let $G_t = \{g_1, \ldots, g_n\} = f^{-1}(t)$ be the set of subgoals mapped by f to t. We say that (f, T, t) is **fine** if $x \in \mathbf{var}(V) - \mathbf{var}(G_t), y \in \mathbf{var}(V)$ then $x \neq y$ implies $f(x) \neq f(y)$.*

Proposition 11.1. *If there is a setup that is unsat, there is a fine setup that is unsat.*

Proof. Consider an unsat setup (f, T, t) that is not fine. We construct a setup (g, T, t) that is fine, and by way of contradiction, we can assume that any fine setup is sat. This implies there is a homomorphism g' from V to $\mathbf{im}(g) - \{t\} = J$. We exhibit a homomorphism h such that $g' \circ h$ is a homomorphism of V to $\mathbf{im}(f) - \{t\}$, a contradiction proving the claim.

First, we define the fine setup, let $g(x) = f(x)$ if $x \in \mathbf{var}(G_t)$ otherwise let $g(x)$ map to a distinct constant. (g, T, t) is a fine setup by construction. Every value in the active domain of $\mathbf{im}(g)$ can be described as $g(x)$, so we $h(g(x)) = f(x)$. This mapping is well-defined: if $g^{-1}(x)$ contains two distinct variables x, y, it must be that $x, y \in \mathbf{var}(G_t)$ hence $g(x) = f(x) = g(y) = f(y)$ and there is no ambiguity. It is a homomorphism from $\mathbf{im}(g) \to \mathbf{im}(f)$. We show that $h^{-1}(t) = t$. First, $h^{-1}(t) \neq \emptyset$ because $h(t) = t$. So suppose t' is such that $h(t') = t$. Since t' is the image of some subgoal g_i, $g(g_i) = t'$ and since $h(t') = t$ we can conclude that $f(g_i) = t$, this implies $g_i \in G_t$ hence is mapped to $t' = f(g_i) = t$, a contradiction.

□

Prop. 11.1 allows us to consider only fine setups, without loss of generality.

11.2 Construction of Subgoals

We define $V^p(h)$ by describing a procedure to construct its subgoals given $\forall x_1, \ldots, x_n \; \exists y_1, \ldots, y_m \; \phi(\vec{x}, \vec{y})$. As shown, V^p has a single variable h as its head. We argue about f^n a (partial) homomorphism from V to $\mathbf{im}(f) - \{t\}$. We shall use the notation $z \mapsto f(z^b)$ as a shorthand to mean, in any homomorphism f^n from V to $\mathbf{im}(f) - \{t\}$ such that $f^n(h) = f(h)$, it is the case that $f^n(z) = f(x)$ (it is forced).

***X*-Goals.** In our problem definition, let $T = \{t_z, t_1, \ldots, t_n\}$ defined below.

- $t_z = \mathtt{R}(z, e_z, e_z, -)$ (Special)
- $t_1 = \mathtt{R}(x_1, d_1, e_1, -), \ldots, t_n = \mathtt{R}(x_n, d_n, e_n, -)$ (*X*-Variables)
- $t_z^b = \mathtt{R}(z^b, e_z^b, e_z^b, h), t_1^b = \mathtt{R}(x_1^b, d_1^b, e_1^b, h), \ldots, t_n^b = \mathtt{R}(x_n^b, d_n^b, e_n^b, h).$ (Backups)

Definition 11.3 (A Good setup)**.** *We call a setup (f, T, t) **good** if $t = f(t_z)$ and **bad** otherwise.*

Special Variables. For the special subgoals we create the following subgoals:

$$\mathtt{R}_z(z, h), \mathtt{R}_z(z^b, h), \mathtt{R}_{z^b}(z^b, h)$$

The effect of this is to restrict the range in the following way in any possible homomoprhism f^n from V into $\mathbf{im}(f) - \{t\}$: $f^n(z) \in \{f(z), f(z^b)\}$, $f^n(z^b) = f(z^b)$. We call variables with subscripted b *backup* variables and enfoce that $v^b \mapsto f(v^b)$.

X-Variables. For each variable x_i we create a separate set of subgoals, with new symbols $\mathtt{R}_i$ and $\mathtt{R}_{i^b}$ described below:

$$\mathtt{R}_i(x_i, z, h), \mathtt{R}_i(x_i, z^b, h), \mathtt{R}_i(x_i^b, z^b, h), \mathtt{R}_i(x_i^b, z^b, h) \text{ and } \mathtt{R}_{i^b}(x_i^b, z, h), \mathtt{R}_{i^b}(x_i^b, z^b, h)$$

We can specify these goals more succinclty as $\mathtt{R}_i(x_i \mid x_i^b, z \mid z^b, h)$ and $\mathtt{R}_{i^b}(x_i^b, z \mid z^b)$.

***Y*-Goals.** For each $i \in 1 \ldots, m$ $\mathtt{Y}_i(z, y_i, h), \mathtt{Y}_i(z \mid z^b, y_i^t, h), \mathtt{Y}_i(z \mid z^b, y_1^f, h)$. The intuition is that these subgoals will ensure that y_i takes exactly one value y_i^t (true) or y_i^f on any *good* instance.

We shall show:

Proposition 11.2. *Given a setup (f, T, t) a homomorphism f corresponds (surjectively) to an assignment for the universally quantified variables, $\vec{x}$. A **good** setup for V is **sat** if and only if there exists an assignment to the $\vec{y}$ such that $\phi(\vec{x}, \vec{y})$ with is satisfied.*

If a setup is bad, then we show it is either sat or there is some good setup that is unsat. The argument requires examining the subgoals produced in the reduction.

11.2.1 Properties of fine setups

Proposition 11.3. *The following hold in any **fine** setup (f, T, t).*

1. $f(x_i) = f(z) \implies f(t_i) = f(t_z)$
2. $\forall i, j\ i \neq j \implies f(y_i^t) \neq f(y_j^f)$
3. *All backup subgoals (e.g. v^b) are mapped to distinct constants by f.*
4. $\forall i \in \{1, \ldots, n\}\ t \neq f(t_x) \implies f(d_i) \neq f(e_i)$.

11.2.2 Properties of good setups

Given a good setup (f, T, t), we say that a variable x_i is *false* if $f(t_i) = t$ and *true* otherwise. Consider any assignment to the $\vec{x}$, let F be the set of variables that are false. We capture this assignment by setting $\{t_z\} \cup \bigcup_{f \in F} t_f = f^{-1}(t)$ and all other variables to distinct constants. Since there are no constants in $t_z, t_1, \ldots, t_z$, such a unification always succeeds.

Proposition 11.4. *If (f, T, t) is good then*

- $z \mapsto f(z^b)$ *and* $f^n(y_j) \in \{f(y_j^t), f(y_j^f)\}$ *for any* j
- *And if X_i is false then* $x_i \mapsto f(x_i^b)$, $e_i \mapsto f(e_i^b)$, $d_i \mapsto f(d_i^b)$ *and* $f(x_i) = f(z)$, $f(e_i) = f(d_i) = f(e_z)$.
- *And if X_i is true then* $f^n(x_i, e_i, d_i) = \{(x_i, e_i, d_i), (x_i^b, e_i^b, d_i^b)\}$

Proof. If $t_z \in f^{-1}(t)$ then $f^n(z) = f(z^b)$ since the range restriction implies it must map to $f(z)$, which is no longer possible in $I - \{t\}$, or $f(z^b)$. This implies that y must map to one of these because of the Y-Goals. The second item follows because in a good setup $f(t_z) = f(t_u)$. The third item is because $t_i \in f^{-1}(t)$ implies we must map t_i to t_i^b, the rest follows. In particular, it must be that $f(t_i) = f(z) = t$. The fourth item is because although we must map t_i to either $f(t_i)$ or $f(t_i^b)$. □

11.2.3 Bad Setups

Proposition 11.5. *In a **bad** setup, there is a partial homomorphism such that $f^n(z) = f(z)$. Further, we can extend it so $i = 1, \ldots, m$ $f^n(y_i) = f(y_i)$ and $f^n(y_i^t) = f^n(y_i^f) = f(y_i)$.*

Proof. We take $f^n(t_i) = f(t_i)$ if $f(t_i) \neq t$ and $f^n(t_i) = f^n(t_i^b)$ otherwise. □

11.3 Triggers

We introduce *triggers*, which are subgoals that form a gadget that encodes when a formula involving a universal quantified variable is satisfied. We state their properties formally below:

Definition 11.4. *A **trigger** for a formula X is a pair of variables (a, a^b) with the following properties:*

1. *if the setup is good and X is true any homomorphism can be extended to the trigger subgoals such that $f^n(a) = f(a^b)$ and all extended homomorphisms must satisfy this property*

2. *If the setup is good and X is false then any homomorphism can be extended so that $f^n(a) = f(a)$ (may take other values)*

3. *If the setup is bad then there exists a partial homomorphism that takes the value $f^n(z) = f(z)$ and $f^n(a) = f(a^b)$ when X is true and $f^n(a) = f(a)$ when false. (There may be others).*

Notice that when X is false, we cannot force $f^n(a) = f(a)$. In spite of this, we are able to use triggers to encode formula as we will show in the next section. We will prove the following property:

Proposition 11.6. *For any conjunctive formula of using only 3 or fewer $\vec{x}$ variables (counting repetition), there is an efficiently constructable (`PTIME`), trigger with a constant number of subgoals.*

The number three is chosen because we are dealing with 3CNF formula. This proposition is proved below by showing that we can construct $\neg x$, x for base variables and combine them using $\wedge$ and $\vee$. We prove this proposition by construction.

11.3.1 Trigger for $\neg x$

We could hope that x by itself is a trigger, but the problem is when f is bad then we cannot fulfill the last condition, x_i may not be able to be mapped to $f(x_i)$. Thus, we need to do a little bit more work. To simplify notation instead, we denote x_i as x. Let `N` be a fresh symbol. Let a and a^b be fresh variables associated with this trigger. Let s be fresh variable not used in other parts of the construction. The last three columns show where each subgoal is mapped to aid in verifying the proofs, which contain homomorphisms given by tables.

Subgoal	Description	Good (X false)	Good (X true)	Bad
(1)	$\texttt{N}(a, x, z, s, s, h)$	(2)	(3)	(1)
(2)	$\texttt{N}(a^b, x^b, z^b \mid z, x, z, h)$	(4)	(4)	(4)
(3)	$\texttt{N}(a, x, z^b, s, s, h)$	(4)	(3)	(3)
closures				
(4)	$\texttt{N}(a^b, x^b, z \mid z^b, x \mid x^b, z \mid z^b, h)$			

Line (4) specifies a set of 8 subgoals (some redundant); one for each choice of alternation. We call them closures because they are the subgoals under the closure of the homomorphisms below.

Proposition 11.7. *If X is false and good then there $a \mapsto f(a^b)$ (is forced).*

Proof. We examine where tuple (1) can map: it cannot map to (1) (since $z \mapsto f(z^b)$). (1) it can map to (2). It cannot go to (3) because x is forced to $f(x^b)$ (not x). It does not matter if it maps to a (4) closure because this forces, a to a^b. To prove the proposition, we now need to exhibit a homomorphism such that $f^n(a) = f(a^b)$. Recall that $f(x) = f(z)$. All backup subgoals are mapped by identity, and the rest are described by this table:

x	$f(x)$
x	x^b
a	a^b
z	z^b
s	$f(x) = f(z)$

□

Proposition 11.8. *If X is true, then there is a homomorphism such that $f^n(a) = f(a)$.*

Proof. All backup subgoals (e.g. x^b) are fixed, we specify the others.

x	$f(x)$
x	x
a	a
z	z^b
s	s

□

Proposition 11.9. *If the setup is bad, then there is a homomorphism such that $f^n(a) = f(a)$.*

Proof. This extends the allowable homomorphisms.

x	X true (x false)	X false (x true)
x	x^b	x
a	a^b	a
z	z	z
s	s	s

□

We have now shown that the trigger specified above satisfies the trigger properties.

11.3.2 Trigger for x

Let `N` be a fresh symbol and a, a^b be fresh variables, associated to this trigger and s, t be fresh variables not used again.

	Description					Good (False)	Good (True)	Bad
(1)	$N(a,$	$z,$	$s,$	$s,$	$h)$	(2)	(4)	(1)
(2)	$N(a,$	$z^b,$	$x,$	$z,$	$h)$	(*)	(*)	(*)
(3)	$N(a^b,$	$z^b,$	$x,$	$z,$	$h)$	(*)	(*)	(*)
(4)	$N(a^b,$	$z^b,$	$s,$	$s,$	$h)$	(3)	(4)	(4)
			…Closures…					
(*)	$N(a \mid a^b,$	$z \mid z^b,$	$x \mid x^b,$	$z \mid z^b,$	$h)$			

Proposition 11.10. *If X is false, then there is a homomorphism such that $f^n(a) = f(a)$.*

Proof. In this case, $f(x) = f(z)$ and $f(e_x) = f(d_x)$.

x	$f^n(x)$
x	$f(x^b)$
a	$f(a)$
z	$f(z^b)$
s	$f(x) = f(z)$
t	$f(e_x) = f(d_x)$
d_d	$f(d_d^b)$
e_d	$f(e_d^b)$

□

Proposition 11.11. *If X is true, then there there is a homomoprhism such that $f^n(a) = f(a)$ and any homomorphism satisfies this condition.*

Proof. We examine where tuple (1) can map, and show it must be that $a \mapsto f(a^b)$. It cannot map to (2) or (3) because $f(x) \neq f(z)$ in this case. It can map to (4). It cannot map to any closure because they all contain e_d^b, d_d^b and so we cannot map t, t.

x	$f^n(x)$
x	x
a	a^b
z	z^b
s	s
d_d	$f(d_d^b)$
e_d	$f(e_d^b)$

□

Proposition 11.12. *If setup is bad, then there is a homomorphism $f(a) = f^n(a)$.*

Proof.

x	X false	X true $f^n(x)$
x	x^b	x
a	a	a^b
z	z	z
s	s	s
t	t	t

□

We observe that in each case on all shared variables are mapped identically. Further, which is allowable in each case.

11.3.3 Trigger for $a \wedge b$

Let N be a fresh symbol and (a, a^b) and (b, b^b) be a pair of trigger variables, then a trigger variable for the conjunction, (c, c^b), is:

$$\mathtt{N}(c, a, b, h), \mathtt{N}(c^b, a^b, b^b, h), \mathtt{N}(c, a, b^b, h), \mathtt{N}(c, a^b, b, h)$$

The correctness of this trigger follows directly from the trigger properties of (a, a^b) and (b, b^b).

11.3.4 Extension property

Loosely speaking triggers behave like the assignment x_i = false iff $t_i \in f^{-1}(t)$. Summarizing, what we have shown about about the bad case:

Proposition 11.13. *Given any bad setup (f, T, t), there exist a good setup (g, T, t) such that $g^{-1}(t) = \{t_z\} \cup f^{-1}(t)$ such that if g is sat with homomorphism g^n, then there is a partial homomorphism f^n that agrees with g^n on all trigger variables but is undefined on the remaining variables: $y_1, y_1^t, y_1^f \ldots, y_m, y_m^t, y_m^f$.*

Proof. We can create g by simply enforcing that $g(t_z) = t$. We have seen that there is a partial homomorphism for a bad setup, f^n, that agrees on all trigger variables with g^n. To see this, observe that if $t_i \in g^{-1}$, then under a bad setup any trigger (a, a^b) for $\neg x_i$ will have $a \mapsto a^b$ and any trigger (b, b^b) for x_i can be set to b. It is also easy to see that both homomorphism could set this other trigger to b^b as well. □

11.4 Writing the Full CNF

We can now use our trigger variables to wire up any conjunct involving on x assignments we need. To get the full CNF clauses, we can write any CNF clause as $b(x) \implies y_1 \vee \bar{y}_2$ where b is some Boolean conjunction of $\vec{x}$. We create a fresh trigger variable using the construction of the previous variables pair, call it (b, b^b). We add then add the following, where `N` is a fresh symbol for each clause. We illustrate for example, the generalization is straightforward.

$\mathsf{C}_k(z, b, y_1, y_2, h)$,		Call this a Key tuple
If the trigger is false	If the trigger is true	
$\mathsf{C}_k(z \mid z^b, b, y_1^t, y_2^t, h)$,	$\mathsf{C}_k(z \mid z^b, b^b, y_1^t, y_2^t, h)$,	conclusion true
$\mathsf{C}_k(z \mid z^b, b, y_1^t, y_2^f, h)$,	$\mathsf{C}_k(z \mid z^b, b^b, y_1^t, y_2^f, h)$,	conclusion true
$\mathsf{C}_k(z \mid z^b, b, y_1^f, y_2^f, h)$,	$\mathsf{C}_k(z \mid z^b, b^b, y_1^f, y_2^f, h)$,	conclusion true
$\mathsf{C}_k(z \mid z^b, b, y_1^f, y_2^t, h)$		conclusion false, this case b cannot be forced to b^b

Non-Key Tuples. All non-key tuples are closed with the exception of the last tuple when $b \mapsto b^b$. We have shown that in a good setup, y_i must be mapped consistently by any homomorphism and so this implies that b must be false else the implication fails.

Key tuple. Notice that if the conclusion is true, then any assignment to the trigger will do. If the conclusion is false, then it must be that the trigger is false as well. Thus, in any good assignment, we can set the assignments to the ys can be consistently if and only if there exist a way to satisfy $\exists \vec{y}\ \vec{x}_f \phi(\vec{x}_f, \vec{y})$. Since none of these tuples can unify with t, any valid partial homomorphism can be extended by sending y to y^t or y^f is valid. There are no restrictions on where to map $f(y_i^t)$ so they can be mapped by identity. Summarizing,

Proposition 11.14. *Given a setup (f, T, t), let $\vec{x}_f$ be the assignment to the $\vec{x}$ corresponding to f, there is a homomorphism of*

V into $\mathbf{im}(f) - \{t\}$ *if and only if* $\exists \vec{y}\, \phi(\vec{x}_f, \vec{y})$ *is satisfiable.*

Thus, if for all setups (f, T, t) $\mathbf{im}(f) - \{t\} \models V^p(f(\vec{h})$ then $\forall \vec{x}\, \exists \vec{y}\, \phi(\vec{x}, \vec{y})$ is true.

11.5 Completing the Reduction

We have now seen all the subgoals, so we can now prove that if there is a bad unsat setup, then there is a good unsat setup, which completes the reduction. The intuition is that, when a setup is bad, it mimics a good setup. If the bad setup is unsat, then the corresponding good setup should be unsat as well. The technical reason this works is because we were careful that whenever we used z^b in the remaining subgoals, we created a subgoal that used z in the same position.

Proposition 11.15. *If the partial homomorphism* f^n *from Prop. 11.13 cannot be extended, then there is an unsat good setup.*

Proof. Given a bad setup (f, T, t) and its partial homomorphism f^n, consider a good setup (g, T, t) which is formed by equating variables so that $t = g(t_d)$. This has the effect of equating constants in trigger subgoals (e.g. z and x) and equating constants in R. However, by our previous proposition we already have a suitable homomorphims into these subgoals. We assume for contradiction that all good sats are setup, there is some homomorphism g^n from V to $\mathbf{im}(g) - \{t\} = J$. We use g^n as a guide to extend f^n to the remaining subgoals.

Let $\mathcal{G}_R = \{Y_1, \ldots, Y_m, C_1, \ldots, C_l\}$, the set of remaining subgoals, and $\mathcal{H}$ be all the subgoals of V except $\mathcal{G}$ (set minus). By inspection, we can see $\mathcal{G}_R$ and $\mathcal{H}$ form a partition of subgoals and that no symbol appears in both sets. Further, if $\mathbf{Tr}$ is the set of trigger variables, then $\mathbf{var}(\mathcal{G}_R) \cap \mathbf{var}(\mathcal{H}) = \{z^b, z, y_1, y_1^t, y_1^f, \ldots, y_m, y_m^t, y_m^f\} \cup \mathbf{Tr}$. We observe that $f^n(\mathcal{H}) \subseteq I - \{t\}$, we now want to extend it to $\mathcal{G}_R$.

We describe h which is a (partial) homomorphism from $\mathbf{im}(g^n) \subseteq J$ when restricted to symbols $\mathbf{pred}(\mathcal{G})$. Any value in the image, $g(v)$, we let $h(g(v)) = f(v)$ unless $v = g(z^b) = f(z^b)$ in which case $h(v) = f(z)$. This mapping is well defined because, by fineness, g is injective when restricted to a variable not in $\mathbf{var}(T)$. Since $\mathbf{var}(\mathcal{G}) \cap \mathbf{var}(T) = \{z\}$, only z could potentially be mapped non-injectively but this is a singleton set so g is injective on $\mathbf{var}(\mathcal{G})$. By inspection, every subgoal in this set containing z^b, there is an subgoal with z^b replaced by z, thus the image of h is contained in $I - \{t\}$. Thus, h is a desired partial homomorphism.

We now show that we can extend f^n to $\mathbf{var}(G_R) - \{z\}$ with $h \circ g^n$, we call the result f^*: For any $v \in \mathbf{var}(V)$, let $f^*(v) = f^n(v)$ if $f^n(v)$ is defined and $(h \circ g^n)(v)$ otherwise. This mapping is well defined because the only variables on which both are defined are trigger variables or z. By Prop. 11.13, f^n and $h(g^n)$ agree on all trigger variables. And for z, $f^n(z) = f(z) = (h \circ g^n)(z)$. Thus, f^* is a homomorphism from V into $I - \{t\}$, contradicting that the bad setup is unsat and proving the claim. □

Given ϕ, we have now constructed $V(h)$ and T with the property that for any homomorphism f of $V(h)$ and t satisfying $f^{-1}(t) \subseteq T$ then $\mathbf{im}(f) - \{t\} \models V(f(h))$ (i.e. a *good unsat setup*) if and only if ϕ is satisfied.

11.6 Completing the Hardness

We now show that the query we have produced has the property that there is doubly critical tuple if and only if there is an unsat good setup. This shows that deciding if doubly critical tuples exist is at least as hard as deciding ∀∃3CNF and completes the proof.

Proposition 11.16. *There is a doubly critical tuple for V^p if and only if there exists an unsat good setup.*

Proof. Suppose there is a doubly critical tuple, t. This implies there exist homomorphisms f, g for V such that $\mathbf{im}(f) - \{t\} \not\models V(f(h))$ and $\mathbf{im}(g) - \{t\} \not\models V(g(h))$ and $f(h) \neq g(h)$. Now suppose that $t' \in f^{-1}(t) \cap g^{-1}(t)$ and t' is not the image of a R subgoal. Consider any g_i and g_j such that $f(g_i) = g(g_j) = t$. However, every non-R subgoal has h in the last position hence $f(h) = g(h)$, a contradiction. All R subgoals that are not a subset of T also have h in their last position. Hence, it must be that $T \subseteq f^{-1}(t)$ or $T \subseteq g^{-1}(t)$. Without loss, we assume that f satisfies $T \subseteq f^{-1}(t)$. Then (f, T, t) is a setup; it must be fine because only changed the mappings of t_u; it must be good because we have shown that any bad setup satisfies $\mathbf{im}(f) - \{t\} \models V(f(h))$, which this setup does not. This shows the forward direction.

We show something stronger in the reverse direction: If $I = \mathbf{im}(f)$ for some homomorphism f of $V^p(\vec{h})$ such that for some t critical for $V(f(\vec{h}))$ and $f^{-1}(t) \cap \vec{h} = \emptyset$, then we will show that t is doubly critical for V. Notice that $f^{-1}(t) \subseteq T$ implies this condition is met. This will complete the reverse direction because it will show that the tuple t in good unsat setup is doubly critical. We define an instance J that is the image of a homomorphism g by setting $g(x) = f(x)$ if $x \notin \vec{h}$ and equal to a fresh constant otherwise. Let $\mathbf{im}(g) = J$. We define a l homomorphism from J to I. We define $l(g(x)) = f(x)$, l, this mapping is well-defined because we mapped each h to distinct constants. This is also clearly a homomorphism, the claim is that this is a homomorphism $J - \{t\}$ to $I - \{t\}$. It is clear that $l(t) = t$, but suppose that $l(t') = t$ for some $t' \neq t$. Then this is the image of some subgoal, and by assumption this subgoal does not contain h, which means that l is identity on it. Thus $t' = t$, a contradiction. □

12 Appendix D: Using Views

We consider two variants of using views to answer queries: The *definition-less* variant, which uses the partial representation to capture some but perhaps not all, of the correlations in the view and The *with-definition* variant, in which the queries are materialized but we retain definitions. In both cases, we have not kept track of lineage and so must deduce if there is a partial representation that captures enough correlations to be uniquely defined. In the *definition-less* variant, we must check that the supplied representation suffices. In the *with-definition* variant, we must determine if such a partial representation exists.

12.1 Preliminaries

We prove the reverse direction of Lem. 5.1.

Lemma 12.1. *If there is not a critical intertwined pair of tuples for a conjunctive query Q then* $\Pr[Q]$ *is uniquely determined.*

Proof. Consider the smallest BID representation that is a counterexample $\Im$, where size is the number of *uncertain* tuples in $\Im$.

For any s, t let $\mathbf{1}_s$ (resp. $\mathbf{1}_t$) be indicator random variables for s, t. Define Δ_x for each $X \in \{\{st\}, \{s\}, \{t\}\}$ $\Delta_x = (I + X \models Q) - (I - X \models Q)$.

$$\Pr[Q] = \mathbf{E}[(\mathbf{1}_s \wedge \mathbf{1}_t)(\Delta_{st} - (\Delta_s + \Delta_t) + \mathbf{1}_s\Delta_s + \mathbf{1}_t\Delta_t + (I - \{s,t\} \models Q)]$$

We observe that all of the Δ terms have fewer uncertain tuples and thus, they are uniquely determined. Also the $\mathbf{E}[\mathbf{1}_s], \mathbf{E}[\mathbf{1}_t$ terms are fixed because these are marginal probabilities of individual tuples. If s, t are not intertwined then $\mathbf{E}[\mathbf{1}_s \wedge \mathbf{1}_t]$ is uniquely defined, so it must be that this term is not uniquely defined and so, s, t are an intertwined pair. Thus, our contradiction assumption also tells us that s, t are not pair critical (else we would have a pair critical intertwined tuple pair). We will show that its coefficient must be 0. Thus, the entire expression is uniquely determined, a contradiction.

Since $\mathbf{E}[\mathbf{1}_s \wedge \mathbf{1}_t]$ and its coefficient are both non identically zero 0, this implies there is a possible world I containing s, t such that $\Delta_{st} \neq \Delta_s + \Delta_t$. On I, Δ_x is Boolean valued, since Q is conjunctive (monotone) thus $\Delta_s = 1 \vee \Delta_t = 1 \implies \Delta_{st} = 1$. Thus because of the inequality either, $\Delta_t = 1 = \Delta_s$ or $\Delta_s = \Delta_t = 0$. In other words, I is an instance such that $Q(I - \{s,t\}) \neq Q(I)$ and $Q(I - \{s\}) = Q(I - \{t\})$, i.e. the intertwined pair s, t is critical for Q. This is a contradiction, since we assumed no such pairs existed. □

12.2 Hardness of Partial Representation: Thm. 5.2

Definition 12.1. *Given a query Q using a predicate* R *and tuple t,. Call t* ***obviously critical*** *for Q if there exists a valuation v such that every* R *subgoal unifies with t.*

Proposition 12.1. *Checking if Q, t is critical but not obviously critical is Π_2^p-Complete.*

Proof. The unification test can be done in PTIME. Consider any that extends the unification, then in its image the extent of R is exactly t, hence removing t causes Q to become unsatisfied. The reduction simply checks this fact, and then can pass the problem verbatim to the oracle. □

Proposition 12.2. *Checking if Q is uniquely defined on a partial representation with only a single view symbol is Π_2^p Hard.*

Proof. Let $Q()$ be an arbitrary query on a tuple independent database and t a tuple. We reduce from the problem of finding if t is not obviously critical for Q. Let the predicate of t be $\mathtt{R}(K)$, create a new symbol, $\mathtt{R}'(K; Z; \emptyset)$, that is partially represented with $D = \{Z\}$. Let z be a fresh variable, replace each occurrence of R with R′ filling in the additional Z attribute with the z variable. Let Q' be the query Q with a single additional subgoal $\star\ \mathtt{R}'(t; \text{`a'}; )$ for some fresh constant 'a'. By construction, the only possible intertwined tuples in the image of $Q'$ are $(t, \text{`a'})$ and the image of (t, z) (there are none if z maps to 'a'). Clearly, $(t, \text{`a'})$ is critical so our claim is that for some distinct value $(t, \text{`b'})$ is critical iff t is critical.

Let I be the instance that witnesses Q and t are critical, we construct I' that witnesses the intertwined pair critical. We will denote valuations for Q without ticks (e.g. v) and valuations for Q' with ticks (v'). We may assume without loss that $\mathbf{im}(v) = I$ for some valuation v. Let $v'(z) =$ 'b' and extended in the obvious way so that $I' = \mathbf{im}(v')$. Now, suppose that $I - \{(t, \text{`b'})\} \models Q()$ with some valuation w'. If $w(z) =$ 'a', this implies that *every* subgoal R can map to t because there is only one tuple in the image with $Z =$ 'a'. We have ruled out this possibility because then t would be *obviously critical* (Def. 12.1). Thus, it must be that $w'(z) =$ 'b'. Since we removed $(t, \text{`b'})$, the corresponding w satisfies $\mathbf{im}(w) \cap \{\mathtt{R}(t)\} = \emptyset$, its image does not contain t, which is a contradiction to t being critical at I. In particular, then $v(Q) \subseteq I - \{t\}$ so $I - \{t\} \models Q()$.

Now suppose there is an intertwined critical pair, let I' be the witness. Without loss, I' satisfies $\mathbf{im}(v') = I$' and so it must be that $(t, v'(z))$ is critical. Let I be the corresponding instance for Q. It is clear that $I \models Q()$ but suppose that $I - \{t\} \models Q()$ with valuation w. This implies we could send $w'(z) =$ 'a' is a valuation for Q' such that $I' - \{(t, b)\} \models Q()$, a contradiction. □

12.3 Many Partially Represented Views

It is straightforward to extend the partial representation, between each pair of views we allow specify a pair of attributes K, K' of the same arity such that if $s[K] \neq t[K]$, then s, t are independent. We illustrate its utility in a simple case.

$$V_{10}(x) \coloneq \mathtt{R}^p(x, y), \mathtt{S}^p(y) \text{ and } V(x)_{10} \coloneq \mathtt{R}^p(x, y), \mathtt{T}^p(y) \tag{10}$$

Each view is individually representable, however they are not together because of tuples in distinct views are *not* independent. Our pair is (X, X), thus tuples that disagree on x are independent. Though the same x tuples in the views that agree on x may be correlated in complex ways.

12.3.1 Hardness with definitions

Definition 12.2. *Given views with their definitions V_1 and V_2 we say that two tuples s, t are **intertwined** if there are critical tuples for V_1 and V_2 that share a possible worlds key.*

Theorem 12.1. *Deciding if a query using two view symbols is uniquely defined, in the definition full variant is Π_2^P Complete.*

Sketch. To see that Q is Π_2^P-Hard, consider the query $Q() :- V_1(), V_2()$. This query's value is uniquely defined if and only if $V_1()$ and $V_2()$ are not independent, which is Π_2^P-Hard by [24]. To see why it is in Π_2^P, we use the view of Π_2^P as a coNP machine with access to an NP oracle. For each I, an image of a homomorphism of Q, and each $s, t, u \in I$ we need to test the following implication:

$$I - \{s,t\} \models_D Q \implies \underbrace{(I - \{u\} \models_D s \vee t)}_{\text{(s,t) not intertwined}} \vee \underbrace{(I - \{s\} \models_D Q() \iff I - \{t\} \not\models_D Q())}_{(s,t) \text{ not pair critical}}$$

To test this implication, we need to make at most four queries to our NP Oracle: $I - \{s,t\} \models_D Q(), I - \{s\} \models_D Q(), I - \{t\} \models_D Q()$ and $I - \{u\} \models_D s \vee t$. This shows the algorithm is in Π_2^P. □

12.4 Practical Algorithm for Using Views: Thm. 5.3

Theorem 12.2 (Restatement of Thm. 5.3)**.** *If no intertwined collisions exist for a conjunctive query Q, then its value is uniquely defined. If the partially representable view symbol V^p is not repeated, this test is complete.*

Proof. Consider a query $Q(H)$, we show that if there exists a critical intertwined pair (s, t) for some $\vec{h}$ $Q(\vec{h})$, then there must be an intertwined collision. Hence, if there is no intertwined collision, the value of Q is uniquely defined. Let I be the instance provided by Def. 5.4. Suppose, $I - \{s\} \models Q()$. Since $I - \{s,t\} \not\models Q(h)$, the image of any valuation v that witnesses $I - \{s\} \models Q()$ must contain t. By symmetry, the image of any valuation that witnesses $I - \{t\} \models Q()$ must contain w. It is easy to see that (v, w) is compatible and hence (v, w) is an intertwined collision. If $I - \{s\} \not\models Q()$ (and hence $I - \{t\} \not\models Q()$), then there is a single valuation v which uses both s, t. Thus, (v, v) is the desired intertwined collision.

To see completeness, observe that if a query Q has compatible valuations and only a single partially represented view V^p, since $s \neq t$ the compatible valuations that witness the collision (v, w) are distinct. In particular, consider $I = \mathbf{im}(v) \cup \mathbf{im}(w)$; I is a possible world because v and w are compatible. Also, $\mathbf{im}(w) \subseteq I - \{s\}$ hence $I - \{s\} \models Q()$ and by symmetry $I - \{t\} \models Q()$. Since Q is conjunctive this implies $I \models Q()$. Since there are no repeated views,the extent of V^p in $I - \{s,t\}$ contains no tuples, thus $I - \{s,t\} \not\models Q()$. □

Theorem 12.3 (Complexity in Thm. 5.3)**.** *Finding an intertwined collision can be implemented in PTIME.*

Given $Q(\vec{h}) :- g_1, \ldots, g_n$ and a set of key variables, consider the query $QQ() :- g_1, \ldots, g_n, \eta(g_1), \ldots, \eta(g_n)$ where η is a function that is identity on $\vec{h}$ and $\mathbf{const}(V^p)$ and maps all other variables to distinct fresh variables. For each pair of key values i, j if $\mathbf{pred}(g_i) = \mathbf{pred}(g_j)$ we need to check if equating $\vec{k_i} = \vec{k_j}$ implies that for any disjoint aware valuation for QQ, it is the case that $\vec{d_i} = \vec{d_j}$ if not then fail. This procedure is just the chase, examining after chasing if in fact $\vec{d_i} = \vec{d_j}$ hold. Thus, we get soundness completeness and efficiency for free.

13 Appendix E: Schemata

In this section, we give the probabilistic schema for the experimental results in the syntax of our implementation. The only changes are for the sake of formatting.

Syntax. Schemata used in materialized view parsers. `(* *)` encloses a comment. Probabilistic relations are denoted with an asterisk (e.g. orders*). **;** separates possible world key attributes.

13.1 Northwind 1 (NW1)

```
(**********************)
(* Northwind Relations *)
(**********************)
CustomerDemographics*(CustomerTypeID;CustomerDesc)
(* FUNCTIONAL DEPENDENCY CustomerDemographics(CustomerTypeID) ->
    CustomerTypeID,CustomerDesc; *)

Region(RegionID,RegionDescription)
FUNCTIONAL DEPENDENCY Region(RegionID) -> RegionID,RegionDescription;

Employees(EmployeeID,LastName,FirstName,Title,TitleOfCourtesy,BirthDate,HireDate,
          Address,City,Region,PostalCode,Country,HomePhone,Extension,Photo,Notes,
          ReportsTo,PhotoPath)

FUNCTIONAL DEPENDENCY Employees(EmployeeID) ->
           EmployeeID,LastName,FirstName,Title,TitleOfCourtesy,
           BirthDate,HireDate,Address,City,Region,PostalCode,Country,
           HomePhone,Extension,Photo, Notes,ReportsTo,PhotoPath;

Categories(CategoryID,CategoryName,Description,Picture)
FUNCTIONAL DEPENDENCY Categories(CategoryID) ->
           CategoryID,CategoryName,Description,Picture;

Customers(CustomerID,CompanyName,ContactName,ContactTitle,Address,City,Region,
          PostalCode,Country,Phone,Fax)
FUNCTIONAL DEPENDENCY Customers(CustomerID) -> CustomerID,CompanyName,ContactName,
           ContactTitle,Address,City,Region,PostalCode,Country,Phone,Fax;

Shippers(ShipperID,CompanyName,Phone)
FUNCTIONAL DEPENDENCY Shippers(ShipperID) -> ShipperID,CompanyName,Phone;

Suppliers(SupplierID,CompanyName,ContactName,ContactTitle,Address,City,
          Region,PostalCode,Country,Phone,Fax,HomePage)
FUNCTIONAL DEPENDENCY Suppliers(SupplierID) -> SupplierID,CompanyName,ContactName,
           ContactTitle,Address,City,Region,PostalCode,Country,Phone,Fax,HomePage;

Order_Details(OrderID,ProductID,UnitPrice,Quantity,Discount)
FUNCTIONAL DEPENDENCY Order_Details(OrderID,ProductID) ->
           OrderID,ProductID,UnitPrice,Quantity,Discount;

CustomerCustomerDemo(CustomerID,CustomerTypeID)
FUNCTIONAL DEPENDENCY CustomerCustomerDemo(CustomerID,CustomerTypeID) ->
           CustomerID,CustomerTypeID;
```

```
Territories(TerritoryID,TerritoryDescription,RegionID)
FUNCTIONAL DEPENDENCY Territories(TerritoryID) ->
           TerritoryID,TerritoryDescription,RegionID;

EmployeeTerritories(EmployeeID,TerritoryID)
FUNCTIONAL DEPENDENCY EmployeeTerritories(EmployeeID,TerritoryID) ->
           EmployeeID,TerritoryID;

Orders*(OrderID,CustomerID,EmployeeID,OrderDate,RequiredDate,ShipName,ShipAddress,
        ShipCity,ShipRegion,ShipPostalCode,ShipCountry;ShippedDate,ShipVia,Freight)
FUNCTIONAL DEPENDENCY Orders(OrderID) -> OrderID,CustomerID,EmployeeID,OrderDate,
           RequiredDate,ShipName,ShipAddress,ShipCity,ShipRegion,ShipPostalCode,ShipCountry;


Products(ProductID,ProductName,SupplierID,CategoryID,QuantityPerUnit,UnitPrice,UnitsInStock,
         UnitsOnOrder,ReorderLevel,Discontinued)
FUNCTIONAL DEPENDENCY Products(ProductID) ->
           ProductID,ProductName,SupplierID,CategoryID,QuantityPerUnit,
           UnitPrice,UnitsInStock,UnitsOnOrder,ReorderLevel,Discontinued;


(*******************)
(* Composite Views *)
(*******************)
Order_Subtotals(OrderId, Subtotal)
Order_Detail_Extended*(ProductID;OrderID;ProductName, UnitPrice, Quantity, Discount,ExtendedPrice)
Product_Sales_for_1997*(CategoryName, ProductName; ProductSales)
```

13.2 Northwind 2 (NW2)

The schema is the same with Northwinds I except for the definition of the `Products` table.

```
Products*(ProductID,ProductName,SupplierID,CategoryID,QuantityPerUnit,UnitPrice;UnitsInStock,UnitsOnOrder,Re
FUNCTIONAL DEPENDENCY Products(ProductID) ->
ProductID,ProductName,SupplierID,CategoryID,QuantityPerUnit;
```

13.3 Northwind 3 (NW3)

The schema is the same with Northwinds I (NW1) except for the definitions of the `Products` and `Customers` table.

```
Customers*(CustomerID,CompanyName,ContactName,ContactTitle,Address,City,Region,
          PostalCode,Country,Phone,Fax);
```

13.4 Adventure Works (AW1)

```
(************************************)
(* These are all tables used by MViews *)
(************************************)

HumanResources_Department(DepartmentID,Name,GroupName,ModifiedDate)
FUNCTIONAL DEPENDENCY HumanResources_Department(DepartmentID) -> DepartmentID,Name,
           GroupName,ModifiedDate;
```

```
HumanResources_Employee*(EmployeeID;NationalIDNumber,ContactID,LoginID,ManagerID,Title,BirthDate,
                         MaritalStatus,Gender,HireDate,SalariedFlag,VacationHours,SickLeaveHours,
                         CurrentFlag,rowguid,ModifiedDate)
FUNCTIONAL DEPENDENCY HumanResources_Employee(EmployeeID) -> EmployeeID,NationalIDNumber,ContactID,
           LoginID,ManagerID,Title,BirthDate,MaritalStatus,Gender,HireDate,SalariedFlag,VacationHours,
           SickLeaveHours,CurrentFlag,rowguid,ModifiedDate;

HumanResources_EmployeeAddress*(EmployeeID;AddressID,rowguid,ModifiedDate)
FUNCTIONAL DEPENDENCY HumanResources_EmployeeAddress(EmployeeID,AddressID) -> EmployeeID,AddressID,
           rowguid,ModifiedDate;

HumanResources_EmployeeDepartmentHistory(EmployeeID,DepartmentID,ShiftID,StartDate,EndDate,ModifiedDate)
FUNCTIONAL DEPENDENCY HumanResources_EmployeeDepartmentHistory(EmployeeID,StartDate,DepartmentID,ShiftID) ->
           EmployeeID,DepartmentID,ShiftID,StartDate,EndDate,ModifiedDate;

HumanResources_Shift(ShiftID,Name,StartTime,EndTime,ModifiedDate)
FUNCTIONAL DEPENDENCY HumanResources_Shift(ShiftID) -> ShiftID,Name,StartTime,
           EndTime,ModifiedDate;

Person_Address*(AddressID;AddressLine1,AddressLine2,City,StateProvinceID,
                PostalCode,rowguid,ModifiedDate)
FUNCTIONAL DEPENDENCY Person_Address(AddressID) -> AddressID,AddressLine1,AddressLine2,City,
           StateProvinceID,PostalCode,rowguid,ModifiedDate;

Person_AddressType(AddressTypeID,Name,rowguid,ModifiedDate)
FUNCTIONAL DEPENDENCY Person_AddressType(AddressTypeID) -> AddressTypeID,Name,rowguid,ModifiedDate;

Person_Contact(ContactID,NameStyle,Title,FirstName,MiddleName,LastName,Suffix,EmailAddress,
               EmailPromotion,Phone,PasswordHash,PasswordSalt,AdditionalContactInfo,rowguid,ModifiedDate)
FUNCTIONAL DEPENDENCY Person_Contact(ContactID) -> ContactID,NameStyle,Title,FirstName,
           MiddleName,LastName,Suffix,EmailAddress,EmailPromotion,Phone,PasswordHash,PasswordSalt,
           AdditionalContactInfo,rowguid,ModifiedDate;

Person_ContactType(ContactTypeID,Name,ModifiedDate)
FUNCTIONAL DEPENDENCY Person_ContactType(ContactTypeID) -> ContactTypeID,Name,ModifiedDate;

Person_CountryRegion(CountryRegionCode,Name,ModifiedDate)
FUNCTIONAL DEPENDENCY Person_CountryRegion(CountryRegionCode) -> CountryRegionCode,Name,ModifiedDate;

Person_StateProvince(StateProvinceID,StateProvinceCode,CountryRegionCode,IsOnlyStateProvinceFlag,Name,
                     TerritoryID,rowguid,ModifiedDate)
FUNCTIONAL DEPENDENCY Person_StateProvince(StateProvinceID) -> StateProvinceID,StateProvinceCode,
           CountryRegionCode,IsOnlyStateProvinceFlag,Name,TerritoryID,rowguid,ModifiedDate;

Production_ProductDescription*(ProductDescriptionID;Description,rowguid,ModifiedDate)
FUNCTIONAL DEPENDENCY Production_ProductDescription(ProductDescriptionID) -> ProductDescriptionID,
           Description,rowguid,ModifiedDate;

Production_ProductModel(ProductModelID,Name,CatalogDescription,Instructions,rowguid,ModifiedDate)
FUNCTIONAL DEPENDENCY Production_ProductModel(ProductModelID) -> ProductModelID,Name,CatalogDescription,
           Instructions,rowguid,ModifiedDate;
```

```
Production_ProductModelProductDescriptionCulture(ProductModelID,ProductDescriptionID,CultureID,ModifiedDate)
FUNCTIONAL DEPENDENCY Production_ProductModelProductDescriptionCulture(ProductModelID,ProductDescriptionID,
           CultureID) -> ProductModelID,ProductDescriptionID,CultureID,ModifiedDate;

Sales_Customer(CustomerID,TerritoryID,CustomerType,rowguid,ModifiedDate)
FUNCTIONAL DEPENDENCY Sales_Customer(CustomerID) -> CustomerID,TerritoryID,CustomerType,
           rowguid,ModifiedDate;

Sales_CustomerAddress*(CustomerID,AddressID,AddressTypeID;rowguid,ModifiedDate)
FUNCTIONAL DEPENDENCY Sales_CustomerAddress(CustomerID,AddressID) -> CustomerID,AddressID,
           AddressTypeID,rowguid,ModifiedDate;

Sales_SalesTerritoryHistory(SalesPersonID,TerritoryID,StartDate,EndDate,rowguid,ModifiedDate)
FUNCTIONAL DEPENDENCY Sales_SalesTerritoryHistory(SalesPersonID,StartDate,TerritoryID) -> SalesPersonID,
           TerritoryID,StartDate,EndDate,rowguid,ModifiedDate;

Sales_StoreContact*(CustomerID,ContactID,ContactTypeID,rowguid;ModifiedDate)
FUNCTIONAL DEPENDENCY Sales_StoreContact(CustomerID,ContactID) -> CustomerID,ContactID,ContactTypeID,
           rowguid,ModifiedDate;
```

The rest of the schema is not used by the views and is omitted.

13.5 Adventure Works 2 (AW2)

The schema is the same as AW1, except that HumanResources_Employee, Production_ProductDescription and Sales_StoreContact are deterministic.

13.6 TPC-H

```
PART*(P_PARTKEY,P_NAME,P_MFGR,P_BRAND,P_TYPE,P_SIZE,P_CONTAINER,P_COMMENT;P_RETAILPRICE)
FUNCTIONAL DEPENDENCY PART(P_PARTKEY) ->
P_NAME,P_MFGR,P_BRAND,P_TYPE,P_SIZE,P_CONTAINER,P_COMMENT;

SUPPLIER*(S_SUPPKEY,S_NAME,S_ADDRESS,S_NATIONKEY,S_PHONE,S_COMMENT;S_ACCTBAL)
FUNCTIONAL DEPENDENCY SUPPLIER(S_SUPPKEY) -> S_NAME,S_ADDRESS,S_NATIONKEY,S_PHONE,S_COMMENT;

PARTSUPP(PS_PARTKEY,PS_SUPPKEY,PS_AVAILQTY,PS_SUPPLYCOST,PS_COMMENT)
FUNCTIONAL DEPENDENCY PARTSUPP(PS_PARTKEY,PS_SUPPKEY) -> PS_AVAILQTY,PS_SUPPLYCOST,PS_COMMENT;

CUSTOMER*(C_CUSTKEY,C_NAME,C_ADDRESS,C_NATIONKEY,C_PHONE,C_ACCTBAL,
          C_MKTSEGMENT,C_COMMENT)
FUNCTIONAL DEPENDENCY CUSTOMER(C_CUSTKEY) -> C_NAME,C_ADDRESS,C_NATIONKEY,C_PHONE,C_ACCTBAL,
           C_MKTSEGMENT,C_COMMENT;

NATION(N_NATIONKEY,N_NAME,N_REGIONKEY,N_COMMENT)
FUNCTIONAL DEPENDENCY NATION(N_NATIONKEY) -> N_NAME,N_REGIONKEY,N_COMMENT;

REGION(R_REGIONKEY,R_NAME,R_COMMENT)
FUNCTIONAL DEPENDENCY REGION(R_REGIONKEY) -> R_NAME,R_COMMENT;

LINEITEM*(L_ORDERKEY,L_LINENUMBER,L_QUANTITY,L_EXTENDEDPRICE,L_DISCOUNT,
 L_TAX,L_RETURNFLAG,L_LINESTATUS,L_SHIPDATE,L_COMMITDATE,L_RECEIPTDATE,
```

```
         L_SHIPINSTRUCT,L_SHIPMODE,L_COMMENT;L_PARTKEY,L_SUPPKEY)

(* FUNCTIONAL DEPENDENCY LINEITEM(L_ORDERKEY) -> L_PARTKEY,L_SUPPKEY,L_LINENUMBER,L_QUANTITY,
        L_EXTENDEDPRICE,L_DISCOUNT,L_TAX,L_RETURNFLAG,L_LINESTATUS,
        L_SHIPDATE,L_COMMITDATE,L_RECEIPTDATE,L_SHIPINSTRUCT,L_SHIPMODE,L_COMMENT; *)

FUNCTIONAL DEPENDENCY LINEITEM(L_ORDERKEY) -> L_LINENUMBER,L_QUANTITY,L_EXTENDEDPRICE,
        L_DISCOUNT,L_TAX,L_RETURNFLAG,L_LINESTATUS,L_SHIPDATE,L_COMMITDATE,L_RECEIPTDATE,
        L_SHIPINSTRUCT,L_SHIPMODE,L_COMMENT;

ORDERS*(O_ORDERKEY,O_CUSTKEY,O_TOTALPRICE,O_ORDERDATE,O_ORDERPRIORITY,O_CLERK,O_SHIPPRIORITY,
        O_COMMENT;O_ORDERSTATUS)
FUNCTIONAL DEPENDENCY ORDERS(O_ORDERKEY) -> O_CUSTKEY,O_TOTALPRICE,O_ORDERDATE,
        O_ORDERPRIORITY,O_CLERK,O_SHIPPRIORITY,O_COMMENT;
```