Prototyping Relational Specifications Using Higher-Order Objects

Rudolf Berghammer[1], Thomas F. Gritzner[2], Gunther Schmidt[3]

[1] Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität Kiel, D-24105 Kiel, Germany
[2] Fakultät für Informatik, Technische Universität München, D–80290 München, Germany
[3] Fakultät für Informatik, Universität der Bundeswehr München, D–85577 Neubiberg, Germany

Abstract. An approach is described for the generation of certain mathematical objects (like sets, correspondences, mappings) in terms of relations using relation-algebraic descriptions of higher-order objects. From non-constructive characterizations executable relational specifications are obtained. We also show how to develop more efficient algorithms from the frequently inefficient specifications within the calculus of binary relations.

1 Introduction

During the last two decades, the axiomatic relational calculus of Tarski [28] has widely been used by computer scientists who view it as a convenient formalism for describing fundamental concepts of programming. The development starts with the work of de Bakker and de Roever in the early 70's; see [12, 13] for example. In the following decade, e.g., Hoare and He [17] related the work of Birkhoff on residuals with Dijkstra's weakest precondition approach to programming, a group in Munich (see [23, 6, 32, 31]) constructed semantic domains by relation-algebraic means and, thus, was able to treat also languages with higher-order functions, and a group in Eindhoven (see [2]) developed a theory of data types based on the calculus of relations. At this point also the approach of a group in Rio should be mentioned which was motivated mainly by the development of a relational programming calculus not bounded by lack of expressiveness; compare [29]. In program development, the relational framework has already been used. E.g., in a series of articles a group around Desharnais and Mili proposed a relational approach to the formal derivation of imperative programs from its specifications. See [19, 20, 14] for example. Recently, Möller [21] used n-ary relations between nested tuples as elements of an applicative program development language, and also the Rio group developed various case studies on formal program construction using relations (cf. [30]).

In order that investigations with relation algebra involved do not stay completely on the theoretical side, several aspects need special consideration. Firstly, relational methods are not so commonly known that competent discussion with researchers from other approaches is easy. So, there should be a tool at hand to facilitate and visualize work with relations. Secondly, some tool for rapid prototyping of program specifications expressed in a relational style should be developed. And, finally, since relational specifications have a high degree of precision and formal structure, program development methods starting from such relational specifications should be considered.

In this paper, we describe an approach to the generation of certain mathematical objects (like sets, correspondences, mappings) in terms of relations using higher-order objects. We aim at the development of executable relational specifications out of non-constructive problem descriptions, where some special functionals on relations in conjunction with a relational description of domains (including relation and function spaces, i.e., higher-order objects) play an important rôle. We present only a carefully selected couple of representative examples; for an extensive treatment of our approach, we refer to the report [8] of the same title.

2 Relation Algebraic Preliminaries

In this section, we briefly introduce the basic concepts of the algebra of relations, some special relations, and some relation-algebraic constructions (i.e., functionals on relations). For more details concerning the algebraic theory of relations, see e.g., [11, 18, 25].

2.1 Basic Operations and Relation Algebraic Laws

For two sets X and Y, a subset R of the Cartesian product $X \times Y$ is a relation between the *domain* X and the *range* Y. We call it *homogeneous* if $X = Y$, otherwise we call it *heterogeneous.* Considering the corresponding characteristic predicates instead of the set representations, a relation R between X and Y becomes a function $R : X \times Y \to \mathbb{B}$, where $\mathbb{B}$ denotes the set $\{0, 1\}$ of truth-values. Therefore, if X and Y are finite and of cardinality m and n, respectively, then we may consider R as a Boolean matrix with m rows and n columns. This matrix interpretation of relations is well-suited for a graphical representation and, e.g., used within the RELVIEW system [1, 4]. Following the notation of the specification language Z [26], we write $R : X \leftrightarrow Y$ if R is a relation between the sets X and Y. Furthermore, we use matrix notation and write R_{xy} instead of $(x, y) \in R$.

We assume the reader to be familiar with the basic operations, viz. R^{T} (transposition), $\overline{R}$ (negation), $R \cup S$ (join), $R \cap S$ (meet), RS (composition), $R \subset S$ (inclusion), and the special relations O (empty relation), L (universal relation), and I (identity relation). In this paper, we only consider relations with non-empty domain and range, and therefore $\mathsf{O} : X \leftrightarrow Y$ and $\mathsf{L} : X \leftrightarrow Y$ are distinct. The set-theoretic operations $\overline{}$, $\cup$, $\cap$, the ordering $\subset$, and the constants O, L are related as usual. Some further well-known rules concerning relations are

$$
\begin{array}{ll}
(R^{\mathrm{T}})^{\mathrm{T}} = R & R \subset S \implies R^{\mathrm{T}} \subset S^{\mathrm{T}} \\
R^{\mathrm{T}} S^{\mathrm{T}} = (SR)^{\mathrm{T}} & \overline{R}^{\mathrm{T}} = \overline{R^{\mathrm{T}}} \\
R \subset S \implies QR \subset QS & R \subset S \implies RQ \subset SQ \\
Q(R \cap S) \subset QR \cap QS & Q(R \cup S) = QR \cup QS \\
(R \cap S)^{\mathrm{T}} = R^{\mathrm{T}} \cap S^{\mathrm{T}} & (R \cup S)^{\mathrm{T}} = R^{\mathrm{T}} \cup S^{\mathrm{T}} \ ,
\end{array}
$$

where the last two lines also hold if binary meet and join are replaced by arbitrary meet and join, respectively. The theoretical framework for all these rules to hold is that of an abstract relation algebra. The axioms of this algebraic structure are the axioms of a complete atomic Boolean algebra for $\overline{}$, $\cup$, $\cap$, $\subset$, O, L, the axioms of a monoid for composition and I, the *Dedekind rule*

$$(QR \cap S) \subset (Q \cap SR^{\mathrm{T}})(R \cap Q^{\mathrm{T}}S) \ ,$$

and the *Tarski rule*

$$R \neq \mathsf{O} \iff \mathsf{L}R\mathsf{L} = \mathsf{L} \ .$$

An alternative form of the Dedekind rule are the *Schröder equivalences*, viz.

$$QR \subset S \iff Q^{\mathrm{T}}\overline{S} \subset \overline{R} \iff \overline{S}R^{\mathrm{T}} \subset \overline{Q} \ .$$

2.2 Special Relations

The basic operations and constants of the last subsection are very helpful for defining simple properties on relations in a component-free manner. In the remainder of the paper we need the following special relations:

Functions: Let $R : X \leftrightarrow Y$ be a relation. R is said to be a *partial function* or, briefly, to be *functional* if $R^{\mathrm{T}}R \subset \mathrm{I}$, and R is said to be *total* if $R\mathsf{L} = \mathsf{L}$, which is, in turn, equivalent to $\mathrm{I} \subset RR^{\mathrm{T}}$. As usual, a functional and total relation is said to be a *(total) function*. A relation R is called *injective* if R^{T} is functional, and R is called *surjective* if R^{T} is total. An injective and surjective relation is said to be *bijective*.

Partial orderings: Let $Q : X \leftrightarrow X$ be homogeneous. Q is *reflexive* if $I \subset Q$, *transitive* if $QQ \subset Q$, and *antisymmetric* if $Q \cap Q^{\mathrm{T}} \subset \mathrm{I}$. A *partial ordering* is a reflexive, antisymmetric, and transitive relation. If Q is a partial ordering, then $Q \cap \overline{\mathrm{I}}$ denotes its *irreflexive part*.

Vectors: A relation $v : X \leftrightarrow Y$ with $v = v\mathsf{L}$ is called a *(row-constant) vector* or a *predicate*. This condition means that an element x from X is either in relation v to none of the elements of Y or to all elements of Y. Vectors may be considered as subsets of X. This becomes obvious if we use the common set-theoretic notation for relations, since then v equals a Cartesian product $X' \times Y$, where X' is a subset of X. As for a vector the range is without relevance, we consider in the following only vectors $v : X \leftrightarrow \mathbb{1}$ with a singleton set $\mathbb{1} = \{u\}$ as their range and write v_x instead of v_{xu}. Then v can be considered as a Boolean matrix with exactly one column, i.e., as a Boolean column vector, and describes the subset $\{x \in X : v_x\}$ of X.

2.3 Quotients and Bounds

In this subsection, we consider some special mappings from relations to relations. To distinguish such "meta-level functions" from the relation-algebraic (or: object level) notion of functions as presented in Subsection 2.2, they are also called *functionals*. In most cases, they happen to be partial functionals, since the basic operations on relations besides the unary ones are only partially defined.

Residuals: Residuals are the greatest solutions of certain inclusions. The *left residual* of S over R (in symbols $S \,/\, R$) is the greatest relation Q such that $QR \subset S$; the *right residual* of S over R (in symbols $R \backslash S$) is the greatest relation Q such that $RQ \subset S$. Both residuals may also be represented using the basic operations: Let $R : X \leftrightarrow Z$, $S : Y \leftrightarrow Z$, $R' : Z \leftrightarrow X$, and $S' : Z \leftrightarrow Y$ be given, then, from the Schröder equivalences we obtain

$$S \,/\, R = \overline{\overline{S} R^{\mathrm{T}}} \qquad R' \backslash S' = \overline{R'^{\mathrm{T}} \overline{S'}} \ .$$

Note also that the two residuals are linked together by the relationships

$$R \backslash S = (S^{\mathrm{T}} \,/\, R^{\mathrm{T}})^{\mathrm{T}} \qquad \overline{R} \backslash \overline{S} = R^{\mathrm{T}} \,/\, S^{\mathrm{T}} \ .$$

Translating the relation-algebraic expressions into component-wise formulations, for left residual $S \,/\, R : Y \leftrightarrow X$ and right residual $R' \backslash S' : X \leftrightarrow Y$ we have the equivalences

$$(S \,/\, R)_{yx} \iff \bigwedge_z R_{xz} \rightarrow S_{yz} \qquad (R' \backslash S')_{xy} \iff \bigwedge_z R'_{zx} \rightarrow S'_{zy} \ .$$

In particular, we have the two following correspondences for single universal quantification:

$$(S \,/\, \mathsf{L})_y \iff \bigwedge_z S_{yz} \qquad (\overline{R'} \backslash \mathsf{O})_x \iff \bigwedge_z R'_{zx} \ .$$

Note that for the quantifications the domain for z may also consist of higher-order objects. The usage of this option is characteristic to our approach.

Symmetric Quotients: In the following, we will frequently need relations which are left and right residuals simultaneously, viz. symmetric quotients. The *symmetric quotient* $\mathsf{syq}(R, S)$ of two relations R and S is defined as the greatest relation Q such that $RQ \subset S$ and $QS^{\mathrm{T}} \subset R^{\mathrm{T}}$. If $R : Z \leftrightarrow X$ and $S : Z \leftrightarrow Y$, then we have $\mathsf{syq}(R, S) : X \leftrightarrow Y$ as

$$\mathsf{syq}(R, S) = \overline{R^{\mathrm{T}} \, \overline{S}} \cap \overline{\overline{R^{\mathrm{T}}} S} = (R \backslash S) \cap (R^{\mathrm{T}} \,/\, S^{\mathrm{T}}) \ .$$

Using a component-wise notation, the symmetric quotient $\mathsf{syq}(R, S)$ satisfies the equivalence

$$\mathsf{syq}(R, S)_{xy} \iff \bigwedge_z R_{zx} \leftrightarrow S_{zy} \ .$$

Bounds and extremal elements: Let $Q : X \leftrightarrow X$ be a partial ordering. Due to later applications, we ask for some order-theoretic concepts such as the set of lower (resp. upper) bounds of a subset wrt. Q or the set of minimal (resp. maximal) elements of a subset wrt. Q. We define four functionals dependent on Q and a further relation $R : X \leftrightarrow Y$ as follows:

$$\begin{array}{ll} \textit{Lower bounds:} & \mathsf{mi}(Q, R) = \overline{\overline{Q}R} = Q \,/\, R^{\mathrm{T}} \\ \textit{Upper bounds:} & \mathsf{ma}(Q, R) = \overline{\overline{Q^{\mathrm{T}}}R} = Q^{\mathrm{T}} \,/\, R^{\mathrm{T}} \\ \textit{Minimal elements:} & \mathsf{min}(Q, R) = R \cap \overline{(Q \cap \overline{\mathrm{I}})^{\mathrm{T}} R} \\ \textit{Maximal elements:} & \mathsf{max}(Q, R) = R \cap \overline{(Q \cap \overline{\mathrm{I}}) R}\ . \end{array}$$

Looking at the corresponding component-wise descriptions and assuming R to be a vector, it is easy to see that $\mathsf{mi}(Q, R)$ (resp. $\mathsf{ma}(Q, R)$) yields the subset of lower (resp. upper) bounds of R wrt. the partial ordering Q, while $\mathsf{min}(Q, R)$ (resp. $\mathsf{max}(Q, R)$) computes to the subset of minimal (resp. maximal) elements of R wrt. the partial ordering Q. If R is not a vector, then the functionals compute bounds and extremal elements column-wise.

3 Relational Domain Construction

To deal with composed and higher-order objects like tuples, sets, or functions, we have to explain how the corresponding domain constructions can be performed with relational algebra. Note that the constructions described in the following, after the definitions of homomorphisms (resp. isomorphisms), may or may not exist in an arbitrary model of relational algebra; however, this problem does not occur at the concrete matrix model underlying this paper.

3.1 Homomorphisms, Direct Products, and Direct Sums

As first domain constructions we consider products and sums. Furthermore, we introduce homomorphisms to show that the characterizations are monomorphic.

Homomorphisms: Let $R : X \leftrightarrow Y$ and $S : X' \leftrightarrow Y'$ be two relations and consider a pair $\mathcal{H} = (\Phi, \Psi)$ of functions $\Phi : X \leftrightarrow X'$ and $\Psi : Y \leftrightarrow Y'$. $\mathcal{H}$ is called a *homomorphism* from R to S if $R \subset \Phi S \Psi^{\mathrm{T}}$ or, equivalently, $R\Psi \subset \Phi S$ holds. If, in addition, the pair $\mathcal{H}^{\mathrm{T}} = (\Phi^{\mathrm{T}}, \Psi^{\mathrm{T}})$ is a homomorphism from S to R, then $\mathcal{H}$ is said to be an *isomorphism* between R and S. Therefore, an isomorphism $\mathcal{I} = (\Phi, \Psi)$ between R and S is a pair of bijective functions $\Phi : X \leftrightarrow X'$ and $\Psi : Y \leftrightarrow Y'$, which satisfies the condition $R\Psi = \Phi S$. If R and S are homogeneous, then Φ is briefly called a homomorphism (isomorphism) if the pair (Φ, Φ) is a homomorphism (isomorphism).

Direct products: Within the framework of abstract relation algebra it is natural to characterize direct products by means of the natural projections, see [25, 31]. Then one obtains the following specification: Let

$$\mathcal{P} = (\pi_k : PX \leftrightarrow X_k)_{1 \le k \le n}$$

be an n-tuple of $n > 0$ relations. We call $\mathcal{P}$ an *(n-ary) direct product* if

$$(P_1)\ \ \pi_k^{\mathrm{T}}\pi_k = \mathrm{I} \qquad (P_2)\ \ j \neq k \implies \pi_j^{\mathrm{T}}\pi_k = \mathsf{L} \qquad (P_3)\ \ \bigcap_{k=1}^{n} \pi_k\pi_k^{\mathrm{T}} = \mathrm{I}\ .$$

It is easy to verify that the natural projections from a Cartesian product $\Pi_{i=1}^{n} X_i$ to the components X_k are a model of (P_1) through (P_3) if the placeholder PX is replaced by $\Pi_{i=1}^{n} X_i$. By purely relation-algebraic reasoning, furthermore, it can be shown that the direct product is uniquely characterized up to isomorphism: If $\mathcal{Q} = (\rho_k : PY \leftrightarrow Y_k)_{1 \leq k \leq n}$ is another model of (P_1) through (P_3) and $(\Psi_k : X_k \leftrightarrow Y_k)_{1 \leq k \leq n}$ is a family of bijective functions, then we can establish an isomorphism between π_k and ρ_k by the pair $\mathcal{I}_k = (\Phi, \Psi_k)$, where the bijective function $\Phi : PX \leftrightarrow PY$ is defined as $\Phi = \bigcap_{i=1}^{n} \pi_i \Psi_i \rho_i^{\mathrm{T}}$.

Based on the binary direct products $\mathcal{P} = (\pi, \rho)$ and $\mathcal{Q} = (\sigma, \tau)$ we define the following two functionals, where the generalization to n-ary direct products $(n > 2)$ is straightforward, but not needed in the remainder of the paper:

$$\begin{array}{ll} \textit{Tupling:} & [R, S]_{\mathcal{P}} = R\pi^{\mathrm{T}} \cap S\rho^{\mathrm{T}} \\ \textit{Parallel composition:} & R\ {}_{\mathcal{P}}\|_{\mathcal{Q}}\ S = \pi R\sigma^{\mathrm{T}} \cap \rho S\tau^{\mathrm{T}}\ . \end{array}$$

Direct sums: The direct sum can be defined in largely the same fashion as the direct product. Dually to the natural projections the natural injections are used, see [31]. Then one obtains the following specification: Let

$$\mathcal{S} = (\iota_k : X_k \leftrightarrow SX)_{1 \leq k \leq n}$$

be an n-tuple of $n > 0$ relations. We call $\mathcal{S}$ an *n-ary direct sum* if

$$(S_1)\ \ \iota_k\,\iota_k^{\mathrm{T}} = \mathrm{I} \qquad (S_2)\ \ j \neq k \implies \iota_j\,\iota_k^{\mathrm{T}} = \mathsf{O} \qquad (S_3)\ \ \bigcup_{k=1}^{n} \iota_k^{\mathrm{T}}\iota_k = \mathrm{I}\ .$$

Given sets X_k, $1 \leq k \leq n$, it is easy to verify that the injections from these sets to the direct sum $\Sigma_{k=1}^{n} X_k$ are a model of (S_1) through (S_3). Again by purely relation-algebraic reasoning it can be shown that by these laws the direct sum is uniquely characterized up to isomorphism and the relations ι_k, $1 \leq k \leq n$, are injective functions.

3.2 Powersets, Relation Spaces, and Function Spaces

Now, we present monomorphic characterizations of higher-order objects using relation algebra. The first construction formalizing the membership relation uses only a small set of set-theoretic axioms. The selection of this set of axioms such that it suffices for a monomorphic characterization has already been considered in category theory in connection with the notion of topos [9], which denotes a category such that every object has a power-object. The axiomatization presented here has been developed from the Munich group in the last decade, aiming at a relation-algebraic characterization of the kinds of function spaces used in denotational semantics. See [32, 7, 31]. Independently, an equivalent development based on the concept of topos mentioned before is provided in the book [15] by the notion of a power-allegory.

Powersets: A relation-algebraic characterization of the powerset 2^X of a set X can conveniently be done using the "is-element-of" relation. Formally, we call $\varepsilon : X \leftrightarrow PSX$ a *powerset relation* if

$$(PS_1)\quad \mathsf{syq}(\varepsilon, \varepsilon) \subset \mathrm{I} \qquad (PS_2)\quad \bigwedge_R \mathsf{L}\,\mathsf{syq}(\varepsilon, R) = \mathsf{L}\ .$$

Since every relation-algebraic equation using ε is translated into a formula with higher-order quantification (over sets), in (PS_2) the higher-order quantification (over relations) does not surprise. Again it can be shown by purely relation-algebraic reasoning that the powerset relation is uniquely characterized up to isomorphism. Indeed, if $\varepsilon' : Y \leftrightarrow PSY$ is another powerset relation, $\Phi : X \leftrightarrow Y$ is a bijective function, and one defines the bijective function $\Psi : PSX \leftrightarrow PSY$ by $\Psi = \mathsf{syq}(\varepsilon, \Phi\varepsilon')$, then $\mathcal{I} = (\Phi, \Psi)$ is an isomorphism between ε and ε'.

Now, assume the concrete case of the "is-element-of" relation $\varepsilon : X \leftrightarrow 2^X$. Then (PS_1) corresponds to the extensionality axiom, whereas (PS_2) says that every vector (set) $v : X \leftrightarrow \mathbb{1}$ has a corresponding point (i.e., a bijective vector) $\mathsf{vp}(v) := \mathsf{syq}(\varepsilon, v) : 2^X \leftrightarrow \mathbb{1}$ in the powerset. This shows that the usual "is-element-of" relation is a powerset relation. The functional vp is injective and its left-inverse on points is $\mathsf{vp}^{-1}(p) = \varepsilon p : X \leftrightarrow \mathbb{1}$. Hence, vp establishes some kind of isomorphism (resp. Galois connection) between subsets of X and elements of 2^X. For details, compare [32, 7].

Based on the relation $\varepsilon : X \leftrightarrow 2^X$, the relational specification of sets merely by equations can be established. Namely, we have:

Empty set:	$\mathbf{E} : 2^X \leftrightarrow \mathbb{1}$	$\mathbf{E} = \mathsf{syq}(\varepsilon, \mathsf{O}) = \varepsilon \setminus \mathsf{O}$
Universal set:	$\mathbf{U} : 2^X \leftrightarrow \mathbb{1}$	$\mathbf{U} = \mathsf{syq}(\varepsilon, \mathsf{L}) = \overline{\varepsilon} \setminus \mathsf{O}$
Singleton embedding:	$\mathbf{S} : X \leftrightarrow 2^X$	$\mathbf{S} = \mathsf{syq}(\mathrm{I}, \varepsilon)$
Complementation:	$\mathbf{C} : 2^X \leftrightarrow 2^X$	$\mathbf{C} = \mathsf{syq}(\varepsilon, \overline{\varepsilon})$
Meet of sets:	$\mathbf{M} : 2^X \times 2^X \leftrightarrow 2^X$	$\mathbf{M} = \mathsf{syq}([\varepsilon, \varepsilon]_{\mathcal{P}}, \varepsilon)$
Join of sets:	$\mathbf{J} : 2^X \times 2^X \leftrightarrow 2^X$	$\mathbf{J} = (\mathbf{C}\ {}_{\mathcal{P}}\|_{\mathcal{P}}\ \mathbf{C})\mathbf{M}\mathbf{C}$
Inclusion of sets:	$\sqsubseteq : 2^X \leftrightarrow 2^X$	$\sqsubseteq = \varepsilon \setminus \varepsilon$

In the description of the binary operations meet and join we use a direct product $\mathcal{P} = (\pi, \rho)$, consisting of the two projections from $2^X \times 2^X$ to the first and the second component, respectively.

Relation spaces and function spaces: We are also interested in describing the set of all relations between two sets and some certain subsets by relation-algebraic means. The set of all relations between X and Y is easy to handle, since it is a powerset relation $\varepsilon_R : X \times Y \leftrightarrow 2^{X \times Y}$. Hence, $\mathcal{R} = (\pi, \rho, \varepsilon_R)$ is called a *relation space* if

(R_1) (π, ρ) is a direct product
(R_2) $\pi^{\mathrm{T}}\varepsilon_R$ and $\rho^{\mathrm{T}}\varepsilon_R$ exist
(R_3) ε_R is a powerset relation.

The transformation of a relation $R : X \leftrightarrow Y$ into a point of the relation space is described by $\mathsf{vp}(\mathsf{rv}(R))$, where $\mathsf{rv}(R) := (\pi R \cap \rho)\mathsf{L} : X \times Y \leftrightarrow \mathbb{1}$ is the vector corresponding to R; the opposite direction uses vp^{-1} and the left-inverse of rv which is $\mathsf{rv}^{-1}(v) = \pi^{\mathrm{T}}(\rho \cap v\mathsf{L}) : X \leftrightarrow Y$.

The set Y^X of all functions from X to Y may be characterized in two different ways. On one hand, it can be described by a vector $f : 2^{X \times Y} \leftrightarrow \mathbb{1}$ such that

$$
\begin{aligned}
f_r \iff & \bigwedge_x \bigwedge_y \bigwedge_{y'} (((x,y) \in r \wedge (x,y') \in r) \rightarrow y = y') \wedge \bigwedge_x \bigvee_y (x,y) \in r \\
\iff & \neg(\bigvee_t t \in r \wedge \bigvee_{t'} (t' \in r \wedge \pi(t') = \pi(t) \wedge \rho(t') \neq \rho(t))) \\
& \wedge \bigwedge_x \bigvee_t (t \in r \wedge \pi(t) = x)
\end{aligned}
$$

giving the intricate expression

$$
f = \overline{[\varepsilon_R^{\mathrm{T}} \cap \varepsilon_R^{\mathrm{T}}(\pi\pi^{\mathrm{T}} \cap \overline{\rho\rho^{\mathrm{T}}})]\mathsf{L}} \;\cap\; (\varepsilon_R^{\mathrm{T}}\pi \,/\, \mathsf{L}) \;\; .
$$

On the other hand, we can define an "is-element-of" relation $\varepsilon_F : X \times Y \leftrightarrow Y^X$ as follows: The triple $\mathcal{F} = (\pi, \rho, \varepsilon_F)$ is called a *function space* if (R_1) and (R_2) (numbered by (F_1) and (F_2), respectively) hold with ε_R replaced by ε_F and, furthermore,

$$
\begin{array}{ll}
(F_3) & \mathsf{syq}(\varepsilon_F, \varepsilon_F) \subset \mathrm{I} \\
(F_4) & \bigwedge_R \mathsf{L}\,\mathsf{syq}(\varepsilon_F, R) = \mathsf{L} \leftrightarrow (RR^{\mathrm{T}} \subset \overline{\pi\pi^{\mathrm{T}}} \cup \rho\rho^{\mathrm{T}} \wedge R^{\mathrm{T}}\pi = \mathsf{L}) \;\; .
\end{array}
$$

Now, for R being a vector $v : X \times Y \leftrightarrow \mathbb{1}$ the second condition (F_4) means that the point $\mathsf{vp}(v) = \mathsf{syq}(\varepsilon_F, v) : 2^{X \times Y} \leftrightarrow \mathbb{1}$ is in the function domain Y^X if and only if v represents a function, i.e., the relation $\mathsf{rv}^{-1}(v) : X \leftrightarrow Y$ is a function. For details, see [1].

Injections: In addition to vectors, we have injective functions as a second concept for representing subsets. Given an injective function $\imath : X' \rightarrow X$, we call X' a *subset of X given by $\imath$.* Clearly, if X' is a subset of X given by $\imath$, then the vector $\imath^{\mathrm{T}}\mathsf{L} : X \leftrightarrow \mathbb{1}$ describes X' in the sense of Subsection 2.2. Since we deal only with concrete relations, the transition in the other direction, i.e., the construction of an injective function $\imath : X' \rightarrow X$ from a given vector $v : X \leftrightarrow \mathbb{1}$, is also possible. Generally, we have: Let the vector $v : X \leftrightarrow \mathbb{1}$ describe the subset X' of the set X. Then, $\imath(v) : X' \leftrightarrow X$ is called an *injection of v* or an *injection of X' into X* if

$$
(I_1) \;\; \imath(v) \text{ is an injective function} \qquad (I_2) \;\; v = \imath(v)^{\mathrm{T}}\mathsf{L} \;\; .
$$

Clearly, it follows that X' is a subset of X given by $\imath(v)$. Again, it can be shown that injections are determined uniquely up to isomorphism by (I_1) and (I_2). Namely, if $\Psi : X \leftrightarrow Y$ is a bijective function and $v' : Y \leftrightarrow \mathbb{1}$ is a vector such that it describes a subset Y' of Y and $v = \Psi v'$ is satisfied (i.e., $v' = \Psi^{\mathrm{T}} v$), then $\mathcal{I} = (\Phi, \Psi)$ is an isomorphism between $\imath(v)$ and $\imath(v')$, where the bijective function $\Phi : X' \leftrightarrow Y'$ is given by $\Phi = \imath(v)\Psi\imath(v')^{\mathrm{T}}$.

In most cases, injections are used within our applications in the context of higher-order objects like sets of sets. Namely, if the vector $v : 2^X \leftrightarrow \mathbb{1}$ describes a subset S of sets, then it is straightforward to compute an injection $\imath(v)$ of S into 2^X. From $\imath(v)$ we obtain the elements of S (represented as vectors) as the columns of the relation $\varepsilon\imath(v)^{\mathrm{T}} : X \leftrightarrow S$, which leads to an economic representation of the set S of sets by a Boolean matrix.

4 Relational Problem Specification and Prototyping

In this section, we show how the computation of certain mathematical objects (like sets, correspondences, mappings) can be described in terms of relations. The general method is as follows. First, we specify the problem with the help of a formula φ which characterizes the objects to be computed. Using the correspondences between certain kinds of formulae and certain relation-algebraic constructions (resp. operations), we then transform φ into a component-free relational expression R such that φ is valid if and only if its free variables are related by R. Hence, this expression R can be seen as a relational problem specification (cf. [5]) which is executable as it stands, i.e., as an algorithm for computing the set of specified objects. At this place it should be mentioned that in easy cases or for people well-trained in the relational calculus the specification R can be written down immediately.

Of course, algorithms produced in the way just described may frequently be fairly inefficient compared to hand-made ones and, thus, in many cases only applicable to small or medium-sized examples. However, they are built up very quickly, which is an important factor of economy. Furthermore, they ensure correctness and their proofs of correctness are very simple. Moreover, an executable relational specification can be the starting point for the derivation of an efficient algorithm using some development method as we will show in Section 5. Hence, we have the typical situation of the rapid prototyping approach (see [10]) for a validation and analysis of specifications.

Most of the following examples of rapid prototyping using a relation-algebraic description of the given problem are borrowed from graph theory. They deal with the computation (strictly speaking: the enumeration) of higher-order objects like sets of sets (represented as vectors $v : 2^X \leftrightarrow \mathbb{1}$) or sets of functions (represented as vectors $v : Y^X \leftrightarrow \mathbb{1}$). Here our approach leads to an extensive use of the relational characterization of sets and functions as presented in Section 3. Therefore, in the following, we have to distinguish between the two meta-level symbols $\in$ and $\subseteq$ and the "is-element-of" relation and the set inclusion relation on the object-level. As in Section 3, we use in the sequel the two relations $\varepsilon : X \leftrightarrow 2^X$ and $\sqsubseteq : 2^X \leftrightarrow 2^X$ on the object level. Especially, we have $x \in s$ (resp. $s \subseteq t$) if and only if the relation ε_{xs} (resp. $\sqsubseteq_{st}$) holds.

Here, only a carefully selected couple of representative examples is presented. Further examples may be found in the report [8], so the computation of further point sets of a directed graph (like strongly connected components, point bases, or hammocks), point/edge coverings of graphs of various kinds, and so on.

4.1 Kernels

Let $\mathcal{G} = (V, B)$ be a *directed graph*, i.e., V a non-empty set of points (also: vertices, nodes) and $B : V \leftrightarrow V$ a relation between points. Furthermore, assume $\varepsilon : V \leftrightarrow 2^V$ to be the "is-element-of" relation between V and its powerset 2^V and $\sqsubseteq : 2^V \leftrightarrow 2^V$ to be the inclusion relation between point sets.

A set $a \in 2^V$ of points is said to be *absorbant* in $\mathcal{G}$ if from every point outside of a there is at least one arc leading into a, i.e., if the first-order formula

$$\bigwedge_x x \in a \ \vee \ \bigvee_y (B_{xy} \wedge y \in a)$$

holds. Furthermore, a set $s \in 2^V$ of points is called *stable* in $\mathcal{G}$ if no two points of s are related via the relation B. This situation is characterized by the first-order formula

$$\bigwedge_x x \in s \rightarrow \bigwedge_y (y \in s \rightarrow \overline{B}_{xy}) \ .$$

Finally, a *kernel* $k \in 2^V$ of $\mathcal{G}$ is a set which is at the same time absorbant and stable. The concept of a kernel plays an import rôle in combinatorial games; for an overview see e.g., [24] and [25], Sections 8.2 and 8.3.

Expressing the above two formulae in terms of the operations on relations introduced in Section 2, we get that the first one is equivalent to $(\mathsf{L} \setminus (\varepsilon \cup B\varepsilon))^{\mathrm{T}}_a$ and the second one is equivalent to $((\overline{\varepsilon \cap B\varepsilon}) \setminus \mathsf{O})_s$. In the second case, for instance, we express $\bigwedge_y (y \in s \rightarrow \overline{B}_{xy})$ as $(\overline{B\varepsilon})_{xs}$ and obtain, thus, the original formula in the form $\bigwedge_x (\overline{\varepsilon \cap B\varepsilon})_{xs}$. Now, the universal quantification can be removed using a right residual construction, cf. Section 2. Summing up, in a component-free notation, we have

$$absorb(B) = (\mathsf{L} \setminus (\varepsilon \cup B\varepsilon))^{\mathrm{T}}$$

as the vector $absorb(B) : 2^V \leftrightarrow \mathbb{1}$ describing the absorbant sets of $\mathcal{G}$ (where $\mathsf{L} : V \leftrightarrow \mathbb{1}$) and

$$stable(B) = (\overline{\varepsilon \cap B\varepsilon}) \setminus \mathsf{O}$$

as the vector $stable(B) : 2^V \leftrightarrow \mathbb{1}$ of the stable sets of $\mathcal{G}$ (where $\mathsf{O} : V \leftrightarrow \mathbb{1}$). Finally, the vector $kernel(B) : 2^V \leftrightarrow \mathbb{1}$ describing the elements of 2^V which are kernels of $\mathcal{G}$ is given by

$$kernel(B) = absorb(B) \cap stable(B) \ .$$

4.2 Dedekind Cuts

Now, we deal with an order-theoretic problem; for a more visualized treatment of this example, compare [3]. Let $\mathcal{O} = (M, Q)$ be a partially ordered set, i.e., M be a non-empty set of points and $Q : M \leftrightarrow M$ be an ordering relation on points. Furthermore, assume again $\varepsilon : M \leftrightarrow 2^M$ to be the "is-element-of" relation between M and its powerset 2^M and $\sqsubseteq : 2^M \leftrightarrow 2^M$ to be the inclusion relation on point sets.

For an element $s \in 2^M$, let $Ma(s)$ denote its upper bounds wrt. Q and $Mi(s)$ denote its lower bounds wrt. Q. Then, $c \in 2^M$ is called a *Dedekind cut* of $\mathcal{O}$ if the equation $c = Mi(Ma(c))$ is valid, i.e, if the first-order formula

$$\bigwedge_x x \in c \leftrightarrow x \in Mi(Ma(c))$$

holds which in turn is equivalent to

$$\bigvee_s \bigwedge_x (x \in c \leftrightarrow x \in Mi(Ma(s))) \wedge c = s \ .$$

Obviously, for each element $x \in M$ the set $(x) = \{y \in M : Q_{yx}\}$ is a cut, called the *principal cut* generated by x. The fact that a set $p \in 2^M$ is a principal cut, hence, is described by the first-order formula

$$\bigvee_x x \in p \wedge \bigwedge_y (y \in p \leftrightarrow Q_{yx}) \ .$$

Now, let $\mathcal{C} \subseteq 2^M$ denote the set of cuts of $\mathcal{O}$ and $\mathcal{P} \subseteq 2^M$ denote the set of principal cuts of $\mathcal{O}$. Furthermore, let $\sqsubseteq_{\mathcal{C}} : \mathcal{C} \leftrightarrow \mathcal{C}$ and $\sqsubseteq_{\mathcal{P}} : \mathcal{P} \leftrightarrow \mathcal{P}$ denote the restrictions of set inclusion to the cuts and principal cuts, respectively. Then $\mathcal{O}' = (\mathcal{C}, \sqsubseteq_{\mathcal{C}})$ is a complete lattice, called the *cut completion* of $\mathcal{O}$, and the function $emb(Q) : M \to \mathcal{C}$ mapping x to the principal cut (x) is an injective order homomorphism.

Using abstract relation algebra and the above formulae (for the characterization of cuts the second version is more suited since it immediately leads to a symmetric quotient construction $\mathsf{syq}(\varepsilon, \ldots)_{cs}$) in combination with the functionals mi, ma, and syq of Section 2.3, by

$$cut(Q) = (\mathsf{syq}(\varepsilon, \mathsf{mi}(Q, \mathsf{ma}(Q, \varepsilon))) \cap \mathrm{I})\mathsf{L}$$

(where $\mathsf{L} : 2^M \leftrightarrow \mathbb{1}$ and $\mathrm{I} : 2^M \leftrightarrow 2^M$) we obtain the vector $cut(Q) : 2^M \leftrightarrow \mathbb{1}$ describing the elements of 2^M which are Dedekind cuts, and by

$$pricut(Q) = (\varepsilon^{\mathrm{T}} \cap \mathsf{syq}(\varepsilon, Q))\mathsf{L}$$

(where $\mathsf{L} : M \leftrightarrow \mathbb{1}$) we get the vector $pricut(Q) : 2^M \leftrightarrow \mathbb{1}$ describing the elements of 2^M which are principal cuts. Since the cuts are ordered by set inclusion, we consider the relation $\sqsubseteq : 2^M \leftrightarrow 2^M$ and the injection $\imath(cut(Q)) : \mathcal{C} \leftrightarrow 2^M$ in the sense of Subsection 3.2 and receive for $\sqsubseteq_{\mathcal{C}}$ the representation

$$\sqsubseteq_{\mathcal{C}} = \imath(cut(Q)) \sqsubseteq \imath(cut(Q))^{\mathrm{T}} \ .$$

Also the function $emb(Q) : M \leftrightarrow \mathcal{C}$ (in the relational sense) can be computed with the help of $\imath(cut(Q))$. From the component-wise definition of $emb(Q)_{xc}$ by the second-order formula

$$\bigwedge_y Q_{yx} \leftrightarrow \bigvee_s (y \in s \wedge \imath(cut(Q))_{cs})$$

we obtain $\bigwedge_y Q_{yx} \leftrightarrow (\varepsilon \imath(cut(Q))^{\mathrm{T}})_{yc}$ and, thus, an application of the functional syq yields the component-free relation-algebraic representation

$$emb(Q) = \mathsf{syq}(Q, \varepsilon \imath(cut(Q))^{\mathrm{T}}) \ .$$

4.3 Sets of Places of a Petri Net

A *Petri net* is a bipartite directed graph $\mathcal{N} = (X, Y, R, S)$. The point set of $\mathcal{N}$ is decomposed into the sets X of *places* and Y of *transitions* and the associated relations are $R : X \leftrightarrow Y$ and $S : Y \leftrightarrow X$. Petri nets have been widely used to design and model concurrent systems, see [22] for example. Many of the static properties of a Petri net (e.g., to be free-choice, to be conflict-free, or to contain specific sets of places/transitions) can be tested using our relational approach. In the following we show, how to compute specific sets of places. In doing so, $\varepsilon_X : X \leftrightarrow 2^X$ denotes the "is-element-of" relation between the places and the sets of places.

A set $d \in 2^X$ of places is called a *deadlock* if each of its predecessors is also a successor. A somewhat dual notion is that of a trap: $t \in 2^X$ is said to be a *trap* if its successor set is a subset of its predecessor set. Both, deadlocks and traps are of interest if one is concerned with liveness properties. E.g., a transition never can be enabled if its predecessor set contains an unmarked place of a deadlock.

Expressed by first-order formulae, we have that a set d of places is a deadlock if and only if

$$\bigwedge_y (\bigvee_x x \in d \wedge S_{yx}) \rightarrow (\bigvee_x x \in d \wedge R_{xy})$$

is valid and that a set t of places is a trap if and only if

$$\bigwedge_y (\bigvee_x x \in t \wedge R_{xy}) \rightarrow (\bigvee_x x \in t \wedge S_{yx})$$

holds. Using the operations on relations, the first formula becomes

$$\bigwedge_y (\varepsilon_X^{\mathrm{T}} S^{\mathrm{T}})_{dy} \rightarrow (\varepsilon_X^{\mathrm{T}} R)_{dy} \quad , \text{ i.e. } \quad \bigwedge_y (\overline{\varepsilon_X^{\mathrm{T}} S^{\mathrm{T}}} \cup \varepsilon_X^{\mathrm{T}} R)_{dy} \quad .$$

Hence, using a left residual construction and the universal relation $\mathsf{L} : Y \leftrightarrow \mathbb{1}$ we get in a component-free notation

$$\mathit{deadlock}(R, S) = (\overline{\varepsilon_X^{\mathrm{T}} S^{\mathrm{T}}} \cup \varepsilon_X^{\mathrm{T}} R)/\mathsf{L}^{\mathrm{T}} = (S\varepsilon_X \cap \overline{R^{\mathrm{T}}\varepsilon_X}) \setminus \mathsf{O}$$

as the vector $\mathit{deadlock}(R, S) : X \leftrightarrow \mathbb{1}$ enumerating all deadlocks of the net $\mathcal{N}$. In the same way one obtains the vector $\mathit{trap}(R, S) : X \leftrightarrow \mathbb{1}$ describing all traps of $\mathcal{N}$ by exchanging the rôle of R and S^{T}, i.e., by

$$\mathit{trap}(R, S) = (\overline{\varepsilon_X^{\mathrm{T}} R} \cup \varepsilon_X^{\mathrm{T}} S^{\mathrm{T}})/\mathsf{L}^{\mathrm{T}} = (R^{\mathrm{T}}\varepsilon_X \cap \overline{S\varepsilon_X}) \setminus \mathsf{O} \quad .$$

It may also be of interest to compute minimal non-empty deadlocks or traps. To this end, we assume $\sqsubseteq \, : 2^X \leftrightarrow 2^X$ to be the inclusion relation. By the min functional and the $\mathbf{E}$ vector, we obtain the minimized version of $op \in \{\mathit{deadlock}, \mathit{trap}\}$ from the expression $\mathsf{min}(\sqsubseteq, op(R, S) \cap \overline{\mathbf{E}})$.

4.4 Dicliques

In the preceding examples we have dealt with sets of points and edges, i.e., elements of the powerset of a given “base set”. The example of this subsection shows how to describe the computation of pairs of elements of the powerset of a given “base set” with relation-algebraic means, i.e., the computation of a relation (correspondence) on a powerset. We consider again a directed graph $\mathcal{G} = (V, B)$ and suppose the “is-element-of” relation $\varepsilon : V \leftrightarrow 2^V$ and the inclusion relation $\sqsubseteq : 2^V \leftrightarrow 2^V$ on point sets.

A pair $(d, c) \in 2^V \times 2^V$ is called a *block* of $\mathcal{G}$ with domain d and co-domain c if the product $d \times c$ is contained in B. The (non-trivial) *dicliques* of $\mathcal{G}$ (this term is introduced in [16]) are the inclusion-maximal blocks with non-empty domains and co-domains. In other words, (d, c) is a diclique if and only if it generates an inclusion-maximal complete bipartite subgraph of $\mathcal{G}$. A decomposition of a graph into its dicliques can be very useful e.g., for storing it in a computer or for determining the essential subsystems of the system it describes. See again [16].

The description of a block $(d, c) \neq (\emptyset, \emptyset)$ by a first-order formula is

$$(\bigvee_x x \in d) \wedge (\bigvee_x x \in c) \wedge (\bigwedge_x x \in d \rightarrow \bigwedge_y (y \in c \rightarrow B_{xy})) \ .$$

Obviously, $\bigwedge_y y \in c \rightarrow B_{xy}$ is equivalent to $(\overline{\overline{B}\varepsilon})_{xc}$, i.e., to $(B/\varepsilon^{\mathrm{T}})_{xc}$, which implies the equivalence of $\bigwedge_x x \in d \rightarrow \bigwedge_y (y \in c \rightarrow B_{xy})$ and $(\varepsilon \setminus (B/\varepsilon^{\mathrm{T}}))_{dc}$. This immediately leads to

$$block(B) = \varepsilon^{\mathrm{T}}\mathsf{L} \cap \mathsf{L}^{\mathrm{T}}\varepsilon \cap (\varepsilon \setminus (B/\varepsilon^{\mathrm{T}})) = \overline{\mathbf{E}} \cap \overline{\mathbf{E}}^{\mathrm{T}} \cap \overline{\varepsilon^{\mathrm{T}}\overline{B}\varepsilon}$$

(where $\mathsf{L} : V \leftrightarrow 2^V$) as the relation $block(B) : 2^V \leftrightarrow 2^V$ of the blocks with non-empty domain and co-domain. I.e., $block(B)_{dc}$ holds if and only if $(d, c) \neq (\emptyset, \emptyset)$ is a block. To describe dicliques, we use that for a relation $R : 2^V \leftrightarrow 2^V$ and a pair $(s, t) \in 2^V \times 2^V$ we have $\mathsf{max}(\sqsubseteq, R)_{st}$ if and only if s is inclusion-maximal in the set $\{s' \in 2^V : R_{s't}\}$. Hence, a two-fold maximalization via the functional max yields

$$diclique(B) = \mathsf{max}(\sqsubseteq, \mathsf{max}(\sqsubseteq, block(B))^{\mathrm{T}})^{\mathrm{T}}$$

as the relation $diclique(B) : 2^V \leftrightarrow 2^V$ of the dicliques of the directed graph $\mathcal{G}$.

Now, let $D \subseteq 2^V$ (resp. $C \subseteq 2^V$) be the set of domains (resp. co-domains) of the dicliques of $\mathcal{G}$. Then we have the two injections

$$\imath(diclique(B)\mathsf{L}) : D \leftrightarrow 2^V \qquad \imath(diclique(B)^{\mathrm{T}}\mathsf{L}) : C \leftrightarrow 2^V$$

for embedding D and C, respectively, into 2^V. Based on these injections, finally, we are able to define a relation $Diclique(B) : D \leftrightarrow C$ describing the correspondence between the domain and the co-domain of a diclique by

$$Diclique(B) = \imath(diclique(B)\mathsf{L})diclique(B)\imath(diclique(B)^{\mathrm{T}}\mathsf{L})^{\mathrm{T}} \ .$$

This means: A pair $(d, c) \in D \times C$ is a diclique of $\mathcal{G}$ if and only if $Diclique(B)_{cd}$ holds.

4.5 Homomorphisms

This example will show how to describe the computation of sets of functions from a set V to a set W with relation-algebraic means. As already mentioned, we consider the set W^V of functions from V to W as the set of the functional and total relations between V and W, i.e., as a subset of the powerset $2^{V \times W}$. Therefore, suppose $\varepsilon_F : V \times W \leftrightarrow W^V$ to be the "is-element-of" relation between $V \times W$ and the set W^V.

Let $\mathcal{G} = (V, B)$ and $\mathcal{H} = (W, C)$ be directed graphs and $\Phi \in 2^{V \times W}$ be a function in the relational sense (see Subsection 2.2). Furthermore, assume (π, ρ) to be the projections of the direct product $V \times W$ to the first and the second component, respectively. If we use the common function notation for π and ρ, the fact that Φ is a *homomorphism* means exactly that the first-order formula

$$\bigwedge_u u \in \Phi \rightarrow \bigwedge_v (v \in \Phi \rightarrow (B_{\pi(u)\pi(v)} \rightarrow C_{\rho(u)\rho(v)}))$$

is valid; see Subsection 3.1. Now, we use the relational notation also for the two projections π and ρ, i.e., write $(\pi B \pi^{\mathrm{T}})_{uv}$ (resp. $(\rho C \rho^{\mathrm{T}})_{uv}$) instead of $B_{\pi(u)\pi(v)}$ (resp. $C_{\rho(u)\rho(v)}$). This leads to the equivalent version

$$\bigwedge_u u \in \Phi \rightarrow \bigwedge_v (v \in \Phi \rightarrow (\overline{\pi B \pi^{\mathrm{T}}} \cup \rho C \rho^{\mathrm{T}})_{uv}) \enspace .$$

Hence, we have again the pattern of the formula defining a set to be stable and, therefore, we get the vector $hom(B, C) : W^V \leftrightarrow 1\!\!1$ describing the homomorphisms from B to C as

$$hom(B, C) = (\varepsilon_F \cap \overline{\overline{\pi B \pi^{\mathrm{T}}} \cup \rho C \rho^{\mathrm{T}} \varepsilon_F}) \setminus \mathsf{O} = (\varepsilon_F \cap (\pi B \pi^{\mathrm{T}} \cap \overline{\rho C \rho^{\mathrm{T}}}) \varepsilon_F) \setminus \mathsf{O} \enspace ,$$

where the typing of the empty relation is $\mathsf{O} : V \times W \leftrightarrow 1\!\!1$.

5 Development of Efficient Algorithms

The execution of a specification produced in the way described in Section 4, frequently may be fairly inefficient. In this subsection we demonstrate by means of an example how to develop more efficient algorithms from the original inefficient specifications using the relational calculus.

We consider again the problem of computing the kernels of a directed graph $\mathcal{G} = (V, B)$. In contrast with Subsection 4.1, however, in the following we do not consider sets as points $p : 2^V \leftrightarrow 1\!\!1$ but as vectors $v : V \leftrightarrow 1\!\!1$. In doing so, $x \in p$ will be replaced by v_x. So, the first-order formulae of Subsection 4.1 defining absorbant and stable sets become

$$\bigwedge_x \overline{a}_x \rightarrow \bigvee_y (a_y \wedge B_{xy}) \qquad\qquad \bigwedge_x s_x \rightarrow \bigwedge_y (s_y \rightarrow \overline{B}_{xy}) \enspace .$$

Translating these formulae into a notation without components, we get the inclusions $\overline{a} \subset Ba$ and $s \subset \overline{Bs}$. As a consequence, a vector $k : V \leftrightarrow 1\!\!1$ is a kernel of $\mathcal{G}$ if and only if $\overline{k} = Bk$, i.e., if and only if it is a fixpoint of the functional $\tau(v) = \overline{Bv}$.

This example shows also that the change of set-representation does not eliminate the use of higher-order structures. Instead of the relation ε, in the new specification now a functional τ mapping vectors to vectors is used. However, for specific classes of graphs the fixpoint specification enables the development of efficient algorithms as will be shown now.

5.1 Progressively Finite Graphs

The functional τ just defined is antitonic, so the fixpoint theorem for monotone functions on complete lattices cannot be applied. We therefore study the fixpoints of $\tau^2(v) := \tau(\tau(v)) = \overline{B\overline{Bv}}$ which is monotonic. Suppose m_{τ^2} and M_{τ^2} to denote the least resp. greatest fixpoint of τ^2. Then we have for each kernel k that

$$\mathsf{O} \subset \tau^2(\mathsf{O}) \subset \tau^4(\mathsf{O}) \subset \ldots \subset m_{\tau^2} \subset k \subset M_{\tau^2} \subset \ldots \subset \tau^4(\mathsf{L}) \subset \tau^2(\mathsf{L}) \subset \mathsf{L} \ .$$

Also the two equations

$$\text{(i)} \ \ \tau(m_{\tau^2}) = M_{\tau^2} \qquad \text{(ii)} \ \ \tau(M_{\tau^2}) = m_{\tau^2}$$

easily can be shown. Hence, if the gap between the lower bound m_{τ^2} and the upper bound M_{τ^2} of the set of all kernel closes, then the uniquely determined fixpoint is a kernel of $\mathcal{G}$:

Theorem. *If the functional τ^2 has exactly one fixpoint (which is equivalent to $M_{\tau^2} \subset \tau(M_{\tau^2})$ or to $\tau(m_{\tau^2}) \subset m_{\tau^2}$), then $\mathcal{G}$ has precisely one kernel.* □

Using this theorem, for instance, it can be shown that a progressively finite graph $\mathcal{G} = (V, B)$ (i.e., a graph in which all paths have finite lengths) has exactly one kernel. When specifying progressive finiteness relationally, we obtain

$$(*) \quad \bigwedge_v (v = v\mathsf{L} \wedge v \subset Bv \rightarrow v = \mathsf{O}) \ .$$

Now, if we use the Schröder equivalences, we obtain $B^{\mathrm{T}} M_{\tau^2} \subset \overline{BM_{\tau^2}}$ from $M_{\tau^2} \subset \tau^2(M_{\tau^2})$. Next, the Dedekind rule yields

$$\begin{aligned} BM_{\tau^2} \cap M_{\tau^2} &\subset (B \cap M_{\tau^2} M_{\tau^2}^{\mathrm{T}})(M_{\tau^2} \cap B^{\mathrm{T}} M_{\tau^2}) \\ &\subset B(M_{\tau^2} \cap B^{\mathrm{T}} M_{\tau^2}) \\ &\subset B(M_{\tau^2} \cap \overline{BM_{\tau^2}}) \end{aligned}$$

and, finally, in combination with $(*)$ we obtain $BM_{\tau^2} \cap M_{\tau^2} = \mathsf{O}$, which implies $M_{\tau^2} \subset \tau(M_{\tau^2})$. Hence, the functional τ has precisely one fixpoint.

For the point set being finite, we have that a directed graph is progressively finite if and only if it is circuit-free. Therefore, we can compute the only kernel of a finite circuit-free graph $\mathcal{G} = (V, B)$ by the iteration $\mathsf{O} \subset \tau^2(\mathsf{O}) \subset \tau^4(\mathsf{O}) \subset \ldots$ which takes at most $|V|$ steps.

At this place it should be mentioned that in the case of a transitive relation B the iteration stops after one step. This is due to the fact that in the case $B = B^+$ and $B = B^* \overline{B\mathsf{L}}$ (this latter condition follows from $(*)$ and means that from each point there is a path to a terminal one) the only kernel of $\mathcal{G}$ is the least absorbant set which equals the set $\overline{B\mathsf{L}} : V \leftrightarrow \mathbb{1}$ of all terminal points:

$$\tau^2(\mathsf{O}) = \overline{B\overline{B\mathsf{O}}} = \overline{B\mathsf{L}} = \overline{B\overline{B\mathsf{L}}} = \tau^2(\mathsf{L})$$

follows from $B\overline{B\mathsf{L}} \subset B\mathsf{L}$ and $\mathsf{L} = B^*\overline{B\mathsf{L}} = \overline{B\mathsf{L}} \cup B^+\overline{B\mathsf{L}} = \overline{B\mathsf{L}} \cup B\overline{B\mathsf{L}}$ which in turn is equivalent to $B\mathsf{L} \subset B\overline{B\mathsf{L}}$.

5.2 Bipartite Graphs

Now, assume $\mathcal{G} = (X, Y, R, S)$ to be a bipartite graph in the sense of Subsection 4.3, i.e., we have $R : X \leftrightarrow Y$ and $S : Y \leftrightarrow X$. Furthermore let $\iota : X \leftrightarrow X + Y$ and $\kappa : Y \leftrightarrow X + Y$ be the natural injections into the binary direct sum $X + Y$. Then

$$\mathcal{H} = (V, B), \text{ where } V := X + Y \text{ and } B := \iota^{\mathrm{T}} R \kappa \cup \kappa^{\mathrm{T}} S \iota \ ,$$

is the "ordinary" directed graph corresponding to $\mathcal{G}$. Generalizing the above technique of the composition of the antitone functional τ with itself to pairs of antitone functionals and using the laws of the direct sum, in the following we will show that $\mathcal{H}$ has at least the kernels according to the following two characterizations. To this end, we consider the functionals $\alpha(v) = \overline{Rv}$ and $\beta(w) = \overline{Sw}$ and obtain for the least and greatest fixpoints of the compositions

$$\begin{array}{ll} \text{(i) } \alpha(M_{\beta\circ\alpha}) = m_{\alpha\circ\beta} & \text{(ii) } \alpha(m_{\beta\circ\alpha}) = M_{\alpha\circ\beta} \\ \text{(iii) } \beta(M_{\alpha\circ\beta}) = m_{\beta\circ\alpha} & \text{(iv) } \beta(m_{\alpha\circ\beta}) = M_{\beta\circ\alpha} \ . \end{array}$$

If we use (only for explanatory purposes) 2×2-matrices and 2-vectors with relations and vectors, respectively, as coefficients, then we have B as matrix

$$B = \begin{pmatrix} \mathsf{O} & R \\ S & \mathsf{O} \end{pmatrix}$$

and obtain two kernels of $\mathcal{H}$ by the vectors

$$k_1 = \begin{pmatrix} M_{\alpha\circ\beta} \\ m_{\beta\circ\alpha} \end{pmatrix} \qquad\qquad k_2 = \begin{pmatrix} m_{\alpha\circ\beta} \\ M_{\beta\circ\alpha} \end{pmatrix} \ .$$

A component-free proof of this fact based on (i) through (iv) and on the relational characterization of the direct sum is given in the following. Define two vectors

$$\begin{array}{l} k_1 := \iota^{\mathrm{T}} M_{\alpha\circ\beta} \cup \kappa^{\mathrm{T}} m_{\beta\circ\alpha} : X + Y \leftrightarrow \mathbb{1} \\ k_2 := \iota^{\mathrm{T}} m_{\alpha\circ\beta} \cup \kappa^{\mathrm{T}} M_{\beta\circ\alpha} : X + Y \leftrightarrow \mathbb{1} \ . \end{array}$$

Then, we obtain the equations $Bk_1 = \overline{k_1}$ and $Bk_2 = \overline{k_2}$. E.g., a proof of $Bk_1 = \overline{k_1}$ proceeds as follows: We use an immediate consequence of (S_1) through (S_3), viz. $\iota^{\mathrm{T}}\mathsf{L} = \overline{\kappa^{\mathrm{T}}\mathsf{L}}$, and obtain

$$\kappa^{\mathrm{T}}\mathsf{L} = \overline{\iota^{\mathrm{T}}\mathsf{L}} \subset \overline{\iota^{\mathrm{T}} M_{\alpha\circ\beta}} \qquad \iota^{\mathrm{T}}\mathsf{L} = \overline{\kappa^{\mathrm{T}}\mathsf{L}} \subset \overline{\kappa^{\mathrm{T}} m_{\beta\circ\alpha}} \ .$$

From the axioms of the direct sum we get also that both ι^{T} and κ^{T} are partial functions and, due to Proposition 4.2.2.v of [25] (saying that $Q\overline{R} = \overline{QR} \cap Q\mathsf{L}$ for Q being a partial function), we have

$$\iota^{\mathrm{T}}\overline{M_{\alpha\circ\beta}} = \overline{\iota^{\mathrm{T}}M_{\alpha\circ\beta}} \cap \iota^{\mathrm{T}}\mathsf{L} \qquad \kappa^{\mathrm{T}}\overline{m_{\beta\circ\alpha}} = \overline{\kappa^{\mathrm{T}}m_{\beta\circ\alpha}} \cap \kappa^{\mathrm{T}}\mathsf{L} \ .$$

Now, we combine these properties and arrive at

$$\begin{aligned}
\iota^{\mathrm{T}}\overline{M_{\alpha\circ\beta}} \cup \kappa^{\mathrm{T}}\overline{m_{\beta\circ\alpha}} &= (\overline{\iota^{\mathrm{T}}M_{\alpha\circ\beta}} \cap \iota^{\mathrm{T}}\mathsf{L}) \cup (\overline{\kappa^{\mathrm{T}}m_{\beta\circ\alpha}} \cap \kappa^{\mathrm{T}}\mathsf{L}) \\
&= (\overline{\iota^{\mathrm{T}}M_{\alpha\circ\beta}} \cup \overline{\kappa^{\mathrm{T}}m_{\beta\circ\alpha}}) \cap (\overline{\iota^{\mathrm{T}}M_{\alpha\circ\beta}} \cup \kappa^{\mathrm{T}}\mathsf{L}) \\
&\quad \cap (\iota^{\mathrm{T}}\mathsf{L} \cup \overline{\kappa^{\mathrm{T}}m_{\beta\circ\alpha}}) \cap (\iota^{\mathrm{T}}\mathsf{L} \cup \kappa^{\mathrm{T}}\mathsf{L}) \\
&= (\overline{\iota^{\mathrm{T}}M_{\alpha\circ\beta}} \cup \overline{\kappa^{\mathrm{T}}m_{\beta\circ\alpha}}) \cap \overline{\iota^{\mathrm{T}}M_{\alpha\circ\beta} \cap \kappa^{\mathrm{T}}m_{\beta\circ\alpha}} \\
&= \overline{\iota^{\mathrm{T}}M_{\alpha\circ\beta}} \cap \overline{\kappa^{\mathrm{T}}m_{\beta\circ\alpha}} \ ,
\end{aligned}$$

which in turn implies the desired result as follows:

$$\begin{aligned}
Bk_1 &= (\iota^{\mathrm{T}}R\kappa \cup \kappa^{\mathrm{T}}S\iota)(\iota^{\mathrm{T}}M_{\alpha\circ\beta} \cup \kappa^{\mathrm{T}}m_{\beta\circ\alpha}) \\
&= \iota^{\mathrm{T}}Rm_{\beta\circ\alpha} \cup \kappa^{\mathrm{T}}SM_{\alpha\circ\beta} && \text{axioms of the direct sum} \\
&= \iota^{\mathrm{T}}\overline{\alpha(m_{\beta\circ\alpha})} \cup \kappa^{\mathrm{T}}\overline{\beta(M_{\alpha\circ\beta})} \\
&= \iota^{\mathrm{T}}\overline{M_{\alpha\circ\beta}} \cup \kappa^{\mathrm{T}}\overline{m_{\beta\circ\alpha}} && \text{due to (ii) and (iii)} \\
&= \overline{\iota^{\mathrm{T}}M_{\alpha\circ\beta}} \cap \overline{\kappa^{\mathrm{T}}m_{\beta\circ\alpha}} \\
&= \overline{\iota^{\mathrm{T}}M_{\alpha\circ\beta} \cup \kappa^{\mathrm{T}}m_{\beta\circ\alpha}} \\
&= \overline{k_1} \ .
\end{aligned}$$

Altogether, we have shown the following generalization of Richardson's theorem an early version of which goes back to [27].

Theorem. *If $\mathcal{G}$ is a bipartite graph, then the corresponding directed graph $\mathcal{H}$ has (not necessarily distinct) kernels k_1 and k_2.* □

Namely, these two kernels can be computed with the help of the iterations

$$\mathsf{O} \subset \alpha\circ\beta(\mathsf{O}) \subset \alpha\circ\beta(\alpha\circ\beta(\mathsf{O})) \subset \ldots \qquad \mathsf{O} \subset \beta\circ\alpha(\mathsf{O}) \subset \beta\circ\alpha(\beta\circ\alpha(\mathsf{O})) \subset \ldots$$

to obtain $m_{\alpha\circ\beta}$ and $m_{\beta\circ\alpha}$, since by applying (iv) and (ii), respectively, from above we can obtain the missing $M_{\beta\circ\alpha}$ and $M_{\alpha\circ\beta}$.

Continuing the preceding treatment, the report [8] also deals with the evaluation of graph games, where moves are one-step walks along the edges, due to loss, draw (by infinite repetition), and win: loss is obtained as the greatest stable set and win is described as the complement of the smallest absorbant set.

6 Concluding Remarks

In this paper we have described a rapid prototyping approach for the enumeration of certain mathematical objects in terms of relations. This has lead to an extensive use of a relational characterization of higher-order objects like sets, sets of sets, or functions. We have also shown how to develop more efficient algorithms from inefficient specifications within the abstract relational framework.

Let us close with a few remarks about the execution of relational specifications. All examples given in this paper have been performed using RELVIEW, a totally interactive and completely video-oriented computer system for the manipulation of concrete relations which are considered as Boolean matrices (see [1, 4]). The system does not only provide commands implementing the basic operations on relations, but also commands for residuals, quotients, and closures, commands for certain tests on relations, and commands which implement the operations important in relation-algebraic domain description (direct product, direct sum, injections from vectors, powersets, and function spaces). And, finally, RELVIEW allows the user to define and apply his own functionals on relations, where in the case of a unary functional with identical domain and range even repeated application is possible by an iteration command. A useful fact in applications is that the latter command can be used to compute fixpoints of monotone functionals, as for the efficient computations of Section 5. Of course, computation with RELVIEW is limited in space and time. The limit, however, depends heavily on the type of problem handled. In general, it is difficult to treat the powerset of a set or function spaces since this means the computation of vectors $v : 2^V \leftrightarrow \mathbb{1}$ or $v : W^V \leftrightarrow \mathbb{1}$ and consumes a lot of space and time. However, the handling of relations $R : V \leftrightarrow V$, vectors $v : V \times V \leftrightarrow 1$, or vectors $v : V \leftrightarrow \mathbb{1}$ is of such a small complexity that admits a wide range of RELVIEW applications. E.g., on our installation (SUN SPARCstation 10) we have treated relations with domain/range up to 5000 elements. As the computation of kernels shows, it seems promising to apply the relational calculus to obtain efficient algorithms from inefficient specifications.

Acknowledgement: We wish to thank the referees for their helpful comments and hints.

References

1. Abold-Thalmann H., Berghammer R., Schmidt G.: Manipulation of concrete relations: The RELVIEW-system. Report Nr. 8905, Fakultät für Informatik, Universität der Bundeswehr München (1989)
2. Backhouse R.C., Hoogendijk P., Voermans E., van der Woude J.C.S.P: A relational theory of datatypes. Eindhoven University of Technology, Dept. of Mathematics and Computer Science (1992)
3. Berghammer R.: Computing the cut completion of a partially ordered set – An example for the use of the RELVIEW-system. Report Nr. 9205, Fakultät für Informatik, Universität der Bundeswehr München (1992)
4. Berghammer R., Schmidt G.: The RELVIEW-system. In: Choffrut C., Jantzen M. (eds.): Proc. STACS '91, Lect. Notes Comput. Sci. 480, Springer, 535–536 (1991)
5. Berghammer R., Schmidt G.: Relational specifications. In: Rauszer C. (ed.): Algebraic Methods in Logic and Computer Science, Banach Center Publications, Volume 28, Institute of Mathematics, Polish Academy of Sciences, 167–190 (1993)
6. Berghammer R., Zierer H.: Relational algebraic semantics of deterministic and nondeterministic programs. Theoret. Comput. Sci. 43, 123–147 (1986)

7. Berghammer R., Schmidt G., Zierer H.: Symmetric quotients and domain constructions. Inform. Proc. Letters 33, 3, 163–168 (1989/90)
8. Berghammer R., Gritzner T.F., Schmidt G.: Prototyping relational specifications using higher-order objects. Report Nr. 9304, Fakultät für Informatik, Universität der Bundeswehr München (1993)
9. Brook T.: Order and recursion in topoi. Notes on Pure Mathematics, Vol. 9, Australian National University of Canberra (1977)
10. Budde R., Kuhlenkamp K., Matthiassen H., Züllinghoven H. (eds.): Approaches to prototyping. Springer (1984)
11. Chin L.H., Tarski A.: Distributive and modular laws in the arithmetic of relation algebras. University of California Publications in Mathematics (new series) 1, 341–384 (1951)
12. De Bakker J.W., de Roever W.P.: A calculus for recursive program schemes. In: Nivat M. (ed.): Proc. ICALP 73, North-Holland, 167–196 (1973)
13. De Roever W.P.: Recursion and parameter mechanisms: An axiomatic approach. In: Loeckx J. (ed.): Proc. ICALP 74, Lect. Notes Comput. Sci. 14, Springer, 34-65 (1974)
14. Desharnais J., Jaoua A., Mili F., Boudriga N., Mili A.: Conjugate kernels: An operator for program construction. Theoret. Comput. Sci., to appear
15. Freyd P.J., Ščedrov A.: Categories, allegories. Mathematical Library, Vol. 39, North-Holland (1990)
16. Haralick R.M.: The diclique representation and decomposition of binary relations. J. ACM 21, 3, 356–366 (1974)
17. Hoare C.A.R., He J.: The weakest prespecification, Parts I&II, Fundamenta Informaticae IX, 51–84 & 217–252 (1986)
18. Jónsson B., Tarski A.: Boolean algebras with operators, Part II. Amer. J. Math. 74, 127–167 (1952)
19. Mili A.: A relational approach to the design of deterministic programs. Acta Informatica 20, 315–328 (1983)
20. Mili A., Desharnais J., Mili F.: Relational heuristics for the design of deterministic programs. Acta Informatica 24, 239–276 (1987)
21. Möller B.: Relations as a program development language. In: Möller B. (ed.): Proc. IFIP TC2/WG2.1 Working Conference on Constructing Programs from Specifications, North-Holland, 373–397 (1991)
22. Reisig W.: Petri nets – An introduction. EATCS Monographs on Theoret. Comput. Sci., Springer (1985)
23. Schmidt G.: Programs as partial graphs I: Flow equivalence and correctness. Theoret. Comput. Sci. 15, 1–25 (1981)
24. Schmidt G., Ströhlein T.: On kernels of graphs and solutions of games: A synopsis based on relations and fixed points. SIAM J. Alg. Disc. Meth. 6,1, 54–65 (1985)
25. Schmidt G., Ströhlein T.: Relationen und Graphen. Springer (1989); English version: Relations and graphs. Discrete Mathematics for Computer Scientists, EATCS Monographs on Theoret. Comput. Sci., Springer (1993)
26. Spivey J.M.: The Z notation: A reference manual. Prentice Hall (1989)
27. Ströhlein T.: Untersuchungen über kombinatorische Spiele. Doctoral Thesis, Technische Universität München (1970)
28. Tarski A.: On the calculus of relations. Journal of Symbolic Logic 6, 73–89 (1941)
29. Veloso P., Haeberer A.: A finitary relational algebra for classical first-order logic. Bull. of the Section on Logic of the Polish Academy of Sciences 20, 52–62 (1991)

30. Veloso P., Haeberer A., Baum G.: Formal program construction within an extended calculus of binary relations. J. Symbolic Comp., to appear
31. Zierer H.: Relation algebraic domain constructions. Theoret. Comput. Sci. 87, 163–188 (1991)
32. Zierer H., Schmidt G., Berghammer R.: An interactive graphical manipulation system for higher objects based on relation algebra. In: Tinhofer G., Schmidt G. (eds.): Proc. 12th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 86), Bernried/Starnberger See, 17.6.-19.6. 1986, Lect. Notes Comput. Sci. 246, Springer: Berlin-Heidelberg-New York, 68-81 (1987)