A useful four-valued database logic

Gösta Grahne
Concordia University, Montreal, Canada
grahne@cs.concordia.ca

Ali Moallemi
Concordia University, Montreal, Canada
moa_ali@encs.concordia.ca

ABSTRACT

Recently there has been an effort to solve the problems caused by the infamous NULL in relational databases, by systematically applying Kleene's three-valued logic to SQL. The third truth-value is *unknown*. In this paper we show that by using a fourth truth-value *inconsistent*, all the advantages of the three-valued approach can be retained, and that negation can be given a constructive, intuitionistic meaning that allows negative knowledge to be specified in the logic explicitly, without having to resort to extra-logical notions of stratification or to non-monotonic reasoning. The four-valued approach also allows for a computationally efficient treatment of query answering in the presence of inconsistencies. This is in contrast to the computationally intractable repair approach to inconsistency management. From a practical view-point we show that the Cylindric Star Algebra, developed by the authors, is particularly well suited for evaluating First Order queries on four-valued databases, and that the framework of data exchange can smoothly adapted to the four truth-values.

CCS CONCEPTS

• **Information systems → Structured Query Language**; **Data exchange**; *Relational database model*; • **Theory of computation → Constructive mathematics**;

KEYWORDS

Relational Databases,Query Languages, Information Integration, Cylindric Algebra, Chase, Query Evaluation, Intuitionistic Logic

ACM Reference format:
Gösta Grahne and Ali Moallemi. 2018. A useful four-valued database logic. In *Proceedings of 22nd International Database Engineering & Applications Symposium, Villa San Giovanni, Italy, June 18–20, 2018 (IDEAS 2018),* 9 pages.
https://doi.org/10.1145/3216122.3216157

1 INTRODUCTION

SQL has for decades been the standard query language for relational databases. Its success is largely due to its declarative nature and sound theoretical basis in first order logic and relational algebra. However, it has long been recognized that with respect to the infamous NULL, the semantics of SQL is a source of confusion [4]. This is due to the fact that SQL interprets the NULLs in three-valued logic, with truth-values "true," "false," and "unknown," whereas SQL otherwise adheres to two-valued logic. Recently, the work of Libkin et al. [10, 15, 16] has shown that these shortcomings can be remedied by some adjustments that make the semantics adhere to Kleene's strong three-valued logic [14].

In the 1970's Nuel Belnap [2] extended Kleene's logic with a fourth truth-value "inconsistent." We show in this paper that Belnap's four-valued logic can be adapted to relational databases, and that it allows for efficient treatment of not only incomplete information but also offers a new, efficient approach to inconsistency management. This is in contrast to the hitherto used inherently intractable repair-approach introduced by Arenas et al. [1]

More precisely, we show that four-valued databases can be losslessly decomposed into a positive and negative part, and that any *First Order (FO)* query can be decomposed as a pair of *Positive*[1] *First Order (FO^+)* queries evaluated on the decomposed database. The four-valued approach also makes the metalogical notion of "closed world" semantics unnecessary, since the negative information can be explicitly specified within the logic and stored in the negative part of the database, thus allowing partially closed relations, in addition to completely "open" or "closed" relations. Storing the negative information however requires the use of *universal nulls*, in addition to the usual well-known *existential nulls*. We show that the *Cylindric Star-tables* and the *Cylindric Star-algebra*, described by the authors in [8], are well suited for this purpose. Any FO query on a decomposed four-valued database can be efficiently evaluated using the positive part of our cylindric star algebra. Finally, we also show that all notions related to dependency satisfaction and data exchange smoothly generalize to our four-valued framework.

Related work. As mentioned above, the four-valued logic was defined by Nuel Belnap in 1977 [2]. It subsequently generated a lot of attention in the AI and Non Monotonic Reasoning communities, as a consequence of Matt Ginsberg's generalization of Belnap's four-valued truth-space into so called *Bilatticies* [7]. Follow ups are too numerous to survey here, but the work of Melvin Fitting (see e.g. [5, 6]) offers a systematic overview. The adaptation of Belnap's logic to data exchange was described in [9], where the notion of negative answer was defined. Negative answers was also the topic of Libkin's paper [15]. Guagliardo and Libkin [10, 16] have also made efforts to show that SQL can be made consistent with Kleene's three-valued logic, thereby providing means to clean up the SQL NULL-mess. The four-valued logic offers in addition a sound and efficient inconsistency management. Universal nulls were first considered by Biskup [3]. The possibility that universal nulls and its algebra can be extracted from the work of Henkin, Monk, and Tarski [11, 12] was recognized by Imielinski and Lipski in [13], but the approach was not fully developed before [8].

[1]A positive FO-formula does not use negation, but universal quantification is allowed.

Outline. The rest of this paper is organized as follows. The next section introduces the notions used throughout the paper, along with Belnap's logic that plays a central role in this paper. Section 3 describes the formula decomposition of FO queries, followed by Section 4 dedicated to interpretation and implementation of the universal null. Section 5 demonstrates the algebraic evaluation of the decomposed queries. Sections 6 and 7 show that dependencies can be given a four-valued interpretation, and that our decomposition technique can be applied to dependencies, and that the classical chase-process adapted to the four-valued approach. Conclusions are drawn in Section 8.

2 BELNAP'S FOUR-VALUED LOGIC

This section introduces Belnap's four-valued logic [2] in the context of databases. We first formally define our database model.

Schema. Throughout this paper we assume a fixed schema $\mathscr{R} = \{R_1, \ldots, R_m, \approx\}$, where each R is a relational symbol, with an associated positive integer $ar(R)$, called the *arity* of R. The symbol $\approx$ represents equality.

Language. Our calculus is the standard domain relational calculus. Let $\{x_1, x_2, \ldots\}$ be a countably infinite set of *variables*. Sequences of variables, such as e.g. $x_1, x_2, \ldots, x_k$ are denoted $\bar{x}$. We define the set of *FO-formulas* (over $\mathscr{R}$) in the usual way: $R(x_{i_1}, \ldots, x_{i_{ar(R)}})$ and $x_i \approx x_j$ are atomic formulas, and these are closed under $\wedge, \vee, \neg, \exists x_i$, and $\forall x_i$. If a formula φ has free variables $\bar{x}$, we sometimes emphasize this by writing $\varphi(\bar{x})$. A formula without free variables is called a *sentence*.

Four-valued instances. Let $\mathbb{D} = \{a_1, a_2, \ldots\}$ be a finite or countably infinite *domain*. Elements of $\mathbb{D}$ are called *constants*, and sequences of constants, such as e.g. $a_1, a_2, \ldots, a_k$ are denoted $\bar{a}$. In a four-valued instance I (with universe $\mathbb{D}$) each relational symbol R determines a mapping $R^I : \mathbb{D}^{ar(R)} \to \{1, 0, \top, \bot\}$, where the codomain represents truth-values *true*, *false*, *inconsistent*, and *unknown*, respectively. In other words, for each $R \in \mathscr{R}$ and each $\bar{a} \in \mathbb{D}^{ar(R)}$, we have $R^I(\bar{a}) \in \{1, 0, \top, \bot\}$. Equality $\approx$ is (for now) interpreted as identity, and equality atoms are consequently assigned only values 1 and 0. The Boolean part of Belnap's four-valued logic is characterized by the following truth-tables. Note that the logic is an extension of Kleene's strong three-valued logic, which again is an extension of classical two-valued logic.

$\wedge^4$	1	0	$\top$	$\bot$
1	1	0	$\top$	$\bot$
0	0	0	0	0
$\top$	$\top$	0	$\top$	0
$\bot$	$\bot$	0	0	$\bot$

$\vee^4$	1	0	$\top$	$\bot$
1	1	1	1	1
0	1	0	$\top$	$\bot$
$\top$	1	$\top$	$\top$	1
$\bot$	1	$\bot$	1	$\bot$

$\neg^4$	
1	0
0	1
$\top$	$\top$
$\bot$	$\bot$

FO-sentences φ can now be assigned truth-values in $\{1, 0, \top, \bot\}$ recursively as follows:

$$\begin{aligned}
(\varphi \wedge \psi)^I &= \varphi^I \wedge^4 \psi^I \\
(\varphi \vee \psi)^I &= \varphi^I \vee^4 \psi^I \\
(\neg\varphi)^I &= \neg^4(\varphi^I) \\
(\exists x\, \varphi(x))^I &= \vee^4_{a\in\mathbb{D}}\, \varphi(a)^I \\
(\forall x\, \varphi(x))^I &= \wedge^4_{a\in\mathbb{D}}\, \varphi(a)^I
\end{aligned}$$

Queries and answers. Queries are expressed in FO and interpreted on four-valued databases. Let $\varphi(\bar{x})$ be an FO-formula with free variables $\bar{x}$. A valuation v is a (partial) mapping from variables to $\mathbb{D}$. Applying v to $\varphi(\bar{x})$ consists of replacing each variable x in $\bar{x}$ with $v(x)$. This is denoted $\varphi(v(\bar{x}))$. The result of applying query $\varphi(\bar{x})$ to a four-valued instance I will yield the following four types of answers:

$$\begin{array}{lcll}
\mathsf{true}(\varphi, I) &=& \{v(\bar{x}) : \varphi(v(\bar{x}))^I = 1\} & \text{The true answer} \\
\mathsf{false}(\varphi, I) &=& \{v(\bar{x}) : \varphi(v(\bar{x}))^I = 0\} & \text{The false answer} \\
\mathsf{inc}(\varphi, I) &=& \{v(\bar{x}) : \varphi(v(\bar{x}))^I = \top\} & \text{The inconsistent answer} \\
\mathsf{unk}(\varphi, I) &=& \{v(\bar{x}) : \varphi(v(\bar{x}))^I = \bot\} & \text{The unknown answer}
\end{array}$$

Example 2.1. Consider binary relations *F(ollows)* and *H(obbies)*, where $F(x, y)$ means that user x follows user y on a social media site, and $H(x, z)$ means that z is a hobby of user x. Let the database instance I be the following.

F		F^I
Alex	Bob	1
Bob	Alex	$\top$
Bob	Alice	1

H		H^I
Alex	Movie	$\top$
Alex	Music	1
Alice	Music	1
Bob	Movies	0

Facts given in I state that Alex follows Bob, but not himself. There is no information as to whether Bob follows himself or not, while there is contradictory information about Bob following Alex. Unequivocally Bob follows Alice. All other possible facts about the Follows relation are unknown. Thus for instance $F^I(\text{Alice}, \text{Alex}) = \bot$. The facts about relation H are interpreted similarly. Let the query ask for people who are following someone who does not have Movies as hobby. This is formulated in FO as $\varphi(x)$, where

$$\varphi(x) \;=\; \exists y\, F(x, y) \wedge \neg H(y, \text{Movies}).$$

Then, assuming that $\mathbb{D}$ consists of the values in the instance only, $\varphi(x)$ will be evaluated as follows.

$$\begin{aligned}
\varphi(\text{Alex})^I \;=&\\
& \textstyle\bigvee^4_{a\in\mathbb{D}} \left(F(\text{Alex}, a)^I \wedge^4(\neg^4(H(a, \text{Movies})^I))\right) \\
=\;& \left(F(\text{Alex}, \text{Alex})^I \wedge^4 (\neg^4(H(\text{Alex}, \text{Movies})^I))\right) \\
& \vee^4 \left(F(\text{Alex}, \text{Bob})^I \wedge^4(\neg^4(H(\text{Bob}, \text{Movies})^I))\right) \\
& \vee^4 \left(F(\text{Alex}, \text{Alice})^I \wedge^4(\neg^4(H(\text{Alice}, \text{Movies})^I))\right) \\
=\;& \left(\bot \wedge^4(\neg^4 \top)\right) \vee^4 \left(1 \wedge^4(\neg^4 0)\right) \vee^4 \left(\bot \wedge^4(\neg^4 \bot)\right) \\
=\;& \left(\bot \wedge^4 \top\right) \vee^4 \left(1 \wedge^4 1\right) \vee^4 \left(\bot \wedge^4 \bot\right) \\
=\;& 1 \vee^4 1 \vee^4 \bot \;=\; 1
\end{aligned}$$

Similarly for Bob and Alice,

$$\begin{aligned}
\varphi(\text{Bob})^I &= \vee^4_{a\in\mathbb{D}} \left(F(\text{Bob}, a)^I \wedge^4 (\neg^4 H(a, \text{Movies})^I)\right) = \top \\
\varphi(\text{Alice})^I &= \vee^4_{a\in\mathbb{D}} \left(F(\text{Alice}, a)^I \wedge^4 (\neg^4 H(a, \text{Movies})^I)\right) = \bot
\end{aligned}$$

We thus have $\mathsf{true}(\varphi, I) = \{(\text{Alex})\}$, $\mathsf{false}(\varphi, I) = \{\}$, $\mathsf{inc}(\varphi, I) = \{(\text{Bob})\}$, and $\mathsf{unk}(\varphi, I) = \{(\text{Alice})\}$.

3 DATABASE AND FORMULA DECOMPOSITION

It turns out that any four-valued instance I can be losslessly represented as a pair $I^{\pm} = (I^+, I^-)$ where I^+ and I^- are two-valued instances. More precisely, for each relation symbol R and sequence of constants $\bar{a} \in \mathbb{D}^{ar(R)}$,

$$R^{I^+}(\bar{a}) = 1 \text{ iff } R^I(\bar{a}) \in \{1, \top\}, \text{ and } R^{I^-}(\bar{a}) = 1 \text{ iff } R^I(\bar{a}) \in \{0, \top\}.$$

Conversely, given a pair of two-valued instances (I^+, I^-), we can construct a four-valued instance $I^+ \otimes I^-$, where

$$\begin{aligned} R^{I^+ \otimes I^-}(\bar{a}) &= 1 \text{ if } R^{I^+}(\bar{a}) = 1 \text{ and } R^{I^-}(\bar{a}) = 0;\\ R^{I^+ \otimes I^-}(\bar{a}) &= 0 \text{ if } R^{I^+}(\bar{a}) = 0 \text{ and } R^{I^-}(\bar{a}) = 1;\\ R^{I^+ \otimes I^-}(\bar{a}) &= \top \text{ if } R^{I^+}(\bar{a}) = 1 \text{ and } R^{I^-}(\bar{a}) = 1;\\ R^{I^+ \otimes I^-}(\bar{a}) &= \bot \text{ if } R^{I^+}(\bar{a}) = 0 \text{ and } R^{I^-}(\bar{a}) = 0. \end{aligned}$$

Also, $\approx^{I^+} = \{(a, a) : a \in \mathbb{D}\}$, and $\approx^{I^-} = \{(a, b) : a, b \in \mathbb{D}, a \neq b\}$. The following lemma follows directly from the definitions and verifies that $I^{\pm}$ indeed is a lossless decomposition of I.

Lemma 3.1. *Let I be a four-valued instance, and $I^{\pm} = (I^+, I^-)$ its decomposition. Then $I^+ \otimes I^- = I$.*

Example 3.2. Consider binary relations *F(ollows)* and *H(obbies)* from Example 2.1. By applying the decomposition to relations F^I and H^I, the two-valued relation F^{I^+} is populated with tuples $(a_1, a_2) \in \mathbb{D}^2$, where $F^I(a_1, a_2) \in \{1, \top\}$, and relation H^{I^+} is populated with tuples $(a_1, a_2) \in \mathbb{D}^2$, where $H^I(a_1, a_2) \in \{1, \top\}$. Similarly, F^{I^-} is populated with tuples $(a_1, a_2) \in \mathbb{D}^2$, where $F^I(a_1, a_2) \in \{0, \top\}$, and H^{I^-} is populated with tuples $(a_1, a_2) \in \mathbb{D}^2$, where $H^I(a_1, a_2) \in \{0, \top\}$. Note that unknown tuples are not stored in the decomposition. The following four tables represents the two-valued decomposition of relations F^I and M^I into F^{I^+}, F^{I^-}, H^{I^+} and H^{I^-}.

F^{I^+}	
Alex	Bob
Bob	Alex
Bob	Alice

F^{I^-}	
Bob	Alex

H^{I^+}	
Alex	Music
Alex	Movies
Alice	Music

H^{I^-}	
Alex	Movies
Bob	Movies

The above are the relations actually stored in the database, and FO-queries on the four-valued instance will be decomposed and executed against these two-valued relations. In the following $\wedge^2$ and $\vee^2$ denote the classical two-valued operators. Of course, when restricted to $\{1, 0\}$, two- and four-valued "and" and "or" agree.

Before we define the FO-formula decomposition, we note that $\pm$ and $\otimes$ can be seen as mappings from $\{1, 0, \top, \bot\}$ to $\{1, 0\} \times \{1, 0\}$, an the other way around, respectively. In other words, $1^{\pm} = (1, 0)$ and $1 \otimes 0 = 1$; $0^{\pm} = (0, 1)$ and $0 \otimes 1 = 0$; $\top^{\pm} = (1, 1)$ and $1 \otimes 1 = \top$; $\bot^{\pm} = (0, 0)$ and $0 \otimes 0 = \bot$. Then next lemma is a straightforward consequence of these definitions.

Lemma 3.3. *Let $p, q \in \{1, 0, \top, \bot\}$. Suppose $p^{\pm} = (p^+, p^-)$, and $q^{\pm} = (q^+, q^-)$. Then*

$$\begin{aligned} p \wedge^4 q &= (p^+ \wedge^2 q^+) \otimes (p^- \vee^2 q^-)\\ p \vee^4 q &= (p^+ \vee^2 q^+) \otimes (p^- \wedge^2 q^-)\\ \neg^4 p &= p^- \otimes p^+ \end{aligned}$$

The next definition describes the decomposition to be used in the evaluation FO-formulas on decomposed four-valued databases. We emphasize the fact that the decomposed formulas do not use negation.

Definition 3.4. Let φ be an FO-sentence, I a four-valued instance, and $I^{\pm} = (I^+, I^-)$ its decomposition. Then $\varphi^{I^{\pm}} = (\varphi^{I^+}, \varphi^{I^-})$ is defined recursively as follows:

φ	φ^{I^+}	φ^{I^-}
$R(\bar{a})$	$R(\bar{a})^{I^+}$	$R(\bar{a})^{I^-}$
$\neg\psi$	ψ^{I^-}	ψ^{I^+}
$a_i \approx a_j$	$(a_i \approx a_j)^{I^+}$	$(a_i \approx a_j)^{I^-}$
$\psi \wedge \xi$	$\psi^{I^+} \wedge^2 \xi^{I^+}$	$\psi^{I^-} \vee^2 \xi^{I^-}$
$\psi \vee \xi$	$\psi^{I^+} \vee^2 \xi^{I^+}$	$\psi^{I^-} \wedge^2 \xi^{I^-}$
$\exists x\, \psi(x)$	$\vee^2_{a \in \mathbb{D}} \psi^{I^+}(a)$	$\wedge^2_{a \in \mathbb{D}} \psi^{I^-}(a)$
$\forall x\, \psi(x)$	$\wedge^2_{a \in \mathbb{D}} \psi^{I^+}(a)$	$\vee^2_{a \in \mathbb{D}} \psi^{I^-}(a)$

We can now verify the desired property of the decomposition of instances and FO-formulas.

Theorem 3.5. *Let φ be an FO-sentence, I a four-valued instance, and $I^{\pm} = (I^+, I^-)$ its decomposition. Then*

$$\varphi^I = \varphi^{I^+} \otimes \varphi^{I^-}.$$

Proof. We do a structural induction on φ. If φ equals $R(\bar{a})$ or $a_i \approx a_j$ the claim follows directly from Lemma 3.1. For the inductive step, if $\varphi = \psi \wedge \xi$, we have

$$\varphi^I = (\psi \wedge \xi)^I = \psi^I \wedge^4 \xi^I,$$

where we assume that $\psi^I = \psi^{I^+} \otimes \psi^{I^-}$, and $\xi^I = \xi^{I^+} \otimes \xi^{I^-}$. By Lemma 3.3 we then have

$$\psi^I \wedge^4 \xi^I = (\psi^{I^+} \wedge^2 \xi^{I^+}) \otimes (\psi^{I^-} \vee^2 \xi^{I^-}).$$

Definition 3.4 now tells us that

$$(\psi^{I^+} \wedge^2 \xi^{I^+}) \otimes (\psi^{I^-} \vee^2 \xi^{I^-}) = \varphi^{I^+} \otimes \varphi^{I^-},$$

showing that indeed $\varphi^I = \varphi^{I^+} \otimes \varphi^{I^-}$.

For the next case of the inductive step, suppose $\varphi = \neg\psi$, where the assumption is that $\psi^I = \psi^{I^+} \otimes \psi^{I^-}$. We have

$$\varphi^I = (\neg\psi)^I = \neg^4(\psi^I) = \neg^4(\psi^{I^+} \otimes \psi^{I^-}) = \psi^{I^-} \otimes \psi^{I^+} = \varphi^{I^+} \otimes \varphi^{I^-}.$$

The case for disjunction is similar that of conjunction, and since the quantifiers $\forall$ and $\exists$ are defined in terms of $\wedge^4$ and $\vee^4$ the result holds in these cases also. □

Corollary 3.6. *Let φ be an FO-sentence, I a four-valued instance, and $I^{\pm} = (I^+, I^-)$ its decomposition. Then*

$$\begin{aligned} true(\varphi, I) &= \varphi^{I^+} \setminus \varphi^{I^-} \\ false(\varphi, I) &= \varphi^{I^-} \setminus \varphi^{I^+} \\ inc(\varphi, I) &= \varphi^{I^+} \cap \varphi^{I^-} \\ unk(\varphi, I) &= \overline{(\varphi^{I^+} \cup \varphi^{I^-})} \end{aligned}$$

Example 3.7. Let $\varphi(x)$ be the query $\exists y\, F(x,y) \wedge \neg H(y, \text{Movies})$. from Example 2.1. Then $(\varphi^{I^+}, \varphi^{I^-})$ is evaluated as follows

$$\begin{aligned} \varphi^{I^+}(\text{Alex}) &= \vee^2_{a\in\mathbb{D}} \left(F^{I^+}(\text{Alex}, a) \wedge^2 H^{I^-}(a, \text{Movies})\right) &= 1 \\ \varphi^{I^-}(\text{Alex}) &= \wedge^2_{a\in\mathbb{D}} \left(F^{I^-}(\text{Alex}, a) \vee^2 H^{I^+}(a, \text{Movies})\right) &= 0 \end{aligned}$$

Similarly

$\varphi^{I^+}(\text{Bob}) = 1$ and $\varphi^{I^-}(\text{Bob}) = 1$
$\varphi^{I^+}(\text{Alice}) = 0$ and $\varphi^{I^-}(\text{Alice}) = 0$

The evaluation of each of there two-valued queries leads to the following results:

φ^{I^+}
Alex
Bob

φ^{I^-}
Bob

The final answers can now be composed according to Corollary 3.6, yielding $\text{true}(\varphi, I) = \{(\text{Alex})\}$, $\text{false}(\varphi, I) = \{\}$, $\text{inc}(\varphi, I) = \{(\text{Bob})\}$, and $\text{unk}(\varphi, I) = \{(\text{Alice})\}$.

4 UNIVERSAL NULLS

In this and the next section we assume that the domain $\mathbb{D}$ is unbounded (countably infinite). We show how the *star tables* introduced in [8] can be used to compactly store the two-valued positive and negative parts of a four-valued database. Since in the negative part we want to be able to record potentially infinite set of facts, we need *universal nulls* "$*$," where a tuple, say $(a, *)$, represents all ordinary tuples $\{(a, b) : b \in \mathbb{D}\}$. We then show how the *cylindric star algebra* described in [8] can conveniently be used to evaluate, on the two-valued positive-negative star-tables, the FO$^+$-queries resulting from the decomposition of an FO-query posed on the four-valued database. The following example represents the positive part of a database, and only true tuples are shown.

Example 4.1. Consider binary relations *F(ollows)* and *H(obbies)* from Examples 2.1 and 3.2. This time, let the database be the following.

F^{I^+}	
Alice	Chris
*	Alice
Bob	*
Chris	Bob

H^{I^+}	
Alice	Movies
Alice	Music
Bob	Movies

This is to be interpreted as expressing the facts that Alice follows Chris and Chris follows Bob. Alice is a journalist who would like to give access to everyone to articles she shares on the social media site. Therefore, everyone can follow Alice. Bob is the site administrator, and is granted the access to all files anyone shares on the site. Consequently, Bob follows everyone. "Everyone" in this context means all current and possible future users. The query below, in domain relational calculus, asks for the interests of people who are followed by everyone:

$$\varphi(x_4) = \exists x_2 \exists x_3 \forall x_1 \Big(F(x_1, x_2) \wedge H(x_3, x_4) \wedge (x_2 \approx x_3)\Big) \quad (1)$$

The answer to our example query is $\{(\text{Movies}), (\text{Music})\}$. Note that star-nulls also can be part of an answer. For instance, the query $\varphi(x_1, x_2) = F(x_1, x_2)$ would return all the tuples in F^{I^+}.

Example 4.2. Continuing Example 4.1, suppose all negative information we have deduced about the $H(obbies)$ relation, is that we know Alice doesn't play Volleyball, that Bob only has Movies as hobby, and that Chris has no hobby at all. This negative information about the relation H is represented by the table H^{I^-} below.

H^{I^-}		
Alice	Volleyball	
Bob	*	$2 \neq$ Movies
Chris	*	

Note that the second tuple has a conditions that says the symbol * in the second column represents all domain values except "Movies." Suppose the query φ asks for people who have a hobby, that is

$$\varphi(x_1) = \exists x_2\, H(x_1, x_2).$$

Then the positive part is evaluated as

$$\begin{aligned} &\varphi^{I^+}(\text{Chris}) = \vee_{a\in\mathbb{D}} H^{I^+}(\text{Chris}, a) = \\ &H^{I^+}(\text{Chris, Movies}) \vee H^{I^+}(\text{Chris, Music}) \vee \\ &H^{I^+}(\text{Chris, Movies}) \vee H^{I^+}(\text{Chris, ..}) \vee \cdots = \\ &0 \vee 0 \vee 0 \vee 0 \vee \cdots = 0. \\ &\varphi^{I^+}(\text{Alice}) = \vee_{a\in\mathbb{D}} H^{I^+}(\text{Alice}, a) = \\ &H^{I^+}(\text{Alice, Movies}) \vee H^{I^+}(\text{Alice, Music}) \vee \\ &H^{I^+}(\text{Alice, Movies}) \vee H^{I^+}(\text{Alice, ..}) \vee \cdots = \\ &0 \vee 1 \vee 0 \vee 0 \vee \cdots = 1. \\ &\varphi^{I^+}(\text{Bob}) = \vee_{a\in\mathbb{D}} H^{I^+}(\text{Bob}, a) = \\ &H^{I^+}(\text{Bob, Movies}) \vee H^{I^+}(\text{Bob, Music}) \vee \\ &H^{I^+}(\text{Bob, Movies}) \vee H^{I^+}(\text{Bob, ..}) \vee \cdots = \\ &0 \vee 0 \vee 1 \vee 0 \vee \cdots = 1\,. \end{aligned}$$

The negative part is evaluated as

$$\varphi^{I^-}(\text{Chris}) = \bigwedge_{a\in\mathbb{D}} H^{I^-}(\text{Chris}, a) = 1 \wedge 1 \wedge 1 \wedge \cdots = 1\,.$$

Note that $H^{I^-}(\text{Bob}, \text{Movies}) = 0$, which yields $\varphi^{I^-}(\text{Bob}) = 0$. Likewise $\varphi^{I^-}(\text{Alice}) = 0$. To summarize, $\text{true}(\varphi, I) = \{(\text{Alice}), (\text{Bob})\}$, $\text{false}(\varphi, I) = \{(\text{Chris})\}$, $\text{inc}(\varphi, I) = \{\}$. For all other possible users the result is unknown.

5 ALGEBRAIC EVALUATION

The *n-dimensional cylindric star-algebra* [8] consists of operators *star union* $\uplus$, *star intersection* $\Cap$, *outer* c_i and *inner* $\mathsf{ɔ}_i$ *cylindrifications on dimension* i, *diagonals* d_{ij}, as well as complement. Complement will however not be used in this context, as FO-formulas are evaluated in the four-valued semantics by decomposing them into positive and negative parts, neither of which uses negation. The star-algebra acts as an evaluation mechanism, and an FO-formula φ with n variables is translated into an equivalent n-dimensional star algebra expression E_φ. At run-time, all star-tables will be expanded to have arity n by filling empty columns with "$*$." These run-time tables are called *star-cylinders*, and denoted C, C' etc. Note that star-cylinders and -tables can have an extra column containing equality constraints which are a Boolean combinations of atoms of the form $(i = j)$ and $(i = a)$, for $i \in \{1, \ldots, n\}$ and $a \in \mathbb{D}$, where columns i and j contain the "$*$"-symbol.

Example 5.1. Continuing Example 4.1, in that database the atoms $F(x_1, x_2)$ and $H(x_3, x_4)$ of query (1) are represented by star-tables C_F and C_H, and the equality atom is represented by the diagonal cylinder d_{23}. Note that these are positional relations, the "attributes" x_1, x_2, x_3, and x_4 are added for illustrative purposes only.

C_F

x_1	x_2	x_3	x_4
Alice	Chris	$*$	$*$
$*$	Alice	$*$	$*$
Bob	$*$	$*$	$*$
Chris	Bob	$*$	$*$

C_H

x_1	x_2	x_3	x_4
$*$	$*$	Alice	Movies
$*$	$*$	Alice	Music
$*$	$*$	Bob	Movies

d_{23}

x_1	x_2	x_3	x_4	
$*$	$*$	$*$	$*$	2=3

C'

x_1	x_2	x_3	x_4
$*$	Alice	Alice	Movies
$*$	Alice	Alice	Music
Bob	Alice	Alice	Movies
Bob	Alice	Alice	Music
Bob	Bob	Bob	Movies
Chris	Bob	Bob	Movies

C''

x_1	x_2	x_3	x_4
$*$	Alice	Alice	Movies
$*$	Alice	Alice	Music

C'''

x_1	x_2	x_3	x_4
$*$	$*$	$*$	Movies
$*$	$*$	$*$	Music

The star union $\dot\cup$ is carried out as a set theoretic union of the star tuples in the arguments. The star intersection is obtained by combining the star tuples in the arguments, where for instance $\{(a, *, *)\} \dot\cap \{(*, b, *)\} = \{(a, b, *)\}$. The outer cylindrification c_i represents $\exists x_i$ and is obtained by replacing column i by an unconstrained "$*$." Inner cylindrification $\mathrm{ɔ}_i$ represents $\forall x_i$ and is carried out be selecting those star tuples where column i contains an unconstrained "$*$." A detailed definition of the cylindric star algebra can be found in [8]. The translation of query (1) is the cylindric star-algebra expression

$$\mathrm{c}_2(\mathrm{c}_3(\mathrm{ɔ}_1((C_F \dot\cap C_H) \dot\cap \mathrm{d}_{23}))) \tag{2}$$

The intersection of C_F and C_H is carried out as star-intersection $\dot\cap$. The result will contain 12 tuples, and when these are star-intersected with d_{23}, the diagonal cylinder d_{23} will act as a selection by columns 2 and 3 being equal. The result is the left-most star-cylinder $C' = (C_F \dot\cap C_H) \dot\cap \mathrm{d}_{23}$ above. Applying the inner star-cylindrification on column 1 results in C'' in the middle above. Finally, applying outer star-cylindrifications on columns 2 and 3 of star-cylinder C'' yields the final result $C''' = \mathrm{c}_2(\mathrm{c}_3(\mathrm{ɔ}_1((C_F \dot\cap C_H) \dot\cap \mathrm{d}_{23})))$ right-most above. The system can now return the answer, i.e. the values of column 4 in cylinder C'''. Note that columns where all rows are "$*$" do not actually have to be materialized at any stage.

Example 5.2. Consider the relation $H(obbies)$ from Examples 2.1 and 4.2. This relation is then stored as the two star-tables $C_{H^{I+}}$ and $C_{H^{I-}}$ below.

$C_{H^{I+}}$

x_1	x_2
Alice	Movies
Alice	Music
Bob	Movies

$C_{H^{I-}}$

x_1	x_2	
Alice	Volleyball	
Bob	$*$	2 ≠ Movies
Chris	$*$	

The query was asking for people who have a hobby, that is $\varphi(x_1) = \exists x_2\ H(x_1, x_2)$. The positive part will be translated as $\mathrm{c}_2(C_{H^{I+}})$, and the negative part as $\mathrm{ɔ}_2(C_{H^{I-}})$. The answers are below

$\mathrm{c}_2(C_{H^{I+}})$

x_1	x_2
Alice	$*$
Bob	$*$

$\mathrm{ɔ}_2(C_{H^{I-}})$

x_1	x_2
Chris	$*$

The above translation is summarized in the next theorem.

Theorem 5.3. [8] *For every query expressed as an FO formula φ, there is a Cylindric Star Algebra Expression E_φ, such that*

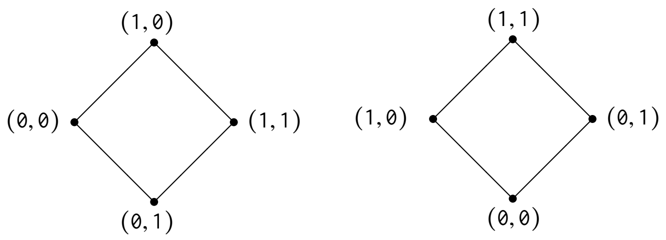
$$E_\varphi(DB) = \varphi^{DB},$$

*for every database DB containing *-nulls. The converse is also true. Moreover, star-databases are closed under star-algebra.*

6 DEPENDENCIES AND THE CHASE

Implication. So far, we have extended the basic Boolean operations to the four-valued case. When restricted to $\{1, 0\}$ the four-valued extension coincides with the two-valued case. When it comes to the conditional $\varphi \to \psi$, it will however no longer be equivalent with $\neg\varphi \vee \psi$. In order to arrive at the proper four-valued meaning of $\to$ we need to take a closer look at Belnap's logic [2].

As seen in Section 3, the truth-values 1, 0, $\top$, and $\bot$ can losslessy be represented as pairs of classical truth values 1 and 0, namely $(1,0)$, $(0,1)$, $(1,1)$, and $(0,0)$, respectively. Indeed, Belnap [2] explains a sentence φ assigned $(1,0)$ as "the computer has been told that φ is true," $(0,1)$ as "the computer has been told that φ is false," $(1,1)$ as "the computer has been told that φ is true, and that φ is false," and $(0,0)$ as "the computer has not been told anything about the truth of φ." Belnap then proposes the following two lattice diagrams:



The diagram to the left reflects the partial order $\leq_t$ of the four truth-values $\{1, 0, \top, \bot\}$, based on $0 \leq_t 1$ with $\top$ and $\bot$ in-between and incomparable between themselves. The diagram to the right reflects the amount of information that "the computer has been told." Belnap regards the structure as a Scott-type approximation lattice, and calls it the *information order*, here denoted $\leq_i$. The diagram shows that $\bot \leq_i 0 \leq_i \top$ and $\bot \leq_i 1 \leq_i \top$, with 1 and 0 incomparable between themselves wrt $\leq_i$.

The two partial orders can be lifted to four-valued instances I and J, by stipulating that $I \leq_t J$ if $R^I(\bar{a}) \leq_t R^J(\bar{a})$ for all atomic sentences $R(\bar{a})$, and similarly for $I \leq_i J$. When interpreting the information content in a four-valued instance $I = I^+ \otimes I^-$ we see that the instances I^+ and I^- are traditional model-theoretic instances "closed" wrt $\leq_t$, whereas I is "open" wrt $\leq_i$ in that the information is open to increase as the computer is told more. The propositional sentence $\varphi \rightarrow \psi$ can now be interpreted as "the computer knows about ψ at least what it knows about φ." More formally, $I \vDash \varphi \rightarrow \psi$ if $\varphi^I \leq_i \psi^I$. Note that implicational sentences are only given truth-values 1 and 0.

The next question is what the computer should do if $I \nvDash \varphi \rightarrow \psi$. Belnap answers the question by saying that the computer should "make minimal mutilations to its information state so as to make ψ true." Since information can only increase as the computer is told more, the mutilation should be done in the $\leq_i$ order. More formally, if $I \nvDash \varphi \rightarrow \psi$, then the computer should find the $\leq_i$-smallest instance J, such that $\varphi^I = \varphi^J$, $I \leq_i J$, and $\varphi^J \leq_i \psi^J$. When enforcing a tuple-generating dependency $\varphi \rightarrow \psi$ in classical two-valued instances, this is exactly what is done, except that $\leq_t$ is used instead of $\leq_i$. Of course, in a two-valued world, "more" means more truth. The effect of enforcing $\varphi \rightarrow \psi$ on a four-valued instance I resulting in instance J is described by the table below. It is easy to see that a repeated application of this enforcement rule will result in a least fixed point in the $\leq_i$-order. For a set Σ of dependencies and four-valued instance I, this least fixed point is denoted $chase^4_\Sigma(I)$.

φ^I	ψ^I	ψ^J
1	$\bot$	1
1	0	$\top$
$\top$	$\bot$	$\top$
$\top$	1	$\top$
$\top$	0	$\top$
0	$\bot$	0
0	1	$\top$

Next we show that the classical chase-procedure can be adapted to work on four-valued instances using the decomposition approach. The idea is to convert an implicational sentence $\varphi \rightarrow \psi$ into two sentences $\varphi^+ \rightarrow \psi^+$ and $\varphi^- \rightarrow \psi^-$, according to Definition 3.4, and then chase the two-valued decomposed instance $I^\pm$ with the converted sentences.

Tuple-generating dependecies. We define (for now) a *tuple generating dependency (tgd)* as an implicational sentence of the form

$$\forall \bar{x} \left(\left(\exists \bar{y}\ \varphi(\bar{x}, \bar{y}) \right) \rightarrow R(\bar{x}) \right), \tag{3}$$

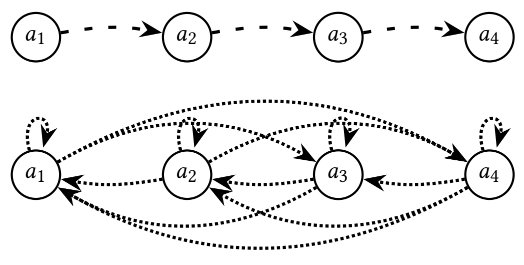
where $\varphi(\bar{x}, \bar{y})$ is an FO-sentence. Let Σ be a set of tgds and I a two-valued instance. Then $chase^2_\Sigma(I)$ is computed by repeatedly checking, for each tgd (3) in Σ, if there is a sequence $\bar{a}, \bar{b}$ of constants such that $\varphi^I(\bar{a}, \bar{b}) \nleq_t R^I(\bar{a})$, in which case $R^I(\bar{a})$ is set to 1. We can now conveniently compute $chase^4_\Sigma(I)$ by combining $chase^2_{\Sigma^{I^+}}(I^\pm)$ and $chase^2_{\Sigma^{I^-}}(I^\pm)$. where $chase^2_{\Sigma^{I^+}}(I^\pm)$ is computed by repeatedly checking, for each $\bar{a}$ and each tgd, whether $\varphi^{I^+}(\bar{a}, \bar{b}) \nleq_t R^{I^+}(\bar{a})$. If this is the case $R^{I^+}(\bar{a})$ is set to 1. Similarly, $chase^2_{\Sigma^{I^-}}(I^\pm)$ is computed by checking if $\varphi^{I^-}(\bar{a}, \bar{b}) \nleq_t R^{I^-}(\bar{a})$ and changing $R^{I^-}(\bar{a})$ to 1 when this is the case. We then have

Theorem 6.1. [9] *Let Σ be a set of tgds and I a four-valued instance. Then*

$$chase^4_\Sigma(I) = chase^2_{\Sigma^{I^+}}(I^\pm) \otimes chase^2_{\Sigma^{I^-}}(I^\pm).$$

The proof of the theorem is based on the fact that $\varphi(\bar{a}, \bar{b})^I \leq_i R(\bar{a})^I$ iff $\varphi(\bar{a}, \bar{b})^{I^+} \leq_t R(\bar{a})^{I^+}$ and $\varphi(\bar{a}, \bar{b})^{I^-} \leq_t R(\bar{a})^{I^-}$.

Example 6.2. We show how to compute the transitive closure of a graph along with the complement of the transitive closure. Let the graph have vertex set $\mathbb{D} = \{a_1, a_2, a_3, a_4\}$, and edge set E. Below is a complete description of the graph, which we call the E-graph. In the first E-graph below the dashed arrows represent true edges, and in the second one the dotted arrows represent false edges.



The transitive closure of the graph is defined by the following set of tgds (leading universal quantifiers are omitted).

$$\begin{aligned} E(x,y) &\rightarrow T(x,y) \\ \exists z \left(E(x,z) \wedge T(z,y) \right) &\rightarrow T(x,y) \end{aligned}$$

Following Fitting [5, 6] we merge all tgds with the same consequent by taking the disjunction of their antecedents. The two tgds above will thus result in the dependency

$$\left(E(x,y) \vee \left(\exists z\ E(x,z) \wedge T(z,y) \right) \right) \rightarrow T(x,y)$$

For the graph T we start with $T^I(i,j) = \bot$, for all vertices i, j. In the first round we fire

$$E^{I^+}(a_1, a_2) \vee \bigvee_{i\in\{1,2,3,4\}} \left(E^{I^+}(a_1, a_i) \wedge T^{I^+}(a_i, a_2)\right) \quad \to \quad T^{I^+}(a_1, a_2)$$

$$E^{I^+}(a_2, a_3) \vee \bigvee_{i\in\{1,2,3,4\}} \left(E^{I^+}(a_2, a_i) \wedge T^{I^+}(a_i, a_3)\right) \quad \to \quad T^{I^+}(a_2, a_3)$$

$$E^{I^+}(a_3, a_4) \vee \bigvee_{i\in\{1,2,3,4\}} \left(E^{I^+}(a_3, a_i) \wedge T^{I^+}(a_i, a_4)\right) \quad \to \quad T^{I^+}(a_3, a_4)$$

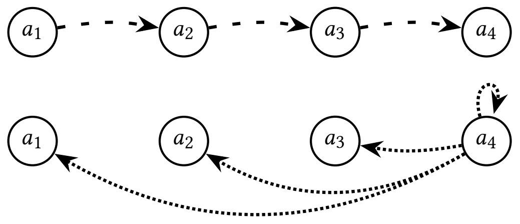
$$E^{I^-}(a_4, a_1) \wedge \bigwedge_{i\in\{1,2,3,4\}} \left(E^{I^-}(a_4, a_i) \vee T^{I^-}(a_i, a_1)\right) \quad \to \quad T^{I^-}(a_4, a_1)$$

$$E^{I^-}(a_4, a_2) \wedge \bigwedge_{i\in\{1,2,3,4\}} \left(E^{I^-}(a_4, a_i) \vee T^{I^-}(a_i, a_2)\right) \quad \to \quad T^{I^-}(a_4, a_2)$$

$$E^{I^-}(a_4, a_3) \wedge \bigwedge_{i\in\{1,2,3,4\}} \left(E^{I^-}(a_4, a_i) \vee T^{I^-}(a_i, a_3)\right) \quad \to \quad T^{I^-}(a_4, a_3)$$

$$E^{I^-}(a_4, a_4) \wedge \bigwedge_{i\in\{1,2,3,4\}} \left(E^{I^-}(a_4, a_i) \vee T^{I^-}(a_i, a_4)\right) \quad \to \quad T^{I^-}(a_4, a_4)$$

The positive and negative T-edges over $\mathbb{D} = \{a_1, a_2, a_3, a_4\}$ resulting from the first round are shown in the two graphs below.



In the second round we fire

$$E^{I^+}(a_1, a_3) \vee \bigvee_{i\in\{1,2,3,4\}} \left(E^{I^+}(a_1, a_i) \wedge T^{I^+}(a_i, a_3)\right) \quad \to \quad T^{I^+}(a_1, a_3)$$

$$E^{I^+}(a_2, a_4) \vee \bigvee_{i\in\{1,2,3,4\}} \left(E^{I^+}(a_2, a_i) \wedge T^{I^+}(a_i, a_4)\right) \quad \to \quad T^{I^+}(a_2, a_4)$$

$$E^{I^-}(a_3, a_1) \wedge \bigwedge_{i\in\{1,2,3,4\}} \left(E^{I^-}(a_3, a_i) \vee T^{I^-}(a_i, a_1)\right) \quad \to \quad T^{I^-}(a_3, a_1)$$

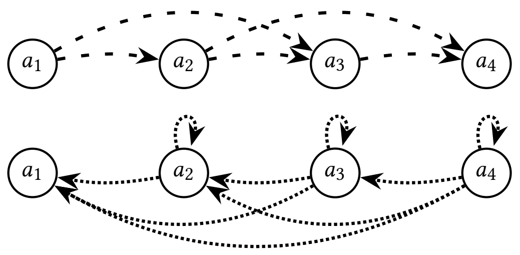
$$E^{I^-}(a_3, a_2) \wedge \bigwedge_{i\in\{1,2,3,4\}} \left(E^{I^-}(a_3, a_i) \vee T^{I^-}(a_i, a_2)\right) \quad \to \quad T^{I^-}(a_3, a_2)$$

$$E^{I^-}(a_3, a_3) \wedge \bigwedge_{i\in\{1,2,3,4\}} \left(E^{I^-}(a_3, a_i) \vee T^{I^-}(a_i, a_3)\right) \quad \to \quad T^{I^-}(a_3, a_3)$$

$$E^{I^-}(a_2, a_1) \wedge \bigwedge_{i\in\{1,2,3,4\}} \left(E^{I^-}(a_2, a_i) \vee T^{I^-}(a_i, a_1)\right) \quad \to \quad T^{I^-}(a_2, a_1)$$

$$E^{I^-}(a_2, a_2) \wedge \bigwedge_{i\in\{1,2,3,4\}} \left(E^{I^-}(a_2, a_i) \vee T^{I^-}(a_i, a_2)\right) \quad \to \quad T^{I^-}(a_2, a_2)$$

The second round results in the positive and negative graphs below.

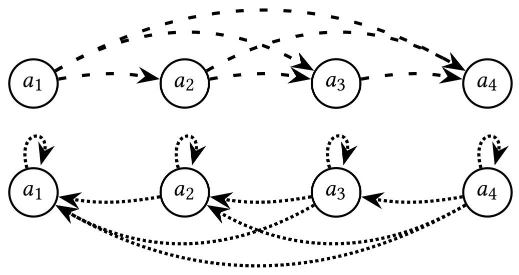


Finally, in the third round we fire

$$E^{I^-}(a_1, a_4) \vee \bigvee_{i\in\{1,2,3,4\}} \left(E^{I^+}(a_1, a_i) \wedge T^{I^+}(a_i, a_4)\right) \quad \to \quad T^{I^+}(a_1, a_4)$$

$$E^{I^-}(a_1, a_1) \wedge \bigwedge_{i\in\{1,2,3,4\}} \left(E^{I^-}(a_1, a_i) \vee T^{I^-}(a_i, a_1)\right) \quad \to \quad T^{I^-}(a_1, a_1)$$

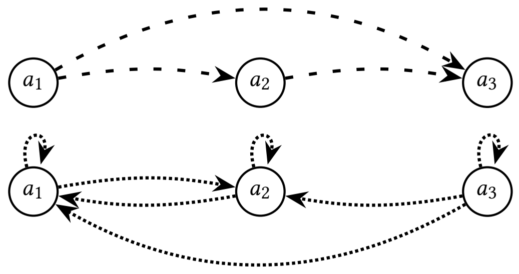
The first graph shows T, the transitive closure of E, and the second the complement of the transitive closure. Note that there is no need for any syntactical notion of stratification or non-monotonic reasoning.



Example 6.3. We recall the classical Win-Move program, where $Win(x)$ means player can win at vertex x, and $Move(x, y)$ means there is a move from vertex x to vertex y. The only rule of this game is

$$\exists y \left(Move(x, y) \wedge \neg Win(y)\right) \to Win(x).$$

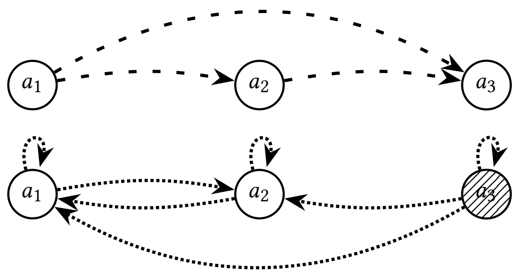
That is, a player can win in state x whenever he can move to state y and y is not a winning state. Below we show the relations $Move^{I^+}$ and $Move^{I^-}$ over $\mathbb{D} = \{a_1, a_2, a_3\}$. Initially we have $Win(a_i) = \bot$ for $i \in \{1, 2, 3\}$.



In the first round we fire

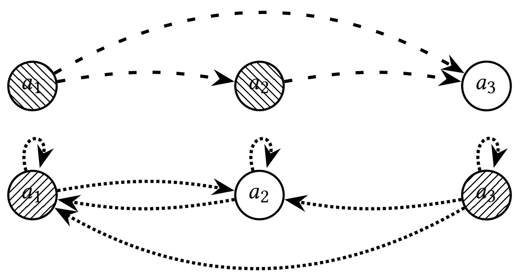
$$\bigwedge_{i\in\{1,2,3,4\}} \left(Move^{I^-}(a_3, a_i) \vee Win^{I^+}(a_i)\right) \to Win^{I^-}(a_3).$$

As a result, $Win^{I^-}(a_3)$ is set to 1 below.



In the second round we fire

$$\bigvee_{i\in\{1,2,3\}} \left(Move^{I^+}(a_2,a_i) \wedge Win^{I^-}(a_i)\right) \rightarrow Win^{I^+}(a_2)$$
$$\bigvee_{i\in\{1,2,3\}} \left(Move^{I^+}(a_1,a_i) \wedge Win^{I^-}(a_i)\right) \rightarrow Win^{I^+}(a_1)$$
$$\bigwedge_{i\in\{1,2,3\}} \left(Move^{I^-}(a_1,a_i) \vee Win^{I^+}(a_i)\right) \rightarrow Win^{I^-}(a_1)$$



There are few interesting notes to be made about Example 6.3. Firstly, regardless of the fact that $Move(1,2)$ is inconsistent in the input graph, the chase could be executed and at the end we have a model which satisfies all conditions. Secondly, inconsistent initial information can lead to inconsistent results as well. Lastly, even though we have inconsistent result for vertex a_1 our results are consistent for vertices a_2 and a_3 for which we are sure they are winning and loosing states, respectively.

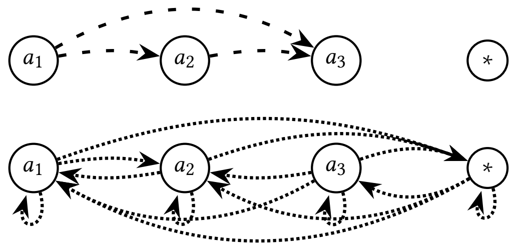
7 CHASING WITH INFINITE DOMAIN

In the previous section we assumed that the domain (of vertices) was finite. When we use star tables we however assumed that the domain is countably infinite. Returning to the Win-Move example, suppose the domain of vertices is $\mathbb{D} = \{a_1, a_2, a_3 \ldots, a_n, \ldots\}$, and the initial graph have edges $Move(a_1,a_2), Move(a_2,a_3)$, and $Move(a_1,a_3)$. We can store this, as well as the fact that there are no other move edges by the following two star-tables.

$Move^{I^+}$	
a_1	a_2
a_2	a_3
a_1	a_3

$Move^{I^-}$		
$*$	$*$	$(1 \neq a_1 \vee 2 \neq a_2) \wedge (1 \neq a_2 \vee 2 \neq a_3) \wedge (1 \neq a_1 \vee 2 \neq a_3)$

Below we show the relations $Move^{I^+}$ and $Move^{I^-}$ graphically. The vertex labeled * represents all vertices in $\mathbb{D} \setminus \{a_1, a_2, a_3\}$. Initially we have $Win^I(a_i) = \perp$ for all $a_i \in \mathbb{D}$.



In the first round we fire

$$\bigwedge_{i\in\{1,2,3,\ldots\}} \left(Move^{I^-}(a_3,a_i) \vee Win^{I^+}(a_i)\right) \rightarrow Win^{I^-}(a_3).$$

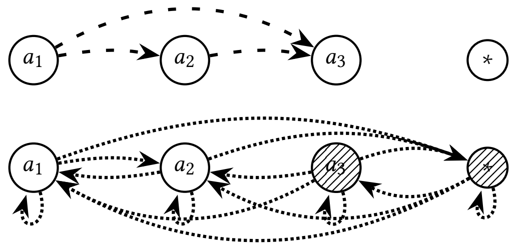
which is equivalent to

$$\bigwedge_{a\in\{a_1,a_2,a_3,*\}} \left(Move^{I^-}(a_3,a) \vee Win^{I^+}(a)\right) \rightarrow Win^{I^-}(a_3).$$

As a result, $Win^{I^-}(a_3)$ is set to 1. Then we fire

$$\bigwedge_{a\in\{a_1,a_2,a_3,*\}} \left(Move^{I^-}(*,a) \vee Win^{I^+}(a)\right) \rightarrow Win^{I^-}(*).$$

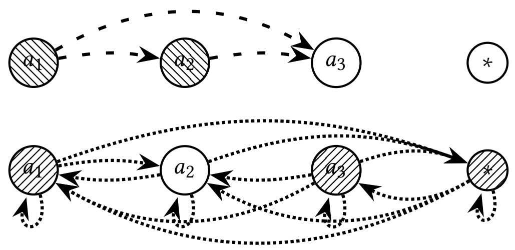
As a result, $Win^{I^-}(*)$ is set to 1 below, as well.



In the second round we fire

$$\bigvee_{a\in\{a_1,a_2,a_3,*\}} \left(Move^{I^+}(a_2,a) \wedge Win^{I^-}(a)\right) \rightarrow Win^{I^+}(a_2)$$
$$\bigvee_{a\in\{a_1,a_2,a_3,*\}} \left(Move^{I^+}(a_1,a) \wedge Win^{I^-}(a)\right) \rightarrow Win^{I^+}(a_1)$$
$$\bigwedge_{a\in\{a_1,a_2,a_3,*\}} \left(Move^{I^-}(a_1,a) \vee Win^{I^+}(a)\right) \rightarrow Win^{I^-}(a_1)$$

resulting in



As it can be seen, states a_1 and a_2 are winning states. States a_1, a_3, and $*$ are non-winning states. Clearly, state a_1 appear as winning and non-winning state, which makes it an inconsistent state. However, having inconsistent vertices and moves does not stop the four-valued chase from concluding consistent information about other states. Moreover, having state $*$ as a non-winning state declares that the set of states $a_4, a_5, a_6, \ldots$ are all non-winning states. In summary, the relations Win^{I^+} and Win^{I^-} contain the vertices $a \in \mathbb{D}$ for which $Win^I(a) = \mathtt{1}$ and $Win^I(a) = \mathtt{0}$, respectively. One should notice that inconsistency in the initial database can be propagated by the chase. However, consistent information is still treated correctly.

Adding existential nulls. In the paper [8] we showed that we can add labeled existential nulls to star-tables, and that these nulls can be evaluated naively (treating them as pairwise distinct constants, different from all constants in the database), and that as long as we only use FO$^+$-queries, the algebraic evaluation on star-tables with universal and existential nulls can be done efficiently. We note that we only need to deal with FO$^+$-queries, as FO-queries under four-valued semantics is processed as a pair of FO$^+$-queries. We also note that so far we have assumed that the interpretation of $\approx$ is the two-valued identity. When existential nulls enter the picture, we however get a four-valued interpretation of $\approx$. Initially $x \approx a$, where x is a null value and a a constant, is given truth-value unknown, and similarly for $x \approx y$ for existential nulls x and y. However, after enforcing equality generating dependencies we might derive facts $(x \approx a)^I \in \{\mathtt{1}, \mathtt{0}, \top\}$, and similarly for $(x \approx y)^I$. This means that we can decouple inconsistency resolution from query answering, in contrast with the intractable repair approach of Arenas et al. [1]. In the four-valued approach the answers to queries only flag certain tuples or attribute values as inconsistent. The consistent part of the answers, obtainable without computational penalty, then corresponds to the consistent answer obtained in the repair approach.

8 CONCLUSION

We have argued that Belnap's four-valued logic [2] is a perfect fit with relational databases, in that it allows an extension of the traditional technology to smoothly and efficiently handle both incomplete and inconsistent information. The required extension can be achieved using our newly introduced star-tables and star-algebra, where FO-queries and dependencies can be processed efficiently under the four-valued semantics.

REFERENCES

[1] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *Proceedings of the Eighteenth ACM*, pages 68–79, 1999.

[2] Jr. Belnap, Nuel D. A useful four-valued logic. In J.M̃ichael Dunn and George Epstein, editors, *Modern Uses of Multiple-Valued Logic*, volume 2 of *Episteme*, pages 5–37. Springer Netherlands, 1977.

[3] Joachim Biskup. Extending the relational algebra for relations with maybe tuples and existential and universal null values. *Fundam. Inform.*, 7(1):129–150, 1984.

[4] Chris Date. *Database in depth: relational theory for practitioners.* " O'Reilly Media, Inc.", 2005.

[5] Melvin Fitting. Kleene's logic, generalized. *J. Log. Comput.*, 1(6):797–810, 1991.

[6] Melvin Fitting. The family of stable models. *J. Log. Program.*, 17:197–225, 1993.

[7] Matthew Ginsberg. Multivalued logics: A uniform approach to inference in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.

[8] Gösta Grahne and Ali Moallemi. Universal (and existential) nulls. *arXiv preprint arXiv:1803.01445*, 2018. https://arxiv.org/abs/1803.01445.

[9] Gösta Grahne, Ali Moallemi, and Adrian Onet. Intuitionistic data exchange. In *9th Alberto Mendelzon International Workshop.*, 2015.

[10] Paolo Guagliardo and Leonid Libkin. A formal semantics of SQL queries, its validation, and applications. *PVLDB*, 11(1):27–39, 2017.

[11] Leon Henkin, J Donald Monk, and Alfred Tarski. *Cylindric Algebras–Part I.* North-Holland Publishing Company, 1971.

[12] Leon Henkin, J Donald Monk, and Alfred Tarski. *Cylindric Algebras–Part II.* North-Holland Publishing Company, 1985.

[13] Tomasz Imielinski and Witold Lipski Jr. The relational model of data and cylindric algebras. *J. Comput. Syst. Sci.*, 28(1):80–102, 1984.

[14] Stephen Cole Kleene. *Introduction to metamathematics.* D. Van Norstrand, 1952.

[15] Leonid Libkin. Negative knowledge for certain query answers. In *Web Reasoning and Rule Systems - 10th International Conference, 2016*, pages 111–127, 2016.

[16] Leonid Libkin. Sql's three-valued logic and certain answers. *ACM Trans. Database Syst.*, 41(1):1:1–1:28, 2016.