Intuitionistic Databases and Cylindric Algebra

Ali Moallemi

A thesis

In the Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy (Computer Science) at

Concordia University

Montréal, Québec, Canada

February 2019

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Ali Moallemi

Entitled: Intuitionistic Databases and Cylindric Algebra

and submitted in partial fulfillment of the requirements for the degree of

Doctor Of Philosophy (Computer Science)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Robin Drew	Chair
Dr. Leopoldo Bertossi	External Examiner
Dr. Robert Raphael	External to Program
Dr. Lata Narayanan	Examiner
Dr. Jamal Bentahar	Examiner
Dr. Gosta Grahne	Thesis Supervisor

Approved by ______________________________

Dr. Volker Haarslev, Graduate Program Director

April 29, 2019

Dr. Amir Asif, Dean
Gina Cody School of Engineering & Computer Science

Abstract

Intuitionistic Databases and Cylindric Algebra

Ali Moallemi, Ph.D.

Concordia University, 2019

The goal of this thesis is to introduce a logical view of databases based on four-valued logic, to revisit the foundations of the relational model and unearth universal nulls, and to handle finitely representable cases of infinite databases. In order to achieve this, we develop an intuitionistic relevance-logic based semantics that allows us to handle Full First Order queries similar to monotone First Order queries.

Universal Null represents all the possible values in the Domain. Next, we fully investigate the relational model and universal nulls, showing that they can be treated on par with the usual existential nulls. To do so, we show that a suitable finite representation mechanism, called Star-Cylinders, handling universal nulls can be developed based on the Cylindric Set Algebra of Henkin, Monk and Tarski. We provide a finitary version of the Cylindric Set Algebra, called Star Cylindric Algebra, and show that our star-cylinders are closed under this algebra.

Moreover, we show that any First Order Relational Calculus query over databases containing universal nulls can be translated into an equivalent expression in our star cylindric algebra, and vice versa. All star cylindric algebra expressions can be evaluated in time polynomial in the size of the database. Furthermore, the representation mechanism is then extended to Naive Star-Cylinders, which are star-cylinders allowing existential nulls in addition to universal nulls.

Beside the theory part, we also provide a practical approach for four-valued databases. We show that the four-valued database instances can be stored as a pair of two-valued instances. These two-valued instances store positive and negative information independently, in format of current databases. This procedure is called decomposition. Decomposed instances can losslessly be merged to form the initial four-valued instance. In a similar way, we show that four-valued queries can be decomposed to two-valued queries and can be executed against decomposed instances to obtain the four-valued result, after merging them back.

Later, we show how these results can be extended to Datalog and we show that there is no need for any syntactical notion of stratification or non-monotonic reasoning, when the intuitionistic logic is implemented. This is followed by presenting the complexity results. For positive queries (with universal quantification), the well known naive evaluation technique can still be applied on the existential nulls, thereby allowing polynomial time evaluation of certain answers on databases containing both universal and existential nulls. If precise answers are required, certain answer evaluation with universal and existential nulls remains in coNP. Note that the problem is coNP-hard, already for positive existential queries and databases with only existential nulls. If inequalities $\neg(x_i \approx x_j)$ are allowed, reasoning over existential databases is known to be Π_2^p-complete, and it remains in Π_2^p when universal nulls and full first order queries are allowed. Finally, we discuss related and future works.

Acknowledgments

First and foremost I want to thank my advisor Dr. Gösta Grahne. It has been an honor to be his Ph.D. student. He has taught me, both how to work professionally in academia and how to be a better human being. I appreciate all his contributions of time, ideas, and funding to make my Ph.D. experience productive and pleasant. The joy and enthusiasm he has for her research was contagious and motivational for me, even during tough times in the Ph.D. pursuit.

I also wish to express my sincere gratitude to Dr. Lata Narayanan, Dr. Jamal Bentahar and Dr. Robert Raphael for serving on my doctoral research committee, for their availability and interest in the work, for their valuable comments and feedbacks at different stages of this thesis.

The members of the database research lab have contributed immensely to my personal and professional time at Concordia. The group has been a source of friendships as well as good advice and collaboration. I have had the pleasure to work with or alongside of Adrian Onet, Shahab Harrafi, Iraj Hedayati, Zahra Asadi, Wei Han, Aminata Kane, Rahmah Brnawy, and Ali Alizadeh Mansouri.

Lastly, I would like to thank my family for all their unconditional love and encouragement. Words cannot express how grateful and appreciative I am of my parents who raised me with a love of science and supported me in all my pursuits, and for their efforts to provide me with the best possible education.

This research was supported in part by Natural Sciences and Engineering Research Council (NSERC) of Canada and Concordia University.

Contents

List of Figures

1 Introduction

The central notion of this thesis, the *Intuitionistic Databases*, is a solution to a common and rather persistent problems in databases. In database theory and application, inconsistent and unknown information are the main source of confusion. The problem arises, for instance when data is collected from different sources. Data integration or in a more general term, information integration involves synthesizing information across heterogeneous and often distributed data sources.

In data integration, mappings and database rules specify how the information from different source schemas is to be integrated in the target schema. There are three fundamental problems to address as the consequence of data integration. (i) Integrated information can lead to contradictory information about the same fact or facts. This causes the inconsistency in target database. (ii) Incomplete information which caused by lack of information on attributes, which are considered as *Null values*. There is another type of incomplete information where for a given tuple in a database, there is no information indicating to prove or disprove that the tuple belongs to the database instance. These cases are considered as *Unknown facts*. (iii) The Negation problem. Negation in current databases is considered as the lack of true information about the fact. However, we show that this is sometimes not sufficient and there is a need for constructive proofs for negative facts as well as for positive facts. These three crucial problems lead us toward the intuitionistic databases, inspired by Intuitionistic-Relevance Logic.

In this chapter, we review the required concepts and structures. The relational model is explained as the basic structure of databases. Next, in this chapter we go through data exchange, and information integration, followed by some examples in order to demonstrate the problems to tackle in this field. The intuitionistic logic and four-valued logic introduced by N. Belnap [11] is our desired framework that we apply to relational databases. In order to implement it, we cover the Cylindric Algebra and the notion of universal nulls. Finally, we need the Star Cylindric Algebra in order to tame infinite databases in finite tables.

1.1 The Relational Model

The relational model introduced by Codd [15] is based on the mathematical notion of a relation. This structure stores data in one or more relations (or tables) of rows and columns. Each row in a relation indicates a tuple (or fact) in the database. Columns in a relation are considered as attributes of the facts. Relational databases are designed to store sets of tuples with the same attributes. Each set of tuples with the same attributes is called a relation, and a set of relations form a database. A database schema describes the structure of instances in terms of relation and attribute names.

To access the information in the database, applications use queries. Formally, a query is a mathematical expression which filters and restructures the desired tuples and returns them as its answer. First Order Logic and datalog (queries with recursion) are considered in this research.

SQL (Structured Query Languages) is a query language specifically useful for relational databases due to their structured and fixed schema. SQL is based upon the domain-independent relational calculus and which makes SQL a declarative query language. Being a declarative query language, SQL lets user to express the filtered data to retrieve. However, in order to speed up the process of answering queries, it is important to have a plan for query execution. Relational Algebra provides the means to specifically plan the steps required to handle data in order to answer a query, which makes Relational Algebra a procedural query language.

Codd's theorem states that the domain-independent relational calculus and relational algebra are equivalent. This implies that, they have the same expressive power. Therefore, the database engine can have query execution plans for each individual operator, which makes the DBMS fast on query execution. As a result, SQL became the most dominant query language for the last few decades since it was introduced by Codd.

Example 1. *Assume that a company wants to store information of its employees in a relational database. Relation $EMP(a_1, a_2)$ states that a_1 is the employee's ID, and a_2 is his name. Relation $SAL(a_1, a_2)$ states that a_1 is the employee's ID, and a_2 is his salary in dollars. The following tables show an instance. The table in the left represents facts in EMP relation. The first fact expresses the employee named Bob with ID 123. The second tuple states the information about the employee named Josh with ID 124. The table in the right represents facts in SAL relation. The first fact shows the employee with ID 123 has a contract for $60k$ dollars. The second tuple states the information about the employee with ID 124 who is paid $80k$ dollars.*

EmployeeID	***EmployeeName***
123	*Bob*
124	*Josh*

EmployeeID	***EmployeeSalary***
123	*60000*
124	*80000*

Now assume, one is interested to acquire the name of employees with salary more than $70k$. In SQL syntax, it would be,

```
SELECT EmployeeNAME
FROM  EMP, SAL
```

```
WHERE EMP.EmployeeID = SAL.EmployeeID
AND SAL.EmployeeSalary > 70000;
```

This query will return "Josh" as its answer as expected.

Equivalently, it can be written as

$$\Big\{x_2.\exists x_1 \exists x_3\big(EMP(x_1,x_2) \wedge SAL(x_1,x_3) \wedge (x_3 > 70000)\big)\Big\}$$

in Relational Calculus, and

$$Q \coloneqq \pi_{EmployeeNAME}\Big(\sigma_{EmployeeSalary>70000}\big(EMP \bowtie_{(EMP.EmployeeID=SAL.EmployeeID)} SAL\big)\Big)$$

in Relational Algebra.

Ever since relational databases were introduced, they have been extremely popular and many different applications and problems over this data structure were introduced. One of the basic problems in relational databases is the *relational data transformation* problem, which is called *Data Exchange*.

1.2 Data Exchange in the Relational Model

There are a number of important transformation problems including data integration, data exchange, peer data exchange, peer-to-peer data sharing, schema integration, schema evolution and data repair. In this research we will consider one of data transformation processes that has gained lots of attention in the last few years, that is, the relational data exchange.

Relational data exchange was widely studied since this model of exchange was proposed in 2003 by Fagin, Kolaitis, Miller and Popa [23]. The model proposed by Fagin et al. considers the transformation of instances structured on a schema, called source schema, into new instances structured on a distinct schema, called target schema. This transformation is driven by some constraints that the transformed instance must satisfy relative to the source instance. Most of the time these constraints are called schema mappings and are expressed using some logical formalism. The problem of data exchange was extended to nested, XML, and DTDs schemas of semi-structured data. However, our focus is toward relational data exchange, because of the properties of relational model.

To be more precise about data exchange, in distributed information processing environments a connection is from a *source database* to a *target database*. This process is called *schema mapping* that guides the middle-ware in restructuring the data from the source database to fit the requirements of the target database and *target dependencies*, specifying the set of constraints that needs to be satisfied by the target instance. Since its inception by Fagin et al. in [23], the field of relational data exchange, and specifically schema mapping has been intensely investigated, [25, 26, 55, 21, 24, 27, 10, 9, 22, 47, 45, 46], and many functionalities are mature for technology transfer.

Figure 1 is a graphical representation of the data exchange problem, where **S** and **T** represent the source and target schemas, the input source instance is represented by I, Σ_{st} the set of source-to-target dependencies, Σ_t the set target dependencies, and next J is the target instance that needs to be computed. Such target instance J is called a solution for the data exchange problem. Then, user can inquire query q against the target database instance J, and the answer $q(J)$ will be returned to the user.

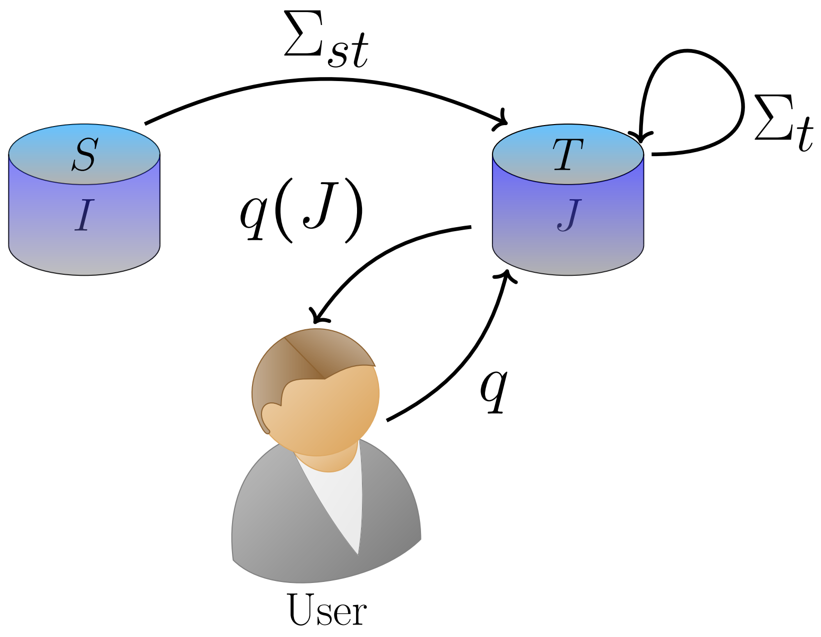


Figure 1: Data Exchange and Target Dependencies

1.2.1 The Application Area

In terms of applications, the problem of data exchange poses one of the major challenges in distributed information processing environments. In the last decade, the number of data exchange applications has increased. Many of today's data-centric applications rely on schema mappings, specially, the web data and the growth of data communication in between companies and research institutes, due to inevitable collaboration required to analysis big data, made data exchange certain to happen.

Despite numerous works in data exchange, there are still fundamental problems left unsolved or ill-treated. Problems such as how a data base should deal with *inconsistency* and *incomplete* information, or even the very the basic problem of how to interpret negation. In particular, one might ask what are the certain *negative* answers to a query, similar to certain (positive) answers.

1.3 Problem Statement

Incompleteness in databases led to the notion of certain answers. In presence of incomplete information in some attribute values, null values are used to represent the missing values. As a result, the concept of possible worlds arises by considering all possible instances that can be obtained from assigning different values to the nulls. Therefore, certain answers are considered as the answers common in all possible worlds. In fact, certain answers is a the set of answers to a query against an incomplete database instance which are surely true. To be able to represent the incomplete database instances, *naive tables* were introduced. In [49] naive tables allow variables in the relational databases, in addition to constants. Therefore, the unknown attribute values can be also represented in relational databases.

These problems have been shown to admit efficient implementation, based on a property colloquially called the *naive evaluation property.* The property roughly says that the incompleteness of some domain values can be ignored, as long as these values are distinguished from each other, and from the ”ordinary” domain values that denote named and known objects. The price to pay is that we have to restrict the schema mappings and target queries to be monotone. Most attempts to include non-monotone features, such as negation ($\neg$), soon run into intractability barriers, due to the underlying issues of incomplete information. For a comparison between different closed world semantics the reader should consult [60].

In the case of incomplete information, a database misses some information. It is easy to store missing or incomplete information, by simply using a symbol, say $\perp$, considered as constant different from constants in the domain of database. The difficulty arises when queries are applied to an incomplete database. For instance, suppose the query asks for all employees that their information are gathered from source S_1, while the database is as following:

EmployeeID	**EmployeeName**	**Source**
111	Alex	$\perp_1$
123	Bob	S_1
123	Josh	S_2
124	Max	$\perp_1$

Should "Alex" be included in the answers, as it is not ruled out that his information came from S_1? His unknown source of information could very well be S_1. It seems that "Alex"'s information *might* come from S_1. Similarly, for the case of Max.

Furthermore, let there be another query, asking for employees whose data come from the same source. Although it is not clear that where the information for Alex and Max come from, it is clear that they are coming from the same source, so they are associated with the same null, namely $\perp_1$. Moreover, adding negation to queries leads to intractable query evaluation [32].

The next problem that comes with information integration is the *inconsistency problem*. An inconsistent database has conflicting facts. Inconsistency can arise from collecting information from multiple sources with conflicting facts, or just from the uncertain nature of the available data. Inconsistency in databases is often captured as violation of integrity constraints, such as key functional dependencies. For example, lets have a key functional dependency,

$$\text{EmployeeID} \rightarrow \text{EmployeeName}$$

which says that, having an employee ID, the name of employee can be uniquely retrieved. Assume we have the following database:

EmployeeID	**EmployeeName**	**Source**
123	Bob	S_1
123	Josh	S_2

Here, we have two EmployeeNames associated with a single EmployeeID, which definitely conflicts the key functional dependency.

Researchers, in order to resolve the inconsistency in the databases suggested the data repair solution [3, 8, 6]. Data repair is the transformation process applied to an inconsistent database instance such that the resulting instance is consistent and it differs "minimally" from the initial instance. The "certain consistent answer" is obtained by query posed on each maximal consistent subsets of the database (there might be an exponential number of such subsets). This approach is computationally intractable in general. Even in the case queries can be answered efficiently, unique repairs are not guaranteed, and it might lead to loss of information. As Fitting [30] (page 10) puts it: "if we have inconsistent information about ducks, it is possible that our information about decimals can still be trusted". Intuitively, it seems plausible that you might throw out too much information (the decimals along with the ducks) in the repair approach. Therefore, there is a need for more investigation to handle the problem in a more convenient way.

Based on Annotated Predicate Calculus of [53], Arenas et. al. [7] presented a semantic framework for studying the problem of query answering in databases that are inconsistent with integrity constraints. Later, [31] used rewriting for query answering which is restricted to subclasses of conjunctive queries with existential quantifier in presence of key constraints.

There is a need to resolve these problems, the four-valued logic, and its translation is proposed this theses. The goal is to be able to (i) distinguish Unknown facts from False facts, (ii) to store inconsistency in databases, (iii) deal with negation in queries in most natural way.

In order to overcomer these difficulties this research implements the four valued Belnap logic, on top of currently existing database management systems. Databases and queries are decomposed from Four-valued into a pair of Two-valued, separately. These leads to universal quantifier in negative part of queries and universal nulls in decomposed databases. Cylindric algebra and star-cylinders are used and modified to cope with universal quantifier, and universal null over an infinite database. Star-cylinders provide a finite representation of an infinite database, which is a finite databases compactly expressing an infinite database. Finally, result are composed back into four-valued.

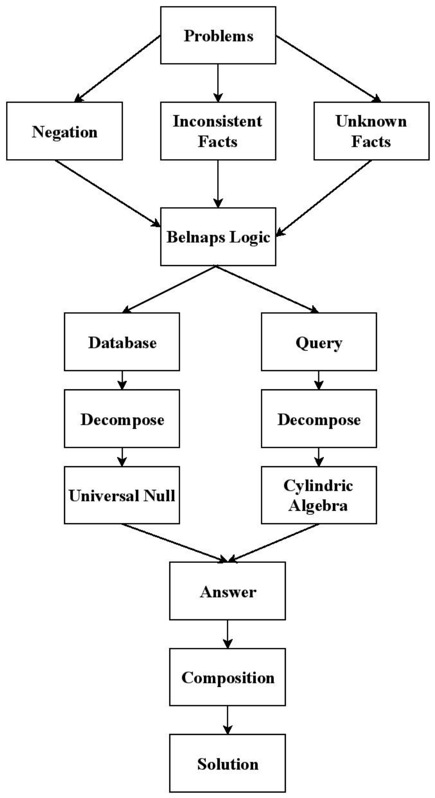


Figure 2: Methodology

Unknown facts and unknown values are discussed in the next section, where Belnap's four-valued relevance logic is explained.

1.4 Belnap's Logic

In presence of incomplete and inconsistent tuples, the Boolean logic with only true and false as truth values is not sufficient to express the database. In this spirit, we propose to use Nuel Belnap's four-valued relevance logic **R** [11] as foundation for negation in databases. Belnap's logic also is *paraconsistent*, meaning that "if we have inconsistent information about ducks, it is possible that our information about decimals can still be trusted" (see Fitting [30], page 10). Therefore it can cope with inconsistency in the most natural way. Beside inconsistency in database, there is a possibility for unknown facts. Unknown facts are those that there is no solid evidences for their truth which makes them neither true nor false. Similarly, for the case of unknown information, the four valued logic can express unknown facts.

As an interesting fact, within the Intuitionistic Logic the law of excluded middle does not hold. Therefore, the statement $(A \vee \neg A)$ is not a tautology. These features allow us to efficiently manage inconsistency, lack of information and incompleteness, which is not in general possible in the *repair*-approach of [8] or in traditional data exchange [40].

In the 1970's Nuel Belnap [11] extended Kleene's strong three-valued logic [54] with a fourth truth-value "inconsistent." Unlike the commonly known two-valued logic, with only true and false as truth values, the four-valued logic has two more truth values, namely, *Unknown* and *Inconsistent*. The unknown fact represents the situation where there is no information about the given fact. The inconsistent truth value declares that there are contradictory information about the same fact, that is from some sources it is admitted to be true while other sources state that the fact is false. This leads to inconsistency stemming from contradictory information from different sources.

Furthermore, the four-valued logic provides the structure for implementation of relevance logic. In the relevance logic, a constructive proof is required for a fact to be true as well as being false. In contrast, in two-valued logic, under the *Closed World Assumption (CWA)*, which is widely used in databases, there is a need of reasoning for a fact to be true, while lack of enough evidences to prove the fact makes it false. In constructive logic, in presence of negation or disjunction, lack of proof for negative facts is not sufficient to infer that the fact is false. Also, in case of dealing with incomplete information, it becomes problematic, since current approaches become computationally intractable.

In relevance logic true and false are not complementary. Since sources are integrated independently, there are cases where contradictory information is aggregated or there is no information. This makes the relevance logic a perfect suit for the Data Exchange Problem. Therefore, there are statements, which are not true, nor false, or in some cases, deduced to obtain both truth values. Thus, a need for extra truth values is essential. This leads us to choose four-valued logic as an excellent nominee for this approach.

In four-valued approach, the negative information can be explicitly specified within the logic and stored in the negative part of the database. This makes the extra-logical notion of closed worlds semantics unnecessary. In fact, a database is allowed to have closed, partially closed or open relations. Therefore, we show in this research 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. [8].

1.4.1 Belnap's Logic in literature

As mentioned above, the four-valued logic was defined by Nuel Belnap in 1977 [11]. 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* [34]. Follow ups are too numerous to survey here, but the work of Melvin Fitting (see e.g. [28, 29, 30]) offers a systematic overview. The adaptation of Belnap's logic to data exchange was described in [39], where the notion of negative answer was defined. Negative answers was also the topic of Libkin's paper [56]. Guagliardo and Libkin [58, 42] 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 [17].

Part of this work was presented in [39] which proposes an intuitionistic data exchange framework, in order to resolve this problem. The proposed framework considers the positive and negative information as independent. In this framework, a piece of data is considered to be true, if there is a constructive proof (based on existing information) showing its truth, and is considered as false, if there is a constructive proof (based on existing information) showing its falsity.

1.4.2 Implementation of Four Valued logic

Later, in chapter 4 we show that four-valued databases can be losslessly decomposed into a positive and negative parts, 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. Decomposition is designed to take care of positive and negative information independently. These later lead to positive and negative databases, being queried with positive and negative queries. The positive part of query returns information which have a constructive proof to be true. The negative part of query provides constructive proof of falsity of a fact. This is in contrast with complementing the positive database to obtain negative part of database. Querying and storing the negative information however requires the use of *Universal Nulls*, in addition to the usual well-known *existential nulls.* The concept of universal null directs us to next section, where we introduce it.

1.5 Universal Nulls

Recall that an existential null in a tuple in a relation represents an existentially quantified variable in an atomic sentence. This corresponds to the intuition "value exists, but is unknown." A universal null, on the other hand, does not represent

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

anything unknown, but stands for all values of the domain. In other words, a universal null represents a universally quantified variable.

Universal nulls were first studied in the early days of database theory by Biskup in [13]. This was a follow-up on his earlier paper on existential nulls [12]. 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 [49] by Imielinski and Lipski, as far as existential nulls were concerned. Universal nulls next came up in [50], where Imielinski and Lipski showed that Codd's Relational Algebra could be embedded in CA, the *Cylindric Set Algebra* of Henkin, Monk, and Tarski [43, 44]. As a side remark, Imielinski and Lipski suggested that the semantics of their "$*$" symbol could be seen as modeling the universal null of Biskup. In this research we follow their suggestion[2], and fully develop a finitary representation mechanism for databases with universal nulls, as well as an accompanying finitary algebra. We show that any FO (First Order / Domain Relational Calculus) query can be translated into an equivalent expression in a finitary version of CA, and that such algebraic expressions can be evaluated "naively" by the rules "$* = *$" and "$* = a$" for any constant "a." Our finitary version is called *Star Cylindric Algebra* (SCA) and operates on finite relations containing constants and universal nulls "$*$." These relations are called *Star-Cylinders* and they are finite representations of a subclass of the infinite cylinders of Henkin, Monk, and Tarski.

Interestingly, the class of star-cylinders is closed under first order querying, meaning that the infinite result of an FO-query on an infinite instance represented by a finite sequence of finite star-cylinders can be represented by a finite star-cylinder.[3] This is achieved by showing that the class of star-cylinders are closed under our star cylindric algebra, and that SCA as a query language is equivalent in expressive power

[2]We note that Sundarmurthy et. al. [62] very recently have proposed a construct related to our universal nulls, and studied ways on placing constraints on them.

[3]Consequently there is no need to require calculus queries to be "domain independent."

with FO.

1.6 Application of Universal Nulls

In this section, we discuss the application of the universal nulls. We start the introduction of it with the following example.

Example 2. *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*
David	*Bob*

H	
Alice	*Movies*
Alice	*Music*
Bob	*Basketball*

This is to be interpreted as expressing the facts that Alice follows Chris and Chris and David follow 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 $\{(Movies), (Music)\}$. 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 [39] and the three-valued approach of [33, 57] the proposition $A \vee \neg A$ is not a tautology. In order for $A \vee \neg A$ to be true, we need either a constructive proof of A or a constructive proof of $\neg A$. Therefore both [39] and [57] 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 [39] and [57] 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^+, I^-), and $\varphi^-(\bar{x})$ returns the tuples $\bar{a}$ for which $\varphi(\bar{a})$ is true in (I^-, I^+) (i.e. $\varphi(\bar{a})$ is false in I). It turns out that databases containing “*”-nulls are suitable for storing I^-.

Example 3. *Suppose that the instance in Example 2 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 $\varphi^+ = \varphi$, 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 David has a hobby or not. ◂

1.6.1 Algebra for Universal Nulls

Our algebra is based on cylindric set algebra of Henkin, Monk, and Tarski [43, 44] which —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 $\nu(i) = \nu(x_i)$. The set 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 has an infinite number of tuples, since the domain of the variables (such as the users of a social media site) should be assumed unbounded. The idea of using countably infinite domain, called **universe** in logic, is inspired from the domain of Second Order dependencies discussed in [26] by Fagin et al. Second Order dependencies involve existentially quantified function symbols, and in order to prevent unnecessary combinatorial traps a sufficiently large domain is needed. The same assumption is also needed for existential nulls that we introduce later.

One of the basic connections 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[43, 44]. 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, where

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

and $\nu(i/a)$ denotes the valuation (tuple) ν', where $\nu'(i) = a$ 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*

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

In addition, in order to represent equality, the cylindric set algebra also contains constant relations d_{ij} 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 [43, 44] are of course infinitary[4]. In this paper we therefore develop a finitary representation mechanism, namely relations containing universal nulls "$\star$" and certain equality literals. We denote these finitary constructs $\dot{C}$ and $\dot{d}_{ij}$, respectively. 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 2 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 4 below.

Example 4. *Continuing Example 2, in that database the atoms $F(x_1, x_2)$ and $H(x_3, x_4)$ of query* (1) *are represented by star-tables $\dot{C}_F$ and $\dot{C}_H$, and the equality atom is represented by the star-diagonal $\dot{d}_{23}$. Note that these are positional relations, the "attributes" x_1, x_2, x_3, x_4 are added for illustrative purposes only.*

$\dot{C}_F$

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

$\dot{C}_H$

x_1	x_2	x_3	x_4
$\star$	$\star$	*Alice*	*Movies*
$\star$	$\star$	*Alice*	*Music*
$\star$	$\star$	*Bob*	*Basketball*

[4]General Cylinders can also have infinitely many dimensions

$\dot{d}_{23}$				
x_1	x_2	x_3	x_4	
$\star$	$\star$	$\star$	$\star$	*2=3*

The algebraic translation of query (1) *is the expression*

$$\dot{\mathsf{c}}_2(\dot{\mathsf{c}}_3(\dot{\mathsf{c}}_1((\dot{C}_F \Cap \dot{C}_H) \Cap \dot{d}_{23}))). \tag{2}$$

The intersection of $\dot{C}_F$ and $\dot{C}_H$ is carried out as star-intersection $\Cap$, where for instance $\{(a, \star, \star)\} \Cap \{(\star, b, \star)\} = \{(a, b, \star)\}$. The final result will contains the following two tuples,

$\dot{\mathsf{c}}_2(\dot{\mathsf{c}}_3(\dot{\mathsf{c}}_1((\dot{C}_F \Cap \dot{C}_H) \Cap \dot{d}_{23})))$			
x_1	x_2	x_3	x_4
$\star$	$\star$	$\star$	*Movies*
$\star$	$\star$	$\star$	*Music*

The system can now return the answer, i.e. the values of column 4 in cylinder $\dot{\mathsf{c}}_2(\dot{\mathsf{c}}_3(\dot{\mathsf{c}}_1((\dot{C}_F \Cap \dot{C}_H) \Cap \dot{d}_{23})))$. Note that columns where all rows are "$\star$" do not actually have to be materialized at any stage. Negation requires some additional details that will be introduced in Section 3.3.3. ◂

1.7 Thesis Structure

The rest of this research thesis is organized as follows. The next chapter introduces the notions used throughout the thesis. Chapter 3 describes FO-queries and cylindric set algebra and in this chapter we show that the language of FO-queries and CA-expressions have the same expressive power. This was of course only the starting point of [43, 44], and we reprove it here for benefit of the reader. Next, we introduce the *Cylindric Star-tables* and the *Cylindric Star algebra*. Finally, we conclude this section showing the equivalence of our star cylindric algebras and cylindric set algebras of Henkin, Monk and Tarski [43, 44]. It is followed by Chapter 4, which demonstrates the concept of four valued intuitionistic databases and queries. In that chapter we explains how databases are stored and queried with a decomposition and composition procedure. Later, we show how this concept can be extended to the Data Exchange Problem. In Chapter 5 the computational complexity is investigated in details. Finally in the last chapter, future work and conclusion are discussed.

1.8 Contribution

This dissertation covers the results published by the author in [39, 36, 37, 38]. The aim is to develop a clean and sound modeling of four-valued databases with universal nulls, and furthermore show that the model can be seamlessly extended to incorporate the existential nulls of Imielinski and Lipski [49]. We show that FO and our SCA are equivalent in expressive power when it comes to querying databases containing universal nulls, and that SCA queries can be evaluated (semi) naively. The equivalence is established in three steps: In Section 3.2 we show the equivalence between FO and cylindric set algebra over infinitary databases. This was of course only the starting point of [43, 44], and we recast the result here in terms of database theory.[5] In Section 3.3 we introduce our finitary star cylindric algebra. Section 3.3.1 develops the machinery for the positive case, where there is no negation in the query or database. This is then extended to include negation in Section 3.3.3. By these two sections we show that certain infinitary cylinders can be finitely represented as star-cylinders, and that our finitary star cylindric algebra on finite star-cylinders mirrors the cylindric set algebra on the infinite cylinders they represent. In Section 3.4 we tie these two results together, delivering the promised SCA evaluation of FO queries on databases containing universal nulls. In Section 3.4.2 we seamlessly extend our

[5]Van Den Bussche [18] has recently referred to [43, 44] in similar terms.

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.

Later, in Section 4.2 we show how four-valued databases can be decomposed into positive and negative two-valued instances. Similarly, we show that four-valued queries have corresponding two-valued decomposed queries that can be used to evaluate positive and negative query answers. In Section 4.3 we add universal nulls into four-valued database using Star-Tables. Furthermore, we show how to evaluate queries in the presence of universal nulls in Section 4.4. In Section 4.5 and Section 4.6 we focus on implication in data exchange in order to show that how to evaluate tuple generating dependencies in four-valued logic.

Chapter 5 then shows that all SCA expressions can be evaluated in time polynomial in the size of the database when only universal nulls are present. 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. When negation is present it has long been known that the problem is coNP-complete for databases containing existential nulls. We show that the problem remains coNP-complete when universal nulls are allowed in addition to the existential ones. For databases containing existential nulls it has been known that database containment and view containment are coNP-complete and Π_2^p-complete, respectively. We also show that the addition of universal nulls does not increase these complexities.

2 Preliminaries

There is a rich set of notations which have been used through this research. In this chapter, we explain the mathematical notations which is going to be utilized in this dissertation, in order to make this research more tangible and comprehensive for readers. We will introduce the basic framework of Relational Databases, Query Languages, the concepts of Data Exchange, Implication, and Incomplete Information along with the Naive Evaluation of queries with Existential Nulls.

2.1 Relational Model

Throughout this thesis 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 relation. The equality relation is a binary constant relation with equal values in the first and second positions.

2.1.1 Language

Our calculus is the standard domain relational calculus. Let $\{x_1, x_2, \ldots\}$ be a countably in finite set of variables. We define the set of FO-formulas φ (over $\mathscr{R}$) in the usual way: $Rp(x_{i_1}, \ldots, x_{i_{ar(R_p)}})$ 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. Let φ be an FO-formula. We denote by $vars(\varphi)$ the set of variables in φ, by $free(\varphi)$ the set of free variables in φ, and by $sub(\varphi)$ the set of subformulas of φ (for formal definitions, see [1]). If φ has n variables we say that φ is an FO_n-formula. If an FO_n-formula φ does not have any free variables, it is called a sentence. In order to keep the notation manageable, we don't allow constants in the calculus, but that the extension to constants is straightforward. Also, we assume without loss of generality that each variable occurs only once in the formula, except in equality literals. Also, we assume that a formula with n variables uses variables $x_1, \ldots, x_n$.

2.1.2 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 $\approx^I = \{(a,a) : a \in \mathbb{D}\}$. [1] It will be convenient for our purposes to adopt the algebraic approach [51] to the semantics of FO. When then identify a relation R_p^I with its characteristic function, i.e. as a mapping we have $R_p(\bar{a})^I \in \{\mathsf{true}, \mathsf{false}\}$ for each $\bar{a} \in \mathbb{D}^{ar(R_p)}$. The mapping I is then extended to a homomorphism sending FO-sentences into the usual two-valued Boolean Algebra $(\{\mathsf{true}, \mathsf{false}\}, \wedge, \vee, \neg, \mathsf{true}, \mathsf{false})$, by defining, recursively,

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

2.1.3 Queries and answers

Queries are expressed as FO-formulas, and answers in terms of the tuples of constants substituting the free variables in the formula. We arrive at the following definition.

[1]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 Chapter 3.4).

Definition 1. *Let φ be an FO_n-formula with $free(\varphi) = \{x_{i_1}, \ldots, x_{i_k}\}$, $k \leq n$. A valuation ν is a mapping from $free(\varphi)$ to $\mathbb{D}$. Let I be an instance. The answer to φ on I is defined as:*

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

This is the standard definition of answers where we are only interested in the "true answer" of the query. Later, in Section 4.1 we shall extend this notion to also include false, inconsistent, and unknown answers.

Example 5. *Let the database schema be $\mathscr{R}(E, \approx)$, where E is a binary relation representing edges in graph $G = (V, E)$. Then the formula*

$$\varphi(x_1, x_4) = \exists x_2, x_3 \ \ (E(x_1, x_2) \wedge E(x_3, x_4) \wedge (x_2 \approx x_3))$$

is an FO_4-formula over Schema $\mathscr{R}(E, \approx)$. Answers to φ are nodes of distance two in graph G. Let I be an instance over schema $\mathscr{R}(E, \approx)$ such that

$$E^I = \{(1,2), (2,3), (3,4)\}$$

and

$$\approx^I = \{(1,1), (2,2), \ldots\}$$

Let ν be a valuation such that $\nu(x_1) = 1, \nu(x_2) = 2, \nu(x_3) = 2$ and $\nu(x_4) = 3$. Then $E^I(\nu(x_1), \nu(x_2)) = E^I(1,2) = \mathsf{true}$. Similarly, $E^I(\nu_1(x_3), \nu(x_4)) = E^I(2,3) = \mathsf{true}$ and $(x_2 \approx x_3)^I = \approx^I (2,2) = \mathsf{true}$ as $\nu(x_2) = \nu(x_3)$. Finally, we have

$$(\nu(x_1), \nu(x_4)) = (1,3) \in \varphi^I.$$

Conjunctive Queries and Unions of Conjunctive Queries

The class of *Conjunctive Queries* (CQ) is a is the class of queries where the expression φ is of the form

$$\varphi(\bar{x}) = \exists \bar{y}\, R_1(\bar{x}_1) \wedge R_2(\bar{x}_2) \wedge \ldots \wedge R_m(\bar{x}_m),$$

where $\bar{x}$ is a sequence of free variables in φ, and each $\bar{x}_i$'s are sequence of variables formed from variables in $\bar{x}\bar{y}$. The class of Unions of Conjunctive Queries (UCQ) is a class of queries, defined as

$$\varphi(\bar{x}) = \{\bar{x} \;:\; \varphi_1(\bar{x}) \vee \varphi_2(\bar{x}) \vee \ldots \vee \varphi_k(\bar{x})\}$$

and each φ_i is a CQ query, as it is clear from its definition all φ_i's are of the same arity. A literal is an atom (equality atom) or its negation, that is $R_p(\bar{x}_i)$, $\neg R_p(\bar{x}_i)$, $(x_i \approx x_j)$, or $\neg(x_i \approx x_j)$ (one can see that negation of equality atom represents inequality atom). A CQ query with negation, is a CQ query built on literals in contrast to CQ queries where just atoms (and equality atoms) are allowed. The class of CQ queries with negation is denoted by CQ$^\neg$. Similarly, UCQ with negation is the class of Unions of CQ$^\neg$ queries and denoted UCQ$^\neg$.

First Order Queries

By definition, an FO-query is a first order formula built from atomic FO-formulae. When writing $\varphi(\bar{x})$ it means that $\bar{x}$ is exactly free variables in φ. If $\bar{x} = \epsilon$, the expression denotes a *Boolean* FO-query. We shall often for an atom R_p of arity k in φ, for notational convenience write $R_p(\bar{x})$ instead of $R_p(x_{i_1}, \ldots, x_{i_k})$.

Example 6. *Consider binary relation F(ollows) from Example 2, where $F(x_1, x_2)$ means that user x_1 follows user x_2 on a social media site. Let the database be the following.*

F	
Alice	*Chris*
*	*Alice*
Bob	*
Chris	*Bob*
David	*Bob*

Now, assume social media administrator decides to recommend users A and B to follow each other. Conditions below are to be considered:

1. *if users A and B are not following each other,*
2. *if both users A and B follow a person C,*
3. *if C is not followed by everyone (in order to prevent every possible pair of users to be suggested,) and*
4. *exclude Bob as admin of network from suggestion rules.*

Expression φ is an FO*-query , looking for pair of users in the domain not currently in F(ollow) (condition 1), who both are following a user (condition 2), whom is not followed by everyone (condition 3). It will also exclude Bob from suggestion list, who is following everyone as administrator (condition 4).*

$$\varphi(x_1, x_2) = \neg F(x_1, x_2) \wedge \neg(Bob \approx x_2) \wedge \exists x_3, x_4 \Big(F(x_1, x_3) \wedge F(x_2, x_4) \wedge (x_3 \approx x_4) \wedge \neg\big(\forall x_5 F(x_5, x_3)\big) \wedge \neg\big(\forall x_6 F(x_6, x_4)\big)\Big)$$

then, $\varphi^I = \{(Chris, David)\}$.

2.1.4 Dependencies and Data Exchange

In various database application, dependencies are the sets of constraints to be satisfied over the database schema. These constraints can be in form of primary-key, foreign-key or functional dependencies, and can be represented by FO-logic. Dependencies also can be utilized to transfer data stored under one database schema, called the source schema to another database schema, called the target schema. The process of mapping data from source to target schema is called data exchange and a set of dependencies is considered as the schema mapping. A dependency is a logical implication rule and is formulated as the sentence

$$\forall \bar{x}(\varphi \to \psi),$$

where $\bar{x}$ consists of all variables in φ and the variables in ψ consists of some (or none) of the variables in $\bar{x}$ plus possibly some existentially quantified variables.

Example 7. *Consider binary relation F(ollows) from Example 6. Let A(dmin) be a unary relation, where $A(x_1)$ describes that x_1 is an administrator in the social medium. Then, the rule*

$$\forall x_1\big(\forall x_2 F(x_1, x_2) \to A(x_1)\big)$$

is a tuple generating dependency (tgd), which can be used to populate the A(dmin) relation. Now, let assume that the social medium is required to have only one administrator in the network. Thus, the rule

$$\forall x_1 x_2\big(A(x_1) \land A(x_2) \to (x_1 \approx x_2)\big)$$

is an equality generating dependency (egd) which ensures that there is only one user taking the role of administration in the social medium.

In practice, $\phi(\bar{x})$ and $\psi(\bar{x})$ are conjunction of atoms while bound variables in ϕ are universally quantified and bound variables in ψ are existentially quantified. In this thesis we will show that the more expressive classes of rules, with negation and universal quantifier, can be tamed using intuitionistic logic.

2.2 Incomplete Information.

Incomplete information arises in relational databases, when a fact (tuple) has to be inserted in a relation, and values for some required attributes (columns) are missing. These missing values are known as *Existential Nulls.* Equivalently, 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." For instance, recall from Section 1.3 in an employee database, that the source of one employee might be missing, as might also the employee ID of another employee. There are numerous reasons for such missing information, e.g. the insertion was done through a view, or the incomplete tuple originated from another database that does not record these fields.

It is easy to store missing or incomplete information, by simply using a set of symbol, say $\bot_1, \bot_2, \ldots$, different from ordinary values. As an illustration,now consider the following employee database:

EmployeeID	**EmployeeName**	**Source**
$\bot_1$	Bob	S_1
123	Josh	S_2
111	Alex	$\bot_2$

In the table above, Bob's employee ID is unknown as is Alex's source. The second and third tuples have complete information in the first column. So far the picture is quite uncomplicated. The difficulties arise when queries are applied to an incomplete database.

Example 8. *Suppose the relation is decomposed into $R(EmployeeID, EmployeeName)$ and $S(EmployeeID, Source)$. Now, joining back R and S, at least the original relation should be recovered. However, it is not clear how to join the R-tuple ($\bot_1$, Bob) with the S-tuple ($\bot_2$, S_1). Nevertheless, whatever Bob's employee ID is, the null-value in the R-tuple must clearly be the same as the null-value in the S-tuple, as they both represent Bob's unknown employee ID. Therefore, the tuple ($\bot_1$, Bob, S_1) should certainly be in the result of the join. This example show why we need labeled (naive) nulls.*

Naive tables and Naive Evaluation

Let $\mathbb{N} = \{\bot_1, \bot_2, \ldots\}$ be a countable infinite set of *existential nulls*. An instance I where the relations are over $\mathbb{D} \cup \mathbb{N}$, is in the literature variably called a naive table [49, 33] or a generalized instance [27]. In either case, such an instance is taken to represent an *incomplete instance*, i.e. a (possibly) infinite set of instances. In this research we follow the model-theoretic approach of [27]. The elements in $\mathbb{D}$ represent known objects, whereas elements in $\mathbb{N}$ represent generic objects. Each generic object could turn out to be equal to one of the known objects, to another generic object, or represent an object different from all other objects. We extend our notation to include $univ(I)$, the *universe* of instance I. So far we have assumed that $univ(I) = \mathbb{D}$, but in this section we allow instances whose universe is any set between $\mathbb{D}$ and $\mathbb{D} \cup \mathbb{N}$. We are lead to the following definitions.

Definition 2. *Let h be a mapping on $\mathbb{D} \cup \mathbb{N}$ that is identity on $\mathbb{D}$, and let I and J be instances (over $\mathscr{R}$), such that $h(univ(I)) = univ(J)$. We say that h is a* possible world homomorphism *from I to J, if $h(R_p^I) \subseteq R_p^J$ for all p, and $h(\approx^I) = \approx^J$. This is denoted $I \rightarrow_h J$.*

Definition 3. *Let I be an instance with $\mathbb{D} \subseteq univ(I) \subseteq \mathbb{D} \cup \mathbb{N}$. Then the set of* *instances represented by I is*

$$Rep(I) = \{J : \exists h \text{ s.t. } I \rightarrow_h J\}.$$

We can now formulate the (standard) notion of a *certain answer* to a query.[2] By FO^+ below we mean the set of all FO-formulas not using negation.

Definition 4. *Let I be an incomplete instance and φ an FO^+-formula. The certain answer to φ on I is*

$$Cert(\varphi, I) = \bigcap_{J \in Rep(I)} \varphi^J.$$

The naive evaluation of a query in a relational database, evaluates the query on a naive table, with marked null values. In naive evaluation, query considers each marked null value, different from any other constant in the instance, and also different from any other null value with a different mark. So, a null value is only equal to itself. Naive evaluation was used to drop the tuples which have null values from the answer, but we will keep those tuples as well. Since, removing tuples with null from the final answer will lose information in case of using nested queries (composition of queries), we are interested in keeping the tuples with null values.

[2]Here we of course assume that valuations have range $univ(J)$, and that other details are adjusted accordingly.

3 Cylindric Set Algebra and Star Cylindric Algebra

We start this chapter by the formal definition of cylindric set algebra in Section 3.1. We follow this by showing the equivalence of cylindric set algebra and first order logic in Section 3.2. Since cylinders can be infinite, we want a finite mechanism to represent (at least some) infinite cylinders, and we want the mechanism to be closed under queries. This mechanism is called the *Star Cylindric Algebra* and we finish this section by showing that star cylindric algebra and cylindric set algebra are equivalent. In Section 3.3 and Section 3.3.3, our representation mechanism comes in two variations, depending on whether negation is allowed or not. We first consider the positive (no negation) case. Section 3.4 is dedicated to describing how to store *Star-Cylinders* in database and its query evaluation.

3.1 Cylindric Set Algebra

As noted in [50] the relational algebra is really a disguised version of the *Cylindric Set Algebra* (CA) of Henkin, Monk, and Tarski [43, 44]. 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 5. *Let C and C' be infinite n-dimensional cylinders. The* Cylindric Set Algebra *consists of the following operators.*

1. Union: $C \bigcup C'$. *Set theoretic union.*
2. Complement: $\overline{C} = \mathbb{D}^n \setminus C$.
3. 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 is called outer cylindrification on the i:th dimension, and will correspond to existential quantification of variable x_i.

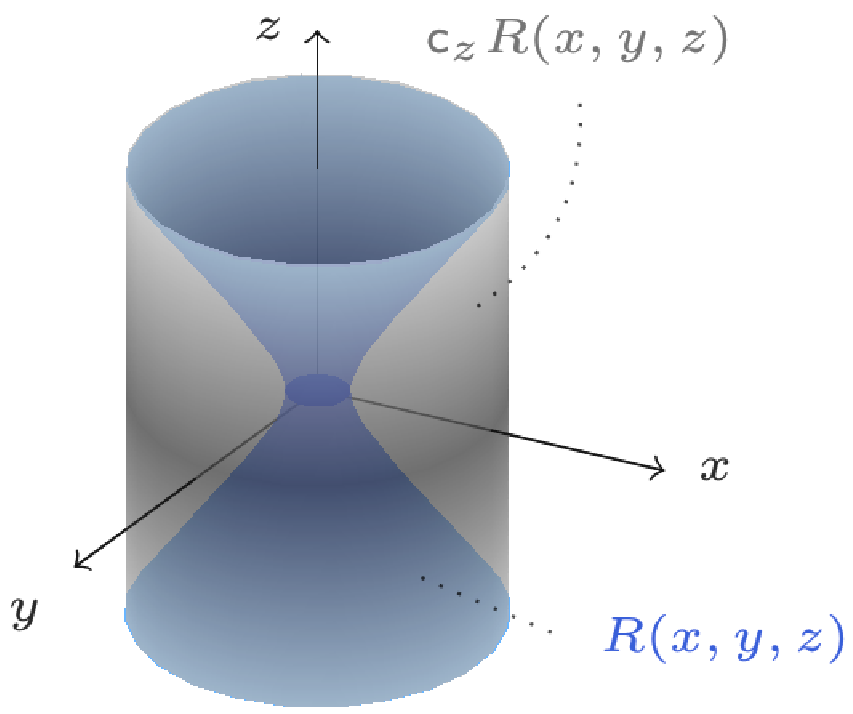


Figure 3: Outer Cylindrification Operator

Figure 3 illustrates the geometric intuition behind the name *Outer Cylindrification*, avid readers can see [43] for more information. Intersection is considered a derived operator, and we also introduce the following derived operators:

4. *Inner cylindrification:* $\mathsf{ɔ}_i(C) = \overline{\mathsf{c}_i(\overline{C})}$, corresponding to universal quantification. Note that

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

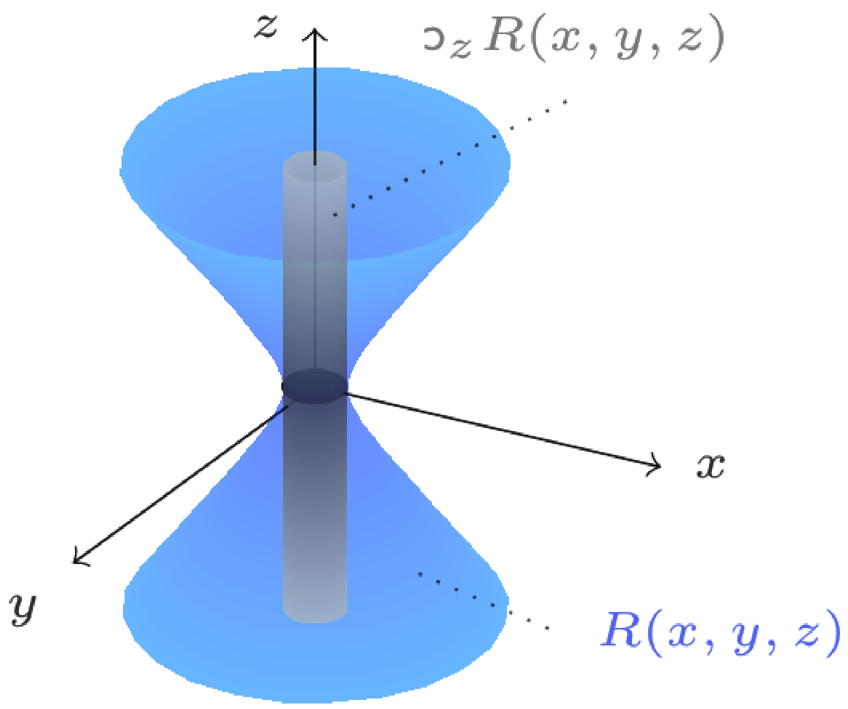


Figure 4: Inner Cylindrification Operator

Figure 4 illustrates the geometric intuition behind the name *Inner Cylindrification.*

5. *Substitution:* $\mathsf{s}_i^j(C) = \mathsf{c}_i(\mathsf{d}_{ij} \cap C)$ if $i \neq j$, and $\mathsf{s}_i^i(C) = C$.

6. *Swapping:* If $i, j \neq k$, and C is a k-full cylinder[1], then $\mathsf{z}_j^i(C) = \mathsf{s}_i^k(\mathsf{s}_j^i(\mathsf{s}_k^j(C)))$. In other words, $\mathsf{z}_j^i(C)$ interchanges the values of dimensions i and j. We also define $\mathsf{z}_{j_1,j_2}^{i_1,i_2}(C) = \mathsf{z}_{j_1}^{i_1}(\mathsf{z}_{j_2}^{i_2}(C))$, and recursively $\mathsf{z}_{j_1,\dots,j_k}^{i_1,\dots,i_k}(C) = \mathsf{z}_{j_1}^{i_1}(\mathsf{z}_{j_2,\dots,j_k}^{i_2,\dots,i_k}(C))$.

We also need the notion of cylindric set algebra expressions. Here in Definition 6 the formal defintion of CA-expressions is given.

[1]Cylinder C is k-full if $\mathsf{c}_k(C) = C$. A cylinder with this property is called *dimension complemented* in [43, 44]. In a k-full cylinder C the dimension k can be used to temporarily store the content of another dimension. This allows the definition of the swapping operators in terms of the substitution operators, which in turn are defined through intersection, diagonal, and outer cylindrification. Following [43, 44] we therefore do not need to define swapping or substitution as primitives, which would require corresponding renaming operators in the language for FO.

Definition 6. *Let $\mathbf{C} = (C_1, \ldots, C_m, d_{ij})_{i,j \in \{1,\ldots,n\}}$ be a sequence of infinite n-dimensional cylinders and diagonals. The set of CA_n-expressions (over $\mathbf{C}$) is obtained by closing the atomic expressions C_p and d_{ij} under union, intersection, complement, and inner and outer cylindrifications. Then $E(\mathbf{C})$, the value of expression E on sequence $\mathbf{C}$ is defined in the usual way, e.g. $\mathsf{C}_p(\mathbf{C}) = C_p$, $\mathsf{d}_{ij}(\mathbf{C}) = d_{ij}$, $\mathsf{c}_i(E)(\mathbf{C}) = \mathsf{c}_i(E(\mathbf{C}))$ etc.*

Example 9. *Let $\mathbb{D} = \{a_1, a_2, \ldots\}$ be a countably infinite* domain. *Let $\mathbf{C} = (C_G, d_{ij})_{i,j \in \{1,\ldots,4\}}$ be a sequence of infinite 4-dimensional cylinder and diagonals. Let $\mathsf{C}_G(\mathbf{C}) = C_G \times \mathbb{D} \times \mathbb{D}$ where, G is the graph in example 5. Let E be a CA_4-expression of form*

$$E = \mathsf{z}_2^4\Big(\mathsf{c}_{2,3}\big(\mathsf{C}_G(\mathbf{C}) \cap \mathsf{z}_{3,4}^{1,2}(\mathsf{C}_G(\mathbf{C})) \cap \mathsf{d}_{23}\big)\Big).$$

Clearly, E expresses the same query as φ in example 5.

3.2 Cylindric Set Algebra and FO

Here in this section, we start with a fundamental requirement of this research which is the pivot building block to show that the cylindric set algebra can be used to evaluate the first order queries, when it comes to infinite domain. To reach this point, we show that CA and FO have the same expressive power.

3.2.1 Equivalence of CA and FO

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 [43, 44]. An expression E in cylindric set algebra of dimension n will be called a CA_n-expression. When translating an FO_n-formula, to be used as a query on an instance I over $\mathscr{R}$, to an 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. Formally, this is expressed as follows:

Definition 7. *The horizontal n-expansion of an infinite k-ary relation R is*

$$\mathsf{h}^n(R) = R \times \mathbb{D}^{n-k}.$$

The equality relation $\approx^I = \{(a,a) : a \in \mathbb{D}\}$ is expanded into diagonals d_{ij} for $i, j \in \{1 \ldots, n\}$, where

$$d_{ij} = \bigcup_{(a,a) \in \approx^I} \mathbb{D}^{i-1} \times \{a\} \times \mathbb{D}^{j-i-1} \times \{a\} \times \mathbb{D}^{n-j},$$

and for an instance $I = (R_1^I, \ldots, R_m^I, \approx^I)$, we have

$$\mathsf{h}^n(I) = (\mathsf{h}^n(R_1^I), \ldots, \mathsf{h}^n(R_m^I), d_{ij})_{i,j}.$$

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 the swapping operator z_j^i. Every atom R_p in φ will correspond to a CA_n-expression $C_p = \mathsf{h}^n(R_p^I)$. 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 $\mathsf{z}_{i_1,\ldots,i_k}^{1,\ldots,k}(\mathsf{C}_p)$.

The entire FO_n-formula φ with $free(\varphi) = \{x_{i_1}, \ldots, x_{i_k}\}$ will then correspond to the CA_n-expression $E_\varphi = \mathsf{z}_{1,\ldots,k}^{i_1,\ldots,i_k}(F_\varphi)$, 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 $F_\varphi = \mathsf{z}_{i_1,\ldots,i_k}^{1,\ldots,k}(\mathsf{C}_p)$.
- 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 $F_\varphi = \mathsf{ɔ}_i(F_\psi)$.

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

$$x_4 \,.\, \exists x_2 \exists x_3 \forall x_1 \Big(R_1(x_1, x_2) \wedge R_2(x_3, x_4) \wedge (x_2 \approx x_3) \Big). \tag{3}$$

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 databases, 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 $d_{23} = \{t \in \mathbb{D}^n : t(2) = t(3)\}$ 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 $E_\varphi(\mathsf{h}^4(I)) =$

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

We now have $E_\varphi(\mathsf{h}^4(I)) = \mathsf{h}^4(\varphi^I)$.

Before we get to the proofs of the main two theorems in this section, we need some technical prerequisites which we borrow directly from [43]. Next proposition demonstrates the properties of the swapping operator in CA.

Proposition 1. *[43]. Let C be an n-dimensional cylinder, and $i, j \in \{1, \ldots, n\}$. Then*

1. $\mathsf{z}_j^i(C) = \mathsf{z}_i^j(C)$.
2. $\mathsf{z}_j^i(\mathsf{z}_i^j(C)) = C$.
3. $\mathsf{c}_i(\mathsf{z}_j^i(C)) = \mathsf{z}_j^i(\mathsf{c}_j(C))$.
4. If $i \neq j$ then $\mathsf{z}_j^i(C \setminus C') = \mathsf{z}_j^i(C) \setminus \mathsf{z}_j^i(C')$.
5. If $\mathsf{c}_i(C) = C$ and $\mathsf{c}_j(C) = C$ then $\mathsf{z}_j^i(C) = C$.

We also need the following proposition before we start the main results.

Proposition 2. *Let i, j, k be pairwise distinct positive integers, such that $\{i, j, k\} \cap \{1, 2, 3\} = \varnothing$, and let C be an n-dimensional cylinder that is 2-full*[2] *and k-full. Then*

$$\mathsf{z}^{i,k}_{1,2}(\mathsf{z}^{3,2,1}_{k,j,i}(C)) \ = \ \mathsf{z}^{1,2,3}_{1,j,2}(C).$$

Proof 1.

$$\mathsf{z}^{i,k}_{1,2}(\mathsf{z}^{3,2,1}_{k,j,i}(C)) = \mathsf{z}^{i,k,3,2,1}_{1,2,k,j,i}(C) = \mathsf{z}^{i,3,2,2,1}_{1,2,k,j,i}(C) = \mathsf{z}^{i,3,2,1}_{1,2,j,i}(C) = \mathsf{z}^{i,1,3,2}_{1,i,2,j}(C) = \mathsf{z}^{i,1,3,2}_{i,1,2,j}(C) = \mathsf{z}^{1,3,2}_{1,2,j}(C).$$

The second equality follows from Theorem 1.5.18 in [43], the third equality holds since $\mathsf{c}_2(C) = C$ and $\mathsf{c}_k(C) = C$, the fourth since $\{1, i\} \cap \{2, 3, j\} = \varnothing$. The last two equalities follow from Theorem 1.5.17 and 1.5.13 in [43], respectively.

We are now ready for the main proofs. The following fundamental result follows from [43, 44]. Here we restate these results and proof the main theorems with our notation and semantics, in order to provide a complete resource in this dissertation for readers.

Theorem 3. *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.

Proof 2. *We prove the stronger claim: For all FO_n-formulas φ, for all $\psi \in sub(\varphi)$, with $free(\psi) = \{x_{i_1}, \ldots, x_{i_k}\}$, there is an CA_n expression E_ψ, such that*

$$\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(E_\psi(\mathsf{h}^n(I))\Big) \ = \ \mathsf{h}^n(\psi^I),$$

[2]Cylinder C is i-full if $\mathsf{c}_i(C) = C$.

for all instances I. The main claim follows since $\varphi \in sub(\varphi)$, and the outermost sequence of swappings can be considered part of the final expression E_φ. In all cases below we assume wlog[3] *that $k < n$ so that the $k+1$: st column can be used in the necessary swappings.*

- $\psi = R_p(x_{i_1}, \ldots, x_{i_k})$, *where $k = ar(R_p)$. We let $E_\psi = \mathsf{z}^{k,\ldots,1}_{i_k,\ldots,i_1}(\mathsf{C}_p)$. We have*

$$
\begin{aligned}
&\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(E_\psi(\mathsf{h}^n(I))\Big) &&= \\
&\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(\mathsf{z}^{k,\ldots,1}_{i_k,\ldots,i_1}(\mathsf{C}_p(\mathsf{h}^n(I))\Big) &&= \\
& &&\textit{By Proposition 1 (2)} \\
&\mathsf{C}_p(\mathsf{h}^n(I)) &&= \\
&\mathsf{h}^n(R_p^I) &&= \\
&\mathsf{h}^n(\psi^I).
\end{aligned}
$$

- $\psi = x_i \approx x_j$. *We assume wlog that $n > 2$ so that swaps can be performed. We let $E_\psi = \mathsf{d}_{ij}$. We then have*

[3]If $k = n$ we can introduce an additional variable x_{n+1} and the conjunct $\exists x_{n+1}(x_{n+1} \approx x_{n+1})$ which would assure that the $n+1$:st dimension is full. Alternatively, we could introduce swapping as a primitive in the algebra. This however would require a corresponding renaming operator in the FO-formulas, see [43].

$$
\begin{array}{lll}
\mathsf{z}^{i,j}_{1,2}\Big(E_\psi(\mathsf{h}^n(I))\Big) & = & \\
\mathsf{z}^{i,j}_{1,2}\Big(\mathsf{d}_{ij}\Big) & = & \mathsf{h}^n(\{(a,a) \,:\, a \in \mathbb{D}\}) \quad = \\
\mathsf{z}^{i,j}_{1,2}\Big(\{t \in \mathbb{D}^n \,:\, t(i) = t(j)\}\Big) & = & \mathsf{h}^n((x_i \approx x_j)^I) \qquad = \\
\{t \in \mathbb{D}^n \,:\, t(1) = t(2)\} & = & \mathsf{h}^n(\psi^I). \\
\{(a,a) \,:\, a \in \mathbb{D}\} \times \mathbb{D}^{n-2} & = &
\end{array}
$$

- *$\psi = \neg\xi$, with $free(\xi) = \{x_{i_1}, \dots, x_{i_k}\}$. We assume wlog that $k < n$. Then $E_\psi = \overline{E_\xi}$, and the inductive hypothesis is*

$$\mathsf{z}^{i_1,\dots,i_k}_{1,\dots,k}\Big(E_\xi(\mathsf{h}^n(I))\Big) = \mathsf{h}^n(\xi^I)$$

We have

$$
\begin{array}{ll}
\mathsf{z}^{i_1,\dots,i_k}_{1,\dots,k}\Big(E_\psi(\mathsf{h}^n(I))\Big) & = \\
\mathsf{z}^{i_1,\dots,i_k}_{1,\dots,k}\Big(\overline{E_\xi(\mathsf{h}^n(I))}\Big) & = \\
\mathsf{z}^{i_1,\dots,i_k}_{1,\dots,k}\Big(\mathbb{D}^n \setminus E_\xi(\mathsf{h}^n(I))\Big) & = \\
 & \textit{By Proposition 1 (2)}
\end{array}
$$

$$\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(\mathbb{D}^n \smallsetminus (\mathsf{z}^{k,\ldots,1}_{i_k,\ldots,i_1}(\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}(E_\xi(\mathsf{h}^n(I)))))\Big) \quad =$$

$$\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(\mathbb{D}^n \smallsetminus (\mathsf{z}^{k,\ldots,1}_{i_k,\ldots,i_1}(\mathsf{h}^n(\xi^I)))\Big) \quad =$$

By Proposition 1 (5)

$$\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(\mathsf{z}^{k,\ldots,1}_{i_k,\ldots,i_1}(\mathbb{D}^n) \smallsetminus (\mathsf{z}^{k,\ldots,1}_{i_k,\ldots,i_1}(\mathsf{z}^n(\xi^I)))\Big) \quad =$$

By Proposition 1 (4)

$$\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(\mathsf{z}^{k,\ldots,1}_{i_k,\ldots,i_1}(\mathbb{D}^n \smallsetminus \mathsf{h}^n(\xi^I)))\Big) \quad =$$

By Proposition 1 (2)

$$D^n \smallsetminus \mathsf{h}^n(\xi^I) \quad =$$

$$\mathsf{h}^n((\neg\xi)^I) \quad =$$

$$\mathsf{h}^n(\psi^I).$$

- *$\psi = \xi \wedge \chi$, with $free(\psi) = \{x_{i_1}, \ldots, x_{i_k}\}$, $free(\xi) = \{x_{r_1}, \ldots, x_{r_p}\}$, $free(\chi) = \{x_{s_1}, \ldots, x_{s_q}\}$, $free(\psi) = free(\xi) \cup free(\chi)$, and[4] $free(\xi) \cap free(\chi) = \varnothing$. Now $E_\psi = E_\xi \bigcap E_\chi$. The inductive hypothesis is*

$$\mathsf{z}^{r_1,\ldots,r_p}_{1,\ldots,p}\Big(E_\xi(\mathsf{h}^n(I))\Big) = \mathsf{h}^n(\xi^I). \quad =$$

$$\mathsf{z}^{s_1,\ldots,s_q}_{1,\ldots,q}\Big(E_\chi(\mathsf{h}^n(I))\Big) = \mathsf{h}^n(\chi^I).$$

We have

[4]The last assumption is needed in steps annotated by †

$$\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(E_\psi(\mathsf{h}^n(I))\Big)\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(E_\xi \bigcap E_\chi\,(\mathsf{h}^n(I))\Big) =$$

$$\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(E_\xi(\mathsf{h}^n(I)) \bigcap\, E_\chi(\mathsf{h}^n(I))\Big) =$$

By Proposition 1 (2)

$$\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(\mathsf{z}^{p,\ldots,1}_{r_p,\ldots,r_1}\Big(\mathsf{z}^{r_1,\ldots,r_p}_{1,\ldots,p}(E_\xi(\mathsf{h}^n(I)))\Big) \bigcap\, \mathsf{z}^{q,\ldots,1}_{s_q,\ldots,s_1}\Big(\mathsf{z}^{s_1,\ldots,s_q}_{1,\ldots,q}(E_\chi(\mathsf{h}^n(I)))\Big)\Big) =$$

$$\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(\mathsf{z}^{p,\ldots,1}_{r_p,\ldots,r_1}\Big(\mathsf{h}^n(\xi^I)\Big) \bigcap\, \mathsf{z}^{q,\ldots,1}_{s_q,\ldots,s_1}\Big(\mathsf{h}^n(\chi^I)\Big)\Big) =$$

$$\begin{aligned}&\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(\\ &\qquad \mathsf{z}^{p,\ldots,1}_{r_p,\ldots,r_1}\Big(\mathsf{h}^n(\{\nu(x_{r_1}),\ldots,\nu(x_{r_p})\,:\,I\vDash_\nu \xi\})\Big) \;\bigcap\\ &\qquad \mathsf{z}^{q,\ldots,1}_{s_q,\ldots,s_1}\Big(\mathsf{h}^n(\{\nu(x_{s_1}),\ldots,\nu(x_{s_q})\,:\,I\vDash_\nu \chi\})\Big)\\ &\quad\Big)\end{aligned} = \dagger$$

By Proposition 1 (5)

$$\begin{aligned}&\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(\\ &\qquad \mathsf{z}^{p+q,\ldots,\,p+1,\,p,\ldots,1}_{s_q,\ldots,s_1,r_p,\ldots,r_1}\Big(\mathsf{h}^n(\{\nu(x_{r_p}),\ldots,\nu(x_{r_1})\,:\,I\vDash_\nu \xi\})\Big) \;\bigcap\\ &\qquad \mathsf{z}^{q+p,\ldots,q+1,q,\ldots,1}_{r_p,\ldots,r_1,s_q,\ldots,s_1}\Big(\mathsf{h}^n(\{\nu(x_{s_1}),\ldots,\nu(x_{s_q})\,:\,I\vDash_\nu \chi\})\Big)\\ &\quad\Big)\end{aligned} = \dagger$$

$$\mathsf{z}^{i_1,\ldots,i_k}_{1,\ldots,k}\Big(\mathsf{z}^{k,\ldots,1}_{i_k,\ldots,i_1}\Big(\mathsf{h}^n(\{\nu(x_{i_1}),\ldots,\nu(x_{i_k})\,:\,I\vDash_\nu \xi\wedge\chi\})\Big)\Big) =$$

By Proposition 1 (2)

$$\mathsf{h}^n(\{\nu(x_{i_1}),\ldots,\nu(x_{i_k})\,:\,I\vDash_\nu \xi\wedge\chi\} =$$

$$\mathsf{h}^n((\xi\wedge\chi)^I) =$$

$$\mathsf{h}^n(\psi^I).$$

- *$\psi = \exists x_{i_j} \xi$, with $free(\xi) = \{x_{i_1}, \dots, x_{i_j}, \dots, x_{i_k}\}$. Let*

$$\begin{array}{lcl} \{i'_1, \dots, i'_{k-1}\} & = & \{i_1, \dots, i_j, \dots, i_k\} \smallsetminus \{i_j\} \\ \{r_1, \dots, r_{n-k}\} & = & \{1, \dots, n\} \smallsetminus \{i_1, \dots, i_j, \dots, i_k\} \\ \{r'_1, \dots, r'_{n-k+1}\} & = & \{r_1, \dots, r_{n-k}\} \cup \{i_j\} \end{array}$$

We assume wlog that $k < n$. Let $E_\psi = \mathsf{c}_{i_j}(E_\xi)$. The inductive hypothesis is

$$\mathsf{z}^{i_1,\dots,i_k}_{1,\dots,k}\Big(E_\xi(\mathsf{h}^n(I))\Big) \;=\; \mathsf{h}^n(\xi^I).$$

We have

$$\mathsf{z}^{i'_1,\dots,i'_{k-1}}_{1,\dots,k-1}\Big(E_\psi(\mathsf{h}^n(I))\Big) \qquad =$$

$$\mathsf{z}^{i'_1,\dots,i'_{k-1}}_{1,\dots,k-1}\Big(\mathsf{c}_{i_j}(E_\xi(\mathsf{h}^n(I)))\Big) \qquad =$$

By Prop. 1 (3)

$$\mathsf{z}^{i'_1,\dots,i'_{k-1}}_{1,\dots,k-1}\Big(\mathsf{c}_{i_j}(\mathsf{z}^{k,\dots,1}_{i_k,\dots,i_1}(\mathsf{z}^{i_1,\dots,i_k}_{1,\dots,k}(E_\xi(\mathsf{h}^n(I)))))\Big) \qquad =$$

$$\mathsf{z}^{i'_1,\dots,i'_{k-1}}_{1,\dots,k-1}\Big(\mathsf{c}_{i_j}(\mathsf{z}^{k,\dots,1}_{i_k,\dots,i_1}(\mathsf{h}^n(\xi^I)))\Big) \qquad =$$

By Prop. 1 (3)

$$\mathsf{z}^{i'_1,\dots,i'_{k-1}}_{1,\dots,k-1}\Big(\mathsf{z}^{k,\dots,1}_{i_k,\dots,i_1}(\mathsf{c}_j(\mathsf{h}^n(\xi^I)))\Big) \qquad =$$

$$\mathsf{z}^{i_1,\dots,i_{j-1},i_{j+1},\dots,i_k}_{1,\dots,j-1,j,\dots,k-1}\Big(\mathsf{z}^{k,\dots,j,j-1,\dots,1}_{i_k,\dots,i_j,i_{j-1},\dots,i_1}(\mathsf{c}_j(\mathsf{h}^n(\xi^I)))\Big) \qquad =$$

$$\mathsf{z}^{i_1,\dots,i_{j-1}}_{1,\dots,j-1} \circ \mathsf{z}^{i_{j+1},\dots,i_k}_{j,\dots,k-1}\Big(\mathsf{z}^{k,\dots,j+1,j}_{i_k,\dots,i_{j+1},i_j} \circ (\mathsf{z}^{j-1,\dots,1}_{i_{j-1},\dots,i_1}\mathsf{c}_j(\mathsf{h}^n(\xi^I)))\Big) \qquad =$$

By Prop. 2

$$\mathsf{z}^{1,\dots,j-1,i_j,j,\dots,k-1}_{1,\dots,j-1,j,j+1,\dots,k}(\mathsf{c}_j(\mathsf{h}^n(\xi^I))) \qquad =$$

By Prop. 1 (3)

$$\mathsf{c}_{i_j}(\mathsf{z}^{1,\dots,j-1,i_j,j,\dots,k-1}_{1,\dots,j-1,j,j+1,\dots,k}(\mathsf{h}^n(\xi^I))) \qquad =$$

$$\mathsf{c}_{i_j}(\mathsf{z}^{1,\ldots,j-1,i_j,j,\ldots,k-1}_{1,\ldots,j-1,j,j+1,\ldots,k}(\mathsf{h}^n(\{(\nu(x_{i_1}),\ldots,\nu(x_{i_j}),\ldots,\nu(x_{i_k})) : I \vDash_\nu \xi\}))) =$$

$$\mathsf{c}_{i_j}(\mathsf{z}^{1,\ldots,j-1,i_j,j,\ldots,k-1}_{1,\ldots,j-1,j,j+1,\ldots,k}(\{(\nu(x_{i_1}),\ldots,\nu(x_{i_j}),\ldots,\nu(x_{i_k}),\nu(x_{r_1}),\ldots,\nu(x_{r_{n-k}})) : I \vDash_\nu \xi\})) =$$

$$\mathsf{c}_{i_j}(\{(\nu(x_{i'_1}),\ldots,\nu(x_{i'_{k-1}}),\nu(x_{r'_1}),\ldots,\nu(x_{i_j}),\ldots,\ldots,\nu(x_{r'_{n-k+1}})) : I \vDash_\nu \xi\}) =$$

$$\bigcup_{a\in D}\{(\nu(x_{i'_1}),\ldots,\nu(x_{i'_k}),\nu(x_{r'_1}),\ldots,\nu(x_{i_j}),\ldots\nu(x_{r'_{n-k+1}}),) : I \vDash_{\nu_{(i_j/a)}} \xi\} =$$

$$\{(\nu(x_{i'_1}),\ldots,\nu(x_{i'_k}),\nu(x_{r'_1}),\ldots,\nu(x_{i_j}),\ldots,\nu(x_{r'_{n-k+1}})) : I \vDash_\nu \exists x_{i_j}\xi\} =$$

$$\mathsf{h}^n(\{(\nu(x_{i'_1}),\ldots,\nu(x_{i'_{k-1}})) : I \vDash_\nu \exists x_{i_j}\xi\}) =$$

$$\mathsf{h}^n(\xi^I).$$

In order to complete the equivalence of CA_n expressions and FO_n formulas, we need to show that, for every CA_n expression there is an FO_n formula which is equivalent to it. The following theorem will take care of this case.

Theorem 4. *For every CA_n expression E there is an FO_n formula φ_E, such that*

$$\varphi_E^I = E(\mathsf{h}^n(I)),$$

for all instances I.

Proof 3. *We do a structural induction*

- $E = \mathsf{C}_p$. *Then* $\varphi_E = R_p(x_1,\dots,x_k) \wedge \bigwedge_{r \in \{k+1,\dots,n\}}(x_r \approx x_r)$, *where* $k = ar(R_p)$. *Clearly*

$$\begin{aligned}
&\varphi_E^I &=\\
&\{(\nu(x_1),\dots,\nu(x_k),\nu(x_{k+1}),\dots,\nu(x_n)) \,:\, I \vDash_\nu R_p(x_1,\dots,x_k)\} &=\\
&R_p^I \times \mathbb{D}^{n-k} &=\\
&\mathsf{C}_p(\mathsf{h}_n(I)) &=\\
&E(\mathsf{h}^n(I)).
\end{aligned}$$

- $E = \mathsf{d}_{ij}$. *Then* $\varphi_E = (x_i \approx x_j) \wedge \bigwedge_{r \in \{1,\dots,n\} \smallsetminus \{i,j\}}(x_r \approx x_r)$. *We have*

$$\begin{aligned}
&\varphi_E^I &=\\
&\{(\nu(x_1),\dots,\nu(x_i),\dots,\nu(x_j),\dots,\nu(x_n)) : I \vDash_\nu (x_i \approx x_j)\} &=\\
&\{t \in \mathbb{D}^n :\ t(i) = t(j)\} &=\\
&\mathsf{d}_{ij} &=\\
&E(\mathsf{h}^n(I)).
\end{aligned}$$

- $E = F_1 \bigcap F_2$. *Then* $\varphi_E = \varphi_{F_1} \wedge \varphi_{F_2}$, *and the inductive hypothesis is*

$$\begin{aligned}
\varphi_{F_1}^I &= F_1(\mathsf{h}^n(I))\\
\varphi_{F_2}^I &= F_2(\mathsf{h}^n(I))
\end{aligned}$$

Then,

$$\begin{array}{ll}
\varphi_E^I & = \\
(\varphi_{F_1} \wedge \psi_{F_2})^I & = \\
\{(\nu(x_1), \ldots, \nu(x_n)) \,:\, I \vDash_\nu \varphi_{F_1} \wedge \psi_{F_2}\} & = \\
\{(\nu(x_1), \ldots, \nu(x_n)) \,:\, I \vDash_\nu \varphi_{F_1}\} \,\cap & \\
\{(\nu(x_1), \ldots, \nu(x_n)) \,:\, I \vDash_\nu \xi_{F_2}\} & = \\
\varphi_{F_1}^I \cap \xi_{F_2}^I & = \\
F_1(\mathsf{h}^n(I)) \bigcap F_2(\mathsf{h}^n(I)) & = \\
F_1 \bigcap F_2\, (\mathsf{h}^n(I)) & = \\
E(\mathsf{h}^n(I)). &
\end{array}$$

- *$E = \overline{F}$, where Then $\varphi_E = \neg\varphi_F$, and the inductive hypothesis is $\varphi_F^I = F(\mathsf{h}^n(I))$. We have*

$$\begin{array}{ll}
\varphi_E^I & = \\
\neg\varphi_F^I & = \\
\overline{\varphi_F^I} & = \\
\overline{F(\mathsf{h}^n(I))} & = \\
E(\mathsf{h}^n(I)). &
\end{array}$$

- *$E = \mathsf{c}_i(F)$, Then $\varphi_E = (\exists x_i\, \varphi_F) \wedge (x_i \approx x_i)$. The inductive hypothesis is $\varphi_F^I = F(\mathsf{h}^n(I))$.*

  We have

$$\varphi_E^I \quad =$$

$$\{(\nu(x_1),\ldots,\nu(x_i),\ldots,\nu(x_n)) : I \vDash_\nu (\exists x_i\,\varphi_F) \wedge (x_i \approx x_i)\} \quad =$$

$$\{(\nu(x_1),\ldots,\nu(x_i),\ldots,\nu(x_n)) : I \vDash_\nu (\exists x_i\,\varphi_F)\} \ \cap$$
$$\{(\nu(x_1),\ldots,\nu(x_i),\ldots,\nu(x_n)) : I \vDash_\nu (x_i \approx x_i)\} \quad =$$

$$\{(\nu(x_1),\ldots,\nu(x_i),\ldots,\nu(x_n)) : I \vDash_\nu (\exists x_i\,\varphi_F)\} \ \cap \ \mathbb{D}^n \quad =$$

$$\{(\nu(x_1),\ldots,\nu(x_i),\ldots,\nu(x_n)) : I \vDash_\nu (\exists x_i\,\varphi_F)\} \quad =$$

$$\textstyle\bigcup_{a\in\mathbb{D}} \{(\nu((x_1),\ldots,\nu(x_i),\ldots,\nu(x_n)) : I \vDash_{\nu_{(i/a)}} \varphi_F\} \quad =$$

$$\mathsf{c}_i(\{(\nu((x_1),\ldots,\nu(x_i),\ldots,\nu(x_n)) : I \vDash_\nu \varphi_F\}) \quad =$$

$$\mathsf{c}_i(\varphi_F^I) \quad =$$

$$\mathsf{c}_i(F(\mathsf{h}^n(I))) \quad =$$

$$E\,(\mathsf{h}^n(I)).$$

In this section, we showed that CA and FO are equivalent. This result is useful in theory however, working with infinite cylinders is not possible when it comes to implementation in databases. This leads us toward the next section where, we demonstrate a finite structure that can represent infinite cylinders and we develop an algebra which is closed under this structure, called *Star Cylindric Algebra* denoted SCA. Finally we show that CA and SCA are equivalent which leads to equivalence of FO and SCA as a consequence.

3.3 Star Cylinders and Star Cylindric Algebra

First in this section we present *Star-Cylinders* and show that they express some infinite cylinders in a finite manner. Next we define positive algebra over star-cylinders which we call *Positive Star Cylindric Algebra* denoted SCA^+. Finally we extend our algebra to full algebra for star-cylinders which we simply call *Star Cylindric Algebra* denoted SCA.

3.3.1 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 $i, j \in \{1, \ldots, n\}$, as well as the logical constant false. Also, $\wp(.)$ denotes the powerset operation. Star-tuples will be denoted $\dot{t}, \dot{u}, \ldots$, where 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 8. *An n-dimensional star-cylinder $\dot{C}$ is said to be in* normal form *if $\dot{t}(n+1) \nvDash \mathsf{false}$, and $\dot{t}(n+1) \vDash (i = j)$ if and only if $(i = j) \in \dot{t}(n+1)$ and $\dot{t}(i) = \dot{t}(j)$, for all star-tuples $\dot{t} \in \dot{C}$ and $i, j \in \{1, \ldots, n\}$.*

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 9. *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 $\dot{u}(n+1) \subseteq \dot{t}(n+1)$.*

Definition 10. *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 $(a_1, \ldots, a_n, \theta)$, where θ contains $(i = j)$ iff $a_i = a_j$. 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 $\dot{t} \preceq \dot{u}$.

Domination in star-cylinders is translatable to inclusion of ordinary cylinders. The following lemma expresses this correspondence.

Lemma 1. *Let $\dot{C}$ and $\dot{D}$ be n-dimensional (positive) star-cylinders. Then*

$$[\![\dot{C}]\!] \subseteq [\![\dot{D}]\!] \text{ iff } \dot{C} \preceq \dot{D}.$$

Proof 4. *We first show that $[\![\{\dot{t}\}]\!] \subseteq [\![\dot{D}]\!]$ if and only if there is a star-tuple $\dot{u} \in \dot{D}$, such that $\dot{t} \preceq \dot{u}$.*[5] *For a proof, we note that if $\dot{t} \preceq \dot{u}$ for some $\dot{u} \in \dot{D}$, then $[\![\{\dot{t}\,\}]\!] \subseteq [\![\dot{D}]\!]$.*

[5]Note here the normal form requirement $\dot{t}(n{+}1) \nvDash \mathsf{false}$, since $\dot{t}(n{+}1) \vDash \mathsf{false}$ means that $[\![\{\dot{t}\}]\!] = \varnothing$, while there is no star-tuple $\dot{u}$, such that $\dot{t} \preceq \dot{u}$ and $\dot{u}(n+1) \nvDash \mathsf{false}$.

For the other direction, assume that $[\![\{\dot{t}\}]\!] \subseteq [\![\dot{D}]\!]$*. Let* $A \subseteq \mathbb{D}$ *be the finite set of constants appearing in* $\dot{t}$ *or* $\dot{D}$*. Construct the tuple* $t \in (A \cup \{*\})^n$*, where* $t(i) = \dot{t}(i)$ *if* $\dot{t}(i) \in A$*, and* $t(i) = a_i$ *if* $\dot{t}(i) = *$*. Here* a_i *is a unique value in the set* $\mathbb{D} \setminus A$*. If* $\dot{t}(n+1)$ *contains an equality* $(i = j)$ *we choose* $a_i = a_j$*. Then* $t \in [\![\{\dot{t}\}]\!] \subseteq [\![\dot{D}]\!]$*, so there must be a tuple* $\dot{u} \in \dot{D}$*, such that* $t \leq \dot{u}$*. It remains to show that* $\dot{t} \leq \dot{u}$*. If* $t(i) = a$ *for some* $a \in A$*, then* $\dot{t}(i) = a$*, and since* $t \leq \dot{u}$ *it follows that* $\dot{t}(i) \leq \dot{u}(i)$*. If* $t(i) = a_i \notin A$ *then* $\dot{t}(i) = *$*, and therefore* $t(i/b) \in [\![\{\dot{t}\}]\!] \subseteq [\![\dot{D}]\!]$*, for any* b *in the infinite set* $\mathbb{D} \setminus A$*. Consequently it must be that* $\dot{u}(i) = *$*, and thus* $\dot{t}(i) \leq \dot{u}(i)$*. This is true for all* $i \in \{1, \ldots, n\}$*. Finally, if* $(i = j) \in \dot{u}(n+1)$*, we have two cases: If* $t(i) \in A$ *then* $\dot{t}(i) = \dot{t}(j)$*, and if* $t(i) \notin A$ *then* $(i = j) \in \dot{t}(n+1)$*. In summary, we have shown that* $\dot{t} \leq \dot{u}$*.*

We now return to the proof of the claim of the lemma. The if-direction follows directly from definitions. For the only-if direction, assume that $[\![\dot{C}]\!] \subseteq [\![\dot{D}]\!]$*. To see that* $\dot{C} \leq \dot{D}$ *let* $\dot{t} \in \dot{C}$*. Then* $[\![\{\dot{t}\}]\!] \subseteq [\![\dot{C}]\!] \subseteq [\![\dot{D}]\!]$*. We have just shown above that this implies that there is a* $\dot{u} \in \dot{D}$ *such that* $\dot{t} \leq \dot{u}$*, meaning that* $\dot{C} \leq \dot{D}$*.*

3.3.2 Positive Star Cylindric Algebra

Next we redefine the positive cylindric set algebra operators so that $[\![\dot{C} \mathbin{\dot{\circ}} \dot{D}]\!] = [\![\dot{C}]\!] \circ [\![\dot{D}]\!]$ or $\circ([\![\dot{D}]\!]) = [\![\dot{\circ}(\dot{D})]\!]$, for each positive cylindric algebra operator $\circ$, its redefinition $\dot{\circ}$, and star-cylinders $\dot{C}$ and $\dot{D}$. We first define the *meet* $\dot{t} \mathbin{\dot{\wedge}} \dot{u}$ of star-tuples $\dot{t}$ and $\dot{u}$:

Definition 11. *Let* $\dot{t}$ *and* $\dot{u}$ *be* n*-ary star-tuples. The* n*-ary star-tuple* $\dot{t} \mathbin{\dot{\wedge}} \dot{u}$ *is defined as follows: If* $\dot{t}(j), \dot{u}(j) \in \mathbb{D}$ *for some* j *and* $\dot{t}(j) \neq \dot{u}(j)$*, then* $\dot{t} \mathbin{\dot{\wedge}} \dot{u}\,(i) = *$ *for* $i \in \{1, \ldots, n\}$*,* *and* $\dot{t} \wedge \dot{u}\,(n+1) = \{\mathsf{false}\}$.[6] *Otherwise, for* $i \in \{1, \ldots, n\}$

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

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

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[7] $\dot{t} \wedge \dot{u} \leq \dot{t}$, and $\dot{t} \wedge \dot{u} \leq \dot{u}$, and if for a star-tuple $\dot{v}$, we have $\dot{v} \leq \dot{t}$ and $\dot{v} \leq \dot{u}$, then $\dot{v} \leq \dot{t} \wedge \dot{u}$.

Definition 12. *The n-dimensional positive cylindric star-algebra consists of the following operators.*

1. Star-diagonal: $\dot{d}_{ij} = \{(\overbrace{*, \ldots, *}^{n}, (i = j))\}$
2. Star-union: $\dot{C} \,\dot{\cup}\, \dot{D} = \{\dot{t} \;:\; \dot{t} \in \dot{C} \textit{ or } \dot{t} \in \dot{D}\}$
3. Star-intersection: $\dot{C} \,\dot{\cap}\, \dot{D} = \{\dot{t} \wedge \dot{u} \;:\; \dot{t} \in \dot{C} \textit{ and } \dot{u} \in \dot{D}\}$
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*

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

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 $\dot{\mathsf{c}}_i(\dot{C}) = \{\dot{\mathsf{c}}_i(\dot{t}\,) \;:\; \dot{t} \in \dot{C}\}$.

[6] Here $*$ can be replaced by any arbitrary constant a in $\mathbb{D}$, but for consistency we prefer to use $*$.
[7] Assuming the normal form requirement $\dot{t} \wedge \dot{u}\,(n+1) \not\models \mathsf{false}$.

5. Inner cylindrification: *Let $\dot{C}$ be an n-dimensional cylinder and $i \in \{1, \ldots, n\}$. Then*

$$\dot{\mathsf{ɔ}}_i(\dot{C}) = \{\dot{t} \in \dot{C} \ : \ \dot{t}(i) = *, \ \textit{and} \ (i = j) \notin \dot{t}(n+1) \ \textit{for any} \ j\}.$$

We illustrate the positive cylindric star-algebra with the following small example.

Example 10. *Let $\dot{C}_1 = \{(a, *, *, *, *, \{(3 = 4)\})\}$, $\dot{C}_2 = \{(*, b, *, *, *, \{(4 = 5)\})\}$, and $\dot{C}_3 = \{(a, b, *, *, *, \{(4 = 5)\})\}$. Consider the expression $\dot{\mathsf{ɔ}}_3((\dot{\mathsf{c}}_{1,4}(\dot{C}_1 \mathbin{\dot\cap} \dot{C}_2)) \mathbin{\dot\cup} \dot{C}_3)$. Then we have the following.*

$$\begin{array}{rcl}
\dot{C}_1 \mathbin{\dot\cap} \dot{C}_2 & = & \{(a, b, *, *, *, \{(3 = 4), (4 = 5), (3 = 5)\})\} \\
\dot{\mathsf{c}}_{1,4}(\dot{C}_1 \mathbin{\dot\cap} \dot{C}_2) & = & \{(*, b, *, *, *, \{(3 = 5)\})\} \\
(\dot{\mathsf{c}}_{1,4}(\dot{C}_1 \mathbin{\dot\cap} \dot{C}_2)) \mathbin{\dot\cup} C_3 & = & \{(*, b, *, *, *, \{(3 = 5)\}), (a, b, *, *, *, \{(4 = 5)\})\} \\
\dot{\mathsf{ɔ}}_3((\mathsf{c}_{1,4}(\dot{C}_1 \mathbin{\dot\cap} \dot{C}_2)) \mathbin{\dot\cup} C_3) & = & \{(a, b, *, *, *, \{(4 = 5)\})\}
\end{array}$$

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

Theorem 5. *Let $\dot{C}$ and $\dot{D}$ be n-dimensional star-cylinders and $\dot{d}_{ij}$ an n-dimensional star-diagonal. Then the following statements hold.*

1. $[[\dot{d}_{ij}]] = d_{ij}$.

2. $[[\dot{C} \mathbin{\dot\cup} \dot{D}]] = [[\dot{C}]] \bigcup \ [[\dot{D}]]$.

3. $[[\dot{C} \mathbin{\dot\cap} \dot{D}]] = [[\dot{C}]] \bigcap \ [[\dot{D}]]$.

4. $[[\dot{\mathsf{c}}_i(\dot{C})]] = \ \mathsf{c}_i([[\dot{C}]])$,

5. $[[\dot{\mathsf{ɔ}}_i(\dot{C})]] = \ \mathsf{ɔ}_i([[\dot{C}]])$,

Proof 5. *1.* $t \in [\![\dot{d}_{ij}]\!]$ *iff* $t \preceq (*, \ldots, *, (i = j))$ *iff* $t \in \{t \in \mathbb{D}^n : t(i) = t(j)\}$ *iff* $t \in d_{ij}$.

2. $t \in [\![\dot{C} \mathbin{\dot\cup} \dot{D}]\!]$ *iff* $\exists \dot{u} \in \dot{C} : t \preceq \dot{u}$ *or* $\exists \dot{v} \in \dot{D} : t \preceq \dot{v}$ *iff* $t \in [\![\dot{C}]\!]$ *or* $t \in [\![\dot{D}]\!]$ *iff* $t \in [\![\dot{C}]\!] \bigcup [\![\dot{D}]\!]$.

3. Let $t \in [\![\dot{C} \mathbin{\dot\cap} \dot{D}]\!]$. *Then there is a star-tuple* $\dot{t} \in \dot{C} \mathbin{\dot\cap} \dot{D}$ *such that* $t \preceq \dot{t}$, *which again means that there are star-tuples* $\dot{u} \in \dot{C}$ *and* $\dot{v} \in \dot{D}$, *such that* $\dot{t} = \dot{u} \curlywedge \dot{v}$. *As a consequence* $t \preceq \dot{u}$ *and* $t \preceq \dot{v}$, *which implies* $t \in [\![\dot{C}]\!]$ *and* $t \in [\![\dot{D}]\!]$, *that is,* $t \in [\![\dot{C}]\!] \bigcap [\![\dot{D}]\!]$. *The proof for the other direction is similar.*

4. Let $t \in [\![\dot{\mathsf{c}}_i(\dot{C})]\!]$. *Then there is a star-tuple* $\dot{t} \in \dot{\mathsf{c}}_i(\dot{C})$ *such that* $t \preceq \dot{t}$. *This in turn means that there is a star-tuple* $\dot{u} \in \dot{C}$ *such that either* $\dot{u} = \dot{t}(i/a)$ *for some* $a \in \mathbb{D}$, *or* $\dot{u}(i) = *$ *and* $\dot{u} = \dot{t}$, *except possibly* $\dot{u}(n+1) \models \theta$ *where* θ *is a set of equalities involving column* i, *and* $\dot{t}(n+1)$ *does not have any conditions on* i.

Case 1. $\dot{u} = \dot{t}(i/a)$ *for some* $a \in \mathbb{D}$. *Then* $\dot{t}(i/a) \in \dot{C}$ *which means that there is a tuple* $u \in [\![\dot{C}]\!]$ *such that* $u \preceq \dot{t}(i/a)$. *Since* $[\![\dot{C}]\!] \subseteq \mathsf{c}_i([\![\dot{C}]\!])$, *it follows that* $u \in \mathsf{c}_i([\![\dot{C}]\!])$. *Suppose* $u \neq t$. *Then* $u(j) \neq t(j)$ *for some* $j \in \{1 \ldots, n\}$.

If $j = i$, *then* $t = u(j/t(j)) \in \mathsf{c}_j([\![\dot{C}]\!]) = \mathsf{c}_i([\![\dot{C}]\!])$.

If $j \neq i$ *and* $\dot{t}(j) = *$ *it means that* $\dot{u}(j) = *$, *and thus* $t = u(j/t(j)) \in [\![\dot{C}]\!]$, *which in turn entails that* $t \in \mathsf{c}_i([\![\dot{C}]\!])$. *Otherwise, if* $\dot{t}(j) \neq *$, *then* $\dot{t}(j) \in \mathbb{D}$, *which means that* $\dot{u}(j) \in \mathbb{D}$, *and* $u(j) = t(j)$ *after all.*

Case 2. $\dot{u}(i) = *$ *and (possibly)* $\dot{u}(n+1)$ *contains a set of equalities say* θ, *involving column* i, *and* $\dot{t}(n+1)$ *does not have any conditions on* i.

Suppose first that $t \models \theta$. *Then* $t \preceq \dot{u}$, *and consequently* $t \in [\![\dot{C}]\!] \subseteq \mathsf{c}_i([\![\dot{C}]\!])$.

Suppose then that $t \not\models \theta$. *If* t *violates an equality* $(i = j) \in \theta$ *it must be that* $\dot{t}(j) = \dot{u}(j) = *$, *and* $\dot{t}$ *and* $\dot{u}$ *have the same conditions on column* j. *Let* u *be a tuple such that* $u \preceq \dot{u}$. *Then* $t(i/u(i)) \in [\![\dot{C}]\!]$, *and hence* $t \in \mathsf{c}_i([\![\dot{C}]\!])$.

For the other direction, let $t \in \mathsf{c}_i([\![\dot{C}]\!])$. Then there is a tuple $u \in [\![\dot{C}]\!]$, such that $t(i/u(i)) = u$. Hence there is a star-tuple $\dot{u} \in \dot{C}$, such that $u \preceq \dot{u}$ and $t(i/u(i)) \preceq \dot{u}$. If $t \npreceq \dot{u}$ it is because $t(i)$ violates some condition in $\dot{u}(n+1)$. Since all conditions involving column i are deleted in $\dot{\mathsf{c}}_i(\dot{C})$, it follows that $\dot{\mathsf{c}}_i(\dot{C})$ must contain a star-tuple $\dot{v}$ obtained by outer cylindrification of $\dot{u}$. Then clearly $t \preceq \dot{v}$ and $t \vDash \dot{v}(n+1)$. Consequently $t \in [\![\dot{\mathsf{c}}_i(\dot{C})]\!]$.

5. *Let $t \in [\![\dot{\text{ɔ}}_i(\dot{C})]\!]$. Then there is a star-tuple $\dot{t} \in \dot{\text{ɔ}}_i(\dot{C})$, such that $t \preceq \dot{t}$. Clearly, $\dot{t} \in \dot{\text{ɔ}}_i(\dot{C})$ means that $\dot{t} \in \dot{C}$ where by definition $\dot{t}(i) = *$, and $(i = j) \notin \dot{t}(n+1)$ for any j. As a consequence $t(i/a) \preceq \dot{t}$ for all $a \in \mathbb{D}$. This implies that $t(i/a) \in [\![\dot{C}]\!]$ for all $a \in \mathbb{D}$, and thus $t \in \text{ɔ}_i([\![\dot{C}]\!])$.*

   For the other direction, let $t \in \text{ɔ}_i([\![\dot{C}]\!]) \subseteq [\![\dot{C}]\!]$. This means that $t(i/a) \in [\![\dot{C}]\!]$ for all $a \in \mathbb{D}$. That is, there exists a star-tuple $\dot{t} \in \dot{C}$, such that $t \preceq \dot{t}$. Also, $t(i/a) \preceq \dot{t}$ for all $a \in \mathbb{D}$, since there otherwise has to be an infinite number of star-tuples in in the finite star-cylinder $\dot{C}$. Thus it must be that $\dot{t}(i) = *$, and $(i = j) \notin \dot{t}(n+1)$ for any j. Consequently, $\dot{t} \in \dot{\text{ɔ}}_i(\dot{C})$ and $t \in [\![\dot{\text{ɔ}}_i(\dot{C})]\!]$.

In order to show the equivalence of positive star cylindric algebra and positive cylindric set algebra we need the concept of algebra expressions.

Definition 13. *Let $\dot{\mathbf{C}} = (\dot{C}_1, \ldots, \dot{C}_m, \dot{d}_{ij})_{i,j}$ be a sequence of n-dimensional star-cylinders and star-diagonals. We define the set of* Positive Star Cylindric Algebra Expressions SCA_n^+ *and values of expressions as in Definition 6, noting that $\dot{\mathsf{C}}_p(\dot{\mathbf{C}}) = \dot{C}_p$, and $\dot{\mathsf{d}}_{ij}(\dot{\mathbf{C}}) = \dot{d}_{ij}$.*

In the following CA_n^+ denotes the set of all n-dimensional positive cylindric algebra expressions. We now have from Theorem 5

Corollary 1. *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 $\dot{\mathbf{C}}$.

3.3.3 Adding Negation

From here on we also allow conditions of the form $(i \neq j)$, $(i \neq a)$, for $a \in \mathbb{D}$ in star-cylinders, which then will be called *extended star-cylinders.* Conditions of the form $(i = j), (i \neq j)$ or $(i \neq a)$ will be called *literals*[8], usually denoted ℓ. In other words, in an extended n-dimensional star-cylinder each (extended) star-tuple $\dot{t}$ has a (finite) set of literals in position $n + 1$.

Example 11. *In Example 17 we were interested in the negative information as well as positive information. The instance from Example 17 can be formally represented as the extended star-cylinder below.*

H^-		
Alice	*Volleyball*	$\{\mathsf{true}\}$
Bob	$*$	$\{(2 \neq \textit{Basketball})\}$
Chris	$*$	$\{\mathsf{true}\}$

We next extend Definitions 8, 9, 10, 11, and 12 to apply to extended star-cylinders. Lemma 1 will be replaced by Lemma 3.

Definition 14. *(Replaces Definition 8). An extended n-dimensional star-cylinder $\dot{C}$ is said to be in* normal form*, if $\dot{t}(n + 1) \nvDash \mathsf{false}$, and $\dot{t}(n + 1) \vDash \ell$ if and only if $\ell \in \dot{t}(n + 1)$, and*

1. *$(i = j) \in \dot{t}(n + 1)$ if and only if $t(i) = t(j)$,*
2. *$(i \neq j) \in \dot{t}(n+1)$ if and only if $t(i) \neq t(j)$, or $(i \neq j) \in \dot{t}(n+1)$ and $t(i) = t(j) = *$,*
3. *$(i \neq a) \in \dot{t}(n + 1)$ entails $t(i) \neq a$,*

for all star-tuples $\dot{t} \in \dot{C}$ and $i, j \in \{1, \dots, n\}$.

[8] false is also a literal

In the proof of Theorem 18 in Section 6 we show that maintaining extended star-cylinders in normal form can be done in polynomial time. We therefore assume in the sequel that all extended star-cylinders and -tuples are in normal form. Keeping the extended notion of normal form in mind, it is easy to see that Definition 9 of dominance $\dot{t} \leq \dot{u}$ suits extended star-tuples $\dot{t}$ and $\dot{u}$ as well. Definition 10 remains unchanged, provided we identify an "ordinary" tuple $(a_1, \ldots, a_n)$ with the extended star-tuple $(a_1, \ldots, a_n, \theta)$, where $(i = j) \in \theta$ iff $a_i = a_j$ and $(i \neq j) \in \theta$ iff $a_i \neq a_j$. Definition 11 also applies as such to extended star-tuples. For the outer cylindrification in Definition 12 we now stipulate that $\dot{\mathsf{c}}_i(\dot{t})(n+1)$ contains all and only those literals from $\dot{t}(n+1)$ that do not involve dimension i. It is an easy exercise to verify that the proofs of parts 1 – 4 of Theorem 5 remain valid in the presence of literals. Finally, inner cylindrification will be redefined below, along with the definition of the complement operator. Before that we introduce the notion of a *sieve*-cylinder.

Definition 15. *Let $\dot{\mathbf{C}}$ be a sequence of n-dimensional extended star-cylinders and $A \subseteq \mathbb{D}$ be the set of constants appearing therein. For $t \in (A \cup \{*\})^n$, define the sets $S_t = \{i : t(i) = *\}$ and $SS_t = \{(i,j) : t(i) = t(j) = *\}$. For each tuple $t \in (A \cup \{*\})^n$ and each subset SS'_t of SS_t, form the star-tuple $\dot{t}$ with $\dot{t}(i) = t(i)$ for $i \in \{1, \ldots, n\}$, and $\dot{t}(n+1) =$*

$$\bigcup_{i \in S_t} \{(i \neq a) : a \in A\} \bigcup_{(i,j) \in SS'_t} \{(i = j)\} \bigcup_{(i,j) \in SS_t \setminus SS'_t} \{(i \neq j)\}.$$

$\dot{A}$ is the extended star-cylinder of all such star-tuples $\dot{t}$, and is called the sieve *of $\dot{\mathbf{C}}$.*

The sieve $\dot{A}$ has some useful properties stated in the next two lemmas. These properties allow us to test containmnent $[\![\dot{C}]\!] \subseteq [\![\dot{D}]\!]$ and to define negation and inner cylindrification on a tuple-by-tuple basis using the partial order $\leq$.

Lemma 2. *Let $\dot{C}$ be an n-dimensional star-cylinder and $\dot{A} = \{\dot{t}_1, \ldots, \dot{t}_m\}$ its sieve. Then*

1. *$[\![\dot{A}]\!] = \mathbb{D}^n$ and $\{[\![\{\dot{t}_1\}]\!], \ldots, [\![\{\dot{t}_m\}]\!]\}$ is a partition of $[\![\dot{A}]\!]$.*
2. *If $\dot{t} \wedge \dot{u} \in \dot{C} \Cap \dot{A}$ and $\dot{t} \wedge \dot{u} \neq \dot{t}_\varnothing$, then $\dot{t} \wedge \dot{u} = \dot{u}$.*

Proof 6. *To see that* $[\![\dot{A}]\!] = \mathbb{D}^n$*, let* t *be an arbitrary tuple in* $\mathbb{D}^n$*. By construction, there are star-tuples* $\dot{t} \in \dot{A}$ *such that* $\dot{t}(i) = t(i)$ *if* $t(i) \in A$*, and* $\dot{t}(i) = *$ *if* $t(i) \in \mathbb{D} \setminus A$*. Since there is the subset* $SS_t' = \{(i,j) : t(i) = t(j),\ \text{and}\ t(i) \in \mathbb{D} \setminus A\}$ *we see that for one of these* $\dot{t}$*-tuples it holds that* $t \preceq \dot{t}$*. The fact that* $[\![\{\dot{t}_i\}]\!] \cap [\![\{\dot{t}_j\}]\!] = \varnothing$ *whenever* $i \neq j$ *follows from the fact that if there were a tuple* t *in the intersection, it would have to agree with* $\dot{t}_i$ *and* $\dot{t}_j$ *on all columns with values in* A*. But the* SS_t' *set used for* $\dot{t}_i$ *would be different than the one used for* $\dot{t}_j$*, which means that we cannot have both* $t \preceq \dot{t}_i$ *and* $t \preceq \dot{t}_j$*.*

For part 2, let $\dot{t} \wedge \dot{u} \in \dot{C} \Cap \dot{A}$ *and* $\dot{t} \wedge \dot{u} \neq \dot{t}_\varnothing$*. We claim that* $\dot{u} \preceq \dot{t}$*, which would imply* $\dot{t} \wedge \dot{u} = \dot{u}$*. Since* $\dot{t} \wedge \dot{u} \neq \dot{t}_\varnothing$ *there is a tuple* $t \in [\![\{\dot{t} \wedge \dot{u}\}]\!]$*. For each* $i \in \{1, \ldots, n\}$*, consider* $\dot{u}(i)$*. If* $\dot{u}(i) = a \in A$*, then* $t(i) = a$*, which means that* $\dot{t}(i) = a$ *or* $\dot{t}(i) = *$*. Consequently* $\dot{u}(i) \preceq \dot{t}(i)$*. If* $\dot{u}(i) = *$ *then* $t(i) \in \mathbb{D} \setminus A$*, since* $(i \neq a) \in \dot{u}(n+1)$ *for all* $a \in \mathbb{D} \setminus A$*. Since* $t(i) \preceq \dot{t}(i)$*, and* $\dot{t}(i) \in A \cup \{*\}$*, it follows that* $\dot{t}(i) = *$*. Then let* $(i = j) \in \dot{t}(n+1)$*. Since* $\dot{t} \wedge \dot{u}\,(n+1)$ *is satisfiable, and* $\dot{u}(n+1)$ *contains either* $(i = j)$ *or* $(i \neq j)$*, it follows that* $(i = j) \in \dot{u}(n+1)$*. We have now shown that* $\dot{u} \preceq \dot{t}$*.*

Lemma 3. *Let* $\dot{C}$ *and* $\dot{D}$ *be* n*-dimensional extended star-cylinders and* $\dot{A}$ *their (common) sieve. Then*

$$[\![\dot{C}]\!] \subseteq [\![\dot{D}]\!] \ \text{ iff } \ \dot{C} \Cap \dot{A} \preceq \dot{D} \Cap \dot{A}.$$

Proof 7. *For the* if*-direction, let* $t \in [\![\dot{C}]\!] = [\![\dot{C} \Cap \dot{A}]\!]$*. Then there is a star-tuple* $\dot{t} \in \dot{C} \Cap \dot{A}$*, such that* $t \preceq \dot{t}$*. Since* $\dot{C} \Cap \dot{A} \preceq \dot{D} \Cap \dot{A}$ *there is a star tuple* $\dot{u} \in \dot{D} \Cap \dot{A}$ *such that* $\dot{t} \preceq \dot{u}$*. Thus* $t \in [\![\dot{D} \Cap \dot{A}]\!] = [\![\dot{D}]\!]$*.*

For the only-if direction, let $\dot{t}_1 \wedge \dot{u}_1 \in \dot{C} \Cap \dot{A}$*, and* $t \preceq \dot{t}_1$ *and* $t \preceq \dot{u}_1$*. Then* $t \in [\![\dot{C}]\!] \subseteq [\![\dot{D}]\!] = [\![\dot{D} \Cap \dot{A}]\!]$*, so there are star-tuples* $\dot{t}_2 \in \dot{D}$ *and* $\dot{u}_2 \in \dot{A}$ *such that* $t \preceq \dot{t}_2$ *and* $t \preceq \dot{u}_2$*. From Lemma 2 it follows that* $\dot{u}_1 = \dot{u}_2$*, and thus* $\dot{t}_1 \wedge \dot{u}_1 = \dot{u}_1 = \dot{u}_2 = \dot{t}_2 \wedge \dot{u}_2$*. Consequently* $\dot{t}_1 \wedge \dot{u}_1 \preceq \dot{t}_2 \wedge \dot{u}_2$*.*

We can now define the desired operations.

Definition 16. *Let $\dot{A}$ be the sieve of $\dot{\mathbf{C}}$ and $\dot{C}$ be an extended star-cylinder in $\dot{\mathbf{C}}$. Then*

1. $\dot{\neg}\dot{C} = \{\dot{t} \in \dot{A} : \{\dot{t}\} \Cap \dot{C} = \{\dot{t}_{\varnothing}\}\}$. *and*

2. $\hat{\text{ɔ}}_i(\dot{C}) = \{\dot{t} \in \dot{C} \Cap \dot{A} : (\dot{\text{c}}_i(\{\dot{t}\}) \Cap \dot{A}) \leq (\dot{C} \Cap \dot{A})\}$.

Example 12. *Let $\dot{C} = \{(a, *, \{\})\}$. Then $\dot{A}$ is shown in the extended star-cylinder below, and $\dot{\neg}\dot{C}$ consists of the first, second, and fourth tuples of $\dot{A}$.*

$\dot{A}$		
$*$	$*$	$\{(1 \neq a), (2 \neq a), (1 = 2)\}$
$*$	$*$	$\{(1 \neq a), (2 \neq a), (1 \neq 2)\}$
a	$*$	$\{(2 \neq a)\}$
$*$	a	$\{(1 \neq a)\}$
a	a	$\{\}$

Now, let $\dot{C} = \{(a, *, \{(2 \neq a)\}), (a, a, \{\})\}$. Then $\dot{A}$ is as above, and $\hat{\text{ɔ}}_2(\dot{C}) = \dot{C}$ as the reader easily can verify.

We can now verify that the new operators work as expected.

Theorem 6. *Let $\dot{C}$ be an extended star-cylinder. Then*

1. $[\![\dot{\neg}\dot{C}]\!] = \overline{[\![\dot{C}]\!]}$

2. $[\![\hat{\text{ɔ}}_i(\dot{C})]\!] = \text{ɔ}_i([\![\dot{C}]\!])$.

Proof 8. *For part 1, it is easy to see that $[\![\dot{\neg}\dot{C}]\!] \cap [\![\dot{C}]\!] = \varnothing$ which implies $[\![\dot{\neg}\dot{C}]\!] \subseteq \overline{[\![\dot{C}]\!]}$. For a proof of the other direction of part 1, for each tuple $t \in \overline{[\![\dot{C}]\!]}$, we construct the star-tuple $\dot{t}$, where $\dot{t}(i) = t(i)$ if $t(i) \in A$, and $\dot{t}(i) = *$ if $t(i) \notin A$. We then choose a subset SS'_t of SS_t where $(i, j) \in SS^+$ if and only if $t(i) = t(j)$. We insert in $\dot{t}(n+1)$ the condition $(i = j)$ for each $(i, j) \in SS'_t$, and $(i \neq j)$ for each $(i, j) \in SS_t \setminus SS'_t$, Then clearly $t \in [\![\{\dot{t}\}]\!]$ and $\dot{t} \in \dot{A}$. It remains to show that $\dot{t} \in \dot{\neg}\dot{C}$. Towards a* *contradiction, suppose that there is a star-tuple* $\dot{u} \in \dot{C}$ *such that* $\dot{t} \wedge \dot{u} \neq \dot{t}_{\varnothing}$. *In other words,* $\dot{t}(n+1) \cup \dot{u}(n+1)$ *is satisfiable. Thus, whenever* $\dot{t}(i) \in \mathbb{D}$, *we must have* $\dot{u}(i) = \dot{t}(i) = t(i) \in A$. *Furthermore, for each* $(i,j) \in SS_t$ *there is a literal involving* i *and* j *in* $\dot{t}(n+1)$. *Therefore* $\dot{u}(n+1)$ *can consist of only a subset of these literals. It follows that* $t \leq \dot{t} \leq \dot{u} \in \dot{C}$, *meaning that* $t \in [\![\dot{C}]\!]$, *contradicting our initial assumption.*

For part 2 of the theorem, let $t \in [\![\dot{\mathsf{\supset}}_i(\dot{C})]\!]$. *Then* $t \in [\![\{\dot{t} \in \dot{A} : (\dot{\mathsf{c}}_i(\{\dot{t}\}) \Cap \dot{A}) \leq (\dot{C} \Cap \dot{A})\}]\!]$. *Therefore there is a star tuple* $\dot{t} \in \dot{A}$ *such that,* $t \leq \dot{t}$ *and* $(\dot{\mathsf{c}}_i(\{\dot{t}\}) \Cap \dot{A}) \leq (\dot{C} \Cap \dot{A})$. *Lemma 3 then gives us* $[\![\dot{\mathsf{c}}_i(\{\dot{t}\})]\!] \subseteq [\![\dot{C}]\!]$, *and Theorem 5, part 4 (which still holds for extended star-cylinders) tell us that* $[\![\dot{\mathsf{c}}_i(\{\dot{t}\})]\!] = \mathsf{c}_i([\![\{\dot{t}\}]\!])$ *which implies* $[\![\dot{\mathsf{c}}_i(\{\dot{t}\})]\!] \subseteq [\![\dot{C}]\!]$. *By the definition of inner cylindrification in CA, the last containment implies that* $[\![\{\dot{t}\}]\!] \subseteq \mathsf{\supset}_i([\![\dot{C}]\!])$. *Consequently* $t \in \mathsf{\supset}_i([\![\dot{C}]\!])$.

For the other direction, let $t \in \mathsf{\supset}_i([\![\dot{C}]\!])$, *which implies* $\mathsf{c}_i(\{t\}) \subseteq [\![\dot{C}]\!]$. *Then there is a star-tuple* $\dot{t} \in \dot{C} \Cap \dot{A}$, *such that* $\mathsf{c}_i(\{t\}) \subseteq \mathsf{c}_i([\![\{\dot{t}\}]\!]) \subseteq [\![\dot{C}]\!]$. *Consequently,* $[\![\dot{\mathsf{c}}_i(\dot{t})]\!] \subseteq [\![\dot{C}]\!]$, *which by Lemma 3 proves that* $(\dot{\mathsf{c}}_i(\{\dot{t}\}) \Cap \dot{A}) \leq (\dot{C} \Cap \dot{A})$. *Moreover, the first part of Lemma 3 implies that* $t \in [\![\{\dot{t} \in \dot{A} : (\dot{\mathsf{c}}_i(\{\dot{t}\}) \Cap \dot{A}) \leq (\dot{C} \Cap \dot{A})\}]\!]$.

3.3.4 Equivalenvce of CA and SCA

Now, we can conclude our main result in this section. Here, we will prove that SCA and CA are equivalent and this implies that SCA, CA and FO are equivalent.

Corollary 2. *For every corresponding pair of* SCA_n*-expression* $\dot{E}$ *and* CA_n*-expression* E*, it holds that*

$$[\![\dot{E}(\dot{\mathbf{C}})]\!] = E([\![\dot{\mathbf{C}}]\!])$$

for every sequence of n-dimensional extended star-cylinders and star-diagonals $\dot{\mathbf{C}}$.

Proof 9. *This is a direct consequence of Theorem 5 and Theorem 6.*

3.4 Stored Databases and Query Evaluation

We now show how to use the cylindric star-algebra to evaluate FO-queries on stored databases containing universal nulls.

3.4.1 Universal Nulls (u-databases)

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 2 and 11 show stored databases. Everything that is defined for star-cylinders applies to k-ary star-relations. The 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 $\dot{\mathbf{R}}$.

Definition 17. *Let $\dot{t}$ be a k-ary star-tuple, and $n \geq k$. Then $\dot{\mathsf{h}}^n(\dot{t})$, the n-expansion of $\dot{t}$, is the n-ary star-tuple $\dot{u}$, where*

$$\dot{u}(i) = \begin{cases} \dot{t}(i) & \text{if } i \in \{1, \ldots, k\} \\ * & \text{if } i \in \{k+1, \ldots, n\} \\ \dot{t}(k+1) & \text{if } i = n+1, \end{cases}$$

For a stored relation $\dot{R}$ and stored database $\dot{\mathbf{R}}$ we have

$$\begin{aligned} \dot{\mathsf{h}}^n(\dot{R}) &= \{\dot{\mathsf{h}}^n(\dot{t}) \, : \, \dot{t} \in \dot{R}\} \\ \dot{\mathsf{h}}^n(\dot{\mathbf{R}}) &= (\dot{\mathsf{h}}^n(\dot{R}_1), \ldots, \dot{\mathsf{h}}^n(\dot{R}_m), \dot{\mathsf{d}}_{ij})_{i,j}. \end{aligned}$$

In other words, $\dot{\mathsf{h}}^n(\dot{\mathbf{R}})$ is the sequence of star-cylinders obtained by moving the conditions in column $k+1$ to column $n+1$, and filling columns $k+1, \ldots, n$ with ”*”’s in each k-ary relation. Examples 2 and 4 illustrate the expansion of star-relations.

On the other hand, a k-ary star-relation $\dot{R}$ can also be viewed as a finite representative of the infinite relation $[\![\dot{R}]\!] = \{t \in \mathbb{D}^k : t \preceq \dot{t} \text{ for some } \dot{t} \in \dot{R}\}$, and the stored database $\dot{\mathbf{R}}$ a finite representative of the infinite instance $I(\dot{\mathbf{R}})$, as in the following definition.

Definition 18. *Let $\dot{\mathbf{R}} = (\dot{R}_1, \dots, \dot{R}_m)$ be a stored database. Then the instance defined by $\dot{\mathbf{R}}$ is*

$$I(\dot{\mathbf{R}}) = ([\![\dot{R}_1]\!], \dots, [\![\dot{R}_m]\!], \{(a,a) : a \in \mathbb{D}\}).$$

The instance and expansion of $\dot{\mathbf{R}}$ are related as follows.

Lemma 4. $[\![\dot{\mathsf{h}}^n(\dot{\mathbf{R}})]\!] \;=\; \mathsf{h}^n(I(\dot{\mathbf{R}}))$.

Proof 10. *Follows directly from the definitions of h^n, $\dot{\mathsf{h}}^n$ and $[\![\,]\!]$.*

We are now ready for our main result.

Theorem 7. *For every FO_n-formula φ there is an (extended) SCA_n expression $\dot{E}_\varphi$, such that for every stored database $\dot{\mathbf{R}}$*

$$\mathsf{h}^n(\varphi^{I(\dot{\mathbf{R}})}) \;=\; [\![\dot{E}_\varphi(\dot{\mathsf{h}}^n(\dot{\mathbf{R}}))]\!].$$

Proof 11. $\mathsf{h}^n(\varphi^{I(\dot{\mathbf{R}})}) \;=\; E_\varphi(\mathsf{h}^n(I(\dot{\mathbf{R}})) \;=\; E_\varphi([\![\dot{\mathsf{h}}^n(\dot{\mathbf{R}})]\!]) \;=\; [\![\dot{E}_\varphi(\dot{\mathsf{h}}^n(\dot{\mathbf{R}}))]\!]$.

The first equality follows from Theorem 3, the second from Lemma 4, and the third from Corollaries 1 and 2.

Figure 5 Shows the result of Theorems 3, 4, 5 and 6. Clearly, Theorem 7 is the consequence of these theorems.

$$\begin{tikzcd}
\mathsf{FO} \arrow[r, shift left, "Theorm\ 3"] & \mathsf{CA} \arrow[l, shift left, "Theorm\ 4"] \arrow[r, leftrightarrow, shift left, "Theorm\ 5"] \arrow[r, leftrightarrow, shift right, "Theorm\ 6"'] & \mathsf{SCA}
\end{tikzcd}$$

Figure 5: Commutative Diagram of FO, CA and SCA

3.4.2 Adding Existential Nulls

In this thesis we follow the model-theoretic approach of [27]. The elements in $\mathbb{D}$ represent known objects, whereas elements in $\mathbb{N}$ represent generic objects. Each generic object could turn out to be equal to one of the known objects, to another generic object, or represent an object different from all other objects. We extend our notation to include $univ(I)$, the *universe* of instance I. So far we have assumed that $univ(I) = \mathbb{D}$, but in this section we allow instances whose universe is any set between $\mathbb{D}$ and $\mathbb{D} \cup \mathbb{N}$. We are lead to the following definitions.

Definition 19. *Let h be a mapping on $\mathbb{D} \cup \mathbb{N}$ that is identity on $\mathbb{D}$, and let I and J be instances (over $\mathscr{R}$), such that $h(univ(I)) = univ(J)$. We say that h is a* possible world homomorphism *from I to J, if $h(R_p^I) \subseteq R_p^J$ for all p, and $h(\approx^I) = \approx^J$. This is denoted $I \rightarrow_h J$.*

Definition 20. *Let I be an instance with $\mathbb{D} \subseteq univ(I) \subseteq \mathbb{D} \cup \mathbb{N}$. Then the set of instances represented by I is*

$$Rep(I) \;=\; \{J \,:\, \exists\, h \text{ s.t. } I \rightarrow_h J\}.$$

We can now formulate the (standard) notion of a *certain answer* to a query.[9] By FO$^+$ below we mean the set of all FO-formulas not using negation.

Definition 21. *Let I be an incomplete instance and φ an FO$^+$-formula. The certain answer to φ on I is*

$$Cert(\varphi, I) \;=\; \bigcap_{J \in Rep(I)} \varphi^J.$$

We now extend positive n-dimensional cylinders to be subsets of $(\mathbb{D} \cup \mathbb{N})^n$, and use the notation $univ(\mathbf{C})$ and $univ(C)$ with the obvious meanings. This also applies to the notation $\mathbf{C} \rightarrow_h \mathbf{D}$, and $Rep(\mathbf{C})$. Cylinders C with $\mathbb{D} \subseteq univ(C) \subseteq \mathbb{D} \cup \mathbb{N}$ will

[9]Here we of course assume that valuations have range $univ(J)$, and that other details are adjusted accordingly.

be called *naive cylinders*. The operators of the positive cylindric set algebra CA^+ remain the same, except $\mathbb{D}$ is substituted with $univ(\mathbf{C})$ or $univ(C)$, i.e. we use naive evaluation. For instance, the outer cylindrification now becomes

$$\mathsf{c}_i(C) = \{t \in univ(C)^n \,:\, t(i/x) \in C, \text{ for some } x \in univ(C)\}.$$

The crucial property of the positive cylindric set algebra is the following.

Theorem 8. *Let E be an expression in CA_n^+, and $\mathbf{C}$ and $\mathbf{D}$ sequences of n-dimensional naive cylinders and diagonals. If $\mathbf{C} \to_h \mathbf{D}$ for some possible world homomorphism h, then $E(\mathbf{C}) \to_h E(\mathbf{D})$.*

Proof 12. *Suppose $\mathbf{C} \to_h \mathbf{D}$. We show by induction on the structure of E that $E(\mathbf{C}) \to_h E(\mathbf{D})$.*

- *For $E = \mathsf{C}_i$ and $E = \mathsf{d}_{ij}$ the claim follows directly from the definition of a possible world homomorphism.*

- *Let $t \in h(F \bigcup G\,(\mathbf{C})) = h(F(\mathbf{C}) \cup G(\mathbf{C})) = h(F(\mathbf{C})) \cup h(G(\mathbf{C}))$. Then there is a tuple s in $F(\mathbf{C})$ or in $G(\mathbf{C})$ such that $t = h(s)$. If s is in, say, $F(\mathbf{C})$, then, since $F(\mathbf{C}) \to_h F(\mathbf{D})$ and $F(\mathbf{D}) \subseteq F \bigcup G\,(\mathbf{D})$, it follows that $t = h(s) \in F \bigcup G\,(\mathbf{D})$.*

- *Let $t \in h(F \bigcap G\,(\mathbf{C})) = h(F(\mathbf{C}) \cap G(\mathbf{C}))$. Then there is a tuple s in $F(\mathbf{C})$ and a tuple s' in $G(\mathbf{C})$ such that $t = h(s) = h(s')$. Thus $h(s) \in h(F(\mathbf{C})) \subseteq F(\mathbf{D})$, and $h(s') \in h(G(\mathbf{C})) \subseteq G(\mathbf{D})$. Consequently $t = h(s) = h(s') \in F(\mathbf{D}) \cap G(\mathbf{D}) = F \bigcap G\,(\mathbf{D})$.*

- *Let $t \in h(\mathsf{c}_i(F(\mathbf{C})))$. Then there is an $s \in \mathsf{c}_i(F(\mathbf{C}))$, such that $t = h(s)$. Furthermore, $s(i/x) \in F(\mathbf{C})$ for some $x \in univ(\mathbf{C})$. Then $h(s(i/x)) = h(s)(i/h(x)) \in h(F(\mathbf{C}))$, for $h(x) \in h(univ(\mathbf{C})) = univ(\mathbf{D})$, This means that $t = h(s) \in \mathsf{c}_i(F(\mathbf{D}))$.*

- *Let $t \in h(\text{ɔ}_i(F(\mathbf{C})))$. Then there is an $s \in \text{ɔ}_i(F(\mathbf{C}))$, such that $t = h(s)$. Furthermore, $s(i/x) \in F(\mathbf{C})$ for all $x \in univ(\mathbf{C})$. Then $h(s(i/x)) = h(s)(i/h(x)) = t(i/h(x)) \in h(F(\mathbf{C}))$ for all $x \in univ(\mathbf{C})$. In other words, $t(i/y) \in h(F(\mathbf{C})) \subseteq F(\mathbf{D})$ for all $y \in h(univ(\mathbf{C})) = univ(\mathbf{D})$. Thus $t \in \text{ɔ}_i(F(\mathbf{D}))$*

Also, for an n-dimensional naive cylinder C, we denote the subset $C \cap \mathbb{D}^n$ by $C_\downarrow$. We now have our main "naive evaluation" theorem.

Theorem 9. *Let $\mathbf{C}$ be a sequence of n-dimensional naive cylinders and diagonals, and let E be an expression in CA_n^+. Then*

$$E(\mathbf{C})_\downarrow \ = \bigcap_{\mathbf{D} \in Rep(\mathbf{C})} E(\mathbf{D}).$$

Proof 13. *Let $t \in E(\mathbf{C})_\downarrow \subseteq E(\mathbf{C})$, and $\mathbf{D} \in Rep(\mathbf{C})$. Since $\mathbf{C} \to_h \mathbf{D}$ for some possible world homomorphism h, by Theorem 8, $h(t) \in E(\mathbf{D})$. Since t is all constants, $h(t) = t$ for all h. In other words, $t \in E(\mathbf{D})$, for all $\mathbf{D} \in Rep(\mathbf{C})$.*

For the $\supseteq$-direction, let $t \in \bigcap_{\mathbf{D} \in Rep(\mathbf{C})} E(\mathbf{D})$. Then $t \in \mathbb{D}^n$, and for all possible world homomorphisms h it holds that $t \in E(h(\mathbf{C}))$. Since identity is a valid h, it follows that $t \in E(\mathbf{C})$, and since t is all constants we have $t \in E(\mathbf{C})_\downarrow$.

3.4.3 Mixing Existential and Universal Nulls

We want to achieve a representation mechanism able to handle both universal nulls and naive existential nulls. To this end we need the following definition.

Definition 22. *A naive n-dimensional (positive) star-cylinder is a finite subset $\ddot{C}$ of $(\mathbb{D} \cup \mathbb{N} \cup \{*\})^n \times \wp(\Theta_n)$. A naive diagonal is defined as $\ddot{d}_{ij} = \{(x, x) : x \in univ(\ddot{\mathbf{C}})\}$. A sequence of n-dimensional star-cylinders and diagonals is denoted $\ddot{\mathbf{C}}$.*

After this we extend Definitions 8, 9, and 11 in Section 3 from star-cylinders to naive star-cylinders, by replacing $\mathbb{D}$ with $univ(\ddot{\mathbf{C}})$ or $univ(\ddot{C})$ everywhere. Theorem 8 will still hold, but Corollary 1 only holds in the weakened form as Corollary 3 below. First we need two lemmas and a definition.

Lemma 5. *Suppose all possible world homomorphisms h are extended by letting $h(*) = *$. Let $\ddot{C}$ be an n-dimensional naive star-cylinder. Then*

$$h([\![\ddot{C}]\!]) = [\![h(\ddot{C})]\!],$$

for all possible world homomorphisms h.

Proof 14. *Let $t \in h([\![\ddot{C}]\!])$. Then there exists a tuple $u \in [\![\ddot{C}]\!]$, such that $t = h(u)$. Also there exists a naive star-tuple $\ddot{u} \in [\![\ddot{C}]\!]$, such that $u \preceq \ddot{u}$. Now it is sufficient to show that $t \preceq h(\ddot{u})$, for all $i \in \{1, 2, \ldots, n\}$.*

If $\ddot{u}(i) \in \mathbb{D}$, then $u(i) = \ddot{u}(i)$. Also, homomorphisms are identity on constants and therefore $h(u(i)) = u(i)$, which implies $t(i) = u(i)$.

If $\ddot{u}(i) = *$, then $u(i) \in univ(\ddot{C})$. As a result $h(u(i)) \in univ(h(\ddot{C}))$, which implies $t(i) \preceq * = h(\ddot{u}(i))$, since homomorphisms map stars to themselves.

If $\ddot{u}(i) \in \mathbb{N}$, then $u(i) = \ddot{u}(i)$, which implies $t(i) = h(u(i)) = h(\ddot{u}(i))$.

For the other direction, let $t \in [\![h(\ddot{C})]\!]$. Then there exists a tuple $\ddot{t} \in h(\ddot{C})$ and a tuple $\ddot{u} \in \ddot{C}$, such that $t \preceq \ddot{t}$ and $\ddot{t} = h(\ddot{u})$. Consequently, $t \preceq h(\ddot{u})$. We show that we can find a tuple $u \in [\![\ddot{u}]\!]$ such that $h(u) = t$.

If $\ddot{u}(i) \in \mathbb{D}$, then $u(i) = \ddot{u}(i)$. Since h is the identity on constants $h(\ddot{u}(i)) = \ddot{u}(i)$, which implies $t(i) = u(i)$.

If $\ddot{u}(i) = *$, then $h(\ddot{u}(i)) = *$. As h is onto $univ(h(\ddot{C}))$, it follows that there is a value $\ddot{u}(i) \in univ(\ddot{C})$, such that $h(u(i)) = t(i)$.

If $\ddot{u}(i) \in \mathbb{N}$, then $u(i) = \ddot{u}(i)$, which implies $t(i) = h(\ddot{u}(i)) = h(u(i))$.

Definition 23. *Let $\mathcal{I}$ and $\mathcal{J}$ be sets of instances. We say that $\mathcal{I}$ and $\mathcal{J}$ are co-initial, denoted $\mathcal{I} \sim \mathcal{J}$, if for each instance $J \in \mathcal{J}$ there is an instance $I \in \mathcal{I}$, and a possible world homomorphism h, such that $I \rightarrow_h J$, and vice-versa.*

In the context of naive star-cylinders Corollary 1 will be weakened as follows.

Corollary 3. *For every SCA_n^+-expression $\dot{E}$ and the corresponding CA_n-expression E, it holds that*

$$Rep([\![\dot{E}(\ddot{\mathbf{C}})]\!]) \sim E(Rep([\![\ddot{\mathbf{C}}]\!])),$$

for every sequence of n-dimensional naive star-cylinders and star-diagonals $\ddot{\mathbf{C}}$.

Proof 15. *We have $Rep([\![\dot{E}(\ddot{\mathbf{C}})]\!]) \sim Rep(E([\![\ddot{\mathbf{C}}]\!]))$ from Corollary 1. It remains to show that $Rep(E([\![\ddot{\mathbf{C}}]\!]) \sim E(Rep([\![\ddot{\mathbf{C}}]\!]))$. Let's denote $[\![\ddot{\mathbf{C}}]\!]$ by $\mathbf{C}$. We'll show that $Rep(E(\mathbf{C})) \sim E(Rep(\mathbf{C}))$.*

Let $D \in E(Rep(\mathbf{C}))$, meaning that $D = E(\mathbf{C}')$ for some $\mathbf{C}' \in Rep(\mathbf{C})$. Then there is a possible world homomorphism h such that $\mathbf{C} \to_h \mathbf{C}'$. Theorem 8 then yields $E(\mathbf{C}) \to_h E(\mathbf{C}')$, and since $E(\mathbf{C}) \in Rep(E(\mathbf{C}))$ one direction of Definition 23 is satisfied.

Then let $D \in Rep(E(\mathbf{C}))$. Then there is a possible world homeomorphism h, such that $E(\mathbf{C}) \to_h D$. Since $E(\mathbf{C}) \in E(Rep(\mathbf{C}))$, it means that the vice-versa direction is also satisfied.

3.4.4 Naive Evaluation of Existential Nulls

We extend Definition 20 from infinite instances to sequences of naive star-cylinders as follows.

Definition 24. *Let $\ddot{\mathbf{C}}$ be a sequence of n-dimensional naive star-cylinders and diagonals with $univ(\ddot{\mathbf{C}}) = \mathbb{D} \cup \mathbb{N}$. Then the (infinite) set of (infinite) n-dimensional cylinders represented by $\ddot{\mathbf{C}}$ is*

$$Rep([\![\ddot{\mathbf{C}}]\!]) = \{\mathbf{D} : [\![\ddot{\mathbf{C}}]\!] \to_h \mathbf{D}\}.$$

For a naive star-cylinder $\ddot{\mathbf{C}}$ we let $\ddot{\mathbf{C}}_\downarrow = \ddot{\mathbf{C}} \cap (\mathbb{D} \cup \{*\})^n$. We note that obviously $[\![\ddot{\mathbf{C}}_\downarrow]\!] = ([\![\ddot{\mathbf{C}}]\!])_\downarrow$, and that if $Rep(\ddot{\mathbf{C}}) \sim Rep(\ddot{\mathbf{D}})$ then $\ddot{\mathbf{C}}_\downarrow = \ddot{\mathbf{D}}_\downarrow$. We now have the main result of this section.

Theorem 10. *For every SCA⁺-expression $\dot{E}$ and the corresponding CA⁺-expression E, it holds that*

$$[\![\dot{E}(\ddot{\mathbf{C}})_{\downarrow}]\!] = \bigcap_{\mathbf{D} \in Rep([\![\ddot{\mathbf{C}}]\!])} E(\mathbf{D}).$$

for every sequence $\ddot{\mathbf{C}}$ of naive star-cylinders and diagonals.

Proof 16. $[\![\dot{E}(\ddot{\mathbf{C}})_{\downarrow}]\!] = [\![\dot{E}(\ddot{\mathbf{C}})]\!]_{\downarrow} = (E([\![\ddot{\mathbf{C}}]\!]))_{\downarrow} = \bigcap_{\mathbf{C} \in Rep([\![\ddot{\mathbf{C}}]\!])} E(\mathbf{C})$. *The second equality follows from Corollary 3, the third from Theorem 9.*

Stored databases with universal and existential nulls (ue-databases)

We extend the Definitions 17 and 18 of Section 3.4 from stored databases to naive stored databases (ue-databases) by substituting $\mathbb{D}$ with $\mathbb{D} \cup \mathbb{N}$ everywhere. Lemma 4 then becomes

Lemma 6. *Let $\ddot{\mathbf{C}}$ be a stored ue-database with universe $\mathbb{D} \cup \mathbb{N}$. Then $[\![\dot{\mathsf{h}}^n(\ddot{\mathbf{R}})]\!] = \mathsf{h}^n(I(\ddot{\mathbf{R}}))$.*

We first note that Theorem 7 in the ue-setting becomes

Theorem 11. *For every FO^+_n-formula φ there is an SCA^+_n expression $\dot{E}_\varphi$, such that*

$$[\![\dot{E}_\varphi(\dot{\mathsf{h}}^n(\ddot{\mathbf{R}}))]\!] = \mathsf{h}^n(\varphi^{I(\ddot{\mathbf{R}})})$$

for every stored ue-database $\ddot{\mathbf{R}}$

We also have

Theorem 12. *For every FO^+_n-formula φ there is a CA^+_n expression $\dot{E}_\varphi$, such that*

$$Rep([\![\dot{E}_\varphi(\dot{\mathsf{h}}^n(\ddot{\mathbf{R}}))]\!]) \sim \{\mathsf{h}^n(\varphi^J) : J \in Rep([\![\ddot{\mathbf{R}}]\!])\}$$

for every stored ue-database $\ddot{\mathbf{R}}$

We have now arrived our main theorem for ue-databases.

Theorem 13. *For every FO_n^+-formula φ there is an SCA_n^+ expression $\dot{E}_\varphi$, such that*

$$[\![\dot{E}_\varphi(\dot{\mathsf{h}}^n(\ddot{\mathbf{R}}))_\downarrow]\!] = \bigcap_{J \in Rep([\![\ddot{\mathbf{R}}]\!])} \mathsf{h}^n(\varphi^J)$$

for every stored ue-database $\ddot{\mathbf{R}}$

Proof 17. *We have* $\{\mathsf{h}^n(\varphi^J) : J \in Rep([\![\ddot{\mathbf{R}}]\!])\} \sim Rep([\![\dot{E}_\varphi(\dot{\mathsf{h}}^n(\ddot{\mathbf{R}}))]\!])$ *by Theorem 12. Hence*

$\bigcap_{J \in Rep([\![\ddot{\mathbf{R}}]\!])} \mathsf{h}^n(\varphi^J) = \bigcap Rep([\![\dot{E}_\varphi(\dot{\mathsf{h}}^n(\ddot{\mathbf{R}}))]\!]) = ([\![\dot{E}_\varphi(\dot{\mathsf{h}}^n(\ddot{\mathbf{R}}))]\!])_\downarrow = [\![\dot{E}_\varphi(\dot{\mathsf{h}}^n(\ddot{\mathbf{R}}))_\downarrow]\!]$.

4 Query Evaluation in Four

This chapter starts with the formal definition of four-valued databases and follows by showing that how the four-valued queries can be Evaluated against four-valued Databases. To reach this goal, we first explain how four-valued instances can be loosely decomposed into a pair of two-valued instances separately representing the positive and negative information. These two-valued instances are equipped with the traditional *Open World Assumption* . Next we show how to decompose the four-valued queries into two-valued queries regarding positive and negative information. We conclude this section by showing how to evaluate queries and aggregate information back into the four-valued database. Later, in Section 4.5 and Section 4.5.1 we focus on implication in data exchange in order to show that how to evaluate tuple generating dependencies in four-valued logic.

4.1 Four Valued Logic

This section introduces Belnap's four-valued logic [11] in the context of databases. We first formally define our database model.

4.1.1 Four-valued instances.

We now extend the hitherto used classical two-valued instances to four-valued ones, according to Belnap's logic. The Boolean part of Belnap's four-valued logic is characterized by the following truth-tables. In the sequel we shall denote an operator, such as $\wedge$, by $\wedge^{\mathbf{4}}$ or $\wedge^{\mathbf{2}}$, to emphasize the context. Note that Belnap's logic is an extension of Kleene's strong three-valued logic, which again is an extension of classical two-valued logic.

$\wedge^{\mathbf{4}}$	true	false	$\top$	$\bot$
true	true	false	$\top$	$\bot$
false	false	false	false	false
$\top$	$\top$	false	$\top$	false
$\bot$	$\bot$	false	false	$\bot$

$\vee^{\mathbf{4}}$	true	false	$\top$	$\bot$
true	true	true	true	true
false	true	false	$\top$	$\bot$
$\top$	true	$\top$	$\top$	true
$\bot$	true	$\bot$	true	$\bot$

$\neg^{\mathbf{4}}$	
true	false
false	true
$\top$	$\top$
$\bot$	$\bot$

We thus arrive at a Boolean algebra

$$(\{\mathsf{true}, \mathsf{false}, \bot, \top\}, \wedge, \vee, \neg, \mathsf{true}, \mathsf{false}, \top, \bot) \tag{5}$$

The role of the designated elements $\mathsf{true}, \mathsf{false}, \top, \bot$ will become clear later below.

Definition 25. *A four-valued instance I is a mapping from atoms $R_p(\bar{a})$ (and equality atoms $a_i \approx a_j$) to $\{\mathsf{true}, \mathsf{false}, \bot, \top\}$, i.e. $R_p(\bar{a})^I \in \{\mathsf{true}, \mathsf{false}, \bot, \top\}$. The mapping I is then extended to a homomorphism from FO-sentences to the algebra 5 by recursively defining*

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

4.1.2 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 $\bar{x}$. The result of applying query $\varphi(\bar{x})$ to a four-valued instance I will yield the following four types of answers:

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

Example 13. *Consider binary relations F(ollows) and H(obbies), where $F(x_1, x_2)$ means that user x_1 follows user x_2 on a social media site, and $H(x_1, x_3)$ means that x_3 is a hobby of user x_1. Let the database instance I be the following.*

F		F^I
Alex	*Bob*	true
Bob	*Alex*	$\top$
Bob	*Alice*	true

H		H^I
Alex	*Movie*	$\top$
Alex	*Music*	true
Alice	*Music*	true
Bob	*Movies*	false

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 $F^I(Alice, Alex) = \bot$. 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 $\varphi(x_1)$, where*

$$\varphi(x_1) = \exists x_2\, F(x_1, x_2) \wedge \neg H(x_2, Movies).$$

Then, assuming that $\mathbb{D}$ consists of the values in the instance only, $\varphi(x_1)$ will be evaluated as follows.

$$
\begin{aligned}
\varphi(Alex)^I \;=\; & \\
& \bigvee\nolimits^{\mathbf{4}}_{a\in\mathbb{D}} \Big(F(Alex,a)^I \wedge^{\mathbf{4}} (\neg^{\mathbf{4}}(H(a,Movies)^I))\Big) \\
&= \Big(F(Alex,Alex)^I \wedge^{\mathbf{4}} (\neg^{\mathbf{4}}(H(Alex,Movies)^I))\Big) \\
&\quad \vee^{\mathbf{4}} \Big(F(Alex,Bob)^I \wedge^{\mathbf{4}} (\neg^{\mathbf{4}}(H(Bob,Movies)^I))\Big) \\
&\quad \vee^{\mathbf{4}} \Big(F(Alex,Alice)^I \wedge^{\mathbf{4}} (\neg^{\mathbf{4}}(H(Alice,Movies)^I))\Big) \\
&= \Big(\bot \wedge^{\mathbf{4}} (\neg^{\mathbf{4}} \top)\Big) \vee^{\mathbf{4}} \Big(\mathsf{true} \wedge^{\mathbf{4}} (\neg^{\mathbf{4}} \mathsf{false})\Big) \vee^{\mathbf{4}} \Big(\bot \wedge^{\mathbf{4}} (\neg^{\mathbf{4}} \bot)\Big) \\
&= \Big(\bot \wedge^{\mathbf{4}} \top\Big) \vee^{\mathbf{4}} \Big(\mathsf{true} \wedge^{\mathbf{4}} \mathsf{true}\Big) \vee^{\mathbf{4}} \Big(\bot \wedge^{\mathbf{4}} \bot\Big) \\
&= \mathsf{true} \vee^{\mathbf{4}} \mathsf{true} \vee^{\mathbf{4}} \bot = \mathsf{true}
\end{aligned}
$$

Similarly for Bob and Alice,

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

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

4.2 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 $\bar{a} \in \mathbb{D}^{ar(R)}$,

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

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

$$
\begin{array}{llll}
R^{I^+\otimes I^-}(\bar{a}) = \mathsf{true} & \text{if } R^{I^+}(\bar{a}) = \mathsf{true} & \text{and} & R^{I^-}(\bar{a}) = \mathsf{false}; \\
R^{I^+\otimes I^-}(\bar{a}) = \mathsf{false} & \text{if } R^{I^+}(\bar{a}) = \mathsf{false} & \text{and} & R^{I^-}(\bar{a}) = \mathsf{true}; \\
R^{I^+\otimes I^-}(\bar{a}) = \top & \text{if } R^{I^+}(\bar{a}) = \mathsf{true} & \text{and} & R^{I^-}(\bar{a}) = \mathsf{true}; \\
R^{I^+\otimes I^-}(\bar{a}) = \bot & \text{if } R^{I^+}(\bar{a}) = \mathsf{false} & \text{and} & R^{I^-}(\bar{a}) = \mathsf{false}.
\end{array}
$$

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

Figure 6 graphically describes the procedure of decomposing a four-valued instance into, positive and negative two-valued instances. Also, it shows that from decomposed two-valued instances, by applying $\otimes$ operation defined here, the initial four-valued instance is obtained.

$$\begin{tikzcd}
 & I \arrow[dl] \arrow[dr] & \\
I^+ \arrow[dr] & \otimes \arrow[u] & I^- \arrow[dl] \\
 & I^\pm \arrow[u, dash] &
\end{tikzcd}$$

Figure 6: Decomposition Diagram of four-valued Instances

Example 14. *Consider binary relations F(ollows) and H(obbies) from Example 13. By applying the decomposition to relations F^I and H^I, the two-valued relation F^{I^+} is populated with tuples $(a_i, a_j) \in \mathbb{D}^2$, where $F^I(a_i, a_j) \in \{\mathsf{true}, \top\}$, and relation H^{I^+} is populated with tuples $(a_i, a_j) \in \mathbb{D}^2$, where $H^I(a_i, a_j) \in \{\mathsf{true}, \top\}$. Similarly, F^{I^-} is populated with tuples $(a_i, a_j) \in \mathbb{D}^2$, where $F^I(a_i, a_j) \in \{\mathsf{false}, \top\}$, and H^{I^-} is populated with tuples $(a_i, a_j) \in \mathbb{D}^2$, where $H^I(a_i, a_j) \in \{\mathsf{false}, \top\}$. 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 $\wedge^{\mathbf{2}}$ and $\vee^{\mathbf{2}}$ denote the classical two-valued operators. Of course, when restricted to $\{\mathsf{true}, \mathsf{false}\}$, two- and four-valued conjunction and disjunction agree.

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

Lemma 8. *Let* $p, q \in \{\mathsf{true}, \mathsf{false}, \top, \bot\}$. *Suppose* $p^{\pm} = (p^{+}, p^{-})$, *and* $q^{\pm} = (q^{+}, q^{-})$. *Then*

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

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 26. *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	$\varphi^{I^{+}}$	$\varphi^{I^{-}}$
$R(\bar{a})$	$R(\bar{a})^{I^{+}}$	$R(\bar{a})^{I^{-}}$
$\neg\psi$	$\psi^{I^{-}}$	$\psi^{I^{+}}$
$a_i \approx a_j$	$(a_i \approx a_j)^{I^{+}}$	$(a_i \approx a_j)^{I^{-}}$
$\psi \wedge \xi$	$\psi^{I^{+}} \wedge^{\mathbf{2}} \xi^{I^{+}}$	$\psi^{I^{-}} \vee^{\mathbf{2}} \xi^{I^{-}}$
$\psi \vee \xi$	$\psi^{I^{+}} \vee^{\mathbf{2}} \xi^{I^{+}}$	$\psi^{I^{-}} \wedge^{\mathbf{2}} \xi^{I^{-}}$
$\exists x\, \psi(x)$	$\bigvee^{\mathbf{2}}_{a \in \mathbb{D}} \psi^{I^{+}}(a)$	$\bigwedge^{\mathbf{2}}_{a \in \mathbb{D}} \psi^{I^{-}}(a)$
$\forall x\, \psi(x)$	$\bigwedge^{\mathbf{2}}_{a \in \mathbb{D}} \psi^{I^{+}}(a)$	$\bigvee^{\mathbf{2}}_{a \in \mathbb{D}} \psi^{I^{-}}(a)$

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

Theorem 14. *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 18. *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 7. For the inductive step, if* $\varphi = \psi \wedge \xi$, *we have*

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

where we assume that $\psi^I = \psi^{I^+} \otimes \psi^{I^-}$*, and* $\xi^I = \xi^{I^+} \otimes \xi^{I^-}$*. By Lemma 8 we then have*

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

Definition 26 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.*

$$\begin{tikzcd}
 & I \arrow[dl] \arrow[dr] \arrow[rrr] & & & \varphi^I \arrow[dl] \arrow[dr] & \\
I^+ \arrow[dr] & \otimes \arrow[u] & I^- \arrow[dl] & \varphi^{I^+} \arrow[dr] & \otimes \arrow[u] & \varphi^{I^-} \arrow[dl] \\
 & I^\pm \arrow[u, dash] \arrow[rrr] & & & \varphi^{I^\pm} \arrow[u, dash] &
\end{tikzcd}$$

Figure 7: Commutative diagram of query decomposition

Figure 7 illustrates the procedure of decomposing the four-valued queries and four-valued instances. As it is shown in Theorem 14, from decomposition by applying $\otimes$

operation, the evaluation of four-valued query against the initial four-valued instance can be obtained.

Corollary 4. *Let φ be an FO-sentence, I a four-valued instance, and $I^{\pm} = (I^+, I^-)$ its decomposition. Then*

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

Example 15. *Let $\varphi(x)$ be the query $\exists y\, F(x,y) \wedge \neg H(y, Movies)$. from Example 13. Then $(\varphi^{I^+}, \varphi^{I^-})$ is evaluated as follows*

$\varphi^{I^+}(Alex) \quad = \vee^{\mathbf{2}}_{a \in \mathbb{D}} \left(F^{I^+}(Alex, a) \ \wedge^{\mathbf{2}} \ H^{I^-}(a, Movies)\right) \quad = \mathsf{true}$

$\varphi^{I^-}(Alex) \quad = \wedge^{\mathbf{2}}_{a \in \mathbb{D}} \left(F^{I^-}(Alex, a) \ \vee^{\mathbf{2}} \ H^{I^+}(a, Movies)\right) \quad = \mathsf{false}$

Similarly

$\varphi^{I^+}(Bob) = \mathsf{true}$ *and* $\varphi^{I^-}(Bob) = \mathsf{true}$

$\varphi^{I^+}(Alice) = \mathsf{false}$ *and* $\varphi^{I^-}(Alice) = \mathsf{false}$

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

φ^{I^+}	φ^{I^-}
Alex	*Bob*
Bob	

The final answers can now be composed according to Corollary 4, yielding $\textit{true}(\varphi, I) = \{(Alex)\}$, $\textit{false}(\varphi, I) = \{\}$, $\textit{inc}(\varphi, I) = \{(Bob)\}$, *and* $\textit{unk}(\varphi, I) = \{(Alice)\}$.

4.3 Adding 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 Section 3.3 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 sets of facts, we need *Universal Nulls* "$*$." We then show how the *Star Cylindric Algebra* described in Section 3.3 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 16. *Consider binary relations F(ollows) and H(obbies) from Examples 13 and 14. 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:

$$\varphi(x_4) = \exists x_2 \exists x_3 \forall x_1 \Big(F(x_1, x_2) \wedge H(x_3, x_4) \wedge (x_2 \approx x_3) \Big) \tag{6}$$

The answer to our example query is $\{(Movies), (Music)\}$*. 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 17. *Continuing Example 16, suppose all negative information we have acquired 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

$$\varphi^{I^+}(Chris) = \vee_{a\in\mathbb{D}}\, H^{I^+}(Chris, a) =$$
$$H^{I^+}(Chris,\ Movies) \vee H^{I^+}(Chris,\ Music) \vee$$
$$H^{I^+}(Chris,\ Movies) \vee H^{I^+}(Chris,\ ..) \vee \cdots =$$
$$\mathsf{false} \vee \mathsf{false} \vee \mathsf{false} \vee \mathsf{false} \vee \cdots = \mathsf{false}.$$

$$\varphi^{I^+}(Alice) = \vee_{a\in\mathbb{D}}\, H^{I^+}(Alice, a) =$$
$$H^{I^+}(Alice,\ Movies) \vee H^{I^+}(Alice,\ Music) \vee$$
$$H^{I^+}(Alice,\ Movies) \vee H^{I^+}(Alice,\ ..) \vee \cdots =$$
$$\mathsf{false} \vee \mathsf{true} \vee \mathsf{false} \vee \mathsf{false} \vee \cdots = \mathsf{true}.$$

$$\varphi^{I^+}(Bob) = \vee_{a\in\mathbb{D}}\, H^{I^+}(Bob, a) =$$
$$H^{I^+}(Bob,\ Movies) \vee H^{I^+}(Bob,\ Music) \vee$$
$$H^{I^+}(Bob,\ Movies) \vee H^{I^+}(Bob,\ ..) \vee \cdots =$$
$$\mathsf{false} \vee \mathsf{false} \vee \mathsf{true} \vee \mathsf{false} \vee \cdots = \mathsf{true}.$$

The negative part is evaluated as

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

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

4.4 Algebraic Evaluation

Recall from Section 3.2 that the *n-dimensional Star Cylindric Algebra* consists of operators *star union* $\dot{\cup}$, *star intersection* $\dot{\cap}$, *outer* $\dot{\mathsf{c}}_i$ and *inner* $\dot{\mathsf{ɔ}}_i$ *cylindrifications on dimension* i, *diagonals* $\dot{\mathsf{d}}_{ij}$, 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 $\dot{C}, \dot{C}'$ etc.

Example 18. *Continuing Example 16, in that database the atoms* $F(x_1, x_2)$ *and* $H(x_3, x_4)$ *of query* (6) *are represented by star-tables* $\dot{C}_F$ *and* $\dot{C}_H$*, 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.*

$\dot{C}_F$

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

$\dot{C}_H$

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

$\dot{\mathsf{d}}_{23}$

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

$\dot{C}'$

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*

$\dot{C}''$

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

$\dot{C}'''$

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

The star union $\uplus$ 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 $\{(a,*,*)\} \Cap \{(*,b,*)\} = \{(a,b,*)\}$. *The outer cylindrification* c_i *represents* $\exists x_i$ *and is obtained by replacing column* i *by an unconstrained "$*$." Inner cylindrification* $\mathsf{ɔ}_i$ *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 [36]. The translation of query* (6) *is the cylindric star-algebra expression*

$$\dot{\mathsf{c}}_2(\dot{\mathsf{c}}_3(\dot{\mathsf{ɔ}}_1((\dot{C}_F \Cap \dot{C}_H) \Cap \dot{\mathsf{d}}_{23}))) \qquad (7)$$

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

Example 19. *Consider the relation* $H(obbies)$ *from Examples 13 and 17. This relation is then stored as the two star-tables* $\dot{C}_{H^{I+}}$ *and* $\dot{C}_{H^{I-}}$ *below.*

$\dot{C}_{H^{I+}}$	
x_1	x_2
Alice	*Movies*
Alice	*Music*
Bob	*Movies*

$\dot{C}_{H^{I-}}$		
x_1	x_2	
Alice	*Volleyball*	
Bob	$*$	$2 \neq$ *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* $\mathsf{c}_2(\dot{C}_{H^{I^+}})$, *and the negative part as* $\mathsf{ↄ}_2(\dot{C}_{H^{I^-}})$.

$\mathsf{c}_2(\dot{C}_{H^{I^+}})$	
x_1	x_2
Alice	$*$
Bob	$*$

$\mathsf{ↄ}_2(\dot{C}_{H^{I^-}})$	
x_1	x_2
Chris	$*$

The answers are $\textsf{true}(\varphi, I) = \{(Alice, *, \{\}), (Bob, *, \{\})\}$ *and* $\textsf{false}(\varphi, I) = \{(Chris, *, \{\})\}$.

The above translation is summarized in the next theorem.

Theorem 15. [36] *For every query expressed as an FO formula* φ, *there is a star cylindric 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.*

4.5 Implication in Four-Valued Databases

So far, we have extended the basic Boolean operations to the four-valued case. When restricted to $\{\textsf{true}, \textsf{false}\}$ the four-valued extension coincides with the two-valued case. When it comes to the implication $\varphi \rightarrow \psi$, it will however no longer be equivalent with $\neg\varphi \vee \psi$. In order to arrive at the proper four-valued meaning of $\rightarrow$ we need to take a closer look at Belnap's logic [11].

As seen in Section 4.2, the truth-values $\textsf{true}$, $\textsf{false}$, $\top$, and $\bot$ can losslessy be represented as pairs of classical truth values $\textsf{true}$ and $\textsf{false}$, namely $(\textsf{true}, \textsf{false})$, $(\textsf{false}, \textsf{true})$, $(\textsf{true}, \textsf{true})$, and $(\textsf{false}, \textsf{false})$, respectively. Indeed, Belnap [11] explains a sentence φ

assigned $(\mathsf{true}, \mathsf{false})$ as "the computer has been told that φ is true," $(\mathsf{false}, \mathsf{true})$ as "the computer has been told that φ is false," $(\mathsf{true}, \mathsf{true})$ as "the computer has been told that φ is true, and that φ is false," and $(\mathsf{false}, \mathsf{false})$ as "the computer has not been told anything about the truth of φ." Belnap then proposes the following two lattice diagrams:

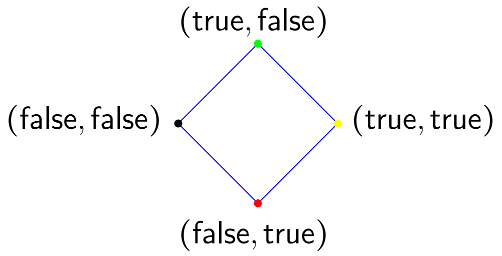


Figure 8: Truth Order

Figure 8 reflects the partial order $\leq_t$ of the four truth-values $\{\mathsf{true}, \mathsf{false}, \top, \bot\}$, based on $\mathsf{false} \leq_t \mathsf{true}$ with $\top$ and $\bot$ in-between and incomparable between themselves. Figure 9 reflects the amount of information that "the computer has been told." Belnap regards this structure as a Scott-type approximation lattice, and calls it the *information order*, here denoted $\leq_i$. The diagram shows that $\bot \leq_i \mathsf{false} \leq_i \top$ and $\bot \leq_i \mathsf{true} \leq_i \top$, with true and false incomparable between themselves wrt $\leq_i$.

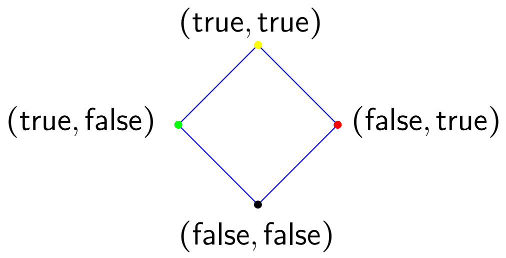


Figure 9: Information Order

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 $\varphi \to \psi$ can now be interpreted as "the computer knows about ψ at least what it knows about φ." More formally, $I \vDash \varphi \to \psi$ if $\varphi^I \leq_i \psi^I$. Note that implicational sentences are only given truth-values true and false.

The next question is what the computer should do if $I \nvDash \varphi \to \psi$. 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 $I \nvDash \varphi \to \psi$, 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 $\varphi \to \psi$ 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 $\varphi \to \psi$ 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 $Chase^{\mathbf{4}}_{\Sigma}(I)$.

φ^I	ψ^I	ψ^J
true	$\bot$	true
true	false	$\top$
$\top$	$\bot$	$\top$
$\top$	true	$\top$
$\top$	false	$\top$
false	$\bot$	false
false	true	$\top$

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 26, and then chase the two-valued decomposed instance $I^\pm$ with the converted sentences.

4.5.1 Tuple-Generating Dependecies

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

$$\forall \bar{x} \left(\left(\exists \bar{y}\, \varphi(\bar{x}, \bar{y}) \right) \rightarrow R(\bar{x}) \right), \tag{8}$$

where $\varphi(\bar{x}, \bar{y})$ is an FO-sentence. Let Σ be a set of tgds and I a two-valued instance. Then $Chase^{\mathbf{2}}_{\Sigma}(I)$ is computed by repeatedly checking, for each tgd of the form (8) 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 true. We can now conveniently compute $Chase^{\mathbf{4}}_{\Sigma}(I)$ by combining $Chase^{\mathbf{2}}_{\Sigma^{I^+}}(I^\pm)$ and $Chase^{\mathbf{2}}_{\Sigma^{I^-}}(I^\pm)$, where $Chase^{\mathbf{2}}_{\Sigma^{I^+}}(I^\pm)$ is computed by repeatedly checking, for each $\bar{a} \in \mathbb{D}$ 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 true. Similarly, $Chase^{\mathbf{2}}_{\Sigma^{I^-}}(I^\pm)$ is computed by checking if $\varphi^{I^-}(\bar{a}, \bar{b}) \nleq_t R^{I^-}(\bar{a})$ and changing $R^{I^-}(\bar{a})$ to true when this is the case. We then have

Theorem 16. [39] *Let Σ be a set of tgds and I a four-valued instance. Then*

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

Proof 19. *Consider a four-valued instance I and a tgd of the form (8). 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 $\varphi(\bar{a}, \bar{b})^{I^-} \leq_t R(\bar{a})^{I^-}$. The following tables show the correspondence of the procedure in detail between instance I and both positive and negative instances I^+, I^-.*

φ^I	ψ^I	ψ^J
true	$\perp$	true
true	false	$\top$
$\top$	$\perp$	$\top$
$\top$	true	$\top$
$\top$	false	$\top$
false	$\perp$	false
false	true	$\top$

$=$

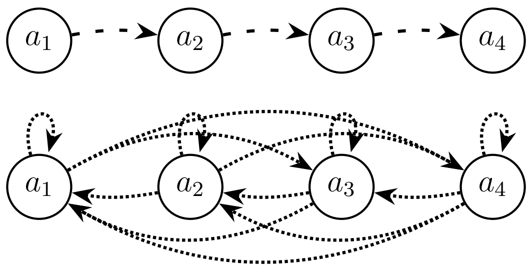
φ^{I^+}	ψ^{I^+}	ψ^{J^+}
true	false	true
true	false	true
true	false	true
true	true	true
true	false	true
false	false	false
false	true	true

$\otimes$

φ^{I^-}	ψ^{I^-}	ψ^{J^-}
false	false	false
false	true	true
true	false	true
true	false	true
true	true	true
true	false	true
true	false	true

In this Section we abandon the technical requirement that $vars(\varphi_i) = x_1, x_2, \ldots$ and use x, y, z for easier reading.

Example 20. *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).

$$E(x,y) \rightarrow T(x,y)$$
$$\exists z\,\big(E(x,z) \wedge T(z,y)\big) \rightarrow T(x,y)$$

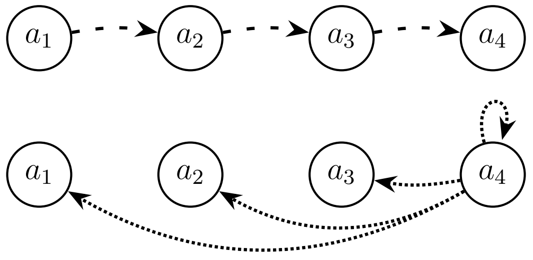
Following Fitting [29, 30] we merge all tgds with the same consequent by taking the *disjunction of their antecedents. The two tgds above will thus result in the dependency*

$$\Big(E(x,y) \lor \big(\exists z\, E(x,z) \land T(z,y)\big)\Big) \to T(x,y)$$

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

$$\begin{aligned}
E^{I^+}(a_1,a_2) \lor \bigvee_{i\in\{1,2,3,4\}} \big(E^{I^+}(a_1,a_i) \land T^{I^+}(a_i,a_2)\big) &\to T^{I^+}(a_1,a_2)\\
E^{I^+}(a_2,a_3) \lor \bigvee_{i\in\{1,2,3,4\}} \big(E^{I^+}(a_2,a_i) \land T^{I^+}(a_i,a_3)\big) &\to T^{I^+}(a_2,a_3)\\
E^{I^+}(a_3,a_4) \lor \bigvee_{i\in\{1,2,3,4\}} \big(E^{I^+}(a_3,a_i) \land T^{I^+}(a_i,a_4)\big) &\to T^{I^+}(a_3,a_4)\\
E^{I^-}(a_4,a_1) \land \bigwedge_{i\in\{1,2,3,4\}} \big(E^{I^-}(a_4,a_i) \lor T^{I^-}(a_i,a_1)\big) &\to T^{I^-}(a_4,a_1)\\
E^{I^-}(a_4,a_2) \land \bigwedge_{i\in\{1,2,3,4\}} \big(E^{I^-}(a_4,a_i) \lor T^{I^-}(a_i,a_2)\big) &\to T^{I^-}(a_4,a_2)\\
E^{I^-}(a_4,a_3) \land \bigwedge_{i\in\{1,2,3,4\}} \big(E^{I^-}(a_4,a_i) \lor T^{I^-}(a_i,a_3)\big) &\to T^{I^-}(a_4,a_3)\\
E^{I^-}(a_4,a_4) \land \bigwedge_{i\in\{1,2,3,4\}} \big(E^{I^-}(a_4,a_i) \lor T^{I^-}(a_i,a_4)\big) &\to T^{I^-}(a_4,a_4)
\end{aligned}$$

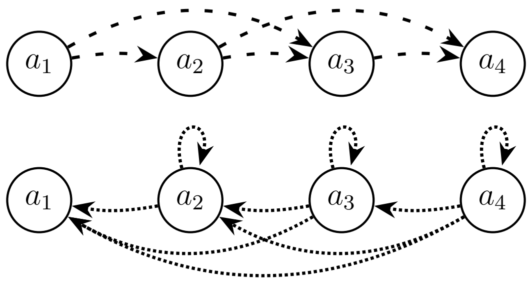
The positive and negative T-edges over $\mathbb{D} = \{a_1,a_2,a_3,a_4\}$ resulting from the first round are shown in the two graphs below.



In the second round we fire

$$\begin{aligned}
E^{I^+}(a_1,a_3) \vee \bigvee_{i\in\{1,2,3,4\}} \big(E^{I^+}(a_1,a_i) \wedge T^{I^+}(a_i,a_3)\big) &\rightarrow T^{I^+}(a_1,a_3)\\
E^{I^+}(a_2,a_4) \vee \bigvee_{i\in\{1,2,3,4\}} \big(E^{I^+}(a_2,a_i) \wedge T^{I^+}(a_i,a_4)\big) &\rightarrow T^{I^+}(a_2,a_4)\\
E^{I^-}(a_3,a_1) \wedge \bigwedge_{i\in\{1,2,3,4\}} \big(E^{I^-}(a_3,a_i) \vee T^{I^-}(a_i,a_1)\big) &\rightarrow T^{I^-}(a_3,a_1)\\
E^{I^-}(a_3,a_2) \wedge \bigwedge_{i\in\{1,2,3,4\}} \big(E^{I^-}(a_3,a_i) \vee T^{I^-}(a_i,a_2)\big) &\rightarrow T^{I^-}(a_3,a_2)\\
E^{I^-}(a_3,a_3) \wedge \bigwedge_{i\in\{1,2,3,4\}} \big(E^{I^-}(a_3,a_i) \vee T^{I^-}(a_i,a_3)\big) &\rightarrow T^{I^-}(a_3,a_3)\\
E^{I^-}(a_2,a_1) \wedge \bigwedge_{i\in\{1,2,3,4\}} \big(E^{I^-}(a_2,a_i) \vee T^{I^-}(a_i,a_1)\big) &\rightarrow T^{I^-}(a_2,a_1)\\
E^{I^-}(a_2,a_2) \wedge \bigwedge_{i\in\{1,2,3,4\}} \big(E^{I^-}(a_2,a_i) \vee T^{I^-}(a_i,a_2)\big) &\rightarrow T^{I^-}(a_2,a_2)
\end{aligned}$$

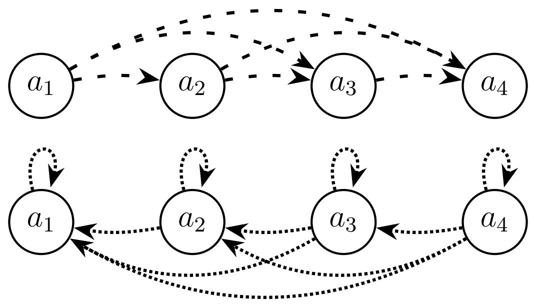
The second round results in the positive and negative graphs below.



Finally, in the third round we fire

$$\begin{aligned}
E^{I^-}(a_1,a_4) \vee \bigvee_{i\in\{1,2,3,4\}} \big(E^{I^+}(a_1,a_i) \wedge T^{I^+}(a_i,a_4)\big) &\rightarrow T^{I^+}(a_1,a_4)\\
E^{I^-}(a_1,a_1) \wedge \bigwedge_{i\in\{1,2,3,4\}} \big(E^{I^-}(a_1,a_i) \vee T^{I^-}(a_i,a_1)\big) &\rightarrow T^{I^-}(a_1,a_1)
\end{aligned}$$

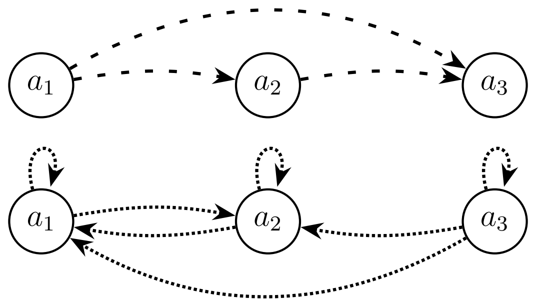
The first graph shows T, the transitive closure of E, and the second the complement of the transitive closure. Note that there is no need for any syntactical notion of stratification or non-monotonic reasoning.



Example 21. *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*

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

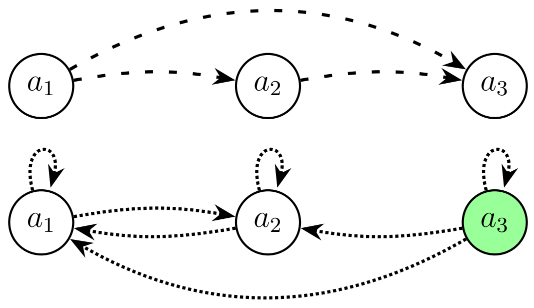
That is, a player can win in state x whenever he can move to state y and y is not a winning state. Below we show the relations $Move^{I^+}$ and $Move^{I^-}$ over $\mathbb{D} = \{a_1, a_2, a_3\}$. Initially we have $Win(a_i) = \bot$ for $i \in \{1,2,3\}$.



In the first round we fire

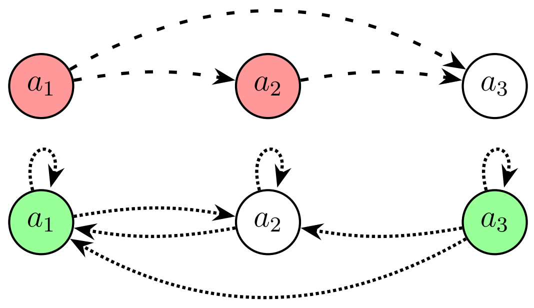
$$\bigwedge_{i \in \{1,2,3,4\}} \left(Move^{I^-}(a_3, a_i) \vee Win^{I^+}(a_i)\right) \rightarrow Win^{I^-}(a_3).$$

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

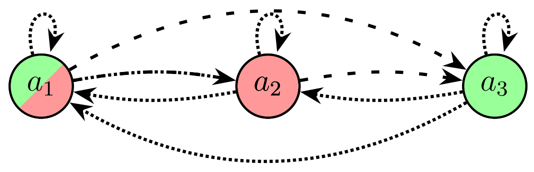


In the second round we fire

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



There are few interesting notes to be made about Example 21. 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. Below the $\otimes$ composition of the positive and the negative graphs is visualized.



4.6 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 $\mathbb{D} = \{a_1, a_2, a_3 \dots, a_n, \dots\}$, 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.

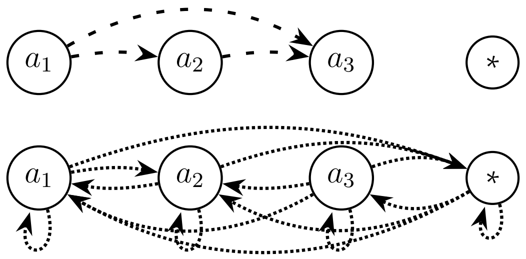
$Move^{I^+}$

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 $Win^I(a_i) = \bot$ for all $a_i \in \mathbb{D}$.



In the first round we fire

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

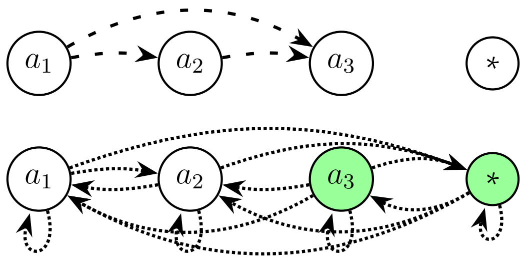
which is equivalent to

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

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

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

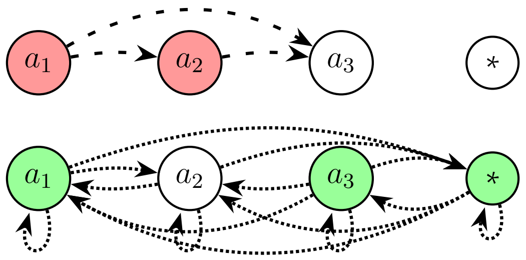
As a result, $Win^{I^-}(*)$ is set to `true` below, as well.



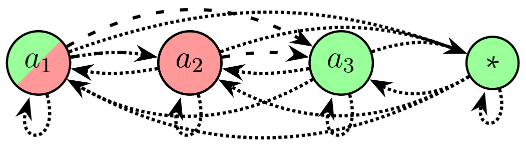
In the second round we fire

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

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 $Win^I(a) = \mathsf{true}$ and $Win^I(a) = \mathsf{false}$, respectively. One should notice that inconsistency in the initial database can be propagated by the chase. However, consistent information is still treated correctly. The $\otimes$-composition of the positive and the negative graphs is illustrated below.



5 Complexity

In this chapter we provide complexity results for Cylindric Star Algebra and Star Cylinders. We start by defining the size of extended star-cylinders. Let $\dot{\mathbf{C}}$ be a sequence of n-dimensional extended star-cylinders and diagonals. By $|\dot{\mathbf{C}}|$ we denote the larger of the number of star-tuples in $\dot{\mathbf{C}}$ and the number of literals in the star-tuple with the largest condition column $n+1$. The same notation also applies to sequences of naive star-cylinders $\ddot{\mathbf{C}}$.

5.1 General Complexity

First, we show that star-cylinders can be transformed into normal form in polynomial time.

Theorem 17. *Let $\dot{C}$ be an n-dimensional extended star-cylinder. Then $\dot{C}$ can in polynomial time be transformed into a normal form star-cylinder $\dot{C}'$ such that $[\![\dot{C}']\!] = [\![\dot{C}]\!]$.*

Proof 20. *The first requirement is that each extended star-tuple has a consistent and logically closed set of literals. To achieve this, we associate with each $\dot{t} \in \dot{C}$ a graph with vertices $\{1, \ldots, n\}$, with a blue edge $\{i, j\}$ if $(i = j) \in \dot{t}(n+1)$, and a red edge $\{i, j\}$ if $(i \neq j) \in \dot{t}(n+1)$. Next, we compute the transitive closure of the graph wrt the blue edges. To account for the inequality conditions, we further extend the graph by repeatedly checking if there is a red edge $\{i, j\}$ and a blue edge $\{j, k\}$, in which case we add, unless already there, a red edge $\{i, k\}$. Then each connected component of* *blue edges represents an equivalence class of dimensions with equal values in $\dot{t}$, unless there is a pair $\{i, j\}$ that has both a blue and a red edge, in which case $\dot{t}(n+1) \vDash \mathsf{false}$, $[\![\{\dot{t}\}]\!] = \varnothing$, and $\dot{t}$ can be removed. We need to consider conditions of the form $(i \neq a)$ in star-tuples $\dot{t}$ as well. They will be handled similarly to the inequality conditions. More precisely, for each $(i \neq a)$ we add a black self-loop labelled a to vertex i. If there is a blue edge $\{i, j\}$, we recursively add an a-labelled self-loop to vertex j. In the end, if there is a vertex i having an a-labelled self-loop while $\dot{t}(i) = b \neq a$, we again have $\dot{t}(n+1) \vDash \mathsf{false}$, $[\![\{\dot{t}\}]\!] = \varnothing$, and therefore remove star-tuple $\dot{t}$. All the above graph-manipulation can clearly be performed in time polynomial in n.*

We still need to verify that $\dot{t}$ satisfies conditions (1) – (3) of Definition 14. If $\dot{t}$ violates a condition, it is easy to see that $[\![\{\dot{t}\}]\!] = \varnothing$, so $\dot{t}$ can be removed from $\dot{C}$. The only exception is for condition (1), when $\dot{t}(n+1) \vDash (i = j)$, $t(i) = *$, and $t(j) = a \in \mathbb{D}$. In this case $\dot{t}$ is retained, but with $t(i)$ replaced by a. If $t(j) = *$ and $t(i) = a$ then $t(j)$ is replaced with a.

The remaining star-tuples form the normalized star-cylinder $\dot{C}'$, and $[\![\dot{C}']\!] = [\![\dot{C}]\!]$ by construction.

Next, we investigate the complexity of evaluating $\mathrm{SCA}_{\text{âĂć}}$-expressions over naive star-cylinders and then we characterize various membership and containment problems. It turns out $\dot{E}(\ddot{\mathbf{C}})$ can be computed efficiently for SCA_n-expressions $\dot{E}$, even though universal quantification and negation are allowed. First we need the following general result.

Theorem 18. *Let $\dot{E}$ be a fixed SCA_n-expression, and $\dot{\mathbf{C}}$ a sequence of n-dimensional extended star-cylinders and diagonals. Then there is a polynomial π, such that $|\dot{E}(\dot{\mathbf{C}})| = \mathcal{O}(\pi(|\dot{\mathbf{C}}|))$. Moreover, $\dot{E}(\dot{\mathbf{C}})$ can be computed in time $\mathcal{O}(\pi(|\dot{\mathbf{C}}|))$, and if negation is not used in $\dot{E}$ this applies to naive star-cylinders $\ddot{\mathbf{C}}$ as well.*

Proof 21. *Since $\dot{E}$ is fixed it is sufficient to prove the first claim for each operator separately. Note that since $\dot{E}$ is fixed, it follows that n is also fixed.*

1. *If $\dot{E}(\dot{\mathbf{C}}) = \dot{\mathsf{C}}_p(\dot{\mathbf{C}})$, then $|\dot{E}(\dot{\mathbf{C}})| = |\dot{C}_p| \leq |\dot{\mathbf{C}}| = \mathcal{O}(\pi(|\dot{\mathbf{C}}|))$.*

2. *If $\dot{E}(\dot{\mathbf{C}}) = \dot{\mathsf{d}}_{ij}(\dot{\mathbf{C}})$, then $|\dot{E}(\dot{\mathbf{C}})| = \mathcal{O}(|\dot{\mathbf{C}}|) \times \mathcal{O}(1) = \mathcal{O}(\pi(|\dot{\mathbf{C}}|))$.*

3. *If $\dot{E}(\dot{\mathbf{C}}) = \dot{\mathsf{C}}_p(\dot{\mathbf{C}}) \cup \dot{\mathsf{C}}_q(\dot{\mathbf{C}})$, then $|\dot{E}(\dot{\mathbf{C}})| \leq |\dot{\mathbf{C}}| = \mathcal{O}(\pi(|\dot{\mathbf{C}}|))$.*

4. *If $\dot{E}(\dot{\mathbf{C}}) = \dot{\mathsf{C}}_p(\dot{\mathbf{C}}) \dot{\cap} \dot{\mathsf{C}}_q(\dot{\mathbf{C}})$, then the number of tuples in $\dot{E}(\dot{\mathbf{C}})$ is at most $|\dot{\mathbf{C}}|^2$, and each tuple in the output can have a condition of length at most $2 \cdot |\dot{\mathbf{C}}|$. As a result, $|\dot{E}(\dot{\mathbf{C}})| \leq 2 \cdot |\dot{\mathbf{C}}|^3 = \mathcal{O}(\pi(|\dot{\mathbf{C}}|))$.*

5. *If $\dot{E}(\dot{\mathbf{C}}) = \dot{\mathsf{c}}_i(\mathsf{C}_p(\dot{\mathbf{C}}))$, then $|\dot{E}(\dot{\mathbf{C}})| \leq |\dot{C}_p| \leq |\dot{\mathbf{C}}| = \mathcal{O}(\pi(|\dot{\mathbf{C}}|))$.*

6. *For the case $\dot{E}(\dot{\mathbf{C}}) = \dot{\mathsf{ɔ}}_i(\mathsf{C}_p(\dot{\mathbf{C}}))$ we note that $\dot{\mathsf{ɔ}}_i(\mathsf{C}_p(\dot{\mathbf{C}})) \subseteq (\mathsf{C}_p(\dot{\mathbf{C}}) \dot{\cap} \dot{A}) \subseteq \dot{A}$. We can construct the star-tuples in $\dot{A}$ by iterating over the star-tuples in $\dot{C}_p$ and using the constants in A. This means that $|\dot{A}| = (n \times (|A|)) + (2^n + |A|) \leq \mathcal{O}(1) \times \mathcal{O}(|\dot{\mathbf{C}}|) + \mathcal{O}(1) \times \mathcal{O}(|\dot{\mathbf{C}}|) = \mathcal{O}(\pi(|\dot{\mathbf{C}}|))$. Note that n is the dimensionality of $\dot{\mathbf{C}}$ and is a constant.*

7. *If $\dot{E}(\dot{\mathbf{C}}) = \dot{\neg}(\mathsf{C}_p(\dot{\mathbf{C}}))$, then similar to the inner cylindrification we have $\dot{\neg}\,\dot{C}_p \subseteq \dot{A}$ which implies $|\dot{E}(\dot{\mathbf{C}})| = \mathcal{O}(\pi(|\dot{\mathbf{C}}|))$.*

5.2 Membership Problem

In the membership problem, we ask if an ordinary tuple t belongs to the set specified by a (naive) star-cylinder, of by a fixed expression $\dot{E}$ and a (naive) star-cylinder. In other words, all results refer to data complexity.

Theorem 19. *Let $t \in \mathbb{D}^n$ and $\ddot{\mathbf{C}}$ a sequence of n-dimensional naive star-cylinders and diagonals. The membership problems and their respective data complexities are as follows.*

1. $t \overset{?}{\in} \bigcap E(Rep([\![\ddot{\mathbf{C}}]\!]))$ *is in polytime for positive* E.

2. $t \overset{?}{\in} \bigcap E(Rep([\![\ddot{\mathbf{C}}]\!]))$ *is coNP-complete for* E *where negation is allowed in equality atoms only.*

Proof 22. *1. By Theorem 10, we have* $\bigcap E(Rep([\![\ddot{\mathbf{C}}]\!])) = [\![\dot{E}(\ddot{\mathbf{C}})_{\downarrow}]\!]$, *so to test if* $t \in \bigcap E(Rep([\![\ddot{\mathbf{C}}]\!]))$, *we compute* $\dot{E}(\ddot{\mathbf{C}})_{\downarrow}$, *and see if there is a star-tuple* $\dot{t} \in \dot{E}(\ddot{\mathbf{C}})_{\downarrow}$, *such that* $t \leq \dot{t}$. *By Theorem 18,* $\dot{E}(\ddot{\mathbf{C}})_{\downarrow}$ *can be computed in polytime.*

2. To check if $t \notin \bigcap E(Rep([\![\ddot{\mathbf{C}}]\!]))$, *it is sufficient to find a homomorphism* h *such that* $t \notin h([\![\ddot{\mathbf{C}}]\!])$. *We guess the homomorphism* h, *and check in polytime if* $t \notin h([\![\ddot{\mathbf{C}}]\!])$. *Thus* $t \notin \bigcap E(Rep([\![\ddot{\mathbf{C}}]\!]))$ *is in NP, and* $t \in \bigcap E(Rep([\![\ddot{\mathbf{C}}]\!]))$ *is in coNP. The lower bound follows from Theorem 5.2.2 in [2], where the complexity of conditional tables are investigated in detail.*

5.3 Containment Problem

The containment problem asks for containment of star-cylinders (naive star-cylinders), or views over star-cylinders (naive star-cylinders). Regarding the sequences of n-dimensional (naive) star-cylinders and CA$_n$ expressions, we have the following containment problem.

Theorem 20. *Let* $\dot{\mathbf{C}}$ *and* $\dot{\mathbf{D}}$ *(resp.* $\ddot{\mathbf{C}}$ *and* $\ddot{\mathbf{D}}$*) be sequences of* n*-dimensional (naive) star-cylinders and diagonals. Then*

1. $E_1([\![\dot{\mathbf{C}}]\!]) \overset{?}{\subseteq} E_2([\![\dot{\mathbf{D}}]\!])$ *is in polytime for CA*$_n$ *expression* E_1 *and* E_2.

2. $Rep([\![\ddot{\mathbf{C}}]\!]) \overset{?}{\subseteq} Rep([\![\ddot{\mathbf{D}}]\!])$ *is NP-complete.*

3. $E_1(Rep([\![\ddot{\mathbf{C}}]\!])) \overset{?}{\subseteq} E_2(Rep([\![\ddot{\mathbf{D}}]\!]))$ *is* Π_2^p*-complete for positive* E_1 *and* E_2.

Proof 23.

1. *By Lemma 3, we have* $[\![\dot{E}_1(\dot{\mathbf{C}})]\!] \subseteq [\![\dot{E}_2(\dot{D})]\!]$ *if and only if* $\dot{E}_1(\dot{\mathbf{C}}) \Cap \dot{A} \preceq \dot{E}_2(\dot{D}) \Cap \dot{A}$. *The latter dominance is true if and only if for each star-tuple* $\dot{t} \in \dot{E}_1(\dot{\mathbf{C}}) \Cap \dot{A}$ *there is a star-tuple* $\dot{u} \in \dot{E}_2(\dot{\mathbf{D}}) \Cap \dot{A}$, *such that* $\dot{t} \preceq \dot{u}$. *From Theorem 18 we know that* $\dot{E}_1(\dot{\mathbf{C}}) \Cap \dot{A}$ *and* $\dot{E}_2(\dot{\mathbf{D}}) \Cap \dot{A}$ *can be computed in polytime.*

2. *We first extend the domain of possible world homomorphisms by stipulating that they are the identity on* $*$. *Then it is easy to see that* $Rep([\![\ddot{\mathbf{C}}]\!]) \subseteq Rep([\![\ddot{\mathbf{D}}]\!])$ *if and only if there exists a possible world homomorphism* h *such that* $\ddot{\mathbf{D}} \rightarrow_h \ddot{\mathbf{C}}$. *This makes the problem NP-complete.*

3. *The lower bound follows from Theorem 4.2.2 in [2], For the upper bound we observe that* $E_1(Rep([\![\ddot{\mathbf{C}}]\!])) \subseteq E_2(Rep([\![\ddot{\mathbf{D}}]\!])$ *iff for every* $\mathbf{C} \in Rep([\![\ddot{\mathbf{C}}]\!])$ *there exists a* $\mathbf{D} \in Rep([\![\ddot{\mathbf{D}}]\!])$ *such that* $E_1(\mathbf{C}) = E_2(\mathbf{D})$ *iff for every possible world homomorphism* h *on* $\ddot{\mathbf{C}}$ *there exists a possible world homomorphism* g *on* $\ddot{\mathbf{D}}$ *such that* $E_1(h([\![\ddot{\mathbf{C}}]\!])) = E_2(g([\![\ddot{\mathbf{D}}]\!]))$. *By Corollary 1, this equality holds iff* $[\![\dot{E}_1(h(\ddot{\mathbf{C}}))]\!] = [\![\dot{E}_1(g(\ddot{\mathbf{D}}))]\!]$. *By Lemma 3, the last equality holds iff* $E_1(h([\![\ddot{\mathbf{C}}]\!])) \Cap \dot{A} \preceq E_2(g([\![\ddot{\mathbf{D}}]\!])) \Cap \dot{A}$, *and vice-versa. By Theorem 18, the star-cylinders in the two dominances* $\preceq$ *can be computed in polynomial time.*

6 Related and Future Work

6.1 Related Work

Cylindric Set Algebra gave rise to a whole subfield of Algebra, called Cylindric Algebra. For a fairly recent overview, the reader is referred to [5]. Within database theory, a simplified version of the star-cylinders and a corresponding Codd-style positive relational algebra with evaluation rules "$\ast = \ast$" and "$\ast = a$" was sketched by Imielinski and Lipski in [50]. Such cylinders correspond to the structures in *diagonal-free* Cylindric Set Algebras [43, 44]. The exact FO-expressive power of these diagonal-free star-cylinders is an open question. Nevertheless, using the techniques of this dissertation, it can be shown that naive existential nulls can be seamlessly incorporated in diagonal-free star-cylinders.

In addition to the above and the work described in Section 1.6, Imielinski and Lipski also showed in [50] that the fact that Codd's Relational Algebra does not have a finite axiomatization, and the fact that equivalence of expressions in it is undecidable, follow from known results in Cylindric Algebra. This is of course true for a host of general results in Mathematical Logic.

Yannakakis and Papadimitriou [63] formulated an algebraic version of dependency theory using Codd's Relational Algebra. Around the same time Cosmadakis [16] proposed an interpretation of dependency theory in terms of equations over certain types of expressions in Cylindric Set Algebra, and described a complete finite axiomatization of his system. It was however later shown by Düntsch, Hodges, and Mikulas [48, 20], again using known results from Cylindric Algebra, that Cosmadakis's axiomatization was incomplete, and that no finite complete axiomatization exists.

Interestingly, it turns out that one of the models for constraint databases in [52] by Kanellakis, Kuper, and Revesz — the one where the constraints are equalities over an infinite domain — is equivalent with our star-tables. Even though [52] develops a bottom-up (recursive) evaluation mechanism for FO-queries, the mechanism is goal-oriented and contrary to our star-cylinders, there is no algebra operating on the constraint databases. We note however that the construction of the sieve $\dot{A}$ in Section 3 is inspired by the constraint solving techniques of [52]. It therefore seems that our star-cylinders and algebra can be made to handle inequality constraints on dense linear orders as well as polynomial constraints over real-numbers, as is done in [52]. We also note that our work is related to the orbit finite sets, treated in a general computational framework in [14].

As noted in Section 1.6, the existential nulls have long been well understood. According to [19] the fact that positive queries (no negation, but allowing universal quantification) are preserved under onto-homomorphisms are folklore in the database community. Using this monotonicity property, Libkin [33] has recently shown that positive queries can be evaluated naively on finite existential databases I under a so called *weak closed world assumption*, where $Rep(I)$ consists of all complete instances J, such that $h(I) \subseteq J$ and J only involves constants that occur in I, and furthermore h is onto from the finite universe of I to the finite universe of J. Our Theorem 9 generalizes Libkin's result to infinite databases. In this context it is worth noting that Lyndon's Positivity Theorem [59] tells us that a first order formula is preserved under onto-homomorphisms on all structures if and only if it is equivalent to a positive formula. It has subsequently been shown that the only-if direction fails for finite structures [4, 61]. Since our star-cylinders represent neither finite nor unrestricted infinite structures, it would be interesting to know whether the only-if direction holds for infinite structures represented by star-cylinders. If it does, it would mean that our Theorem 9 would be optimal, meaning that if φ is not equivalent to a positive formula, then naive evaluation does not work for φ on databases represented by naive star-cylinders.

Furthermore, we note that Sundarmurthy et al. [62] have generalized the conditional tables of [49, 35] by replacing the labeled nulls with a single null $\mathbf{m}$ that initially represents all possible domain values. They then add constraints on the occurrences of these $\mathbf{m}$-values, allowing them to represent a finite or infinite subset of the domain, and to equate distinct occurrences of $\mathbf{m}$. Sundarmurthy et al. then show that their $\mathbf{m}$-tables are closed under positive (but not allowing universal quantification) queries by developing a difference-free Codd-style relational algebra that $\mathbf{m}$-tables are closed under. Merging our approach with theirs could open up interesting possibilities.

6.2 Future Work

In this research dissertation we addressed the Intuitionistic Databases, along with query evaluation and data exchange problem. This problem has been studied widely but there are still many issues to be solved. As one the most fundamental problems, the interpretation of negation and quring the databases in presence of negation has been revisited. We proposed a four valued logical semantic as a clear semantic for intuitionist logical view of databases. There are still further works to be done such as:

- *Adding different forms of negation:* A negation can be expressed in different forms in intuitionistic logic. Also, in four valued logic we have the negation of information, which is called conflation. Conflation and negation can be combined. Therefore, it can add to expressive power of the language. These different forms of negation can be interpreted in our approach, meaning that the decomposition technique and algebraic evaluation can be applied to queries using $\leq_i$-based operators. An application of this for data exchange with GAV-tgd's can be found in [41]. So, it would be interesting to consider the result of queries after applying these form of negations.

- *Extension to other problems:* There are problems in theory of databases, such as data repair and data recovery. These problems have a close connection with the problem of data exchange. Therefore, we propose to extend the intuitionistic semantics to these problems and hopefully there will be more accurate results in comparison to currently existing approaches.

- *Expressive power of star-cylinders:* In this research we showed that FO, CA and SCA have the same expressive power. However, the set of cylinders (Database Instances) that can be expressed using star-cylinders has not been shown to fit in any category. Our preliminary results characterize the $*$-representable infinite instances in terms of orbit-finite sets. Showing what are the exact class of instances and cylinders which can be losslessly expressed by star-cylinders is still open.

- *Chase Termination Problem:* We already showed that tuple generating dependencies in presence of negation with incomplete and inconsistent instances can be evaluated with our intuitionistic logic based approach. It would be interesting to know which class of dependencies are the upper bound for our approach and how the chase termination conditions will be influenced by universal nulls.

Bibliography

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] Serge Abiteboul, Paris C. Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. *Theor. Comput. Sci.*, 78(1):158–187, 1991.

[3] Foto N. Afrati and Phokion G. Kolaitis. Repair checking in inconsistent databases: algorithms and complexity. In *ICDT*, pages 31–41, 2009.

[4] Miklos Ajtai and Yuri Gurevich. Monotone versus positive. *J. ACM*, 34(4):1004–1015, October 1987.

[5] Hajnal Andréka, Miklós Ferenczi, and István Németi. *Cylindric-like Algebras and Algebraic Logic*, volume 22. Springer Science & Business Media, 2014.

[6] Marcelo Arenas, Pablo Barceló, Ronald Fagin, and Leonid Libkin. Locally consistent transformations and query answering in data exchange. In *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 14-16, 2004, Paris, France*, pages 229–240, 2004.

[7] Marcelo Arenas, Leopoldo Bertossi, and Michael Kifer. Applications of annotated predicate calculus to querying inconsistent databases. In *International Conference on Computational Logic*, pages 926–941. Springer, 2000.

[8] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pages 68–79, 1999.

[9] Marcelo Arenas, Ronald Fagin, and Alan Nash. Composition with target constraints. In *ICDT*, pages 129–142, 2010.

[10] Marcelo Arenas, Jorge Pérez, and Cristian Riveros. The recovery of a schema mapping: Bringing exchanged data back. *ACM Trans. Database Syst.*, 34(4), 2009.

[11] 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.

[12] Joachim Biskup. A foundation of codd's relational maybe-operations. *ACM Trans. Database Syst.*, 8(4):608–636, 1983.

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

[14] Mikołaj Bojańczyk. Orbit-finite sets and their algorithms (invited talk). In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 1:1–1:14, 2017.

[15] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[16] Stavros S. Cosmadakis. Database theory and cylindric lattices (extended abstract). In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 411–420, 1987.

[17] C. J. Date. *Database in depth - relational theory for practitioners.* O'Reilly, 2005.

[18] Jan Van den Bussche. Applications of Alfred Tarski's ideas in database theory. In Laurent Fribourg, editor, *Computer Science Logic, 15th International Workshop, CSL 2001. 10th Annual Conference of the EACSL, Paris, France, September 10-13, 2001, Proceedings*, volume 2142 of *Lecture Notes in Computer Science*, pages 20–37. Springer, 2001.

[19] Alin Deutsch, Alan Nash, and Jeffrey B. Remmel. The chase revisited. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, pages 149–158, 2008.

[20] Ivo Düntsch and Szabolcs Mikulás. Cylindric structures and dependencies in relational databases. *Theor. Comput. Sci.*, 269(1-2):451–468, 2001.

[21] Ronald Fagin. Inverting schema mappings. *ACM Trans. Database Syst.*, 32(4), 2007.

[22] Ronald Fagin, Benny Kimelfeld, and Phokion G. Kolaitis. Probabilistic data exchange. In *ICDT*, pages 76–88, 2010.

[23] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *ICDT*, pages 207–224, 2003.

[24] Ronald Fagin, Phokion G. Kolaitis, Alan Nash, and Lucian Popa. Towards a theory of schema-mapping optimization. In *PODS*, pages 33–42, 2008.

[25] Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. Data exchange: getting to the core. In *PODS*, pages 90–101, 2003.

[26] Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. Composing schema mappings: Second-order dependencies to the rescue. In *PODS*, pages 83–94, 2004.

[27] Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. Reverse data exchange: coping with nulls. In *PODS*, pages 23–32, 2009.

[28] Melvin Fitting. Bilattices and the semantics of logic programming. *J. Log. Program.*, 11(1&2):91–116, 1991.

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

[30] Melvin Fitting. The family of stable models. *J. Log. Program.*, 17(2/3&4):197–225, 1993.

[31] Ariel Fuxman and Renée J Miller. First-order query rewriting for inconsistent databases. *Journal of Computer and System Sciences*, 73(4):610–635, 2007.

[32] Amélie Gheerbrant, Leonid Libkin, and Cristina Sirangelo. When is naive evaluation possible? In *PODS*, pages 75–86, 2013.

[33] 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.

[34] Matthew L. Ginsberg. Bilattices and modal operators. In Rohit Parikh, editor, *TARK*, pages 273–287. Morgan Kaufmann, 1990.

[35] Gösta Grahne. *The Problem of Incomplete Information in Relational Databases*, volume 554 of *Lecture Notes in Computer Science*. Springer, 1991.

[36] Gösta Grahne and Ali Moallemi. Universal (and existential) nulls. *CoRR, Submitted to Journal*, abs/1803.01445, 2018.

[37] Gösta Grahne and Ali Moallemi. Universal nulls (extended abstract). In *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, Cali, Colombia, May 21-25, 2018.*, 2018.

[38] Gösta Grahne and Ali Moallemi. A useful four-valued database logic. In Bipin C. Desai, Sergio Flesca, Ester Zumpano, Elio Masciari, and Luciano Caroprese, editors, *Proceedings of the 22nd International Database Engineering & Applications Symposium, IDEAS 2018, Villa San Giovanni, Italy, June 18-20, 2018*, pages 22–30. ACM, 2018.

[39] Gösta Grahne, Ali Moallemi, and Adrian Onet. Intuitionistic data exchange. In Andrea Calì and Maria-Esther Vidal, editors, *Proceedings of the 9th Alberto Mendelzon International Workshop on Foundations of Data Management, Lima, Peru, May 6 - 8, 2015.*, volume 1378 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[40] Gösta Grahne, Ali Moallemi, and Adrian Onet. Recovering exchanged data. In Tova Milo and Diego Calvanese, editors, *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 105–116. ACM, 2015.

[41] Gösta Grahne, Nicolas Spyratos, and Daniel Stamate. Semantics and containment with internal and external conjunctions. In *Database Theory - ICDT '97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings*, pages 71–82, 1997.

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

[43] 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.

[44] 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.

[45] André Hernich. Answering non-monotonic queries in relational data exchange. *Logical Methods in Computer Science*, 7(3), 2011.

[46] André Hernich. Computing universal models under guarded tgds. In *ICDT*, pages 222–235, 2012.

[47] André Hernich, Leonid Libkin, and Nicole Schweikardt. Closed world data exchange. *ACM Trans. Database Syst.*, 36(2):14, 2011.

[48] Ian Hodkinson and Szabolcs Mikulás. Axiomatizability of reducts of algebras of relations. *Algebra Universalis*, 43(2):127–156, 2000.

[49] Tomasz Imielinski and Witold Lipski. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

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

[51] Thomas Jech. Boolean-valued models. *Handbook of Boolean algebras (Edited by JD Monk), North-Holland*, 3:1197–1211, 1989.

[52] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26 – 52, 1995.

[53] Michael Kifer and Eliezer L Lozinskii. A logic for reasoning with inconsistency. *Journal of Automated reasoning*, 9(2):179–215, 1992.

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

[55] Phokion G. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS*, pages 61–75, 2005.

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

[57] Leonid Libkin. Negative knowledge for certain query answers. In Magdalena Ortiz and Stefan Schlobach, editors, *Web Reasoning and Rule Systems - 10th International Conference, RR 2016, Aberdeen, UK, September 9-11, 2016, Proceedings*, volume 9898 of *Lecture Notes in Computer Science*, pages 111–127. Springer, 2016.

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

[59] Roger C. Lyndon. Properties preserved under homomorphism. *Pacific J. Math.*, 9(1):143–154, 1959.

[60] Adrian Onet. *The chase procedure and its applications.* PhD thesis, Concordia University, 2012.

[61] Alexei P. Stolboushkin. Finitely monotone properties. In *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science*, LICS '95, pages 324–, Washington, DC, USA, 1995. IEEE Computer Society.

[62] Bruhathi Sundarmurthy, Paraschos Koutris, Willis Lang, Jeffrey F. Naughton, and Val Tannen. m-tables: Representing missing data. In *20th International Conference on Database Theory, ICDT 2017, March 21-24, 2017, Venice, Italy*, pages 21:1–21:20, 2017.

[63] Mihalis Yannakakis and Christos H. Papadimitriou. Algebraic dependencies. *J. Comput. Syst. Sci.*, 25(1):2–41, 1982.