ALGEBRAIC METHODS IN LOGIC AND IN COMPUTER SCIENCE
BANACH CENTER PUBLICATIONS, VOLUME 28
INSTITUTE OF MATHEMATICS
POLISH ACADEMY OF SCIENCES
WARSZAWA 1993

RELATIONAL SPECIFICATIONS

RUDOLF BERGHAMMER and GUNTHER SCHMIDT

Fakultät für Informatik, Universität der Bundeswehr München
Werner-Heisenberg-Weg 39, D-8014 Neubiberg, Germany

1. Introduction. In the last years the relational calculus of Tarski has widely been used by computer scientists who view it as a convenient formalism for describing fundamental concepts of programming languages. In this paper, abstract relation algebra is proposed as a practical means for specification of data types and programs. We demonstrate that the relational calculus allows also a completely formal treatment of this topic. Furthermore, as it is based on a small set of axioms, a supporting (of course not an automatic) computer system can be implemented (cf. [Kern 87], [Brethauer 91]) and, hence, the manipulations can even be checked with computer assistance.

The idea itself of characterizing data types by relational axioms is not new. The paper [de Bakker–de Roever 73] seems to be the first one containing relational specifications of data types. Further examples can be found in, e.g., [de Roever 74], [Zierer 83], [Schmidt 84], [Berghammer–Zierer 86], [Desharnais 89] and [Desharnais–Madhavji 90]. However, all those papers have in common that they present only specific examples of relational specifications and do not treat this topic in a more general manner.

The first objective of this paper is to define the concept of a relational specification by transferring some fundamental notions of the algebraic specification approach (see, e.g., [Ehrig–Mahr 85], or [Wirsing 90]) to the relational case. This is done in Section 3 after a short introduction to abstract relation algebra in Section 2. The second and main objective of the paper is to demonstrate the usefulness of the relational approach and to convey an idea of relational calculations in the field of specifications. This is done in the remainder of the paper.

1991 *Mathematics Subject Classification*: 03B70, 03G15, 68Q60, 68Q65.
The paper is in final form and no version of it will be published elsewhere.

2. Relation algebraic preliminaries. In the following we briefly recall some fundamental concepts of abstract relation algebra. Further details can be found in textbooks about relation algebra, e.g. in [Schmidt–Ströhlein 89].

2.1. *Abstract relation algebra.* A (*heterogeneous*) *relation algebra* is an algebraic structure $(B, \cup, \cap, \overline{}, \cdot, {}^{\mathrm{T}})$ over a nonempty set B of elements, called (*heterogeneous*) *relations*. Every relation R belongs to a subset B_R of B such that these sets form a partition of B and the following holds:

(1) The structure $(B_R, \cup, \cap, \overline{})$ is a complete atomistic Boolean algebra with *join* $\cup$, *meet* $\cap$, *negation* $\overline{}$, *inclusion* $\subset$, *null* (or least) *element* O, and *universal* (or greatest) *element* L.

(2) *Multiplication* (or *composition*) of relations—in the remainder of the paper simply indicated by juxtaposition—is a partial associative operation and the existence of a product RS implies that QS is defined for all relations $Q \in B_R$. There exist right and left identities for every set B_R of relations, which, for simplicity, are all denoted by I.

(3) For each R, there exist the *transposed relation* R^{T} and the products $R^{\mathrm{T}}R$ and RR^{T}.

(4) The Schröder equivalences $QR \subset S \Leftrightarrow Q^{\mathrm{T}}\overline{S} \subset \overline{R} \Leftrightarrow \overline{S}R^{\mathrm{T}} \subset \overline{Q}$ are valid whenever one of the three inclusions is defined.

(5) For every relation $R \neq O$ the equation $LRL = L$ is satisfied (the Tarski rule).

If the product $RR \in B_R$ exists, then R is called *homogeneous*. An immediate consequence of the Schröder equivalences is the *Dedekind rule* stating that for all relations Q, R, S

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

holds, whenever one of the three parenthesized expressions is defined. With the help of the Schröder equivalences and/or the Dedekind rule the well-known rules of the relation calculus can be derived by elementary reasoning (see, e.g., [Chin–Tarski 51], [Schmidt–Ströhlein 89]).

According to the above definition, elements of an abstract relation algebra are not endowed with a domain and a range. In the case of a concrete relation, however, we find it useful to mention these sets. Since we consider a concrete relation as a Boolean matrix, we write $\mathbb{B}^{X \times Y}$ for the set of all subsets of $X \times Y$. (This matrix interpretation is well suited for a graphical representation and also used within the RELVIEW system; see [Berghammer–Schmidt 91].)

2.2. *Partial functions and mappings.* A relation R is said to be a (*partial*) *function* (or *univalent* or *functional*) if $R^{\mathrm{T}}R \subset I$. In the case $RL = L$ (or equivalently $I \subset RR^{\mathrm{T}}$), the relation R will be called *total.* A total function (also called a *mapping*) may be characterized by $R\overline{I} = \overline{R}$ (or equivalently by $R\overline{S} = \overline{RS}$ for arbitrary relations S). A relation R is *injective* if R^{T} is a partial function (i.e., $RR^{\mathrm{T}} \subset I$) and *surjective* if R^{T} is total (i.e., $R^{\mathrm{T}}L = L$). Usually, an injective and surjective relation is said to be *bijective*. Note that these notions are defined for arbitrary relations, not only for mappings.

2.3. *Vectors and points*. A relation v with $v = vL$ is called *row constant*. In the case $v \in \mathbb{B}^{X \times Y}$ this condition means: Whatever set Z and universal relation $L \in \mathbb{B}^{Y \times Z}$ we choose, an element $x \in X$ is either in relation vL to none of the elements $z \in Z$ or to all elements $z \in Z$. Relations of this kind may be considered as subsets of the set X, predicates on X, or *vectors*. As for a vector v from $\mathbb{B}^{X \times Y}$ the set Y is irrelevant, we denote by $\mathbf{V}(X)$ the *set of all vectors* with domain X.

A *point* (element) is a subset containing exactly one element. Hence, a vector is a point if and only if it is bijective: surjectivity means that it describes a set containing at least one element and injectivity means that it describes a set containing at most one element. If p is a point, then $p \neq O$, $p \cap p^{\mathrm{T}} = pp^{\mathrm{T}}$, and $p^{\mathrm{T}}p = L$. $\mathbf{P}(X)$ denotes the *set of all points* with domain X.

Sometimes, the Tarski rule is not part of the axioms for a relation algebra. In this case a bijective vector does not have the given meaning that it describes a set containing exactly one element. To describe without the Tarski rule such a set by a vector v, instead of surjectivity of v it has to be postulated that vv^{T} is an atom (see [Gritzner 89]).

2.4. *Homomorphisms and isomorphisms.* Let R and S be relations. A pair $(\varPsi, \varPhi)$ of relations is called a *homomorphism* from R to S if $\varPsi$ and $\varPhi$ are mappings and $R \subset \varPsi S \varPhi^{\mathrm{T}}$ holds. An equivalent version of this postulate is $R\varPhi \subset \varPsi S$, which is equivalent to $\varPsi^{\mathrm{T}} R \varPhi \subset S$ and to $\varPsi^{\mathrm{T}} R \subset S \varPhi^{\mathrm{T}}$. Clearly, the composition $(\varPsi_1 \varPsi_2, \varPhi_1 \varPhi_2)$ of two homomorphisms $(\varPsi_1, \varPhi_1)$ and $(\varPsi_2, \varPhi_2)$ is also a homomorphism.

A homomorphism $(\varPsi, \varPhi)$ from R to S is called an *isomorphism* between R and S if the pair $(\varPsi^{\mathrm{T}}, \varPhi^{\mathrm{T}})$ is also a homomorphism (from S to R). An isomorphism $(\varPsi, \varPhi)$ between R and S is thus characterized by the conditions of $\varPsi$ and $\varPhi$ being bijective mappings and the equation $R\varPhi = \varPsi S$ to hold.

2.5. *Closures*. Assume R to be a homogeneous relation. Then the least transitive relation containing R is called its *transitive closure* R^+ and the least reflexive and transitive relation containing R is called its *reflexive-transitive closure* R^*. Obviously, $R^+ = \bigcup_{n \geq 1} R^n$ and $R^* = \bigcup_{n \geq 0} R^n$.

3. Relational specifications. In this section we transfer some fundamental notions of the algebraic specification approach to the relational case. For reasons of brevity, we confine ourselves to the non-hierarchical case.

3.1. *Syntactic aspects*: *signatures and axioms.* A (*relational*) *signature* $\varSigma = (T, F)$ consists of a non-empty set T of types and a set F of relation symbols. Both sets are disjoint and the set F is the union of the pairwise disjoint sets

- K_m, the set of constant symbols of type $m \in T$, and
- $F_{m,n}$, the set of (proper) relation symbols with functionality $(m, n) \in T \times T$.

The set K_m will always include O_m and L_m and the set $F_{m,n}$ will always include $O_{m,n}$ and $L_{m,n}$ and, in the case of $m = n$, also $I_{m,m}$. In the sequel subscripts are omitted from these distinguished symbols to enhance readability, provided there is no risk of confusion.

Let $\Sigma = (T, F)$ be a signature. Then a relational Σ-term is constructed from the relation symbols of F and certain variables (for relations) using relational operator symbols for the five operations (join, meet, negation, multiplication, and transposition) from Section 2.1. Every Σ-term is endowed with a functionality $(m, n) \in T \times T$ or a type $m \in T$. (The typing rules are obvious and, thus, omitted.) Now, an atomic relational Σ-formula is an inclusion $t_1 \subset t_2$ or an equation $t_1 = t_2$, where t_1 and t_2 are relational Σ-terms of the same functionality resp. type. From atomic formulae we obtain relational Σ-formulae in the usual way using the logical symbols of predicate calculus. Finally, a (*heterogeneous*) *relational specification* (Σ, A) consists of a signature Σ and a set A of closed relational Σ-formulae (called *axioms* or *laws*).

For reasons of simplicity, the difference between syntax and semantics will not be made explicit, i.e., no difference in notation will be made between a relational operator symbol (as part of the syntax) and its corresponding relational operation (as part of the semantics). In the remainder of the paper we use the following syntactical representation:

spec ≫identifier≪ ≡

types ≫list of types≪

rels ≫list of constant/relation symbols each with a type/functionality≪

laws ≫list of relational formulae≪

Furthermore, we use the notation $r : m$ (resp. $r : m \rightarrow n$) to indicate that m is the type of the constant symbol r (resp. (m, n) is the functionality of the relation symbol r). To enhance readability, in the sequel we will sometimes also use English sentences as axioms.

3.2. *Semantic aspects*: *relational structures and models*. Assume $\Sigma = (T, F)$ to be a signature. Then a *relational Σ-structure* S is given by two families $(m_S)_{m \in T}$ and $(r_S)_{r \in F}$, where the m_S are non-empty sets (called *base sets* or *domains*) and the r_S are (concrete) relations such that

- $r_S \in \mathbf{V}(m_S)$ if and only if r is a constant symbol of type m,
- $r_S \in \mathbb{B}^{m_S \times n_S}$ if and only if r is a relation symbol with functionality (m, n).

In the remainder of this paper we consider only relational Σ-structures S in which the interpretation of the special relation symbols O, L, and I agrees with the intuition, i.e., is given by $O_S := O$, $L_S := L$, and $I_S := I$.

The *value* $\mathbf{v}_{S,v}(t)$ of a relational Σ-term t in a given relational Σ-structure S w.r.t. a valuation v of the variables in S is inductively defined as follows: The induction base is given by $\mathbf{v}_{S,v}(r) = r_S$ and $\mathbf{v}_{S,v}(x) = v(x)$ for all relation symbols r and all variables x. For composed terms the value is defined by

$$\mathbf{v}_{S,v}(t_1 t_2) = \mathbf{v}_{S,v}(t_1)\mathbf{v}_{S,v}(t_2)\,, \qquad \mathbf{v}_{S,v}(t_1 \cup t_2) = \mathbf{v}_{S,v}(t_1) \cup \mathbf{v}_{S,v}(t_2)\,,$$
$$\mathbf{v}_{S,v}(t_1 \cap t_2) = \mathbf{v}_{S,v}(t_1) \cap \mathbf{v}_{S,v}(t_2)\,, \qquad \mathbf{v}_{S,v}(t^{\mathrm{T}}) = (\mathbf{v}_{S,v}(t))^{\mathrm{T}}\,, \qquad \mathbf{v}_{S,v}(\overline{t}) = \overline{\mathbf{v}_{S,v}(t)}\,.$$

Note that $\mathbf{v}_{S,v}(t)$ is a relation from $\mathbb{B}^{m_S \times n_S}$ if (m,n) is the functionality of t and a vector from $\mathbf{V}(m_S)$ if m is the type of t. If t is a closed relational term, then $\mathbf{v}_{S,v}(t)$ does not depend on the valuation v. In this case we use the notation $\mathbf{v}_S(t)$ instead.

Let S be a relational Σ-structure and v a valuation of the variables in S. Then we define the *validity* of atomic formulae by

$$t_1 \subset t_2 \text{ is valid in } S \text{ w.r.t. } v \text{ if and only if } \mathbf{v}_{S,v}(t_1) \subset \mathbf{v}_{S,v}(t_2),$$
$$t_1 = t_2 \text{ is valid in } S \text{ w.r.t. } v \text{ if and only if } \mathbf{v}_{S,v}(t_1) = \mathbf{v}_{S,v}(t_2)$$

and, based on this definition, the validity of general relational formulae in S w.r.t. v as usual. Now, a *model* of the specification (Σ, A) is a relational Σ-structure which satisfies all Σ-formulae from the set A for all valuations. The (loose) *semantics* of a specification is the class of all models. Note that we do not restrict ourselves to those models which are term-generated (in contrast with some approaches in the case of algebraic specifications; see [Wirsing 90]).

Assume S and S' to be two relational Σ-structures. Furthermore, let $(\Phi_m)_{m\in T}$ be a family of relations (the index set is the set of types of Σ) such that m_S and $m_{S'}$ are the domain resp. the range of each Φ_m. Then the family is called a *Σ-homomorphism* from S to S' if each member is a mapping and if the inclusions

$$\text{(i) } c_S \subset \Phi_m c_{S'}\,, \qquad \text{(ii) } r_S \Phi_n \subset \Phi_m r_{S'}$$

hold for all constant symbols c of type m and all relation symbols r of functionality (m,n). This notion corresponds exactly to what is called a loose element-valued Σ-homomorphism in the algebraic approach (see, e.g., [Hussmann 89]).

If, in addition, each member of a Σ-homomorphism is a bijective mapping and if equality holds in (i) and (ii), then the family is called a *Σ-isomorphism* between S and S'. Two relational Σ-structures are called *isomorphic* if there exists a Σ-isomorphism between them. A specification is said to be *monomorphic* if any pair of models is isomorphic.

Because of the totality of Φ_n and of the vector property of $c_{S'}$, $c \in K_m$, we have

$$c_S \subset \Phi_m c_{S'} \Leftrightarrow c_S \subset \Phi_m c_{S'} (\Phi_n L)^{\mathrm{T}} \Leftrightarrow c_S \subset \Phi_m c_{S'} L \Phi_n^{\mathrm{T}} \Leftrightarrow c_S \Phi_n \subset \Phi_m c_{S'}\,.$$

Thus, the family $(\Phi_m)_{m\in T}$ is a Σ-homomorphism from S to S' if and only if for each pair $m,n \in T$ and all $r \in F$ of type m resp. functionality (m,n) the pair (Φ_m, Φ_n) is a relational homomorphism from r_S to $r_{S'}$. In the same way one shows that $(\Phi_m)_{m\in T}$ is a Σ-isomorphism between S and S' if and only if for all m, n, and r as above the pair (Φ_m, Φ_n) is a relational isomorphism between the relations r_S and $r_{S'}$.

3.3. *Extensions of relational specifications.* A signature Σ' is called a *sub-signature* of a signature Σ if every type of Σ' is also a type of Σ and the same holds for the relation symbols. A relational specification (Σ', A') is called a *sub-specification* of the relational specification (Σ, A) if Σ' is a sub-signature of Σ and A' is contained in A. (Σ, A) is said to be an *extension* of (Σ', A'). In our syntactical representation, for extensions of specifications we use the notation

spec ≫identifier≪ ≡ ≫identifier′≪ ⊕

≫types, constant/relation symbols, and relational formulae of the rest≪

where ≫identifier′≪ is the name of the representation of the sub-specification.

3.4. *Parameterized relational specifications.* Using the concept of sub-specification, we can also define what it means for a specification to be parameterized: A *parameterized relational specification* is a relational specification (Σ, A) a sub-specification (Σ_P, A_P) of which is marked as the *formal parameter specification.* It is called monomorphic if any pair S and S' of models is isomorphic provided their reductions to the parameter signature Σ_P are (as models of the parameter part) isomorphic.

In the remainder of the paper we use the key words **param** and **target** in our syntactical representation in order to distinguish between the parameter and the non-parameter part. Furthermore, instantiation of the parameters of a specification in combination with a renaming of the types and the constant/relation symbols of the non-parameter part is denoted by

include ≫identifier≪[≫argument list≪] **as** [≫list of new names≪]

Semantically, this notation is explained by textual substitution of the non-parameter part of ≫identifier≪ with the name of each parameter replaced by the name of the respective argument and each non-parameter renamed according to the list of new names (call-by-specification parameter passing mechanism; cf. [Wirsing–Broy 82]). In particular, for a specification M in which an instantiation **include** $N[\dots]$ **as** $[\dots]$ occurs we require that the formulae of the parameter part of N (with the parameters replaced by the arguments) hold in all models of M.

3.5. *Second-order formulae vs. second-order terms.* Let $t^x_{t'}$ denote the replacement of the variable x by the term t' in the term t. Generally, a relational axiomatization of the fact that every model of a specification is finitely generated is expressed by a formula like

$$t^x_L = L \wedge \forall y\, (t^x_y = y \rightarrow L \subset y)\,,$$

where x is the only variable in t and every of its occurrences is contained only within an even number of subterms of the form $\overline{t'}$. Provided (m, n) is the functionality of the relational term t, this formula is valid in a relational structure S w.r.t. a valuation v if and only if the least fixed point μ_τ of the monotonic *functional corresponding to* t, i.e., the functional

$$\tau : \mathbb{B}^{m_S \times n_S} \to \mathbb{B}^{m_S \times n_S} , \qquad \tau(X) := \mathbf{v}_{S,v(x|X)}(t) ,$$

equals the universal relation L on m_S. (Here $v(x|X)$ is the function which is exactly like the valuation v except for one variable. At x it assumes the value X.) Due to this fact, in the sequel we will use the more readable version $L = \inf\{x : t = x\}$. In general, we use the expression $r = \inf\{x : t = x\}$ as a shorthand for

$$t^x_r = r \wedge \forall y \, (t^x_y = y \to r \subset y) .$$

A different approach is taken in [de Bakker 71]. In that paper, second-order terms, the so-called μ-terms, are introduced which model recursive procedures. Using our notation, a μ-term is of the form $\inf\{x : t = x\}$, where x and t are as above. Its value is the least fixed point of the functional τ defined as above. Thus, in de Bakker's approach it often (but not always, cf. the relational description of the powerset given in Section 4.4) suffices for the axioms of a specification to be of the form $t_1 \subset t_2$ or $t_1 = t_2$, where t_1 and t_2 are relational terms in the sense of 3.1 or μ-terms.

4. Relational calculations in the field of specifications. In this section we present some relational specifications. By means of these examples we also want to exhibit some advantages of the relational approach and to convey an idea of relational calculations in the field of specifications. In particular, we want to demonstrate the usefulness of second-order formulae as axioms.

4.1. *Proofs of monomorphy and totality*. Consider the following relational specification NAT:

spec NAT $\equiv$
  types nat
  rels z : nat
    s, p : nat $\to$ nat
  laws (N_1) z is a point
    (N_2) s is injective
    (N_3) s is a mapping
    (N_4) $sz = O$
    (N_5) $p = s^{\mathrm{T}}$
    (N_6) $L = \inf\{x : z \cup px = x\}$

Clearly, the natural numbers with zero, the successor function, and the (partial) predecessor function—considered as a point from $\mathbf{P}(\mathbb{N})$ resp. as relations from $\mathbb{B}^{\mathbb{N}\times\mathbb{N}}$—are a model of the specification NAT. The axioms (N_2), (N_4) and (N_6) of NAT can be regarded as a relational variant of the well-known Peano axioms. In particular, (N_6) corresponds to the induction axiom since it says that every natural number is reducible to zero by finitely many applications of the predecessor function: In a relational structure S for the signature of NAT axiom (N_6) holds if and only if L is the least fixed point of

$$\tau_N : \mathbb{B}^{\mathrm{nat}_S \times \mathrm{nat}_S} \to \mathbb{B}^{\mathrm{nat}_S \times \mathrm{nat}_S}, \qquad \tau_N(X) := z_S \cup p_S X\,,$$

and the fixed point theorem for continuous functions on complete lattices shows that this fixed point coincides with $p_S^* z_S$.

The above specification of natural numbers also shows that partially defined operations present no problems when using relations.

Now, we prove that the specification NAT is monomorphic. The proof of the following proposition shows a typical pattern for a proof of monomorphy in the relational case.

4.1.1. PROPOSITION. *The specification* NAT *is monomorphic.*

Proof. Let S and S' be two models of NAT. Furthermore, define the relation Φ to be the least fixed point μ_σ of the continuous functional $\sigma(X) := z_S z_{S'}^{\mathrm{T}} \cup p_S X s_{S'}$.

We show first that Φ is a mapping. For the proof of $\Phi^{\mathrm{T}}\Phi \subset I$ we apply computational induction with $X^{\mathrm{T}}X \subset I$ as an admissible predicate $P(X)$. Clearly, $P(O)$ holds. For the induction step we use (N_2), (N_4), (N_5), and $z_S^{\mathrm{T}} z_S = L$ and obtain

$$\sigma(X)^{\mathrm{T}}\sigma(X) = z_{S'} z_{S'}^{\mathrm{T}} \cup p_{S'} X^{\mathrm{T}} s_S p_S X s_{S'} \subset I \cup p_{S'} X^{\mathrm{T}} X s_{S'} \subset I\,.$$

For the proof of totality, $\Phi L = L$, we use the fact that the point $z_{S'}$ is surjective and that the relation $s_{S'}$ is total (which follows from (N_3)) and get

$$\Phi L = \sigma(\Phi)L = z_S z_{S'}^{\mathrm{T}} L \cup p_S \Phi s_{S'} L = z_S L \cup p_S \Phi L = \tau_N(\Phi L)\,,$$

where τ_N is as above. Hence, (N_6) (resp. $L = \mu_{\tau_N}$) shows the desired result.

Analogously it can be shown that Φ^{T} is a mapping.

Finally, the equations $z_S = \Phi z_{S'}$, $s_S \Phi = \Phi s_{S'}$, and $p_S \Phi = \Phi p_{S'}$ remain to be shown. From (N_4), $z_{S'}^{\mathrm{T}} z_{S'} = L$, and $\Phi = \sigma(\Phi)$ we obtain

$$z_S = z_S z_{S'}^{\mathrm{T}} z_{S'} = (z_S z_{S'}^{\mathrm{T}} \cup p_S \Phi s_{S'}) z_{S'} = \Phi z_{S'}\,,$$

i.e., the first equation. For the proof of the second equation we use (N_4), $s_S p_S = I$ (which follows from totality and injectivity of s_S in conjunction with (N_5)), and $\Phi = \sigma(\Phi)$:

$$s_S \Phi = s_S (z_S z_{S'}^{\mathrm{T}} \cup p_S \Phi s_{S'}) = s_S p_S \Phi s_{S'} = \Phi s_{S'}\,.$$

The third equation is proved in the same way. ■

At this place it should be pointed out that only the proofs of totality of Φ and Φ^{T} require axiom (N_6). With the usual function notation for univalent relations this fact is illustrated by viewing Φ as least solution of the recursion

$$f(u) = \begin{cases} z_{S'} & \text{if } u = z_S, \\ s_{S'}(f(p_S(u))) & \text{if } u \neq z_S, \end{cases}$$

and Φ^{T} as least solution of the recursion

$$g(v) = \begin{cases} z_S & \text{if } v = z_{S'}, \\ s_S(g(p_{S'}(v))) & \text{if } v \neq z_{S'}. \end{cases}$$

Now, totality of Φ (resp. Φ^{T}) means that the first (second) recursion has to terminate and termination requires that the domain nat_S (resp. $\mathrm{nat}_{S'}$) has to be finitely generated.

Next, we show by means of an example that in the relational case it is possible to prove totality by computational induction if the generation principle is described by a functional as in the case of (N_6). In the usual algebraic approach such totality proofs are impossible or rather complicated. (Cf. also the remarks on p. 397 of [Manna 74].) We deal with a specification of stacks. Translating the usual algebraic description into the relational framework, we are able to drop the append operation, because the "tupling" of the projection symbols w.r.t. the operations top and rest produces the same result as append (see also [Desharnais 89]).

spec STACK $\equiv$
param types m
target types stack
rels e : stack
t : stack $\rightarrow m$
r : stack $\rightarrow$ stack
laws (S_1) e is a point
(S_2) $t^{\mathrm{T}}e = O$
(S_3) $r^{\mathrm{T}}e = O$
(S_4) $t^{\mathrm{T}}t = I$
(S_5) $r^{\mathrm{T}}r = I$
(S_6) $r^{\mathrm{T}}t = L$
(S_7) $tt^{\mathrm{T}} \cap rr^{\mathrm{T}} \subset I$
(S_8) $L = \inf\{x : e \cup rx = x\}$

For an arbitrary interpretation of m the stacks over this interpretation with the empty stack (considered as a point), the top operation, and the rest operation (both considered as relations) form a model of STACK. E.g., (S_2) and (S_3) postulate that an application of top resp. rest to the empty stack is undefined. Axiom (S_8) corresponds to the generation principle. In a relational structure S for the signature of STACK it holds if and only if the least fixed point of

$$\tau_S : \mathbb{B}^{\mathrm{stack}_S \times \mathrm{stack}_S} \rightarrow \mathbb{B}^{\mathrm{stack}_S \times \mathrm{stack}_S}, \qquad \tau_S(X) := e_S \cup r_S^X ,$$

coincides with the universal relation or, equivalently, if $L = r_S^* e_S$. In this form,

(S_8) postulates that every stack is reducible to the empty stack by n-fold application of the rest operation.

4.1.2. Proposition. *The parameterized specification* STACK *is monomorphic.*

Sketch of proof. Let S and S' be two models of STACK. Furthermore, assume a bijective mapping $\Phi_m \in \mathbb{B}^{m_S \times m_{S'}}$. As in the case of NAT it can be verified that the least fixed point Φ of the continuous functional $\sigma(X) := e_S e_{S'}^{\mathrm{T}} \cup (t_S \Phi_m t_{S'}^{\mathrm{T}} \cap r_S X r_{S'}^{\mathrm{T}})$ is also a bijective mapping, and fulfills $e_S = \Phi e_{S'}$, $t_S \Phi_m = \Phi t_{S'}$, and $r_S \Phi = \Phi r_{S'}$. Thus, the pair $\Omega = (\Omega_{\text{stack}}, \Omega_m)$, where $\Omega_{\text{stack}} := \Phi$ and $\Omega_m := \Phi_m$, is an isomorphism between S and S'. ∎

The next specification, firstly, extends STACK by NAT and, secondly, extends the result by adding an operation for computing the length of a stack:

$$\begin{aligned}
&\textbf{spec } \text{LSTACK} \equiv \text{STACK} \oplus \text{NAT} \oplus \\
&\quad \textbf{rels } \; l : \text{stack} \to \text{nat} \\
&\quad \textbf{laws } (\mathrm{L}_1) \; l = \inf\{x : ez^{\mathrm{T}} \cup rxs = x\}
\end{aligned}$$

The proof of the following proposition demonstrates that by relation algebraic techniques totality of operations may elegantly be shown by using computational induction.

4.1.3. Proposition. *Let S be a model of* LSTACK. *Then l_S is a mapping.*

Proof. l_S is the least fixed point μ_{λ_l} of the continuous functional $\lambda_l(X) := e_S z_S \mathrm{T} \cup r_S X s_S$.

Firstly, we show that the least fixed point of λ_l is univalent. We apply computational induction with $X^{\mathrm{T}} X \subset I$ as admissible predicate $P(X)$. The induction base $P(O)$ is obvious. For the induction step we use (N_3), (S_3), (S_5), and $e_S^{\mathrm{T}} e_S = L$ and obtain

$$\lambda_l(X)^{\mathrm{T}} \lambda_l(X) = z_S z_S^{\mathrm{T}} \cup s_S^{\mathrm{T}} X^{\mathrm{T}} r_S^{\mathrm{T}} r_S X s_S = z_S z_S^{\mathrm{T}} \cup s_S^{\mathrm{T}} X^{\mathrm{T}} X s_S \subset I \cup s_S^{\mathrm{T}} s_S = I\,.$$

Now, we prove that the least fixed point of λ_l is total. To this end, we define the admissible predicate $P(X, Y, Z)$ as the inclusion $YL \subset XZ$ and show that $P(\mu_{\lambda_l}, \mu_{\tau_S}, \mu_{\tau_N})$ holds. Since NAT and STACK are finitely generated, this implies $L = LL \subset \mu_{\lambda_l} L$, i.e., the desired result. Again the induction base is obvious. For the induction step we use $z_S^{\mathrm{T}} z_S = L$, $s_S p_S = I$, and formula (N_4) and calculate

$$\begin{aligned}
\lambda_l(X)\tau_N(Z) &= e_S z_S^{\mathrm{T}} z_S \cup e_S z_S^{\mathrm{T}} p_S Z \cup r_S X s_S z_S \cup r_S X s_S p_S Z \\
&= e_S L \cup e_S z_S^{\mathrm{T}} p_S Z \cup r_S X Z\,.
\end{aligned}$$

On the other hand (we use axioms (N_4) and (N_5)),

$$\tau_S(Y)L = (e_S \cup r_S Y)L = e_S L \cup r_S Y L = e_S L \cup e_S z_S^{\mathrm{T}} p_S Z \cup r_S Y L\,.$$

Hence, the induction hypothesis applies. ∎

4.2. *Transition into algorithmic form.* Let (Σ, A) be a relational specification and r a relation symbol from the signature Σ. We call a relational Σ-formula an *explicit equation* for r if it is of the form $r = t$, where t is a closed relational Σ-term, or a second-order formula $r = \inf\{x : t = x\}$. A relational specification the non-parameter part of which includes only explicit definitions as axioms is called *explicit equational*. If the parameter part has a model, then such a specification also has one, provided the following two conditions hold:

a) For each relation symbol of the non-parameter part there exists precisely one explicit equation.

b) The explicit equations of the non-parameter part can be arranged in such a way that in the right-hand side of equation n at most the left-hand sides of equations i, $i < n$, occur. ("The equations can be sequentialized.")

In this case, a model S of the parameter part can be extended to a model S' of the entire specification, if for a relation symbol r of the non-parameter part, $r_{S'}$ is defined as

- $r_{S'} = \mathbf{v}_{S'}(t)$ if the explicit equation for r is $r = t$,
- $r_{S'} = \mu_\tau$ if the explicit equation for r is $r = \inf\{x : t = x\}$ and τ is the monotonic functional corresponding to t.

Obviously, the first case provides an algorithmic solution for the symbol r. The same holds also for the second case, if the functional τ is moreover continuous (e.g., since the variable x occurs in no subterm of the form $\overline{t'}$ or if the base sets of the model are finite). Then the least fixed point μ_τ of τ can be iteratively computed according to the fixed point theorem for continuous functions.

For many graph-theoretic problems a useful way to obtain an algorithmic solution is to turn them into a relational formulation and then apply the well-developed apparatus of abstract relational algebra. Our next aim is to demonstrate that this exactly fits into our framework. We present two examples of relational specifications each of which describes a problem on directed graphs without certain paths of infinite lengths. Then we transform the specifications into explicit equational ones which provide algorithmic solutions.

Firstly, we concentrate on regressively finite graphs, that is, directed graphs in which all paths ending in a point have finite lengths. We consider the problem of computing for such a graph a minimal (w.r.t. inclusion) set of points from which every point can be reached via a path, a so-called *point base*. Turning the point base problem on regressively finite graphs into our framework, we get the following relational specification:

spec BASE $\equiv$
  param types m
    rels $r : m \to m$
    laws (B_1) $\forall x\,(x \subset r^{\mathrm{T}} x \to x = O)$

target rels $b : m$
laws (B$_2$) $L = \inf\{x : b \cup r^{\mathrm{T}} x = x\}$
(B$_3$) $\forall c\,(L = \inf\{x : c \cup r^{\mathrm{T}} x = x\} \wedge c \subset b \rightarrow c = b)$

Let S be a model of this specification. Then axiom (B$_1$) is the relational version of the law that the interpretation r_S is regressively finite. In the case of m_S being finite, the axiom holds if and only if r_S is cycle-free. Furthermore, the two axioms (B$_2$) and (B$_3$) define the interpretation b_S to be minimal in the set $\{v \in \mathbf{V}(m_S) : L = r_S^{\mathrm{T}*} v\}$. In terms of concrete relations, the property $L = r_S^{\mathrm{T}*} v$ describes that each element of m_S can be reached from the set v.

Since the symbol r is a parameter of the relational specification BASE, it suffices to derive an "algorithmic" explicit equation for the constant symbol b. As we will see in the following, this can be done without auxiliary symbols. The key idea of the algorithm is given by

4.2.1. Lemma. *For all R and v, from $L = R^{\mathrm{T}*} v$ it follows that $\overline{R^{\mathrm{T}} L} \subset v$.*

Proof. Firstly, we have $\overline{R^{\mathrm{T}} L} \subset L = R^{\mathrm{T}*} v = (I \cup R^{\mathrm{T}+}) v = v \cup R^{\mathrm{T}+} v$, which in turn implies $\overline{R^{\mathrm{T}} L} \cap \overline{R^{\mathrm{T}+} v} \subset v$. In conjunction with the equation $R^{\mathrm{T}+} v = R^{\mathrm{T}} R^{\mathrm{T}*} v = R^{\mathrm{T}} L$ this yields the result. ■

Now, we use axiom (B$_2$) and deduce from this lemma that in each model S of BASE the inclusion $\overline{r_S^{\mathrm{T}} L} \subset b_S$ is valid. This fact suggests considering the relational specification

spec BASE$'$ $\equiv$
param $\gg$ identical to BASE $\ll$
target rels $b : m$
laws (B$_4$) $b = \overline{r^{\mathrm{T}} L}$

containing an algorithmic solution for the symbol b. In the concrete case, the vector $\overline{r^{\mathrm{T}} L}$ describes the set of initial points (or sources). In terms of graphs, therefore, the following proposition shows that the only point base of a regressively finite graph is the set of all points without predecessor.

4.2.2. Proposition. *Each model of* BASE$'$ *is also a model of* BASE.

Proof. Let S be a model of BASE$'$. From axiom (B$_4$) and Lemma 4.2.1 we find that b_S is a lower bound of the set $\{v \in \mathbf{V}(m_S) : L = r_S^{\mathrm{T}*} v\}$. Since least elements are also minimal, it remains to show that b_S is an element of this set. To do this, we use (B$_4$) and calculate

$$\begin{aligned}
r_S^{\mathrm{T}*} b_S \cup r_S^{\mathrm{T}} \overline{r_S^{\mathrm{T}*} b_S} &= (I \cup r_S^{\mathrm{T}} r_S^{\mathrm{T}*}) b_S \cup r_S^{\mathrm{T}} \overline{r_S^{\mathrm{T}*} b_S} \\
&= b_S \cup r_S^{\mathrm{T}} r_S^{\mathrm{T}*} b_S \cup r_S^{\mathrm{T}} \overline{r_S^{\mathrm{T}*} b_S} = b_S \cup r_S^{\mathrm{T}} (r_S^{\mathrm{T}*} b_S \cup \overline{r_S^{\mathrm{T}*} b_S}) \\
&= b_S \cup \overline{b_S} = L\,,
\end{aligned}$$

which in turn implies $\overline{r_S^{\mathrm{T}*}b_S} \subset r_S^{\mathrm{T}}\overline{r_S^{\mathrm{T}*}b_S}$. Now, ($B_1$) shows $\overline{r_S^{\mathrm{T}*}b_S} = O$, i.e., $L = r_S^{\mathrm{T}*}b_S$. ∎

Now, we concentrate on progressively finite graphs, that is, directed graphs in which all paths starting from a point have finite lengths. We consider the problem of computing a *kernel* of such a graph, that is, a set c of points such that from every point outside of c there is an arc leading into c and no two points of c are connected via an arc. When specifying this problem relationally we get

```
spec KERNEL ≡
  param types  $m$
        rels   $r : m \to m$
        laws   ($K_1$) $\forall x\,(x \subset rx \to x = O)$
  target rels  $c : m$
        laws   ($K_2$) $c = \overline{rc}$
```

Let S be a model of this specification. For $\kappa(X) := \overline{r_S X}$, axiom ($K_2$) holds if and only if c_S is a fixed point of the functional κ. This functional is antitonic, so the fixed point theorem for monotonic functions cannot be applied. The situation can be mended by considering $\kappa^2 := \kappa \circ \kappa$ instead of κ, which is monotonic. Clearly, every fixed point of κ is a fixed point of κ^2 as well. But, conversely, not every fixed point of κ^2 is one of κ. Here one needs additional properties. E.g., it suffices to demand that κ^2 has precisely one fixed point. This will be shown now.

4.2.3. PROPOSITION (Fixed point theorem for antitone mappings). *Let $(V, \leq)$ be a complete lattice and $f : V \to V$ be antitonic. Furthermore, let m and M be the least fixed point resp. greatest fixed point of f^2. Then $f(m) = M$ and $f(M) = m$. Thus, if f^2 has precisely one fixed point, this is also the only fixed point of f.*

Proof. From $m = f^2(m)$ we obtain $f(m) = f^2(f(m))$. Hence, $f(m)$ is a fixed point of f^2, which implies $f(m) \leq M$. In the same way $m \leq f(M)$ is obtained from $M = f^2(M)$, since this implies that $f(M)$ is also a fixed point of f^2.

Using these two estimates, we obtain $m \leq f(M) \leq f^2(m) = m$ and $M = f^2(M) \leq f(m) \leq M$ from the fact that f is an antitone mapping. ∎

Now, we consider the relational specification

```
spec KERNEL′ ≡
  param ≫identical to KERNEL≪
  target rels  $c : m$
         laws  ($K_3$) $c = \inf\{x : \overline{r\overline{rx}} = x\}$
```

and show that it also specifies the problem of computing a kernel.

4.2.4. PROPOSITION. *Each model of* KERNEL′ *is also a model of* KERNEL.

P r o o f. Let S be a model of KERNEL$'$. Since (K_2) holds if and only if c_S is a fixed point of κ and (K_3) holds if and only if c_S is the least fixed point of κ^2, it suffices to show that the latter functional has precisely one fixed point. Then Proposition 4.2.3 shows the desired result.

Let M denote the greatest fixed point of κ^2. We use the Schröder equivalences and get $r_S^{\mathrm{T}} M \subset r_S M$ from $M \subset \kappa^2(M)$. Now, the Dedekind rule yields

$$r_S M \wedge M \subset (r_S \wedge M M^{\mathrm{T}})(M \wedge r_S^{\mathrm{T}} M) \subset r_S(M \wedge r_S^{\mathrm{T}} M) \subset r_S(M \wedge r_S M)$$

and in combination with axiom (K_1) we obtain $r_S M \wedge M = O$, i.e., $M \subset \kappa(M)$. Due to Proposition 4.2.3 the relation $\kappa(M)$ is the least fixed point of κ^2. Hence, this functional has precisely one fixed point. ∎

Thus, the only kernel of a progressively finite graph can be computed as follows: One starts with c_0 as the empty set or the set of all terminal points (or sinks) and collects iteratively in c_{i+1} the points having all their successors in the set of predecessors of c_i. If the graph is finite (in this case it is progressively finite if and only if it is cycle-free), then the exhaustion process is also finite.

4.3. *Construction of implementations from type equations*. Following the algebraic specification approach (see, e.g., [Wirsing 90]), we call a relational specification (Σ', A') an *implementation* of the relational specification (Σ, A) if Σ is a sub-signature of Σ' and each formula of A holds in each model of (Σ', A'). In the case of parameterized specifications we demand additionally the two parameter parts to be identical. Transition into algorithmic form, therefore, is only a very special case of proceeding from a relational specification to an implementation.

In [Berghammer *et al.* 89] and [Zierer 91] it is shown that relation algebra is an appropriate technical means for describing all domain constructions used in denotational semantics. All these descriptions can immediately be translated into (parameterized) relational specifications and those can be used as a basis for constructing implementations from data type equations. Generally, such a construction proceeds in three steps, where for the first two steps a graphical representation of the equation (with the types as vertices and the canonical functions like projections and injections as arcs) is very profitable. Firstly, the (recursive) data type equation is translated into a sequence of instantiations of the just mentioned kind of specifications. In doing so, in combination with an appropriate renaming, all the types (also the auxiliary ones) and the first part of the operations are implemented. Next, the remaining operations are defined in terms of the already present ones. Here transposition of the injection symbols plays a prominent role. And, finally, the generation principle is formulated if it holds also for the original specification.

The use of this procedure is now illustrated by means of the example NAT. For natural numbers we have the equation $\mathbb{N} = \mathbf{1} + \mathbb{N}$, where "=" means "is isomorphic to", $\mathbf{1}$ is a single-element set, and the operator "+" produces the direct sum (disjoint union) of two sets. Since only in a single-element set the universal and identity relations coincide, we get for such sets the following monomorphic relational specification:

spec UNIT $\equiv$
  types unit
  laws $(\mathrm{U}_1)\ I = L$

For a relational specification of the direct sum it is natural to use symbols for the two injections. Then one obtains

spec DSUM $\equiv$
  param types m, n
  target types sum
  rels $\iota : m \rightarrow \mathrm{sum}$
  $\kappa : n \rightarrow \mathrm{sum}$
  laws $(\mathrm{D}_1)\ \iota\iota^{\mathrm{T}} = I$
  $(\mathrm{D}_2)\ \kappa\kappa^{\mathrm{T}} = I$
  $(\mathrm{D}_3)\ \iota^{\mathrm{T}}\iota \cup \kappa^{\mathrm{T}}\kappa = I$
  $(\mathrm{D}_4)\ \iota\kappa^{\mathrm{T}} = O$

By equations (D_1) through (D_4) the direct sum is uniquely determined up to isomorphism. If S and S' are two models of DSUM and $\Phi_m \in \mathbb{B}^{m_S \times m_{S'}}$ and $\Phi_n \in \mathbb{B}^{n_S \times n_{S'}}$ are two bijective mappings, the isomorphism from S to S' is given by $\Phi := \iota^{\mathrm{T}}\Phi_m\iota \cup \kappa^{\mathrm{T}}\Phi_n\kappa$. Furthermore, the interpretations of the injection symbols ι and κ are injective mappings.

Since the successor operation s equals the injection κ from the second variant nat, the first step of the above procedure yields the instantiation

include DSUM[unit, nat] **as** $[\mathrm{nat}, \iota, s]$

In the second step we define z and p in terms of ι and s by $z = \iota^{\mathrm{T}}L$ and $p = s^{\mathrm{T}}$. And, finally, we have to formulate the generation principle since it holds also for the original specification NAT. Altogether, we have

spec NAT$'$ $\equiv$
  include DSUM[unit, nat] **as** $[\mathrm{nat}, \iota, s]$
  rels $z : \mathrm{nat}$
  $p : \mathrm{nat} \rightarrow \mathrm{nat}$
  laws $(\mathrm{N}_7)\ z = \iota^{\mathrm{T}}L$
  $(\mathrm{N}_5)\ p = s^{\mathrm{T}}$
  $(\mathrm{N}_6)\ L = \inf\{x : z \cup px = x\}$

4.3.1. Proposition. *The specification* NAT$'$ *is an implementation of* NAT.

P r o o f. Let S be a model of NAT′. We only have to show (N_1) and (N_4), i.e., that z_S is a point and $s_S z_S = O$ is valid.

To prove injectivity of the vector z_S, we use the laws (N_7) and (U_1) and the fact that ι_S is univalent and get $z_S z_S^{\mathrm{T}} = \iota_S^{\mathrm{T}} L (\iota_S^{\mathrm{T}} L)^{\mathrm{T}} = \iota_S^{\mathrm{T}} L \iota_S = \iota_S^{\mathrm{T}} I \iota_S \subset I$. Surjectivity of z_S follows from (N_7) and Tarski's rule: $z_S^{\mathrm{T}} L = (\iota_S^{\mathrm{T}} L)^{\mathrm{T}} L = L \iota_S L = L$. And, finally, $s_S z_S = s_S \iota_S^{\mathrm{T}} L = O$ is a consequence of axiom (D_4). ∎

4.4. *Finiteness of base sets.* For termination considerations it is often necessary to specify the interpretation of a sort of a specification to be a finite set. In the following we shall discuss a method for solving this problem in our relational approach.

We use the fact that a set X is finite if and only if the irreflexive part $\subset$ of the subset ordering $\subseteq$ on the powerset 2^X is a progressively finite relation. Hence, we need a relational specification of the structure $(X, 2^X, \subseteq)$. As shown in [Berghammer *et al.* 89], this can conveniently be done using the is-element-of-relation $\in$ and the so-called symmetric quotient of two relations $R \in \mathbb{B}^{X \times Y}$ and $S \in \mathbb{B}^{X \times Z}$, which is a relation from $\mathbb{B}^{Y \times Z}$ and associates $y \in Y$ with $z \in Z$ if and only if the predecessor sets of y w.r.t. R and of z w.r.t. S coincide. Translating the characterization of [Berghammer *et al.* 89] into our notation, we get the following parameterized relational specification:

spec POWER ≡
  param types m
  target types set
    rels $E : m \to \mathrm{set}$
      $\Omega : \mathrm{set} \to \mathrm{set}$
    laws (P_1) $\mathrm{syq}(E, E) \subset I$
      (P_2) $\forall x \, (L = \mathrm{syq}(E, x)^{\mathrm{T}} L)$
      (P_3) $\Omega = \overline{E^{\mathrm{T}} \overline{E}}$

Here we use $\mathrm{syq}(t_1, t_2)$ as a shorthand for the relational term $\overline{t_1^{\mathrm{T}} \overline{t_2}} \cap \overline{\overline{t_1}^{\mathrm{T}} t_2}$. The value of the term $\overline{t_1^{\mathrm{T}} \overline{t_2}} \cap \overline{\overline{t_1}^{\mathrm{T}} t_2}$ is exactly the symmetric quotient of the values of t_1 and t_2 in the sense of [Berghammer *et al.* 89].

Now, assume the concrete case of E_S being the is-element-of-relation $\in$ between a set $X := m_S$ and its powerset $2^X := \mathrm{set}_S$. Then axiom (P_1) states that two elements M, N of 2^X are equal provided that every element of X is contained in M if and only if it is contained in N. Condition (P_2) corresponds to set comprehension saying that every column of a relation $R \in \mathbb{B}^{X \times X}$ (which is a vector $v \in \mathbf{V}(X)$ describing a subset of X) is represented by the symmetric quotient of $\in$ and v (which is a point in $\mathbf{P}(2^X)$ describing an element of the powerset 2^X). And, finally, (P_3) is a component-free definition of the subset ordering using the is-element-of-relation.

By using the fact mentioned at the beginning of this section in combination with the parameterized relational specification POWER, it is now very easy to specify the interpretation of a sort m of a relational specification to be finite. Firstly, we use the instantiation

$$\textbf{include } \mathrm{POWER}[m] \textbf{ as } [\mathrm{set}, E, \Omega]$$

and then we specify the irreflexive part of the subset ordering Ω to be progressively finite using the axiom

$$\forall x\,(x \subset (\Omega \cap \bar{I})x \to x = O)\,.$$

5. Nondeterminism. In this section we show that relational specifications can easily deal with angelic and demonic nondeterminism within a single context. By means of two examples we demonstrate some similarities and some differences between these two kinds of nondeterminism.

5.1. *Angelic nondeterminism.* Since relations may be one-to-many, they immediately provide a means for handling nondeterminism. Join plays the role of nondeterministic choice. We obtain the specific kind of angelic nondeterminism in that a possibility of a defined result is equivalent to the guarantee of a defined result. This is also mirrored by the neutrality of the null relation O w.r.t. join, since this relation stands for the least element $\perp$ as known from denotational semantics (cf. [Schmidt 86]).

5.2. *Demonic nondeterminism.* In sharp contrast to angelic nondeterminism is demonic nondeterminism. Here a possibility of an undefined result is equivalent to the guarantee of an undefined result.

For an integration of demonic nondeterminism we need two additional relational constructs. Implicitly, they have already been used in [Berghammer–Zierer 86] to describe relationally the semantics of nondeterministic functional programs; in a component-wise notation they are employed in [Nguyen 91]. The first construct $R \bullet S$ is called *demonic composition*; the second one $R \,\Box\, S$ is called *demonic join.* They are defined by

$$R \bullet S := RS \cap \overline{R\overline{SL}}\,, \qquad R \,\Box\, S := (R \cup S) \cap RL \cap SL\,.$$

Both constructs reflect the essence of demonic nondeterminism, viz. that an evaluation is guaranteed to yield a defined result if and only if any of the choices yields a defined result. The operation $\Box$ corresponds exactly to Dijkstra's demonic nondeterministic branching operator (see [Dijkstra 75]).

The demonic variants of composition and join obey some nice algebraic laws (see also [Nguyen 91]). For example: Demonic composition and join are associative operations, the latter is commutative, and the former is distributive w.r.t. the second one. As we have defined both operations in terms of abstract relation algebra, the proofs are also given in this fashion. We will give an example:

5.2.1. PROPOSITION. *Demonic composition is an associative operation.*

P r o o f. Firstly, we use some fundamental properties of vectors (see, e.g., [Schmidt–Ströhlein 89]) and get

$$\begin{aligned}(Q \bullet R) \bullet S &= (QR \cap \overline{Q\overline{RL}}) \bullet S \\ &= (QR \cap \overline{Q\overline{RL}})S \cap \overline{(QR \cap \overline{Q\overline{RL}})\overline{SL}} \\ &= QRS \cap \overline{Q\overline{RL}} \cap \overline{QR\overline{SL} \cap \overline{Q\overline{RL}}} \\ &= QRS \cap \overline{Q\overline{RL}} \cap (\overline{QR\overline{SL}} \cup Q\overline{RL}) \\ &= QRS \cap \overline{Q\overline{RL}} \cap \overline{QR\overline{SL}}\,,\end{aligned}$$

$$\begin{aligned}Q \bullet (R \bullet S) &= Q(R \bullet S) \cap \overline{Q\overline{(R \bullet S)L}} \\ &= Q(RS \cap \overline{R\overline{SL}}) \cap \overline{Q\overline{(RS \cap \overline{R\overline{SL}})L}} \\ &= Q(RS \cap \overline{R\overline{SL}}) \cap \overline{Q\overline{(RSL \cap \overline{R\overline{SL}})}} \\ &= Q(RS \cap \overline{R\overline{SL}}) \cap \overline{Q(\overline{RSL} \cup R\overline{SL})} \\ &= Q(RS \cap \overline{R\overline{SL}}) \cap \overline{Q\overline{RSL}} \cap \overline{QR\overline{SL}}\,.\end{aligned}$$

From these two equations it immediately follows that $Q \bullet (R \bullet S) \subset (Q \bullet R) \bullet S$. For the proof of $(Q \bullet R) \bullet S \subset Q \bullet (R \bullet S)$ we show that $QRS \cap \overline{Q\overline{RL}} \cap \overline{QR\overline{SL}}$ is contained in the three relations $Q(RS \cap \overline{R\overline{SL}})$, $\overline{Q\overline{RSL}}$, and $\overline{QR\overline{SL}}$.

In order to prove the first inclusion, we start with $QR\overline{SL} \subset Q(\overline{RL} \cup R\overline{SL}) = \overline{\overline{Q\overline{RL}} \cap \overline{QR\overline{SL}}}$ and obtain with the help of the Schröder equivalences $Q^{\mathrm{T}}(\overline{Q\overline{RL}} \cap \overline{QR\overline{SL}}) \subset \overline{R\overline{SL}}$. Now, we use Dedekind's rule and get

$$\begin{aligned}QRS \cap \overline{Q\overline{RL}} \cap \overline{QR\overline{SL}} &\subset (Q \cap \ldots)(RS \cap Q^{\mathrm{T}}(\overline{Q\overline{RL}} \cap \overline{QR\overline{SL}})) \\ &\subset Q(RS \cap \overline{R\overline{SL}})\,.\end{aligned}$$

In the second case we use $L = R(SL \cup \overline{SL}) \cup \overline{RL} = RSL \cup R\overline{SL} \cup \overline{RL}$. This equation implies $\overline{RSL} \subset R\overline{SL} \cup \overline{RL}$. As a consequence, we have

$$QRS \cap Q\overline{RSL} \subset Q(R\overline{SL} \cup \overline{RL}) = QR\overline{SL} \cup Q\overline{RL}$$

from which the desired inclusion follows. The proof of the third inclusion is obvious. ■

In the following proposition we list some sufficient conditions for the usual operations and their demonic variants to coincide. Proofs are trivial and, thus, omitted.

5.2.2. PROPOSITION. (i) *$R \bullet S = RS$ provided R is univalent, S is total, or $R \bullet S$ is total.*

(ii) *$R \,\square\, S = R \cup S$ provided both R and S are total or $R \,\square\, S$ is total.* ■

The proof of the following proposition is also trivial. However, the stated result is essential, since it often allows the symbol r of a second-order formula $r = \inf\{x : t = x\}$ to be interpreted as the least fixed point of the monotonic functional corresponding to t.

5.2.3. PROPOSITION. *Demonic composition of relations is monotonic w.r.t. the second argument and demonic join of relations is monotonic w.r.t. both arguments.* ■

With the help of demonic composition and join it is very easy to introduce demonic nondeterminism, too. Firstly, it seems to be natural to replace in the inductive definition of the value $\mathsf{v}_{S,v}(t)$ of the relational term t the usual composition resp. join of relations by their demonic variants. This leads to a framework which contains demonic nondeterminism only. However, this approach has a serious drawback: If one uses only demonic composition and join, then sometimes specifications may become quite unnatural and complicated. Furthermore, it seems that for an explicit formulation of the generation principle the usual composition and join are absolutely necessary.

Therefore, we prefer to introduce besides usual composition and join also their demonic variants as "term-forming" operator symbols. I.e., we also admit relational terms of the form $t_1 \bullet t_2$ resp. $t_1 \Box t_2$. The definition of the value of a term is extended as follows:

$$\mathsf{v}_{S,v}(t_1 \bullet t_2) = \mathsf{v}_{S,v}(t_1) \bullet \mathsf{v}_{S,v}(t_2)\,, \qquad \mathsf{v}_{S,v}(t_1 \Box t_2) = \mathsf{v}_{S,v}(t_1) \Box \mathsf{v}_{S,v}(t_2)\,,$$

Thereby, we obtain a mixed system which supports both forms of nondeterminism.

5.3. *Examples*. In the following two examples we want to demonstrate some similarities and some differences between angelic and demonic nondeterminism.

Firstly, we extend the specification NAT of Section 4.1 by an operation r which assigns to a given natural number n an arbitrary natural number less then or equal to n. Using the "classical" algebraic approach with functions instead of relations (see, e.g., [Ehrig–Mahr 85], [Wirsing 90]), the nondeterminism of r is expressed by the fact that the extended specification is non-monomorphic. In each model the operation yields an arbitrary element but every call yields the same element. In the relational approach, it is possible to specify the interpretation of r as a total but non-univalent relation monomorphically as follows:

spec RNAT ≡ NAT ⊕
  types None
  rels $r : \text{nat} \to \text{nat}$
  laws $(\mathrm{R}_1)\ r = \inf\{x : I \cup px = x\}$

Assume the relational structure S to be a model of RNAT. Then (R_1) holds in S if and only if r_S is the least fixed point μ_{λ_r} of the functional

$$\lambda_r : \mathbb{B}^{\text{nat}_S \times \text{nat}_S} \to \mathbb{B}^{\text{nat}_S \times \text{nat}_S}\,, \qquad \lambda_r(X) := I \cup p_S X\,,$$

and in conjunction with the fixed point theorem for continuous functions on complete lattices we obtain μ_{λ_r} as the reflexive-transitive closure p_S^* of p_S. I.e., the interpretation of the relation symbol r behaves exactly as expected.

In the above relational specification RNAT we are in the case of angelic non-determinism. Therefore, in the recursion specifying the meaning of r there is no termination case needed. This becomes apparent if one represents relations as set-valued functions. Then formula (R$_1$) of RNAT expresses the fact that the interpretation of r is the least solution of the recursion $f(x) = \{x\} \cup f(x-1)$.

If in axiom (R$_1$) of RNAT composition and join are replaced by their demonic variants, then it can easily be shown that $r_S = O$ for each model S of the resulting specification. To specify the meaning of r also in the demonic case "correctly", an additional termination case is needed saying that for the argument 0 the operation returns 0 as the only result. This leads to the relational specification

$$\begin{aligned}
&\textbf{spec}\ \mathrm{RNAT}' \equiv \mathrm{NAT} \oplus \\
&\quad \textbf{rels}\ \ r : \mathrm{nat} \to \mathrm{nat} \\
&\quad \textbf{laws}\ (\mathrm{R}_2)\ r = \inf\{x : zz^{\mathrm{T}} \cup (I \,\square\, (p \bullet x)) = x\}
\end{aligned}$$

Now, assume S to be a model of this specification. Then Proposition 5.2.3 implies that the functional

$$\gamma_r : \mathbb{B}^{\mathrm{nat}_S \times \mathrm{nat}_S} \to \mathbb{B}^{\mathrm{nat}_S \times \mathrm{nat}_S}\,, \qquad \gamma_r(X) := z_S z_S^{\mathrm{T}} \cup (I \,\square\, (p_S \bullet X))\,,$$

is monotonic and axiom (R$_2$) specifies r_S as its least fixed point μ_{γ_r}. Since the relation p_S is univalent, Proposition 5.2.2 shows $p_S \bullet X = p_S X$ and in conjunction with $z_S z_S^{\mathrm{T}} = I \cap z_S$ we obtain the equation

$$\gamma_r(X) = (I \cap z_S) \cup (I \cap p_S X L) \cup (p_S X \cap p_S X L) = (I \cup p_S X) \cap (z_S \cup p_S X L)\,.$$

I.e., the functional is even continuous. The next theorem shows that the two (monomorphic) specifications RNAT and RNAT$'$ have the same class of models.

5.3.1. Proposition. *Let S be a model of the specification* RNAT$'$. *Then* $r_S = p_S^*$.

P r o o f. We define $P_n := \bigcup_{i=0}^{n-1} p_S^i$ and prove by induction $\gamma_r^n(O) = P_n \cap P_n z_S$ for all $n \geq 1$.

The induction base $n = 1$ is trivial. For the induction step we calculate

$$\begin{aligned}
\gamma^{n+1}(O) &= (I \cup p_S(P_n \cap P_n z_S)) \cap (z_S \cup p_S(P_n \cap P_n z_S)L) \\
&= (I \cup p_S P_n) \cap (I \cup p_S P_n z_S) \cap (z_S \cup p_S P_n L) \cap (z_S \cup p_S P_n z_S) \\
&= P_{n+1} \cap (I \cup p_S P_n z_S) \cap P_{n+1} z_S \\
&= P_{n+1} \cap P_{n+1} z_S
\end{aligned}$$

because $(P_n \cap P_n z_S)L = P_n L \cap P_n z_S$, and from $p_S P_n \cap z_S \subset p_S L \cap z_S \subset \overline{z_S} \cap z_S = O$

we get

$$\begin{aligned} P_{n+1} \cap P_{n+1} z_S &= (I \cup p_S P_n) \cap (z_S \cup p_S P_n z_S) \\ &= (I \cap z_S) \cup (I \cap p_S P_n z_S) \cup (p_S P_n \cap z_S) \cup (p_S P_n \cap p_S P_n z_S) \subset I \cup p_S P_n z_S \,. \end{aligned}$$

Now, the fixed point theorem for continuous functions yields $r_S = \mu_{\gamma_r} = p_S^* \cap p_S^* z_S$ and, finally, the result follows from $p_S^* z_S = L$, i.e., from the fact that NAT is finitely generated. ■

As a second example, we consider the following nondeterministic specification, where the set of values of the constant symbol c contains zero and for each of its elements also the successor (cf. also [Hussmann 89], Example 1.12).

$$\begin{array}{ll} \textbf{spec}\ \mathrm{CNAT} \equiv \mathrm{NAT} \oplus & \\ \quad \textbf{types} & \mathrm{None} \\ \quad \textbf{rels} & c : \mathrm{nat} \\ \quad \textbf{laws} & (\mathrm{C}_1)\ z \subset c \\ & (\mathrm{C}_2)\ pc \subset c \end{array}$$

Let S be a model of CNAT and let the predicate $P(X)$ be defined by $X \subset c_S$. Then P is admissible for computational induction. With the help of this principle it can easily be shown that P holds for the least fixed point of the functional τ_N of Section 4.1. The induction base is trivial; the induction step uses (C_1) and (C_2). Thus, axiom (N_6) of NAT implies $c_S = L$, i.e., the interpretation of c yields an arbitrary natural number.

Obviously, the class of models of CNAT remains unchanged if (C_1) and (C_2) are replaced by the single formula $z \cup pc \subset c$. However, the same is not true in the case of demonic nondeterminism. If in the formula (C_2) composition is replaced by its demonic variant, then we also have $c_S = L$ for each model S of the resulting specification. This follows from the univalence of p_S. But, for the relational specification

$$\begin{array}{ll} \textbf{spec}\ \mathrm{CNAT}' \equiv \mathrm{NAT} \oplus & \\ \quad \textbf{rels} & c : \mathrm{nat} \\ \quad \textbf{laws} & (\mathrm{C}_3)\ z \,\square\, (p \bullet c) \subset c \end{array}$$

the following property holds:

5.3.2. Proposition. *The specification* CNAT′ *possesses infinitely many non-isomorphic models.*

P r o o f. Let S be a model of the specification NAT. We define for every natural number n a model S_n of CNAT′ by $z_{S_n} := z_S$, $s_{S_n} := s_S$, $p_{S_n} := p_S$, and $c_{S_n} := p_S^n L$. Only axiom (C_3) needs a proof:

$$z_{S_n} \,\square\, (p_{S_n} \bullet p_S^n L) = (z_{S_n} \cup p_{S_n} p_S^n L) \cap z_{S_n} L \cap p_{S_n} p_S^n L = z_S \cap p_S^{n+1} L \subset p_S^n L \,.$$

If m and n are two different natural numbers, then the models S_m and S_n are non-isomorphic. This can be shown as follows: Suppose Φ is an isomorphism between the models S_m and S_n. If $m < n$, then $c_{S_m} = \Phi c_{S_n}$ equals $p_S^m L = \Phi p_S^n L$ and by using $s_S p_S = I$ this implies $L = s_S^m \Phi p_S^n L = s_S^m p_S^n \Phi L = p_S^{n-m} L \subset p_S L$. However, this is a contradiction since the relation p_S is non-total. The remaining case $n < m$ is proved in exactly the same way. ∎

The interpretation of c in S_n yields an arbitrary natural number greater than or equal to n and the model S_0 of the relational specification CNAT′ is—up to isomorphism—the only model of CNAT. Hence, CNAT is an implementation of CNAT′.

6. Concluding remarks. In the preceding sections we have proposed abstract relation algebra as a means for specification of data types and programs. We have also demonstrated the usefulness of the relational approach by providing many examples. By means of the examples, among other things we have exhibited a few advantages of second-order formulae as axioms and have shown that a relational specification of data types and programs allows a very natural treatment of angelic and demonic nondeterminism.

When compared to algebraic specifications, dealing with relations can sometimes lead to a heavy notation, in particular, if we have to describe operations with more than one argument. These difficulties, however, can be compensated by defining abbreviations for relational terms of specific form occurring frequently. Then one can often formulate the axioms of a relational specification in an FP-like form. If we also introduce the semantic counterparts of the abbreviations as operations on relations, these obey many nice algebraic laws which can be used to simplify proofs of properties. Examples can be found in [Berghammer 91].

In the paper only the basic concepts for a theory of relational specifications have been presented. There are still a number of open problems that are subject of current and future research. In particular, to answer the question “how powerful is the approach for the treatment of practical problems” further experiments in specifying and proving relational specifications need to be conducted. The two examples of Section 4.2 indicate that the relational approach could be very useful for the development of graph-theoretic algorithms. It should also be useful in the case of relationships between (the domains of) two structures (examples are homomorphisms, simulations [Nipkow 86], or refinements [Desharnais 89], [Desharnais *et al.* 92]). As our notion of homomorphisms shows, the relational calculus is a convenient formalism for dealing with such concepts.

Acknowledgement. We gratefully acknowledge a number of valuable discussions with J. Desharnais, T. Gritzner, B. von Karger, and B. Möller, and thank them also for carefully reading a draft version of the paper.

References

[Berghammer 91] R. Berghammer, *Relational specification of data types and programs*, Report 9109, Univ. der Bundeswehr München, Fakultät für Informatik, 1991.

[Berghammer–Schmidt 91] R. Berghammer and G. Schmidt, *The RELVIEW-system*, in: C. Choffrut and M. Jantzen (eds.), Proc. STACS 91, Lecture Notes in Comput. Sci. 480, Springer, 1991, 535–536 .

[Berghammer *et al.* 89] R. Berghammer, G. Schmidt and H. Zierer, *Symmetric quotients and domain construction*, Inform. Process. Lett. 33 (3) (1989), 163–169.

[Berghammer–Zierer 86] R. Berghammer and H. Zierer, *Relational algebraic semantics of deterministic and nondeterministic programs*, Theoret. Comput. Sci. 43 (1986), 123–147.

[Brethauer 91] R. Brethauer, *Ein Formelmanipulationssystem zur computergestützten Beweisführung in der Relationenalgebra*, Diplomarbeit, Univ. der Bundeswehr München, Fakultät für Informatik, 1991.

[Chin–Tarski 51] L. H. Chin and A. Tarski, *Distributive and modular laws in the arithmetic of relation algebras*, Univ. California Publ. Math. 1 (1951), 341–384.

[de Bakker 71] J. W. de Bakker, *Recursive procedures*, Math. Centre Tracts 24, Mathematisch Centrum, Amsterdam 1971.

[de Bakker–de Roever 73] J. W. de Bakker and W. P. de Roever, *A calculus for recursive program schemes*, in: M. Nivat (ed.), Proc. ICALP 73, North-Holland, 1973, 167–196.

[de Roever 74] W. P. de Roever, *Recursion and parameter mechanisms: An axiomatic approach*, in: J. Loeckx (ed.), Proc. ICALP 74, Lecture Notes in Comput. Sci. 14, Springer, 1974, 34–65.

[Desharnais 89] J. Desharnais, *Abstract relational semantics*, Ph.D. thesis, McGill Univ., Montreal, Faculty of Graduate Studies and Research, 1989.

[Desharnais *et al.* 92] J. Desharnais, A. Jaoua, N. Belkhiter and F. Tchier, *Data refinement in a relation algebra*, in: Proc. Second Maghrebian Conference on Software Engineering and Artificial Intelligence, Tunis 1992, Fondation Nationale de la Recherche Scientifique, 1992, 222–236.

[Desharnais–Madhavji 90] J. Desharnais and N. H. Madhavji, *Abstract relational specifications*, in: M. Broy and J. B. Jones (eds.), Proc. TC 2 Working Conference on Programming Concepts and Methods, North-Holland, 1990, 267–284.

[Dijkstra 75] E. W. Dijkstra, *Guarded commands, nondeterminacy and the formal derivation of programs*, Comm. ACM 18 (1975) 453–457.

[Ehrig–Mahr 85] H. Ehrig and B. Mahr, *Fundamentals of Algebraic Specifications 1. Equations and Initial Semantics*, EATCS Monogr. Theoret. Comput. Sci. 6, Springer, 1985.

[Gritzner 89] T. F. Gritzner, *Die Axiomatik abstrakter Relationenalgebren: Darstellung der Grundlagen und Anwendung auf das Unschärfeproblem relationaler Produkte*, Diplomarbeit, Techn. Univ. München, Institut für Informatik, 1989.

[Hussmann 89] H. Hussmann, *Nichtdeterministische algebraische Spezifikationen*, Dissertation, Univ. Passau, Fakultät für Mathematik und Informatik, 1989.

[Kern 87] J. Kern, *Ein interaktives relationenalgebraisches Formelmanipulationssystem*, Diplomarbeit, Techn. Univ. München, Institut für Informatik, 1987.

[Manna 74] Z. Manna, *Mathematical Theory of Computation*, McGraw-Hill, New York 1974.

[Nipkow 86] T. Nipkow, *Nondeterministic data types: Models and implementations*, Acta Inform. 22 (1986), 629–661.

[Nguyen 91] T. T. Nguyen, *A relational model of demonic nondeterministic programs*, Internat. J. Found. Comput. Sci. 2 (1991), 101–131.

[Schmidt 84] M. Schmidt, *Behandlung abstrakter Typen auf relationenalgebraischer Grundlage*, Diplomarbeit, Techn. Univ. München, Institut für Informatik, 1984.

[Schmidt 86] D. A. Schmidt, *Denotational Semantics—A Methodology for Language Development*, Allyn and Bacon, 1986.

[Schmidt–Ströhlein 89] G. Schmidt and T. Ströhlein, *Relationen und Graphen*, Springer, 1989; English version: *Relations and Graphs*, EATCS Monogr. Theoret. Comput. Sci., Springer, 1992, to appear.

[Wirsing 90] M. Wirsing, *Algebraic specifications*, in: J. van Leeuwen (ed.), Handbook of Theoretical Computer Science B, North-Holland, 1990, 675–788.

[Wirsing–Broy 82] M. Wirsing and M. Broy, *An analysis of semantic models for algebraic specifications*, in: M. Broy and G. Schmidt (eds.), Theoretical Foundations of Programming Methodology, Reidel, 1982, 351–412.

[Zierer 83] H. Zierer, *Relationale Semantik*, Diplomarbeit, Techn. Univ. München, Institut für Informatik, 1983.

[Zierer 91] H. Zierer, *Relation algebraic domain constructions*, Theoret. Comput. Sci. 87 (1991), 163–188.