

# Functional completeness and primitive positive decomposition of relations on finite domains

Sergiy Koshkin

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

## Abstract

We give a new and elementary construction of primitive positive decomposition of higher arity relations into binary relations on finite domains. Such decompositions come up in applications to constraint satisfaction problems, clone theory and relational databases. The construction exploits functional completeness of 2-input functions in many-valued logic by interpreting relations as graphs of partially defined multivalued ‘functions’. The ‘functions’ are then composed from ordinary functions in the usual sense. The construction is computationally effective and relies on well-developed methods of functional decomposition, but reduces relations only to ternary relations. An additional construction then decomposes ternary into binary relations, also effectively, by converting certain disjunctions into existential quantifications. The result gives a uniform proof of Peirce’s reduction thesis on finite domains, and shows that the graph of any Sheffer function composes all relations there.

**Keywords:** relational operations, primitive positive formula, coclone, constraint satisfaction problem, irreducible relation, many-valued logic, functional completeness, Sheffer functions, Post algebra, relative product, hypostatic abstraction, Peirce’s reduction thesis

**MSC:** 08A02, 03G20, 08A70, 03B50, 08A40

## Introduction

First examples of primitive positive definitions appeared at the dawn of mathematical logic in de Morgan’s compositions (relative products) of binary relations, and were originally motivated by linguistic expressions like “brother of a parent”. The problem of decomposing higher arity relations into binary relations was studied soon after de Morgan by C.S. Peirce [6]. In a more formal context, primitive positive formulas (ppfs), those that apply only conjunctions and existential quantifiers to predicates, appeared in Robinson’s work on model theory in the 1950-s [23]. They came up independently in universal algebra in the 1960-s in the context of Pol-Inv Galois connection [5, 18], as preserving invariance of relations under a function.

Since the 1970-s, ppfs appeared in more practical contexts, as conjunctive queries in relational databases [16] and as templates for polynomial time reductions of constraint satisfaction problems (CSP) [10, 14, 30]. Schaefer exploited the latter in his proof of the dichotomy theorem for Boolean CSP [30], and ppfs are featured prominently in Jeavons’s algebraic approach to CSP since the 1990-s [7, 14]. In particular, a recent generalization of Schaefer’s CSP dichotomy to arbitrary finite domains relies on primitive positive reductions [2].

We will be mainly interested in the perspective from Peirce’s classical work, which was motivated, in part, by a functional analogy. It is a trivial observation that composition of 1-input functions does not produce functions with more inputs, while composition of 2-input functions can produce  $n$ -input functions with any  $n \geq 2$ . Let us call the functional reduction thesis (FRT) the two-part claim that  $n$ -input functions decompose into (reduce to) 2-input functions, but the latter do not all decompose into 1-input ones (are irreducible). While the irreducibility part is trivial, proving the reducibility part needs some work.

On Boolean domains, it follows from the existence of disjunctive normal forms, with only  $\wedge$ ,  $\vee$  and  $\neg$  composed, which are implicit in Peirce’s work of 1880-s and were explicitly derived by Schröder in 1890, see [19]. The case of arbitrary finite domains was settled in Post’s seminal 1920 dissertation, where the general question about functionally complete sets of functions was also raised, and solved in the Boolean case [32]. Post proved FRT by introducing a special class of many-valued logics with cyclic negation, and generalizing truth tables and disjunctive normal forms to them. The case of infinite domains was settled only in 1945 by Sierpiński [29], but, surprisingly, it turned out to be simpler.

Even before these results, Peirce suggested that such reductions transfer from functions to relations. Interpreting, say,  $f(t, z)$  and  $g(x, y)$  as relations,  $u = f(t, z)$  and  $t = g(x, y)$ , respectively, we can express their composition  $f(g(x, y), z)$  as

$$[u = f(g(x, y), z)] = \exists t [(u = f(t, z)) \wedge (t = g(x, y))], \quad (1)$$

a special case of the relative product of arbitrary relations, e.g.

$$R(x, y, z, u) = \exists t [S(x, y, t) \wedge T(t, z, u)]. \quad (2)$$

Compositions of more than two functions are generalized by expressions like the “triple junction” [12]:

$$R(x, y) = \exists t [F(t, x) \wedge G(t, y) \wedge H(t, z)]. \quad (3)$$

Using identity relations, one can also express conjunctions with identified variables, like  $I_3(x, y, z) := [(x = y) \wedge (y = z)]$ , i.e. joins of the relational database theory. The primitive positive composition of relations, pp-composition for short, is then seen as a natural extension of functional composition.

By analogy, let us call the relational reduction thesis (RRT) the two-part claim that  $n$ -ary relations for  $n \geq 2$  reduce to binary relations, but not all binary relations reduce to unary ones. This is equivalent to a part of Peirce’s reduction thesis formulated even before the more elementary FRT [15]. Peirce gave a clever general construction of reductions, which he called “hypostatic abstraction”, that, suitably modified, works on infinite domains. However, on finite domains, it requires adjoining additional elements and extending original relations to the larger domain, see Section 2.

One might expect that RRT on finite domains could be analogously derived from standard results on relational completeness, but such results are largely missing. For example, there are no general analogs of either Post’s or Slupecki’s criteria [26] for relationally complete systems. Nonetheless, RRT on finite domains has been resolved by other means, but the results are scattered in the literature, often formulated in different terms and proved by non-elementary methods.

The most non-trivial reducibility part follows from an auxiliary construction of a pp-complete system of two binary relations in the seminal 1969 work of BKKR (at the end of part II) that introduced the Pol-Inv Galois connection [3]. One can also derive the result non-constructively from the Galois connection itself [22, 1.1.22]. That RRT *fails* on Boolean domains follows from another landmark, Schaefer’s 1978 paper that proved the dichotomy theorem for Boolean CSP [30]. Namely, Schaefer proved in passing that all binary relations on Boolean domains are bijunctive while some ternary relations are not, and bijunctivity is preserved by the pp-composition. However, none of the above or subsequent works spell out the result ([11] comes closest) or connect RRT to FRT.

Is there a connection? In a 1906 letter to James, Peirce declared that they are “one and the same”. Even making room for rhetorical flourishes, this is much too rash. Not all relations are functions, and pp-composition of graphs from graphs does not always come from composition of their functions. On infinite domains, hypostatic abstraction bypasses FRT altogether rather than builds on it, so it is unclear why FRT even suggests RRT. Note that the connection that is supposed to be at play here is an elementary one – treating graphs of *individual* functions *as* relations, see (1). It is quite distinct from the Galois connection between clones and coclones, which has a *class* of functions *preserving* a class of relations.

In this paper, we will establish the desired connection and give elementary proofs for all cases of RRT based on it. Unlike the BKKR construction, it provides a wide selection of pp-complete systems, rather than a single one, and capitalizes on well-developed methods of functional decomposition. In fact, graphs of functions in any functionally complete system form a relationally pp-complete system (Theorem 2). This is somewhat surprising because there are many non-graph relations, and they cannot always be pp-composed from graphs in general coclones. As a simple consequence, we obtain the existence of ‘Sheffer relations’ that alone form a pp-complete system. The graph of the Sheffer stroke is such a relation on Boolean domains. The proof is based on treating relations as multivalued partially defined ‘functions’ that are then pp-decomposed into ordinary 2-input functions, e.g. by using canonical normal forms in Post’s many-valued logics.

Conceptually, our construction explains why pp-reductions to ternary relations are always possible (assuming that reducibility of functions to 2-input functions is taken for granted). From the practical point of view, it is of interest because pp-composition is less intuitive than functional composition and computational methods for it are much less developed. Moreover, constructing convenient sets of relations that are pp-complete in coclones (their *bases*) has received much attention in the recent CSP literature [4, 17].

However, passing from functions to their graphs increases arity by 1, so existence of functionally complete systems of 2-input functions translates into reducibility of higher arity relations to *ternary*, not binary, relations. This is as it should be in general, given that pp-irreducible ternary relations do exist on Boolean domains ( $\neg I_3$  is one). As a supplement, we give another elementary construction that reduces cofinite relations

of small arity to binary relations (Theorem 3). In a sense, it is dual to hypostatic abstraction, and converts disjunctions of unary cosingletons into existential quantifications over certain binary predicates on non-Boolean domains. For completeness, we also give an elementary discussion of the Boolean case and relate pp-composition to Peirce’s original “bonding” that deviates from it on ternary relations.

The paper is organized as follows. In Section 2 we recall hypostatic abstraction that reduces higher arity relations to binary ones after extending the domain, and Löwenheim-Sierpiński’s pairing construction that does the same for functions, proving RRT and FRT on infinite domains. In Section 3 we interpret relations with a distinguished argument (called *relatives*) as multivalued partially defined ‘functions’ and reduce higher arity relations to ternary ones on arbitrary domains by exploiting FRT. Section 4 introduces Post’s logics and illustrates the algorithm for converting functional into relational reductions by our method. In Section 5 we give the ‘existentialization of disjunctions’ construction that reduces cofinite relations of small arity to binary ones on domains with at least three elements. In Section 6 we discuss bijunctive relations and give an elementary proof of pp-irreducibility of some ternary relations on Boolean domains. Section 7 introduces Peirce’s notion of bonding and his conversion of pp-reductions to bond reductions for arities 4 and higher by clever use of ternary identities. In the final section, we summarize our conclusions and state some open problems.

## 1 Preliminaries

We use the standard notation and terminology of set theory and predicate calculus. Relations are defined on a set  $\mathcal{D}$  called the **domain** and are subsets of its Cartesian powers  $\mathcal{D}^n := \mathcal{D} \times \dots \times \mathcal{D}$ . The cardinality of a set  $S$  is denoted  $|S|$ . As is well known,  $|\mathcal{D}^n| = |\mathcal{D}|^n$ . When  $R \subseteq \mathcal{D}^n$  the number  $n$  is called the relation’s **arity** and the relation is called  $n$ -ary relation. For  $n = 1, 2, 3, 4$  we use the shorthands unary, binary, ternary, quaternary, respectively. Relations are called **finite** when they have finite cardinality, **small** when their cardinality is no more than that of the domain, **cofinite** when their complements are finite, and **finitary** when they have finite arity. We only consider finitary relations.

Elements of  $\mathcal{D}^n$  are called  $n$ -tuples or just **tuples**, when  $n$  is understood or immaterial. Small Latin letters are typically used for elements of  $\mathcal{D}$ , and small Greek letters for tuples in  $\mathcal{D}^n$ . If  $\alpha \in \mathcal{D}^n$  it’s  $i$ -th member is denoted  $\alpha_i$ . 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. Some standard  $n$ -ary relations are the **universal relations**  $U_n := \mathcal{D}^n$  with all  $n$ -tuples, and the **identity relations**  $I_n$  with all and only  $n$ -tuples of identical elements. Occasionally, we abbreviate the latter as  $\vec{a} := (a, \dots, a)$ .

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.  $R(\alpha) = R(\alpha_1, \dots, \alpha_n)$  will mean the same as  $\alpha \in R$ , and  $\neg R$  will denote the complement of  $R$  in  $\mathcal{D}^n$ . Furthermore, we identify relations with their truth value functions, 0-1 functions on  $\mathcal{D}^n$ . Logical connectives, conjunction  $\wedge$ , disjunction  $\vee$ , etc., will be used with the usual meaning except in Section 4, where they are explicitly redefined in many-valued logics.

Technically, **primitive positive formulas (ppfs)** are those with only existential

quantifiers in the prefix and only conjunctions as connectives in the quantifier-free part [10, 17, 23]. However, any formula with only existential quantifiers and conjunctions can be converted into such a form by prenexing quantifiers, so we will apply the term more broadly to all such formulas. In particular, our ppfs are closed under taking conjunctions and existentially quantifying. The following definition introduces the central concepts used in the paper.

**Definition 1 (pp-composition).** *A relation is a **composition of relations**, or **decomposes** into them, when its predicate can be expressed by a ppf that contains only the predicates of the composing relations. A decomposition is a **reduction** when the composing relations have strictly lower arity than the composite. A relation is **reducible** when it admits a reduction.*

We prefer “pp-composition” to the more common “pp-definition” to emphasize the functional analogy and algebraic aspects, and drop “pp” when context allows. Cartesian products (free conjunctions), joins and projections (existential quantifications) of the relational database theory, relative products (2), triple (3) and higher junctions are all special cases of pp-composition. **Clones** are sets of functions containing projections and closed under composition, **coclones** are sets of relations containing diagonals and closed under pp-composition, they are dual to each other under the Pol-Inv Galois connection [18, 22].

## 2 The infinite we do immediately...

In this section we will describe Peirce’s hypostatic abstraction that reduces  $n$ -ary relations to binary ones, and the pairing construction of Löwenheim and Sierpiński that does the same for functions. Both require the domain to be infinite to work without extending it. This is one of those cases that prompted Erdős and Tarski to quip [33]: “The infinite we do immediately, the finite takes a little time.”

Given a domain  $\mathcal{D}$  and a relation  $R(x_1, \dots, x_n)$  on it, extend the domain by  $n$ -tuples of elements of  $\mathcal{D}$ , and define binary relations  $R_i(t, x)$  to hold if and only if there is a tuple  $t = (x_1, \dots, x_n) \in R$  with  $x = x_i$ . ‘Abstract’ tuples are hypostatized into new elements here, hence the name. By construction,

$$R(x_1, \dots, x_n) = \exists t [R_1(t, x_1) \wedge \dots \wedge R_n(t, x_n)], \quad (4)$$

and an arbitrary  $n$ -ary relation is reduced to binary ones by this ppf. The domain extension can sometimes be avoided using the following trick due to Herzberger [12] (Sierpiński used a similar trick for functions much earlier). Instead of adjoining the tuples, define an injective map  $\tau : R \rightarrow \mathcal{D}$  that uniquely ‘codes’ them into elements of  $\mathcal{D}$ , and redefine  $R_i(t, x)$  into a relation on the original  $\mathcal{D}$  that holds when  $t = \tau(x_1, \dots, x_n)$  is in the range of  $\tau$  and  $x = x_i$ . Then (4) still holds and no domain extension is needed.

For such encoding map to exist, we clearly need  $|R| \leq |\mathcal{D}|$ , such relations were termed *small* in [15]. In general,  $|R| \leq |\mathcal{D}|^n$  for  $n$ -ary  $R$ , and when  $\mathcal{D}$  is infinite  $|\mathcal{D}|^n = |\mathcal{D}|$  for any  $n$ . In other words, any (finitary) relation on an infinite domain is small and, hence, reducible. However, in general, the requisite bijections between  $\mathcal{D}$  and  $\mathcal{D}^n$  are non-constructive. By a theorem of Tarski [13, 11.3],  $|\mathcal{D}|^2 = |\mathcal{D}|$  for all infinite  $\mathcal{D}$  is already equivalent to the axiom of choice.

Since hypostatic abstraction can reduce ternary to binary relations it does not reduce functions to functions, or we could decompose 2-input functions into 1-input

ones. Löwenheim’s pairing construction [20] does not have this shortcoming, but it requires adjoining (or encoding) not just  $n$ -tuples, but also 2-tuples, 3-tuples, ...,  $(n-1)$ -tuples of the domain elements. We will describe the functional version of it used by Sierpiński [29], who was apparently unaware of Löwenheim’s work, to prove FRT on infinite domains.

After extending the domain, define recursively

$$i_2(x_1, x_2) := (x_1, x_2), \quad i_k((x_1, \dots, x_{k-1}), x_k) := (x_1, \dots, x_k)$$

for  $3 \leq k \leq n$ , and define  $\widehat{f}((x_1, \dots, x_n)) := f(x_1, \dots, x_n)$ . Then extend  $i_k$  and  $\widehat{f}$  arbitrarily to other elements. It is easy to see by induction that

$$f(x_1, \dots, x_n) = \widehat{f}\left(i_n(\dots i_4(i_3(i_2(x_1, x_2), x_3), x_4) \dots, x_n)\right)$$

reduces  $f$  to a composition of 2-input  $i_k$  and 1-input  $\widehat{f}$  (which can be turned into a 2-input by adding a dummy variable if one so wishes). Again, to avoid extending the domain one can encode the adjoined elements into the original ones. This is always possible on infinite domains, but requires the axiom of choice, as Sierpiński explicitly noted. We can summarize the constructions in this section in the following theorem that combines the reducibility clauses of RRT and FRT on infinite domains.

**Theorem 1.** *Let  $n \geq 3$ . Any  $n$ -ary relation on an infinite domain is a composition of  $n$  binary relations. Any  $n$ -input function on an infinite domain is a composition of  $n$  2-input functions.*

It is doubly ironic that the (ostensibly) harder case of relations was settled several decades earlier than the case of functions, and that the case of functions on finite domains, where one cannot produce simple explicit constructions, was settled before the one on infinite domains.

### 3 Functions, relations, relatives

In this section we will give an elementary construction that reduces relations on a domain to ternary relations using that functions on it can be reduced to 2-input functions. As with the hypostatic abstraction and the pairing construction, we will need the axiom of choice for the general case. However, the construction is of most interest on finite domains, where it is unnecessary and the reductions can be performed constructively. That any function on a finite domain reduces to a composition of 2-input functions follows from Post’s results for many-valued logics discussed in the next section.

Considered as relations, functions have a distinguished argument designated as output. Our approach will be to analogize relations to functions, so we will extend the notion of output to general relations. Linguistic expressions that Aristotle, de Morgan and Peirce took as motivation, such as “brother of \_\_\_” or “son of \_\_\_ by \_\_\_”, distinguish the output explicitly in the noun phrases. We will follow them in calling relations with a distinguished argument *relatives* [9]. In clone theory, relatives on finite domains are called “hyperoperations” [28], “partial hyperoperations” [24] or “multifunctions” [5].

**Definition 2.** *A relative is a relation on  $\mathcal{D}$  with a distinguished argument called the **output**. The remaining arguments are called **inputs**. Unless otherwise stated,*

the output is designated to be the last listed argument, e.g.  $x_n$  in  $R(x_1, \dots, x_n)$ . For  $\alpha \in \mathcal{D}^{n-1}$  we define its **image under**  $R$  as the set  $R(\alpha) := \{a \in \mathcal{D} \mid (\alpha, a) \in R\}$ . We call a relative **partial function** when  $R(\alpha)$  is empty or a singleton for every  $\alpha$ , and a **multifunction** when  $R(\alpha)$  is non-empty for every  $\alpha$ . When  $R(\alpha)$  is a singleton for every  $\alpha$ , we call  $R$  the **graph of a function**  $\mathcal{D}^{n-1} \rightarrow \mathcal{D}$ .

Thus, general relatives can be characterized as ‘partial multifunctions’. It will be convenient for us to interpret the domain of a relative as itself a relative rather than as just a set.

**Definition 3.** For any relative  $R(x_1, \dots, x_n)$ , we define the domain relative

$$\text{Dom}_R(x_1, \dots, x_{n-1}) := \exists t [R(x_1, \dots, x_{n-1}, t)]$$

as the projection on its inputs.

Note that  $\text{Dom}_R$  is a relative of lower arity. The following lemma codifies that any relative is a multifunction on its domain. The proof is straightforward and we omit it.

**Lemma 1.** Any relative  $R$  decomposes as

$$R(x_1, \dots, x_n) = F(x_1, \dots, x_n) \wedge \text{Dom}_R(x_1, \dots, x_{n-1}), \quad (5)$$

where  $F$  is a multifunction. If  $R$  is a partial function then  $F$  can be chosen to be a function, i.e.  $F(x_1, \dots, x_n)$  if and only if  $x_n = f(x_1, \dots, x_{n-1})$ .

Thus, any relation can be reduced to a multifunction and a relation of lower arity. In turn, we will reduce multifunctions to (ordinary) functions by using value selectors.

**Definition 4.** Let  $F$  be a multifunction on  $\mathcal{D}$ . A function  $f : \mathcal{D}^{n-1} \rightarrow \mathcal{D}$  is called a **value selector** for  $F$  when for any  $\alpha \in \mathcal{D}^{n-1}$  we have  $(\alpha, f(\alpha)) \in F$ , i.e.  $f(\alpha) \in F(\alpha)$ .

When  $\mathcal{D}$  is infinite, existence of selectors depends, in general, on the axiom of choice. There are  $\prod_{\alpha \in \mathcal{D}^{n-1}} |F(\alpha)|$  possible selectors, which can run up to  $(\mathcal{D}^{n-1})^{|\mathcal{D}|} = |\mathcal{D}|^{(n-1)|\mathcal{D}|}$  in cardinality. However, it will be important that at most  $|\mathcal{D}|$  are needed to cover all tuples in  $F$ . This is exploited in the next lemma.

**Lemma 2.** Let  $\mathcal{D}$  be a well-orderable set. Then any  $n$ -ary multifunction  $F$  on  $\mathcal{D}$  decomposes as

$$F(x_1, \dots, x_n) = \exists t [x_n = f(t, x_1, \dots, x_{n-1})], \quad (6)$$

where  $f : \mathcal{D}^n \rightarrow \mathcal{D}$  is a function.

*Proof.* Well-order  $\mathcal{D}$  and let  $f_1$  be the value selector of  $F$  that picks the least element of  $F(\alpha)$  at each  $\alpha \in \mathcal{D}^{n-1}$ . For every ordinal  $\sigma$  we define  $f_\sigma$  to pick the least element of  $F(\alpha)$  not picked by any  $f_\tau$  with  $\tau < \sigma$ , if any, and the least element of  $F(\alpha)$  otherwise. By transfinite induction, at some ordinal  $\xi$  with  $|\xi| \leq |\mathcal{D}|$  there will be no unpicked elements left for any  $\alpha \in \mathcal{D}^{n-1}$ , because  $|F(\alpha)| \leq |\mathcal{D}|$ . We put  $\xi$  into 1-1 correspondence with a subset  $T$  of  $\mathcal{D}$  and define  $\sigma(t)$  to be the image of  $t$  under this correspondence when  $t \in T$  and 1 otherwise. Finally, we set  $f(t, x_1, \dots, x_{n-1}) := f_{\sigma(t)}(x_1, \dots, x_{n-1})$ . Since  $f_\sigma$  is a selector for every  $\sigma$  only tuples in  $F$  will satisfy  $x_n = f_{\sigma(t)}(x_1, \dots, x_{n-1})$  for any  $t$ , and all of them are accounted for because every  $\sigma$  in  $\xi$  is  $\sigma(t)$  for some  $t$ . Thus, (6) holds.  $\square$

We will refer to  $f(t, x_1, \dots, x_{n-1})$  as the **indexing function** (of value selectors). When  $\mathcal{D}$  is finite, it is obtained constructively, and if we identify  $\mathcal{D}$  with the truth value set of a many-valued logic then we even have a natural well-order on  $\mathcal{D}$  that defines it uniquely. It is known that any  $n$ -input function with  $n \geq 3$  on any domain decomposes into 2-input functions (see Section 4 for finite domains). Relations of the form  $u = f(t, x)$  can then be reduced to ternary relations by recursively unfolding nested compositions. Combining this procedure with Lemmas 1 and 2, we get our main result.

**Theorem 2 (Relative reduction).** *Any  $n$ -ary relation  $R$  on any domain  $\mathcal{D}$  decomposes as*

$$R(x_1, \dots, x_n) = \exists t [x_n = f(t, x_1, \dots, x_{n-1})] \wedge \text{Dom}_R(x_1, \dots, x_{n-1}), \quad (7)$$

where  $f : \mathcal{D}^n \rightarrow \mathcal{D}$  is a function. Moreover, given a functionally complete set, every relation can be decomposed into graphs of functions from this set. In particular, every  $n$ -ary relation on  $\mathcal{D}$  with  $n \geq 4$  is reducible to ternary relations.

*Proof.* Formula (7) is a direct consequence of Lemmas 1 and 2. To decompose the first conjunct in (7), set  $x_0 := t$  and suppose

$$f(t, x_1, \dots, x_{n-1}) = g(h_1(x_{\Lambda_1}), \dots, h_m(x_{\Lambda_m})),$$

where  $\Lambda_i \subseteq \{0, 1, \dots, n-1\}$  and  $x_\Lambda$  is the list of  $x_i$  with  $i \in \Lambda$ . Then

$$\begin{aligned} & \exists t [x_n = f(t, x_1, \dots, x_{n-1})] \\ &= \exists t \exists t_1 \dots \exists t_m [x_n = g(t_1, \dots, t_m) \wedge t_1 = h_1(x_{\Lambda_1}) \wedge \dots \wedge t_m = h_m(x_{\Lambda_m})]. \end{aligned} \quad (8)$$

By iterating the procedure, if necessary, we can decompose the first conjunct into graphs of functions from a given complete set. The same process is then applied to  $\text{Dom}_R$ . Since the arity is reduced by 1 at each step the process terminates in finitely many steps. Since there exist functionally complete sets of 2-input functions, and their graphs are ternary, the last claim follows.  $\square$

Theorem 2 not only extends Theorem 1 to all domains, but its proof also reduces RRT to FRT, as Peirce's remarks suggested. However, the result is weaker. The hypostatic abstraction decomposed all relations into *binary* relations and bounded their number by arity. We will mend the first flaw in Section 5, but there can be no uniform arity bound on finite domains [16].

Recall that a function is called Sheffer when it alone forms a functionally complete system [18, 11.2],[25]. Peirce found one such function, NOR, a.k.a. Peirce arrow, on Boolean domains in an unpublished 1902 manuscript. Sheffer published the result for NAND, a.k.a. Sheffer stroke, in 1913 [1]. On general finite domains, the first Sheffer function was found by Webb in 1935, and it is a 2-input one [25, 33]. Their graphs give us ternary Sheffer relations on the corresponding domains, a result that appears to be new.

**Corollary 1 (Sheffer relations).** *On any finite domain, there exist ternary relations that alone compose all relations.*

Let us illustrate the construction of Theorem 2 with a couple of examples.



**Example 1.** Consider the  $n$ -identity relation  $I_n$ , whose tuples are all and only  $n$ -tuples of identical elements. Taking  $x_n$  as the output, we see that  $I_n$  is a partial function with  $\text{Dom}_{I_n} = I_{n-1}$ . On this domain,  $I_n$  can be identified with a function that extends to all of  $\mathcal{D}$  as e.g.  $f(x_1, \dots, x_{n-1}) := x_1$ . This means, according to (5), that

$$I_n(x_1, \dots, x_n) = [x_n = x_1] \wedge I_{n-1}(x_1, \dots, x_{n-1}).$$

Iterating  $n - 1$  times we obtain a familiar decomposition of  $I_n$  into diagonals, namely

$$I_n(x_1, \dots, x_n) = [x_n = x_1] \wedge \dots \wedge [x_2 = x_1].$$

**Example 2.** The  $n$ -identity is small even on finite domains, and hence can be decomposed into binaries by hypostatic abstraction even there. In contrast,  $|\neg I_n| = n^{|\mathcal{D}|} - n > n$  for  $n \geq 3$  and  $|\mathcal{D}| \geq 2$ , i.e. for  $n \geq 3$  the non- $n$ -identity is large already on Boolean domains. We will decompose it there first, where we can take advantage of the familiar Boolean algebra, and postpone the case of general finite domains until the next section, after Post's many-valued logics are introduced for the task.

Note that  $\text{Dom}_{\neg I_n} = U_{n-1}$  is the universal relation (because any  $(n - 1)$ -tuple can be complemented to an  $n$ -tuple with not all entries equal), and we can drop it from the decomposition in (7). It remains to pick value selectors for the multifunction  $\neg I_n$  and decompose the resulting indexing function into 2-input functions.

In the Boolean case we can identify  $\mathcal{D}$  with  $\{0, 1\}$  and pick

$$f_0(x) := \begin{cases} 0, & x \neq (0, \dots, 0) \\ 1, & x = (0, \dots, 0) \end{cases} \quad f_1(x) := \begin{cases} 1, & x \neq (1, \dots, 1) \\ 0, & x = (1, \dots, 1) \end{cases}$$

as the value selectors. Then, using the standard notation for Boolean operations,

$$\begin{aligned} f_0(x) &= \bar{x}_1 \wedge \dots \wedge \bar{x}_{n-1}, & f_1(x) &= \bar{x}_1 \vee \dots \vee \bar{x}_{n-1}; \\ \text{so } f(t, x) &= t \wedge \overline{x_1 \wedge \dots \wedge x_{n-1}} \vee \bar{t} \wedge \bar{x}_1 \wedge \dots \wedge \bar{x}_{n-1} \end{aligned} \quad (9)$$

satisfies  $f(i, x) = f_i(x)$  for  $i = 0, 1$ . Since (9) expresses  $f(t, x)$  via 2-input and 1-input functions the template (8) will decompose  $\neg I_n$  into ternary and binary relations of the form  $z = x \wedge y$ ,  $z = x \vee y$  and  $y = \bar{x}$ .

## 4 Reduction on Post's logics

This section is expository and can be skipped without loss of continuity. Its purpose is to demonstrate that the reduction procedure of Theorem 2 is effective on all finite domains, and quite practical considering the wealth of functional decomposition methods. We recall the analogs of Boolean operations in Post's many-valued logics and of the canonical disjunctive normal form that reduces any function on them to 2-input and 1-input functions. The construction is well-known, see e.g. [18, 1.4], [25], [31, 2.8], but is rarely spelled out explicitly. As an illustration, we generalize to finite domains the Boolean reduction of  $\neg I_n$ .

From the modern perspective, a simple way to introduce (most of) Post's  $(k + 1)$ -valued logic is by restricting the standard fuzzy logic with truth values in  $[0, 1]$  to the subset  $\left\{0, \frac{1}{k}, \dots, \frac{k-1}{k}, 1\right\}$ . However, it is more convenient to work with integers, so,

as is customary in the clone theory, we multiply the truth values by  $k$ . Then their set becomes  $E_{k+1} := \{0, 1, \dots, k-1, k\}$ , and the standard connectives are defined by

$$a \wedge b := \min(a, b); \quad a \vee b := \max(a, b); \quad \bar{a} := k - a.$$

For  $k = 1$  we recover the Boolean algebra  $E_2$ . Alas, for  $k > 1$  the system  $\{\wedge, \vee, \bar{\cdot}\}$  is not functionally complete. Instead of  $\bar{\cdot}$ , called the *diametric negation*, Post introduced the *cyclic shift*

$$a' := a + 1 \pmod{k+1},$$

sometimes also called cyclic negation since  $a' = \bar{a}$  in  $E_2$ . The system  $\{\wedge, \vee, '\}$  turns out to be functionally complete.

To see this, first note that for any  $x$

$$0 = x \wedge x' \wedge \dots \wedge x^{(k)}, \quad 1 = 0', \quad 2 = 0'', \dots, \quad k = 0^{(k)},$$

so we can generate all the constant functions. Next, consider the shifts  $x^{(j)}$  with  $j \neq i$ . One of them is  $k$  unless  $k = x^{(i)}$ , i.e.  $x = k - i$ . Therefore,

$$\bigvee_{j \neq i} x^{(j)} = \begin{cases} k-1, & x = k-i \\ k, & x \neq k-i \end{cases},$$

and we can generate the *literals* (a.k.a. value isolators) [25]:

$$x^i := \begin{cases} k, & x = i \\ 0, & x \neq i \end{cases} = \left( \bigvee_{j \neq k-i} x^{(j)} \right)'.$$

In the Boolean algebra,  $x^0 = \bar{x}$  and  $x^1 = x$ . From the literals we generate the analogs of Boolean monomials ( $x$  and  $\alpha$  below are now  $n$ -tuples of variables and truth values, respectively):

$$x^\alpha := x_1^{\alpha_1} \wedge \dots \wedge x_n^{\alpha_n} = \begin{cases} k, & x = \alpha \\ 0, & x \neq \alpha \end{cases}. \quad (10)$$

Since  $k$  is maximal (the “true”)  $i \wedge x^\alpha$  returns the given  $i$  on  $\alpha$  and 0 otherwise. With the above, any function  $f : E_{k+1} \rightarrow E_{k+1}$  decomposes into the *canonical disjunctive normal form* (CDNF) [18, 1.4], [25], which can be read off from its truth table as in the Boolean case (taking the empty disjunction to return 0):

$$f(x) = \bigvee_{i \in E_{k+1} \setminus \{0\}} \left( \bigvee_{f(\alpha)=i} i \wedge x^\alpha \right). \quad (11)$$

For the diametric negation, we get, for example,  $\bar{x} = \bigvee_{i \in E_{k+1} \setminus \{0\}} i \wedge x^{k-i}$ . In the Boolean case,

CDNF reduces to the familiar  $f(x) = \bigvee_{f(\alpha)=1} x^\alpha$ .

Note that CDNF directly uses, in addition to the 2-input  $\wedge, \vee$ , only 0-input constants and 1-input literals, so we can replace the cyclic shift with them, and get another complete system of functions with at most 2 inputs. Webb’s 2-input Sheffer function, a multi-valued generalization of Sheffer stroke, is given by  $(a \vee b)'$ .

Identifying  $\mathcal{D}$  with  $E_{k+1}$  for  $k := |\mathcal{D}| - 1$ , we get 2-input decompositions on any finite domains. Let us illustrate the above constructions by generalizing Example 3 to the case of arbitrary  $k \geq 1$ .

**Example 3.** As in the Boolean case,  $\neg I_n$  is a multifunction, so we only need to find its value selectors and decompose their indexing function  $f(t, x_1, \dots, x_{n-1})$  into CDNF. Let  $\vec{i} := (i, \dots, i)$ . With  $\mathcal{D} = E_{k+1}$ , the value selectors can be

$$f_i(x) := \begin{cases} i, & x \neq \vec{i} \\ i', & x = \vec{i} \end{cases}.$$

Indeed,  $f_i$  select from  $\neg I_n$  because  $i \neq i'$  for any  $i$ , and they exhaust  $E_{k+1}$  because every  $(n-1)$ -tuple other than  $\vec{i}$  is evaluated to  $i$  by  $f_i$ , while  $\vec{i}$  is evaluated to any  $j \neq i$  by  $f_j$ . These selectors can be uniformized as  $f_i(x) = i \wedge \overline{x^{\vec{i}}} \vee i' \wedge x^{\vec{i}}$ . As a result, we get an expression for the indexing function analogous to the Boolean one (9), with the cyclic shift and diametric negation replacing the Boolean negation in different places:

$$f(t, x) = t \wedge \overline{x_1^t \wedge \dots \wedge x_{n-1}^t} \vee t' \wedge x_1^t \wedge \dots \wedge x_{n-1}^t.$$

To decompose  $\neg I_n$  into ternaries and binaries on any  $E_{k+1}$ , it remains to repeatedly apply the template (8).

## 5 Ternary from binary relations

A natural question, left unanswered by Theorem 2, is whether ternary relations reduce to binary relations on finite domains, as they do on infinite domains by hypostatic abstraction. This is the question we will take up in this section. Converting a function into a relation requires adding an extra argument to it, so binary reduction along the same lines would require us to compose all functions from 1-input ones, which is, obviously, impossible. If the binary reduction is possible it must exploit a different idea.

Our idea is to reduce cofinite relations of small *arity* (up to the cardinality of the domain) to binary relations. In a sense, it is dual to hypostatic abstraction, which reduces relations of small *cardinality*. Unlike the BKKR construction [3], or the non-constructive proof via the Galois connection [11], [22, 1.1.22], our construction, together with Theorem 2, can be said to explain ‘why’ RRT is true when  $|\mathcal{D}| \geq 3$  and suggest ‘why’ it breaks down on Boolean domains. Of course, the failure of a construction does not establish the negative claim in and of itself, but it is known that RRT fails on Boolean domains and irreducible ternary relations do exist [11]. For example,  $\neg I_3$  is one. For completeness, we will give an elementary proof in the next section.

Treating relations on  $\mathcal{D}$  as predicates, i.e. functions  $\mathcal{D}^n \rightarrow E_2$ , we can apply Boolean operations to them. This allows to easily build many relations from fairly simple primitives. However, Boolean formulas do not decompose them, in our sense, into the primitives when they involve non-positive operations  $\neg$  or  $\vee$ . We will write  $I_2(x, y)$  and  $\neg I_2(x, y)$  in the more traditional form  $x = y$  and  $x \neq y$ , respectively, and bracket them when confusion with the equality of formulas may result.

The predicates  $x = a$  coincide with the literals  $x^a$  from (10) if we identify  $\mathcal{D}$  with  $E_{k+1}$  and  $E_2$  with  $\{0, k\} \subseteq E_{k+1}$ , so we will also call them literals. Conjoining the literals, we express the singleton relations for  $\alpha \in \mathcal{D}^n$  as monomials

$$[x = \alpha] := [(x_1 = \alpha_1) \wedge \dots \wedge (x_n = \alpha_n)] = \bigwedge_{j=1}^n (x_j = \alpha_j), \quad (12)$$

where now  $x := (x_1, \dots, x_n)$ . Any finite cardinality relation on  $\mathcal{D}$  can be built from them using disjunctions, namely

$$R(x) = \bigvee_{\alpha \in R} (x = \alpha). \quad (13)$$

This is just the predicate version of Boolean CDNF.

In decompositions, aside from conjunctions, we can only use existential quantification. One can emulate some disjunctions with it, but only when the number of disjuncts is at most  $|\mathcal{D}|$  and at the expense of increasing their arity by 1. The idea is to convert the index into a (bound) variable, as in  $\bigvee_j R_j(x) = \exists t R(t, x)$ . The literals are unary, so converting them into binary relations is acceptable, but the number of disjuncts in the outer disjunction of (13) is  $|R|$ . This can go up to  $|\mathcal{D}|^n$  for  $n$ -ary  $R$ , so we cannot convert CDNF into a ppf in general. When it does work, i.e. for small cardinality relations, it reproduces the hypostatic abstraction (4). This is a dead end.

It could have been anticipated if we noticed that both conjunction (intersection) and existential quantification (projection) do not increase the cardinality of relations. If we are to compose all relations we must start from relations of largest cardinality, not the singletons. Aside from the universal relation  $U_n$ , those are the cosingletons expressed by comonomials

$$[x \neq \alpha] := [(x_1 \neq \alpha_1) \vee \dots \vee (x_n \neq \alpha_n)] = \bigvee_{j=1}^n (x_j \neq \alpha_j). \quad (14)$$

Negating the CDNF for  $\neg R$  we obtain the predicate version of CCNF

$$R(x) = \bigwedge_{\alpha \notin R} (x \neq \alpha). \quad (15)$$

for any cofinite  $R$ . On finite domains, all relations are both finite and cofinite, so CCNF decomposes them all into the comonomials.

Alas, the arity of  $x \neq \alpha$  must match the arity of  $R$ , and their Boolean reduction (14) to unary coliterals is not a ppf. However, the disjunction there is not as bad as in CDNF. The number of disjuncts is only  $n$ , so for  $n \leq |\mathcal{D}|$  the essential condition of converting disjunction into existential quantification is met. There is also a technical issue that disjuncts coming from turning  $\exists t R(t, x)$  into  $\bigvee_j R_j(x)$  must have the same list of variables, which is not the case in (14). It can be handled as follows. Note that

$$[x_j \neq \alpha_j] = \bigwedge_{i=1}^n [(j \neq i) \vee (x_i \neq \alpha_i)]$$

since the only non-trivial conjunct has  $j = i$ , and the right hand side now lists all  $n$  variables. When  $n \leq |\mathcal{D}|$ , fix an injection  $\varphi : \{1, \dots, n\} \rightarrow \mathcal{D}$ , and define the unary relation  $\text{Ran}_\varphi(t)$  that holds on the range of  $\varphi$ . Now replace  $j$  by an indexing variable  $t$ , and note that  $j \neq i$  is equivalent to  $\varphi(j) \neq \varphi(i)$ . For  $i = 1, \dots, n$  and  $a \in \mathcal{D}$  define the binary relations

$$B_{i,a}(t, x) := [(t \neq \varphi(i)) \vee (x \neq a)] \wedge \text{Ran}_\varphi(t).$$

When  $\mathcal{D}$  is identified with  $E_{k+1}$  one can take  $\varphi(i) := i - 1$ . The centerpiece of our construction is a reduction of low arity comonomials  $x \neq \alpha$  to these binary relations by ‘existentialization’ of their defining disjunction (14). It is similar to the procedure used in the proof of Lemma 2 to ‘existentialize’ the disjunction of value selectors.

**Theorem 3.** For any  $3 \leq n \leq |\mathcal{D}|$  any cofinite  $n$ -ary relation on a domain  $\mathcal{D}$  reduces to binary relations as follows:

$$R(x) = \bigwedge_{\alpha \notin R} \exists t \left[ \bigwedge_{i=1}^n B_{i,\alpha_i}(t, x_i) \right]. \quad (16)$$

*Proof.* The existentially quantified conjuncts in (16) simply reduce  $x \neq \alpha$  to binary relations  $B_{i,\alpha}(t, x)$ . Indeed, due to the conjunction with  $\text{Ran}_\varphi(t)$ , we can replace  $t$  by  $\varphi(j)$  and  $\exists t$  by disjunction over  $j$  without changing the corresponding relation. The disjuncts then simplify to

$$\begin{aligned} \bigwedge_{i=1}^n B_{i,\alpha_i}(\varphi(j), x_i) &= \bigwedge_{i=1}^n [(\varphi(j) \neq \varphi(i)) \vee (x_i \neq \alpha_i)] \\ &= \bigwedge_{i=1}^n [(j \neq i) \vee (x_i \neq \alpha_i)] = [x_j \neq \alpha_j]. \end{aligned}$$

By (14), their disjunction further simplifies to just  $x \neq \alpha$ . It remains to apply CCNF (15) to the outer conjunction over  $\alpha$ .  $\square$

Combining this result (even just for  $n = 3$ ) with Theorem 2 we can now reduce any cofinite relation to binary ones as long as  $|\mathcal{D}| \geq 3$ . For  $|\mathcal{D}| = 2$  the above construction obviously breaks down because the injection  $\varphi$  does not exist. There are not enough elements in Boolean domains to emulate triple and higher disjunctions by existential quantification. Although this does not prove irreducibility of some ternary relations, it strongly suggests it.

While the BKKR reduction [3] is less transparent than ours and works only on finite domains, it does prove a stronger result: relations of higher arity are reduced not just to *some* binary relations, but to two specific ones. Let us briefly sketch it for comparison. Identifying  $\mathcal{D}$  with  $E_{k+1}$ , we have the binary relation  $\leq$  on it since  $E_{k+1}$  is well-ordered. From  $\leq$  and  $\neq$ , we can compose  $<$  as  $[x < y] = [(x \leq y) \wedge (x \neq y)]$ . The constants and literals are then composed using  $<$ . Negations of the literals  $x = a$  are composed from them and  $\neq$  as  $\exists t [(x \neq t) \wedge (t = a)]$ . The ternary relation

$$Q_3(x, y, z) := [(x = y) \rightarrow (x = z)] = [(x \neq y) \vee (x = z)]$$

plays a key role in the BKKR construction [3]. Its defining formula is not a ppf, but BKKR do explicitly reduce  $Q_3$  to  $\leq$  and  $\neq$  for  $|\mathcal{D}| \geq 3$ . A rather long and opaque ppf then reduces cosingletons  $x \neq \alpha$  to  $Q_3$  and some binary relations, and an application of CCNF (15) completes the reduction, as in our case. When reducing cosingletons, BKKR have to separately consider different patterns of identical entries in them. Much of that complexity is removed in our construction by outsourcing reduction of high arity cosingletons to functional decompositions in the relative reduction.

## 6 Boolean wrinkle

We will now confirm that the exclusion of Boolean domains from Theorem 3 is not an artifact of the construction. The fact that some ternary relations are irreducible on Boolean domains is well-known and follows from the structure of the coclone lattice, which is dual to the Post lattice of Boolean clones. The first explicit proof seems to be

due to Schaefer [30], who did not use clone theory, but it is somewhat convoluted. In the spirit of the paper, we will give a simple elementary proof.

Boolean relations, when identified with their predicates, coincide with Boolean functions as sets, both are maps  $E_2^n \rightarrow E_2$ . Therefore, relations can be represented by standard Boolean expressions, like  $x \oplus y$  for  $x \neq y$ , or  $x \rightarrow y$  for  $x \leq y$ . Relational CDNF (13) and CCNF (15) then turn into the familiar functional versions, but the composition is different. Functional composition naturally aligns with substitutions into standard connectives, while relational pp-composition is less intuitive. We can split it into three operations: taking conjunctions, identifying variables in predicates and existentially quantifying. The former two translate into the corresponding operations on representing functions, while the latter becomes

$$\exists t[R(x_1, \dots, x_{n-1}, t)] = R(x_1, \dots, x_{n-1}, 0) \vee R(x_1, \dots, x_{n-1}, 1). \quad (17)$$

Of particular interest to us are relations called *bijunctive* [10, 30], which are defined, in terms of their truth value functions, as those admitting conjunctive normal forms with at most two literals (2CNF). Recall that conjuncts of CNF are, in general, finite disjunctions of literals called *clauses*. Clearly, all unary and binary relations are bijunctive, as their CCNF is a 2CNF. Moreover, bijunctivity is preserved by pp-composition.

**Lemma 3.** *A pp-composition of bijunctive relations is bijunctive.*

*Proof.* Conjunction of 2CNF is trivially a 2CNF, and identification of variables either leaves 2-clauses as 2-clauses, collapses them into 1-clauses (when the literals coincide), or into constant 1 (when the literals are complementary). So the result is still a 2CNF. For the quantification, note that substituting 0 or 1 into 2-clauses turns them into either 1-clauses or constants. Then 1-s can be removed from the resulting conjunction, and 0-s would turn it into 0. Unless it is the latter, the disjunction in (17) becomes a disjunction of two monomials, and distributing  $\wedge$  over  $\vee$  turns it into a 2CNF.  $\square$

Thus, to show that a Boolean relation does not decompose into binary ones it suffices to show that it is not bijunctive. That we can do by elementary means already for ternary relations.

**Theorem 4.**  *$\neg I_3$  is non-bijunctive, and hence irreducible to binary relations on Boolean domains.*

*Proof.* By contradiction, suppose  $\neg I_3$  has a 2CNF. Then  $I_3$  has a 2DNF, a disjunction of monomials with at most two literals, obtained by negating the 2CNF and applying de Morgan laws. Unary monomials, like  $x$  or  $\bar{x}$ , would make the entire disjunction equal 1 for  $x = 1$  or  $x = 0$ , respectively, regardless of  $y$  and  $z$  values. So no such monomials are there. There are no binary monomials with mismatched literals either. Indeed, say  $x \wedge \bar{y}$  or  $\bar{x} \wedge y$  would make the disjunction equal 1 for unequal values of  $x$  and  $y$ . Finally, binary monomials with matched literals, like  $x \wedge y$  or  $\bar{x} \wedge \bar{y}$ , would make the disjunction equal 1 for  $x = y = 1$  or  $x = y = 0$ , respectively, regardless of  $z$  values. But if no monomials are there, the supposed 2DNF of  $I_3$  must be a constant, contradiction.  $\square$

We now have at least one ternary relation that is not reducible to binary ones on Boolean domains, and we can present several more. Theorem 2 decomposes all relations into ternary relations of the form  $z = g(x, y)$ , the graphs of 2-input functions used in functional decompositions. Consider any 2-input Boolean function  $g$  that alone, or

together with 1-input negation, forms a functionally complete system. If the ternary relation  $z = g(x, y)$  were bijunctive then, by Theorem 2, every relation would have been bijunctive. Thus, the graphs of  $\wedge$ ,  $\vee$ ,  $\rightarrow$ ,  $\leftarrow$ ,  $|$  (Sheffer stroke),  $\downarrow$  (Peirce arrow) are all non-bijunctive.

Leaving aside the elementary approach, one can prove much stronger results. Bijunctive relations form a maximal coclone, meaning that adding any non-bijunctive relation to them gives a relationally complete set [4, 10]. Moreover, this coclone is pp-compositionally generated by binary relations. In fact, one can take  $\leq$  and  $\neq$ , those two relations that generate the coclone of *all* relations when  $|\mathcal{D}| \geq 3$  by the BKKR construction, as the bijunctive generators for  $E_2$  [4]. This means that  $\leq$ ,  $\neq$ ,  $\neg I_3$  generate all Boolean relations, and one can replace  $\neg I_3$  by any non-bijunctive Boolean relation of any arity.

There is also an alternative characterization that allows to test relations for bijunctivity based on their truth value set. It was proved by Schaefer, but was probably known earlier. To describe it, define the 3-input majority function as follows

$$\mu_3(x, y, z) := (x \wedge y) \vee (y \wedge z) \vee (x \wedge z). \quad (18)$$

It returns 1 when a majority of its inputs is 1, and 0 when a majority is 0. It turns out that bijunctive relations are all and only those preserved by  $\mu_3$ . That is, if we extend  $\mu_3$  entrywise to a tuple-valued function on tuples then  $\mu_3(\alpha, \beta, \gamma)$  must be in the relation whenever  $\alpha, \beta, \gamma$  are.

By the Pol-Inv Galois connection, maximality of the bijunctive coclone is equivalent to minimality of the clone generated by  $\mu_3$ . On non-Boolean domains, there are many 3-input functions that return the majority value when there is one. There are 729 of them already on 3-element domains [27]. From this point of view, the reason all relations can be reduced to binary relations on non-Boolean domains is that none of those majority functions preserves all binary relations.

## 7 Bonds and teridentity

The original version of the reduction thesis for relations goes back to Peirce [9, 15]. However, he operated with a more restrictive notion of composition that iterates relative products and excludes many ppfs, making reduction to binary relations impossible even on non-Boolean domains. After Löwenheim's result on binary reducibility [20], Peirce's thesis became controversial [15], and was not treated mathematically until the formalizations of Herzberger [12] and Burch [8] in 1980-s. And its connection to universal algebra and pp-decompositions was only pointed out in 2004 by Hereth and Pöschel [11].

In this section, we will show, following Burch [8, 9], that, somewhat surprisingly, for arities  $n \geq 4$  Peirce's restricted composition leads to the same notion of reducibility as pp-composition. However, some ternary relations, are irreducible on Peircean definition. One such relation, the ternary identity  $I_3$ , **teridentity** for short, plays a key role in converting pp-decompositions into Peircean ones, and is the only ternary relation needed in Peircean reductions.

The relative product is a binary operation on predicates where one inserts the same variable into a single argument of each and existentially quantifies over it. As we saw in the Introduction, this is the most direct generalization of the composition

of two functions. One can also insert the same variable into *two* different arguments of the same predicate and existentially quantify over it. These are the only ways to identify variables, so only free conjunctions are allowed, no joins. The triple junction  $\exists t [P(t, x) \wedge Q(t, y) \wedge R(t, z)]$  and higher cannot be generated by Peirce's means either, because the identified variables are quantified over and no new variable can be identified with them afterwards. But those junctions are exactly the ones featured in hypostatic abstraction (4).

To summarize, in predicate formulas produced by iterating Peirce's operations no variable can occur in more than two positions, and if it does it has to be a bound variable. We will adopt Herzberger's term "bond" for this restricted class of compositions, although our bond is slightly more permissive than his, along the lines of [11]. The name is inspired by the analogy with chemical bonding promoted by Peirce himself. For a detailed study of bonding and its relation to graph theory, see [16].

**Definition 5 (Bond).** *A bonding formula (bf) is a primitive positive formula where no free variables repeat, and bound variables repeat at most twice. We say that a relation is a **bond of relations**, or that it is **bonded** from them, when its predicate can be expressed by a bf with their predicates being the only predicate symbols in it. A bond is a **bond reduction** when the bonded relations have strictly lower arity than their bond. A relation is **bond reducible** when it admits a bond reduction.*

Peirce's first observation was that multiple variable identifications bond reduce to pairwise ones by using  $n$ -ary identity predicates  $I_n$ , e.g.

$$R_1(t, x_{\Lambda_1}) \wedge \cdots \wedge R_n(t, x_{\Lambda_n}) = \exists t_1 \dots \exists t_n [I_{n+1}(t, t_1, \dots, t_n) \wedge R_1(t_1, x_{\Lambda_1}) \wedge \cdots \wedge R_n(t_n, x_{\Lambda_n})]. \quad (19)$$

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

$$I_n(x_1, \dots, x_n) = \exists t_1 \dots \exists t_{n-3} [I_3(x_1, x_2, t_1) \wedge I_3(t_1, x_3, t_2) \wedge \cdots \wedge I_3(t_{n-3}, x_{n-1}, x_n)]. \quad (20)$$

Applying the above identities, one can convert a pp-composition into a bond of the original relations and teridentities. As in [15, 16], we call this conversion **bond explication**. Since the explication adds only ternary relations we have the following theorem.

**Theorem 5.** *For  $n \geq 4$ , an  $n$ -ary relation is pp-reducible if and only if it is bond reducible. A ternary relation is pp-reducible if and only if it is a bond of binary relations and teridentities.*

*Proof.* The only claim not covered by the bond explication is that ternary bonds of binary relations and teridentities are 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 (19), this produces an equivalent expression, and, since all teridentities are removed, it is a composition of binary relations only.  $\square$

In other words, bond reducibility to ternary relations is equivalent to pp-reducibility to binary relations. This justifies our talking of Peirce's reduction thesis despite using



a different notion of composition. Ironically, Löwenheim’s result on binary reducibility [20] once made Peirce’s reduction thesis controversial due to the confusion between these two closely related notions. However, while bond and compositional reducibilities are (almost) the same, bond reductions are much more special than general relational reductions, and typically involve many more ternary relations (teridentities).

As for the “almost”, the two reducibilities do diverge on ternary relations. Ideologically, the proof goes back to Peirce himself, and variants of it are given in [8, 11, 12, 15].

**Theorem 6.**  $I_3$  is bond irreducible on any domain with at least two elements.

*Proof.* By contradiction, suppose we have a bf expressing  $I_3(x, y, z)$  with only binary predicates in it. Since free variables are not shared there is a single predicate with  $x$  in it. Let  $t_1$  be the other variable in it. If it is shared, it is with a single other predicate and we let  $t_2$  be the other variable in that one. And so on. This chain of variables stops at a variable not shared, either free or bound. None of the shared variables can be free, so our chain has at most two free variables (the first and the last). Moreover, variables from the chain can only appear in the predicates encountered in its construction, or they would be shared by more than two predicates.

Therefore, after prefixing quantifiers if necessary, we can split off the conjunction of the encountered predicates and represent our bf as a free conjunction of the form  $P(x) \wedge Q(y, z)$ , or  $P(x, y) \wedge Q(z)$ , or  $P(x, z) \wedge Q(y)$ . However, none of these can represent  $I_3$ . Indeed, suppose  $I_3(x, y, z) = P(x, y) \wedge Q(z)$  and let  $a \neq b \in \mathcal{D}$ . Then  $P(a, a) \wedge Q(a)$  and  $P(b, b) \wedge Q(b)$  are both true, so  $P(a, a)$  and  $Q(b)$  are true, and hence  $P(a, a) \wedge Q(b)$  must be true as well. But  $I_3(a, a, b)$  is false, and analogous contradiction results in the other two cases.  $\square$

The proof implicitly employs graph-theoretic concepts, which are made explicit and developed in [11, 16]. It also shows that any bond reducible ternary relation must be a free conjunction (Cartesian product) of unary and/or binary relations. Peirce saw it as a vindication of a special role of ternary relations, and teridentity in particular, in actually *relating* their arguments, which Cartesian products fail to do [6, 15].

## 8 Conclusions and open problems

We gave an elementary construction of pp-reductions of higher arity relations to ternary and binary relations on finite domains. A key part of the proof reduces decomposition of relations to decomposition of functions, which is of general interest.

Unlike functional completeness, its relational analog is little studied. In particular, [25], there are no general relational analogs of completeness criteria, nor of full characterizations of maximal coclones on general finite domains. As a result, some basic questions long settled for functions remain open for relations. For example, our construction implies existence of Sheffer relations on any finite domain, but we do not know if there are any *binary* Sheffer relations on non-Boolean domains. The smallest known pp-complete system for them was constructed by BKKR and contains two binary relations,  $\leq$  and  $\neq$ . There are some intriguing parallels between the role of unary relations in the CSP tractability conditions [7] and of 1-input functions in Slupecki’s criterion of functional completeness [26] that suggest a possible direction of research.

There are even fewer completeness results for subsets of relations closed under the pp-composition (coclones), and it might be useful to relativize our construction to them.

Recall that a set of relations that pp-compose all other relations in a coclone is called its *base* [4]. Our Theorem 2 can be restated as saying that graphs of functionally complete systems are bases in the coclone of all relations. New bases of small arity for Boolean coclones were recently constructed in a number of works, including [4, 17, 21], but the non-Boolean case remains almost untouched. While not all coclones admit bases of consisting of function graphs even in the Boolean case, many do, and even those that do not have bases where graphs are supplemented by simple non-functional relations (like  $\leq$ ). It is also of interest which coclones admit graph bases of small arity. The minimal required arity has been studied in the clone theory literature under the name of relation degree [18, ch. 10].

Finally, in a stark contrast to logic functions [25], Shannon complexity of pp-decompositions into relational “elements” has not been studied almost at all. Even qualitative analysis reveals a striking discontinuity between decompositions on finite and infinite domains that deserves further study [16]. On infinite domains, one can reduce any  $n$ -ary relation to a projected conjunction of just  $n$  binary relations. On finite domains, the structure of the decomposition is much more complex even for small arities, and the minimal number of binary relations in it grows with the size of the domain. The minimal number of ternary relations in bond reductions on finite domains has some information theory implications and is studied in [16] in connection with Peirce’s work.

## References

- [1] I. Anellis, The genesis of the truth-table device, *Russell: the Journal of the Russell Archives*, 24 (2004) 55-70.
- [2] L. Barto, Z. Brady, A. Bulatov, M. Kozik, D. Zhuk, Minimal Taylor algebras as a common framework for the three algebraic approaches to the CSP, in *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), IEEE Press, Piscataway, NJ*, 2021, 13 pp.
- [3] V. Bodnarchuk, L. Kaluzhnin, V. Kotov, B. Romov, Galois theory for Post algebras I, II, *Cybernetics*, 5 (1969) 243-252 and 531-539.
- [4] E. Böhrer, S. Reith, H. Schnoor, H. Vollmer, Bases for Boolean coclones, *Information Processing Letters*, 96 (2005) no. 2, 59-66.
- [5] F. Börner, Basics of Galois connections, in *Complexity of Constraints, Lecture Notes in Computer Science, v. 5250*, Springer, Berlin, 2008, 38-67.
- [6] J. Brunning, C. S. Peirce’s relative product, *Modern Logic*, 2 (1991) no. 1, 33-49.
- [7] A. Bulatov, P. Jeavons, A. Krokhin, Classifying the complexity of constraints using finite algebras, *SIAM Journal on Computing*, 34 (2005), no.3, 720-742.
- [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] N. Creignou, H. Vollmer, Boolean constraint satisfaction problems: when does Post’s lattice help? in *Complexity of Constraints, Lecture Notes in Computer Science, vol. 5250*, 2008, Springer, Berlin, 3-37.
- [11] J. Hereth, R. Pöschel, The power of Peircean Algebraic Logic (PAL), in *Concept Lattices, Second International Conference on Formal Concept Analysis, Springer, Berlin*, 2004, 337-351.
- [12] H. Herzberger, Peirce’s remarkable theorem, in *Pragmatism and Purpose: Essays Presented to Thomas A. Goudge*, *University of Toronto Press*, 1981, 41-58.
- [13] T. Jech, *The axiom of choice*, Elsevier, New York, 1973.

- [14] P. Jeavons, D. Cohen, M. Gyssens, Closure properties of constraints, *Journal of the ACM*, 44(4) (1997) 527-548.
- [15] S. Koshkin, Is Peirce’s reduction thesis gerrymandered? *Transactions of the Charles S. Peirce Society*, 58 (2022) no. 4, 271-300.
- [16] S. Koshkin, Logical reduction of relations: from relational databases to Peirce’s reduction thesis, *Logic Journal of the IGPL*, 31 (2023) no. 4, doi: 10.1093/jigpal/jzad010.
- [17] 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.
- [18] D. Lau, *Function algebras on finite sets*, Springer-Verlag, Berlin, 2006.
- [19] C.I. Lewis, *A survey of symbolic logic*, University of California Press, Berkeley, 1918.
- [20] L. Löwenheim, *Über Möglichkeiten im Relativkalkül* (German), *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.
- [21] S. Marchenkov, The invariants of Post classes, *Fundamental’naya i Prikladnaya Matematika*, 4 (1998) no. 4, 1385-1404.
- [22] D. Pöschel, L. Kalužnin, *Funktionen- und Relationenalgebren*, *Mathematische Monographien* 15, Deutscher Verlag der Wissenschaften, Berlin, 1979.
- [23] A. Robinson, Recent developments in model theory, in *Proceedings of the 1960 International Congress on Logic, Methodology and Philosophy of Science, Stanford, CA*, 1962, pp. 60-79.
- [24] B. Romov, Partial hyperclones on a finite set, in *Proceedings of 32nd IEEE International Symposium on Multiple-Valued Logic, Boston*, 2002, 17-22.
- [25] I. Rosenberg, Some algebraic and combinatorial aspects of multiple-valued circuits, in *Proceedings of 6th IEEE International Symposium on Multiple-Valued Logic, Logan, UT*, 1976, 9-23.
- [26] I. Rosenberg, Completeness properties of multiple-valued logic algebras, in *Computer Science and Multiple-Valued Logic, ch. 6*, Elsevier, Amsterdam, 1977, 144-186.
- [27] I. Rosenberg, Minimal clones I: the five types, *Colloquia Mathematica Societatis J’anos Bolyai*, 43 (1986) 405-427.
- [28] I. Rosenberg, An algebraic approach to hyperalgebras, in *Proceedings of 26th IEEE International Symposium on Multiple-Valued Logic, Santiago de Compostela, Spain*, 1996, 203-207.
- [29] W. Sierpiński, *Sur les fonctions de plusieurs variables*, *Fundamenta Mathematicae*, 33 (1945) 169-173
- [30] 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.
- [31] A. Urquhart, Basic many-valued logic, in *Handbook of philosophical logic, vol. 2*, Kluwer, Dordrecht, 2001, 249-295.
- [32] A. Urquhart, Emil Post, in *Handbook of the history of logic, vol. 5, Logic from Russell to Church*, Elsevier, Amsterdam, 2009, 430-478.
- [33] P. Zusmanovich, On the last question of Stefan Banach, *Expositiones Mathematicae*, 34 (2016), no. 4, 454-466.