Universal Nulls (Extended Abstract)

Gösta Grahne and Ali Moallemi

Concordia University, Montreal, Canada
grahne@cs.concordia.ca,moa_ali@encs.concordia.ca

1 Introduction

In this paper we revisit the foundations of the relational model and unearth *universal nulls*, showing that they can be treated on par with the usual *existential nulls* [10,4,5,2] Recall that an existential null in a tuple in a relation R represents an existentially quantified variable in an atomic sentence $R(..)$. This corresponds to the intuition "value exists, but is unknown." A universal null, on the other hand, does not represent anything unknown, but stands for *all* values of the domain. In other words, a universal null represents a universally quantified variable. Universal nulls have an obvious application in databases, as the following example shows. The symbol "$*$" denotes a universal null.

Example 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 be the following.

F	
Alice	Chris
$*$	Alice
Bob	$*$
Chris	Bob

H	
Alice	Movies
Alice	Music
Bob	Basketball

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:

$$x_4 \,.\, \exists x_2 \exists x_3 \forall x_1 \Big(F(x_1, x_2) \wedge H(x_3, x_4) \wedge (x_2 \approx x_3) \Big). \tag{1}$$

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 $x_1, x_2 \,.\, F(x_1, x_2)$ would return all the tuples in F. ◂

Another area of applications of "*"-nulls relates to intuitionistic, or constructive database logic. In the constructive four-valued approach of [7] and the three-valued approach of [5,12] the proposition $A \vee \neg A$ is not a tautology. In order for $A \vee \neg A$ to be true, we need a constructive proof of A or a constructive proof of $\neg A$. Therefore both [7] and [12] assume that the database I has a theory of the negative information, i.e. that $I = (I^+, I^-)$, where I^+ contains the positive information and I^- the negative information. The papers [7] and [12] then show how to transform an FO-query $\varphi(\bar{x})$ to a pair of queries $(\varphi^+(\bar{x}), \varphi^-(\bar{x}))$ such that $\varphi^+(\bar{x})$ returns the tuples $\bar{a}$ for which $\varphi(\bar{a})$ is true in I, and $\varphi^-(\bar{x})$ returns the tuples $\bar{a}$ for which $\varphi(\bar{a})$ is true in I (i.e. $\varphi(\bar{a})$ is proven to be false in I). It turns out that databases containing "*"-nulls are suitable for storing I^-.

Example 2. Suppose that the instance in Example 1 represents I^+, and that all negative information we have deduced about the $H(obbies)$ relation, is that we know Alice doesn't play Volleyball that Bob only has Basketball as hobby, and that Chris has no hobby at all. This negative information about the relation H is represented by the table H^- below. Note that H^- is part of I^-.

H^-	
Alice	Volleyball
Bob	$*$ (except Basketball)
Chris	$*$

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 $\varphi^+ = x_1 \,.\, \exists x_2\, H^+(x_1, x_2)$, and $\varphi^- = x_1 \,.\, \forall x_2\, H^-(x_1, x_2)$. Evaluating φ^+ on I^+ returns {(Alice), (Bob)}, and evaluating φ^- on I^- returns {(Chris)}. Note that there is no closed-world assumption as the negative facts are explicit. Thus it is unknown whether someone called David has a hobby or not. ◂

This example highlights our main motivation for adding universal nulls to databases. When intuitionistic logic is used in answering queries with negation against four-valued database instances, universal quantifiers are necessary. In fact, even the simplest queries, such as Conjunctive Queries will be translated to a universally quantified disjunction of the atoms in the query. The cost of evaluating FO-queries φ intuitionistically on four-valued databases remains the same as evaluating φ with standard semantics on two-valued databases. Note however that φ^+ and φ^- are always negation free.

The Cylindric Set Algebra [8,9] —as an algebraization of first order logic— is an algebra on sets of valuations of variables in an FO-formula. A *valuation* ν of variables $\{x_1, x_2, \ldots\}$ can be represented as a tuple ν, where $\nu(i) = \nu(x_i)$. Any subset of all valuations can then be represented by a relation C of such tuples. In particular, if the FO-formula only involves a finite number n of variables, then the representing relation C has arity n. Note however that C can have an infinite number of tuples, since the domain, named $\mathbb{D}$ from now on, of the variables (such as the users of a social media site) should be assumed unbounded. One of the basic connections [8,9] between FO and Cylindric Set Algebra is that, given any interpretation I and FO-formula φ, the set of valuations ν under

which φ is true in I can be represented as such a relation C. Moreover, each logical connective and quantifier corresponds to an operator in the Cylindric Set Algebra. Naturally disjunction corresponds to union, conjunction to intersection, and negation to complement. More interestingly, existential quantification on variable x_i corresponds to *cylindrification* c_i on column i, defined as

$$\mathsf{c}_i(C) \;=\; \{\nu : \nu(i/a) \in C, \text{ for some } a \in \mathbb{D}\},$$

where $\nu(i/a)$ denotes the valuation (tuple) ν', and $\nu'(i) = a$ and $\nu'(j) = \nu(j)$ for $i \neq j$. The algebraic counterpart of universal quantification can be derived from cylindrification and complement, or be defined directly as *inner cylindrification*

$$\mathsf{c}_i(C) \;=\; \{\nu : \nu(i/a) \in C, \text{ for all } a \in \mathbb{D}\}.$$

In addition, in order to represent equality, the Cylindric Set Algebra also contains constant relations $d_{ij} = \{t \in \mathbb{D}^n : t(i) = t(j)\}$, representing the equality $x_i \approx x_j$. That is, d_{ij} is the set of all valuations ν, such that $\nu(i) = \nu(j)$.

The objects C and d_{ij} of [8,9] are of course infinitary. In this paper we therefore develop a finitary representation mechanism, namely relations containing universal nulls "$*$" and certain equality literals. These objects are called *Star Tables* when they represent the records stored in the database. When used as run-time constructs in algebraic query evaluation, they will be called *Star Cylinders*. Example 1 showed star-tables in a database. The run-time variable binding pattern of the query (1), as well as its algebraic evaluation is shown in the star-cylinders in Example 3 below.

Example 3. Continuing Example 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 star-cylinder C_{23}. Note that these are positional relations, the "attributes" x_1, x_2, x_3, 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	Basketball

C_{23}

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

The algebraic translation of query (1) is the SCA-expression

$$\dot{\mathsf{c}}_2(\dot{\mathsf{c}}_3(\dot{\mathsf{c}}_1((C_F \curlywedge C_H) \curlywedge C_{23}))) \tag{2}$$

The intersection of C_F and C_H is carried out as star-intersection $\curlywedge$, where for instance $\{(a, *, *)\} \curlywedge \{(*, b, *)\} = \{(a, b, *)\}$. The result will contain 12 tuples, and when these are star-intersected with C_{23}, the star-cylinder C_{23} will act as a selection by columns 2 and 3 being equal. The result is the left-most star-cylinder $C' = (C_F \curlywedge C_H) \curlywedge C_{23}$ below.

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	Basketball
Chris	Bob	Bob	Basketball

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

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 $C''' = \dot{c}_2(\dot{c}_3(\dot{\supset}_1((C_F \barwedge C_H) \barwedge C_{23})))$ 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. Negation requires some additional details that can be found in the full version [6]. ◂

The aim of this paper is to develop a clean and sound modeling of universal nulls, and furthermore show that the model can be seamlessly extended to incorporate the existential nulls of Imielinski and Lipski [10]. We show that FO and our Cylindric Star Algebra are equivalent in expressive power when it comes to querying databases containing universal nulls, and that our algebraic queries can be evaluated naively. This will be done in three steps: In Section 2 we show the equivalence between FO and *Cylindric Set Algebra (CA)* [8,9] over infinitary databases. This was of course only the starting point of [8,9], and we recast the result here in terms of database theory.[1] In Section 3 we introduce our finitary *Cylindric Star Algebra (SCA)* [2], and show the equivalence between CA and SCA, thus denomstrating that certain infinitary cylinders can be finitely represented as star-cylinders, and that our finitary Cylindric Star Algebra on finite star-cylinders mirrors the Cylindric Set Algebra on the infinite cylinders they represent.

At the end of Section 3 we we take the third step show how to tie these two results together, delivering the promised SCA evaluation of FO queries on databases containing universal nulls. In the full paper [6] we seamlessly extend our framework to also handle existential nulls, and show that naive evaluation can still be used for positive queries (allowing universal quantification, but not negation) on databases containing both universal and existential nulls. The full paper [6] shows that all SCA expressions can be evaluated in time polynomial in the size of the database when only universal nulls are present. In [6] we also show that when both universal and existential nulls are present, the certain answer to any negation-free (allowing inner cylindrification, i.e. universal quantification) SCA query can be evaluated naively in polynomial time. The full paper [6] also includes complexity results related to database and view containment for databases with universal nulls.

[1] Van Den Bussche [3] has recently referred to [8,9] in similar terms.

[2] In this extended abstract we only describe the positive case, where there is no negation in the query or database. The extension to include negation is developed in the full paper [6].

2 Relational calculus and cylindric set algebra

Throughout this paper we assume a fixed schema $\mathscr{R} = \{R_1, \ldots, R_m, \approx\}$, where each R_p, $p \in \{1, \ldots, m\}$, is a relational symbol with an associated positive integer $ar(R_p)$, called the *arity* of R_p. The symbol $\approx$ represents equality.

Logic. Our calculus is the standard domain relational calculus. Let $\{x_1, x_2, \ldots\}$ be a countably infinite set of *variables*. We define the set of *FO-formulas* φ (over $\mathscr{R}$) in the usual way: $R_p(x_{i_1}, \ldots, x_{i_{ar(R_p)}})$ and $x_i \approx x_j$ are atomic formulas, and these are closed under $\wedge, \vee, \neg, \exists x_i$, and $\forall x_i$, in a well-formed manner possibly using parenthesis's for disambiguation. If an FO-formula uses n variables, it will be called an FO_n-formula.

Instances. Let $\mathbb{D} = \{a_1, a_2, \ldots\}$ be a countably infinite *domain*. An *instance* I (over $\mathscr{R}$) is a mapping that assigns a possibly infinite subset R_p^I of $\mathbb{D}^{ar(R_p)}$ to each relation symbol R_p, and $\approx^I = \{(a, a) : a \in \mathbb{D}\}$. Note that our instances are infinite model-theoretic ones. The set of tuples actually recorded in the database will be called the *stored database* (to be defined in Section 3).

In order to define the (standard) notion of truth of an FO_n-formula φ in an instance I we first define a *valuation* to be a mapping $\nu : \{x_1, \ldots, x_n\} \to \mathbb{D}$. If ν is a valuation, x_i a variable and $a \in \mathbb{D}$, then $\nu_{(i/a)}$ denotes the valuation which is the same as ν, except $\nu_{(i/a)}(x_i) = a$. Then we use the usual recursive definition of $I \vDash_\nu \varphi$, meaning *instance I satisfying φ under valuation ν*, i.e. $I \vDash_\nu (x_i \approx x_j)$ if $(\nu(x_i), \nu(x_j)) \in \approx^I$, $I \vDash_\nu R_p(x_{i_1}, \ldots, x_{i_{ar(R_p)}})$ if $(\nu(x_{i_1}), \ldots, \nu(x_{i_{ar(R_p)}})) \in R_p^I$, and $I \vDash_\nu \exists x_i \varphi$ if $I \vDash_{\nu_{(i/a)}} \varphi$ for some $a \in \mathbb{D}$, and so on. Our stored databases will be finite representations of infinite instances, so the semantics of answers to FO-queries will be defined in terms of the infinite instances:

Definition 1. *Let I be an instance, and φ an FO_n-formula with $free(\varphi) = \{x_{i_1}, \ldots, x_{i_k}\}$, $k \leq n$. Then the answer to φ on I is defined as*

$$\varphi^I = \{(\nu(x_{i_1}), \ldots, \nu(x_{i_k})) : I \vDash_\nu \varphi\}.$$

Algebra. As noted in [11] the relational algebra is really a disguised version of the Cylindric Set Algebra of Henkin, Monk, and Tarski [8,9]. We shall therefore work directly with the Cylindric Set Algebra instead of Codd's Relational Algebra. Apart from the conceptual clarity, the Cylindric Set Algebra will also allow us to smoothly introduce the promised universal nulls.

Let n be a fixed positive integer. The basic building block of the Cylindric Set Algebra is an *n-dimensional cylinder* $C \subseteq \mathbb{D}^n$. Note that a cylinder is essentially an infinite n-ary relation. They will however be called cylinders, in order to distinguish them from instances. The rows in a cylinder will represent run-time variable valuations, whereas tuples in instances represent facts about the real world. We also have special cylinders called *diagonals*, of the form $d_{ij} = \{t \in \mathbb{D}^n : t(i) = t(j)\}$ representing the equality $x_i \approx x_j$. We can now define the Cylindric Set Algebra.

Definition 2. *Let C and C' be infinite n-dimensional cylinders. The* Cylindric Set Algebra *consists of set theoretic union $C \cup C'$, complement $\overline{C} = \mathbb{D}^n \setminus C$, diagonals $\mathsf{d}_{ij} = \{t \in \mathbb{D}^n : t(i) = t(j)\}$, and outer cylindrification:*

$$\mathsf{c}_i(C) = \{t \in \mathbb{D}^n \ : \ t(i/a) \in C, \ \textit{for some } a \in \mathbb{D}\}.$$

The operation c_i corresponds to existential quantification of variable x_i. For the geometric intuition behind the name cylindrification, see [8,11]. Intersection is considered a derived operator, and we also introduce the inner cylindrification as a derived operator $\mathsf{c}_i(C) = \overline{\mathsf{c}_i(\overline{C})}$, corresponding to universal quantification. Note that

$$\mathsf{c}_i(C) = \{t \in \mathbb{D}^n \ : \ t(i/a) \in C, \ \textit{for all } a \in \mathbb{D}\}.$$

Equivalence of FO and CA. In the next two theorems we will restate, in the context of the relational model, the correspondence between domain relational calculus and cylindric set algebra as query languages on instances [8,9]. An expression E in cylindric set algebra of dimension n will be called a CA_n-expression. When translating an FO_n-formula to a CA_n-expression we first need to extend all k-ary relations in I to n-ary by filling the $n-k$ last columns in all possible ways. The result is denoted $\mathsf{h}^n(I)$.

Once an instance is expanded it becomes a sequence $\mathbf{C} = (C_1, \ldots, C_m, d_{ij})_{i,j}$ of n-dimensional cylinders and diagonals, on which Cylindric Set Algebra Expressions can be applied. The main technical difficulty in the translation from FO_n to CA_n is the correlation of the variables in the FO_n-sentence φ with the columns in the expanded relations in the instance. This can be achieved using a derived "swapping" operator $\mathsf{z}^{i_1,\ldots,i_k}_{j_1,\ldots,j_k}$ that interchanges the columns i_l and j_l, where $l \in \{1,\ldots,k\}$.[3] Every atom R_p in φ will correspond to a CA_n-expression $\mathsf{C}_p = \mathsf{h}^n(R_p^I)$. However, for every occurrence of an atom $R_p(x_{i_1},\ldots,x_{i_k})$ in φ we need to interchange the columns $1,\ldots,k$ with columns $i_1,\ldots,i_k$. This is achieved by the expression $\mathsf{z}^{1,\ldots,k}_{i_1,\ldots,i_k}(\mathsf{C}_p)$. The entire FO_n-formula φ with $free(\varphi) = \{x_{i_1},\ldots,x_{i_k}\}$ will then correspond to the CA_n-expression $E_\varphi = \mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}(F_\varphi)$, where F_φ is defined recursively as follows:

- If $\varphi = R_p(x_{i_1},\ldots,x_{i_k})$ where $k = ar(R_p)$, then $F_\varphi = \mathsf{z}^{1,\ldots,k}_{i_1,\ldots,i_k}(\mathsf{C}_p)$.
- If $\varphi = x_i \approx x_j$, then $F_\varphi = \mathsf{d}_{ij}$.
- If $\varphi = \psi \vee \chi$, then $F_\varphi = F_\psi \bigcup F_\chi$, if $\varphi = \psi \wedge \chi$, then $F_\varphi = F_\psi \bigcap F_\chi$, and if $\varphi = \neg\psi$, then $F_\varphi = \overline{F_\psi}$.
- If $\varphi = \exists x_i \psi$, then $F_\varphi = \mathsf{c}_i(F_\psi)$.
- If $\varphi = \forall x_i \psi$, then $F_\varphi = \mathsf{c}_i(F_\psi)$.

For an example, let us reformulate the FO_4-query φ from (1) as

$$x_4 \ . \ \exists x_2 \exists x_3 \forall x_1 \Big(R_1(x_1,x_2) \wedge R_2(x_3,x_4) \wedge (x_2 \approx x_3) \Big).$$

[3] For a definition of swapping using primitive operators, see Definition 1.5.12 in [8].

When translating φ, the relation R_1^I is first expanded to $C_1 = R_1^I \times \mathbb{D} \times \mathbb{D}$, and R_2^I is expanded to $C_2 = R_2^I \times \mathbb{D} \times \mathbb{D}$. In order to correlate the variables in φ with the columns in the expanded relations, we do the shifts $\mathsf{z}_{1,2}^{1,2}(C_1)$ and $\mathsf{z}_{3,4}^{1,2}(C_2)$. The equality $(x_2 \approx x_3)$ was expanded to the diagonal $\mathsf{d}_{23} = \{t \in \mathbb{D}^n : t(2) = t(3)\}$ so here the variables are already correlated. After this the conjunctions are replaced with intersections and the quantifiers with cylindrifications. Finally, the column corresponding to the free variable x_4 in φ (whose bindings will constitute the answer) is shifted to column 1. The final CA_n-expression will then be evaluated against I as $E_\varphi(\mathsf{h}^4(I)) =$

$$\mathsf{z}_1^4\Big(\mathsf{c}_{23}(\mathsf{c}_1(\mathsf{z}_{1,2}^{1,2}(R_1^I \times \mathbb{D}^2) \bigcap \mathsf{z}_{3,4}^{1,2}(R_2^I \times \mathbb{D}^2) \bigcap d_{23}))\Big).$$

We now have $E_\varphi(\mathsf{h}^4(I)) = \mathsf{h}^4(\varphi^I)$. The following fundamental result follows from [8,9]. For the benefit of the readers who don't want to consult [8,9], an explicit proof (in the current notation) can be found in [6].

Theorem 1. *For all FO_n-formulas φ, there is a CA_n expression E_φ, such that $E_\varphi(\mathsf{h}^n(I)) = \mathsf{h}^n(\varphi^I)$, for all instances I.* ◂

The translation from CA_n-expressions to FO_n-formulas is fairly straightforward an can be found in the full version [6].

3 Cylindric Set Algebra and Cylindric Star Algebra

Since cylinders can be infinite, we want a finite mechanism to represent (at least some) infinite cylinders, and the mechanism to be closed under queries. Our representation mechanism comes in two variations, depending on whether negation is allowed or not. Here we only consider the positive (no negation) case. The full machinery is described in [6].

Star Cylinders. We define an *n-dimensional (positive) star-cylinder* $\dot{C}$ to be a finite set of *n-ary star-tuples*, the latter being elements of $(\mathbb{D} \cup \{*\})^n \times \wp(\Theta_n)$, where Θ_n denotes the set of all *equalities* of the form $i = j$, with $i, j \in \{1, \ldots, n\}$. Star-tuples will be denoted $\dot{t}, \dot{u}, \ldots$. A star-tuple such as $\dot{t} = (a, *, c, *, *, \{(4 = 5)\})$ is meant to represent the set of all "ordinary" tuples (a, x, c, y, y) where $x, y \in \mathbb{D}$. It will be convenient to assume that all our star-cylinders are in the following normal form.

Definition 3. *An n-dimensional star-cylinder $\dot{C}$ is said to be in* normal form *if $\dot{t}(n+1) \vDash (i = j)$ entails $(i = j) \in \dot{t}(n+1)$ and $\dot{t}(i) = \dot{t}(j)$, for all star-tuples $\dot{t} \in \dot{C}$.*

The symbol $\vDash$ above stands for standard logical implication. It is easily seen that maintaining star-cylinders in normal form can be done efficiently in polynomial time. We shall therefore assume without loss of generality that all star-cylinders and star-tuples are in normal form. We next define the notion of dominance, where a dominating star-tuple represents a superset of the ordinary tuples represented by the dominated star-tuple. First we define a relation $\preceq \subseteq (\mathbb{D} \cup \{*\})^2$ by $a \preceq a$, $* \preceq *$, and $a \preceq *$, for all $a \in \mathbb{D}$.

Definition 4. *Let $\dot{t}$ and $\dot{u}$ be n-dimensional star-tuples. We say that $\dot{u}$* dominates *$\dot{t}$, denoted $\dot{t} \preceq \dot{u}$, if $\dot{t}(i) \preceq \dot{u}(i)$ for all $i \in \{1, \ldots, n\}$, and $(i = j) \in \dot{u}(n+1)$ entails $(i = j) \in \dot{t}(n+1)$ when $\dot{t}(i) = \dot{t}(j) = *$, and entails $\dot{t}(i) = \dot{t}(j)$ otherwise.*

We complete the definition by stipulating that $\dot{t} \preceq \dot{u}$ whenever $\dot{t}(n+1) = \{\mathsf{false}\}$[4]. We can now define the meet $\dot{t} \wedge \dot{u}$ of star-tuples $\dot{t}$ and $\dot{u}$:

Definition 5. *Let $\dot{t}$ and $\dot{u}$ be n-ary star-tuples. If $\dot{t}(j), \dot{u}(j) \in \mathbb{D}$ for some j and $\dot{t}(j) \neq \dot{u}(j)$ then $\dot{t} \wedge \dot{u}\,(i) = a$ for $i \in \{1, \ldots, n\}$, and $\dot{t} \wedge \dot{u}(n+1) = \{\mathsf{false}\}$.* [5] *Otherwise, for $i \in \{1, \ldots, n\}$*

$$\dot{t} \wedge \dot{u}\,(i) = \begin{cases} \dot{t}(i) & \text{if } \dot{t}(i) \in \mathbb{D} \\ \dot{u}(i) & \text{if } \dot{u}(i) \in \mathbb{D} \\ * & \text{if } \dot{t}(i) = \dot{u}(i) = * \end{cases}$$

and $\dot{t} \wedge \dot{u}\,(n+1) = \dot{t}(n+1) \cup \dot{u}(n+1)$.

For an example, let $\dot{t} = (a, *, *, *, *, \{(3 = 4)\})$ and $\dot{u} = (*, b, *, *, *, \{(4 = 5)\})$. Then we have $\dot{t} \wedge \dot{u} = (a, b, *, *, *, \{(3 = 4), (4 = 5), (3 = 5)\})$. Note that $\dot{t} \wedge \dot{u} \preceq \dot{t}$, and $\dot{t} \wedge \dot{u} \preceq \dot{u}$. Note also that for n-ary star-tuples $\dot{t}_{\varnothing} = (a, a, \ldots, a, \{\mathsf{false}\})$ and $\dot{t}_{\mathbb{D}^n} = (*, *, \ldots, *, \{\mathsf{true}\})$, and for any n-ary star-tuple $\dot{t}$, it holds that $\dot{t} \wedge \dot{t}_{\varnothing} = \dot{t}_{\varnothing}$, $\dot{t} \wedge \dot{t}_{\mathbb{D}^n} = \dot{t}$, and $\dot{t}_{\varnothing} \preceq \dot{t} \preceq \dot{t}_{\mathbb{D}^n}$.

We extend the order $\preceq$ to include "ordinary" n-ary tuples $t \in \mathbb{D}^n$ by identifying $(a_1, \ldots, a_n)$ with star-tuple $(a_1, \ldots, a_n, \{\mathsf{true}\})$. Let $\dot{C}$ be an n-dimensional star-cylinder. We can now define the meaning of $\dot{C}$ to be the set $[\![\dot{C}]\!]$ of all ordinary tuples it represents, where $[\![\dot{C}]\!] = \{t \in \mathbb{D}^n : t \preceq \dot{u} \text{ for some } \dot{u} \in \dot{C}\}$. We lift the order to n-dimensional star-cylinders $\dot{C}$ and $\dot{D}$, by stipulating that $\dot{C} \preceq \dot{D}$, if for all star-tuples $\dot{t} \in \dot{C}$ there is a star-tuple $\dot{u} \in \dot{D}$, such that $\dot{t} \preceq \dot{u}$.

Lemma 1. *Let $\dot{C}$ and $\dot{D}$ be n-dimensional (positive) star-cylinders. Then it holds that $[\![\dot{C}]\!] \subseteq [\![\dot{D}]\!]$ iff $\dot{C} \preceq \dot{D}$.*

Positive Cylindric Star Algebra. Next we redefine the positive cylindric set operators so that $[\![\dot{C} \mathbin{\dot{\circ}} \dot{D}]\!] = [\![\dot{C}]\!] \circ [\![\dot{D}]\!]$ or $\circ([\![\dot{C}]\!]) = [\![\dot{\circ}(\dot{C})]\!]$, for each positive cylindric operator $\circ$, its redefinition $\dot{\circ}$, and star-cylinders $\dot{C}$ and $\dot{D}$.

Definition 6. *The* positive cylindric star-algebra *consists of the operators*

1. Star-diagonal: $\dot{d}_{ij} = \{(*, \ldots, *, (i = j))\}$
2. Star-union: $\dot{C} \cup \dot{D} = \{\dot{t} : \dot{t} \in \dot{C} \text{ or } \dot{t} \in \dot{D}\}$
3. Star-intersection: $\dot{C} \cap \dot{D} = \{\dot{t} \wedge \dot{u} : \dot{t} \in \dot{C} \text{ and } \dot{u} \in \dot{D}\}$
4. Outer cylindrification: *Let $i \in \{1, \ldots, n\}$, let $\dot{C}$ be an n-dimensional star-cylinder, and $\dot{t} \in \dot{C}$. Then*

$$\dot{\mathsf{c}}_i(\dot{t})(j) = \begin{cases} \dot{t}(j) & \text{if } j \neq i \\ * & \text{if } j = i \end{cases}$$

[4] Note that only in this case $\dot{t}$ is not in normal form.

[5] Here a is an arbitrary constant in $\mathbb{D}$.

for $j \in \{1, \ldots, n\}$, *and*

$$\dot{\mathsf{c}}_i(\dot{t})(n+1) = \{(j = k) \in \dot{t}(n+1) : j, k \neq i\}.$$

We then let $\dot{\mathsf{c}}_i(\dot{C}) = \{\dot{\mathsf{c}}_i(\dot{t}) : \dot{t} \in \dot{C}\}$.

5. Inner cylindrification: *Let* $\dot{C}$ *be an* n*-dimensional cylinder and* $i \in \{1, \ldots, n\}$. *Then* $\dot{\mathsf{d}}_i(\dot{C}) = \{\dot{t} \in \dot{C} : \dot{t}(i) = * \text{ and } (i = j) \notin \dot{t}(n+1) \text{ for any } j\}$.

The class of positive SCA_n- and CA_n-expressions will be denoted SCA_n^+ and CA_n^+. We illustrate the positive cylindric star-algebra with the following small example.

Example 4. Let $\dot{C}_1 = \{(a, *, *, *, *, \{(3 = 4)\})\}$, $\dot{C}_2 = \{(*, b, *, *, *, \{(4 = 5)\})\}$, $\dot{C}_3 = \{(a, b, *, *, *, \{(4 = 5)\})\}$, and consider $\dot{\mathsf{d}}_3((\dot{\mathsf{c}}_{1,4}(\dot{C}_1 \barwedge \dot{C}_2)) \barvee \dot{C}_3)$. Then we have the following.

$$\begin{aligned}
\dot{C}_1 \barwedge \dot{C}_2 &= \{(a, b, *, *, *, \{(3 = 4), (4 = 5), (3 = 5)\})\} \\
\dot{\mathsf{c}}_{1,4}(\dot{C}_1 \barwedge \dot{C}_2) &= \{(*, b, *, *, *, \{(3 = 5)\})\} \\
(\dot{\mathsf{c}}_{1,4}(\dot{C}_1 \barwedge \dot{C}_2)) \barvee C_3 &= \{(*, b, *, *, *, \{(3 = 5)\}) \\
&\qquad (a, b, *, *, *, \{(4 = 5)\})\} \\
\dot{\mathsf{d}}_3((\mathsf{c}_{1,4}(\dot{C}_1 \barwedge \dot{C}_2)) \barvee C_3) &= \{(a, b, *, *, *, \{(4 = 5)\})\}
\end{aligned}$$

Next we show that the cylindric star-algebra has the promised property.

Theorem 2. *For every* SCA_n^+*-expression* $\dot{E}$ *and the corresponding* CA_n^+ *expression* E, *it holds that* $[\![\dot{E}(\dot{\mathbf{C}})]\!] = E([\![\dot{\mathbf{C}}]\!])$ *for every sequence of* n*-dimensional star-cylinders and star-diagonals* $\dot{\mathbf{C}}$.

Stored databases with universal nulls. We now show how to use star cylindric algebra to evaluate FO-queries on stored databases containing universal nulls. Let k be a positive integer. Then a k-ary star-relation $\dot{R}$ is a finite set of star-tuples of arity k. In other words, a k-ary star-relation is a star-cylinder of dimension k. A sequence $\dot{\mathbf{R}}$ of star-relations (over schema $\mathbf{R}$) is called a *stored database*. Examples 1 and 3 show stored databases. A stored database $\dot{\mathbf{R}}$ represents the infinite instance $[\![\dot{\mathbf{R}}]\!] = ([\![\dot{R}_1]\!], \ldots, [\![\dot{R}_m]\!], \{(a, a)\}_{a \in \mathbb{D}})$.

Everything that is defined for star-cylinders applies to k-ary star-relations. The major exception is that no operators from the cylindric star-algebra are applied to star-relations. To do that, we first need to expand the stored database $\dot{\mathbf{R}}$, by filling columns $n-k \ldots n$ with $*$ in all k-ary relations, and similarly for the diagonals. The result is denoted $\dot{\mathsf{h}}^n(\dot{\mathbf{R}})$. We are now ready for our main result.[6]

Theorem 3. *For every* FO_n*-formula* φ *there is an* SCA_n *expression* $\dot{E}_\varphi$, *such that for every stored database* $\dot{\mathbf{R}}$, *we have* $\mathsf{h}^n(\varphi^{[\![\dot{\mathbf{R}}]\!]}) = [\![\dot{E}_\varphi(\dot{\mathsf{h}}^n(\dot{\mathbf{R}}))]\!]$.

[6] It might be argued for some applications that the "*"-value should range over the finite active domain only. This can be achieved in the positive framework by requiring for each star-cylinder $\dot{C}$ (expanded star-relation), and each star-tuple $\dot{t} \in \dot{C}$, that if $\mathsf{c}_i([\![\dot{t}]\!]) \subseteq [\![\dot{C}]\!]$, then $\dot{C}$ also contains a star-tuple $\dot{u}$, where $\dot{u} = \dot{t}(i/*)$.

4 Related and future work

Universal nulls were first studied in the early days of database theory by Biskup in [2]. This was a follow-up on his earlier paper on existential nulls [1]. The problem with Biskup's approach, as noted by himself, was that the semantics for his algebra worked only for individual operators, not for compound expressions (i.e. queries). This was remedied in the foundational paper [10] by Imielinski and Lipski, as far as existential nulls were concerned. Universal nulls next came up in [11], where Imielinski and Lipski showed that Codd's Relational Algebra could be embedded in CA, the Cylindric Set Algebra of Henkin, Monk, and Tarski [8,9]. As a side remark, Imielinski and Lipski suggested that the semantics of their "$*$" symbol could be seen as modeling the universal null of Biskup, and proposed a simplified version of the star-cylinders corresponding to the structures in *Diagonal-free Cylindric Set Algebras* [8,9]. The exact FO-expressive power of the finite diagonal-free star-cylinders is an open question. Nevertheless, using the techniques of [6], it can be shown that naive existential nulls can be seamlessly incorporated in diagonal-free star-cylinders.

References

1. Joachim Biskup. A foundation of codd's relational maybe-operations. *ACM Trans. Database Syst.*, 8(4):608–636, 1983.
2. Joachim Biskup. Extending the relational algebra for relations with maybe tuples and existential and universal null values. *Fundam. Inform.*, 7(1):129–150, 1984.
3. Jan Van den Bussche. Applications of Alfred Tarski's ideas in database theory. In *Computer Science Logic, 15th International Workshop, Paris, France, September 10-13, 2001*, pages 20–37, 2001.
4. Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. Reverse data exchange: coping with nulls. In *PODS*, pages 23–32, 2009.
5. Amélie Gheerbrant, Leonid Libkin, and Cristina Sirangelo. Naïve evaluation of queries over incomplete databases. *ACM Trans. Database Syst.*, 39(4):31:1–31:42, December 2014.
6. G. Grahne and A. Moallemi. Universal (and Existential) Nulls. *ArXiv e-prints*, March 2018.
7. Gösta Grahne, Ali Moallemi, and Adrian Onet. Intuitionistic data exchange. In *Proceedings of the 9th Alberto Mendelzon International Workshop on Foundations of Data Management, Lima, Peru, May 6 - 8, 2015.*, 2015.
8. Leon Henkin, J Donald Monk, and Alfred Tarski. *Cylindric Algebras–Part I,volume 64 of Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Company, 1971.
9. Leon Henkin, J Donald Monk, and Alfred Tarski. *Cylindric Algebras–Part II, volume 115 of Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Company, 1985.
10. Tomasz Imielinski and Witold Lipski Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.
11. Tomasz Imielinski and Witold Lipski Jr. The relational model of data and cylindric algebras. *J. Comput. Syst. Sci.*, 28(1):80–102, 1984.
12. Leonid Libkin. Negative knowledge for certain query answers. In *Web Reasoning and Rule Systems - 10th International Conference, 2016*, pages 111–127, 2016.