Cylindric structures and dependencies in relational databases

Version 2a

Ivo Düntsch
School of Information and Software Engineering
University of Ulster at Jordanstown
Newtownabbey, BT 37 0QB, N.Ireland
I.Duentsch@ulst.ac.uk

Szabolcs Mikulás
Department of Computer Science
King's College London
Strand, London, WC2R 2LS, England
szabolcs@dcs.kcl.ac.uk

Abstract

In this paper, we explore the precise connection between dependencies in relational databases and variants of cylindric algebras, and apply recent algebraic results to problems of axiomatizing dependencies. We will consider project-join dependencies and the corresponding class of cylindric semilattices. We will also look at Cosmadakis (1987) who introduces cylindric dependencies, and makes several claims regarding the structural properties of these dependencies. However, recent algebraic investigations provide counterexamples to the main theorems.

1 Introduction

The relational data model as introduced in Codd (1970) has achieved wide success because of its clarity and succinctness. What was lacking in the approach — as pointed out in Fagin & Vardi (1986) — were the semantics which could describe constraints among and within the base relations. It turned out that most dependencies could be formulated in fragments of first order logic.

In machine learning, data dependencies play a major role as a means of feature reduction and they are crucial for rule based methods of soft computing, for example in rough set data analysis (Düntsch & Gediga, 1997, 1998, Novotný, 1997a,b).

Due to the formulation of the data model with relational operators, there has always been an algebraic trend in the study of data dependencies in the sense that one looked for axioms for operators on relations — in other words, for suitable algebras (Imielinski & Lipski, 1984) — which would have the same expressive power as the first order sentences. Yannakakis & Papadimitriou (1982) introduced algebraic dependencies which generalized all hitherto known dependencies, and which were, in fact, equivalent to the embedded implicational dependencies (Fagin, 1982). Subsequently, Cosmadakis (1987) generalized the algebraic dependencies to include a union operator, and defined cylindric dependencies. As we shall see below, it turns out, however, that the situation is more complicated than described there: The main claims are incorrect, and, in a sense, cannot be repaired.

Algebraic reasoning as manipulation of relations has a long–standing tradition, going back to the latter half of the last century, e.g. by de Morgan (1864), Peirce (1870) and Schröder (1890 - 1905). From the 1940's onwards, A. Tarski (1941) and his colleagues have continued the work on the calculus of relations which eventually led to an algebraization of first order logic via cylindric algebras (Henkin et al., 1971, 1985), and its finite fragments, in particular, first order logic with three variables via relation algebras (cf. Tarski & Givant, 1987). As an introduction to logic on the basis of relations we invite the reader to consult Andréka et al. (1998), and an overview of relations and their algebras can be found in Németi (1991).

The aim of this paper is to shed some light on the algebraic and logical fine structure of these dependencies using recent results from algebraic logic (Hodkinson & Mikulás, 1998).

The outline of the paper is as follows: To make the paper more self contained, we shall first give a brief outline of the connection between first order logic, database relations, and cylindric algebras. This will serve as the basis for subsequent discussions. We shall then look at project-join dependencies, algebraic dependencies (Yannakakis & Papadimitriou, 1982) and cylindric dependencies (Cosmadakis, 1987).

Even though the original investigations date back twenty years, it is only now that new results in algebraic logic shed more light on some of the fine structure of the dependencies under consideration.

2 Databases and first order logic

Let U be a fixed set[1], which we call the set of *attributes*. For every $i \in U$, let D_i be a non-empty set. Assume that X is a finite subset of U. We call f an *X-tuple* if $f \in \prod_{i \in X} D_i$. An *$X$-relation* R is a set of X-tuples. If R is an X-relation, we set $a(R) = X$, and call $a(R)$ the *scheme of* R. By a *database* we mean a structure $\mathcal{D} = \langle U, D_i, R_j \rangle_{i \in U, j \in J}$ such that, for every $j \in J$, R_j is an $a(R_j)$-relation. We say that R *is a data table over* $\mathcal{D}$ if R is a relation in $\mathcal{D}$.

In this paper, we will assume that $U = \alpha = \{\kappa : \kappa < \alpha\}$ for some countable cardinal α (i.e. α is either ω, the first infinite ordinal, or a natural number), called the *dimension* of the database. Thus, we usually will not denote U, and we simply write $\mathcal{D} = \langle D_i, R_j \rangle_{i<\alpha, j \in J}$.

We consider positive fragments of first order logic. Let α be a natural number or ω; a language $\mathcal{L}_\alpha^C$ consists of

1. a set V of individuum variables $\{v_i : i < \alpha\}$,
2. a set $\{P_i : i \in I\}$ of predicate symbols; each P_i has a finite arity $\gamma(P_i)$,
3. a set $C \subseteq \{\wedge, \vee, \exists, \mathbf{T}, \mathbf{F}, =\}$ of logical constants and operators.

If C is understood or not relevant in the context, we shall usually just write $\mathcal{L}_\alpha$.

The set Fml of $\mathcal{L}_\alpha$-formulas is defined recursively in the usual way (Chang & Keisler, 1971).

[1]It is usual to assume that U is finite. In this paper, however, we do not restrict our investigations to the finite case.

A model of $\mathcal{L}_\alpha$ is a structure $\mathcal{D} = \langle D, \mathcal{R} \rangle$, where D is a non-empty set, called the *domain of* $\mathcal{D}$, and $\mathcal{R}$ is a set of (finitary) relations over D corresponding to the predicate symbols of $\mathcal{L}_\alpha$. A *valuation for* $\mathcal{D}$ is a mapping $f : V \to D$; we denote the set of all valuations for $\mathcal{D}$ by ${}^\alpha D$. Satisfaction $\models_f$ of a formula φ with respect to a valuation f is defined recursively in the usual way.

For an $\mathcal{L}_\alpha$-formula φ, $\mathrm{var}(\varphi)$ is the set of all variables occurring in φ, and $\mathrm{frv}(\varphi)$ denotes the set of free variables occurring in φ. If $\mathrm{frv}(\varphi) = \{x_0, \dots, x_k\}$, we indicate this by writing $\varphi(x_0, \dots, x_k)$.

We can think of the relations $R \in \mathcal{R}$ as data tables, and of $\mathcal{D}$ as a database. In this interpretation, each formula φ of $\mathcal{L}_\alpha$ defines a query on $\mathcal{D}$. The result $\mathrm{def}_\mathcal{D}(\varphi)$ of this query is just the relation defined by φ in $\mathcal{D}$. In terms of evaluation mappings,

$$\mathrm{def}_\mathcal{D}(\varphi) = \{f \in {}^\alpha D : \mathcal{D} \models_f \varphi\}.$$

If $\mathcal{D}$ is understood, we usually omit the subscript.

Each database $\mathcal{D} = \langle D_i, R_j \rangle_{i<\alpha, j \in J}$ gives rise to a sorted (typed) version ${}^s\mathcal{L}_\alpha$ of $\mathcal{L}_\alpha$ as follows:

1. The formulas are that of $\mathcal{L}_\alpha$.
2. Models are of the form $\langle D, \mathcal{R} \rangle$, where $D = \bigcup_{i<\alpha} D_i$, each $R \in \mathcal{R}$ is a relation on $\prod\{D_i : v_i \in \mathrm{frv}(R)\}$.
3. Valuations $f : V \to D$ are functions such that $f(v_i) \in D_i$ for every $i < \alpha$.

Suppose that $\mathcal{D} = \langle D, \mathcal{R} \rangle$ is a model of ${}^s\mathcal{L}_\alpha$. Define the *truth relation* $\mathrm{def}(\varphi)$ of a formula φ over $\mathcal{D}$ as

$$\mathrm{def}(\varphi) = \{f \restriction \mathrm{frv}(\varphi) : \mathcal{D} \models_f \varphi\}. \tag{2.1}$$

Note that each $f \in \mathrm{def}(\varphi)$ can be considered an evaluation of the free variables of φ. If φ is a sentence, then we define

$$\mathrm{def}(\varphi) = \begin{cases} {}^\alpha D, & \text{if } \mathcal{D} \models \varphi \\ \emptyset, & \text{otherwise.} \end{cases} \tag{2.2}$$

3 Project-join expressions

Basic operations of the relational data model are *projection* and *join*; project-join expressions lead to the simplest form of relational dependencies.

In this section, we assume that $\alpha = n$ is finite. If $f \in \prod_{i<n} D_i$ and $Y \subseteq n$, we denote the restriction of f to Y by $f \restriction Y$. That is, $f \restriction Y = g$ iff $g \in \prod_{i \in Y} D_i$ and $g_i = f_i$ for every $i \in Y$.

Let $\mathcal{D} = \langle D_i, R_j \rangle_{i<n, j \in J}$ be a database, R and S be data tables and $Y \subseteq a(R)$. The *projection of* R *to* Y, written as $\pi_Y R$, is defined as

$$\pi_Y R = \begin{cases} \{f \in \prod_{i \in Y} D_i : (\exists g \in R)(g \restriction Y = f)\}, & \text{if } Y \neq \emptyset \\ {}^U D, & \text{otherwise.} \end{cases}$$

Then $\pi_Y R$ is a data table with scheme $a(\pi_Y R) = Y$. The *join* of R and S is defined as

$$R \bowtie S = \{f \in \prod\{D_i : i \in a(R) \cup a(S)\} : f \upharpoonright a(R) \in R \text{ and } f \upharpoonright a(S) \in S\}.$$

Thus $R \bowtie S$ is a data table with scheme $a(R \bowtie S) = a(R) \cup a(S)$.

Next we define project-join expressions using a set of generators and symbols for project and join operators; for simplicity we use the same symbols as above. Let us assume that a set $\{P_j : j \in J\}$ is given and that, for each $j \in J$, P_j has the same scheme n. The set of *project-join expressions* (over the generators $\{P_j : j \in J\}$), pje's for short, is defined as follows:

1. each P_j is a pje with scheme n;
2. if τ is a pje and $Y \subseteq a(\tau)$, then $\pi_Y \tau$ is a pje with scheme $a(\pi_Y \tau) = Y$;
3. if τ and σ are pje's, then $\tau \bowtie \sigma$ is a pje with scheme $a(\tau \bowtie \sigma) = a(\tau) \cup a(\sigma)$;
4. no other expression is a pje.

Given a database $\mathcal{D} = \langle D_i, R_j \rangle_{i<n, j \in J}$ such that, for every $j \in J$, $a(R_j) = n$, we evaluate pje's in the obvious way:

- $e(P_j) = R_j$ for every $j \in J$
- $e(\pi_Y \tau) = \pi_Y e(\tau)$
- $e(\tau \bowtie \sigma) = e(\tau) \bowtie e(\sigma)$

for all pje's τ, σ.

By a *project-join dependency*, a pjd for short, we mean a formula $\tau = \sigma$ with τ and σ pje's. A database $\mathcal{D}$ *satisfies* the pjd $\tau = \sigma$, in symbols $\mathcal{D} \models \tau = \sigma$, iff $e(\tau) = e(\sigma)$. A set Σ of pjd's implies a pjd τ iff, for every database $\mathcal{D}$, $\mathcal{D} \models \Sigma$ implies $\mathcal{D} \models \tau$. We note that we can express that a pje σ follows from a pje τ ($\tau \leq \sigma$), since $e(\tau) \subseteq e(\sigma)$ iff $e(\tau \bowtie \sigma) = e(\sigma)$.

Next we define a first order language corresponding to project-join expressions. We let $\mathcal{L}_n^*$ be the n-variable restricted fragment of ${}^s\mathcal{L}_\omega^{\{\wedge,\exists\}}$, where restricted means that every atomic formula has the form $P(v_0, \ldots, v_{n-1})$. The language $\mathcal{L}_n^*$ is rather weak; it does not contain, for instance, equality, and thus, substitution of variables is not possible; furthermore, only n variables can be used (always in the same order), one for each sort.

Our first result shows that $\mathcal{L}_n^*$ is a proper language for project-join expressions:

Proposition 3.1. *Suppose that $\mathcal{D} = \langle D_i, R_j \rangle_{i<n, j \in J}$ and that τ is a pje generated by P_j ($j \in J$). Then, there is an $\mathcal{L}_n^*$-formula φ_τ such that $e(\tau) = \mathrm{def}(\varphi_\tau)$. Conversely, if φ is an $\mathcal{L}_n^*$-formula, then there is a pje τ_φ with $e(\tau_\varphi) = \mathrm{def}(\varphi)$.*

Proof. We associate with each pje τ the formula φ_τ as follows:

1. If τ is P_j for some $j \in J$, then $\varphi_\tau \stackrel{\text{def}}{=} P_j(v_0, \ldots, v_{n-1})$.

2. If σ is a pje and φ_σ is already defined, $Y \subseteq a(\sigma)$, $\{i_0, \ldots, i_k\} = a(\sigma) \setminus Y$, and τ is $\pi_Y \sigma$, then $\varphi_\tau \stackrel{\text{def}}{=} (\exists v_{i_0} \ldots v_{i_k})\varphi_\sigma$.

3. If σ and ξ are pje's with φ_σ, φ_ξ already defined, and τ is $\sigma \bowtie \xi$, then $\varphi_\tau \stackrel{\text{def}}{=} \varphi_\sigma \wedge \varphi_\xi$.

Note that for a pje τ, $\operatorname{frv}(\varphi_\tau) = \{v_i : i \in a(\tau)\}$.

We show the first part by induction.

i. If $\tau = P_j$, then $\operatorname{def}(\varphi_\tau) = \operatorname{def}(P_j(v_0, \ldots, v_{n-1})) = R_j = e(P_j) = e(\tau)$.

ii. Let σ be a pje, $\tau = \pi_Y \sigma$, $\operatorname{frv}(\varphi_\sigma) = \{v_{i_0}, \ldots, v_{i_r}\}$, $Y = \{i_0, \ldots, i_s\}$, where $\{i_0, \ldots, i_s\} \subseteq \{i_0, \ldots, i_r\}$, and $e(\sigma) = \operatorname{def}(\varphi_\sigma)$. Then, $\varphi_\tau = (\exists v_{i_{s+1}} \ldots v_{i_r})\varphi_\sigma$, and we have

$$\begin{aligned}
\operatorname{def}(\varphi_\tau) &= \{f \restriction Y : \mathcal{D} \models_f (\exists v_{i_{s+1}} \ldots v_{i_r})\varphi_\sigma\} \\
&= \{f \in \prod\{D_i : i \in Y\} : (\exists g \in \operatorname{def}(\varphi_\sigma))[g \restriction Y = f]\} \\
&= \{f \in \prod\{D_i : i \in Y\} : (\exists g \in e(\sigma))[g \restriction Y = f]\} \\
&= \pi_Y e(\sigma) \\
&= e(\tau).
\end{aligned}$$

iii. Let $\tau = \sigma \bowtie \xi$. Set $\varphi_\tau \stackrel{\text{def}}{=} \varphi_\sigma \wedge \varphi_\xi$. Let $\operatorname{frv}(\varphi_\sigma) = \{v_{i_0}, \ldots, v_{i_r}\}$ and $\operatorname{frv}(\varphi_\xi) = \{v_{j_0}, \ldots, v_{j_s}\}$. Then $\operatorname{frv}(\varphi_\tau) = \{v_{i_0}, \ldots, v_{i_r}\} \cup \{v_{j_0}, \ldots, v_{j_s}\}$. Set $Y_0 \stackrel{\text{def}}{=} \{i_0, \ldots, i_r\}$, $Y_1 \stackrel{\text{def}}{=} \{j_0, \ldots, j_s\}$ and $Y \stackrel{\text{def}}{=} Y_0 \cup Y_1$. Now,

$$\begin{aligned}
\operatorname{def}(\varphi_\tau) &= \{f \restriction Y : \mathcal{D}_f \models \varphi \wedge \xi\} \\
&= \{f \in \prod\{D_i : i \in Y\} : f \restriction Y_0 \in \operatorname{def}(\varphi_\sigma) \text{ and } f \restriction Y_1 \in \operatorname{def}(\varphi_\xi)\} \\
&= \{f \in \prod\{D_i : i \in Y\} : f \restriction Y_0 \in e(\sigma) \text{ and } f \restriction Y_1 \in e(\xi)\} \\
&= e(\tau).
\end{aligned}$$

Conversely, we associate with each $\mathcal{L}_n^*$ formula φ a pje τ_φ as follows:

1. if $\varphi = P_j(v_0, \ldots, v_{n-1})$, then $\tau_\varphi \stackrel{\text{def}}{=} P_j$ and $a(P_j) = n$;

2. if $\varphi = (\exists v_i)\psi$, then $\tau_\varphi \stackrel{\text{def}}{=} \pi_Z \tau_\psi$, where $Z = a(\tau_\psi) \setminus \{v_i\}$, and $a(\tau_\varphi) = Z$;

3. if $\varphi = \psi \wedge \chi$, then $\tau_\varphi \stackrel{\text{def}}{=} \tau_\psi \bowtie \tau_\chi$ and $a(\tau_\varphi) = a(\tau_\psi) \cup a(\tau_\chi)$.

It is not hard to see that, for each pje τ and each $\mathcal{L}_n^*$-formula φ, we have $\tau_{\varphi_\tau} = \tau$, and $\varphi_{\tau_\varphi} = \varphi$. $\square$

Corollary 3.2. *The query results obtainable from project-join expressions over n-ary relations are exactly the truth relations of restricted $\mathcal{L}_n^*$-formulas.*

If $\mathcal{D} = \langle D, R_0, \ldots, R_k \rangle$ is a model of $\mathcal{L}_n^*$, and τ a pje, then $e(\tau)$ is the data table obtained by interpreting $P_0, \ldots, P_k$ by $R_0, \ldots, R_k$, and projection and join in a natural way. Each pje τ defines an operator on data tables of an appropriate scheme: we can think of τ as a query, and $e(\tau)$ as its result.

Observe that we have not prescribed that the formulas of interest are in prenex form where all quantifiers appear in the front; thus, variables can be reused. However, each n-variable formula is semantically equivalent to an ω-variable formula ψ in prenex form with the same free and possibly more bound variables, in the sense that for any model, $\mathrm{def}(\varphi) = \mathrm{def}(\psi)$ in that model. To see this, we define inductively a mapping ζ from $\mathcal{L}_n$-formulas to prenex $\mathcal{L}_\omega$-formulas as follows (see also Aho et al., 1979):

1. If $\varphi = P(v_0, \ldots, v_{n-1})$ or $\varphi = (\exists v_i)\psi$, then φ is already a prenex $\mathcal{L}_\omega$-formula, and we set $\zeta(\varphi) = \varphi$.
2. Let $\varphi = \psi \wedge \chi$. For each variable $v_i \in \mathrm{var}(\psi) \cup \mathrm{var}(\chi)$ which does not occur freely in $\zeta(\psi)$ and $\zeta(\chi)$, consistently replace v_i by a new variable v_j in one of $\zeta(\psi)$ or $\zeta(\chi)$, where v_i is bound. Then, move the quantifiers to the front to obtain $\zeta(\varphi)$.

Even though there are pje's which translate into arbitrarily long prenex $\mathcal{L}_\omega$-formulas, the expressive power of pje's is strictly weaker than that of prenex $\mathcal{L}_\omega$-formulas, as was shown in Aho et al. (1979). In view of the results above — and those to follow — this comes as no surprise.

4 Cylindric structures

In this section, we shall interpret pje's in algebraic structures the investigation of which was begun in 1940's by A. Tarski. One aim of his work was to find an algebraic interpretation of first order logic in analogy to the successful algebraization of propositional logic via Boolean algebras. For notational reasons, we suppose from now on that $U = n = \{0, \ldots, n-1\}$.

One difficulty we encounter when considering a data table R is that $a(R)$ can be proper subset of U. However, in the translation we have in mind, we consider all occurring relations as n-ary; in other words, data tables need to be interpreted as n-ary without losing $a(R)$. The reason why we want to deal with n-ary relations is that algebras of relations of the same arity have nicer behavior than "heterogeneous" algebras.

One way to achieve the above goal is to take the view that a column of R which contains all information does not contain any useful information. For example, if $a(R) = U$, $Y = U \setminus \{0\}$, and $R = D_0 \bowtie \pi_Y R$, then it stands to reason that $\pi_Y R$ carries the same amount of information as R. This is supported by the fact that for such R, the formula $(\exists v_1, \ldots, v_{n-1})P(v_0, \ldots, v_{n-1})$ is universally valid in the model $\langle D, R \rangle$. Hence, the translation of $D_0 \bowtie \pi_Y R$ is semantically equivalent to to translation of $\pi_Y R$. Alternatively, since all dependencies which we encounter are domain independent in the sense of Fagin (1982), we could add a new element to each domain D_i, while retaining R. If, say, $a(R) = \{0, 1\}$, then we set $R' \stackrel{\mathrm{def}}{=} R \times D_2' \times \ldots \times D_{n-1}'$, where $D_i' \stackrel{\mathrm{def}}{=} D_i \cup \{u\}$ for some $u \notin D$.

The relevant columns of R' are those in which u does not occur. At any rate, let us assume from now on that all considered data tables R have $R \neq D_i \bowtie \pi_{U \smallsetminus \{i\}} R$.

Let $\mathcal{D} = \langle D_i, R_j \rangle_{i<n, j \in J}$ be a database, and let R be a relation: $R \subseteq \prod_{i \in a(R)} D_i$. Following Németi (1991) and Imielinski & Lipski (1984) we can then interpret R as an n-ary relation

$$\xi(R) \stackrel{\text{def}}{=} \{f \in \prod_{i<n} D_i : f \upharpoonright a(R) \in R\}.$$

Note that $\xi(\emptyset) = \emptyset$, and $\xi(\prod_{i<n} D_i) = \prod_{i<n} D_i$. We require an operation that tells us which columns of $\xi(R)$ are the relevant ones obtained from R. To this end, we define the *cylindrifications* C_i, $i < n$, as follows. Let $S \subseteq \prod_{i<n} D_i$, and $i < n$. The i-th cylindrification C_i is defined by

$$C_i S \stackrel{\text{def}}{=} \{f \in \prod_{i<n} D_i : (\exists g \in S)(g \upharpoonright n \smallsetminus \{i\} = f \upharpoonright n \smallsetminus \{i\})\}.$$

The *dimension set* ΔS of S is defined as

$$\Delta S \stackrel{\text{def}}{=} \{i < n : C_i S \neq S\}. \tag{4.1}$$

We now can recover the relevant columns of R:

Lemma 4.1. *Let R_j be a data table over a database $\langle D_i, R_j \rangle_{i<n, j \in J}$. Then, $a(R_j) = \Delta\xi(R_j)$ for every $j \in J$.*

Proof. Suppose that $i \notin a(R_j)$, and that $f \in C_i \xi(R_j)$ with $g \in \xi(R_j)$ witnessing this fact. Then, $f \upharpoonright n \smallsetminus \{i\} = g \upharpoonright n \smallsetminus \{i\}$; in particular, $f \upharpoonright a(R_j) = g \upharpoonright a(R_j)$, since $a(R_j) \subseteq n \smallsetminus \{i\}$ by the hypothesis. Thus, $f \in \xi(R_j)$ by the definition of ξ. Hence $i \notin \Delta\xi(R_j)$.

For the converse, let $i \in a(R_j)$. By our global assumption that $R_j \neq D_i \bowtie \pi_{n \smallsetminus \{i\}} R_j$, there are $g \in R_j$ and $x \in D_i$, such that the function $g' : a(R_j) \to \prod_{i<n} D_i$ which agrees with g on $a(R_j) \smallsetminus \{i\}$, and $g'(i) = x$ is not an element of R_j. Then any extension of g' is in $C_i \xi(R_j)$. On the other hand, $g' \upharpoonright a(R_j) \notin R_j$. It follows that $C_i \xi(R_j) \neq R_j$, and therefore $i \in \Delta\xi(R_j)$. $\square$

For every database $\mathcal{D} = \langle D_i, R_j \rangle_{i<n, j \in J}$ we define another database $\mathcal{D}'$:

$$\mathcal{D}' = \langle D_i, \xi(R_j) \rangle_{i<n, j \in J}.$$

Thus every data table $\xi(R_j)$ has scheme n. For any pje τ we define its value $e'(\tau)$ in $\mathcal{D}'$ inductively:

- $e'(P_j) = \xi(R_j)$
- $e'(\pi_Y \sigma) = \xi(\pi_Y e'(\sigma))$
- $e'(\sigma \bowtie \rho) = e'(\sigma) \bowtie e'(\rho)$.

This construction enables us to interpret any $\pi_Y \tau$ even if Y is not a subset of $a(\tau)$, since every $e'(\tau)$ has scheme n. Following Imielinski & Lipski (1984), we define *unrestricted pje*'s, upje's, as pje's without the restriction that we can form $\pi_Y \tau$ only if $Y \subseteq a(\tau)$. We say that $\mathcal{D}' \models \tau = \sigma$ iff $e'(\tau) = e'(\sigma)$ for any upje's σ and τ. It is easy to check that $\mathcal{D} \models \tau = \sigma$ iff $\mathcal{D}' \models \tau = \sigma$ for any pje's τ and σ.

We are ready to define the class of algebras that we will use to interpret pje's.

Definition 4.2. *Let α be an ordinal. The structure*

$$\mathfrak{Rel}(D_i : i < \alpha) = \langle \mathcal{P}(\prod_{i<\alpha} D_i), \cap, C_i \rangle_{i<\alpha}$$

is called the full cylindric (lower) semilattice of dimension α over D_i $(i < \alpha)$. *If $\mathfrak{A}$ is a subalgebra of $\mathfrak{Rel}(D_i : i < \alpha)$, then $\mathfrak{A}$ is called a* set cylindric (lower) semilattice of dimension α, *in short, a* scsl_α. *That is,*

$$\mathsf{scsl}_\alpha \stackrel{\text{def}}{=} \mathbf{S}\{\mathfrak{Rel}(D_i : i < \alpha) : \textit{ sets } D_i\}.$$

Using the fact that cylindrifications commute (see e.g. Henkin et al., 1971), we define generalized cylindrifications $C_{(X)}$ for $X = \{x_0, \ldots, x_k\}$ by

$$C_{(X)}S = \begin{cases} C_{x_0}C_{x_1}\ldots C_{x_k}S & \text{if } X \neq \emptyset, \\ S & \text{otherwise.} \end{cases}$$

Thus $C_iS = C_{(\{i\})}S$.

It is not hard to see that the following are valid equations:

$$\tau \cap \tau = \tau \tag{4.2}$$

$$\tau \cap \sigma = \sigma \cap \tau \tag{4.3}$$

$$(\tau \cap \sigma) \cap \rho = \tau \cap (\sigma \cap \rho) \tag{4.4}$$

$$\tau \cap C_i\tau = \tau \tag{4.5}$$

$$C_iC_j\tau = C_jC_i\tau \tag{4.6}$$

$$C_i(\tau \cap C_i\sigma) = C_i\tau \cap C_i\sigma. \tag{4.7}$$

These properties say that $\cap$ is a semilattice operation, the cylindrifications are commuting closure operations, and 4.7 is a modularity law. This set of properties gives rise to an abstract class of algebras: A *(diagonal free) cylindric (lower) semilattice of dimension* α, in short, a csl_α is an algebra $\langle A, \cdot, c_i, \rangle_{i<\alpha}$ such that for all $x, y \in A$ and $i, j < \alpha$,

$$\langle A, \cdot \rangle \text{ is a semilattice.} \tag{C1}$$

$$x \cdot c_ix = x \tag{C2}$$

$$c_ic_jx = c_jc_ix \tag{C3}$$

$$c_i(x \cdot c_iy) = c_ix \cdot c_iy. \tag{C4}$$

Some structural properties of cylindric semilattices can be found in Düntsch (1993).

Proposition 4.3. *The equational theory of upje's over n-dimensional databases is equivalent to the equational theory of* scsl_n. *That is, there is a translation* tr *of upje's onto cylindric expressions such that for all upje's τ and σ, $\tau = \sigma$ holds in every n-dimensional database iff* $\mathrm{tr}(\tau) = \mathrm{tr}(\sigma)$ *is valid in every* scsl_n.

Proof. We define tr as follows:

1. $\mathrm{tr}(P_j) = P_j$;
2. $\mathrm{tr}(\pi_Y \sigma) = c_{(n \smallsetminus Y)}\, \mathrm{tr}(\sigma)$;
3. $\mathrm{tr}(\sigma \bowtie \rho) = \mathrm{tr}(\sigma) \cdot \mathrm{tr}(\rho)$.

Let $\mathcal{D}' = \langle D_i, \xi(R_j)\rangle_{i<n, j\in J}$ be a database, and let $\mathfrak{A} \subseteq \mathfrak{Rel}(D_i : i < n)$ be the subalgebra generated by $\xi(R_j)$, $j \in J$. Let P_j take the value $\xi(R_j)$ in $\mathfrak{A}$: $P_j^{\mathfrak{A}} = \xi(R_j)$. We claim that, for any upje τ, the value of τ in $\mathcal{D}'$ and the value of its translation in $\mathfrak{A}$ coincide: $e'(\tau) = \mathrm{tr}(\tau)^{\mathfrak{A}}$. We proceed by induction. First let $\tau = P_j$:

$$e'(P_j) = \xi(R_j) = P_j^{\mathfrak{A}} = \mathrm{tr}(P_j)^{\mathfrak{A}}.$$

If $\tau = \pi_Y \sigma$, then

$$\begin{aligned} e'(\pi_Y \sigma) &= \xi(\pi_Y e'(\sigma)) \\ &= \xi(\pi_Y \,\mathrm{tr}(\sigma)^{\mathfrak{A}}) \\ &= \{f \in \prod_{i<n} D_i : (\exists g \in \mathrm{tr}(\sigma)^{\mathfrak{A}}) g \restriction Y = f \restriction Y\} \\ &= C_{(n \smallsetminus Y)}\, \mathrm{tr}(\sigma)^{\mathfrak{A}} \\ &= (c_{(n \smallsetminus Y)}\, \mathrm{tr}(\sigma))^{\mathfrak{A}} \\ &= \mathrm{tr}(\pi_Y \sigma)^{\mathfrak{A}}. \end{aligned}$$

Finally, assume that $\tau = \sigma \bowtie \rho$:

$$\begin{aligned} e'(\sigma \bowtie \rho) &= e'(\sigma) \bowtie e'(\rho) \\ &= \mathrm{tr}(\sigma)^{\mathfrak{A}} \bowtie \mathrm{tr}(\rho)^{\mathfrak{A}} \\ &= \mathrm{tr}(\sigma)^{\mathfrak{A}} \cap \mathrm{tr}(\rho)^{\mathfrak{A}} \\ &= (\mathrm{tr}(\sigma) \cdot \mathrm{tr}(\rho))^{\mathfrak{A}} \\ &= \mathrm{tr}(\sigma \bowtie \rho)^{\mathfrak{A}}. \end{aligned}$$

Hence

$$\mathcal{D}' \models \sigma = \rho \text{ iff } \mathfrak{A} \models \mathrm{tr}(\sigma) = \mathrm{tr}(\rho)$$

under the evaluation $P_j^{\mathfrak{A}} = \xi(R_j)$. Since every n-dimensional database defines a scsl_n together with a valuation of the variables, we get that

$$\not\models \sigma = \rho \text{ implies } \not\models \mathrm{tr}(\sigma) = \mathrm{tr}(\rho).$$

Conversely, every scsl_n with a valuation gives rise to an n-dimensional database, whence

$$\not\models \mathrm{tr}(\sigma) = \mathrm{tr}(\rho) \text{ implies } \not\models \sigma = \rho.$$

Finally, tr is a map onto cylindric expressions, since we defined it on the class of all upje's. Thus the equational theory of n-dimensional upje's is equivalent to the equational theory of scsl_n. $\square$

It was shown in Imielinski & Lipski (1984) that there are infinitely many non-equivalent upje's over one ternary relation. Below, we use the preceding result to show that the same is true for pje's over two binary relations:

Proposition 4.4. *There is an infinite two-generated* scsl_2.

Proof. Let $D = D_0 = D_1 = \{0, 1, 2, \ldots\},\ R = \{\langle 3k, 5k\rangle : k \in D\}$, and $S = \{\langle 2k, 7k\rangle : k \in D\}$. Now, define inductively

$$\begin{aligned} T_0 &= C_0 R \cap S \\ T_{2n+1} &= C_1 T_{2n} \cap R, \quad n \in D \\ T_{2n} &= C_0 T_{2n-1} \cap S, \quad n > 0. \end{aligned}$$

It is straightforward to check that

$$\begin{aligned} T_{2n} &= \{5 \cdot 10^n \cdot \langle 2k, 7k\rangle : k \in D\} \\ T_{2n+1} &= \{10^{n+1} \cdot \langle 3k, 5k\rangle : k \in D\} \end{aligned}$$

and thus, the scsl_2 generated by R and S is infinite. $\square$

Corollary 4.5. *There are infinitely many non-equivalent pje's in two binary relations over the same attribute set.*

The free scsl_2 on one generator contains just eight elements. Thus the presence of two generators is necessary to obtain infinitely many non-equivalent expressions, and the result above is the best possible.

5 Axiomatizability of project-join dependencies

Yannakakis & Papadimitriou (1982) define a class of dependencies, which they call *algebraic dependencies*: an algebraic dependency is a project-join dependency $\tau \leq \sigma$ such that both τ and σ are pje's built up using only one propositional variable. They give a complete axiomatization for implications between algebraic dependencies. During the proof, they consider an equation whose translation to cylindric expressions is the following:

$$\begin{aligned} &c_1(c_2(c_1(c_0R \cdot c_2R) \cdot c_0R) \cdot c_0(c_1R \cdot c_2R)) = \\ &c_1(c_2R \cdot c_0(c_1R \cdot c_2(c_0R \cdot c_1(c_2R \cdot c_0R)))). \end{aligned} \tag{5.1}$$

It is shown by Yannakakis & Papadimitriou that while (5.1) is a scsl_n-valid equation, it cannot be derived from the equations defining csl_3.

The solution suggested by Yannakakis & Papadimitriou is to define another semantics for algebraic dependencies, in which every n-tuple is represented as an ω-sequence of the tuple in question, e.g.

$$\overline{\langle x_0, x_1, x_2\rangle} = \langle\langle x_0, x_1, x_2\rangle, \langle x_0, x_1, x_2\rangle, \ldots\rangle.$$

The *extended relation* $\overline{R}$ of R is defined as

$$\overline{R} = \{\overline{f} : f \in R\}.$$

Then they give a finite schema[2] of equations axiomatizing the above extended relations. The result below suggests that without considering extended relations finite axiomatizability of pjd's is impossible.

In the preceding section we proved that (valid) upjd's are equivalent to (valid) equations of scsl_n. It follows that the implication $\Sigma \models \tau$ ($\Sigma \cup \{\tau\}$ a finite set of upjd's) is equivalent to a quasi-equation (a universal Horn-formula with equations as atoms) of cylindric expressions. The following result shows that there is no finite set of formulas axiomatizing n-dimensional upjd's if $n > 2$.

Theorem 5.1. *(Hodkinson & Mikulás, 1998)*
The quasi-equational theory of scsl_n *is not finitely axiomatizable whenever* $n > 2$.

The idea of the proof (which is rather technical) is an ultraproduct construction. For every $k \in \omega$, an algebra $\mathfrak{A}_k$ is defined such that $\mathfrak{A}_k$ is not representable as a scsl_n while their non-trivial ultraproducts are in scsl_n. Then non-finite axiomatizability of scsl_n follows. Since scsl_n is axiomatizable by a set of quasi-equations, there must be valid quasi-equations q_k such that $\mathfrak{A}_k \not\models q_k$, $k \in \omega$. However, we do not know these quasi-equations. We conjecture that, in fact, there are valid equations (probably using only one variable) similar to (5.1) showing the non-representability of $\mathfrak{A}_k$.

Another open problem is whether scsl_ω is axiomatizable by a finite schema of equations. Note that in scsl_ω there are infinitely many operations c_i ($i \in \omega$), thus finite axiomatizability is impossible (without using schemas). See Section 6 for more on finite schema axiomatizability of ω-dimensional algebras of relations.

6 Cylindric dependencies

Cosmadakis (1987) has introduced a more general type of data dependency which uses a larger fragment of first order logic than pj-expressions, and which is untyped. In this setting, a database is a pair $\langle D, \mathcal{R}\rangle$, where D is a nonempty set, and $\mathcal{R} = \{R_i : i \in I\}$ is a set of relations over D with finite arity $\gamma(i)$. The corresponding query language $\mathcal{L}^+_\omega$ is a first order language with connectives $C = \{\wedge, \vee, \exists, \mathbf{T}, \mathbf{F}, =\}$, which has an infinite set $V = \{v_i : i \in \omega\}$ of individual variables, and a set $\mathcal{P} = \{P_i : i \in I\}$ of predicate symbols such that each P_i has the same arity as R_i. Formulas are defined in the usual way.

A formula is called *restricted*, if all its atomic subformulas are of the form $R_i(v_0, \ldots, v_{\gamma(i)-1})$. It is well known that, in the presence of equality and sufficiently many variables, each basic formula is equivalent to a restricted formula, see Henkin et al. (1985) 4.3.1.

The queries defined by the language are just the truth sets of formulas as defined in (2.1) and (2.2). A *first order dependency* (fod) is an expression of the form $\varphi \to \psi$, where φ and ψ are $\mathcal{L}^+_\omega$-formulas.

[2] A finite schema of equations is a finite set of equations using parameters. By substituting actual values for these parameters we get equations. For instance, $c_i x = c_i c_i x$ is a schema and $c_0 x = c_0 c_0 x$ is one of its instances.

Note that the symbol $\to$ is not part of our language. Again, we can identify databases and models of $\mathcal{L}^+_\omega$. We say that a database $\mathcal{D}$ *satisfies a dependency* $\varphi \to \psi$, if $\mathcal{D} \models \varphi$ implies $\mathcal{D} \models \psi$. A set Σ of fod's *semantically implies an fod* σ, if $\mathcal{D} \models \Sigma$ implies $\mathcal{D} \models \sigma$ for all models $\mathcal{D}$ of $\mathcal{L}^+_\omega$.

To define the corresponding algebraic language requires some preparation, since in the intended models of this language, relations of different rank have to live together. In order to achieve this, we need to enhance our notion of database, similar to our procedure in Section 4. Given a database $\mathcal{D} = \langle D, \{R_i : i \in I\}\rangle$ we define a "dummy embedding" ν of the relations R_i into ${}^\omega D$ by

$$\nu(R_i) = \{a \in {}^\omega D : \langle a_0, \dots, a_{\gamma(i)-1}\rangle \in R_i\},$$

see Németi (1991).

We also need to define the *diagonal relations* $D_{jk} = \{a \in {}^\omega D : a_j = a_k\}$. Since we have made the relations R_i compatible over the same base set, we have the set operations $\cap,\ \cup$ at our disposal. As before, the i-th cylindrification is a unary operator defined by

$$C_i R = \{a \in {}^\omega D : (\exists b \in R) a \restriction \omega \setminus \{i\} = b \restriction \omega \setminus \{i\}\}.$$

The language of cylindric expressions consists of the predicate symbols of $\mathcal{L}^+_\omega$, two binary operators $\cdot,\ +$, unary operators $c_i,\ i < \omega$, and constants $d_{ij},\ i, j < \omega$, as well as $0, 1$. The set CE of *cylindric expressions* is the smallest set H such that, for all $i, j < \omega$,

1. $0, 1 \in H$,
2. $P_i \in H$,
3. $d_{ij} \in H$,
4. if $\tau, \sigma \in H$, then $\tau \cdot \sigma,\ \tau + \sigma \in H$,
5. if $\tau \in H$, then $c_i \tau \in H$.

With some abuse of language, we shall also use CE as an abbreviation for "cylindric expression".

The semantics of CE's are provided by the following meaning function:

1. $mng(0) = \emptyset,\ mng(1) = {}^\omega D$,
2. $mng(P_i) = \nu(R_i)$,
3. $mng(d_{ij}) = D_{ij}$,
4. $mng(\tau \cdot \sigma) = mng(\tau) \cap mng(\sigma)$,
5. $mng(\tau + \sigma) = mng(\tau) \cup mng(\sigma)$,
6. $mng(c_i \tau) = C_i(mng(\tau))$.

We can now define a translation function ξ from restricted formulas to CE's as follows:

1. $\xi(\mathbf{F}) = 0,\ \xi(\mathbf{T}) = 1$,
2. $\xi(P_i) = P_i$,
3. if φ, ψ are $\mathcal{L}^+_\omega$-formulas, then
   (a) $\xi(\varphi \wedge \psi) = \xi(\varphi) \cdot \xi(\psi)$,
   (b) $\xi(\varphi \vee \psi) = \xi(\varphi) + \xi(\psi)$,
4. if φ is an $\mathcal{L}^+_\omega$-formula, then $\xi((\exists v_i)\varphi) = c_i\xi(\varphi)$.

It is not hard to see that ξ is bijective, and that

Proposition 6.1. *(Henkin et al., 1985, Cosmadakis, 1987)*
Let $\mathcal{D}$ be a model of $\mathcal{L}^+_\omega$. Then, for every $\mathcal{L}^+_\omega$-formula φ and every CE τ,

i. $\mathrm{def}_{\mathcal{D}}(\varphi) = mng(\xi(\varphi))$,

ii. $mng(\tau) = \mathrm{def}_{\mathcal{D}}(\xi^{-1}(\tau))$.

Thus, $\mathcal{L}^+_\omega$-formulas and CE's are equal in expressive power.

A *cylindric dependency* is a string of the form

$$\tau = \sigma,$$

where τ and σ are CE's. We denote the set of all cylindric dependencies by CD.

Cosmadakis (1987) gives a finite set of axiom schemas for CD's, which we have somewhat shortened, using results from Henkin et al. (1971); the reader will have no difficulty in checking that the systems are equivalent.

C1. A set of equations which say that $\langle CE, \cdot, +, 0, 1\rangle$ is a distributive lattice with smallest element 0 and largest element 1.

C2. $c_i 0 = 0$

C3. $\tau \cdot c_i\tau = \tau$

C4. $c_i(\tau \cdot c_i\sigma) = c_i\tau \cdot c_i\sigma$

C5. $c_i(\tau + \sigma) = c_i\tau + c_i\sigma$

C6. $c_i c_j \tau = c_j c_i \tau$

C7. $d_{ii} = 1$

C8. If $j \neq i, k$, then $d_{ik} = c_j(d_{ij} \cdot d_{jk})$

C9. $c_j P_m = P_m$ for all $j \geq \gamma(m)$

for all $i, j, k \in \omega$, $m < n$, and $\tau, \sigma \in CE$.

An abstract algebra $\mathfrak{C}$ which satisfies these axioms is called a *cylindric lattice with generators* P_k. If each element of $\mathfrak{C}$ is finite dimensional in the sense of (4.1), we call $\mathfrak{C}$ *finite dimensional*. If the set of generators is understood or unimportant, we just speak of $\mathfrak{C}$ as a cylindric lattice. As usual, a cylindric lattice $\mathfrak{C}$ is called *representable* if it can be embedded into the product of algebras of the form

$$\langle \mathcal{P}(^{\omega}D), \cap, \cup, \emptyset, {}^{\omega}D, C_i, D_{ij} \rangle_{i,j \in \omega}.$$

The axioms are clearly sound for the intended models, i.e. for representable cylindric lattices. We define $\vdash$ as the derivability relation in equational logic using the cylindric lattice axioms. Cosmadakis (1987) now makes the following

Claim 1. *The axiom system is complete, i.e. whenever* $\Sigma \cup \{\tau\}$ *is a set of cylindric dependencies, then* $\Sigma \models \tau$ *implies* $\Sigma \vdash \tau$.

One simple reason why the claim failsis that one axiom which is included in the axioms for cylindric algebras of Henkin et al. (1971) has been overlooked:

Proposition 6.2. *Let* $i \neq j$, *and* X *be the dependency* $d_{ij} \cdot c_i(d_{ij} \cdot \tau) \leq \tau$.

1. *X holds in every database.*
2. *X cannot be derived from the axioms.*

Proof. 1. Let $\mathfrak{D}$ be a representable cylindric lattice, and suppose that $f \in mng(d_{ij} \cdot c_i(d_{ij} \cdot \tau))$. Then, $f_i = f_j$, and $f \in C_i(D_{ij} \cap mng(\tau))$. Thus, there is some $g \in {}^{\omega}D$ such that $g = D_{ij} \cap mng(\tau)$ and, for every $j \neq i$, $f_j = g_j$. Now, $g \in D_{ij}$ implies that $g_i = g_j = f_j$, and $f \in D_{ij}$ shows that in fact $f = g$. Hence, $f \in mng(\tau)$.

2. Consider an algebraic query language with one relational symbol P. Let $\mathfrak{A}$ be an abstract algebra which satisfies the cylindric lattice axioms and such that $0 < P < d_{01} < 1$ and $c_0 P = c_1 P = 1$ in $\mathfrak{A}$. It is not difficult to check that such an $\mathfrak{A}$ exists, witnessing the fact that X is not derivable from the axioms. □

Even adding a new axiom

C10. $d_{ij} \cdot c_i(d_{ij} \cdot \tau) \leq \tau$

will not make the system complete as the following shows:

Proposition 6.3. *The axiom system enlarged by C10 is not complete.*

Proof. Consider the following cylindric dependency from 3.2.68 of Henkin et al. (1985)

$$c_0\rho \cdot c_1\sigma \cdot c_2\tau \leq c_0 c_1 c_2 (c_2(c_1\rho \cdot c_0\sigma) \cdot c_1(c_2\rho \cdot c_0\tau) \cdot c_0(c_2\sigma \cdot c_1\tau)).$$

It is straightforward, if somewhat tedious, to show that this dependency holds in all models. On the other hand, it is shown in Henkin et al. (1985), that it cannot be derived from the axioms for cylindric algebras given in Henkin et al. (1971), p. 162, of which our system is a part. □

The following result shows that, in fact, it is impossible to give a finite schema of universal axioms for axiomatizing representable cylindric lattices[3].

Theorem 6.4. *Let Ax be a set of universal formulas axiomatizing (the quasi-equational theory of) representable cylindric lattices of dimension ω. Then Ax contains infinitely many variables.*

Proof. Recall from Henkin et al. (1985) that a representable cylindric algebra of dimension α, an RCA_α, is a representable cylindric lattice that is also closed under complementation w.r.t. the top element. That is, we have a Boolean set algebra with additional operators instead of a bounded lattice.

Andréka (1998), Theorem 3, shows that the class RCA_ω cannot be axiomatized by universal formulas using finitely many variables. Her strategy is to define, for every $n \in \omega$, abstract algebras $\mathfrak{A}_n$ such that $\mathfrak{A}_n$ is not representable, while its n-generated subalgebras are representable. She defines, for every $n \in \omega$, a valid equation that fails to hold in $\mathfrak{A}_n$, thus showing the non-representability of $\mathfrak{A}_n$.

It turns out that we can define valid quasi-equations q_n in the language of cylindric lattices that fail in the cylindric lattice reduct of $\mathfrak{A}_n$. The operation of *substitution* s^i_j is defined as

$$s^i_j x = \begin{cases} c_i(d_{ij} \cdot x) & \text{if } i \neq j, \\ x & \text{otherwise.} \end{cases}$$

Then q_n is defined as the sentence

$$\begin{gathered} (\forall R, R_0, \ldots, R_m, X_{01}, \ldots, X_{m-1m}) \\ \textstyle\bigwedge_{i\neq j} R_i \cdot R_j = 0 \wedge \bigwedge_i R_i \leq R \wedge \bigwedge_{i,k} c_k R_i = c_k R \wedge \bigwedge_{i\neq j}(X_{ij} \cdot d_{ij} = 0 \wedge X_{ij} + d_{ij} = 1) \\ \textstyle\to R \leq c_{(m)}(\prod_i s^0_i c_1 \ldots c_m R \cdot \prod_{i\neq j} X_{ij}). \end{gathered}$$

Intuitively, q_n says that if we can "split" the projection of the element R onto the 0-th coordinate into $m+1$ disjoint parts $R_0, \ldots, R_m$, then there must be an ω-sequence f such that $f_i \neq f_j$ for all distinct $i, j \leq m$. The validity of q_n is routine to check. The other details of the proof are identical to the proof of Andréka (1998), to which we refer the interested reader. ☐

Corollary 6.5. *There is no finite set of (universal) axiom schemas such that $\Sigma \models \tau$ iff $\Sigma \vdash \tau$ for every set $\Sigma \cup \{\tau\}$ of cylindric dependencies.*

Let

$$\langle \mathsf{Fml}, \wedge, \vee, \mathbf{F}, \mathbf{T}, P_k, \exists v_i, v_i = v_j \rangle_{i,j\in\omega, i\in I}$$

be the absolutely free algebra of formulas of $\mathcal{L}^+_\omega$. If Σ is a set of first order dependencies and $\varphi, \psi \in \mathsf{Fml}$, we define an equivalence relation $\equiv_\Sigma$ on Fml by

(6.1) $$\varphi \equiv_\Sigma \psi \text{ iff } \Sigma \models \varphi \to \psi \text{ and } \Sigma \models \psi \to \varphi.$$

The set $\mathsf{Fml}/\equiv_\Sigma$ can be made into a cylindric lattice with generators P_k, which we denote by $\mathrm{cl}(\Sigma)$. It is claimed in Cosmadakis (1987) that the following logical representation theorem holds:

[3] We did not precisely define the notionof a finite schema. One necessary condition, however, is that it should use finitely many (algebraic) variables.

Claim 2. *For every cylindric lattice $\mathfrak{M}$ with generators P_k, there is a set Σ of fod's such that*

$$\mathfrak{M} \equiv \mathrm{cl}(\Sigma).$$

If we replace *cylindric lattice* by *cylindric algebra*, then the claim is true (see Henkin et al., 1985). However, having a distributive lattice instead of a Boolean algebra as a ground structure is too weak for the claim to continue to hold for cylindric lattices, as the following counterexample shows:

Proposition 6.6. *There is a cylindric lattice $\mathfrak{M}$ such that $\mathfrak{M}$ is not isomorphic to $\mathrm{cl}(\Sigma)$ for any set Σ of fod's.*

Proof. Let $M = \{0, p, 1\}$ with $0 < p < 1$, $d_{ij} = 1$, $c_i p = p$ for all $i, j < \omega$. It is easily checked that there is such a cylindric lattice $\mathfrak{M}$. Assume that $\mathfrak{M} \cong \mathrm{cl}(\Sigma)$ for some set Σ of fod's. The conditions $d_{ij} = 1$ imply that $\Sigma \models (\forall x)(\forall y)x = y$. Thus, if $\langle D, P\rangle$ is a model of Σ, then $|D| = 1$. Furthermore, $c_i p = p$ tells us that $\Sigma \models (p \leftrightarrow (\exists v_i)p)$ for all $i < \omega$. Thus, in any model, $P = \emptyset$ or $P = {}^{\omega}D$ which shows that $|\,\mathrm{cl}(\Sigma)| \leq 2$. $\square$

The reason behind the failure of the claim seems to be the following: In cylindric algebras, a locally finite structure (i.e. a structure where every element has finite dimension in the sense of (4.1)) is simple if and only if it is subdirectly irreducible (Henkin et al., 1971, 2.4.43). Thus, a non-trivial onto homomorphic mapping from a subdirectly irreducible locally finite algebra is in fact an isomorphism. For (locally finite) cylindric lattices, the two concepts fall apart as the example shows, and an onto homomorphism need not be an isomorphism, as assumed in the proof of the claim in Cosmadakis (1987).

Representation problems in algebra are closely connected to completeness problems in logic. Loosely speaking, non-representable algebras correspond to non-standard models which exist in case the axiom system for the logic under consideration is not complete for the intended class of models. In Cosmadakis (1987) it is stated that every locally finite cylindric lattice is representable:

Claim 3. *For every locally finite cylindric lattice $\mathfrak{M}$ there is a family of databases $\langle \mathcal{D}_i\rangle_{i\in I}$ such that $\mathfrak{M}$ is isomorphic to a subalgebra of a product of the models $\mathcal{D}_i$.*

In contrast we show

Proposition 6.7. *There is a locally finite cylindric lattice with three generators which is not representable.*

Proof. We use the following example adapted from Andréka & Németi (1991): Let $\mathfrak{M}$ be a cylindric lattice generated by the elements p, q, r which have the following properties:

1. $|\{p, q, r\}| = 3.$
2. $\Delta p = \Delta q = \Delta r = \{0, 1\}$
3. $0 < p = c_0 p \cdot c_1 p \leq d_{01} < 1$

4. $0 < q + r < c_1 p$

5. $q \cdot r = 0$

6. $c_0 q = c_0 r$.

Assume that $\{P, Q, R\} \subseteq {}^{\omega}D$ for some D, satisfying the properties above. For $S \in \{P, Q, R\}$ and $i < 2$ we denote by $pr_i S$ the set $\{f_i : f \in S\}$. Then

$$\begin{aligned} C_0 P &= D \times pr_1 P \times D \times D \times \ldots \\ C_1 P &= pr_0 P \times D \times D \times \ldots \end{aligned}$$

and $P = C_0 P \cap C_1 P$ implies that

$$P = pr_0 P \times pr_1 P \times D \times D \times \ldots.$$

By $P \subseteq D_{01}$, there is some $d \in D$ such that $pr_0 P = pr_1 P = \{d\}$. From $C_0 Q = C_0 R$ and $\Delta Q = \Delta R = \{0, 1\}$, we infer that $pr_1 Q = pr_1 R$. Furthermore, $Q, R \subseteq C_1 P$ implies that $pr_0 Q = pr_0 R = \{d\}$. It follows that

$$Q = \{d\} \times pr_1 Q \times D \times D \times \ldots = \{d\} \times pr_1 R \times D \times D \times \ldots = R,$$

contradicting that $Q \cap R = \emptyset$. $\square$

There is no way to (equationally) repair the claim, since it is shown in Andréka & Németi (1991) that, unlike representable cylindric algebras, the representable cylindric lattices do not form an equational class. One reason for this is that, again unlike for locally finite cylindric algebras, the concepts *simple*, *subdirectly irreducible*, and *directly indecomposable* do not coincide (see also Düntsch, 1993).

One might wonder what happens if we restrict our investigation to the finite dimensional case as we did with project-join dependencies. The definitions of fod's and (representable) cylindric lattices are easily modified to finitely many variables and finite dimensions. Then the question is whether there is a finite set of (quasi-)equations axiomatizing n-dimensional cylindric dependencies. Again the answer is negative:

Theorem 6.8. *(Hodkinson & Mikulás, 1998)*
The quasi-equational and the equational theories of representable cylindric lattices of dimension n is not finitely axiomatizable if $n > 2$.

Note that in this case we have a stronger negative result than in the case of pjd's: we cannot finitely axiomatize even the valid equations between cylindric expressions, not to mention valid inferences between equations. The proof of the above theorem is another ultraproduct construction. From this construction it is easy to define valid equations that show the non-representability of the algebras. We refer the reader to Hodkinson & Mikulás (1998) for details.

References

Aho, A. V., Sagiv, Y. & Ullman, J. D. (1979). Equivalences among relational expressions. *SIAM Journal of Computing*, **8**, 218–246.

Andréka, H. (1998). Complexity of equations valid in algebras of relations. Part I. *Annals of Pure and Applied Logic*, **89**, 149–209.

Andréka, H. & Németi, I. (1991). Representable positive cylindric algebras do not form a variety. Preprint, Mathematical Institute, Hungarian Academy of Science.

Andréka, H., Németi, I. & Sain, I. (1998). Algebraic logic. Mathematical Institute, Hungarian Academy of Sciences.

Chang, C. & Keisler, J. (1971). Model Theory. Amsterdam: North Holland.

Codd, E. F. (1970). A relational model of data for large shared databanks. *Comm. of the ACM*, **13**, 377–387.

Cosmadakis, S. S. (1987). Database theory and cylindric lattices. In A. K. Chandra (Ed.), *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, 411–420, Los Angeles, CA. IEEE Computer Society Press.

de Morgan, A. (1864). On the syllogism: IV, and on the logic of relations. *Transactions of the Cambridge Philosophical Society*, **10**, 331–358. (read April 23, 1860) Reprinted in de Morgan (1966).

de Morgan, A. (1966). On the Syllogism, and Other Logical Writings. New Haven: Yale Univ. Press.

Düntsch, I. (1993). A note on cylindric lattices. In C. Rauszer (Ed.), *Algebraic Methods in Logic and Computer Science*, vol. 28 of *Banach Center Publications*, 231–238. Warsaw: Banach Center.

Düntsch, I. & Gediga, G. (1997). Algebraic aspects of attribute dependencies in information systems. *Fundamenta Informaticae*, **29**, 119–133.

Düntsch, I. & Gediga, G. (1998). Logical tools for rule based data analysis. In J. Komorowski, I. Düntsch & A. Skowron (Eds.), *Workshop on "Synthesis Of Intelligent Agent Systems From Experimental Data", ECAI 98.*

Fagin, R. (1982). Horn clauses and data dependencies. *Journal of the ACM*, **29**, 952–985.

Fagin, R. & Vardi, M. Y. (1986). The theory of data dependencies - a survey. In M. Anshel & W. Gewirtz (Eds.), *Mathematics of Information Processing*, 19–71. AMS.

Henkin, L., Monk, J. D. & Tarski, A. (1971). Cylindric algebras, Part I. Amsterdam: North Holland.

Henkin, L., Monk, J. D. & Tarski, A. (1985). Cylindric algebras, Part II. Amsterdam: North Holland.

Hodkinson, I. & Mikulás, S. (1998). Non-finite axiomatizability of reducts of algebras of relations. Submitted for publication.

Imielinski, T. & Lipski, W. (1984). The relational model of data and cylindric algebras. *Journal of Computer and System Sciences*, **28**, 80–102.

Németi, I. (1991). Algebraizations of quantifier logics. *Studia Logica*, **50**, 485–569. Updated version available from `ftp.math-inst.hu/pub/algebraic-logic/survey.ps`.

Novotný, M. (1997a). Applications of dependence spaces. In Orłowska (1997), 193–246.

Novotný, M. (1997b). Dependence spaces of information systems. In Orłowska (1997), 193–246.

Orłowska, E. (Ed.) (1997). Incomplete Information – Rough Set Analysis. Heidelberg: Physica – Verlag.

Peirce, C. S. (1870). Description of a notation for the logic of relatives, resulting from an amplification of the conceptions of Boole's calculus of logic. *Memoirs of the American Academy of Sciences*, **9**, 317–378. Reprint by Welch, Bigelow and Co., Cambridge, Mass., 1870, pp. 1–62.

Schröder, E. (1890 - 1905). Vorlesungen über die Algebra der Logik, Volumes 1 to 3. Leipzig: Teubner. Reprinted by Chelsea, New York, 1966.

Tarski, A. (1941). On the calculus of relations. *Journal of Symbolic Logic*, **6**, 73–89.

Tarski, A. & Givant, S. (1987). A formalization of set theory without variables, vol. 41 of *Colloquium Publications*. Providence: Amer. Math. Soc.

Yannakakis, M. & Papadimitriou, C. H. (1982). Algebraic dependencies. *Journal of Computer and System Sciences*, **25**, 80–102.