Journal on Relational Methods in Computer Science, Vol. 1, 2004, pp. 277 - 313

Refactoring Heterogeneous Relation Algebras around Ordered Categories and Converse*

Wolfram Kahl

Department of Computing and Software, McMaster University

Abstract. We present a reorganisation of popular theories of "reasoning with relational flavour", including allegories, Kleene algebras, and Dedekind categories, into an relatively symmetric picture using ordered categories as common base and defining converse independently from joins and meets.

As an example application, we use this to regroup results about formalisation of algebraic graph rewriting and thus exhibit opportunities for applying these approaches in new settings.

Finally we discuss how this approach influences the design of compatible approaches to formalisation and mechanisation of relation-algebraic theories.

1 Introduction

Various textbooks and collections have popularised a relation-based approach to reasoning in Computer Science [1, 14, 15, 77, 78] (see also the proceedings series [13, 22, 32, 35, 45, 67, 83]). Most of these assume the presence of all the axioms and laws of full relation algebra. Over time, a rich body of knowledge has been accumulated in this very expressive theory.

Currently, many of these formalisations are being re-assessed in the light of a growing number of sub-theories and amalgamated theories — for example the relation-algebraic treatment of pointers in [58] has recently been clarified and simplified through transfer into the more spartan setting of Kleene algebras with domain [24].

Two important branches of such subtheories have emerged as driving further development: On the one hand the *allegories* of Freyd and Scedrov [30], which

Received by the editors March 08, 2004, and, in revised form, October 14, 2004.
Published on December 10, 2004.

* This research is supported by NSERC (National Science and Engineering Research Council of Canada).

incorporate composition, meet (intersection), and converse, and on the other hand *Kleene algebras* which incorporate composition, join (union), and the *Kleene star* closure operator.

In each of these branches, an ordering is defined as a derived concept, either from meet, or from join. At the level of, for example, distributive allegories, where both meet and join are available, and are axiomatised as lattice operations of a single lattice, the orderings from the two branches naturally coincide. But at the lower levels, the two different definitions of the ordering produce technically distinct entities — when using a theorem prover, for example, it is non-trivial to transfer statements proven for the ordering derived from meet to theories where the ordering is derived from join.

In this paper, we propose a reorganisation of the subtheories of relation algebra in such a way that the common ground between the two directions is made explicit. In addition, we also elaborates the contribution of conversion independent of any (semi-)lattice structure.

We outline our theory organisation in Sect. 2, and then present a quick introduction into each of the theories in Sect. 3 and 4.

As an application of this reorganisation, we show in Sect. 5 how the relation-algebraic approach to graph structure transformation [42, 43] can be streamlined and have more symmetries exposed by carefully identifying the required context for each situation.

Since machine assistance for proving in relational theories is increasingly appreciated, and use of relational (sub-)theories as programming paradigms is becoming more and more popular, in Sect. 6 we survey previous approaches to formalisation and mechanisation of relation-algebraic theories and present current efforts to produce mechanised support that is compatible with the theory organisation outlined in Sect. 3.

2 Theory Organisation Overview

Like linear algebra has moved reasoning from the level of coefficients to the level of vectors, matrices, and their multiplication, relation algebra has moved from the level of element-wise reasoning “xRy” to reasoning using operations on relations, most prominently relational composition.

There are two basic approaches to axiomatisation of binary composition of binary relations: In the *homogeneous* approach, composition is a total operation on a single, “untyped” universe of “relations”, just like multiplication in monoids or rings. However, applications to concrete relations can profit from an understanding of relations as matrices of Boolean values, and then, as in linear algebra, composition does not work on all pairs of matrices, and therefore becomes a partial operation. This view is reflected in the *heterogeneous* approach, and can be axiomatised by considering "relations" as *typed* entities, each with a *source* and a *target* type, and composition $R;S$ is defined if and only if the target of R coincides with the source of S.

Composition with this kind of type discipline and in addition associativity and units is exactly what is provided by a *category*. Therefore it has become customary to use category-theoretic notions for presenting composition in a heterogeneous approach. In a category, what we informally called "type" is called *object*, and the arguments of composition are called *morphisms* (Sect. 3.1).

Since one of the hallmarks of calculational "relational" reasoning is the use of inclusion chains, we use *ordered categories* as common subtheory for all reasoning with "relation-algebraic flavour" (Sect. 3.2).

From this point, different directions may be taken:

- Adding *conversion* as involution operation enables the definition of many important properties such as univalence, totality, symmetry, equivalence, etc. (Sect. 3.4). This yields on the one hand the sub-category of mappings, and on the other hand can be used to abstractly characterise subobjects and quotient objects.

  Diagrams in ordered categories with converse (OCCs) correspond to a restricted kind of relational and algebraic structures, and allow to define *simulations* as a relational generalisation of structure homomorphisms. *Bisimulations* of structures over an OCC form again OCCs (Sect. 3.6).
- *Kleene categories*, the typed variant of Kleene algebras, add join and iteration to ordered categories (Sect. 4.2). A diagram in a Kleene category can be understood as a control-flow graph, which opens up numerous applications in computer science.
- A *domain* operation in an ordered category (Sect. 3.5) maps every morphism R to an endomorphism included in the identity on the source of R; this is useful for modelling "guards" even in the absence of meet.

  If such guard functionality is added to Kleene algebras, as in Kleene algebras with domain (KAD) [24] or Kleene algebras with tests (KAT) [52], then this yields a particularly useful tool for program analysis and verification.
- *Allegories* (Sect. 4.3) add meet and converse to ordered categories (from these two one can then *define* the domain operation). Diagrams in certain allegories can be understood as data-flow graphs.

  The duality between the data-flow and control-flow interpretations of relational diagrams is studied by Ştefănescu [16, 80, 81] in the calculus of flownomials which provides a syntax for such diagrams and can be equipped with either semantics.

Adding converse to Kleene categories adds the power to characterise direct sums (Sect. 4.5), while allegories already come equipped with the power to characterise direct products (Sect. 4.4).

3 Relational Concepts in Ordered Categories

We now present some basic concepts from category theory and then consider ordered categories and different extensions of ordered categories, but leave extensions that assume an upper or a lower semilattice structure in the ordering of each homset to Sect. 4. Even with this restriction, a remarkable array of relational concepts can be covered, in particular after the introduction of ordered categories with converse (OCCs) in Sect. 3.4.

3.1 Categories

We introduce categories for the sole purpose to serve as basic substrate of the composition aspects of all kinds of heterogenous relational algebras. Morphisms will therefore usually be relations, or other objects we want to consider as being "relation-like". Therefore, we use the symbol "$\leftrightarrow$" for declaring morphism types[1], thus reducing the usually rather heavy overloading of the single arrow "$\rightarrow$".

Definition 1. A *category* $\mathbf{C}$ is a tuple $(\mathsf{Obj}_{\mathbf{C}}, \mathsf{Mor}_{\mathbf{C}}, \mathsf{src}, \mathsf{trg}, \mathbb{I}, ;)$ where

- $\mathsf{Obj}_{\mathbf{C}}$ is a collection of *objects*.
- $\mathsf{Mor}_{\mathbf{C}}$ is a collection of *arrows* or *morphisms*.
- src (resp. trg) maps each morphism to its source (resp. target) object. Instead of $\mathsf{src}(f) = \mathcal{A} \wedge \mathsf{trg}(f) = \mathcal{B}$ we write $f : \mathcal{A} \leftrightarrow \mathcal{B}$. The collection of all morphisms f with $f : \mathcal{A} \leftrightarrow \mathcal{B}$ is denoted as $\mathsf{Mor}_{\mathbf{C}}[\mathcal{A}, \mathcal{B}]$ and also called a *homset*.
- "$;$" is the binary *composition* operator, and composition of two morphisms $f : \mathcal{A} \leftrightarrow \mathcal{B}$ and $g : \mathcal{B}' \leftrightarrow \mathcal{C}$ is defined iff $\mathcal{B} = \mathcal{B}'$, and then $(f;g) : \mathcal{A} \leftrightarrow \mathcal{C}$; composition is associative.
- $\mathbb{I}$ associates with every object $\mathcal{A}$ a morphism $\mathbb{I}_{\mathcal{A}}$ which is both a right and left unit for composition. □

[1] The use of a symmetric symbol does of course *not* introduce any assumptions about algebraic properties of that symbol: subtraction "$-$" is not commutative, either.

Frequently, the simple category-theoretic concepts of *span* and *co-span* is useful to achieve more concise formulations: A *span* is an ordered pair (f, g) of morphisms $f : \mathcal{A} \leftrightarrow \mathcal{B}$ and $g : \mathcal{A} \leftrightarrow \mathcal{C}$ with the same source. Such a span is often written $\mathcal{B} \xleftarrow{f} \mathcal{A} \xrightarrow{g} \mathcal{C}$. Analogously, a *co-span* is an ordered pair (h, k) of morphisms $h : \mathcal{B} \leftrightarrow \mathcal{D}$ and $k : \mathcal{C} \leftrightarrow \mathcal{D}$ with the same target, written $\mathcal{B} \xrightarrow{h} \mathcal{D} \xleftarrow{k} \mathcal{C}$.

Factorisation of identities induces important morphism properties:

Definition 2. For a morphism $R : \mathcal{A} \leftrightarrow \mathcal{B}$ in a category we define:

- R is *right-invertible* if there is a morphism $S : \mathcal{B} \leftrightarrow \mathcal{A}$ such that $R;S = \mathbb{I}_{\mathcal{A}}$. We then call S a *right-inverse* of R.
- R is *left-invertible* if there is a morphism $S : \mathcal{B} \leftrightarrow \mathcal{A}$ such that $S;R = \mathbb{I}_{\mathcal{B}}$. We then call S a *left-inverse* of R.
- R is an *isomorphism* if R is both right- and left-invertible.

□

A morphism is called an *endomorphism* iff its source and target objects coincide. In typed relation algebras, such morphisms are often called *homogeneous*.

Definition 3. An endomorphism $R : \mathcal{A} \leftrightarrow \mathcal{A}$ is *idempotent* if $R;R = R$. □

Certain kinds of idempotent morphisms are important since they characterise (uniquely up to isomorphism) in a general way objects that are induced by certain morphisms. As examples of this we will see subobjects and quotients below.

Obviously, if $S : \mathcal{B} \leftrightarrow \mathcal{A}$ is a right-inverse of $R : \mathcal{A} \leftrightarrow \mathcal{B}$, then

$$S;R;S;R = S;\mathbb{I}_{\mathcal{A}};R = S;R \ ,$$

so $S;R$ is idempotent. We adapt the nomenclature of Freyd and Scedrov [30] for this kind of situation:

Definition 4. If for an idempotent morphism $E : \mathcal{A} \leftrightarrow \mathcal{A}$ there are an object $\mathcal{X}$ and two morphisms $R : \mathcal{A} \leftrightarrow \mathcal{X}$ and $S : \mathcal{X} \leftrightarrow \mathcal{A}$ such that $S;R = \mathbb{I}_{\mathcal{X}}$ and $E = R;S$, then we say that the morphism pair $\mathcal{A} \xrightarrow{R} \mathcal{X} \xrightarrow{S} \mathcal{A}$ *splits* E, and E is then called a *split idempotent*. □

Proposition 5. [30, 1.282] Different splittings of the same idempotent morphism factor through unique morphisms, and these morphisms are isomorphisms. □

3.2 Ordered Categories

One of the hallmarks of the relational flavour of reasoning is the use of the *inclusion* ordering between morphisms. Adding this to categories gives rise to what is usually called “locally ordered categories”; for the sake of brevity, we shall use the name *ordered category.*

Definition 6. An *ordered category* is a category $\mathbf{C}$ such that

- for each two objects $\mathcal{A}$ and $\mathcal{B}$, the relation $\sqsubseteq_{\mathcal{A},\mathcal{B}}$ is a partial order on $\mathsf{Mor}_{\mathbf{C}}[\mathcal{A},\mathcal{B}]$ (the indices will usually be omitted), and
- composition is monotonic with respect to $\sqsubseteq$ in both arguments. □

A number of important endomorphism properties can already be defined using the language of ordered categories — the following are all standard except for co-transitivity:

Definition 7. For a morphism $R : \mathcal{A} \leftrightarrow \mathcal{A}$ in an ordered category we define the following properties:

- R is *reflexive* iff $\mathbb{I} \sqsubseteq R$,
- R is *transitive* iff $R{;}R \sqsubseteq R$,
- R is a *pre-order* iff R is reflexive and transitive,
- R is *co-reflexive* or a *sub-identity* iff $R \sqsubseteq \mathbb{I}_{\mathcal{A}}$,
- R is *co-transitive* iff $R \sqsubseteq R{;}R$. □

These properties are not completely independent of each other:

Lemma 8. In every ordered category, the following hold:

1. all co-reflexive morphisms are transitive;
2. all reflexive morphisms are co-transitive.

Proof. 1. $R \sqsubseteq \mathbb{I}_{\mathcal{A}}$ implies $R{;}R \sqsubseteq \mathbb{I}_{\mathcal{A}}{;}R = R$.
2. Dually, $\mathbb{I}_{\mathcal{A}} \sqsubseteq R$ implies $R = \mathbb{I}_{\mathcal{A}}{;}R \sqsubseteq R{;}R$. □

A direct consequence of this is:

Corollary 9. All pre-orders and all co-transitive sub-identities are idempotent. □

This also shows that idempotent sub-identities are exactly the co-transitive and co-reflexive morphisms, and therefore dual to preorders. Idempotent sub-identities have been identified by Desharnais *et al.* [24] as playing an important rôle when considering domain operators, see Sect. 3.5.

For homsets that have least or greatest elements, we introduce corresponding notation:

Definition 10. In an ordered category, for each two objects $\mathcal{A}$ and $\mathcal{B}$ we introduce the following notions:

- If the homset $\mathsf{Mor}_{\mathbf{C}}[\mathcal{A},\mathcal{B}]$ contains a greatest element, then this is denoted $\mathbb{T}_{\mathcal{A},\mathcal{B}}$.
- If the homset $\mathsf{Mor}_{\mathbf{C}}[\mathcal{A},\mathcal{B}]$ contains a least element, then this is denoted $\perp\!\!\!\perp_{\mathcal{A},\mathcal{B}}$.□

For these extremal morphisms and for identities we frequently omit indices where these can be induced from the context.

Existence of least morphisms is usually assumed together with the *zero law*:

Definition 11. An *ordered category with zero morphisms* is an ordered category such that

- each homset $\mathsf{Mor}_{\mathbf{C}}[\mathcal{A},\mathcal{B}]$ has a least element $\perp\!\!\!\perp_{\mathcal{A},\mathcal{B}}$, and
- each least element $\perp\!\!\!\perp_{\mathcal{A},\mathcal{B}}$ is a left- and right-zero for composition. □

3.3 Residuation

Given an ordering $\leq$ and a binary operator $\otimes$, an element R is called *right-residual of S with respect to P* iff for all X (of appropriate type),

$$P \otimes X \leq S \qquad \Leftrightarrow \qquad X \leq R$$

Analogously, P is called *left-residual of S with respect to R* iff for all X (of appropriate type),

$$X \otimes R \leq S \qquad \Leftrightarrow \qquad X \leq P$$

For commutative operators, one just talks about “residuals”.

Where a certain kind of residual always exists, this gives rise to a Galois connection; this fact makes many useful laws available, see for example [1, 70].

For example, Boolean implication $R \Rightarrow S$ is the residual of conjunction with respect to the implication ordering:

$$(X \wedge R) \Rightarrow S \qquad \Leftrightarrow \qquad X \Rightarrow (R \Rightarrow S)$$

Such residuals exist automatically if the relevant ordering is complete. If this is not the case, they are frequently introduced axiomatically as existing for certain operators.

The plain names “left- and right-residual” are used for composition in ordered categories; where these residuals exist, we have for $X, P : \mathcal{A} \leftrightarrow \mathcal{B}$ and $Y, R : \mathcal{B} \leftrightarrow \mathcal{C}$ and $S : \mathcal{A} \leftrightarrow \mathcal{C}$:

$$\begin{array}{llll} P;Y \sqsubseteq S & \Leftrightarrow & Y \sqsubseteq (P\backslash S) & \textit{right-residual} \\ X;R \sqsubseteq S & \Leftrightarrow & X \sqsubseteq (S/R) & \textit{left-residual} \end{array}$$

$$\begin{tikzcd}
\mathcal{A} \arrow[rr, "S"] \arrow[dr, "P"'] & & \mathcal{C} \\
& \mathcal{B} \arrow[ur, "P\backslash S"', very thick] &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\mathcal{A} \arrow[rr, "S"] \arrow[dr, "S/R"', very thick] & & \mathcal{C} \\
& \mathcal{B} \arrow[ur, "R"'] &
\end{tikzcd}$$

These residuals of composition frequently open alternative ways of obtaining results that would otherwise be obtained using converse.

3.4 Ordered Categories with Converse (OCCs)

We now introduce converse into ordered categories *before* we consider join or meet, and we shall see that, even with only the basic axiomatisation of converse as an involution, the resulting theory is quite expressive.

Definition 12. An *ordered category with converse (OCC)* is an ordered category $\mathbf{C}$ such that

- each morphism $R : \mathcal{A} \leftrightarrow \mathcal{B}$ has a *converse* $R^{\smile} : \mathcal{B} \leftrightarrow \mathcal{A}$,
- the *involution equations* hold for all $R : \mathcal{A} \leftrightarrow \mathcal{B}$ and $S : \mathcal{B} \leftrightarrow \mathcal{C}$:

$$\begin{aligned}(R^{\smile})^{\smile} &= R\\ \mathbb{I}_{\mathcal{A}}^{\smile} &= \mathbb{I}_{\mathcal{A}}\\ (R;S)^{\smile} &= S^{\smile};R^{\smile}\end{aligned}$$

- conversion is monotonic with respect to $\sqsubseteq$. □

Many standard properties of relations can be characterised in the context of OCCs — the following set is closed both under $^{\smile}$-duality and under $\sqsubseteq$-duality:

Definition 13. For a morphism $R : \mathcal{A} \leftrightarrow \mathcal{B}$ in an OCC we define:

- R is *univalent* iff $R^{\smile};R \sqsubseteq \mathbb{I}_{\mathcal{B}}$,
- R is *total* iff $\mathbb{I}_{\mathcal{A}} \sqsubseteq R;R^{\smile}$,
- R is *injective* iff $R;R^{\smile} \sqsubseteq \mathbb{I}_{\mathcal{A}}$,
- R is *surjective* iff $\mathbb{I}_{\mathcal{B}} \sqsubseteq R^{\smile};R$,
- R is a *mapping* iff R is univalent and total,
- R is *bijective* iff R is injective and surjective. □

Furthermore, we denote the sub-category of an OCC $\mathbf{C}$ that contains all objects of $\mathbf{C}$, but only mappings as arrows with $\mathsf{Map}\,\mathbf{C}$, and the sub-category of all partial functions (i.e., univalent morphisms) with $\mathsf{Pfn}\,\mathbf{C}$.

For endomorphisms, there are a few additional properties of interest:

Definition 14. For a morphism $R : \mathcal{A} \leftrightarrow \mathcal{A}$ in an OCC we define:

- R is *symmetric* iff $R^{\smile} \sqsubseteq R$,
- R is an *equivalence* iff R is a symmetric pre-order,
- R is a *co-equivalence* iff R is co-reflexive, co-transitive, and symmetric. □

The concept of co-equivalence is the strict $\sqsubseteq$-dual of the concept of equivalence since symmetry is self-dual: using monotonicity of converse and involution one sees that $R^{\smile} \sqsubseteq R$ and $R \sqsubseteq R^{\smile}$ are equivalent; therefore each alone is also equivalent to $R^{\smile} = R$.

Some interesting properties are collected in the following list:

Lemma 15. In every OCC, the following hold:

1. R is a co-equivalence iff R is a symmetric idempotent sub-identity.
2. If $F : \mathcal{A} \leftrightarrow \mathcal{B}$ is a mapping, then $F{;}F^{\smile}$ is an equivalence.
3. If $F : \mathcal{A} \leftrightarrow \mathcal{B}$ is a mapping, then $F^{\smile}{;}F$ is a co-equivalence.

Proof. 1. From Corollary 9.

2. $F{;}F^{\smile}$ is obviously symmetric, and reflexive since F is total. For transitivity we need univalence: $F{;}F^{\smile}{;}F{;}F^{\smile} \sqsubseteq F{;}\mathbb{I}_{\mathcal{B}}{;}F^{\smile} = F{;}F^{\smile}$
3. Dually, $F^{\smile}{;}F$ is obviously symmetric, and a co-reflexive since F is univalent. For co-transitivity we need totality: $F^{\smile}{;}F = F^{\smile}{;}\mathbb{I}_{\mathcal{A}}{;}F \sqsubseteq F^{\smile}{;}F{;}F^{\smile}{;}F$ □

Each of the last two items can be generalised by replacing the two occurences of the mapping F by two *different* mappings; the more general resulting properties are the following:

Definition 16. In an OCC $\mathbf{C}$, for a morphism $R : \mathcal{A} \leftrightarrow \mathcal{B}$ we define the following properties:

- R is *difunctional* iff $R{;}R^{\smile}{;}R \sqsubseteq R$,
- R is *co-difunctional* iff $R \sqsubseteq R{;}R^{\smile}{;}R$.
- R is *strictly difunctional* iff $R{;}R^{\smile}{;}R = R$,

An OCC is called *(strictly) (co-)difunctional* if all morphisms are (strictly) (co-)difunctional. □

Difunctionality of relations has been studied in particular by Riguet [71] and Schmidt [9, 78]. As mentioned above, difunctionality is motivated by the following generalisation of Lemma 15.(2) and (3).

Lemma 17. If a morphism $R : \mathcal{A} \leftrightarrow \mathcal{B}$ in an OCC can be represented by a co-span $\mathcal{A} \xrightarrow{F} \mathcal{X} \xleftarrow{G} \mathcal{B}$ of univalent morphisms in the sense that $R = F{;}G^{\smile}$, then R is difunctional.

Dually, if R can be represented by a span $\mathcal{A} \xleftarrow{H} \mathcal{X} \xrightarrow{K} \mathcal{B}$ of total morphisms in the sense that $R = H^{\smile}{;}K$, then R is co-difunctional.

Proof. Assuming $R = F{;}G^{\smile}$, then univalence of F and G implies

$$R{;}R^{\smile}{;}R = F{;}G^{\smile}{;}G{;}F^{\smile}{;}F{;}G^{\smile} \sqsubseteq F{;}\mathbb{I}_{\mathcal{X}}{;}\mathbb{I}_{\mathcal{X}}{;}G^{\smile} = F{;}G^{\smile} = R\ .$$

Dually, assuming $R = H^{\smile}{;}K$, totality of H and K implies:

$$R = H^{\smile}{;}K = H^{\smile}{;}\mathbb{I}_{\mathcal{X}}{;}\mathbb{I}_{\mathcal{X}}{;}K \sqsubseteq H^{\smile}{;}K{;}K^{\smile}{;}H{;}H^{\smile}{;}K{;} = R{;}R^{\smile}{;}R \qquad \square$$

We now make the relation between the difunctionality properties and those they generalise explicit:

Lemma 18. 1. Each equivalence or co-equivalence is strictly difunctional.
2. Each morphism that is co-difunctional and co-reflexive is a co-equivalence.
3. Each morphism that is difunctional and reflexive is an equivalence.

Proof. 1. Assume $R : \mathcal{A} \leftrightarrow \mathcal{A}$ is an equivalence or co-equivalence. Then R is idempotent and symmetric, and we have: $R = R{;}R = R{;}R{;}R = R{;}R^{\smile}{;}R$.
2. Assume that $R : \mathcal{A} \leftrightarrow \mathcal{A}$ is both co-difunctional $R \sqsubseteq R{;}R^{\smile}{;}R$ and co-reflexive $R \sqsubseteq \mathbb{I}_{\mathcal{A}}$. From this we obtain symmetry: $R \sqsubseteq R{;}R^{\smile}{;}R \sqsubseteq \mathbb{I}_{\mathcal{A}}{;}R^{\smile}{;}\mathbb{I}_{\mathcal{A}} = R^{\smile}$, and, in a similar way, co-transitivity: $R \sqsubseteq R{;}R^{\smile}{;}R \sqsubseteq R{;}\mathbb{I}_{\mathcal{A}}^{\smile}{;}R = R{;}\mathbb{I}_{\mathcal{A}}{;}R = R{;}R$.
3. Dual to 2. □

While our OCC axioms correspond to what Desharnais *et al.* call "preconverse" in [24] (with the difference that we consider converse in the absence of join, starting from the inclusion ordering instead), co-difunctionality of OCCs is the condition used for "idempotent semiring with converse" in [24], motivated by Ésik and Bernátsky who showed that co-difunctionality holds in the variety generated by all full Kleene algebras *of relations* with converse [29].

While in general OCCs, not even all idempotent sub-identities need to be symmetric, Lemma 18.2) shows that co-difunctionality implies a very strong property:

Lemma 19. In a co-difunctional OCC, all sub-identities are co-equivalences. □

As we shall see later, in allegories all morphisms are co-difunctional. In that context, the conclusion of the following lemma collapses to an unconditional equality $F^{\smile} = G$ and has been shown by Freyd and Scedrov [30, 2.162]. In general OCCs, we obtain:

Lemma 20. If $\mathcal{A} \xrightarrow{F} \mathcal{X} \xrightarrow{G} \mathcal{A}$ splits a symmetric idempotent $E : \mathcal{A} \leftrightarrow \mathcal{A}$, then:

1. $F^{\smile} \sqsubseteq G$ if G is co-difunctional or F is difunctional;
2. $G^{\smile} \sqsubseteq F$ if F is co-difunctional or G is difunctional.

Proof. We only show (1) since (2) is dual. Assume the stated splitting, i.e., $G{;}F = \mathbb{I}_{\mathcal{X}}$ and $F{;}G = E$. Then $F{;}G = E^{\smallsmile} = (F{;}G)^{\smallsmile}$, and we have:

- if G is co-difunctional, i.e., $G \sqsubseteq G{;}G^{\smallsmile}{;}G$, then:

$$F^{\smallsmile} = G{;}F{;}F^{\smallsmile} \sqsubseteq G{;}G^{\smallsmile}{;}G{;}F{;}F^{\smallsmile} = G{;}G^{\smallsmile}{;}F^{\smallsmile} = G{;}(F{;}G)^{\smallsmile} = G{;}F{;}G = G$$

- if F is difunctional, i.e., $F{;}F^{\smallsmile}{;}F \sqsubseteq F$, then:

$$\begin{aligned} F^{\smallsmile} &= G{;}F{;}F^{\smallsmile} = G{;}F{;}G{;}F{;}F^{\smallsmile} = G{;}(F{;}G)^{\smallsmile}{;}F{;}F^{\smallsmile} = G{;}G^{\smallsmile}{;}F^{\smallsmile}{;}F{;}F^{\smallsmile} \\ &\sqsubseteq G{;}G^{\smallsmile}{;}F^{\smallsmile} = G{;}(F{;}G)^{\smallsmile} = G{;}F{;}G = G \end{aligned}$$ □

If both items of this lemma are satisfied, for example in co-difunctional OCCs, we obtain $G = F^{\smallsmile}$.

In general, if $\mathcal{A}\xrightarrow{F}\mathcal{X}\xrightarrow{F^{\smallsmile}}\mathcal{A}$ splits an idempotent $E : \mathcal{A} \leftrightarrow \mathcal{A}$, we call this a *symmetric splitting* of E. If an idempotent E has a symmetric splitting, then E is symmetric: $E = F{;}F^{\smallsmile} = (F{;}F^{\smallsmile})^{\smallsmile} = E^{\smallsmile}$. Special cases of this are quotients for equivalences and subobjects for co-equivalences; defining these via splittings has the advantage that their uniqueness up to isomorphism is already shown by Proposition 5.

Definition 21. A *quotient* for an equivalence $\Xi : \mathcal{A} \leftrightarrow \mathcal{A}$ is a pair $(\mathcal{Q}, H)$ consisting of a *quotient object* $\mathcal{Q}$ and a *projection* morphism $H : \mathcal{A} \leftrightarrow \mathcal{Q}$ such that $\mathcal{A}\xrightarrow{H}\mathcal{Q}\xrightarrow{H^{\smallsmile}}\mathcal{A}$ is a symmetric splitting of Ξ.

An OCC **C** *has quotients* iff for every equivalence there is a quotient. □

It is easy to see that the quotient projection is a surjective mapping: surjectivity and univalence together are equivalent to the splitting property $H^{\smallsmile}{;}H = \mathbb{I}_{\mathcal{Q}}$, and totality follows from reflexivity of the equivalence: $\mathbb{I}_{\mathcal{A}} \sqsubseteq \Xi = H{;}H^{\smallsmile}$.

Dually, subobject injections are injective mappings:

Definition 22. A *subobject* for a co-equivalence $q : \mathcal{A} \leftrightarrow \mathcal{A}$ is a pair $(\mathcal{S}, J)$ consisting of an object $\mathcal{S}$ and a *injection* morphism $J : \mathcal{S} \leftrightarrow \mathcal{A}$ such that $\mathcal{A}\xrightarrow{J^{\smallsmile}}\mathcal{Q}\xrightarrow{J}\mathcal{A}$ is a symmetric splitting of q.

The OCC **C** *has subobjects* iff for every co-equivalence there is a subobject. □

A number of important more complex constructions than subobjects and quotients starts from commuting square diagrams in category theory, in particular pullbacks and pushouts as archetypes of limits and colimits in categories.

Central to the connection between pullbacks and pushouts in categories of mappings on the one hand and constructions in relational theories on the other hand is the fact a square of mappings commutes iff the “relation” induced by the source span is contained in that induced by the target co-span. The allegory proof by Freyd and Scedrov [30, 2.146] really only uses OCC material:

Lemma 23. Given a span $\mathcal{B} \xleftarrow{P} \mathcal{A} \xrightarrow{Q} \mathcal{C}$ and a co-span $\mathcal{B} \xrightarrow{R} \mathcal{D} \xleftarrow{S} \mathcal{C}$ of mappings in an OCC, we have $P{;}R = Q{;}S$ iff $P^{\smile}{;}Q \sqsubseteq R{;}S^{\smile}$.

$$\begin{tikzcd}
& \mathcal{A} \arrow[dl, "P"'] \arrow[dr, "Q"] & \\
\mathcal{B} \arrow[dr, "R"] & & \mathcal{C} \arrow[dl, "S"'] \\
& \mathcal{D} &
\end{tikzcd}$$

Proof. $P{;}R = Q{;}S$ implies

$$\begin{aligned}
P^{\smile}{;}Q &\sqsubseteq P^{\smile}{;}Q{;}S{;}S^{\smile} && S \text{ total}\\
&= P^{\smile}{;}P{;}R{;}S^{\smile} && \text{assumption}\\
&\sqsubseteq R{;}S^{\smile} && P \text{ univalent.}
\end{aligned}$$

Conversely, $P^{\smile}{;}Q \sqsubseteq R{;}S^{\smile}$ implies

$$\begin{aligned}
Q{;}S &\sqsubseteq P{;}P^{\smile}{;}Q{;}S && P \text{ total}\\
&\sqsubseteq P{;}R{;}S^{\smile}{;}S && \text{assumption}\\
&\sqsubseteq P{;}R && S \text{ univalent}\\
&\sqsubseteq Q{;}Q^{\smile}{;}P{;}R && Q \text{ total}\\
&\sqsubseteq Q{;}S{;}R^{\smile}{;}R && \text{assumption}\\
&\sqsubseteq Q{;}S && R \text{ univalent.}
\end{aligned}$$ □

This implies that when looking for a pushout of the span $\mathcal{B} \xleftarrow{P} \mathcal{A} \xrightarrow{Q} \mathcal{C}$, the identity of the two mappings P and Q does not matter, we only need to consider the diagonal $P^{\smile}{;}Q$. Dually, when looking for a pullback of the co-span $\mathcal{B} \xrightarrow{R} \mathcal{D} \xleftarrow{S} \mathcal{C}$, only $R{;}S^{\smile}$ needs to be considered. The gap can be significant: according to Lemma 17, $P^{\smile}{;}Q$ is always co-difunctional, while $R{;}S^{\smile}$ is always difunctional.

3.5 Domain

Desharnais *et el.* [24] discussed the definition of domain and range operators[2] for Kleene algebras in considerable detail, and much of the material there can be transferred into the setting of ordered categories by replacing preservation of

[2] It is important not to confuse these *domain* and *range* operations, which only make sense in ordered categories, with the categorical concepts of source and target of a morphism!

joins with monotonicity. In particular, their core definition of "predomain" only requires ordered categories and is given as a special residual of composition with respect to the ordering $\sqsupseteq$:

Definition 24. An *ordered category with predomain* is an ordered category where for every morphism $R : \mathcal{A} \leftrightarrow \mathcal{B}$ there is a morphism $\mathsf{dom}\, R : \mathcal{A} \leftrightarrow \mathcal{A}$ such that for every $X : \mathcal{A} \leftrightarrow \mathcal{A}$, we have:

$$X{;}R \sqsupseteq R \qquad \Leftrightarrow \qquad X \sqsupseteq \mathsf{dom}\, R$$

In an *ordered category with domain*, additionally the following "locality" condition holds:

$$\mathsf{dom}\,(R{;}S) = \mathsf{dom}\,(R{;}\mathsf{dom}\, S) \qquad \square$$

Co-reflexivity of $\mathsf{dom}\, R$ follows by substitution $\mathbb{I}_{\mathcal{A}}$ for X in the predomain condition.

Range can be defined similarly; an OCC with domain also has range, and range is then related with domain via converse:

$$\mathsf{ran}\, R = (\mathsf{dom}\,(R^{\smile}))^{\smile}$$

(Without co-difunctionality, sub-identities need not be symmetric, so we need the outer conversion, too.)

3.6 Σ-Structures and OCCs of Bisimulations

A *diagram over* a category $\mathbf{C}$ is a graph Σ together with a graph homomorphism into the graph underlying $\mathbf{C}$. It is frequently useful to interpret the graph Σ as a *unary signature* $\Sigma = (\mathcal{S}, \mathcal{F}, \mathsf{src}, \mathsf{trg})$ where

- $\mathcal{S}$ is a set, the elements of which are called *sorts*,
- $\mathcal{F}$ is a set, the elements of which are called *(function or relation) symbols*, and
- $\mathsf{src}, \mathsf{trg} : \mathcal{F} \to \mathcal{S}$ map each symbol to its source and target sort, respectively. As usual, we write "$f : s \to t$" to denote "$\mathsf{src}(f) = s$ and $\mathsf{trg}(f) = t$".

From this point of view, a diagram D with graph Σ is now considered as a *Σ-structure D over* $\mathbf{C}$, which consists of:

- a *carrier object* s^D for each sort $s \in \mathcal{S}$, and
- an *interpretation* morphism $f^D : s^D \leftrightarrow t^D$ for each symbol $f \in \mathcal{F}$ with $f : s \to t$.

If **C** is an OCC, then a *Σ-algebra* over **C** is a Σ-structure over **C** in which all symbols are interpreted as mappings.

For general Σ-structures, we are now aiming towards a homomorphism concept where homomorphisms have to behave "essentially like relations", and so it is only natural that we consider a relational generalisation of conventional (functional) Σ-homomorphisms.

This is closely related to the field of data refinement, where usually unary homogeneous operations $f : s \to s$ over a signature with single sort s are considered, and interpretations are allowed to be arbitrary relations, see for example the book by de Roever and Engelhardt [72]. In that context, an "L-simulation" from $\mathcal{A}$ to $\mathcal{B}$ is a relation $\Phi : s^{\mathcal{A}} \leftrightarrow s^{\mathcal{B}}$ satisfying the following inclusion for every operation f:

$$\Phi^{\smile};f^{\mathcal{A}} \sqsubseteq f^{\mathcal{B}};\Phi^{\smile}$$

The name "L-simulation" is derived from the L-shape of the inclusion's left-hand side in the following sub-commuting diagram:

$$\begin{tikzcd}
s^{\mathcal{B}} \arrow[r, "f^{\mathcal{B}}"] & s^{\mathcal{B}} \\
s^{\mathcal{A}} \arrow[u, "\Phi"] \arrow[r, "f^{\mathcal{A}}"'] \arrow[ur, phantom, "\sqsubseteq"] & s^{\mathcal{A}} \arrow[u, "\Phi"']
\end{tikzcd}$$

A detailed relation-algebraic analysis of the different simulation concepts presented by de Roever and Engelhardt [72] and of L-bisimulation has been performed by Khedri [49].

In our context of Σ-structures, a symbol f may have different source and target sorts, so instead of a single simulation relation Φ, we need a family of simulation relations Φ_s, one for each sort s.

For a simulation Φ of Σ-structures from $\mathcal{A}$ to $\mathcal{B}$, the following diagram has to sub-commute for every symbol $f : s \to t$:

$$\begin{tikzcd}
\mathcal{B} & & s^{\mathcal{B}} \arrow[r, "f^{\mathcal{B}}"] & t^{\mathcal{B}} \\
\mathcal{A} \arrow[u, "\Phi"] & & s^{\mathcal{A}} \arrow[u, "\Phi_s"] \arrow[r, "f^{\mathcal{A}}"'] \arrow[ur, phantom, "\sqsubseteq"] & t^{\mathcal{A}} \arrow[u, "\Phi_t"']
\end{tikzcd}$$

Definition 25. Let a unary signature $\Sigma = (\mathcal{S}, \mathcal{F}, \mathsf{src}, \mathsf{trg})$ and an OCC $\mathbf{C}$ be given, and let $\mathcal{A}$ and $\mathcal{B}$ be two Σ-structures over $\mathbf{C}$.

A *Σ-compatible family of morphisms from $\mathcal{A}$ to $\mathcal{B}$* is an $\mathcal{S}$-indexed family of morphisms, $\Phi = (\Phi_s)_{s:\mathcal{S}}$, such that $\Phi_s : s^{\mathcal{A}} \leftrightarrow s^{\mathcal{B}}$ for every sort $s : \mathcal{S}$.

Such a Φ is called a *Σ-simulation from $\mathcal{A}$ to $\mathcal{B}$* if for every symbol $f \in \mathcal{F}$ with $f : s \to t$, the following inclusion holds:

$$\Phi_s^{\smile} ; f^{\mathcal{A}} \sqsubseteq f^{\mathcal{B}} ; \Phi_t^{\smile} \ .$$

Φ is called a *Σ-bisimulation from $\mathcal{A}$ to $\mathcal{B}$* if in addition, for every symbol $f \in \mathcal{F}$ with $f : s \to t$, also the following inclusion holds:

$$\Phi_s ; f^{\mathcal{B}} \sqsubseteq f^{\mathcal{A}} ; \Phi_t \ . \qquad \square$$

From this, we immediately obtain ordered categories of simulations:

Proposition 26. For each unary signature $\Sigma = (\mathcal{S}, \mathcal{F}, \mathsf{src}, \mathsf{trg})$, Σ-structures over an OCC $\mathbf{C}$ together with Σ-simulations form an ordered category, denoted $\mathbf{C}^{\Sigma}$.

Proof. Identities, composition, and inclusion in $\mathbf{C}^{\Sigma}$ are defined component-wise — this implies the identity laws and associativity of composition. Identities are obviously well-defined, and well-definedness of the composition of $\Phi : \mathcal{A} \leftrightarrow \mathcal{B}$ and $\Psi : \mathcal{B} \leftrightarrow \mathcal{C}$ follows easily:

$$(\Phi_s ; \Psi_s)^{\smile} ; f^{\mathcal{A}} = \Psi_s^{\smile} ; \Phi_s^{\smile} ; f^{\mathcal{A}} \sqsubseteq \Psi_s^{\smile} ; f^{\mathcal{B}} ; \Phi_t^{\smile} \sqsubseteq f^{\mathcal{C}} ; \Psi_t^{\smile} ; \Phi_t^{\smile} = f^{\mathcal{C}} ; (\Phi_t ; \Psi_t)^{\smile} \ .$$

The component-wise definition of inclusion implies that inclusion in each homset of $\mathbf{C}^{\Sigma}$ is a partial ordering again, and, together with the component-wise definition of composition, also yields monotonicity of composition. □

We can also lift the conversion operation via a component-wise definition:

$$\Phi^{\smile} := (\Phi_s^{\smile})_{s:\mathcal{S}} \ .$$

This component-wise definition produces a monotonic operator on Σ-compatible morphism families, and also guarantees the involution equations. However, $\Phi^{\smile}$ is in general *not* a Σ-simulation again, even if Φ is. But since the bisimulation condition is dual to the simulation condition, we have:

Proposition 27. A simulation Φ is a bisimulation if and only if $\Phi^{\smile}$ is a simulation. □

In [42], where we considered the restriction of Σ-simulations to Σ-algebras under the name of "relational Σ-homomorphisms", we showed that this converse is well-defined thanks to the mapping properties of the interpretations of the (function) symbols:

Proposition 28. If $\mathcal{A}$ and $\mathcal{B}$ are Σ-algebras, then every simulation $\Phi : \mathcal{A} \to \mathcal{B}$ is a bisimulation. □

In any case, we obtain OCCs of bisimulations by combining Proposition 27 with Proposition 26:

Proposition 29. For each unary signature $\Sigma = (\mathcal{S}, \mathcal{F}, \mathsf{src}, \mathsf{trg})$, Σ-structures over an OCC $\mathbf{C}$ together with Σ-bisumulations form an OCC, denoted $\mathbf{C}^{\Sigma^{\smile}}$. □

Since this does not make any assumptions over Σ-structures, it applies to relational structures in the same way as to algebraic structures. In particular, it makes all the properties and constructions of Sect. 3.4 available for bisimulation OCCs of relational structures.

For example, equivalences in Σ-bisimulation OCCs are exactly Σ-congruences; for the case of Σ-algebras, this has been shown in [42, The. 3.3.4]. Schmidt discusses congruences and multi-coverings of relational structures in [79, Sect. 3.3]; it is easy to see that these congruences are just equivalences in OCCs of bisimulations, and multi-coverings are the corresponding quotient projections.

4 Allegories and Kleene Categories

We now consider ordered categories where joins, repectively meets, have to exist in the ordering of each homset (Sect. 4.1), and obtain as most important representants of the resulting two direction on the one hand Kleene categories (Sect. 4.2), and on the other hand allegories (Sect. 4.3).

The binary operations meet and join can now be used to define tabulations and co-tabulations that can be seen as relational formulations of pullbacks and pushouts in mapping categories.

In comparison with the category-theoretic definitions of pullbacks and pushouts, the "relational" definitions are perfectly *local* in that they involve only the morphisms under discussion. In addition, they are syntactically first-order, since they do not use quantification, so their relational treatment makes reasoning much more accessible both for human readers and for mechanised proof checking.

The attraction of the alternative characterisations in Props. 42 and 45 is that they allow purely equational reasoning concerning all aspects of (co-)tabulations.

Tabulations and co-tabulations can also be seen as generalisations of the symmetric splittings in the subobject and quotient constructions (Defs. 21 and 22). This is in parallels the way we showed how equivalences and co-equivalences can be seen as generalising to difunctionality and co-difunctionality, which accordingly play an important rôle in this context.

As special cases of tabulations and co-tabulations we obtain direct products and sums (Sects. 4.4 and 4.5), and arriving at their definition via this route provides a nice explanation for what otherwise might be perceived to be a flaw in their duality.

Finally, we mention a number of important models of these theories (Sect. 4.6).

4.1 Semilattice Categories

We now consider ordered categories where homsets have (semi-)lattice structure.

Definition 30. A *lower semilattice category* is an ordered category such that each homset is a lower semilattice with binary meet $\sqcap$. □

In contexts where the inclusion ordering $\sqsubseteq$ is not primitive, but defined using meet, the following property is usually listed as an axiom; here it follows from monotonicity of composition:

Lemma 31. In a lower semilattice category, for all morphisms $P : \mathcal{A} \leftrightarrow \mathcal{B}$ and $Q, R : \mathcal{B} \leftrightarrow \mathcal{C}$, and $S : \mathcal{C} \leftrightarrow \mathcal{D}$, *meet-subdistributivity* holds:

$$P;(Q \sqcap R) \sqsubseteq P;Q \sqcap P;R$$
$$(Q \sqcap R);S \sqsubseteq Q;S \sqcap R;S$$
□

Pseudo-complements are residuation of meet in lower semilattice categories; where pseudo-complements exist, we denote the pseudo-complement or R with respect to S as $R \rightarrow S$, and we have:

$$X \sqcap R \sqsubseteq S \qquad \Leftrightarrow \qquad X \sqsubseteq (R \rightarrow S)$$

Dually, we obtain "join-superdistributivity" in the presence of joins. However, it is customary to demand distributivity of composition over join, so upper semilattice categories are *not* completely dual to lower semilattice categories:

Definition 32. An *upper semilattice category* is an ordered category such that

- each homset is an upper semilattice with binary join $\sqcup$, and
- composition distributes over joins from both sides. □

If we consider *upper* or *lower semilattice categories with converse*, i.e., upper respectively lower semilattice categories that are at the same time OCCs, then the involution law for join respectively meet follows from isotony of converse.

One frequently assumes the existence of least upper bounds with respect to $\sqsubseteq$ of arbitrary subsets of homsets:

Definition 33. An upper semilattice category $\mathbf{C}$ is called *complete* if

- each homset $\mathsf{Mor}_{\mathbf{C}}[\mathcal{A},\mathcal{B}]$ is a *complete* upper semilattice, i.e., for each subset $\mathcal{S}$ of $\mathsf{Mor}_{\mathbf{C}}[\mathcal{A},\mathcal{B}]$, the join $\bigsqcup \mathcal{S}$ exists, and
- composition distributes over arbitrary joins from both sides. □

In a complete upper semilattice category, each homset automatically has a least element, but this does not have to be a zero — counter-examples arise for example with simulations of Σ-algebras with constant symbols.

The result of adding zero morphisms to upper semilattice categories can be seen as a heterogoneous version of *idempotent semirings*, which come equipped with the two binary operations of "multiplication" (or composition) and "addition" (or join), and two constants, "0" which is a unit for addition and a zero for multiplication, and "1" as a unit for multiplication. This motivates out choice of name for the corresponding class of categories:

Definition 34. An *idempotent semiring category* (*ISR category*) is an upper semilattice categories with zero morphisms. □

4.2 Kleene Categories

Kleene algebras are a generalisation of the algebra of regular languages, and are frequently presented as extensions of *idempotent semirings*, where "multiplication" corresponds to concatenation of languages, "addition" to union of languages, "0" to the empty language, and "1" to the language containing only the empty word.

On top of the idempotent semiring structure, a Kleene algebra also has one unary operation, the Kleene star, for which we use an axiomatisation by Kozen [50]. Most treatments of Kleene algebras are untyped; for a typed version see [53] — in the categorical setting, the Kleene star is defined only on endomorphisms:

Definition 35. A *Kleene category* is an ISR category such that on homsets of endomorphisms there is an additional unary operation $\_^*$ which satisfies the following axioms for all $R : \mathcal{A} \leftrightarrow \mathcal{A}$, $Q : \mathcal{B} \leftrightarrow \mathcal{A}$, and $S : \mathcal{A} \leftrightarrow \mathcal{C}$:

$$\begin{array}{rclcll} & & R^* & = & \mathbb{I}_{\mathcal{A}} \sqcup R \sqcup R^* ; R^* & \text{recursive star definition} \\ Q ; R \sqsubseteq Q & \Rightarrow & Q ; R^* & \sqsubseteq & Q & \text{right induction} \\ R ; S \sqsubseteq S & \Rightarrow & R^* ; S & \sqsubseteq & S & \text{left induction} \end{array}$$

□

A frequently used alternative approach is to assume completeness of the ordering $\sqsubseteq$ on each homset and distributivity of composition over arbitrary joins. This stronger setting is also known as S-algebras or standard Kleene algebras as introduced by Conway [20], or quantales [59]; we call the typed version *complete Kleene categories.* In these, the Kleene star can be defined via an infinite join:

$$R^* = \bigsqcup\{R^i | i \in \mathbb{N}\}$$

The Kleene star axioms from above then turn into theorems.

Definition 36. A *Kleene category with converse* is a Kleene category that is at the same time an OCC, and the involution law for Kleene star holds: $(R^*)^{\smile} = (R^{\smile})^*$. □

In Kleene categories with converse, difunctional closures always exist:

Lemma 37. For an arbitrary morphism $R : \mathcal{A} \leftrightarrow \mathcal{B}$ in a Kleene category with converse, the least difunctional morphism containing R, i.e., its *difunctional closure*, is the morphism $R^{\boxed{*}}$ with:

$$R^{\boxed{*}} = (R\mathbin{;}R^{\smile})\mathbin{;}R \qquad \square$$

Residuated Kleene categories are Kleene categories with residuals of composition; Kozen introduced the untyped variant of these as *residuated Kleene algebras* [51] and showed that they are equivalent to the *action algebras* of Pratt [69] that do not include the Kleene star as primitive operation.

4.3 Allegories

Definition 38. An *allegory* is a lower semilattice category with converse such that for all $Q : \mathcal{A} \leftrightarrow \mathcal{B}$, $R : \mathcal{B} \leftrightarrow \mathcal{C}$, and $S : \mathcal{A} \leftrightarrow \mathcal{C}$, the *modal rule* holds:

$$Q\mathbin{;}R \sqcap S \sqsubseteq (Q \sqcap S\mathbin{;}R^{\smile})\mathbin{;}R \ . \qquad \square$$

From the given modal rule above we may — using properties of conversion — obtain the dual modal rule

$$Q\mathbin{;}R \sqcap S \sqsubseteq Q\mathbin{;}(R \sqcap Q^{\smile}\mathbin{;}S) \ ,$$

which is used by Olivier and Serrato for their axiomatisation of Dedekind categories [65, 66] (see below) and there called “Dedekind formula” — however, Jacques Riguet had much earlier attached the name “Dedekind formula” to the following formula [70], which is equivalent to the modal rules:

$$Q\mathbin{;}R \sqcap S \sqsubseteq (Q \sqcap S\mathbin{;}R^{\smile})\mathbin{;}(R \sqcap Q^{\smile}\mathbin{;}S) \ .$$

The Dedekind formula (and any modal rule) implies that in allegories, all morphisms are co-difunctional: $R = \mathbb{I}\mathbin{;}R \sqcap R \sqsubseteq (\mathbb{I} \sqcap R\mathbin{;}R^{\smile})\mathbin{;}R \sqsubseteq R\mathbin{;}R^{\smile}\mathbin{;}R$.

Definition 39. In an allegory, for a morphism $R : \mathcal{A} \leftrightarrow \mathcal{A}$ we define the following properties:

- R is *antisymmetric* iff $R^{\smile} \sqcap R \sqsubseteq \mathbb{I}$,
- R is an *ordering* iff R is reflexive, transitive, and antisymmetric. □

In allegories, one can *define* domain and range operators

Definition 40. For every morphism $R : \mathcal{A} \leftrightarrow \mathcal{B}$ in an allegory, we define $\mathsf{dom}\, R : \mathcal{A} \leftrightarrow \mathcal{A}$ and $\mathsf{ran}\, R : \mathcal{B} \leftrightarrow \mathcal{B}$ as:

$$\mathsf{dom}\, R := \mathbb{I}_{\mathcal{A}} \sqcap R;R^{\smile} \qquad\qquad \mathsf{ran}\, R := \mathbb{I}_{\mathcal{B}} \sqcap R^{\smile};R \qquad \square$$

This definition of domain for allegories turns every allegory into an ordered category with domain in the sense of Def. 24.

We now just list the definitions for the most important allegories with additional structure:

- A *distributive allegory* is an allegory that is also an ISR category.
- *Division allegories* [30] are distributive allegories with residuals.
- *Dedekind categories* [65, 66] (locally complete distributive allegories in [30]) are complete distributive allegories. These are automatically division allegories and also Kleene categories.
- *Relation algebras* are Dedekind categories where all homsets are Boolean lattices.

4.4 Tabulations and Direct Products

We no prepare to introduce a relational characterisation of direct products as special case of what Freyd and Scedrov call *tabulation* [30, 2.14].

Definition 41. In a lower semilattice category with converse let a morphism $V : \mathcal{B} \leftrightarrow \mathcal{C}$ be given. The span $\mathcal{B} \xleftarrow{P} \mathcal{A} \xrightarrow{Q} \mathcal{C}$ of *mappings* P and Q is called a *tabulation of* R iff the following equations hold:

$$P^{\smile};Q = V \qquad\qquad P;P^{\smile} \sqcap Q;Q^{\smile} = \mathbb{I}\ . \qquad \square$$

$$\begin{tikzcd}
& \mathcal{A} \arrow[dl, "P"'] \arrow[dr, "Q"] & \\
\mathcal{B} \arrow[rr, "V"] & & \mathcal{C}
\end{tikzcd}$$

So only co-difunctional morphisms can be tabulated — in an allegory, this is no restriction.

The following equivalent characterisation is easily checked. Notice that, according to Def. 40, $\mathbb{I} \sqcap V{;}V^\smile = \mathsf{dom}\, V$; we use the expanded form to emphasise the duality with Proposition 45 below.

Proposition 42. In an allegory, the span $\mathcal{B} \xleftarrow{P} \mathcal{A} \xrightarrow{Q} \mathcal{C}$ is a tabulation of $V : \mathcal{B} \leftrightarrow \mathcal{C}$ if and only if the following equations hold:

$$P^\smile{;}Q = V \qquad \begin{array}{c} P^\smile{;}P = \mathbb{I} \sqcap V{;}V^\smile \\ Q^\smile{;}Q = \mathbb{I} \sqcap V^\smile{;}V \end{array} \qquad P{;}P^\smile \sqcap Q{;}Q^\smile = \mathbb{I}_{\mathcal{A}} \ . \qquad \Box$$

If $\mathbf{A}$ is an allegory, tabulations in $\mathbf{A}$ are unique up to isomorphism, and if $V = R{;}S^\smile$ for two mappings R and S, then a tabulation for V is a pullback in $\mathsf{Map}\,\mathbf{A}$. If direct products and subobjects are available, then a tabulation can be constructed for each morphism of the allegory $\mathbf{A}$.

In categories with terminal objects, direct products can be defined as pullbacks of a co-span with a terminal object as target of both morphisms. Corresponding to terminal objects, we can have *unit* objects in OCCs (the definition of Freyd and Scedorv can be used [30, 2.15]), and the resulting morphisms will be top morphisms, including the span morphism $\mathbb{T}_{\mathcal{A},\mathcal{B}} = \mathbb{T}_{\mathcal{A},\mathbb{1}}{;}\mathbb{T}_{\mathbb{1},\mathcal{B}}$ [30, 2.152].

A *direct product* of $\mathcal{A}$ and $\mathcal{B}$ is therefore defined as a tabulation of this greatest morphism $\mathbb{T}_{\mathcal{A},\mathcal{B}}$. We can also give an equivalent, direct definition in the style of the "Munich approach" of Schmidt and coworkers [74, 17, 8, 89, 90, 78, 10]. As a result of being a special case of tabulations, our definition differs from that usually given in the Munich approach essentially by not demanding surjectivity of the projections; taking the allegory of sets and concrete relations as motivation, this is necessary to cover also the case of empty products where only *one* of the two sets $\mathcal{A}$ and $\mathcal{B}$ is empty.

Definition 43. In an allegory, a *direct product* for two objects $\mathcal{A}$ and $\mathcal{B}$ for which $\mathbb{T}_{\mathcal{A},\mathcal{B}}$ exists is a triple $(\mathcal{P}, \pi, \rho)$ consisting of an object $\mathcal{P}$ and two *projections*, i.e., relations $\pi : \mathcal{P} \leftrightarrow \mathcal{A}$ and $\rho : \mathcal{P} \leftrightarrow \mathcal{B}$ for which the following conditions hold:

$$\pi^\smile{;}\rho = \mathbb{T}_{\mathcal{A},\mathcal{B}} \qquad \begin{array}{c} \pi^\smile{;}\pi = \mathsf{dom}\,(\mathbb{T}_{\mathcal{A},\mathcal{B}}) \\ \rho^\smile{;}\rho = \mathsf{ran}\,(\mathbb{T}_{\mathcal{A},\mathcal{B}}) \end{array} \qquad \pi{;}\pi^\smile \sqcap \rho{;}\rho^\smile = \mathbb{I}_{\mathcal{P}} \ . \qquad \Box$$

Because of the uniqueness of tabulations, this definition is a monomorphic characterisation of direct products.

It is well-known that the self-duality of categories of relations implies that categorical sums are at the same time categorical products, so these direct products are not categorical products in the category underlying the allegory $\mathbf{A}$ under consideration, but only in $\mathsf{Map}\,\mathbf{A}$.

The fact that the direct definition of direct products ends up being not perfectly dual to direct sums is due to the fact that $\mathbb{T}$ is not perfectly dual to $\bot\!\!\!\bot$, either: in most interesting models, zero laws for $\mathbb{T}$ do not hold.

4.5 Co-Tabulations and Direct Sums

While a tabulation can be seen as a certain kind of decomposition of a co-difunctional morphism in an allegory into a span, the dual of a tabulation is then a certain kind of decomposition of a difunctional morphism in an upper semilattice category into a co-span. A special case of this decomposition has been discussed by Schmidt and Ströhlein, for example in [78, 4.4.10]. Although the formal material here is dual to that in the previous section, we still spell it out in full detail for reference and better intuition.

Definition 44. In an upper semilattice category with converse let a morphism $W : \mathcal{B} \leftrightarrow \mathcal{C}$ be given. The co-span $\mathcal{B}\xrightarrow{R}\mathcal{D}\xleftarrow{S}\mathcal{C}$ of *mappings* R and S is called a *co-tabulation of* W iff iff the following equations hold:

$$R{;}S^{\smile} = W \qquad\qquad R^{\smile}{;}R \sqcup S^{\smile}{;}S = \mathbb{I}_{\mathcal{D}}\ . \qquad \square$$

$$\begin{tikzcd}
\mathcal{B} \arrow[rr, "W"] \arrow[dr, "R"] & & \mathcal{C} \arrow[dl, "S"'] \\
& \mathcal{D} &
\end{tikzcd}$$

Because of Lemma 17, only difunctional morphisms can be co-tabulated — in most contexts, this is a heavy restriction.

We explicitly give the equivalent characterisation that is dual to the one in Proposition 42:

Proposition 45. In an upper semilattice category with converse, the span $\mathcal{B}\xrightarrow{R}\mathcal{D}\xleftarrow{S}\mathcal{C}$ is a co-tabulation of $W : \mathcal{B} \leftrightarrow \mathcal{C}$ iff the following equations hold:

$$R{;}S^{\smile} = W \qquad \begin{array}{c} R{;}R^{\smile} = \mathbb{I} \sqcup W{;}W^{\smile} \\ S{;}S^{\smile} = \mathbb{I} \sqcup W^{\smile}{;}W \end{array} \qquad R^{\smile}{;}R \sqcup S^{\smile}{;}S = \mathbb{I}_{\mathcal{D}}\ . \qquad \square$$

In each upper semilattice category with converse, co-tabulations are unique up to isomorphism.

If $\mathbf{C}$ is a Kleene category with converse, and if $W = (P^{\smile} \mathbin{;} Q)^{\boxed{*}}$ for two mappings P and Q in $\mathbf{C}$, then a co-tabulation for W is a pushout for P and Q in $\mathsf{Map}\,\mathbf{C}$. If direct sums and quotients are available, then a co-tabulation can be constructed for each difunctional morphism.

A *gluing* as introduced by the author in [42] is a co-tabulation of the difunctional closure of an arbitrary morphism, essentially using the characterisation of Proposition 45 instantiated accordingly. Kawahara is the first to have characterised pushouts relation-algebraically in this way [47].

Each zero morphism $\perp\!\!\!\perp_{\mathcal{A},\mathcal{B}}$ is difunctional, and therefore can have a co-tabulation; this is how direct sums can be defined in ISR categories with converse. From the direct definition resulting from this we see that the injections are total and therefore subobject injections; the terms resulting from $W \mathbin{;} W^{\smile}$ and $W^{\smile} \mathbin{;} W$ disappear thanks to the zero laws.

Definition 46. In an ISR category with converse, a *direct sum* for two objects $\mathcal{A}$ and $\mathcal{B}$ is a triple $(\mathcal{S}, \iota, \kappa)$ consisting of an object $\mathcal{S}$ and two *injections*, i.e., morphisms $\iota : \mathcal{A} \leftrightarrow \mathcal{S}$ and $\kappa : \mathcal{B} \leftrightarrow \mathcal{S}$ for which the following conditions hold:

$$\iota \mathbin{;} \kappa^{\smile} = \perp\!\!\!\perp_{\mathcal{A},\mathcal{B}} \qquad \begin{array}{c} \iota \mathbin{;} \iota^{\smile} = \mathbb{I}_{\mathcal{A}} \\ \kappa \mathbin{;} \kappa^{\smile} = \mathbb{I}_{\mathcal{B}} \end{array} \qquad \iota^{\smile} \mathbin{;} \iota \sqcup \kappa^{\smile} \mathbin{;} \kappa = \mathbb{I}_{\mathcal{S}} \ . \qquad \square$$

Direct sums in an ISR category with converse $\mathbf{C}$ are at the same time categorical coproducts in $\mathbf{C}$ and in the mapping category $\mathsf{Map}\,\mathbf{C}$.

4.6 Some Important Models

The algebra of regular languages and other set-based Kleene algebras are in fact standard Kleene algebras (or quantales) since set-theoretic union of arbitrary sets of sets is available, and composition, when defined via an element-wise monoid operation, naturally distributes over arbitrary unions. Transferred into our typed setting, all these are therefore complete Kleene categories. Most of these models extend to Kleene categories with converse through the addition of an element-wise definition of converse.

Simulations (Def. 25) between algebras over fixed, not necessarily unary signatures form allegories [42]. A noteworthy special case of this is the allegory of relational group homomorphisms, where all morphisms are (strictly) difunctional[3]. Logical theories give rise to allegories of *derived predicates* [30, App. B].

In any topos one finds a Dedekind category [47]. A special case of this are relational graph structure homomorphisms, presented in [42] directly without using

[3] This was pointed out to me by Yasuo Kawahara (Kyushu Univ.), personal communication, 2001

topos theory. In such a Dedekind category of *relational* graph structure homomorphisms, conventional graph structure homomorphisms are recovered as the *mappings* in these Dedekind categories, that is, as total and univalent relational graph structure homomorphisms.

Fuzzy relations give rise to Dedekind categories, too [48], although fuzzy relations are more appropriately axiomatised in the Goguen categories proposed by Winter, where there are two additional operations for handling crispness [87, 88].

5 Application to Graph Structure Transformation

The so-called "algebraic approach to graph transformation" is really a collection of approaches that essentially rely on category-theoretic abstractions. Historically, Ehrig, Pfender, and Schneider developed the double-pushout approach as a way to generalise Chomsky grammars from strings to graphs, using pushouts as "gluing construction" to play the rôle of concatenation on strings [27], see also [21].

In the double-pushout approach, a rewriting rule is a *span* $\mathcal{L} \xleftarrow{\Phi_\mathrm{L}} \mathcal{G} \xrightarrow{\Phi_\mathrm{R}} \mathcal{R}$ of morphisms starting from the *gluing object* $\mathcal{G}$. A *redex* for such a rule is a morphism $X_\mathrm{L} : \mathcal{L} \leftrightarrow \mathcal{A}$ from the rule's left-hand side $\mathcal{L}$ into some *application graph* $\mathcal{A}$. Application of the rule has to establish a *double-pushout diagram* of the following shape:

$$\begin{tikzcd}
\mathcal{L} \arrow[d, "X_\mathrm{L}"'] & \mathcal{G} \arrow[l, "\Phi_\mathrm{L}"'] \arrow[r, "\Phi_\mathrm{R}"] \arrow[d, "H"'] & \mathcal{R} \arrow[d, "X_\mathrm{R}"'] \\
\mathcal{A} & \mathcal{H} \arrow[l, "\Psi_\mathrm{L}"'] \arrow[r, "\Psi_\mathrm{R}"] & \mathcal{B}
\end{tikzcd}$$

Note that for the left-hand side pushout, the "wrong" arrows are given, so the completion to a pushout square, called *pushout complement*, is not a universally characterised categorical construction.

Replication is notoriously impossible with pushouts, but is inherent in the dual concept of pullbacks. Accordingly, there is a less well-known variant of the categoric approach that uses pullbacks for graph transformation, put forward by Bauderon and Jacquet [34, 5, 4, 6].

However, that approach never really gained popularity, probably mostly because its rules as such appear to be quite unintuitive and usually have to be regarded as "encodings" of the rules of other approaches. With that encoding attitude, however, the pullback approach is able to cover most popular approaches to graph rewriting, including the double-pushout approach.

In this section, we present a short summary of the central results of [42], with an improved presentation thanks to the now separate concept of co-tabulations and the finer-grained theory organisation.

Using the fact that Σ-simulations between Σ-algebras form allegories, and in the case of a unary signature Σ even Dedekind categories, we consider graph structures as unary Σ-algebras, and then perform *all* reasoning in the abstract theories. In comparison with the topos-based approach of Kawahara [47], we have the advantage that we have modular set of theories available, and can easily identify sufficient weaker theories for results that do not rely on the full language, and therewith make them available to a wider application area. For example, our characterisation of pullback complements (Sect. 5.1) takes place entirely within the allegory setting, and therefore can be used in arbitrary Σ-algebra allegories, even in the presence of constants and binary operations.

Dually, pushout rewriting can be formalised in idempotent semiring categories with converse, and the single-pushout approach translates into an appealing generalisation of co-tabulations (Sect. 5.2).

Finally, we show how the unifying setting of Dedekind categories permits a "unified" rewriting concept that employs the power of pullback rewriting on "subgraph variables", and the precise control of pushout rewriting on "fixed parts", to yield an intuitively accessible relation-algebraic approach to graph transformation with proper subgraph variables.

5.1 Pullback Rewriting in Allegories

Just like in the double-pushout approach, the left-hand-side square of a rewriting step in the double-pullback approach also poses the problem that the "wrong" arrows are to be constructed.

Assuming two mappings $\mathcal{D} \xrightarrow{P} \mathcal{B} \xrightarrow{R} \mathcal{A}$ to be given, we therefore need a *pullback complement* $\mathcal{D} \xrightarrow{Q} \mathcal{C} \xrightarrow{S} \mathcal{A}$, that is, an object $\mathcal{C}$ and two mappings $Q : \mathcal{D} \leftrightarrow \mathcal{C}$ and $S : \mathcal{C} \leftrightarrow \mathcal{A}$ such that the resulting square is a pullback for R and S.

$$\begin{tikzcd}
\mathcal{B} \arrow[r, "R"] & \mathcal{A} \\
\mathcal{D} \arrow[u, "P"] \arrow[r, dotted, "Q"] & \mathcal{C} \arrow[u, dotted, "S"']
\end{tikzcd}$$

A necessary and sufficient condition for the existence of pullback complements in the category of concrete graphs has been given by Bauderon and Jacquet [4, 34, 6]. This is an extremely complex condition formulated on the level of edges and nodes, postulating enumerations of pre-images satisfying certain compatibility conditions, and using quite intricate notation.

In the relational approach, we can replace this with an abstract, component-free condition that is necessary and sufficient for the existence of pullback complements in the sub-category of mappings in arbitrary allegories. This condition is extremely simple, and thus offers valuable insight into the essence of pullback complements (we continue to use the name "coherent" proposed by Bauderon and Jacquet):

Definition 47. Two mappings $\mathcal{D}\xrightarrow{P}\mathcal{B}\xrightarrow{R}\mathcal{A}$ are called *coherent* iff there exists an equivalence $\Theta : \mathcal{D} \leftrightarrow \mathcal{D}$ such that the following conditions hold:

1. $P{;}P^{\smile} \sqcap \Theta \sqsubseteq \mathbb{I}$,
2. $\Theta{;}P = P{;}R{;}R^{\smile}$.

We also say that P and R are *coherent via* Θ. □

Note that (2) implies $\Theta{;}P{;}P^{\smile} = P{;}R{;}R^{\smile}{;}P^{\smile}$. Since with the right-hand side, also the left-hand side is an equivalence, and since the composition of two equivalences is always contained in their equivalence join, but not in any smaller equivalence, this means that $P{;}R{;}R^{\smile}{;}P^{\smile}$ is the equivalence join of Θ and $P{;}P^{\smile}$. Together with (1) it then follows that Θ is a complement of $P{;}P^{\smile}$ in the lattice of all equivalences contained in $P{;}R{;}R^{\smile}{;}P^{\smile}$. Since this lattice is, in general, not distributive, this complement need not be uniquely determined even if it exists. And not all complements in addition fulfil (2).

Proposition 48. If $\mathcal{B}\xleftarrow{P}\mathcal{D}\xrightarrow{Q}\mathcal{C}$ is a pullback for $\mathcal{B}\xrightarrow{R}\mathcal{A}\xleftarrow{S}\mathcal{C}$, then P and R are coherent via $\Theta := Q{;}Q^{\smile}$.

Proof. With commutativity and alternative commutativity we obtain (2):

$$P{;}R{;}R^{\smile} = Q{;}S{;}R^{\smile} = Q{;}Q^{\smile}{;}P = \Theta{;}P$$

The coherence condition (1) is part of the tabulation properties. □

Theorem 49. If there exists an equivalence $\Theta : \mathcal{D} \leftrightarrow \mathcal{D}$ such that P and R are coherent via Θ, then a pullback complement $\mathcal{D}\xrightarrow{Q}\mathcal{C}\xrightarrow{S}\mathcal{A}$ for $\mathcal{D}\xrightarrow{P}\mathcal{B}\xrightarrow{R}\mathcal{A}$ is obtained as follows:

- Let $\mathcal{C}$ be a quotient of $\mathcal{D}$ for Θ, with projection $Q : \mathcal{D} \leftrightarrow \mathcal{C}$.
- Define $S : \mathcal{C} \leftrightarrow \mathcal{A}$ as $S := Q^{\smile}{;}P{;}R$. □

In comparison with Bauderon and Jacquet's, ours is a much simpler formulation of the pullback complement condition. From our formulation it also becomes clearer that the reason for the absence of efficient implementations is the intractability of the general problem of searching for an (appropriate) equivalence complement.

5.2 Pushout Rewriting in ISR Categories with Converse

The *pushout complement* necessary for the double-pushout approach exists iff the so-called *gluing condition* holds — for items from left-hand side outside the image of the interface, the redex must not perform any identification or mapping to nodes incident with “dangling edges”.

This gluing condition is usually stated using the concrete language of graphs with nodes and edges; Kawahara gave a component-free formulation [47] employs an embedding of relational calculus in topos theory, which essentially corresponds to a Dedekind category setting. For the additional condition of *conflict-freeness* that is important in the single-pushout approach, a component-free formulation was given in [42]. Closer inspection reveals that these conditions do use domain and range, and pseudo-complements on domain and range elements, but otherwise no meet. Therefore, we can set the environment accordingly:

Definition 50. In an ISR category with converse, domain, and pseudo-complements on domain elements let two morphisms $\Phi : \mathcal{G} \leftrightarrow \mathcal{L}$ and $X : \mathcal{L} \leftrightarrow \mathcal{A}$ be given. We define:

- The *identification condition* holds iff $X;X^\smile \sqsubseteq \mathbb{I} \sqcup (\mathsf{ran}\,\Phi);X;X^\smile;\mathsf{ran}\,\Phi$.
- The *dangling condition* holds iff $\mathsf{ran}\,X \sqcup (\mathsf{ran}\,X \rightarrow \mathsf{ran}\,(\Phi;X)) = \mathbb{I}$.
- We call X *conflict-free for* Φ iff $\mathsf{ran}\,(\Phi;X;X^\smile) \sqsubseteq \mathsf{ran}\,\Phi$. □

When the identification and dangling conditions both hold, then a subobject for $\mathsf{ran}\,X \rightarrow \mathsf{ran}\,(\Phi;X)$ produces a pushout complement.

Motivated by the shortcomings of the double-pushout approach that was based on *total* graph homomorphisms, the end of the 1980s saw the emergence of the *single-pushout approach* based on categories of *partial* graph homomorphisms; the most general approach is by Löwe [55, 56, 54, 28]. Construction of pushouts of partial morphisms is technically rather involved, and the properties are not obvious.

In [42], we arrived at a characterisation that is a relatively simple generalisation of co-tabulations:

Definition 51. Let two difunctional morphisms $Q, U : \mathcal{R} \leftrightarrow \mathcal{A}$ with $Q \sqsubseteq U$ be given.

A co-span $\mathcal{R} \xrightarrow{X} \mathcal{B} \xleftarrow{\Psi} \mathcal{A}$ is called a *restricted gluing for* Q *in* U if:

$$\begin{aligned} X;\Psi^\smile &= Q \\ X;X^\smile &= (\mathsf{dom}\,U \rightarrow \mathsf{dom}\,Q) \sqcup Q;Q^\smile \\ \Psi;\Psi^\smile &= (\mathsf{ran}\,U \rightarrow \mathsf{ran}\,Q) \sqcup Q^\smile;Q \\ X^\smile;X \sqcup \Psi^\smile;\Psi &= \mathbb{I} \end{aligned}$$

□

If one starts from a span $\mathcal{A} \xleftarrow{Y} \mathcal{L} \xrightarrow{\Phi} \mathcal{R}$ of partial functions, i.e., univalent morphisms, and defines $U := \Phi^{\smile};\mathbb{T};Y$ and $Q := \mathsf{satSyq}(\Phi, Y)$ using a function satSyq that can be defined in complete Kleene categories with domain and converse, then each restricted gluing for Q in U is a pushout of partial functions.

This pushout can be easily constructed using two subobjects induced by $(\mathsf{dom}\, U \rightarrow \mathsf{dom}\, Q)$ and $(\mathsf{ran}\, U \rightarrow \mathsf{ran}\, Q)$ and a standard co-tabulation.

5.3 Rewriting with Graph Variables in Dedekind Categories

In [42, 43], we presented a formalism that allows the abstract specification of graph transformation with "graph variables" and replication of their images. This approach is superficially be modelled at the double-pushout approach, but works in the setting of relational graph morphisms; the *pullout* construction used there can be understood as amalgamated from a pushout used for "fixed" parts of the rule, and a pullback used for the "variable" or "parameter" parts.

With a few *interface preservation* conditions that we leave out here, a co-span $G_1 \xrightarrow{X} G_3 \xleftarrow{\Psi} G_2$ is defined to be a *glued tabulation for* V *along* U *on* c_1 *and* c_2 (where U is difunctional, and c_1 and c_2 are partial identities comprising the respective interface parts and contexts) iff the following conditions hold:

$$\begin{array}{rcccc}
X;\Psi^{\smile} & = & U & \sqcup & V \\
X;X^{\smile} & = & c_1;(\mathbb{I} \sqcup U;U^{\smile});c_1 & \sqcup & (\mathbb{I} \sqcap V;V^{\smile}) \\
\Psi;\Psi^{\smile} & = & c_2;(\mathbb{I} \sqcup U^{\smile};U);c_2 & \sqcup & (\mathbb{I} \sqcap V^{\smile};V) \\
\mathbb{I} & = & (X^{\smile};c_1;X \sqcup \Psi^{\smile};c_2;\Psi) & \sqcup & (X^{\smile};v_1;X \sqcap \Psi^{\smile};v_2;\Psi)
\end{array}$$

Obviously, these pullout conditions are an amalgamation of the co-tabulation and tabulation conditions, and as such can be formulated in distributive allegories with pseudo-complements on sub-identities. However, obtaining U from the original span requires difunctional closure. Since completeness also guarantees the existence of pseudo-complements, Dedekind categories are the natural setting for rewriting using pullouts.

The ease with which this amalgamation of the pushout and pullback approaches is possible is the essential advantage of the relational approach over purely category theoretic approaches.

6 Formalisation and Mechanisation

Reasoning in the "relational flavour" theories is relatively high-level and frequently calculational; it therefore tends to be fairly rigorous. Nevertheless, because of the large number of different axiomatisations considered even within investigations of relatively narrow scope, the large number of laws valid in many theories, and because frequently "trivial" properties of familiar theories "unexpectedly" fail to hold under reduced combinations of axioms, confusion can arise. Therefore, mechanised proof support becomes very desirable.

Based on our experience for example with formalising graph transformation as described in Sect. 5, we have identified the following requirements for user-friendly proof support for reasoning in "theories with relational flavour":

- Many different theories, including all those described in Sect. 3 need to be supported, and adding new theories should be easy and systematic.
- Since untyped theories can always be considered as special cases of typed theories, and most computer science applications are naturally typed, support for typed theories is essential.
- The calculational style is an essential tool for both proof development and proof presentation in the theories we are interested in; support for calculational proofs is therefore necessary for acceptance.
- Reasoning *within* structures, in the style of "let an arbitrary but fixed OCC be given", should be supported with minimal overhead.
- Reasoning *about* structures is essential for applications: Concrete models, for example Kleene categories of regular languages, or allegories of relations, as well as model constructions, such as matrix Kleene algebras and allegories of Σ-algebras, need to be accessible and verifiable.
- Reasoning *between* structures involves several structures over identical or overlapping signatures; these overlaps should be handled gracefully.

Quite a few projects in the recent past have striven to provide computer-aided proof assistance for reasoning in abstract relation algebras or Kleene algebras, each with its particular motivation and priorities.

- The relation-algebraic formula manipulation system and proof checker RALF [11, 33, 40] was designed as a special-purpose proof assistant for abstract (heterogeneous) relation algebras with the goal of supporting proofs in the calculational style. RALF has a graphical user interface presenting goal formulae in their tree structure, a feature that allows easy interactive selection of the subexpressions to be transformed by proof steps. RALF is based on a fixed axiomatisation, and only supports reasoning *within* a *single* relation algebra.
- Math∫pad is a flexible quasi-WYSIWYG syntax-directed editing environment for mathematical documents that has been designed to support calculational proof presentation. It has been connected with the theorem prover PVS to enable checking of relation-algebraic proofs contained in Math∫pad documents

[84], but is normally used as powerful editing support for a rigorous style of calculational reasoning without formal proof checking.

- "PCP: Point and Click Proofs" is a proof assistant based on a small JavaScript rewriting engine that allows users to interactively construct proofs of properties in a wide range of mathematical structures, characterised by equational and quasi-equational theories [37, 36]. Currently, (homogeneous) relation algebra and Kleene algebra are already supported by the system, which is extensible and still under development. It appears not to be geared to the addition of a type system, and is also limited to reasoning *within single* structures.
- A previous formalisation of heterogeneous relation algebras in Isabelle, RALL [64], uses the Isabelle/HOL type system to support reasoning *within* abstract *heterogeneous* relation algebras with minimal effort, but at the cost of limiting itself to reasoning *within* a *single* relation algebra, as well. RALL's approach of *atomising* relation-algebraic formulae into predicate logic does not carry over to weaker structures like allegories or Kleene algebras.
- Struth realised a formalisation in Isabelle-1999 of untyped Kleene algebras, and used this to fully formalise Church-Rosser proofs in Kleene algebras [82]. He formalised Kleene algebras via a hierarchy of axiomatic type classes [86], which to some extent do support reasoning *between* several structures; but impose severe limitations to reasoning *about* structures.
- KAT-ML is an interactive calculational theorem prover for reasoning *within* a *single* (untyped) Kleene algebra with tests [2]. Its (text-based) user interface includes a focusing feature similar to the selection feature of the graphical user interface of RALF.

Obviously, each of these systems was designed with a different set of requirements in mind, and none of them appears appropriate as a basis for a system that might satisfy all of the requirements listed above.

In [44, 31] we therefore embarked on a new approach to mechanised support of reasoning in "theories with relational flavour" using recent developments in the theorem prover Isabelle [60] that support our long-term objectives relatively well:

- For reasoning *within* abstract algebraic structures in essentially the same way as it is done in pencil-and-paper mathematics, the concept of *locales* has been introduced into Isabelle [46, 3]. If such locales are based on records, this also allows reasoning *about* and *between* algebraic structures.
- While internally Isabelle still is a tactical theorem prover, the addition of the interpreted "Isar" language for "Intelligible semi-automated reasoning" allows proofs to be structured in the same way as in traditional mathematical

proof presentation [61, 85]. One additional aspect of Isar is its support for calculational reasoning [7]; this has been designed to also support user-defined transitive relations, such as the inclusion ordering of relations.

These features together allow us to implement a collection of theories reflecting the organisation of Sect. 3 and already covering most of the material presented there, together with useful derived properties in each theory. The granularity of the theories in this collection is even finer than that presented in Sect. 3, and the whole design is geared towards calculational reasoning *in and about* ordered categories and all the different kinds of allegories and Kleene categories up to heterogeneous relation algebras.

A different aspect of mechanised support is the use of languages with "relational flavour" as *programming paradigms*. The field here seems to be populated almost exclusively by languages that allow programming of concrete relations using the language of relation algebras:

- The most prominent system in this class is RelView [12] which provides a simple imperative programming language with finite concrete relations (implemented efficiently using BDDs) as its only datatype, and a graphical user interface to produce argument relations and access result relations.
- The languages Drusilla [18, 19] and Libra [25, 26] allow relational programming with standard primitive datatypes and potentially infinite relations considered as *generators*.
- The relational circuit design language Ruby [38, 39] uses the language of allegories with direct products for specification and refinement of VLSI circuit designs; there are interpreters that allow evaluation of (certain) Ruby expressions.
- McPhee proposes to represent concrete relations via their tabulations, and evaluate relational operations as operations on tabulations [57]. However, no report on further progress appears to be available.

A completely different approach is the framework RATH [41] for relation-algebraic programming which has started to fill the gap between on the one hand the system RelView and on the other hand high-level programming languages. RATH provides mechanisms for

- defining arbitrary algebras with the signatures of the most of the theories listed in Sect. 3,
- performing algebra-level operations to generate product algebras, matrix algebras, sub-algebras, etc.,

– performing calculations *within* those algebras (accessing their operations via the respective signatures), effectively turning each theory into a programming language embedded in the purely functional programming language Haskell [68], and
– testing the validity of the respective axioms in finite algebras.

RATH has been used for automatic generation of non-standard allegories and relation algebras [63], and for implementing the graph rewriting approach described in Sect. 5 [42].

7 Conclusion and Outlook

One may discern two main flavours of sub-theories of the theory of relation algebras that are by themselves important tools in computer science: On the one hand Kleene algebras, which represent a "control-flow view", and on the other hand allegories, which represent a "data-flow view".

Recent efforts to move investigations that previously were performed in the full theory of relation algebras into weaker, more specialised theories frequently started from an algebraic point of view, for example with the single-sorted idempotent semirings underlying Kleene algebras, or with allegories. In both cases, the most common axiomatisations do not include the inclusion ordering as primitive, but derive it either from join or from meet.

In this paper, we reorganised the presentation of the spectrum of theories with "relational flavour" around ordered categories, and introduced converse directly on top of those. With this approach, we strove to present a more unifying and symmetric view of these theories. Although real symmetry is probably not easy to achieve, given on the one hand join-distributivity of composition and on the other hand the strong coupling between composition, meet, and converse in the modal rules, we feel that *not* assuming co-difunctionality in OCCs still brought a significant advance in visible duality.

Putting some of the new distinctions to use in revisiting the relation-algebraic formalisation of algebraic approaches to graph rewriting served as an example to show how two sub-approaches naturally reside in Kleene categories with converse and in allegories, and how using the joint power of Dedekind categories allows to amalgamate these approaches.

We believe that in the near future, more combined theories such as "Kleene algebras with domain" and "Kleene algebras with relations" [23] will be investigated and used, and that elegant and usable formalisation and mechanisation of the resulting ever-growing "theory-web" will become a higher priority, and will remain, for quite some time, a significant challenge.

Acknowledgements

I am grateful to Millie Rhoss de Guzman, Hitoshi Furusawa, Yasuo Kawahara, Gunther Schmidt, and Michael Winter for many related discussions; and I would like to thank Ridha Khedri and the anonymous reviewers for their valuable comments.

References

1. C. Aarts, R. C. Backhouse, P. Hoogendijk, E. Voermans, J. van der Woude. *A Relational Theory of Datatypes.* Working document, 1992. 387 pp., available at http://www.cs.nott.ac.uk/~rcb/MPC/book.ps.gz.
2. K. Aboul-Hosn, D. Kozen. *KAT-ML: An Interactive Theorem Prover for Kleene Algebra with Tests.* In B. Konev, R. Schmidt, eds., Proc. 4th Intl. Workshop on the Implementation of Logics (WIL'03), pp. 2–12. University of Manchester, 2003.
3. C. Ballarin. *Locales and Locale Expressions in Isabelle/Isar.* In S. Berardi et al., eds., Types for Proofs and Programs, International Workshop TYPES 2003, Torino, Italy, LNCS. Springer, 2004. (in press).
4. M. Bauderon, H. Jacquet. *Categorical Product as a Generic Graph Rewriting Mechanism.* Technical Report 1166–97, LaBRI, University of Bordeaux, 1996. see also [6].
5. M. Bauderon. *A Uniform Approach to Graph Rewriting: The Pullback Approach.* In M. Nagl, ed., Graph Theoretic Concepts in Computer Science, WG '96, LNCS **1017**, pp. 101–115. Springer, 1997.
6. M. Bauderon, H. Jacquet. *Pullback as a Generic Graph Rewriting Mechanism.* Applied Categorical Structures **9**(1) 65–82, 2001.
7. G. Bauer, M. Wenzel. *Calculational reasoning revisited, an Isabelle/Isar experience.* In R. J. Boulton, P. B. Jackson, eds., Theorem Proving in Higher-Order Logics: TPHOLs 2001, LNCS **2152**, pp. 75–90. Springer, 2001.
8. R. Berghammer, H. Zierer. *Relational Algebraic Semantics of Deterministic and Nondeterministic Programs.* Theoretical Computer Science **43** 123–147, 1986.
9. R. Berghammer, G. Schmidt, H. Zierer. *Symmetric Quotients and Domain Constructions.* Inform. Process. Lett. **33**(3) 163–168, 1989.
10. R. Berghammer, A. M. Haeberer, G. Schmidt, P. A. S. Veloso. *Comparing Two Different Approaches to Products in Abstract Relation Algebra.* In [62], pp. 167–176.
11. R. Berghammer, C. Hattensperger. *Computer-Aided Manipulation of Relational Expressions and Formulae using RALF.* In B. Buth, R. Berghammer, eds., Systems for Computer-Aided Specification, Development and Verification, Bericht Nr. 9416, pp. 62–78. Universität Kiel, 1994.
12. R. Berghammer, T. Hoffmann, B. Leoniuk, U. Milanese. *Prototyping and Programming with Relations.* ENTCS **44**(3) 3.1–3.24, 2003.
13. R. Berghammer, B. Möller, G. Struth, eds. *Relational and Kleene-Algebraic Methods in Computer Science: 7th International Seminar on Relational Methods in Computer Science and 2nd International Workshop on Applications of Kleene Algebra, Bad Malente, Germany, May 12–17, 2003, Revised Selected Papers*, LNCS **3051**. Springer, 2003.
14. R. S. Bird, O. de Moor. *Algebra of Programming*, International Series in Computer Science **100**. Prentice Hall, 1997.
15. C. Brink, W. Kahl, G. Schmidt, eds. *Relational Methods in Computer Science.* Advances in Computing Science. Springer, Wien, New York, 1997.
16. M. Broy, Gheorghe Ştefănescu. *The Algebra of Stream Processing Functions.* Theoretical Computer Science **258** 99–129, 2001.
17. R. Cardoso. *Untersuchung paralleler Programme mit relationenalgebraischen Methoden.* Diplomarbeit under supervision of Gunther Schmidt, TU München, 1982.

18. D. M. Cattrall. *The Design and Implementation of a Relational Programming System.* PhD thesis, Dept. of Computer Science, University of York, 1992.
19. D. M. Cattrall, C. Runciman. *A Relational Programming System with Inferred Representations.* In M. Bruynooghe, M. Wirsing, eds., PLILP '92, LNCS **631**, pp. 475–476, Leuven, Belgium, 1992. Springer. system presentation.
20. J. H. Conway. *Regular Algebra and Finite Machines.* Chapman and Hall, London, 1971.
21. A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, M. Löwe. *Algebraic Approaches to Graph Transformation, Part I: Basic Concepts and Double Pushout Approach.* In [73], Chapt. 3, pp. 163–245.
22. J. Desharnais, M. Frappier, A. Jaoua, W. MacCaull. *Relational Methods in Computer Science, Special Issue devoted to RelMiCS 5 in Valcartier, Québec, January 2000.* Information Sciences **139**(3–4) 165–307, 2001.
23. J. Desharnais. *Kleene ALgebras with Relations.* In [13]. (Invited Talk).
24. J. Desharnais, B. Möller, G. Struth. *Kleene Algebra with Domain.* Technical Report 2003-7, Universität Augsburg, Institut für Informatik, 2003.
25. B. Dwyer. *LIBRA: A Lazy Interpreter of Binary Relational Algebra.* Technical Report 95-10, Dept. of Computer Science, University of Adelaide, 1995. http://www.cs.adelaide.edu.au/ dwyer/.
26. B. Dwyer. *Relational Programming in Libra.* In A. Jaoua, P. Kempf, G. Schmidt, eds., Using Relational Methods in Computer Science, Technical Report Nr. 1998-03, pp. 35–58. Fakultät für Informatik, Universität der Bundeswehr München, 1998.
27. H. Ehrig, M. Pfender, H. J. Schneider. *Graph Grammars: An Algebraic Approach.* In: Proc. IEEE Conf. on Automata and Switching Theory, SWAT '73, pp. 167–180, 1973.
28. H. Ehrig, R. Heckel, M. Korff, M. Löwe, L. Ribeiro, A. Wagner, A. Corradini. *Algebraic Approaches to Graph Transformation, Part II: Single Pushout Approach and Comparison with Double Pushout Approach.* In [73], Chapt. 4, pp. 247–312.
29. Z. Ésik, L. Bernátsky. *Equational properties of Kleene algebras of relations with conversion.* Theoretical Computer Science **137** 237–251, 1995.
30. P. J. Freyd, A. Scedrov. *Categories, Allegories*, North-Holland Mathematical Library **39**. North-Holland, Amsterdam, 1990.
31. M. R. de Guzman. *Relation-Algebraic Proofs in Isabelle/Isar.* Master's thesis, McMaster University, Department of Computing and Software, 2004.
32. A. Haeberer, M. Frias. *Relational Methods in Computer Science, Special issue devoted to RelMiCS 2 in Paraty, July 1995.* Journal of the IGPL **6**(2), 1998.
33. C. Hattensperger, R. Berghammer, G. Schmidt. *RALF — A relation-algebraic formula manipulation system and proof checker. Notes to a system demonstration.* In [62], pp. 405–406.
34. H. Jacquet. *Une approche catégorique de la réécriture de sommets dans les graphes.* PhD thesis, Université Bordeaux 1, 1999.
35. A. Jaoua, G. Schmidt. *Relational Methods in Computer Science, Special Issue devoted to RelMiCS 3 in Hammamet, January 1997.* Information Sciences **119**(3–4) 131–314, 1999.
36. P. Jipsen. *Implementing quasi-equational logic on the web.* Talk given at the AMS Sectional Meeting, University of South Carolina, 2001. http://www.chapman.edu/ jipsen/PCP/usctalk.html.
37. P. Jipsen. *PCP: Point and Click Proofs.* Web-based system at URL: http://www.chapman.edu/ jipsen/PCP/PCPhome.html, 2003.
38. G. Jones, M. Sheeran. *Circuit Design in Ruby.* In J. Staunstrup, ed., Formal Methods in VLSI Design, pp. 13–70. North-Holland, 1990.
39. G. Jones, M. Sheeran. *Designing Arithmetic Circuits by Refinement in Ruby.* In R. S. Bird, C. C. Morgan, J. C. P. Woodcock, eds., MCP '92, LNCS **669**, pp. 208–232. Springer, 1992. see also http://web.comlab.ox.ac.uk/oucl/work/geraint.jones/ruby/.
40. W. Kahl, C. Hattensperger. *Second-Order Syntax in HOPS and in RALF.* In B. Buth, R. Berghammer, J. Peleska, eds., Tools for System Development and Verification, BISS Monographs **1**, pp. 140–164, Aachen, 1998. Shaker Verlag. ISBN: 3-8265-3806-4.
41. W. Kahl, G. Schmidt. *Exploring (Finite) Relation Algebras Using Tools Written in Haskell.* Technical Report 2000-02, Fakultät für Informatik, Universität der Bundeswehr München, 2000. see also the RATH page http://ist.unibw-muenchen.de/relmics/tools/RATH/.

42. W. Kahl. *A Relation-Algebraic Approach to Graph Structure Transformation*, 2001. Habil. Thesis, Fakultät für Informatik, Univ. der Bundeswehr München, Techn. Bericht 2002-03.
43. W. Kahl. *A Relation-Algebraic Approach to Graph Structure Transformation.* In [83], pp. 1–14. (Invited Talk).
44. W. Kahl. *Calculational Relation-Algebraic Proofs in Isabelle/Isar.* In [13], pp. 178–190.
45. W. Kahl, D. L. Parnas, G. Schmidt, eds. *Relational Methods in Software — Special Issue devoted to RelMiS 2001, April 7–8, 2001, Genova.* Electronic Notes in Theoretical Computer Science **44**(3), 2003.
46. F. Kammüller, M. Wenzel, L. C. Paulson. *Locales — A Sectioning Concept for Isabelle.* In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, L. Théry, eds., Theorem Proving in Higher-Order Logics, 12th International Conference, TPHOLs'99, LNCS **1690**, pp. 149–166. Springer, 1999.
47. Y. Kawahara. *Pushout-Complements and Basic Concepts of Grammars in Toposes.* Theoretical Computer Science **77** 267–289, 1990.
48. Y. Kawahara, H. Furusawa. *Crispness in Dedekind Categories.* Bull. Inform. Cybernet. **33**(1–2) 1–18, 2001.
49. R. Khedri. *Concurrence, bisimulations et équation d'interface: une approche relationelle.* PhD thesis, Départment d'Informatique, Faculté des Sciences et de Genie, Université Laval, Québec, 1998.
50. D. Kozen. *A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events.* Inform. and Comput. **110**(2) 366–390, 1991.
51. D. Kozen. *On Action Algebras.* In J. van Eijck, A. Visser, eds., Logic and Information Flow, pp. 78–88. MIT Press, 1994.
52. D. Kozen. *Kleene Algebra with Tests.* ACM Transactions on Programming Languages and Systems pp. 427–443, 1997.
53. D. Kozen. *Typed Kleene Algebra.* Technical Report 98-1669, Computer Science Department, Cornell University, 1998.
54. M. Löwe. *Algebraic Approach to Single-Pushout Graph Transformation.* In B. Courcelle, G. Rozenberg, eds., Selected Papers of the International Workshop on Computing by Graph Transformation, Bordeaux, France, March 21–23, 1991, pp. 181–224. Elsevier, 1993. Theoretical Computer Science **109** (1–2).
55. M. Löwe. *Algebraic Approach to Graph Transformation Based on Single Pushout Derivations.* Technical Report 90/05, TU Berlin, 1990.
56. M. Löwe, H. Ehrig. *Algebraic Approach to Graph Transformation Based on Single Pushout Derivations.* In R. H. Möhring, ed., Graph-Theoretic Concepts in Computer Science, WG '90, LNCS **484**, pp. 338–353. Springer, 1991.
57. R. McPhee. *Towards a Relational Programming Language.* Qualifying dissertation for transfer to d.phil., Oxford University Computing Laboratory, 1995.
58. B. Möller. *Towards Pointer Algebra.* Science of Computer Programming **21** 57–90, 1993.
59. C. Mulvey. *&.* Rend. Circ. Mat. Palermo **12** 99–104, 1986.
60. T. Nipkow, L. C. Paulson, M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, LNCS **2283**. Springer, 2002.
61. T. Nipkow. *Structured Proofs in Isar/HOL.* In H. Geuvers, F. Wiedijk, eds., Types for Proofs and Programs, International Workshop TYPES 2002, LNCS **2646**, pp. 259–278. Springer, 2003.
62. M. Nivat, C. Rattray, T. Rus, G. Scollo, eds. *Proc. 3^{rd} Internat. Conf. Algebraic Methodology and Software Technology, Enschede, June 21–25*, Workshops in Computing. Springer, 1994.
63. E. Offermann. *Konstruktion Relationaler Kategorien.* PhD thesis, Fakultät für Informatik, Universität der Bundeswehr München, 2003.
64. D. von Oheimb, T. F. Gritzner. *RALL: Machine-supported Proofs for Relation Algebra.* In W. McCune, ed., Conference on Automated Deduction – CADE-14, LNCS **1249**, pp. 380–394. Springer, 1997.
65. J.-P. Olivier, D. Serrato. *Catégories de Dedekind. Morphismes dans les catégories de Schröder.* C. R. Acad. Sci. Paris Ser. A-B **290** 939–941, 1980.

66. J.-P. Olivier, D. Serrato. *Squares and Rectangles in Relation Categories – Three Cases: Semilattice, Distributive Lattice and Boolean Non-unitary.* Fuzzy Sets and Systems **72** 167–178, 1995.
67. E. Orlowska, A. Szalas, eds. *Relational Methods for Computer Science Applications*, Studies in Fuzziness and Soft Computing **65**, Heidelberg, 2001. Springer-Physica Verlag.
68. S. Peyton Jones et al. *The Revised Haskell 98 Report.* Cambridge Univ. Press, 2003. Also on http://haskell.org/.
69. V. Pratt. *Action Logic and Pure Induction.* In J. van Eijck, ed., JELIA 1990: Proc. European Workshop on Logics in Artificial Intelligence, LNCS **478**, pp. 97–120. Springer, 1991.
70. J. Riguet. *Relations Binaires, Fermetures, Correspondances de Galois.* Bull. Soc. Math. France **76** 114–155, 1948.
71. J. Riguet. *Quelques propriétés des relations difonctionnelles.* C. R. Acad. Sci. Paris Ser. A-B **230** 1999–2000, 1950.
72. W.-P. de Roever, K. Engelhardt. *Data Refinement: Model-Oriented Proof Methods and their Comparison*, Cambridge Tracts Theoret. Comput. Sci. **47**. Cambridge Univ. Press, 1998.
73. G. Rozenberg, ed. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 1: Foundations.* World Scientific, Singapore, 1997.
74. G. Schmidt. *Programme als partielle Graphen.* Habil. Thesis, Fachbereich Mathematik der Technischen Univ. München, Bericht 7813, 1977. English as [75, 76].
75. G. Schmidt. *Programs as Partial Graphs I: Flow Equivalence and Correctness.* Theoretical Computer Science **15**(1) 1–25, 1981.
76. G. Schmidt. *Programs as Partial Graphs II: Recursion.* Theoretical Computer Science **15**(2) 159–179, 1981.
77. G. Schmidt, T. Ströhlein. *Relationen und Graphen.* Mathematik für Informatiker. Springer, Berlin, 1989. English as [78].
78. G. Schmidt, T. Ströhlein. *Relations and Graphs, Discrete Mathematics for Computer Scientists.* EATCS-Monographs on Theoretical Computer Science. Springer, 1993.
79. G. Schmidt. *Decomposing Relations.* Technical Report 2002-09, Universität der Bundeswehr München, Fakultät für Informatik, 2002.
80. Gheorghe Ştefănescu. *Reaction and Control I. Mixing Additive and Multiplicative Network Algebras.* Logic Journal of the IGPL **6**(2) 349–368, 1998.
81. Gheorghe Ştefănescu. *Network Algebra.* Springer, London, 2000.
82. G. Struth. *Calculating Church-Rosser Proofs in Kleene Algebra.* In [83], pp. 276–290.
83. H. de Swart, ed. *Proc. RelMiCS 6, International Workshop on Relational Methods in Computer Science, Oisterwijk near Tilburg, Netherlands, 16–21 October 2001*, LNCS **2561**. Springer, 2002.
84. R. Verhoeven, R. Backhouse. *Towards Tool Support for Program Verification* and *Construction.* In J. Wing, J. Woodcock, J. Davies, eds., FM '99 – Formal Methods, LNCS **1709**, pp. 1128–1146. Springer, 1999.
85. M. M. Wenzel. *Isabelle/Isar — A Versatile Environment for Human-Readable Formal Proof Documents.* PhD thesis, Technische Universität München, Fakultät für Informatik, 2002.
86. M. Wenzel. *Type classes and overloading in higher-order logic.* In E. L. Gunter, A. Felty, eds., Theorem Proving in Higher-Order Logics, TPHOLs '97, LNCS **1275**, pp. 307–322. Springer, 1997.
87. M. Winter. *A New Algebraic Approach to L-fuzzy Relations Convenient to Study Crispness.* Information Sciences **139**(3–4) 233–252, 2001.
88. M. Winter. *Representation Theory of Goguen Categories.* Fuzzy Sets and Systems **138** 85–126, 2003.
89. H. Zierer. *Programmierung mit Funktionsobjekten: Konstruktive Erzeugung semantischer Bereiche und Anwendung auf die partielle Auswertung.* Dissertation, Technische Univ. München, Fakultät für Informatik, 1988. Report TUM-I8803.
90. H. Zierer. *Relation-Algebraic Domain Constructions.* Theoretical Computer Science **87** 163–188, 1991.

Journal on Relational Methods in Computer Science, Vol. 1, 2004, pp. 277 - 313
Received by the editors March 08, 2004, and, in revised form, October 14, 2004.
Published on December 10, 2004.

This article may be accessed via WWW at http://www.jormics.org.