Logical reduction of relations: from relational databases to Peirce's reduction thesis

Sergiy Koshkin

Department of Mathematics and Statistics
University of Houston-Downtown
One Main Street
Houston, TX 77002
e-mail: koshkins@uhd.edu

Abstract

We study logical reduction (factorization) of relations into relations of lower arity by Boolean or relative products that come from applying conjunctions and existential quantifiers to predicates, i.e. by primitive positive formulas of predicate calculus. Our algebraic framework unifies natural joins and data dependencies of database theory and relational algebra of clone theory with the bond algebra of C.S. Peirce. We also offer new constructions of reductions, systematically study irreducible relations and reductions to them, and introduce a new characteristic of relations, ternarity, that measures their 'complexity of relating' and allows to refine reduction results. In particular, we refine Peirce's controversial reduction thesis, and show that reducibility behavior is dramatically different on finite and infinite domains.

Keywords: relational algebra, relation scheme, attribute, Cartesian product, Boolean product, natural join, primitive positive formula, constraint satisfaction problem, co-clone, project-join expression, relative product, irreducible relation, teridentity, bonding algebra, Peirce's reduction thesis, subcubic graph

Introduction

We study decomposition of relations into simpler relations by logical operations expressible in terms of conjunctions and existential quantifiers on predicates, i.e. by primitive positive formulas of predicate calculus [22]. Such analysis goes back to the work of C.S. Peirce at the dawn of algebraic logic, see [7, 9, 21] for modern accounts. However, while some aspects of it have been developed by Schröder, Löwenheim, Tarski, and others [10], they gradually shifted the focus to predicates in formal theories and questions of axiomatization. On the other hand, decomposition of relations in algebras of relational clones or co-clones is closely related to Peirce's, but was developed independently of his work, from Post's study of the dual decomposition of functions on finite sets, see [23, 33]. Peirce's results were largely forgotten until modern formalizations of his bond algebra in [8, 17].

The original interest was linguistic and philosophical. A different stimulus came from the relational database model introduced by Codd [12], and developed by Fagin

[16], Rissanen [34], and others. It featured weaker than Peirce's, but closely related, notion of decomposition. Unfortunately, the two streams of literature remained largely disjoint. This paper is, in part, a unified survey of classical but little known (to logicians and mathematicians) results and their interconnections, and, in part, their extension by the author.

From the mathematical perspective, join decomposition of relations is somewhat analogous to factorization of polynomials. There is even a ready analog of polynomial's degree that measures its 'multiplicative complexity' – relation's arity (adicity, rank), the number of its attributes (places, positions). There is an even stronger analogy to the recent theory of factorization in highly non-cancellative monoids with degree-like height function [37]. Accordingly, we will call decomposition terms *factors*, and call decompositions *reductions* when all the factors have strictly lower arity than the relation itself.

A technical device of database theory that we will extensively use in this paper is the calculus of *attributed relations* (the term is due to [38], see also *named perspective* in [1, 3.2]). Those are collections of maps from a set of attributes, called the relation scheme, to the domain, rather than ordered lists of domain elements, as in ordinary set-theoretic relations. This allows us to identify positions in different relations through relation schemes, the same flexibility one gets by placing the same variable into different predicates in logical formulas, and removes much of combinatorial clutter that plagues definitions and calculus of operations on ordinary relations (compare to [8, 14]). Moreover, operations on attributed relations have better algebraic properties.

For example, the Cartesian product is commutative and associative on attributed relations, albeit only partially defined: relations that share attributes cannot be multiplied. What we get by extending it to all attributed relations is classically known as Boolean product, and it is still commutative and associative. In logical terms, it amounts to taking conjunctions of predicates with some variables identified instead of only free conjunctions. In predicate calculus, such expressions are called quantifier-free primitive positive formulas [22]. Its iteration, called the natural join, is the primary operation of the database theory. The reason is that join decompositions are closely associated with various data dependencies, and can be used to store, query and update the data more efficiently [2, 3, 16, 24, 28, 34].

We start by reviewing some known join reduction methods that rely on exploiting various dependencies among relation's tuples. The simplest one is functional dependency, when there is an attribute, called *key*, whose value determines the rest of its tuple (ID columns play this role in database tables). However, despite the abundance of reduction methods, it turns out that join irreducible relations also abound. Not only are there ones of arbitrarily high arity, but also 'almost all' relations are join irreducible on large domains (Theorem 8). Although this fact must have been known to experts, the author did not encounter it stated in print.

Thus, join reduction is analogous to factorization of polynomials over the field of rationals, with irreducibles of arbitrarily high degrees. This prompts one to look for extensions with a more manageable set of irreducibles. However, in line with the literature on the subject, rather than enlarging the class of relations we opt for stronger algebra on the same class. The stronger operation is projective join (projoin for short) that combines join with projections. In logical terms, we allow to existentially quantify ('project out') some variables in predicate conjunctions, i.e. consider all primitive positive predicate formulas.

While some project-join algebras have been studied in the more theoretical database literature [15, 38], they appeared much more prominently in the theory of constraint satisfaction problems (CSP) that also dates back to 1970s. The constraints are expressed by relations on finite domains, and the problems are to decide whether relations from a given set can simultaneously hold on some elements of the domain (be satisfied). Computational complexity of CSP was actively studied in computer science, and in 1978 Schaefer discovered a deep connection between it and closure properties of sets of relations under projective joins on 2-element domains [35]. The work initiated by Feder-Vardi and Jeavons in 1990s extended this connection to all finite domains, giving rise to what is now called the algebraic approach to complexity of CSP [6]. In particular, it turned out that if constraints are projoin complete, i.e. generate all relations, then the CSP is NP-complete (assuming P$\neq$NP). In general, projoin closures of relations are called relational clones or co-clones, and Schaefer characterized NP completeness in terms of them on 2-element domains. They are dual to the functional clones of Post and studied since 1960s [23, 33]. Join closures, called weak partial co-clones, also found applications to studying complexity of CSP, namely to refined complexity classification of NP-complete problems [20].

The projoin reduction problem on finite domains can be reformulated as asking whether the set of all unary and binary relations is projoin complete, i.e. whether they generate the co-clone of all relations. In hindsight, the affirmative answer to the same question on infinite domains goes back to Peirce. The proof device, introduced by Peirce under the quaint name of 'hypostatic abstraction', amounts, in database terms, to attaching key attribute(s) to the relation, and then projecting them out after join reducing the augmented relation (Theorem 9). Reducibility of all relations to unary and binary ones is more analogous to factorization of polynomials over the field of real numbers, where they reduce to linear and quadratic factors.

Counterintuitively, the situation is much more complex on finite domains, and we introduce some further devices that allow to projoin reduce some relations when hypostatic abstraction does not (Section 5). They exploit the connection between projections and unions (existential quantification and disjunction). In particular, we generalize to projoins Fagin's characterization of certain joins in terms of multivalued dependencies (Theorem 11), and show how to convert complements (negations) of certain joins into projoins. However, we do not resort to a central device of clone theory, the Pol-Inv Galois connection [6, 23], as our approach is to develop the more elementary Peircean methods that construct reductions explicitly.

While projection and join can be conveniently folded into a single projoin operation, this operation is not an iteration of any binary operation, like Cartesian product and join were. However, one can iterate a particular binary projoin classically known as relative product [7] (composition of binary relations is its restriction to them). Relative product is associative only on a restricted class of factors, and generates only an (ostensibly) narrow subclass of projoins, which we call bonds following [17].

The bond algebra is, more or less, Peirce's project-join algebra. In a surprise, he established that bond reducibility is almost equivalent to general projoin reducibility. All relations of arity 4 and higher are projoin reducible if and only if they are bond reducible, and it is only on ternaries (ternary relations) that the two notions diverge (Theorem 13). There are some projoin reducible but bond irreducible ternaries, notably the teridentity relation I_3 that contains all and only identical triples of domain elements and plays a key role in converting projoins into bonds. Bond reducibility of all relations on infinite domains to unaries, binaries and ternaries is known as Peirce's reduction thesis (see Theorem 19 for a precise formulation), and it is equivalent to the better known projoin reducibility to binaries alone, see e.g. [27]. The seeming discrepancy fueled a long historical controversy [9, 21].

Motivated by the thesis, we introduce the notion of ternarity of a relation as the minimal number of ternaries in its bond reductions to unaries, binaries and ternaries, and study its properties. The reason for singling out ternaries is that they do the main work of 'relating' different attributes in a relation as represented by reductions. We establish a close connection between complete bond reductions and subcubic graphs, and prove that ternarity of non-degenerate n-ary relations is always $n-2$ on infinite domains (Theorem 16), one unit of ternarity per unit of arity over $n=2$. This refines the original reduction thesis. The proof uses graph-theoretic methods, pioneered by Peirce and prominent in the recent work on reduction [13, 14]. A counterexample then shows that on finite domains this equality can fail already for $n=4$, and we prove that it fails for 'almost all' relations on large finite domains.

The paper is organized as follows. Section 1 introduces our terminology and notation. In Sections 2-3 we review some standard results on Cartesian products and natural joins, stated in terms of attributed relations, and add results on join irreducibility. Sections 4-5 introduce projective joins (projoins) and associated reduction methods, old and new, including the hypostatic abstraction that settles the reduction problem on infinite domains. The irreducibility results are weaker for projoins, and only concern some restricted classes of them. We also explain why reduction behavior is so different in finite and infinite cases. In Section 6 we introduce Peirce's bonds, and explain his algorithm for converting projoins into them that features teridentity. Peirce's reduction thesis is then derived as a corollary of the results for projoins. Section 7 defines projoin graphs and bonding diagrams that allow us to apply graph-theoretic methods to bond reductions in Section 8, which studies complete reductions and introduces ternarity. In Section 9 we use ternarity to refine Peirce's reduction thesis and give some counterexamples. In the last section, we summarize our conclusions and state some open problems.

1 Preliminaries

We use the standard set-theoretic notation and terminology for sets and relations [25]. Relations are defined on a set $\mathcal{D}$ called the *domain* and are subsets of its Cartesian powers $\mathcal{D}\times\cdots\times\mathcal{D}$. When $R\subseteq\mathcal{D}^n$ the number n is called the relation's *arity* and the relation is called n-ary relation or simply *n-ary* used as a noun. For $n=1,2,3,4$ we use the shorthands unary, binary, ternary, quaternary, respectively.

Elements of $\mathcal{D}^n$ are called n-tuples or just *tuples*, when n is understood or immaterial. If $a\in\mathcal{D}^n$ it's i-th member is denoted a_i. We adopt the usual convention of canonically identifying tuples of tuples with longer tuples, and hence of identifying $\mathcal{D}^n\times\mathcal{D}^m$ with $\mathcal{D}^{n+m}$, and so on. It is often convenient to interpret tuples as maps from the set of relation's positions $\mathbb{N}_n := \{1,2,\ldots,n\}$ to $\mathcal{D}$.

Some standard n-aries that can be defined on any domain will be called and denoted as follows: the *empty relations* $\emptyset_n$ with no tuples; the *universal relations* $U_n := \mathcal{D}^n$ that contain all possible n-tuples; the *identity relations* I_n that contain all and only n-tuples with identical members; and the *diversity relations* D_n that contain all and only n-tuples with pairwise distinct members. Each of those can be relativized to proper subsets $\mathcal{A} \subset \mathcal{D}$ that we will indicate by the upper index, e.g. $I_n^{\mathcal{A}}$ contains all and only n-tuples with identical members from $\mathcal{A}$.

The set of all maps from S to $\mathcal{D}$ is denoted $\mathcal{D}^S$, the set of all subsets of S is denoted $\mathcal{P}(S)$, and its cardinality is denoted $|S|$. As is well known, $|\mathcal{D}^n| = |\mathcal{D}|^n$, $|\mathcal{D}^S| = |\mathcal{D}|^{|S|}$, and $|\mathcal{P}(S)| = 2^{|S|}$.

Attributed relation R is a subset $R \subseteq \mathcal{D}^\Sigma$ [13, 15, 38], where Σ is a set called the *relation scheme*, and its elements are called *attributes*. The *arity* of R is defined to be $|\Sigma|$, and its elements, which are now functions from Σ to $\mathcal{D}$, are also called tuples. We only consider finitary relations, so Σ is always finite, the ordinary relations correspond to $\Sigma = \mathbb{N}_n$. When Σ is linearly ordered, e.g. $\Sigma \subset \mathbb{N}$, there is a canonical 1-1 correspondence between Σ and $\mathbb{N}_{|\Sigma|}$ obtained by listing elements of Σ in order, which induces a canonical 1-1 correspondence between relations on the scheme Σ and ordinary relations.

When $\Lambda \subseteq \Sigma$ we will denote $a_\Lambda \in \mathcal{D}^\Lambda$ the attributed subtuple of a consisting of its members in the Λ positions. When $\Lambda, \mathrm{M} \subseteq \Sigma$, and $\alpha \in \mathcal{D}^\Lambda$, $\beta \in \mathcal{D}^{\mathrm{M}}$ with $\alpha_{\Lambda\cap\mathrm{M}} = \beta_{\Lambda\cap\mathrm{M}}$, we define their *concatenation* $\alpha \mid \beta \in \mathcal{D}^{\Lambda\cup\mathrm{M}}$ as the union of ordered pairs when they are taken as functions from Σ to $\mathcal{D}$. The intersection condition is needed for the union to also be a function, and the concatenation is always defined when Λ, M are disjoint. When $\Sigma \subset \mathbb{N}$ the members of the concatenated tuple are listed according to the order of their attributes in $\mathbb{N}$. For example, $((\alpha_1, \alpha_3) \mid (\beta_2, \beta_3, \beta_5)) = (\alpha_1, \beta_2, \alpha_3, \beta_5)$ assuming $\alpha_3 = \beta_3$.

In the database model relations are visualized as rectangular tables with columns labeled by positions (attributes) and rows listing the tuple members. Database queries are then interpreted as operations on relations that extract relevant information and package it into simpler relations. Two operations inspired by this interpretation will be useful to us. The first is *projection* to a subset of positions $\Lambda \subseteq \Sigma$:

$$\pi_\Lambda R := \{a_\Lambda \,\big|\, a \in R\},$$

that simply deletes all non-Λ columns and removes duplicate tuples, if any, in the remaining Λ columns. When no confusion results, we write simply $\pi_{i_1,\dots,i_k} R$ instead of $\pi_{\{i_1,\dots,i_k\}} R$. The second operation is *selection* over a subset:

$$\sigma_{x_\Lambda=\alpha} R := \{a_{\Lambda^c} \,\big|\, a \in R, a_\Lambda = \alpha\}, \tag{1}$$

that leaves only rows with prescribed values in the Λ columns (given by $\alpha \in \mathcal{D}^\Lambda$), and then deletes those columns. Here $\Lambda^c := \Sigma \backslash \Lambda$ is the complement of Λ. Note that both $\pi_\Lambda R$ and $\sigma_{x_\Lambda=\alpha} R$ are attributed relations on the schemes Λ, Λ^c, respectively.

Unless otherwise stated, all predicates will be interpreted on a domain, and relations will be identified with the predicates they interpret. In particular, the same letter will be used for a relation and its predicate, i.e. $R(a_1, \dots, a_n)$ will mean the same as $(a_1, \dots, a_n) \in R$, and $\neg R$ will denote the complement of R in $\mathcal{D}^\Sigma$. This identification is convenient because attribution maps are naturally expressed in predicates by placing the same variables into multiple positions.

The same notational conventions apply to tuples of variables as to tuples of values, e.g. x_Λ is the subtuple of variables with the indices from Λ, $x_\Lambda \mid x_M$ is the concatenation of variable tuples, etc. The standard logical operations, conjunction $\wedge$, disjunction $\vee$, etc., will be used with the usual meaning, and for $\Lambda = \{i_1, \dots, i_k\}$ the multiple quantification $\exists x_{i_1} \dots \exists x_{i_1} R(x)$ will be abbreviated as $\exists x_\Lambda R(x)$. With these conventions and in terms of predicates, the projection is expressed simply as $\pi_\Lambda R\,(x_\Lambda) = \exists x_{\Lambda^c} R(x)$, and the selection as $\sigma_{x_\Lambda=\alpha} R\,(x_{\Lambda^c}) = R(\alpha \mid x_{\Lambda^c})$. For example, $\pi_{1,3} R\,(x, y) = \exists t R(x, t, y)$, and $\sigma_{x_{1,3}=(\alpha',\alpha'')} R\,(z) = R(\alpha', z, \alpha'')$.

2 Cartesian Products

The simplest way a relation decomposes into lower arity ones is when it is a Cartesian product of them. We need a slight generalization so that the relation does not cease to be a Cartesian product simply because its positions are permuted. This is taken care of when using attributed relations. We give the definition directly for any finite number of factors, but one can see that it comes from iterating the binary product. Recall that a *partition* $\Sigma = \Lambda_1 \cup \cdots \cup \Lambda_m$ of a set Λ is its representation as a disjoint union of subsets.

Definition 1. *Given disjoint finite subsets $\Lambda_i \subseteq \Sigma$ and attributed relations $R_i \subseteq \mathcal{D}^{\Lambda_i}$, their Cartesian product $R \subseteq \mathcal{D}^{\cup_i \Lambda_i}$ is the set of concatenations of tuples from R_i:*

$$R_1 \times \cdots \times R_m := \{(a^1 | \dots | a^m) \,\Big|\, a^i \in R_i\}.$$

*A relation $R \in \mathcal{D}^\Sigma$ is a **Cartesian product over a partition** $\Sigma = \Lambda_1 \cup \cdots \cup \Lambda_m$ when there exist $R^{\Lambda_i} \in \mathcal{D}^{\Lambda_i}$ such that $R = R^{\Lambda_1} \times \cdots \times R^{\Lambda_m}$, and R^{Λ_i} are called its **Cartesian factors**. It is called **degenerate** when it is a Cartesian product over a partition with $m > 1$ and all $\Lambda_i \neq \emptyset$, and it is called **l-aric** when all factors have the same arity l.*

We reiterate that, on this definition, Cartesian factors are attributed relations, and R is not their Cartesian product in the ordinary sense, but rather some permutation of it. By definition, Cartesian factorization is a reduction because all R^{Λ_i} have smaller arity $|\Lambda_i| < n$, and when it exists, the factors are none other than the projections $R^{\Lambda_i} = \pi_{\Lambda_i} R$.

The definition is more straightforward in terms of predicates, it means that the predicate of R is a *free conjunction* of lower arity predicates:

$$R(x_1, \dots, x_n) = R^{\Lambda_1}(x_{\Lambda_1}) \wedge \cdots \wedge R^{\Lambda_m}(x_{\Lambda_m}). \tag{2}$$

The following is a characteristic property of Cartesian products.

Theorem 1 (Independence criterion). *A relation $R \subseteq \mathcal{D}^\Sigma$ is a Cartesian product over a partition $\Sigma = \Lambda_1 \cup \cdots \cup \Lambda_m$ if and only if the values of its tuples on Λ_i can be chosen independently, i.e. for any collection of $\alpha^i \in \pi_{\Lambda_i} R$ there exists a common $a \in R$ such that $a_{\Lambda_i} = \alpha^i$.*

Proof. Let R be a Cartesian product over Λ_i. If $\alpha^i \in \pi_{\Lambda_i} R = R^{\Lambda_i}$ then $R^{\Lambda_i}(\alpha^i)$ holds for all i, and hence $R^{\Lambda_1}(\alpha^1) \wedge \cdots \wedge R^{\Lambda_m}(\alpha^m)$ holds. But by (2) this means that $R(a)$ holds for the concatenation $a := (\alpha^1 | \dots | \alpha^m)$. Thus, $a \in R$ and $a_{\Lambda_i} = \alpha^i$.

Conversely, suppose that R satisfies the independence condition over Λ_i. If $a \in R$ then $a_{\Lambda_i} \in \pi_{\Lambda_i} R$ by definition of projection, so $R \subseteq \pi_{\Lambda_1} R \times \cdots \times \pi_{\Lambda_m} R$. On the other hand, if $a \in \pi_{\Lambda_1} R \times \cdots \times \pi_{\Lambda_m} R$ then $a_{\Lambda_i} \in \pi_{\Lambda_i} R$, and, by the independence condition, there is a common $b \in R$ with $b_{\Lambda_i} = a_{\Lambda_i}$. Since Λ_i form a partition we have $a = b \in R$. Thus, $\pi_{\Lambda_1} R \times \cdots \times \pi_{\Lambda_m} R \subseteq R$ and they are equal. $\square$

Note that if we assign equal probability to all tuples of R degenerate over a partition Λ_i then a_{Λ_i} will be statistically independent random vectors on this sample space [29].

When testing for degeneracy it is sufficient to consider bipartitions only. Indeed, if R can be split into several factors then we can always group them into just two non-trivial factors, say, the first one and the product of the rest.

Example 1. The independence criterion can be used directly to establish degeneracy or non-degeneracy of some relations. Clearly, all unary relations are non-degenerate on any domain. On domains $\mathcal{D}$ with $|\mathcal{D}| \geq 2$ the identity relations I_n are non-degenerate for $n \geq 2$. Indeed, constant tuples $(\alpha, \ldots, \alpha), (\beta, \ldots, \beta) \in I_n$ for any $\alpha, \beta \in \mathcal{D}$. If I_n were degenerate, we could, by independence, assign constant α values to some Λ_i and constant β values to others. But if $\alpha \neq \beta$ their concatenation will not be in I_n, contradiction.

Similarly, for $|\mathcal{D}| \geq n$ the diversity relations D_n are non-degenerate ($D_n = \emptyset_n$ for $|\mathcal{D}| < n$). Indeed, any diverse (with no equal values) subtuple is in the projection of D_n to any proper subset of positions. But subtuples over disjoint subsets of positions can share values, and their concatenation will not be in D_n.

The following simple proposition gives two sufficient conditions of non-degeneracy that are easier to check in examples.

Theorem 2. *Suppose one of the following conditions holds for a relation $R \subseteq \mathcal{D}^\Sigma$.*
(i) *R is not universal, but for any non-empty proper subset $\emptyset \subset \Lambda \subset \Sigma$ we have $\pi_\Lambda R = \mathcal{D}^\Lambda$.*
(ii) *$\neg R$ is not empty, but for any $i \in \Sigma$ we have $\pi_i(\neg R) \neq \mathcal{D}$.*
Then R is non-degenerate.

Proof. (i) According to (2), R is a conjunction of $\pi_{\Lambda_i} R$ with proper subsets Λ_i. Since each projection is universal, by assumption, so must be their conjunction, contradiction.

(ii) Let $\delta \in \mathcal{D}$ be arbitrary. If R is a product negating (2) we obtain, by the de Morgan law:

$$\neg R(x_1, \ldots, x_n) = \neg R^{\Lambda_1}(x_{\Lambda_1}) \vee \cdots \vee \neg R^{\Lambda_m}(x_{\Lambda_m}).$$

Since $\neg R$ is not empty so is one of the disjuncts, say $\neg R^{\Lambda_1}$. Pick some $\alpha \in \neg R^{\Lambda_1}$ and some $i \notin \Lambda_1$, which exists because Λ_1 is proper, and set $a_i := \delta$, $a_{\Lambda_1} := \alpha$, then assign values for a_j with $j \notin \Lambda_1 \cup \{i\}$ arbitrarily. By construction, $\neg R^{\Lambda_1}(a_{\Lambda_1})$ holds and hence so does $\neg R(a)$ because $\neg R$ is a disjunction of $\neg R^{\Lambda_i}$. But δ was arbitrary, and so $\pi_i(\neg R) = \mathcal{D}$, contrary to the assumption. $\square$

Example 2. By Theorem 2 (i), the non-identity relations $\neg I_n$ with any $n \geq 2$ are non-degenerate (they trivially are for $n = 1$) when $|\mathcal{D}| \geq 2$. Indeed, $\pi_\Lambda(\neg I_n)$ is universal for any non-empty proper Λ because any tuple constant over Λ can be complemented by assigning a different value on $i \notin \Lambda$, and non-constant tuples can be complemented arbitrarily, with the result in $\neg I_n$ in both cases.

The above reasoning no longer applies to $R := \neg I_n^{\mathcal{A}}$ for $\mathcal{A}$ a non-empty proper subset of $\mathcal{D}$. However, $\pi_i(\neg R) = \pi_i(I_n^{\mathcal{A}}) = \mathcal{A} \neq \mathcal{D}$ for any i by assumption, and $\neg I_n^{\mathcal{A}}$ is non-degenerate by Theorem 2 (ii). In particular, by picking $\mathcal{A} = \{\alpha\}$ we see that $U_n \backslash \{(\alpha, \ldots, \alpha)\}$ is non-degenerate for any $\alpha \in \mathcal{D}$ when $|\mathcal{D}| \geq 2$.

The above examples already show that there are non-degenerate relations of arbitrarily high arity. Therefore, the situation with Cartesian factorization is similar to factorization of polynomials over $\mathbb{Q}$ with irreducible polynomials of arbitrarily high degrees. The next theorem shows, by a counting argument, that, as far as reductive power is concerned, the situation is even worse – 'almost all' relations are non-degenerate.

Theorem 3. *The share of degenerate n-ary relations among all such relations on a domain $\mathcal{D}$ asymptotically vanishes when $|\mathcal{D}| \geq 2$ and $n \to \infty$, or when $n \geq 2$ and $|\mathcal{D}| \to \infty$.*

Proof. The set of all n-ary relations on $\mathcal{D}$ is $\mathcal{P}(\mathcal{D}^n)$, so their total number is $|\mathcal{P}(\mathcal{D}^n)| = 2^{|\mathcal{D}|^n}$. For any degenerate relation there is a non-empty proper subset $\Lambda \subset \mathbb{N}_n$ such that it is a Cartesian product over the bipartition $\Lambda \cup \Lambda^c$, and every bipartition is counted twice by Λ due to the symmetry between Λ and Λ^c. There are $\binom{n}{k}$ subsets of $\mathbb{N}_n$ with $|\Lambda| = k$, and we can assign $2^{|\mathcal{D}|^{|\Lambda|}} = 2^{|\mathcal{D}|^k}$ and $2^{|\mathcal{D}|^{|\Lambda^c|}} = 2^{|\mathcal{D}|^{n-k}}$ different relations to each of the Cartesian factors. Of course, some of the products obtained in this way may coincide, so we only get an upper bound for the total number N_{deg} of degenerate relations:

$$N_{deg} \leq \frac{1}{2} \sum_{k=1}^{n-1} \binom{n}{k} 2^{|\mathcal{D}|^k + |\mathcal{D}|^{n-k}}.$$

By calculus, the function $|\mathcal{D}|^k + |\mathcal{D}|^{n-k}$ on $1 \leq k \leq n-1$ takes its maximal values at the endpoints when $|\mathcal{D}| \geq 2$, so $|\mathcal{D}|^k + |\mathcal{D}|^{n-k} \leq |\mathcal{D}| + |\mathcal{D}|^{n-1}$ for all k. Therefore,

$$N_{deg} \leq 2^{|\mathcal{D}|+|\mathcal{D}|^{n-1}-1} \sum_{k=1}^{n-1} \binom{n}{k} < 2^{n-1+|\mathcal{D}|+|\mathcal{D}|^{n-1}}, \qquad (3)$$

because $\sum_{k=1}^{n-1} \binom{n}{k} = 2^n - 2 < 2^n$. The conclusions now follow by applying calculus to the fraction $\frac{2^{n-1+|\mathcal{D}|+|\mathcal{D}|^{n-1}}}{2^{|\mathcal{D}|^n}}$. □

3 Joins and dependencies

As far as analysis of relations is concerned, Cartesian products do not take us very far. Asymptotically, in either domain size or arity, almost all relations do not factor into them. A natural move is to consider more general decompositions. In terms of predicates, Cartesian products correspond to free conjunctions, so let us allow conjunctions that are not necessarily free, i.e. some variables in them are identified. This leads to the notion of *natural join*, or join for short, introduced by Codd in the context of relational databases [12]. Its binary version, *Boolean product*, goes back to the 19th century [7]. Our definition of join parallels the definition of Cartesian product, the only difference is that the relation schemes no longer have to form a partition of Σ, only a cover.

Definition 2. *Given finite subsets $\Lambda_i \subseteq \Sigma$ and attributed relations $R_i \in \mathcal{D}^{\Lambda_i}$, their (natural) join $R \in \mathcal{D}^{\cup_i \Lambda_i}$ is defined as the set of concatenations of tuples from R_i that match on overlaps of Λ_i:*

$$R_1 \bowtie \cdots \bowtie R_m := \{(a^1 | \ldots | a^m) \,\Big|\, a^i \in R_i,\, a^i_{\Lambda_i \cap \Lambda_j} = a^j_{\Lambda_i \cap \Lambda_j}\}.$$

A relation $R \in \mathcal{D}^\Sigma$ is a ***(natural) join over a cover*** *$\Sigma = \Lambda_1 \cup \cdots \cup \Lambda_m$ when there exist $R^{\Lambda_i} \in \mathcal{D}^{\Lambda_i}$ such that $R = R^{\Lambda_1} \bowtie \cdots \bowtie R^{\Lambda_m}$, and R^{Λ_i} are called its* ***join factors****. It is called* ***join reducible*** *when $0 < |\Lambda_i| < |\Sigma|$.*

As the notation indicates, join of multiple factors is obtained by iterating Boolean product $\bowtie$. On attributed relations Boolean product is commutative and associative, just like Cartesian product, because ordering of positions does not get in the way.

In terms of predicates, joins are just conjunctions of predicates over subsets of variables with some variables shared among them:

$$R(x_1,\dots,x_n) = R^{\Lambda_1}(x_{\Lambda_1}) \wedge \cdots \wedge R^{\Lambda_m}(x_{\Lambda_m}). \tag{4}$$

This formula looks exactly like the free conjunction (2), except now Λ_i may overlap, e.g. $R^{1,3}(x_1,x_3) \wedge R^{1,2,4}(x_1,x_2,x_4) \wedge R^{3,4}(x_3,x_4)$ is a join. In mathematical logic, joins of predicates are called quantifier-free primitive positive formulas, and join closures are called weak partial co-clones in CSP literature [20, 22].

Note that (4) no longer implies that $R^{\Lambda_i} = \pi_{\Lambda_i}R$, as it did for Cartesian products. This is because some tuples in R^{Λ_i} may not have companion tuples in other R^{Λ_j} consistent with them on overlaps, and do not make it into the join. However, (4) remains valid if we *replace* R^{Λ_i} by $\pi_{\Lambda_i}R$ because all and only joinable tuples are present in the projections. In this weakened sense, joins share the constructive property of Cartesian products – their (minimized) factors can be recovered as projections of the join itself, as pointed out by Rissanen [34]. When the join factors contain only joinable tuples they are said to join *completely* [28, 2.4].

There is also an analog of the independence criterion for Cartesian products [24].

Theorem 4 (Join criterion). *A relation $R \subseteq \mathcal{D}^\Sigma$ is a join over a cover $\Sigma = \Lambda_1 \cup \cdots \cup \Lambda_m$ if and only if the values of its tuples on Λ_i can be chosen independently up to consistency on overlaps, i.e. for any collection of $\alpha^i \in \pi_{\Lambda_i}R$ with $\alpha^i_{\Lambda_i\cap\Lambda_j} = \alpha^j_{\Lambda_i\cap\Lambda_j}$ there is common $a \in R$ such that $a_{\Lambda_i} = \alpha^i$.*

The proof is analogous to the proof of Theorem 1 and we omit it.

Unfortunately, this criterion is not nearly as useful in detecting either join reducibility or irreducibility. More constructive reducibility conditions are usually formulated in terms of *data dependencies* [3, 16, 24, 28, 30], the simplest of which is functional.

Definition 3. *Let $\Lambda, \mathrm{M} \subseteq \Sigma$. We say that a relation $R \subseteq \mathcal{D}^\Sigma$ has a functional dependency $\Lambda \to \mathrm{M}$ when its Λ values determine its M values, i.e. for any $a, b \in R$ if $a_\Lambda = b_\Lambda$ then $a_\mathrm{M} = b_\mathrm{M}$. When $\mathrm{K} \to \mathrm{K}^c$ for some $\mathrm{K} \subset \Sigma$ then K is called a **key** of R, and more precisely a **k-key**, where $k = |\mathrm{K}|$.*

In databases, key column(s) are those whose entries determine the entire record, for example, the ID column. This situation was prototypical for choosing the terminology, the key column(s) code objects the relation is about, and the rest list their attributes. What matters to us is that availability of keys ensures join reducibility: instead of relating all attributes to their objects in a single relation we can relate them one at a time. The idea goes back to the work of Peirce and is in close affinity to his “hypostatic abstraction” (see [9, 21] and Section 4).

Theorem 5. *Any $R \subseteq \mathcal{D}^\Sigma$ with a k-key $\mathrm{K} = \{i_1,\dots,i_k\} \subseteq \Sigma$ decomposes into a join of $(k+1)$-aries as*

$$R = \bigotimes_{i\notin \mathrm{K}} \pi_{\mathrm{K}\cup\{i\}}R = R_{i_{k+1}} \bowtie \cdots \bowtie R_{i_n}, \tag{5}$$

where $R_i := \pi_{\mathrm{K}\cup\{i\}}R$. In the predicate form,

$$R(x_1,\dots,x_n) = R_{i_{k+1}}(x_{i_1},\dots,x_{i_k},x_{i_{k+1}}) \wedge \cdots \wedge R_{i_n}(x_{i_1},\dots,x_{i_k},x_{i_n}) \tag{6}$$

In particular, R is join reducible when $k \leq n-2$, and join reducible to binaries when $k = 1$.

Proof. Without loss of generality, we may rename the attributes into integers and assume $\Sigma = \mathbb{N}_n$, $\mathrm{K} = \{1, \ldots, k\}$. Suppose $a \in R$, then for any $i \geq k+1$ the subtuple $(a_1, \ldots, a_k, a_i) \in R_i = \pi_{\mathrm{K} \cup \{i\}} R$ because it is the projection of a to $\mathrm{K} \cup \{i\}$. Hence $a \in R_{k+1} \bowtie \cdots \bowtie R_n$.

Now suppose $a \in R_{k+1} \bowtie \cdots \bowtie R_n$, i.e. $(a_1, \ldots, a_k, a_i) \in R_i$ for all $i \geq k+1$. By definition of R_i, there must be tuples $b^i \in R$ with $b^i_j = a_j$ for $j \leq k$ and $b^i_i = a_i$. But the first k positions are a key, and all b^i coincide on them with a, therefore, $a = b^1 \in R$. $\square$

Example 3. In the identity relation I_n every position is a 1-key. Taking $\mathrm{K} = \{1\}$ and applying (6), we get the familiar reduction to binaries

$$I_n(x_1, \ldots, x_n) = I_2(x_1, x_n) \wedge \cdots \wedge I_2(x_{n-1}, x_n) = (x_1 = x_n) \wedge \cdots \wedge (x_{n-1} = x_n).$$

Example 4. The division with remainder quaternary $\mathrm{Div}(n, d, q, r)$ ($n = dq + r$ with n the dividend, $d \neq 0$ the divisor, q the quotient and $0 \leq r \leq d-1$ the remainder) on $\mathbb{N} \cup \{0\}$ has a 2-key consisting of the dividend and the divisor. Therefore, it reduces to the join of two ternaries: $\mathrm{Div}^{1,2,3}(n, d, q) = \big(q = \lfloor \frac{n}{d} \rfloor\big)$, where $\lfloor \cdot \rfloor$ is the floor function, and $\mathrm{Div}^{1,2,4}(n, d, r) = \big(n \equiv r \pmod{d}\big)$.

Functional dependency is not necessary for join reducibility, not even by join decompositions of the special form (5)-(6). This is fortuitous because it is a rare occurrence, just like its polar opposite, Cartesian independence of values, and we could not join reduce many relations if it was required.

Example 5. Consider the ternary R on $\mathcal{D} = \{\alpha, \beta\}$ given by the table below.

$$R = \begin{array}{|c|c|c|} \hline \alpha & \alpha & \alpha \\ \hline \alpha & \alpha & \beta \\ \hline \alpha & \beta & \alpha \\ \hline \alpha & \beta & \beta \\ \hline \beta & \alpha & \beta \\ \hline \end{array} \qquad R^{1,2} = \begin{array}{|c|c|} \hline \alpha & \alpha \\ \hline \alpha & \beta \\ \hline \beta & \alpha \\ \hline \end{array} \quad R^{1,3} = \begin{array}{|c|c|} \hline \alpha & \alpha \\ \hline \alpha & \beta \\ \hline \beta & \beta \\ \hline \end{array}$$

Since α is repeated in the first column the latter cannot be a key. Nonetheless, R is a join of the form (6): $R(x, y, z) = R^{1,2}(x, y) \wedge R^{1,3}(x, z)$, with the factors as shown above. This is because all possible combinations are present in R in the second and third position after α. As a result, $\sigma_{x_1 = \alpha} R$ is a Cartesian product of unaries, and so is $\sigma_{x_1 = \beta} R$ over the *same* partition (trivially, as a singleton).

The above example illustrates a more general dependency introduced by Delobel, Fagin and Zaniolo around 1977 [28, 7.11], the *multivalued dependency.*

Definition 4. *Let $\Sigma = \mathrm{M} \cup \Lambda_1 \cup \cdots \cup \Lambda_m$ be a partition. We say that an $R \subseteq \mathcal{D}^\Sigma$ has a multivalued dependency $\mathrm{M} \twoheadrightarrow \Lambda_1 \cup \cdots \cup \Lambda_m$ when for all $\alpha \in \pi_\mathrm{M} R$ the selections $\sigma_{x_\mathrm{M} = \alpha} R$ are Cartesian products over the common partition $\Lambda_1 \cup \cdots \cup \Lambda_m$. When Λ_i are singletons for all i then $\mathrm{K} := \mathrm{M}$ is called a **multikey** of R, and, more precisely, a **k-multikey** when $k = |\mathrm{K}|$.*

Note that both functional dependency and Cartesian independence (with $\mathrm{M} = \emptyset$) are special cases of multivalued dependency. It turns out that such dependency is both necessary and sufficient for join decompositions of the special form (7)-(8), when the same attributes are shared by all factors.

Theorem 6 (**Fagin [16]**). *Let $\Sigma = \mathrm{M} \cup \Lambda_1 \cup \cdots \cup \Lambda_m$ be a partition. An $R \subseteq \mathcal{D}^\Sigma$ has a join decomposition of the form*

$$R = \bigotimes_{i=1}^{m} \pi_{\mathrm{M}\cup\Lambda_i} R = R_1 \bowtie \cdots \bowtie R_m, \tag{7}$$

where $R_i := \pi_{\mathrm{M}\cup\Lambda_i} R$, or, equivalently,

$$R(x_1, \ldots, x_n) = R_1(x_\mathrm{M} \,|\, x_{\Lambda_1}) \wedge \cdots \wedge R_m(x_\mathrm{M} \,|\, x_{\Lambda_m}), \tag{8}$$

if and only if $\mathrm{M} \twoheadrightarrow \Lambda_1 \cup \cdots \cup \Lambda_m$ *is a multivalued dependency in R.*

Proof. One direction is trivial, if R is of the form (8) then substituting $x_\mathrm{M} = \alpha$ turns it into a free conjunction since Λ_i are disjoint. For the other direction, let $\Lambda := \Lambda_1 \cup \cdots \cup \Lambda_m$ and note that for any $\alpha \in \pi_\mathrm{M} R$ we have

$$\sigma_{x_\mathrm{M}=\alpha} R\,(x_\Lambda) = R_1^\alpha(x_{\Lambda_1}) \wedge \cdots \wedge R_m^\alpha(x_{\Lambda_m}), \tag{9}$$

with some $R_i^\alpha \subseteq \mathcal{D}^{\Lambda_i}$, by definition of Cartesian product over a partition. It remains to set $R_i(a) := R_i^{a_\mathrm{M}}(a_{\Lambda_i})$ when $a_\mathrm{M} \in \pi_\mathrm{M} R$ and 0 (false) otherwise to get (8). $\square$

Multivalued dependency reductions are popular in database design due to their constructive nature, but they do not exhaust all possible join reductions. For example, the diversity relation can be join reduced even to binaries, but not in the form (8):

$$D_n(x_1, \ldots, x_n) = \bigwedge_{i<j} I_2(x_i, x_j) = \bigwedge_{i<j} (x_i \neq x_j). \tag{10}$$

However, the more complex a dependency the harder it is to detect and use to produce reductions. The definition of join dependency, for example, amounts to just saying that the relation is a join (but see [30] for a somewhat more cogent characterization).

Be it as it may, we will now show that even taking all possible join reductions into account there are still irreducible relations of arbitrarily high arities. To this end, we observe that the proof of Theorem 2 did not use the fact that Λ_i form a partition, and so goes through for join reductions without a change. Therefore, its conclusion can be strengthened.

Theorem 7. *Suppose one of the following conditions holds for $R \subseteq \mathcal{D}^\Sigma$.*
(i) *R is not universal, but for any non-empty proper subset $\emptyset \subset \Lambda \subset \Sigma$ we have $\pi_\Lambda R = \mathcal{D}^\Lambda$.*
(ii) *$\neg R$ is not empty, but for any $i \in \Sigma$ we have $\pi_i(\neg R) \neq \mathcal{D}$.*
Then R is join irreducible.

The reasoning from Example 2 now shows that $\neg I_n$ and $\neg I_n^\mathcal{A}$ for non-empty proper $\mathcal{A} \subset \mathbb{N}_n$ are join irreducible for any $n \geq 1$ and $|\mathcal{D}| \geq 2$. Worse yet, join reductions leave asymptotically 'almost all' relations irreducible, albeit not in as strong a sense as Cartesian factorizations.

Theorem 8. *The share of join reducible n-ary relations among all such relations on a domain $\mathcal{D}$ is < 1 for $|\mathcal{D}| > n$, and asymptotically vanishes when $n \geq 1$ and $|\mathcal{D}| \to \infty$.*

Proof. Suppose R is join reducible to a conjunction (4). We can always interpret R^{Λ_i} as having attributes from any superset $\Lambda_i' \supset \Lambda_i$ by conjoining it with the universal relation on the scheme $\Lambda_i' \backslash \Lambda_i$. Since all Λ_i are proper subsets of $\mathbb{N}_n$ each of them is contained in one of $\Lambda \subset \mathbb{N}_n$ with $|\Lambda| = n-1$. Therefore, after merging predicates with identical schemes if necessary, we can represent R in the form (see [30])

$$R(x_1, \ldots, x_n) = \bigwedge_{|\Lambda|=n-1} R^\Lambda(x_\Lambda) = \bigwedge_{i=1}^{n} R^{\mathbb{N}_n \backslash \{i\}}\big(x_{\mathbb{N}_n \backslash \{i\}}\big).$$

There are n such Λ and $|\mathcal{P}(\mathcal{D}^\Lambda)| = 2^{|\mathcal{D}|^{|\Lambda|}} = 2^{|\mathcal{D}|^{n-1}}$ choices for each R^Λ. Some of them may be incompatible, so we get an upper bound on the number of join reducible relations: $N_{jred} \leq 2^{n|\mathcal{D}|^{n-1}}$. The conclusions now follow by applying calculus to the fraction

$$\frac{2^{n|\mathcal{D}|^{n-1}}}{2^{|\mathcal{D}|^n}} = 2^{-(|\mathcal{D}|-n)|\mathcal{D}|^{n-1}}. \tag{11}$$

□

Note that if $|\mathcal{D}|$ is fixed and $n \to \infty$ the ratio in (11) goes to ∞, not to 0, so we cannot conclude that increasing arity on a fixed domain also leads to a vanishing share of join reducible relations, as we could for degenerate ones in Theorem 3.

4 Projective joins and hypostatic abstraction

As noted by Rissanen [34], joins are the most general form of decomposition where the constituents can be recovered by taking projections. Yet they still admit irreducibles of arbitrarily high arity, and many of them. If we want a more manageable collection of irreducibles we need to relax the constructivism. There is also a more practical reason. When relations are entered into databases the expected data dependencies are often prescribed by design. What if potential entrants do not have them? They are then "pre-treated" before incorporation to mend that. For example, full names are expected to function as relation keys, but identical namesakes do occur sometimes. This is mended by attaching ID columns to the tables that restore uniqueness (and hence, functional dependency).

Such pre-treatment is often done "under the desk", but it is of interest to develop its devices systematically. Attaching a key appears already in C.S. Peirce's works on relations from 1890s under the name of "hypostatic abstraction" [8, 9]. Consider the ternary relation $G(x, y, z) :=$"x gives y to z". To reduce it to binaries, Peirce introduces ("hypostatizes") new abstracted objects t, the acts of giving, and treats x, y and z as their attributes, the giver, the gift, and the recipient. If we denote their relations to the object by $G'(t, x)$, $G''(t, y)$ and $G'''(t, z)$, respectively, then we can form a key-form join $G'(t, x) \wedge G''(t, y) \wedge G'''(t, z)$ that carries all the information of $G(x, y, z)$. Except it also features the acts of giving t that are not among the original attributes. What we really need to say is that *there is* such an act for given x, y and z:

$$G(x, y, z) = \exists t \left[G'(t, x) \wedge G''(t, y) \wedge G'''(t, z)\right]. \tag{12}$$

This is a reduction of sorts, but it is no longer a join, rather a projection of a join. Its factors cannot be recovered from G even if we know their relation schemes because of the freedom in t assignments. But it does fit well with the algebraic ideology of joins: first, we represent a given R as a projection of a higher arity relation $\widehat{R}$, and then factor $\widehat{R}$ into a join. While the factors are not projections of R, they are projections of $\widehat{R}$, as is R itself.

Definition 5. *The* ***projective join****, or just* ***projoin****, of attributed relations $R_i \subseteq \mathcal{D}^{\Lambda_i}$ with the set of projected attributes $\Gamma \subseteq \cup_i \Lambda_i$ is $\pi_\Gamma [R_1 \bowtie \cdots \bowtie R_m]$, i.e. the projection of their (natural) join to Γ. A relation $R \subseteq \mathcal{D}^\Sigma$ is a* ***projoin over a cover*** *$\mathrm{T} \cup \Sigma = \Lambda_1 \cup \cdots \cup \Lambda_m$ with $\mathrm{T} \cap \Sigma = \emptyset$ when there exist $R^{\Lambda_i} \subseteq \mathcal{D}^{\Lambda_i}$, called its* ***projoin factors****, such that*

$$R = \pi_\Sigma \big[R^{\Lambda_1} \bowtie \cdots \bowtie R^{\Lambda_m} \big] = \pi_\Sigma \widehat{R}.$$

The join $\widehat{R}$ is called the ***augmented relation****, and elements of T (projoin)* ***parameters****. R is called* ***projoin reducible*** *when it is a projoin with $0 < |\Lambda_i| < |\Sigma|$.*

The corresponding predicate formula for projoins is:

$$R(x_1, \dots, x_n) = \exists x_\mathrm{T} \left[R^{\Lambda_1}(x_{\Lambda_1}) \wedge \cdots \wedge R^{\Lambda_m}(x_{\Lambda_m}) \right]. \tag{13}$$

It is more typical to consider reductions in the algebra with two separate operations, join and projection, and project-join expressions are studied in [15, 38] and are part of a link between the relational model and Tarski's cylindric algebras [10, 18]. In terms of predicates, passing from joins to projoins means that in reductions we allow existential quantification on top of conjunction and identification of variables, i.e. consider primitive positive expressions (assuming the binary identity is among the available predicates). Up to switching from attributed to ordinary relations, the projoin reduction is essentially equivalent to reduction in the relational (co-clone) algebra of clone theory [14, 23, 33]. As such, it plays a key role in characterizing complexity of constraint satisfaction problems (CSP), and the study of generating sets of small arity in connection with them can be applied to solving Peircian reduction problems on finite domains [5].

Just as join dependencies can be expressed using joins, generalized data dependencies can be expressed using project-join equations [38]. With projoins, one can use the flexibility of augmenting the relation to make the decomposition methods for joins more broadly applicable. The simplest method used a key, but whether a relation has a key is, in some ways, a bookkeeping matter. This is another reason for moving from joins to projoins.

Definition 6. *We say that a relation $R \subseteq \mathcal{D}^n$* ***admits a key (multikey)*** *when it is a projection of a relation $\widehat{R} \subseteq \mathcal{D}^{\mathrm{T}\cup\Sigma}$ with a key (multikey), i.e. $R = \pi_\Sigma \widehat{R}$.*

Admitting a key is less of a bookkeeping matter than already having it. Note that T need not be the key in itself, the key may combine T with some attributes from the original relation. However, whether R admits a key or not does not depend on which attributes are used in it.

Lemma 1. *A relation $R \subseteq \mathcal{D}^\Sigma$ admits a k-key if and only if $|R| \leq |\mathcal{D}|^k$.*

Proof. Suppose $R = \pi_\mathrm{T} \widehat{R}$ and $\mathrm{K} \subseteq \mathrm{T} \cup \Sigma$ is a key of $\widehat{R}$ with $|\mathrm{K}| = k$. Since there are at most $|\mathcal{D}|^k$ distinct tuples in the K positions of $\widehat{R}$, and they determine the remaining values, $|\widehat{R}| \leq |\mathcal{D}|^k$. But projection never increases the cardinality of a relation, so $|R| = |\pi_\Sigma \widehat{R}| \leq |\widehat{R}| \leq |\mathcal{D}|^k$.

Conversely, suppose $|R| \leq |\mathcal{D}|^k$. Then we can pick any T of cardinality k with unused attributes, and assign to every tuple in $a \in R$ a unique label $t(a) \in \mathcal{D}^{\mathrm{T}}$. We define $\widehat{R}$ as the set of augmented tuples $\widehat{R} := \{(t(a), a) \,|\, a \in R\}$. By construction, the first k positions of $\widehat{R}$ are its key and $\pi_\Sigma \widehat{R} = R$. □

Recall from the previous section that relations *with* a 1-key are quite rare on finite domains. A striking consequence of this lemma is that on infinite domains, any relation *admits* a 1-key. Indeed, by the cardinal arithmetic, $|D|^n = |\mathcal{D}|$ for any finite n, so $|R| \leq |\mathcal{D}|^n = |\mathcal{D}|$ for any n-ary relation on $\mathcal{D}$. The next theorem generalizes Peirce's trick (12) for the relation of giving and connects it to relation keys, see also [8] for the general case.

Theorem 9 (Hypostatic abstraction). *Any $R \subseteq \mathcal{D}^\Sigma$ with $|R| \leq |D|^k$ decomposes into a projoin of $(k+1)$-aries as*

$$R = \pi_\Sigma \left[\bowtie_{i=1}^{n} \pi_{\mathrm{T}\cup\{i\}} \widehat{R} \right] = \pi_\Sigma [R_1 \bowtie \cdots \bowtie R_n] \,, \tag{14}$$

where $R_i := \pi_{\mathrm{T}\cup\{i\}} \widehat{R}$ for some $\widehat{R} \subseteq \mathcal{D}^{\mathrm{T}\cup\Sigma}$ with $|\mathrm{T}| = k$. In the predicate form,

$$R(x_1, \ldots, x_n) = \exists t_1 \ldots \exists t_k \left[R_1(t_1, \ldots, t_k, x_1) \wedge \cdots \wedge R_n(t_1, \ldots, t_k, x_n) \right]. \tag{15}$$

In particular, R is projoin reducible when $k \leq n-2$, and projoin reducible to binaries when $k = 1$.

Proof. The augmented relation $\widehat{R} \subseteq \mathcal{D}^{\mathrm{T}\cup\Sigma}$ is constructed in the proof of Lemma 1. Since the augmented attributes T are its k-key, by construction, the decomposition formula (14) follows from Theorem 5. In (15) we specialized to $\mathrm{T} = \{t_1, \ldots, t_k\}$ for concreteness, and ordered $\mathrm{T} \cup \Sigma$ so that the T positions go before the original ones. □

Hypostatic abstraction radically simplifies the picture of reducibility on infinite domains, and delivers a "small" set of irreducibles that eluded us with Cartesian products and joins. As we mentioned, this is similar to factorization of polynomials over $\mathbb{R}$, where the only irreducible ones are linear and quadratic.

Corollary 1. *Any n-ary with $n \geq 3$ on an infinite domain reduces to a projoin of n binaries. The only projoin irreducible relations on such domains are all unaries and non-degenerate binaries.*

Proof. The first claim is a direct consequence of cardinal arithmetic for infinite $|\mathcal{D}|$ and Theorem 9. Unaries have nothing to be reduced to. Binaries can only be reduced to unaries. But quantifying over a conjunction of unary predicates still produces a conjunction of unary predicates. Since unaries have only one variable each it is a free conjunction, i.e. an unaric Cartesian product. Thus, projoin reducible binaries must be degenerate. □

Löwenheim might have been the first to state and prove this result in a more formal manner in 1915 [27]. His proof used somewhat more cumbersome iterated pairing construction instead of hypostatic abstraction. In fact, the result is even stronger than stated. Recall that projoins are ranked by the number of parameters, with joins having none, 1-key hypostatic reductions like (12) having one, and so on. It follows from the proof that all higher arity relations are not only projoin reducible on infinite domains, but even projoin reducible with a single parameter. This is not the case on finite domains, as we will now demonstrate.

Example 6. Consider $\neg I_n$ on a finite domain $\mathcal{D}$. If it is projoin reducible then factors without augmented attributes can be dropped. Indeed, in the predicate representation (13) those factors can be taken out of the scope of quantifiers, and each carries a proper subset of $\neg I_n$'s variables. But $\neg I_n$ is universal on any proper subset of its positions, and hence so must be those factors since they enter conjunctively. Lower arity factors can be absorbed into those of arity $n-1$, so for $n=3$ any projoin reduction with one parameter condenses to just this

$$\neg I_3(x_1,x_2,x_3) = \exists t\,[R_1(t,x_1)\wedge R_2(t,x_2)\wedge R_3(t,x_3)]\,. \tag{16}$$

We will now restrict to a two-element domain $\mathcal{D} := \{\alpha,\beta\}$. Then t can only take two values, and, recalling the interpretation of the existential quantifier, (16) represents $\neg I_3$ as a disjunction of two unary conjunctions, i.e. a union of two unary Cartesian products. Possible cardinalities of such products on a two-element domain are $1,2,4$ and 8. But 8 is impossible because $|\neg I_3| = 6 < 8$, and 4 comes from a product of two doublets and a singleton. But a product of two doublets will include both (α,α) and (β,β), so multiplying it by any singleton will produce a constant tuple not in $\neg I_3$. So 4 is also impossible. As for 1 and 2, no pair of them can cover all 6 tuples of $\neg I_3$.

Thus, on two-element domains $\neg I_3$ is not projoin reducible with only one parameter. A similar counting argument works also for $|\mathcal{D}| = 3$. For larger $|\mathcal{D}|$ combinatorial considerations of this sort quickly become unmanageable. For $n > 3$ even single parameter projoin reductions are no longer of the simple form (16), as free variables can be shared and $\neg I_n$ need not be a union of Cartesian products, see Example 8.

A counting argument similar to that of Theorem 8 further shows that the reducibility behavior on finite domains is quite different.

Theorem 10. *The share of n-ary, $n \geq 3$, relations projoin reducible with k parameters among all such relations on a domain $\mathcal{D}$ is < 1 for $|\mathcal{D}| > \binom{n+k}{n-1}$, and asymptotically vanishes when $|\mathcal{D}| \to \infty$.*

Proof. As in the proof of Theorem 8, we transform the projoin into a form with factors of maximal possible arity in a reduction, $n-1$. Only this time, due to the augmented positions, their total number is $\binom{n+k}{n-1}$ rather than n (to which it reduces for $k=0$). The rest of the proof is analogous, and leads to $2^{-\left(|\mathcal{D}|-\binom{n+k}{n-1}\right)|\mathcal{D}|^{n-1}}$ as the upper bound for the share. $\square$

What are we to make of such a striking discrepancy between reducibilities on finite and infinite domains? The universal applicability of hypostatic abstraction on infinite domains derives directly from non-constructive and 'unnatural' bijections between $\mathcal{D}$ and its Cartesian powers. By a theorem of Tarski [19, 11.3], $|\mathcal{D}|^2 = |\mathcal{D}|$ for all infinite $\mathcal{D}$ is equivalent to the axiom of choice. Perhaps, it is of interest to consider reduction in models of set theory where availability of such bijections is restricted and the $|\mathcal{D}|, |\mathcal{D}|^2, \ldots$ hierarchy does not collapse, such as ZF models with Dedekind finite sets, or to restrict maps allowed in definitions of reducing relations directly. These may model relations on large finite domains better than do all relations on infinite domains of ZFC.

5 Projections as unions

As the results of the previous section show, reductions more general than (14)-(15) are still of interest on finite domains. Indeed, they may be of interest even on infinite domains as analyses of relations alternative to hypostatic abstraction. In this section we build on the relationship between existential quantification (projection) and unions exploited in Example 6 to construct such reductions. It is more technical than the rest of the paper and can be skipped without loss of continuity. We start by giving a projective version of Fagin's theorem.

Theorem 11. *An $R \subseteq \mathcal{D}^{\Sigma}$ has a projoin decomposition of the form:*

$$R = \pi_{\Sigma}\left[\bigbowtie_{i=1}^{m} R_i\right] = \pi_{\Sigma}[R_1 \bowtie \cdots \bowtie R_m]\,, \tag{17}$$

where $R_i \subseteq \mathcal{D}^{\mathrm{T}\cup\Lambda_i}$ for some partition $\Sigma = \Lambda_1 \cup \cdots \cup \Lambda_m$, or, in the predicate form,

$$R(x_1, \ldots, x_n) = \exists t_1 \ldots \exists t_k\, [R_1(t_1, \ldots, t_k, x_{\Lambda_1}) \wedge \cdots \wedge R_n(t_1, \ldots, t_k, x_{\Lambda_m})]\,, \tag{18}$$

if and only if it is a union of Cartesian products over the common partition Λ_i with no more than $|\mathcal{D}|^{|\mathrm{T}|}$ terms.

Proof. Suppose $R = \pi_{\Sigma}\widehat{R}$ with $\widehat{R} = \bigbowtie_{i=1}^{m} R_i$. Then, by Fagin's theorem, there is a multivalued dependency $\mathrm{T} \twoheadrightarrow \Lambda_1 \cup \cdots \cup \Lambda_m$ in $\widehat{R}$. But then, by definition, for all $a \in \pi_T\widehat{R}$ the selections $\sigma_{x_{\mathrm{T}}=a}R$ are Cartesian products over Λ_i, and

$$R = \bigcup_{a\in\pi_T\widehat{R}} \sigma_{x_{\mathrm{T}}=a}R\,.$$

Since $\pi_T\widehat{R} \subseteq \mathcal{D}^{\mathrm{T}}$ the number of terms in the union is no more than $|\mathcal{D}^{\mathrm{T}}| = |\mathcal{D}|^{|\mathrm{T}|}$.

Conversely, suppose R is a union of Cartesian products over Λ_i with no more than $|\mathcal{D}|^{|\mathrm{T}|}$ terms. Select a distinct $a \in \mathcal{D}^{\mathrm{T}}$ for each term and label its term R^a. Then define

$$\widehat{R} := \bigcup_a \{(a, x) \,|\, x \in R^a\}.$$

By construction, $R = \bigcup_a R^a$, and since R^a are Cartesian products over Λ_i we have $\mathrm{T} \twoheadrightarrow \Lambda_1 \cup \cdots \cup \Lambda_m$. Now Fagin's theorem gives the desired decomposition of $\widehat{R}$. $\square$

Theorem 11 underscores a close relationship between projection and union, in predicate terms, between existential quantification and disjunction. One can represent existentially quantified formula as a disjunction by moving quantified variables into an index, as in $\exists t R(t, x) = \bigvee_t R^t(x)$. For finite relations, existential quantification can be converted into disjunctions completely. However, there are two limitations on such disjunctions. First, the number of disjuncts is limited by $|\mathcal{D}|^k$, where k is the number of bound variables used, and second, the arity of disjuncts is then increased by k. As a result, if $k \geq n$ is needed to get the number of disjuncts under $|\mathcal{D}|^k$ then the disjunction does not convert into a projoin reduction.

On the other hand, allowing disjunctions/unions without restrictions completely trivializes the reduction problem. Any relation is a union of its tuples, and each tuple is the unary Cartesian product of singletons containing its members, i.e. $R = \bigcup_{a\in R} \{a_1\} \times$

$\cdots \times \{a_n\}$. Even if only finite unions are allowed every relation on a finite domain will 'reduce' to unary Cartesian products. However, in principle, it may be of interest to explore unions with restrictions other than those imposed by the existential quantifier, e.g. with bounds on the number of terms independent of the size of the domain.

Example 6 may suggest that non-identities $\neg I_n$ are projoin irreducible on finite domains. Indeed, $|\neg I_n| = |\mathcal{D}|^n - |\mathcal{D}| > |\mathcal{D}|^{n-2}$ for $n \geq 3$ and $|\mathcal{D}| \geq 2$, so hypostatic abstraction cannot reduce them. We will now exploit the relationship between projections and unions to show that this is not the case when $\mathcal{D}$ is sufficiently large. But that requires somewhat more general projoins than Fagin-type decompositions (17)-(18), namely, unions of joins that are not Cartesian products.

Example 7. Recall the join reduction of I_n from Example 3. Negating and applying de Morgan's law, we get

$$\neg I_n(x_1, \dots, x_n) = \neg I_2(x_1, x_n) \vee \cdots \vee \neg I_2(x_{n-1}, x_n) = \bigvee_{j=1}^{n-1} \neg I_2(x_j, x_n).$$

We cannot convert this disjunction into an existentially quantified formula because the attribute sets in each disjunct are different, but we can combine all of them into a single cover $\Lambda_i := \{i, n\}$ for $i = 1, \dots, n-1$. Then we define

$$R_i^j(x, y) := \begin{cases} \neg I_2(x, y),\ i = j \\ 1,\ i \neq j, \end{cases}$$

so that

$$\bigwedge_{i=1}^{n-1} R_i^j(x_i, x_n) = R_j^j(x_j, x_n) = \neg I_2(x_j, x_n),$$

and

$$\neg I_n(x_1, \dots, x_n) = \bigvee_{j=1}^{n-1} \bigwedge_{i=1}^{n-1} R_i^j(x_i, x_n).$$

This disjunction is already amenable to "existentialization". Assign a distinct $\alpha(j) \in \mathcal{D}$ to each disjunct, which requires $|\mathcal{D}| \geq n-1$, and set

$$R_i(t, x, y) := \begin{cases} R_i^j(x, y),\ t = \alpha(j) \\ 0,\ t \neq \alpha(j), \end{cases} \quad = \begin{cases} \neg I_2(x, y),\ t = \alpha(j),\ i = j \\ 1,\ t = \alpha(j),\ i \neq j \\ 0,\ t \neq \alpha(j). \end{cases}$$

Then the disjunction converts into a one parameter projoin of ternaries:

$$\neg I_n(x_1, \dots, x_n) = \exists t \left[\bigwedge_{i=1}^{n-1} R_i(t, x_i, x_n) \right]. \tag{19}$$

Thus, for $n \geq 4$ and $|\mathcal{D}| \geq n-1$ the non-n-identity is projoin reducible with a single parameter. A similar construction works for non-n-diversity relation based on the join reduction (10), but the condition is instead $|\mathcal{D}| \geq \frac{n^2-n}{2}$.

Going back to $\neg I_n$, the required size of the domain can be traded for arity. If we use pairs of elements to index the disjuncts then only $|\mathcal{D}|^2 \geq n-1$ is required, and similarly $|\mathcal{D}|^k \geq n-1$ if we use k-tuples. But we must have $k+2 \leq n-1$ so that it is still a reduction. Taking $k = n-3$ and noticing that $(n-1)^{\frac{1}{n-3}} \leq 2$ for $n \geq 5$ we conclude that $\neg I_n$ is reducible for $n \geq 5$ on any $\mathcal{D}$ with $|\mathcal{D}| \geq 2$ (albeit not necessarily to ternaries).

The construction in the above example can be generalized to prove the following theorem.

Theorem 12. *Suppose an n-ary R is join reducible to N factors of arity at most l, and $N \leq |\mathcal{D}|^{n-l-1}$. Then $\neg R$ is projoin reducible. In particular, if R has a k-key, with $k \leq n-4$ for $|\mathcal{D}| = 2$ and $k \leq n-3$ for $|\mathcal{D}| \geq 3$, then $\neg R$ is projoin reducible.*

Proof. As in Example (7), we present $\neg R$ as a disjunction and then convert it into a projoin. To get enough tuples for N disjuncts we need $N \leq |\mathcal{D}|^k$, where k is the number of projoin parameters, and $k + l \leq n-1$ so that the converted factors still have lower arity than R.

For the second claim, apply Theorem 5 to obtain a join reduction of R with $N = n-k$ and $l = k+1$, and note that $N \leq |\mathcal{D}|^{n-l-1}$ becomes $n-k \leq |\mathcal{D}|^{n-k-2}$. By calculus, $x \leq 2^{x-2}$ for $x \geq 4$ and $x \leq d^{x-2}$ for $x \geq 3$ when $d \geq 3$. □

The next example gives a taste of intricacies involved in ruling out general projoin reductions to prove unconditional projoin irreducibility.

Example 8. Consider reducing $\neg I_3$ on a domain of size $|\mathcal{D}| = d$ with *two* parameters. As in Example 6, the ansatz reduces to

$$\neg I_3(x_1, x_2, x_3) = \exists t_1 \exists t_2 \left[A(t_1, t_2) \wedge \bigwedge_{i=1}^{3} \Big(P^i(t_1, x_1) \wedge Q^i(t_2, x_2) \Big) \right].$$

Replacing $t_1, t_2 \in \mathcal{D}$ by indices $j, k \in \mathbb{N}_d$ we transform it into a disjunction

$$\neg I_3(x_1, x_2, x_3) = \bigvee_{j,k=1}^{d} a_{jk} \wedge \left[\bigwedge_{i=1}^{3} \Big(P^i_j(x_i) \wedge Q^i_k(x_i) \Big) \right].$$

Since unaries represent subsets of $\mathcal{D}$, conjunctions with different x_i their Cartesian products, and with the same x_i their intersections we obtain a union of unary Cartesian products

$$\neg I_3 = \bigcup_{j,k=1}^{d} a_{jk} \left[\bigtimes_{i=1}^{3} \Big(P^i_j \cap Q^i_k \Big) \right],$$

where $a_{jk} = 0, 1$ determines whether the term is included into the union. In contrast to Example 6, we can use up to d^2 terms in the union, but, as a tradeoff, the unary factors are not independent but must form a subset-valued rank 1 Boolean matrix $R^i_{jk} = P^i_j \cap Q^i_k$ for each i. Not only do we have to check whether $\neg I_3$ splits into a union of up to d^2 unary Cartesian products, but also whether Boolean matrices R^i_{jk} of their factors simultaneously factorize into outer products of Boolean vectors.

The Boolean matrix factorization problem is quite involved even for 0-1 matrices [32], and with more parameters one would have to deal with simultaneous factorization of even more interdependent Boolean tensors. We can now observe the progression from a simple cardinality condition for key reductions (Theorem 9), to finding union decompositions with independent Cartesian factors for multikey reductions (Theorem 11), and to finding such decompositions with intricate tensor factorizations for general projoin reductions with many parameters. This should dispel the initial impression that with 'enough' parameters every relation 'clearly should be' projoin reducible. And it suggests a fruitful connection between reduction of relations and factorization of Boolean tensors, an active area of research in modern data science [31].

6 Bonds and teridentity

In this section we will connect the theory of projoin reductions motivated by the database theory to the older theory of C.S. Peirce that supported his once controversial reduction thesis. While the main ideas are scattered in Peirce's writings, they did not gain currency until the formalizations by Herzberger [17] and Burch [8]. For a modern mathematical approach see [13, 14].

Unlike Cartesian products and joins, projoin is not a single operation on attributed relations. Its definition additionally depends on the set of projected attributes. One may feel that this is too permissive and/or clumsy. A natural way to specify projected attributes intrinsically is to choose all and only those that are not shared by the factors, i.e. to project out the shared attributes. If we think of joining as selecting tuples that match on shared attributes and splicing them together then leaving out the matched parts can give a meaningful response to a query.

Let us call such special projoins *pure*. Purity is a significant restriction on the operation: while the Fagin-type projoins (17)-(18) are pure, the reduction (19) we constructed for $\neg I_n$ is not. In the case of two factors, the pure projoin is what Peirce called *relative product* ("relative" was his term for relation) [7]. For binary relations, functions in particular, it is simply their composition: $\exists y\,[P(x,y) \wedge Q(y,z)]$. Peirce considered the relative product more basic than joins and projections through which we defined it, and preferred to reverse the order of definitions.

Like Cartesian and Boolean products, the relative product is a commutative binary operation on attributed relations. This is because the positions with the quantified variable are determined by the shared attribute, not by the order of factors, so commutativity reflects the commutativity of conjunction. However, unlike the other two, the relative product is not associative. Consider binary relations $P(u,x), Q(u,y), R(u,z)$. Associating the first two first gives $\exists t\,[P(t,x) \wedge Q(t,y) \wedge R(u,z)]$, but the last two first gives $\exists t\,[P(u,x) \wedge Q(t,y) \wedge R(t,z)]$. And $\exists t\,[P(t,x) \wedge Q(t,y) \wedge R(t,z)]$ cannot be generated by relative products at all, so not even all pure projoins are generated. This is because identified variables (shared attributes) in joins remain free, and it does not matter whether we identify two of them at a time or more, we can repeat the exercise when iterating. But in the relative product identified variables are quantified over (projected out), and no new variable can be identified with them afterwards.

In other words, in iterated relative products no attribute can be shared by more than two factors, and relative product is associative when restricted to triples of relations satisfying this condition. This is a further restriction on admissible projoins that restricts even relation schemes of factors that can appear in them. We will adopt Herzberger's term bond for this restricted class of projoins, although our bond is slightly more permissive than his along the lines adopted in [14].

Definition 7. *A collection of attributed relations $R_i \subseteq \mathcal{D}^{\Lambda_i}$ is called* ***bondable*** *when no three of Λ_i share an attribute. Their* ***bond*** *is then the projective join with all the shared attributes projected out, i.e. the projected set is $\Gamma := \cup_i \Lambda_i \setminus (\cup_{i \neq j} \Lambda_i \cap \Lambda_j)$. A relation $R \in \mathcal{D}^n$ is a* ***bond over a cover*** *$\mathrm{T} \cup \Sigma = \Lambda_1 \cup \cdots \cup \Lambda_m$ with $\mathrm{T} \cap \Sigma = \emptyset$ when it is a bond of some $R^{\Lambda_i} \subseteq \mathcal{D}^{\Lambda_i}$, called its* ***bond factors****, with elements of T called its (bond)* ***parameters****. R is called* ***bond reducible*** *when it is a bond with $0 < |\Lambda_i| < |\Sigma|$.*

Note that we allow $\mathrm{T} = \emptyset$, and so, in contrast to Peirce and Herzberger, Cartesian products are bonds, with 0 parameters. This makes bonds a generalization of Cartesian products alternative to joins, and simplifies some formulations. In terms of predicates, our definition means that the sets of free variables in different bond factors are disjoint, and every bound variable is present in exactly two factors. So the projoins $\exists t\,[P(x,t) \wedge Q(x,t)]$, $\exists t\,[P(x,t) \wedge Q(y,z)]$ are not bonds, and neither are hypostatic abstractions like (12).

We will now show, following Peirce and Burch [8, 9], that, somewhat surprisingly, this severely restricted operation leads to essentially the same notion of reducibility as general projoins. Peirce's first observation was that multiple variable identifications bond reduce to pairwise ones by using identity predicates, e.g.

$$R_1(t, x_{\Lambda_1}) \wedge \cdots \wedge R_n(t, x_{\Lambda_n}) = \\ \exists t_1 \ldots \exists t_n\,[I_{n+1}(t, t_1, \ldots, t_n) \wedge R_1(t_1, x_{\Lambda_1}) \wedge \cdots \wedge R_n(t_n, x_{\Lambda_n})]\,. \quad (20)$$

And his second observation was that n-identities I_n for $n \geq 4$ bond reduce to teridentities I_3:

$$I_n(x_1, \ldots, x_n) = \\ \exists t_1 \ldots \exists t_{n-3}\,[I_3(x_1, x_2, t_1) \wedge I_3(t_1, x_3, t_2) \wedge \cdots \wedge I_3(t_{n-3}, x_{n-1}, x_n)]\,. \quad (21)$$

Projoins with quantification into a single position can also be reduced to bonds by replacing the factors with the quantified variables by factors of lower arity, as in $\widetilde{P}(x_\Lambda) := \exists t P(x_\Lambda, t)$. Applying the above identities converts any projoin into a bond of the original factors (up to renaming of attributes), their projections, and teridentities. Let us call this conversion *bond explication.* For example, the bond explication of $\exists\, t\,[P(t, x_1, x_2, t) \wedge \exists\, s\, Q(x_2, x_3, s, t)]$ is

$$\exists\, t_1 \exists\, t_2 \exists\, t_3\, \exists\, y_1 \exists\, y_2 \big[I_3(t_1, t_2, t_3) \wedge I_3(y_1, x_2, y_2) \wedge P(t_1, x_1, y_1, t_2) \\ \wedge \exists\, s\,[I_1(s) \wedge Q(y_2, x_3, s, t_3)]\big]. \quad (22)$$

Since projection does not increase arity, and explication can add only ternaries, we have the following theorem.

Theorem 13. *An n-ary with $n \geq 4$ is projoin reducible if and only if it is bond reducible. A ternary is projoin reducible if and only if it decomposes into a bond of unaries, binaries and teridentities.*

Proof. The only claim not covered by bond explication is that ternaries that are bonds of unaries, binaries and teridentities are projoin reducible. Given such a bond, assign a new variable to each teridentity and replace by it all occurrences of the original variables from the teridentity. Then remove the teridentity and the quantifiers over its variables. By (20), this produces an equivalent expression, and, since all teridentities are removed, it is a projoin of unaries and binaries only. □

In particular, the projoin reduction (19) of $\neg I_n$ can be transformed not just into a pure one, but even into a bond reduction. Note that hypostatic abstraction (15) with $k = 1$, when it applies, decomposes any ternary into a bond of binaries and teridentities. This gives us a bond analog of Corollary 1.

Theorem 14 (**Peirce's reduction thesis**). *Any n-ary with $n \geq 3$ on an infinite domain reduces to a bond of unaries, binaries and teridentities. The only bond irreducible relations on such domains are all unaries, non-degenerate binaries, and non-degenerate ternaries.*

The first, reducibility, clause of the thesis is a direct consequence of Corollary 1 and Theorem 13. Indeed, it is essentially equivalent to projoin reducibility to binaries alone. Ironically, Löwenheim's result [27] to that effect once made it controversial due to the confusion between these closely related notions of reducibility [9, 21].

The second, irreducibility, clause, as applied to unaries and binaries, also follows trivially. Since bond is a special case of projoin, and they are projoin irreducible, they are all the more bond irreducible. However, the part concerning ternaries is non-trivial. It will follow from a graph-theoretic argument in Section 9 (Theorem 16).

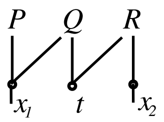
While bond and projoin reducibilities are (almost) the same, bond reductions are much more special than general projoin reductions. Peirce felt that general projoin reductions conceal the complexity involved in attribute matching (variable identifications), and, as a result, do not provide "true" or "complete" analysis of a relation delivered by bonds [7, 21].

7 Bonding diagrams

In this section we introduce graphical representation of joins, projoins and bonds which pictures the structure of relations by analogy to diagrams of chemical decompositions of compounds into elements. It also allows to bring in graph-theoretic methods into analysis of reductions. The diagrams we describe are simplified versions of Peirce's existential graphs [14] covering only a fragment of predicate logic (conjunction and existential quantifier) without the associated graphical calculus. We use some standard notation and terminology from graph theory [4] throughout this and the following sections.

Definition 8. *The* ***projoin graph*** *is a labeled bipartite graph with vertices for each factor and each attribute of the projoin. An edge joins them when the attribute is in the relation scheme of the factor. The vertices are labeled by relation and attribute names, and sorted into* ***predicate vertices*** *(for factors),* ***free attribute vertices*** *(for projected attributes) and* ***bound attribute vertices*** *(for projected out attributes). The* ***valency*** *of a vertex is its graph-theoretic degree (the number of incident edges) for predicate and bound attribute vertices, and the graph-theoretic degree increased by* 1 *for free attribute vertices.*

In practice, we depict predicate vertices as predicate letters and attribute vertices as dots labeled by variables. This makes it easier to associate graphs to predicate formulas, and, since only matching of attributes matters in decompositions rather than their proper names, variable labels work well enough for our purposes. Free attribute vertices are additionally labeled by an extra stem (hanging edge) coming out of them, which

a) 

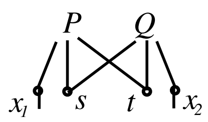
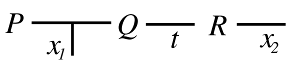
b) 

Figure 1: Projoin graphs for a) $\exists\, t[P(x_1) \wedge Q(x_1,t) \wedge R(t,x_2)]$; b) $\exists\, s \exists\, t[P(x_1,s,t) \wedge Q(s,t,x_2)]$.

explains the valency convention. Examples of projoin graphs and the corresponding predicate formulas are shown on Figure 1.

Projoin graphs can get quite cluttered and are made somewhat more readable by converting them into bonding diagrams defined below. These diagrams are also better equipped to depict bonds.

Definition 9. *The* ***bonding diagram*** *is obtained from the projoin graph by replacing each bivalent bound vertex by an edge connecting the corresponding predicate vertices, and replacing each bivalent free vertex by a hanging edge from the corresponding predicate vertex. The new edges carry the attribute labels of the removed vertices, and the labels of free attribute vertices are moved to their stems. We call the remaining attribute vertices of valency greater than* 1 ***branch points****, and of valency* 1 ***dead ends****. Hanging edges, incident to predicate vertices and branch points, are called* ***loose ends****. Bonding diagrams of bonds are called* ***bond diagrams****.*

a) 

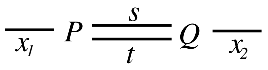
b) 

Figure 2: Bonding diagrams of projoins from Figure 1.

Loose ends correspond to free variables, and dead ends to bound variables occurring in a single position. Branch points are the device for identifying variables in different predicates – all edges attached to a branch point carry the same variable. The variable is free when one of the attached edges is a loose end, as in the T-shaped link on Figure 2 a), otherwise it is existentially quantified, as t on Figure 3 a). When a variable appears in only two predicates, no branch point is necessary, a simple edge connecting them suffices. Thus, bonding diagrams of bonds are graphically distinguished by having no branch points. When two or more variables appear in the same two predicates, the diagram displays a multiedge connecting their vertices, as in Figure 2 b).

Note that if a bonding diagram has disconnected subdiagrams then the relations they represent are Cartesian factors of the original, except for the case when they have no free attributes, i.e. are closed formulas. If they are true all predicates in them can be dropped without any loss, and if false the factored relation is itself empty. From now on we will only consider projoins without such redundant predicates and call them *non-redundant.*

Corollary 2. *If a non-redundant bonding diagram of a relation is disconnected then the relation is degenerate. The connected components are bonding diagrams of its Cartesian factors.*

The converse is false for the trivial reason that one can use degenerate predicates in a reduction. But if a relation is degenerate it *admits* reductions with disconnected and non-redundant bonding diagrams.

We intentionally used a two-step definition instead of defining bonding diagrams directly to emphasize the singling out of bivalent vertices (pairwise attribute identifications), which highlights binary bonding, i.e. relative products. A bond diagram will have no attribute vertices, and all its attribute labels will attach to edges, including hanging edges.

Our next observation is that bond explication also has a simple graphical interpretation in terms of bonding diagrams.

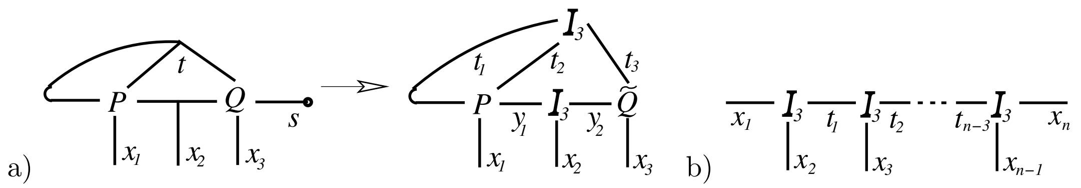


Figure 3: a) Bond explication of $\exists\, t\, [P(t, x_1, x_2, t) \wedge \exists\, s\, Q(x_2, x_3, s, t)]$ as (22), here $\widetilde{Q}(t_3, y_2, x_3) := \exists s\, Q(t_3, y_2, x_3, s)$; b) bonding diagram of the reduction (21) of n-identity to teridentities.

Corollary 3. *The bonding diagram of a bond explicated projoin is obtained from its original bonding diagram by absorbing dead ends into the adjacent predicate vertices, and replacing n-valent branch points by the diagrams of bond reductions of n-identities to teridentities (Figure 3).*

It is particularly pronounced in the diagrams that attribute identifications (branch points) function like hidden factors in projoin reductions. Indeed, nothing substantive distinguishes them from predicate vertices in assembling the relation. Thus, one can see bond explication as analogous to adjoining "ideal elements" (I_n) to uniformize factorizations in ring algebra.

8 Complete reductions and ternarity

So far we considered only general reductions, not complete reductions down to irreducibles. In this section we will start looking at their structure of their complete reductions, but, in the light of Peirce's reduction thesis, we consider only relations reducible to unaries, binaries and ternaries. They form a bond subalgebra of all relations, which is of interest even if irreducible higher arity relations exist on finite domains.

One can see from bonding diagrams that much work at putting a relation together is done by branch points. Hypostatic abstraction, for example, has a single branch point that holds together an otherwise loose collection of binaries. Bond explication removes branch points, but at a price of adding ternaries to the reduction. This suggests that ternaries play the role of relays in information exchange among the attributes, and their number quantifies the 'complexity of relating attributes'. Mutual information between attributes has been studied in the context of database theory [29], and, more recently, as a measure of information integration in biological systems [36]. Another potential application is to designing conceptual schemas of databases friendly to natural language and human representation of knowledge and reasoning, as in Sowa's conceptual graphs that are based on bonding diagrams [11]. For more motivation and further discussion we refer to [21].

Thus, we will be interested in counting the number of ternaries in complete reductions of a relation, assuming that it is reducible to unaries, binaries and ternaries. Of course, this number may vary from one reduction to another, and what we really want is the *minimal* number over all possible reductions.

Definition 10. *A bond is called* ***subternaric*** *when all of its factors have arity at most* 3. ***Ternarity*** *of a relation, denoted* $\mathbf{ter}$ *, is the minimal number of ternaries in its subternaric bond reductions, and* ∞ *if no such reductions exist. Non-redundant bond reductions with* $\mathbf{ter}$ *ternaries will be called* ***minimal bond reductions****.*

Ternarity of unaries and binaries is obviously 0, and of ternaries is at most 1. By the reducibility clause of Peirce's reduction thesis (Theorem 14), all relations on infinite domains have subternaric bond reductions. Whether this holds on finite domains, i.e. whether $\mathbf{ter}\,(R) < \infty$ for all R, is an open problem, equivalent to projoin reducibility of all relations to binaries by Theorem 13.

The following lemma is a direct consequence of the definitions.

Lemma 2. *Ternarity is subadditive on relative products. For a projoin R with factors R_i, free attributes indexed by j with the j-th shared by m_j factors, and bound attributes indexed by k with the k-th shared by n_k factors,*

$$\mathbf{ter}\,(R) \leq \sum_i \mathbf{ter}\,(R_i) + \sum_j (m_j - 1) + \sum_k (n_k - 2). \tag{23}$$

Proof. Subadditivity is obvious because bonding two bond diagrams does not add predicate vertices or branch points. In the projoin graph of R, aside from R_i vertices, we have free branch points of valencies $m_j + 1$, and bound branch points of valencies n_k. According to (20)-(21), bond explication replaces the former with m_j+1-2 teridentities and the latter with $n_k - 2$ teridentities, hence the sum in (23). □

Bounds on ternarity from above can be obtained from constructions of bond reductions. Recall that any projoin reduction can be explicated into bond reduction, and one can obtain projoin reductions by using keys (Theorem 9). However, only 1- or 2-keys produce subternaric reductions because for a k-key the factors have arity $k + 1$.

Theorem 15. *If an n-ary R admits a 1-key then* $\mathbf{ter}\,(R) \leq n - 2$*, if it admits a 2-key then* $\mathbf{ter}\,(R) \leq 3n - 4$*, and if it already has a 2-key then* $\mathbf{ter}\,(R) \leq 3n - 8$*.*

Proof. Hypostatic abstraction (15) with $k = 1$ reduces R to a projoin of n binaries with a single shared attribute, which is bound and shared by all n factors. Therefore, the first two terms in (23) vanish and the last one produces $n - 2$. In the case of $k = 2$ we obtain a projoin of n ternaries with two bound shared attributes, each by all n factors. Therefore, the right hand side of (23) reduces to $n + 2(n - 2) = 3n - 4$. When R has a 2-key, there are only $n - 2$ factors, and the shared attributes are now free, so the count changes to $n - 2 + 2(n - 2 - 1) = 3n - 8$. □

It is interesting that there is no drop in ternarity bound when the relation already has a 1-key as opposed to just admitting one. There is a change from projoin to join, but all the ternaries come from the branch point that has valency n in both cases. Other special constructions also provide upper bounds. For example, it follows from (19) that $\mathbf{ter}\,(\neg I_n) \leq 3n - 6$ for $n \geq 4$ and $|\mathcal{D}| \geq n - 1$. Bounds from below are conceptually harder because we have to rule out all bonds with fewer ternaries as reductions. We will obtain such a bound for non-degenerate relations in the next section by exploiting graph-theoretic properties of bond diagrams.

Upon reflection, absence of reducible factors in a reduction is too weak a property. Not only can complete reductions have redundant predicates, as long as those are irreducible, but they may not be minimal. For example, four teridentities bonded in a square reduce I_4 to irreducibles, but hypostatic abstraction gives a minimal reduction with only two teridentities.

On the other hand, minimal reductions can be incomplete for only trivial reasons. And they can be converted into complete reductions by a trimming procedure that is reflected in diagrams by *merging* of predicate vertices. In algebraic terms, when two factors share bound attribute(s) we replace them in the bond by their relative product. Merging cannot be used on a pair of ternaries with a single shared attribute, because it creates a quaternary, but in all other cases the bond remains subternaric. For example, the bond on Figure 2 b) can be merged into a single binary. The next lemma uses merging to produce complete minimal reductions, and gives additional support to discounting unaries and binaries in ternarity counts.

Lemma 3. *Let R be a subternarily reducible relation.*
(i) *In any minimal reduction of R any two factors share at most one attribute, i.e. the bonding diagram has no multiedges.*
(ii) *If R does not have an unary Cartesian factor then its minimal reductions have no unaries at all.*
(iii) *If R does not have a binary Cartesian factor then there exist its minimal reductions with no binaries at all.*

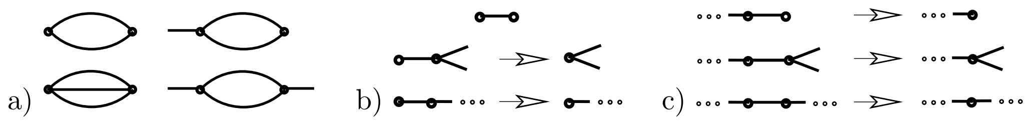


Figure 4: Schematic bond diagrams, the dots stand for predicate vertices: a) subcubic multiedges; b) merging unaries; c) merging binaries.

Proof. (i) Possible multiedge configurations are shown on Figure 4 a). Two of them have no loose ends and cannot occur because minimal reductions are non-redundant. In the other two merging would eliminate a ternary, so they cannot occur by minimality.

(ii) If the unary shares its attribute with another factor there are three cases. It is another unary and they form a redundant component, which is ruled out. It is a ternary and merging will turn it into a binary, which is also ruled out. Finally, if it is a binary then merging will reduce it to an unary, and we can repeat the process, Figure 4 b). By induction on the number of binaries, it must stop, and it can only stop when the unary's attribute is free (we hit a loose end). But then R has an unary Cartesian factor, contrary to the assumption.

(iii) By (i), a binary can share at most one attribute with another factor, and when it does, merging reduces the number of binaries, Figure 4 c). By induction, all binaries can be eliminated except for those with both attributes free, i.e. binary Cartesian factors. □

After applying the Lemma's procedure, the only reducible factors left, if any, are degenerate binary Cartesian factors. Factoring them into pairs of unary factors produces a complete minimal reduction.

9 Ternarity and Peirce's reduction thesis

In this section we will use ternarity and graph theory to refine Peirce's reduction thesis on infinite domains, and show that its strengthened form fails dramatically on finite domains.

Definition 11. *The* ***bond graph*** *is obtained from the bond diagram by placing additional vertices at the loose ends and removing the labels.*

The bond graph is just a multigraph of graph theory [4]. If the bond was subternaric then the multigraph will be *subcubic* (subtrivalent), i.e. have vertices of degree at most 3. Such graphs are widely studied, particularly due to applications in structural chemistry and knot theory. By Lemma 3, when the bond is a minimal reduction of a relation without unary Cartesian factors, the bond graph is a simple graph and its vertices of degree 1, called *pendants* in graph theory, are in 1-1 correspondence with the relation's attributes.

To prove the next lemma, we will need a graph-theoretic formula originally due to Listing [26]. Let V be the number of vertices, E the number of edges, C the number of fundamental cycles, and K the number of connected components, then $V - E + C - K = 0$. This formula is often used in modern graph theory as the definition of C, called the cyclomatic number, which is then proved to be equal to the number of fundamental cycles [4].

Lemma 4. *If I, II, III denote the numbers of vertices of degrees $1, 2, 3$, respectively, in a subcubic multigraph then $III - I = 2(C - K)$. In particular, I and III have the same parity. If the graph is connected with at least n pendants then $III \geq n - 2$.*

Proof. In a subcubic multigraph we have $I + II + III = V$. And, by the handshaking theorem, the sum of all vertex valencies is twice the number of its edges, so $I + 2II + 3III = 2E$. Therefore, $V - E = \frac{1}{2}(I - III)$. It remains to multiply both sides by 2 and note that $V - E = -(C - K)$ by the Listing's formula. In a connected multigraph $K = 1$, so $III = I - 2 + 2C \geq n - 2$. □

It is instructive to give a direct intuitive argument for the case $n = 3$. Since the multigraph is connected there exist paths going from two of the pendants to the third. They must meet at some vertex, and then proceed jointly to the destination (of course, they may meet and diverge several times). That meeting vertex must have degree 3.

Corollary 4. *If R does not have an unary Cartesian factor then its ternarity and arity have the same parity.*

Proof. We have $\mathbf{ter}\,(R) = III$ in the bond graph of a minimal bond reduction of R. By Lemma 3 (ii), it contains no unaries, so the only pendants come from loose ends, and I is the arity of R. The result now follows directly from Lemma 4. □

The no-unary-factor condition cannot be dropped, the quaternary $P(u) \wedge Q(x, y, z)$ with non-degenerate Q has arity 4 but ternarity 3.

Peirce's reduction thesis is essentially equivalent to $1 \leq \mathbf{ter}\,(R) < \infty$ for non-degenerate R with $n \geq 3$. The next theorem gives us the exact number, on infinite domains, and the promised irreducibility of non-degenerate ternaries, on any domains.

Theorem 16. *Ternarity of any non-degenerate n-ary R with $n \geq 2$ satisfies* $\mathbf{ter}\,(R) \geq n - 2$. *In particular, any non-degenerate ternary is bond irreducible. If, moreover, $|R| \leq |\mathcal{D}|$ then* $\mathbf{ter}\,(R) = n - 2$. *In particular,* $\mathbf{ter}\,(R) = n - 2$ *for all relations on infinite domains.*

Proof. Consider a bond decomposition of R. Since R is non-degenerate its bond multigraph is connected, and has at least n pendants coming from the loose ends, the free variables. Therefore, $I \geq n$. The lower bound $\mathbf{ter}\,(R) \geq n - 2$ now follows directly from Lemma 4. For $n = 3$ this means that any bond decomposition of a non-degenerate ternary must contain a ternary, i.e. such ternaries are bond irreducible.

If $|R| \leq |\mathcal{D}|$ then, by Lemma 1, it admits a 1-key, and, by Theorem 15, $\mathbf{ter}\,(R) \leq n - 2$, hence $\mathbf{ter}\,(R) = n - 2$. On infinite domains $|R| \leq |\mathcal{D}|$ holds for all relations. □

The next example shows that the inequality in the lower bound on **ter** can be strict on finite domains.

Example 9 (**Herzberger's quaternary**)**.** Consider the quaternary H on $\mathcal{D} = \{\alpha, \beta, \gamma\}$ introduced by Herzberger in [17] and given by the table below.

$$H := \begin{array}{|c|c|c|c|}\hline \alpha & \beta & \beta & \alpha \\ \hline \beta & \alpha & \alpha & \beta \\ \hline \gamma & \beta & \gamma & \beta \\ \hline \beta & \gamma & \beta & \gamma \\ \hline \end{array} \qquad H^{1,2,3} = \begin{array}{|c|c|c|}\hline \alpha & \beta & \beta \\ \hline \beta & \alpha & \alpha \\ \hline \gamma & \beta & \gamma \\ \hline \beta & \gamma & \beta \\ \hline \end{array} \qquad H^{1,2,4} = \begin{array}{|c|c|c|}\hline \alpha & \beta & \alpha \\ \hline \beta & \alpha & \beta \\ \hline \gamma & \beta & \beta \\ \hline \beta & \gamma & \gamma \\ \hline \end{array}$$

One can check by cases that H is non-degenerate. We have $|H| = 4 > 3 = |\mathcal{D}|$, and Herzberger shows by combinatorial search that H is, indeed, not a relative product of two ternaries (actually, he only shows that for one partition of attributes, but the argument works analogously for others). Therefore, $\mathbf{ter}\,(H) > 2$.

However, H is not a counterexample to reducibility. One can see by inspection that it has a 2-key (in fact, any two of its columns are a 2-key). Therefore, by Theorem 5, it is a join of two ternaries, e.g. $H = H^{1,2,3} \bowtie H^{1,2,4}$ if we pick the first two columns as the 2-key. Bond explication (20) of the two shared attributes converts this join into a bond of *four* ternaries, its projections $H^{1,2,3}$, $H^{1,2,4}$, and two teridentities. Since $\mathbf{ter}\,(H)$ must be even by Lemma 4 and $\mathbf{ter}\,(H) > 2$ we conclude that $\mathbf{ter}\,(H) = 4$.

Herzberger's observation can be strengthened by relating ternarity to the number of parameters in projoin reductions and applying Theorem 10. It turns out that linear bounds on ternarity in terms of arity, as in Theorem 15, are not typical for general relations. However, we cannot infer existence of relations of infinite ternarity, i.e. of irreducible n-aries with $n \geq 4$, and hence refute Peirce's original thesis.

Theorem 17. *The share of n-ary relations with $n \geq 4$ and* $\mathbf{ter}\,(R) \leq m$ *among all such relations on a domain $\mathcal{D}$ is < 1 for $|\mathcal{D}| > \binom{\frac{3m+n}{2}}{n-1}$, and asymptotically vanishes when $|\mathcal{D}| \to \infty$. In particular, there exist n-ary relations of arbitrarily high ternarity.*

Proof. Suppose R is non-degenerate with $\mathbf{ter}\,(R) \leq m$. By Lemma 3, there is a purely ternary minimal reduction of it. In its bond graph $I = n$, $III = \mathbf{ter}\,(R)$, and $2E = I + 3III$ by the handshaking theorem. Of the edges, n are incident to pendants and correspond to free variables, while $k := E - n$ correspond to bond parameters. Therefore, our minimal reduction is a projoin reduction with $k = \frac{3\,\mathbf{ter}\,(R)-n}{2} \leq \frac{3m-n}{2}$ parameters.

By Theorem 10, the share of relations projoin reducible with k parameters is < 1 when $|\mathcal{D}| > \binom{n+k}{n-1}$ and it goes to 0 when $|\mathcal{D}| \to \infty$. Since $n+k \leq \frac{3m+n}{2}$ this inequality is satisfied for our k. Degenerate relations are projoin reducible with even 0 parameters, let alone k, so there must be non-degenerate relations with $\mathbf{ter} > m$ on our domain. Moreover, their share approaches 1 when $|\mathcal{D}| \to \infty$. □

The estimate we used in the theorem is very rough. Indeed, k is not the number of parameters in just any projoin reduction, but in a complete reduction down to ternaries. One could merge ternaries, as long as the merged factors still have arity $< n$, and reduce that number. To get a more accurate estimate one can count the number of subcubic graphs with n pendants, m cubic and no degree 2 vertices, and bound the number of relations that have them as their bond graphs.

Finally, Peirce's reduction thesis on infinite domains and bond explication show that of all ternaries only one is needed in reductions – teridentity. It is to unaries, binaries and teridentities that we should aim to reduce all relations. This suggests our next definition.

Definition 12. ***I_3-ternarity*** *of a relation, denoted* $\mathbf{ter}_{I_3}$*, is the minimal number of teridentities in its subternaric bond reductions where the only ternaries are teridentities, and* ∞ *if no such reductions exist.*

Clearly, $\mathbf{ter}_{I_3} \leq \mathbf{ter}$ and $\mathbf{ter}_{I_3} = \mathbf{ter}$ on infinite domains because any ternary decomposes into binaries and teridentities by hypostatic abstraction. The next example shows that on finite domains, again, the inequality can be strict.

Example 10. Suppose a non-degenerate ternary has a subternaric decomposition with a single teridentity. Since it has no unary factors unaries can be eliminated, and we are left with the teridentity with up to three chains of binaries attached to it in the graph. A chain of binaries can be merged into a single one by relative products, and represent our ternary as a teridentity directly bonded with three (or fewer) binaries. Turning the teridentity into a branch point we obtain a projoin of binaries with a single parameter.

However, we showed in Example 6 that $\neg I_3$ on a domain with $|\mathcal{D}| = 2, 3$ is not projoin reducible with one parameter. Therefore, while $\mathbf{ter}\,(\neg I_3) = 1$ trivially, $\mathbf{ter}_{I_3}(\neg I_3) > 1$. Since $\mathbf{ter}_{I_3}(\neg I_3)$ must be odd we can conclude that $\mathbf{ter}_{I_3}(\neg I_3) \geq 3$, i.e. it takes at least 3 teridentities to bond $\neg I_3$ on small domains, if it is possible at all.

10 Conclusions and open problems

We studied reduction of relations to relations of smaller arity under three relational operations: join, projoin and bond. All three can be expressed by conjunctions and existential quantification on predicates, and are motivated by algebraic analogies and practical applications in the database theory. Aside from unifying and extending known reduction results and constructions, we described the sets of irreducible relations and the structure of complete reductions to them. We also clarified the relationship between projoin and bond reducibility, and the import of Peirce's reduction thesis. Finally, we introduced the notion of ternarity that, intuitively, measures complexity of 'relating' in a relation, and used it to sharpen reducibility results.

Aside from concrete results, a major takeaway from this work is the striking gap between reduction behavior on finite and infinite domains. As far as we know, the only author to notice the phenomenon before was Herzberger [17]. We showed that the gap gets wider as the size of the domain grows: the share of irreducible relations with bounded number of parameters (Theorem 10) or with bounded ternarity (Theorem 17), grows with it, even though it is 0 at ∞. The root cause of this discrepancy is the equality $|\mathcal{D}| = |\mathcal{D}|^2$ for infinite cardinalities, which is equivalent to the axiom of choice.

This raises a big question: to what extent does Peirce's reduction thesis hold on finite domains? While non-degenerate ternaries are still irreducible there (even by stronger means than bonds and projoins [14]), reduction is obstructed by the lack of enough domain elements for classical constructions.

Problem 1: Are there irreducible n-ary relations with $n \geq 4$?

Such relations would have to have a lot of tuples, $|R| > |\mathcal{D}|^{n-2}$. Otherwise, they will have a k-key with $k \leq n-2$ and hypostatic abstraction will reduce them (Theorem 9). Counting arguments we used would not settle the question alone, because general projoins and bonds do not have a finite combinatorial description like Cartesian products, joins, projoins with bounded number of parameters, or bonds with bounded ternarity. On the other hand, general tests of irreducibility, like the ones for join irreducibility in Theorem 7, also seem to be elusive. A promising approach is provided by the clone theory, where one can dualize the problem into one about functional clones via the Pol-Inv Galois connection. Projoin bases of small arity for maximal sub-co-clones of the co-clone of all relations on 2-element domains are constructed in [5]. If one could construct bases containing only unaries, binaries and ternaries for the co-clone of all relations on any finite domain that would resolve the question negatively.

While we suspect a negative answer to the first problem, it is more likely to be affirmative for the next one. Although non-degenerate ternaries are trivially 'reducible' to themselves on any domains, there is a non-trivial reducibility question about them. To resolve it negatively, one would need a basis of the co-clone of all relations containing only unaries and binaries.

Problem 2: Are there ternary relations indecomposable into bonds of unaries, binaries and teridentities (equivalently, projoin irreducible to unaries and binaries)?

We already saw that the strongest form of the reduction thesis, that gives ternarity of non-degenerate n-ary relations as $n-2$, fails on finite domains. Even if reductions are always possible their complexity must be higher than that of their infinite counterparts. In particular, there can be no bound on ternarity in terms of arity alone. However, since there are finitely many n-aric relations of finite ternarity on a finite domain ternarity must attain a maximum on them.

Problem 3: Find sharp upper bounds on ternarity in terms of arity and the size of the domain.

We did not address the question of uniqueness, but a relation can have purely ternaric minimal reductions whose bond graphs are not even isomorphic. Indeed, bonding teridentities on any cubic graph with n hanging edges produces n-identity, and it is easy to construct non-isomorphic tree graphs with equal numbers of cubic vertices. Perhaps, this diversity is due to overabundance of symmetry in I_n.

> **Problem 4:** Are the bond graphs of purely ternaric minimal reductions unique for 'generic' non-degenerate relations?

One can think of minimal reductions as revealing the structure of information processing within a relation, which suggests an affirmative answer. Attributes of a relation generalize inputs and outputs of a function, and functions are commonly interpreted as information processors [23, 1.5.4]. In [29] a measure of information exchange among relation's attributes is introduced and studied, similar measures are studied in computational biology [36]. The intuition of ternarity as 'complexity of relating attributes' suggests a connection.

> **Problem 5:** Is there an information-theoretic interpretation of ternarity, e.g. bounds on measures of information exchange in terms of it?

To summarize, logical factorization of relations poses many interesting challenges at the intersection of mathematical logic, combinatorics, graph theory and data science.

References

[1] S. Abiteboul, R. Hull, V. Vianu, *Foundations of databases*, Addison-Wesley, New York, 1995.

[2] A. Aho, C. Beeri, J. Ullman, Theory of joins in relational databases, ACM Transactions on Database Systems, 4 (1979) no. 3, 297-314.

[3] C. Beeri, R. Fagin, D. Maier, A. Mendelzon, J. Ullman, Properties of acyclic database schemes, in *Proceedings of the 13th annual ACM Symposium on Theory of Computing, Milwaukee, May 11-13*, 1981, 355-362.

[4] C. Berge, *The theory of graphs*, Dover, Mineola, NY, 2001.

[5] E. Böhler, S. Reith, H. Schnoor, H. Vollmer, Bases for Boolean co-clones, Information Processing Letters, 96 (2005) no. 2, 59-66.

[6] F. Börner, Basics of Galois connections, in *Complexity of Constraints. Lecture Notes in Computer Science, v. 5250, Springer-Verlag, Berlin*, 2008, 38-67.

[7] J. Brunning, C. S. Peirce's relative product, Modern Logic, 2 (1991) no. 1, 33-49.

[8] R. Burch, *A Peircean reduction thesis: the foundations of topological logic*, Texas Tech University Press, Lubbock, TX, 1991.

[9] R. Burch, Peirce's reduction thesis, in *Studies in the Logic of Charles Sanders Peirce, ch. 16, Indiana University Press*, 1997, 234-252.

[10] J. van den Bussche, Applications of Alfred Tarski's ideas in database theory, in *Computer Science Logic. Lecture Notes in Computer Science, v. 2142, Springer, Berlin*, 2001, 20-37.

[11] T. Cao, Conceptual graphs and fuzzy Logic, Springer, Berlin, 2010.

[12] E. Codd, A relational model for large shared data banks, Communications of the ACM, 13 (1979) no. 6, 377-387.

[13] J. Hereth Correia, F. Dau, Two instances of Peirce's reduction thesis, in *Formal Concept Analysis. Lecture Notes in Computer Science, v. 3874, Springer, Berlin*, 2006, 106-118.

[14] J. Hereth Correia, R. Pöschel, The power of Peircean Algebraic Logic (PAL), in *Concept Lattices, Second International Conference on Formal Concept Analysis, Springer, Berlin*, 2004, 337-351.

[15] I. Düntsch, Sz. Mikulas, Cylindric structures and dependencies in relational databases, Theoretical Computer Science, 269 (2001) no. 1-2, 451-468.

[16] R. Fagin, Multivalued dependencies and a new normal form for relational databases, ACM Transactions on Database Systems, 2 (1977) no. 3, 262-278.

[17] H. Herzberger, Peirce's remarkable theorem, in *Pragmatism and Purpose: Essays Presented to Thomas A. Goudge, University of Toronto Press*, 1981, 41-58.

[18] T. Imielinski, W. Lipski, The relational model of data and cylindric algebras, Journal of Computer and System Sciences, 28 (1984) no. 1, 80-102.

[19] T. Jech, *The axiom of choice*, Elsevier, New York, 1973.

[20] P. Jonsson, V. Lagerkvist, G. Nordh, B. Zanuttini, Complexity of SAT problems, clone theory and the exponential time hypothesis, in *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2013, 1264-1277.

[21] S. Koshkin, Is Peirce's reduction thesis gerrymandered? Transactions of the Charles S. Peirce Society, 58 (2022) no. 4, 271-300.

[22] V. Lagerkvist, M. Wahlström, The power of primitive positive definitions with polynomially many variables, Journal of Logic and Computation, 27 (2017) no. 5, 1465-1488.

[23] D. Lau, Function algebras on finite sets, Springer-Verlag, Berlin, 2006.

[24] T. Lee, An algebraic theory of relational databases, Bell System Technical Journal, 62 (1983) no. 10, 3159-3204.

[25] S. Lipschutz, *Set theory and related topics*, McGraw Hill, New York, 1998.

[26] J. Listing, Der Census räumlicher Complexe, der Verallgemeinerung des Euler'schen Satzes von den Polyedern, Abhandlungen der Königlichen Gesellschaft der Wissenschaften in Göttingen, 10 (1862) 97-182.

[27] L. Löwenheim, Über Möglichkeiten im Relativkalkül, Mathematische Annalen 76 (1915) 447-470. English translation: On possibilities in the calculus of relatives, in *From Frege to Gödel: a source book in mathematical logic 1879-1931, Harvard University Press, Cambridge, MS*, 1967, 228-251.

[28] D. Maier, *The theory of relational databases*, Computer Science Press, Rockville, MD, 1983.

[29] F. Malvestuto, Statistical treatment of the information content of a database, Information Systems, 11 (1986) no. 3, 211-223.

[30] A. Mendelzon, D. Maier, Generalized mutual dependencies and the decomposition of database relations, in *Fifth International Conference on Very Large Data Bases, IEEE*, 1979, 75-82.

[31] P. Miettinen, Boolean tensor factorizations, in *11th International Conference on Data Mining, IEEE, New York*, 2011, 447-456.

[32] P. Miettinen, Recent developments in Boolean matrix factorization, in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, Yokohama*, 2021, 4922-4928.

[33] D. Pöschel, L. Kalužnin, *Funktionen- und Relationenalgebren*, Mathematische Monographien 15, Deutscher Verlag der Wissenschaften, Berlin, 1979.

[34] J. Rissanen, Independent components of relations, ACM Transactions on Database Systems, 2 (1977) no. 4, 317-325.

[35] T. Schaefer, The complexity of satisfiability problems, in *Conference Record of the 10th Annual ACM Symposium on Theory of Computing, San Diego*, 1978, 216-226.

[36] M. Tegmark, Improved measures of integrated information, PLoS Computational Biology, 12(11) (2016) e1005123.

[37] S. Tringali, An abstract factorization theorem and some applications, Journal of Algebra, 602 (2022) 352-380.

[38] M. Yannakakis, C. Papadimitriou, Algebraic dependencies, Journal of Computer and System Sciences, 25 (1982) no.1, 2-41.