Metaphorisms in Programming

José N. Oliveira

High Assurance Software Laboratory
INESC TEC and University of Minho
Braga, Portugal
(jno@di.uminho.pt)

Abstract. This paper introduces the *metaphorism* pattern of relational specification and addresses how specification following this pattern can be refined into recursive programs.
Metaphorisms express input-output relationships which preserve relevant information while at the same time some intended optimization takes place. Text processing, sorting, representation changers, etc., are examples of metaphorisms.
The kind of metaphorism refinement proposed in this paper is a strategy known as *change of virtual data structure*. It gives sufficient conditions for such implementations to be calculated using relation algebra and illustrates the strategy with the derivation of *quicksort* as example.

Keywords: Programming from specifications. Algebra of programming.

Politicians and diapers should be changed often and for the same reason.

(attributed to Mark Twain)

1 Introduction

The witty quote by 19th century author Mark Twain that provided inspiration for the title of this paper embodies a *metaphor* which the reader will surely appreciate. But, what do metaphors of this kind have to do with computer programming?

Programming theory has been structured around concepts such as *syntax*, *semantics*, *generative grammar* and so on, which have been imported from Chomskian linguistics. The basis is that syntax provides the *shape* of information and that semantics express information *contents* in a syntax-driven way (e.g. meaning of the whole dependent on the meaning of the parts).

Cognitive linguistics breaks with such a *generative* tradition in its belief that semantics are conveyed in a different way, just by juxtaposing concepts in the form of *metaphors* which let meanings permeate each other by an innate capacity of our brain to function metaphor-wise. Thus we are led to the *metaphors we live by*, quoting the classic textbook by Lakoff and Johnson [8]. If in a public discussion one of the opponents is said to have *counterattacked* with a *winning* argument, the underlying metaphor is *argument is war*; metaphor *time is money* underlies everyday phrases such as *wasting time*, *investing time* and so on; Twain's quote lives in the metaphor *politics is dirt*, the same that would enable one to say that somebody might need to *clean his/her reputation*, for instance.

In his *Philosophy of Rhetoric* [14], Richards finds three kernel ingredients in a metaphor, namely a *tenor* (e.g. *politicians*), a *vehicle* (e.g. *diapers*) and a shared *attribute* (e.g. ... left for the reader to guess). The *flow of meaning* is from vehicle to tenor, through the (as a rule left unspecified) common attribute.

In [11] the author sketched a brief characterization of this construction in the form of a "cospan"

$$\begin{tikzcd} \mathsf{T} \arrow[dr, "f"'] & & \mathsf{V} \arrow[dl, "g"] \\ & A & \end{tikzcd} \tag{1}$$

where $f : \mathsf{T} \to A$ and $g : \mathsf{V} \to A$ are functions extracting a common attribute (A) from both tenor (T) and vehicle (V). The cognitive, æsthetic, or witty power of a metaphor is obtained by *hiding* A, thereby establishing a *composite*, binary relationship[1] $\mathsf{T} \xleftarrow{f^\circ \cdot g} \mathsf{V}$ between tenor and vehicle — the "T is V" metaphor — which leaves A implicit.

It turns out that, in the field of program specification, many problem statements are *metaphorical* in the same (formal) sense: they are characterized as input-output relationships in which the *preservation* of some kernel information is kept implicit, possibly subject to some form of optimization.

An example of this is *text formatting*, a relationship between formatted and unformatted text whose metaphor consists in preserving the sequence of words of both, while the output text is optimized wrt. some visual criteria.[2] Other examples could have been given:

– Change of base of numeric representation — the number represented in the source is the same represented by the result, cf. the 'representation changers' of [5].
– Conversion of finite lists into balanced search trees — the information preserved is the set of elements of the source list; the optimization is the invariant induced on the output tree, making it adequate for searching, etc.

[1] Given a binary relation R, writing $b\ R\ a$ ($\equiv$ "b is related to a by R") means the same as $a\ R^\circ\ b$, where R° is said to be the *converse* of R. So R° corresponds to *passive voice*, check e.g. *John loves Mary* compared to *Mary is loved by John*: $(\mathit{loves})^\circ = (\mathit{is\ loved\ by})$.

[2] It is the privilege of those who don't work with WYSIWYG text processors to feel the rewarding (if not æsthetic) contrast between the window where source text is edited and that showing the corresponding, nice-looking PDF output.

– Source code refactoring — the meaning of the source program is preserved, the target code being better styled wrt. coding conventions and best practices.
– Sorting — the bag (multiset) of elements of the source list is preserved, the optimization consisting in obtaining an ordered output.

The *optimization* implicit in all these examples can be expressed by reducing the *vagueness* of relation $f^\circ \cdot g$ in (1) according to some criterion telling which outputs are better than others. This can be achieved by adding such criteria in the form of a relation R which "shrinks" $f^\circ \cdot g$,

$$M = (f^\circ \cdot g) \upharpoonright R \tag{2}$$

$$\begin{tikzcd}
 & \mathsf{T} & \\
\mathsf{T} \arrow[ur, "R"] \arrow[dr, "f"'] & & \mathsf{V} \arrow[ul, "M"'] \arrow[ll, "f^\circ \cdot g"'] \arrow[dl, "g"] \\
 & A &
\end{tikzcd}$$

using the "shrinking" operator of [9] for reducing non-determinism, see the diagram above. By unfolding the meaning of this relational operator, the relationship established by M (2) is the following:

$$t\ M\ v \equiv (f\ t = g\ v) \wedge \langle \forall\ t'\ :\ f\ t' = g\ v :\ t\ R\ t' \rangle$$

In words: for each input v, choose among all outputs t' with the same (hidden) attribute of v those which are better than any other with respect to R, if any.

We will refer to construction (2) as a *metaphorism* wherever V and T are inductive types and functions f and g are recursive on such types. A *metaphorism* $M = (f^\circ \cdot g) \upharpoonright R$ therefore involves two functions and an optimization criterion. In the text formatting metaphorism, for instance,

$$\begin{tikzcd}
{[String]} \arrow[dr, "\mathsf{concat} \cdot (\mathsf{map}\ words)"'] & & String \arrow[ll, "Format"'] \arrow[dl, "words"] \\
 & {[String]} &
\end{tikzcd}$$

arrow *Format* relates a string (source text) to a list of strings (output text lines) such that the original sequence of words is preserved when white space is discarded. Formatting consists in (re)introducing white space evenly throughout the output text lines. For economy of presentation, the diagram omits the optimization part,

$$Format = (\mathsf{map}\ words^\circ \cdot \mathsf{concat}^\circ \cdot words) \upharpoonright R \tag{3}$$

where $R : [String] \to [String]$ should capture the formatting criterion on lines of text, e.g. even spaced lines better than ill-spaced ones, etc. Metaphorism (3) also relies on a well-known property of relational converse, $(R \cdot S)^\circ = S^\circ \cdot R^\circ$.

Formally, nothing impedes f and g from being the same attribute function, in which case types V and T are also the same. Although less interesting from the strict (cognitive) metaphorical perspective, metaphorisms of this instance of (2) are very common in programming — take *sorting* as example, where V

and T are inhabited by finite sequences of the same type. Interestingly, some sorting algorithms actually involve *another* data-type, but this is hidden and kept implicit in the whole algorithmic process. Quicksort, for instance, unfolds recursively in a binary fashion which makes its use of the run-time heap look like a binary search tree — a pattern found in any *divide & conquer* algorithm. Because such a tree is not visible from outside, some authors refer to it as a *virtual* data structure [15].

Contribution. This paper addresses a generic process of implementing metaphorisms in a way that introduces *divide & conquer* strategies and the implicit virtual data structures. Conditions for the semantics of (2) to be preserved along the calculation process are discussed. Altogether, the reasoning shows how the "outer metaphor" of the specification (2) disappears and is replaced by a more implicit but more interesting "inner metaphor" which is at the heart of the implementation. We will restrict to a special case of (2) which is described in the next section and will use quicksort as running example.

Related work. This paper follows the line of research of reference [9] in investigating relational specification patterns which involve the "shrinking" combinator for controling vagueness and non-determinism. It also relates to previous work on representation changers [5] and on the relational algebra of programming, in general [1, 10]. Our calculation of sufficient conditions for implementing metaphorisms via change of virtual data-structure, illustrated with quicksort, can be regarded as a generalization and expansion of the derivation of the same algorithm in [1], where it is given in a rather brief and terse style.

Paper structure. The remainder of this paper is structured as follows. Sections 2 and 4 identify the class of metaphorisms addressed in the paper. Sect. 3 discusses implementation strategies for such metaphorisms. Sect. 5 finds generic conditions for these to be implemented by change of recursive pattern (virtual data-structure), an example of which is given in Sect. 6. Finally, Sect. 7 concludes. Some background on relation algebra and proofs of auxiliary results are given in appendices A and B, respectively.

2 Shrunken equivalence relations as metaphorisms

Wherever $f = g$ in (2) we get $M = (f^{\circ} \cdot f) \upharpoonright R$, a "shrunken" equivalence relation because $f^{\circ} \cdot f$ is an equivalence, known as the *kernel* of f, $\mathsf{ker}\, f = f^{\circ} \cdot f$:

$$M = (\mathsf{ker}\, f) \upharpoonright R \tag{4}$$

So $y\ M\ x$ means not only that $f\ y = f\ x$ (this is the information to be preserved), but also that y is "best" among all other y' such that $f\ y' = f\ x$ holds, as expressed by the meaning of the shrinking combinator [9, 13], see property (37) in the appendix: $S \upharpoonright R$ is the largest sub-relation X of S such that, for all $b', b \in B$, if there exists some $a \in A$ such that $b' X a \wedge b S a$ holds, then $b' R b$ holds.

Example: take $\mathsf{V} = \mathsf{T} = [A]$ parametric on type A and $f = \mathit{bag}$, the function that extracts the bag of elements of a finite list. The equivalence relation is $\mathit{Perm} = \mathsf{ker}\ \mathit{bag}$, that is $y\ \mathit{Perm}\ x$ means that y is a *permutation* of x. What about R? If sorting is the intended optimization, one might want to specify that $y\ R\ x$ holds wherever y has less "out-of-order" entries than x, something like e.g. (in Haskell concrete syntax)

$$\begin{array}{l} y\ R\ x = \mathit{oo}\ y \leqslant \mathit{oo}\ x\ \textbf{where} \\ \quad \mathit{oo}\ s = \mathsf{length}\ [n \mid n \leftarrow [0\,..\,\mathsf{length}\ s], n+1 < \mathsf{length}\ s, s\,!!\,n > s\,!!\,(n+1)] \end{array}$$

where oo is the function that counts "out-of-order" entries.

For the calculational theory of [1, 9] to be applicable to metaphorism (4), one needs to express either $\mathsf{ker}\,f$ or R (or both) as relational (un)folds, also referred to as ana/catamorphisms in the literature [1]. This makes perfect sense since, in many situations, T will be an inductive (initial, tree-like) data-type and f a *fold* which recursively extracts information from T using some function k for this. The popular notation $f = (\!|k|\!)$ will be used to express (relational) folds, see Appendix A for the basic properties of such a combinator.

It turns out that, if f is surjective, then the equivalence relation $\mathsf{ker}\,f$ will be a fold too, this time relational

$$\mathsf{ker}\,f = (\!|\mathsf{ker}\,f \cdot \mathsf{in}|\!) \tag{5}$$

where $\mathsf{T} \xleftarrow{\mathsf{in}} \mathsf{F}\,\mathsf{T}$ is the initial algebra of type T, for some functor F. (The proof of (5) is given in Appendix B.) So

$$\mathsf{ker}\,f \cdot \mathsf{in} = \mathsf{ker}\,f \cdot \mathsf{in} \cdot \mathsf{F}\,(\mathsf{ker}\,f) \tag{6}$$

holds, by fold-cancellation (28). In the case of lists, $\mathsf{F}\,X = 1 + A \times X$ and $\mathsf{in} = [\mathsf{nil}\ ,\mathsf{cons}]$, where $\mathsf{nil}\ x = [\,]$ is the constant function which yields the empty list and $\mathsf{cons}\ (a, s) = a : s$ adds a to the front of s. For $f = \mathit{bag}$, the fold which extracts the multiset of elements of a given list, $\mathsf{ker}\,f = \mathit{Perm}$ and we have the following property of the list permutation equivalence relation:

$$\mathit{Perm} \cdot \mathsf{in} = \mathit{Perm} \cdot \mathsf{in} \cdot (\mathsf{F}\,\mathit{Perm}) \tag{7}$$

The useful part of (7) is

$$\mathit{Perm} \cdot \mathsf{cons} = \mathit{Perm} \cdot \mathsf{cons} \cdot (\mathit{id} \times \mathit{Perm}) \tag{8}$$

where we use notation $R \times S$ to express the (Kronecker) *product* of two relations: $(b, d)\ (R \times S)\ (a, c)$ holds iff both $b\ R\ a$ and $d\ S\ c$ hold. Thus (8) is the same as

$$y\ \mathit{Perm}\ (a : x) = \langle \exists\ z\ \ :\ \ z\ \mathit{Perm}\ x :\ \ y\ \mathit{Perm}\ (a : z) \rangle$$

which means that permuting a sequence with at least one element is the same as adding it to the front of a permutation of the tail and permuting again.

The usefulness of (5, 6) is that the inductive definition of an equivalence relation $\mathsf{ker}\,f$ generated by a surjective fold f is such that the recursive branch $\mathsf{F}\,(\mathsf{ker}\,f)$ in the unfolding of $\mathsf{ker}\,f$ can be removed if convenient.

Another meaning of (6) is that $\mathsf{ker}\ f$ is a *congruence* for the initial algebra in, cf. the following theorem.

Theorem 1. *Let R be a congruence for an algebra $h : \mathsf{F}\,A \to A$ of functor F, that is*

$$h \cdot (\mathsf{F}\ R) \ \subseteq\ R \cdot h \tag{9}$$

holds and R is an equivalence relation. Then this is the same as stating:

$$R \cdot h = R \cdot h \cdot (\mathsf{F}\ R) \tag{10}$$

(Proof: see Appendix B.) □

3 Calculating metaphorisms

Given a metaphorism M (4) such that $f = \llparenthesis k \rrparenthesis$, it can immediately be shown that

$$M = (\mathsf{ker}\ \llparenthesis k \rrparenthesis) \upharpoonright R = (\llparenthesis k \rrparenthesis^\circ \upharpoonright R) \cdot \llparenthesis k \rrparenthesis \tag{11}$$

by this law of shrinking: $(S \cdot f) \upharpoonright R = (S \upharpoonright R) \cdot f$ [9]. Thus we have two main ways of calculating metaphorisms:

- either we shrink $\mathsf{ker}\ \llparenthesis k \rrparenthesis$ as a whole — a relational fold (5), as we have seen, or
- we shrink $\llparenthesis k \rrparenthesis^\circ$ and then fuse the outcome with $\llparenthesis k \rrparenthesis$ (11).

There is still a third way, known as *changing the virtual data structure* [15]. Given any *surjective* function $f : A \to B$, its image $\mathsf{img}\ f = f \cdot f^\circ$ — the converse-dual of $\mathsf{ker}\ f = f^\circ \cdot f$ — is such that $\mathsf{img}\ f = id$, where function $id\ x = x$ is the identity function, i.e. the equality relation on its type. So $\mathsf{img}\ f : B \to B$ can be pasted anywhere it typechecks, i.e. where type B is present. Suppose another $\llparenthesis h \rrparenthesis : \mathsf{W} \to \mathsf{T}$ is given which is surjective. Then

$$\begin{aligned} M &= (\mathsf{ker}\ \llparenthesis k \rrparenthesis) \upharpoonright R \\ &= (\mathsf{img}\ \llparenthesis h \rrparenthesis \cdot \mathsf{ker}\ \llparenthesis k \rrparenthesis) \upharpoonright R \\ &= \llparenthesis h \rrparenthesis \cdot (N \upharpoonright R') \ \mathbf{where}\ N = \llparenthesis h \rrparenthesis^\circ \cdot \mathsf{ker}\ \llparenthesis k \rrparenthesis \end{aligned} \tag{12}$$

for some R' to be calculated. Using type diagrams, the strategy starts from

$$\begin{tikzcd}
 & \mathsf{W} \arrow[dl, "\llparenthesis h \rrparenthesis"'] & & \mathsf{T} & \\
\mathsf{T} & & \mathsf{T} \arrow[ll, "id = \mathsf{img}\ \llparenthesis h \rrparenthesis"] \arrow[ul, "\llparenthesis h \rrparenthesis^\circ"'] \arrow[ur, "R"] \arrow[dr, "\llparenthesis k \rrparenthesis"'] & & \mathsf{T} \arrow[ll, "\mathsf{ker}\ \llparenthesis k \rrparenthesis"'] \arrow[ul, "M"'] \arrow[dl, "\llparenthesis k \rrparenthesis"] \\
 & & & A &
\end{tikzcd} \tag{13}$$

and then shifts the "ictus" of algorithmic control from type T to type W:

$$\begin{tikzcd}
& \mathsf{W} \arrow[dl, "(\!|h|\!)"'] \arrow[ddr, "(\!|k|\!)\cdot(\!|h|\!)"'] & & \\
\mathsf{T} & & & \mathsf{T} \arrow[ull, "N=(\!|h|\!)^\circ\cdot\mathsf{ker}\ (\!|k|\!)"'] \arrow[dl, "(\!|k|\!)"] \\
& & A &
\end{tikzcd} \tag{14}$$

In this way, the starting, "outer" metaphor involving only T disappears and gives place to an "inner" metaphor between inductive types W and T, moving the optimization inside in the form of a relation R', which needs to be calculated:

$$\begin{tikzcd}
& & \mathsf{W} \arrow[ddll, bend right, "(\!|h|\!)"'] & \\
& \mathsf{W} \arrow[ur, "R'"'] \arrow[ddr, "(\!|k|\!)\cdot(\!|h|\!)"'] & & \\
\mathsf{T} & & & \mathsf{T} \arrow[uul, "M"'] \arrow[ull, "N"] \arrow[dl, "(\!|k|\!)"] \\
& & A &
\end{tikzcd} \tag{15}$$

W is the (virtual) data type chosen to command the *divide & conquer* algorithmic control. It is usually a binary or n-ary tree structure and is regarded as virtual because, as mentioned above, it is doomed to disappear once the two-step composition process is fused into a single step.

In summary, finding a generic *divide & conquer* version of metaphorism $M = (\mathsf{ker}\ (\!|k|\!)) \upharpoonright R$ relying on virtual type W as *representation* of the original type T amounts to finding a function that implements the *divide* step, $(N \upharpoonright R')$ where $N = (\!|h|\!)^\circ \cdot \mathsf{ker}\ (\!|k|\!)$ and $(\!|h|\!)$ is an *abstraction* function. Finding R' is the hard part of the exercise, as we shall soon see.

4 Special case of shrinking

R in (2,4) is in general a *metric* indicating which structures are better than others, usually in the form of a preorder $R = \leqslant_h$ where h is the metric attribute to be compared and $\leqslant_h$ abbreviates $h^\circ \cdot (\leqslant) \cdot h$, that is: $y \leqslant_h x \equiv (h\ y) \leqslant (h\ x)$. For instance, trees can be compared by measuring their depth; programs under refactoring compared by counting LoC, and so on.

However, R can also take the form $R = \Psi \cdot \top$ in (4), where $\top$ is the "topmost" relation of its type (32) — $b \top a$ is true for every a and b — and $\Psi \subseteq id$ is a *partial identity* specifying some form of *selection*.[3] This indicates that only the outputs satisfying Ψ are regarded as good enough.

In case $R = \Psi \cdot \top$, (4) reduces to $M = \Psi \cdot \mathsf{ker}\, f$, since $\mathsf{ker}\, f$ is an equivalence relation and therefore entire (i.e. totally defined) and the following result holds

$$R \restriction (\Psi \cdot \top) = \Psi \cdot R \quad \Leftarrow \quad R \text{ is entire} \tag{16}$$

(Proof in Appendix B.) It is this special case of (4) which will concern us in the sequel, leaving the full generality of (4) for future work.

5 Shrinking metaphorisms into hylomorphisms

Consider metaphorisms of form $M = \Psi \cdot \mathsf{ker}\ (\!|k|\!)$ which, as we have seen above, are special cases of (4). Suppose $(\!|h|\!) : \mathsf{W} \to \mathsf{T}$ is an abstraction function (surjective) which ensures that every inhabitant of T can be represented by one or more inhabitants of W, as in diagrams (13) to (15). Below we record the calculation implicit in such diagrams:

$$\begin{array}{cl}
& M = \Psi \cdot \mathsf{ker}\ (\!|k|\!) \\
\equiv & \quad \{\ \mathsf{img}\ (\!|h|\!) = id \text{ because } (\!|h|\!) \text{ is surjective }\ \} \\
& M = \mathsf{img}\ (\!|h|\!) \cdot \Psi \cdot \mathsf{ker}\ (\!|k|\!) \\
\equiv & \quad \{\ \text{inline image}\ \} \\
& M = (\!|h|\!) \cdot (\!|h|\!)^\circ \cdot \Psi \cdot \mathsf{ker}\ (\!|k|\!) \\
\equiv & \quad \{\ \text{hint: assume } \Phi \text{ such that } (\!|h|\!) \cdot \Phi = \Psi \cdot (\!|h|\!)\ ;\ \text{converses; } \Psi^\circ = \Psi\ \} \\
& M = (\!|h|\!) \cdot \Phi \cdot \underbrace{(\!|h|\!)^\circ \cdot \mathsf{ker}\ (\!|k|\!)}_{N}
\end{array}$$

The goals are, therefore: (a) to find Φ such that

$$(\!|h|\!) \cdot \Phi = \Psi \cdot (\!|h|\!) \tag{17}$$

holds, and (b) to convert $\Phi \cdot (\!|h|\!)^\circ \cdot \mathsf{ker}\ (\!|k|\!)$ into the converse of a fold, which we denote as usual by $[\!(g)\!]$, for some g.[4] Then the original metaphorism will be converted into a so-called *hylomorphism* [1] $(\!|h|\!) \cdot [\!(g)\!]$ with a "change of data-structure".

As W and T are inductive types, the two partial identities (coreflexives) will take the shape (say) $\Phi = (\!|\mathsf{in}_\mathsf{W} \cdot \Omega|\!)$ and $\Psi = (\!|\mathsf{in}_\mathsf{T} \cdot \Theta|\!)$, where in_W and in_T are the initial algebras of types W and T, respectively.

[3] We use uppercase Greek letters (e.g. Ψ, Φ, ...) to denote *partial identities*, also known as *coreflexives*, *monotypes* or *tests* [2, 3, 7]. Every partial identity Ψ is such that $\Psi \subseteq id$ and is in one-to-one correspondence with some predicate q. As in [9] we write $\Psi = q?$ wherever we want to indicate that q is the predicate captured by Ψ. Thus $\Psi = q?$ has the pointwise meaning $b\ \Psi\ a \equiv b = a \wedge q\ a$.

[4] Converses of folds are usually termed *unfolds* or anamorphisms. Notation $[\!(R)\!]$ means $(\!|R^\circ|\!)^\circ$.

Calculation of (17) proceeds by fusion (27), aiming to reduce both $(\!|h|\!) \cdot \Phi$ and $\Psi \cdot (\!|h|\!)$ to some fold $(\!|R|\!)$ over W. On the one side,

$$\Psi \cdot (\!|h|\!) = (\!|R|\!) \Leftarrow \Psi \cdot h = R \cdot (\mathsf{F}\,\Psi) \tag{18}$$

On the other side:

$$\begin{array}{ll}
& (\!|h|\!) \cdot \Phi = (\!|R|\!) \\
\equiv & \quad \{ \text{ inline } \Phi = (\!|\mathsf{in_W} \cdot \Omega|\!) \ \} \\
& (\!|h|\!) \cdot (\!|\mathsf{in_W} \cdot \Omega|\!) = (\!|R|\!) \\
\Leftarrow & \quad \{ \text{ fusion (27) } \} \\
& (\!|h|\!) \cdot \mathsf{in_W} \cdot \Omega = R \cdot \mathsf{F}\,(\!|h|\!) \\
\equiv & \quad \{ \text{ cancellation of } (\!|h|\!) \text{ (28) } \} \\
& h \cdot \mathsf{F}\,(\!|h|\!) \cdot \Omega = R \cdot \mathsf{F}\,(\!|h|\!) \\
\equiv & \quad \{ \text{ assume } \Lambda \text{ such that } \mathsf{F}\,(\!|h|\!) \cdot \Omega = \Lambda \cdot \mathsf{F}\,(\!|h|\!) \ \} \\
& h \cdot \Lambda \cdot \mathsf{F}\,(\!|h|\!) = R \cdot \mathsf{F}\,(\!|h|\!) \\
\Leftarrow & \quad \{ \text{ Leibniz } \} \\
& h \cdot \Lambda = R
\end{array}$$

Replacing this in $\Psi \cdot h = R \cdot \mathsf{F}\,\Psi$, the side condition of (18), we get: $\Psi \cdot h = h \cdot \Lambda \cdot (\mathsf{F}\,\Psi)$. Let us summarize both calculations in the form of a theorem.

Theorem 2. *Let $(\!|h|\!) : \mathsf{W} \to \mathsf{T}$ be an abstraction of inductive type T by W, and $\Psi = (\!|\mathsf{in_T} \cdot \Theta|\!)$ and $\Phi = (\!|\mathsf{in_W} \cdot \Omega|\!)$ be partial identities representing inductive predicates over such types.*

For $(\!|h|\!) \cdot \Phi = \Psi \cdot (\!|h|\!)$ (17) to hold, search for the existence of $\Lambda : \mathsf{F\,T} \to \mathsf{F\,T}$ such that

$$\Psi \cdot h = h \cdot \Lambda \cdot \mathsf{F}\Psi \tag{19}$$

$$\mathsf{F}\,(\!|h|\!) \cdot \Omega = \Lambda \cdot \mathsf{F}\,(\!|h|\!) \tag{20}$$

hold, where F is the base functor of W, that is, $\mathsf{in_W} : \mathsf{F\,W} \to \mathsf{W}$.
□

Note that condition (20) establishes Ω as *weakest precondition* for $\mathsf{F}\,(\!|h|\!)$ to ensure Λ on its output, cf. (35) in Appendix A. Likewise, (19) establishes Λ as weakest precondition for h to maintain invariant Ψ.

Searching for the anamorphism. Thus far, the starting metaphor $\mathsf{ker}\ (\!|k|\!)$ has been left aside. Going back to

$$M = (\!|h|\!) \cdot \Phi \cdot \underbrace{(\!|h|\!)^{\circ} \cdot \mathsf{ker}\ (\!|k|\!)}_{N}$$

our aim is to convert $N = \Phi \cdot (\!|h|\!)^\circ \cdot \mathsf{ker}\ (\!|k|\!)$ into $[\!(R)\!]$ for some R. Below we shall need the extra condition that $\mathsf{ker}\ (\!|k|\!)$ is a congruence for h, that is,

$$h \cdot \mathsf{F}\ \mathsf{ker}\ (\!|k|\!) \ \subseteq\ \mathsf{ker}\ (\!|k|\!) \cdot h \tag{21}$$

holds, equivalent to

$$\mathsf{ker}\ (\!|k|\!) \cdot h = \mathsf{ker}\ (\!|k|\!) \cdot h \cdot (\mathsf{F}\ \mathsf{ker}\ (\!|k|\!)) \tag{22}$$

by Theorem 1. Another alternative to state (21) is

$$(\!|k|\!) \cdot h \leqslant \mathsf{F}\ (\!|k|\!) \tag{23}$$

meaning that $(\!|k|\!) \cdot h$ should be *less injective* (39) than $\mathsf{F}\ (\!|k|\!)$, see Appendix B. We shall also need the assumption:

$$\mathsf{F}\,(\mathsf{ker}\ (\!|k|\!)) \cdot \Lambda = \Lambda \cdot \mathsf{F}\,(\mathsf{ker}\ (\!|k|\!)) \tag{24}$$

We calculate:

$$\begin{array}{ll}
 & \Phi \cdot (\!|h|\!)^\circ \cdot \mathsf{ker}\ (\!|k|\!) = [\!(R)\!] \\
\equiv & \qquad \{\ \text{converses}\ \} \\
 & \mathsf{ker}\ (\!|k|\!) \cdot (\!|h|\!) \cdot \Phi = (\!|R^\circ|\!) \\
\equiv & \qquad \{\ (\!|h|\!) \cdot \Phi = \Psi \cdot (\!|h|\!)\ (17),\ \text{Theorem 2}\ \} \\
 & \mathsf{ker}\ (\!|k|\!) \cdot \Psi \cdot (\!|h|\!) = (\!|R^\circ|\!) \\
\Leftarrow & \qquad \{\ \text{fusion (27)}\ \} \\
 & \mathsf{ker}\ (\!|k|\!) \cdot \Psi \cdot h = R^\circ \cdot \mathsf{F}\ (\mathsf{ker}\ (\!|k|\!) \cdot \Psi) \\
\Leftarrow & \qquad \{\ (19);\ \text{functor}\ \mathsf{F};\ \text{Leibniz}\ \} \\
 & \mathsf{ker}\ (\!|k|\!) \cdot h \cdot \Lambda = R^\circ \cdot \mathsf{F}\ \mathsf{ker}\ (\!|k|\!) \\
\equiv & \qquad \{\ (22)\ \} \\
 & \mathsf{ker}\ (\!|k|\!) \cdot h \cdot (\mathsf{F}\ \mathsf{ker}\ (\!|k|\!)) \cdot \Lambda = R^\circ \cdot \mathsf{F}\ \mathsf{ker}\ (\!|k|\!) \\
\Leftarrow & \qquad \{\ (24)\ ;\ \text{Leibniz}\ ;\ \text{converses}\ \} \\
 & R = \Lambda \cdot h^\circ \cdot \mathsf{ker}\ (\!|k|\!) \\
\Box &
\end{array}$$

In summary, note how the original metaphorism $\Psi \cdot \mathsf{ker}\ (\!|k|\!)$ gets converted into a hylomorphism whose *divide* step is another metaphorism:

$$R = \Lambda \cdot ((\!|k|\!) \cdot h)^\circ \cdot (\!|k|\!) \tag{25}$$

That is, the "outer" metaphor which we started from (involving only T) disappears and gives place to an "inner" metaphor between inductive types W and T, whereby the optimization is internalized.

This "inner" metaphor is more interesting, as we can see by looking at an example of this reasoning.

6 Example: Quicksort

This section shows how the derivation of *quicksort* as given in e.g. [1] corresponds to the implementation strategy for metaphorisms given above, under the following instantiations:

- T is the usual finite list datatype with constructors (say) nil and cons, that is, $\mathsf{in}_{\mathsf{T}} = [\mathsf{nil}\ , \mathsf{cons}]$.
- W is the binary tree data type whose base is $\mathsf{F} f = id + id \times (f \times f)$ and whose initial algebra is (say) $\mathsf{in}_{\mathsf{W}} = [\mathsf{empty}\ , \mathsf{fork}]$.
- $(\!| k |\!) = bag$, the function which converts a list into the bag (multiset) of its elements.
- $\mathsf{ker}\ bag = Perm$, the list permutation relationship (the metaphor we start from).
- $(\!| h |\!) = \mathit{flatten}$, for $h = [\mathsf{nil}\ , inord]$ where $inord\ (a,(x,y)) = x +\!\!+ [a] +\!\!+ y$; that is, *flatten* is the binary tree into finite list surjection.
- Ψ filters ordered lists, $\Psi = (\!|[\mathsf{nil}\ , \mathsf{cons}] \cdot (id + \Theta)|\!)$ where $\Theta = mn?$ for $mn\ (x, xs) = \langle \forall\ x' \ :\ x' \in_{\mathsf{T}} xs :\ x' \geqslant x \rangle$, where $\in_{\mathsf{T}}$ denotes list membership; that is, predicate $mn\ (x, xs)$ ensures that list $x : xs$ is such that x is at most the minimum of xs, if it exists.

As seen in Sect. 5, we have to search for some partial identity $\Lambda = id + \Upsilon : id + id \times (\mathsf{T} \times \mathsf{T}) \to id + id \times (\mathsf{T} \times \mathsf{T})$ which, following (19), should be the weakest precondition for $[\mathsf{nil}\ , inord]$ to preserve ordered lists (Ψ):

$$\begin{array}{ll} & \Psi \cdot [\mathsf{nil}\ , inord] = [\mathsf{nil}\ , inord] \cdot (id + \Upsilon) \cdot (id + id \times (\Psi \times \Psi)) \\ \equiv & \qquad \{\ \text{coproducts; } \Psi \cdot \mathsf{nil} = \mathsf{nil}\text{, since the empty list is trivially ordered}\ \} \\ & \Psi \cdot inord = inord \cdot \Upsilon \cdot (id \times (\Psi \times \Psi)) \end{array}$$

Let *ord* and *wpl* be the predicates represented by partial identities Ψ and Υ, respectively, that is $\Psi = ord?$ and $\Upsilon = wpl?$. Unfolding *inord* we get the following pointwise calculation of weakest pre-condition *wpl*:

$$\begin{array}{ll} & ord\ (x +\!\!+ [a] +\!\!+ y) \\ \equiv & \qquad \{\ \text{pointwise definition of ordered lists}\ \} \\ & (ord\ x) \wedge (ord\ y) \wedge \underbrace{\langle \forall\ b\ :\ b \in_{\mathsf{T}} x :\ b \leqslant a \rangle \wedge \langle \forall\ b\ :\ b \in_{\mathsf{T}} y :\ a \leqslant b \rangle}_{wpl\ (a,(x,y))} \end{array}$$

From this we get the following relational definition of the *divide* step (25) of the implementation,

$$\begin{array}{l} R : [A] \to 1 + A \times ([A] \times [A]) \\ R = (id + wpl?) \cdot (bag \cdot [\mathsf{nil}\ , inord])^{\circ} \cdot bag \end{array} \qquad (26)$$

which we unfold as follows, by letting $R^\circ = [R_1^\circ\ , R_2^\circ]$ and using the converse of (26):

$$
\begin{array}{ll}
& [R_1^\circ\ , R_2^\circ] = \mathit{bag}^\circ \cdot (\mathit{bag} \cdot [\mathsf{nil}\ , \mathit{inord}]) \cdot (\mathit{id} + \mathit{wpl}?) \\
\equiv & \quad \{\ \mathit{bag}^\circ \cdot \mathit{bag} = \mathit{Perm};\ \mathit{Perm}.\mathsf{nil} = \mathsf{nil};\ \text{converses}\ \} \\
& \left\{ \begin{array}{l} R_1 = \mathsf{nil}^\circ \\ R_2 = \mathit{wpl}? \cdot \mathit{inord}^\circ \cdot \mathit{Perm} \end{array} \right.
\end{array}
$$

In summary, $y\ R\ x$ has the following meaning: either $x = [\,]$ and R yields the unique inhabitant of singleton type 1 (cf. R_1) or x is non-empty and R splits a permutation of x into two halves y and z separated by a "pivot" a, cf.

$$(a, (y, z))\ R_2\ x = \mathit{wpl}\ (a\ (y, z)) \wedge (y +\!\!+ [a] +\!\!+ z)\ \mathit{Perm}\ x$$

where wpl was calculated above. Pivot a can be taken from any position in the list. In the standard version, a is the head of x. There is, still, a check-list of proofs to discharge.

Ensuring bi-ordered (virtual) intermediate trees. We start from the instantiation of (20) for this exercise,

$$\mathsf{F}\,\mathit{flatten} \cdot (\mathit{id} + \mathit{wp}'?) = (\mathit{id} + \mathit{wpl}?) \cdot \mathsf{F}\,\mathit{flatten}$$

where the goal is to find another weakest precondition wp' which is basically wpl "passed along" $\mathsf{F}\,\mathit{flatten}$ from lists to trees:

$$
\begin{array}{ll}
& (\mathit{id} \times (\mathit{flatten} \times \mathit{flatten})) \cdot \mathit{wp}'? = \mathit{wpl}? \cdot (\mathit{id} \times (\mathit{flatten} \times \mathit{flatten})) \\
\equiv & \quad \{\ (35)\ \} \\
& \mathit{wp}' = \mathsf{wp}(\mathit{id} \times (\mathit{flatten} \times \mathit{flatten}), \mathit{wpl}) \\
\equiv & \quad \{\ \text{go pointwise}\ \} \\
& \mathit{wp}'\ (a, (t_1, t_2)) = \mathit{wpl}\ (a, (\mathit{flatten}\ t_1, \mathit{flatten}\ t_2)) \\
\equiv & \quad \{\ \text{definition of } \mathit{wpl}\ \} \\
& \mathit{wp}'\ (a, (t_1, t_2)) = \left\{ \begin{array}{l} \langle \forall\ b\ :\ b \in_{\mathsf{T}} (\mathit{flatten}\ t_1) :\ b \leqslant a \rangle \\ \langle \forall\ b\ :\ b \in_{\mathsf{T}} (\mathit{flatten}\ t_2) :\ a \leqslant b \rangle \end{array} \right. \\
\equiv & \quad \{\ \text{define } \in_{\mathsf{W}} = \in_{\mathsf{T}} \cdot \mathit{flatten}\ \} \\
& \mathit{wp}'\ (a, (t_1, t_2)) = \langle \forall\ b\ :\ b \in_{\mathsf{W}} t_1 :\ b \leqslant a \rangle \wedge \langle \forall\ b\ :\ b \in_{\mathsf{W}} t_2 :\ a \leqslant b \rangle)
\end{array}
$$

Recall that $\Omega = \mathit{id} + \mathit{wp}'?$. In words, wp' in $\Phi = (\!|\mathsf{in}_{\mathsf{W}} \cdot \Omega|\!) = (\!|\mathsf{in}_{\mathsf{W}} \cdot (\mathit{id} + \mathit{wp}'?)|\!)$ ensures that the first part of the implementation, controlled by the *divide step* coalgebra R calculated above (26) yields trees which are *bi-ordered.* Trees with this property are known as *binary search trees* [6].

Preserving the metaphor. Next we consider side condition (23), which instantiates to:

$$
\begin{array}{ll}
& bag \cdot [\mathsf{nil}\ , inord] \leqslant id + id \times (bag \times bag) \\
\Leftarrow & \quad \{ \text{ coproducts; (40) } \} \\
& bag \cdot \mathsf{nil} + bag \cdot inord \leqslant id + id \times (bag \times bag) \\
\equiv & \quad \{ \text{ (41) ; any } f \leqslant id \text{ [12] ; let } bag' = bag \cdot inord \ \} \\
& bag' \leqslant id \times (bag \times bag) \\
\equiv & \quad \{ \ bag' \text{ loses more information than } id \times (bag \times bag) \ \} \\
& true
\end{array}
$$

In the last step we can easily observe that, while from $(a, (bag\ x, bag\ y))$ we can obtain $bag'\ (a, (x, y))$, the converse is false: bag' merges the multisets of x and y too quickly. Thus bag' is less injective than $id \times (bag \times bag)$.

Downto the multiset level. Finally, we have to check (24), for $\Lambda = id + \Upsilon = id + wpl?$:

$$
\begin{array}{ll}
& \mathsf{F}\,Perm \cdot \Lambda = \Lambda \cdot \mathsf{F}\,Perm \\
\equiv & \quad \{ \ Perm = \mathsf{ker}\ bag \ ; \ \mathsf{F}\,(R^\circ) = (\mathsf{F}\,R)^\circ \ \} \\
& \mathsf{ker}\ (\mathsf{F}\ bag) \cdot \Lambda = \Lambda \cdot \mathsf{ker}\ (\mathsf{F}\ bag) \\
\equiv & \quad \{ \ \mathsf{F}\,R = id + id \times (R \times R) \text{ ; kernel of the sum (42); } \Lambda = id + wpl? \ \} \\
& \mathsf{ker}\ (id \times (bag \times bag)) \cdot wpl? = wpl? \cdot \mathsf{ker}\ (id \times (bag \times bag)) \\
\Leftarrow & \quad \{ \text{ (36), assuming that condition } q \text{ exists } \} \\
& wpl = \mathsf{wp}(id \times (bag \times bag), q)
\end{array}
$$

Thus we have to find post-condition q ensured by $id \times (bag \times bag)$ with wpl as weakest-precondition. We proceed as before:

$$
\begin{array}{ll}
& wpl\ (a, (x, y)) = q\ (a, (bag\ x, bag\ y)) \\
\equiv & \quad \{ \text{ unfold } wpl \ \} \\
& q\ (a, (bag\ x, bag\ y)) = \begin{cases} \langle \forall\ b\ :\ b \in_{\mathsf{T}} x :\ b \leqslant a \rangle \\ \langle \forall\ b\ :\ b \in_{\mathsf{T}} y :\ a \leqslant b \rangle \end{cases} \\
\equiv & \quad \{ \text{ assume } \in_{\mathsf{B}} \text{ such that } \in_{\mathsf{T}} = \in_{\mathsf{B}} \cdot bag \ \} \\
& q\ (a, (bag\ x, bag\ y)) = \begin{cases} \langle \forall\ b\ :\ b \in_{\mathsf{B}} (bag\ x) :\ b \leqslant a \rangle \\ \langle \forall\ b\ :\ b \in_{\mathsf{B}} (bag\ y) :\ a \leqslant b \rangle \end{cases} \\
\Leftarrow & \quad \{ \text{ substitution } \} \\
& q\ (a, (b_1, b_2)) = \begin{cases} \langle \forall\ b\ :\ b \in_{\mathsf{B}} b_1 :\ b \leqslant a \rangle \\ \langle \forall\ b\ :\ b \in_{\mathsf{B}} b_2 :\ a \leqslant b \rangle \end{cases}
\end{array}
$$

□

Finally, multiset membership $\epsilon_{\mathsf{B}} = \in \cdot \mathit{support}$ can be obtained by taking multiset *supports*, whereby we land in standard set membership ($\in$). Thus we have a chain of memberships, from sets, to multisets, to finite lists and finally to binary (search) trees.

Note how this last proof of the check-list goes down to the very essence of sorting as a metaphorism: the attribute of a finite list which any sorting function is bound to preserve is the multiset (bag) of its elements.

7 Conclusions and future work

This paper identifies a pattern of relational specification, termed *metaphorism*, in which some kernel information of the input is preserved at the same time some form of optimization takes place towards the output. Text processing, sorting and representation changers are given as examples of metaphorisms. It then addresses the problem of refining metaphorisms into recursive programs.

The kind of metaphorism refinement proposed is known as *changing the virtual data structure*, whereby *divide & conquer* strategies can be introduced. The paper gives sufficient conditions for such implementations to be calculated in general, and gives the derivation of *quicksort* as example. This derivation can be regarded as a generalization of the reasoning about the same algorithm given in [1].

Altogether, the paper shows how such *divide & conquer* refinement strategies consist of replacing the “outer metaphor” of the starting specification (metaphorism) by a more implicit but more interesting “inner metaphor”, which is at the heart of the implementation. In the quicksort example, the “outer metaphor” relates lists which permute each other, while the “inner metaphor” relates lists with binary search trees.

This research can be framed into the area of investigating how to manage or refine specification vagueness (non-determinism) by means of the “shrinking” combinator proposed in references [9, 13]. The pattern of shrinking addressed in the current paper is, however, far too restrictive: what is expected in general is shrinking over *preorders* which measure *progress* with respect to some other attribute, e.g. reducing the number of “out-of-order” entries in sorting, as presented in the introduction. Note how such metaphorisms expose the *variant*/*invariant* duality essential to program correctness and termination proofs, in their own way: there are two main attributes in the game, one is to be preserved (the essence of the metaphor, cf. *invariant*) while the other is to be mini(maxi)mized (the essence of the optimization, cf. *variant*).

This paper is intended as starting point for future work in exploiting the metaphorism concept in program derivation. Candidate case studies in program refactoring or text processing already pose significant challenges when compared to the sorting example given in the current paper. Comparative work is also welcome, in particular checking what benefits can be expected from regarding representation changers [5] from the metaphorism perspective, or (back to sorting) checking how the ideas of this paper combine with the work on parametric permutation functions by Henglein [4].

From the linguistics perspective, metaphorisms are *formal* metaphors and not exactly *cognitive* metaphors. But computer science is full of these as well, as its terminology (e.g. "stack", "pipe", "memory", "driver") amply shows. If a picture is worth a thousand words, perhaps a good metaphor is worth a thousand axioms?

Acknowledgements

The author wishes to thank the anonymous referees for their comments and suggestions. This work is funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the *FCT - Fundação para a Ciência e a Tecnologia* (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-020537.

References

[1] R. Bird and O. de Moor. *Algebra of Programming.* Series in Computer Science. Prentice-Hall International, 1997.

[2] H. Doornbos, R. Backhouse, and J. van der Woude. A calculational approach to mathematical induction. *TCS*, 179(1–2):103–135, 1997.

[3] P.J. Freyd and A. Scedrov. *Categories, Allegories*, volume 39 of *Mathematical Library.* North-Holland, 1990.

[4] F. Henglein. What is a sorting function? *J. Logic and Algebraic Programming (JLAP)*, 78(5):381–401, 2009.

[5] G. Hutton and E. Meijer. Back to basics: Deriving representation changers functionally. *Journal of Functional Programming*, 6(1):181–188, 1996.

[6] D.E. Knuth. *The Art of Computer Programming.* Addison/Wesley, 2nd edition, 1997/98. 3 volumes. First edition's dates: 1968 (volume 1), 1969 (volume 2) and 1973 (volume 3).

[7] D. Kozen. Kleene algebra with tests. *ACM Trans. Program. Lang. Syst.*, 19(3):427–443, 1997.

[8] G. Lakoff and M. Johnson. *Metaphors we live by.* University of Chicago Press, Chicago, 1980.

[9] S.-C. Mu and J.N. Oliveira. Programming from Galois connections. *JLAP*, 81(6):680–704, 2012.

[10] J.N. Oliveira. Extended Static Checking by Calculation using the Pointfree Transform. volume 5520 of *LNCS*, pages 195–251. Springer-Verlag, 2009.

[11] J.N. Oliveira. On the *'A'* that links the *'M'*s of maths, music and maps, 2013. Contributed talk to the 2013 CEHUM Autumn Colloquium XV (Maths and Comp. Science Panel), U. Minho, Braga, 21-23 Nov. 2013.

[12] J.N. Oliveira. A relation-algebraic approach to the "Hoare logic" of functional dependencies. *JLAP*, 83(2):249–262, 2014.

[13] J.N. Oliveira and M.A. Ferreira. Alloy meets the algebra of programming: A case study. *IEEE Trans. Soft. Eng.*, 39(3):305–326, 2013.
[14] I.A. Richards. *The Philosophy of Rhetoric.* Oxford University Press, 1936.
[15] D. Swierstra and O. de Moor. Virtual data structures. In B. Möller, H. Partsch, and S. Schuman, editors, *Formal Program Development*, volume 755 of *LNCS*, pages 355–371. Springer, 1993.

A Background — basic definitions and results of relation algebra

Relational folds: this paper relies on basic properties of relational folds over a type T defined by initial algebra $\mathsf{T} \xleftarrow{\mathsf{in}} \mathsf{F}\,\mathsf{T}$ on functor F, namely *fusion*

$$S \cdot (\!|R|\!) = (\!|Q|\!) \quad \Leftarrow \quad S \cdot R = Q \cdot \mathsf{F}\,S \tag{27}$$

and *cancellation*,

$$(\!|R|\!) \cdot \mathsf{in} = R \cdot \mathsf{F}\,(\!|R|\!) \tag{28}$$

both stemming from *universal property*:

$$X = (\!|R|\!) \quad \equiv \quad X \cdot \mathsf{in} = R \cdot \mathsf{F}\,X \tag{29}$$

Shunting rules for function f, where R, S are arbitrary binary relations:

$$f \cdot R \subseteq S \equiv R \subseteq f^\circ \cdot S \tag{30}$$
$$R \cdot f^\circ \subseteq S \equiv R \subseteq S \cdot f \tag{31}$$

Top relation — the topmost relation of its type can be defined by

$$!^\circ \cdot ! = \top \tag{32}$$

where $! : A \to 1$ is the constant function which maps every argument to the unique element of singleton type 1.

Pre/post restrictions where Φ and Ψ are partial identities:

$$R \cdot \Phi = R \cap \top \cdot \Phi \tag{33}$$
$$\Psi \cdot R = R \cap \Psi \cdot \top \tag{34}$$

Weakest pre-conditions: let $p?$ and $q?$ be the partial identities for predicates p and q, respectively, and $\mathsf{wp}(f, q)$ denote the *weakest precondition* for function f to ensure post-condition q, that is: $\mathsf{wp}(f, q)\ x = q\ (f\ x)$. Then the following properties hold (proofs in Appendix B):

$$f \cdot p? = q? \cdot f \quad \equiv \quad p = \mathsf{wp}(f, q) \tag{35}$$
$$\mathsf{ker}\,f \cdot p? = p? \cdot \mathsf{ker}\,f \quad \Leftarrow \quad p = \mathsf{wp}(f, q) \tag{36}$$

"**Shrinking**" — let $B \xleftarrow{X,S} A$ and $B \xleftarrow{R} B$ be binary relations in universal property [9]:

$$X \subseteq S \upharpoonright R \;\;\equiv\;\; X \subseteq S \;\wedge\; X \cdot S^{\circ} \subseteq R \tag{37}$$

Coproducts: coproduct notation $C \xleftarrow{[R\ ,S]} A + B$ denotes the junction of relations $C \xleftarrow{R} A$ and $C \xleftarrow{S} B$ (coproduct). Direct sum $R + S$ is the same as $[i_1 \cdot R\ , i_2 \cdot S]$, where i_1 and i_2 are the *injections* associated to datatype sums.

Injectivity preorder: the kernel of a relation R,

$$\mathsf{ker}\ R \;\overset{\mathrm{def}}{=}\; R^{\circ} \cdot R \tag{38}$$

measures the *injectivity* of R. As in [12] we capture this by introducing a preorder on relations which compares their *injectivity*

$$R \leqslant S \;\;\equiv\;\; \mathsf{ker}\ S \subseteq \mathsf{ker}\ R \tag{39}$$

and satisfies, among many others, the following properties:

$$[R\ , S] \leqslant R + S \tag{40}$$

$$R + S \leqslant P + Q \;\equiv\; R \leqslant P \wedge S \leqslant Q \tag{41}$$

Moreover:

$$\mathsf{ker}\ (R + S) = \mathsf{ker}\ R + \mathsf{ker}\ S \tag{42}$$

$$\mathsf{ker}\ (R \times S) = \mathsf{ker}\ R \times \mathsf{ker}\ S \tag{43}$$

B Proofs of auxiliary results

Proof of (5), where $f = (\!|k|\!)$:

$$\begin{array}{ll}
& \mathsf{ker}\, f = (\!|\mathsf{ker}\, f \cdot \mathsf{in}|\!) \\
\equiv & \quad \{\ \text{inline definition } f = (\!|k|\!)\ ;\ \mathsf{ker}\, f = f^{\circ} \cdot f\ \} \\
& (\!|k|\!)^{\circ} \cdot (\!|k|\!) = (\!|(\!|k|\!)^{\circ} \cdot (\!|k|\!) \cdot \mathsf{in}|\!) \\
\Leftarrow & \quad \{\ \text{fusion (27)}\ \} \\
& (\!|k|\!)^{\circ} \cdot k = (\!|k|\!)^{\circ} \cdot (\!|k|\!) \cdot \mathsf{in} \cdot \mathsf{F}\,(\!|k|\!)^{\circ} \\
\equiv & \quad \{\ \text{cancellation (28)}\ \} \\
& (\!|k|\!)^{\circ} \cdot k = (\!|k|\!)^{\circ} \cdot k \cdot \mathsf{F}\,(\!|k|\!) \cdot \mathsf{F}\,(\!|k|\!)^{\circ} \\
\Leftarrow & \quad \{\ \text{factor } (\!|k|\!)^{\circ} \cdot k \text{ out (Leibniz) ; functor } \mathsf{F}\ \} \\
& id = \mathsf{F}\,((\!|k|\!) \cdot (\!|k|\!)^{\circ}) \\
\equiv & \quad \{\ f = (\!|k|\!)\ ;\ \mathsf{img}\, f = f \cdot f^{\circ} = id \text{ assuming } f \text{ surjective}\ \} \\
& id = \mathsf{F}\, id
\end{array}$$

$\equiv$ $\{$ functor F: $\mathsf{F}\, id = id$ $\}$

$true$

$\square$

Proof of Theorem 1:

$$R \cdot h = R \cdot h \cdot (\mathsf{F}\ R)$$

$\equiv$ $\{$ $R \cdot h \subseteq R \cdot h \cdot (\mathsf{F}\ R)$ holds by $id \subseteq \mathsf{F}\ R$, since $id \subseteq R$ $\}$

$$R \cdot h \cdot (\mathsf{F}\ R) \subseteq R \cdot h$$

$\equiv$ $\{$ the lower R can be cancelled, since R is an equivalence (see below) $\}$

$$h \cdot (\mathsf{F}\ R) \subseteq R \cdot h$$

$\square$

The last step can be justified by assuming the function k_R which maps every object to its equivalence class, as dictated by R. Then $R = \mathsf{ker}\ k_R$ and, for any suitably typed relations X and Y:

$$R \cdot X \subseteq R \cdot Y$$

$\equiv$ $\{$ inline $R = \mathsf{ker}\ k_R$ $\}$

$$\mathsf{ker}\ k_R \cdot X \subseteq \mathsf{ker}\ k_R \cdot Y$$

$\equiv$ $\{$ $\mathsf{ker}\ k_R = k_R^\circ \cdot k_R$; shunting (30) $\}$

$$k_R \cdot k_R^\circ \cdot k_R \cdot X \subseteq k_R \cdot Y$$

$\equiv$ $\{$ $f \cdot f^\circ \cdot f = f$ (difunctionality) $\}$

$$k_R \cdot X \subseteq k_R \cdot Y$$

$\equiv$ $\{$ shunting (30) ; $R = \mathsf{ker}\ k_R$ $\}$

$$X \subseteq R \cdot Y$$

$\square$

Proof of (16):

$$X \subseteq R \upharpoonright (\Phi \cdot \top)$$

$\equiv$ $\{$ (37) $\}$

$$X \subseteq R \wedge X \cdot R^\circ \subseteq \Phi \cdot \top$$

$\equiv$ $\{$ (32) ; shunting (31) ; converses $\}$

$$X \subseteq R \wedge X \cdot (! \cdot R)^\circ \subseteq \Phi \cdot !^\circ$$

$\equiv$ $\{$ assume R entire $\}$

$$X \subseteq R \wedge X \cdot !^\circ \subseteq \Phi \cdot !^\circ$$

$$
\begin{array}{ll}
\equiv & \quad \{ \text{ shunting (31) ; (32) } \} \\
& X \subseteq R \wedge X \subseteq \Phi \cdot \top \\
\equiv & \quad \{ \text{ (34) } \} \\
& X \subseteq \Phi \cdot R
\end{array}
$$

□

Proof that (23) is equivalent to (21), where g abbreviates $(\!|k|\!)$:

$$
\begin{array}{ll}
& h \cdot \mathsf{F}\,(\mathsf{ker}\ g) \ \subseteq\ \mathsf{ker}\ g \cdot h \\
\equiv & \quad \{ \ \mathsf{F}\,(R^\circ) = (\mathsf{F}\,R)^\circ\text{; shunting (30) ; kernel (38) } \} \\
& \mathsf{ker}\ (\mathsf{F}\ g) \ \subseteq\ h^\circ \cdot g^\circ \cdot g \cdot h \\
\equiv & \quad \{ \text{ kernel (38) ; injectivity preorder (39) } \} \\
& g \cdot h \leqslant \mathsf{F}\,g
\end{array}
$$

□

Proof of (35): abbreviating $\mathsf{wp}(f, q)$ by w, $p = \mathsf{wp}(f, q)$ is the same as $p? = w? = f^\circ \cdot q? \cdot f \cap id = \mathsf{dom}\,(q? \cdot f)$, where $\mathsf{dom}\,R$ denotes the *domain* of definition of relation R.
Step (⇒): $f \cdot p? = q? \cdot f$ is stronger than $f \cdot p? \ \subseteq\ q? \cdot f$ which immediately grants $p? \ \subseteq\ w?$. So we only have to ensure $w? \ \subseteq\ p?$:

$$
\begin{array}{ll}
& w? \ \subseteq\ p? \\
\equiv & \quad \{ \ w? = f^\circ \cdot q? \cdot f \cap id \ \} \\
& f^\circ \cdot q? \cdot f \cap id \ \subseteq\ p? \\
\equiv & \quad \{ \ f \cdot p? = q? \cdot f \text{ assumed } \} \\
& f^\circ \cdot f \cdot p? \cap id \ \subseteq\ p? \\
\equiv & \quad \{ \text{ trivia } \} \\
& (f^\circ \cdot f \cap id) \cdot p? \ \subseteq\ p? \\
\Leftarrow & \quad \{ \text{ monotonicity } \} \\
& f^\circ \cdot f \cap id \ \subseteq\ id \\
\equiv & \quad \{ \ R \cap S \ \subseteq\ S \ \} \\
& \mathit{true}
\end{array}
$$

□

Step ($\Leftarrow$): $p? \subseteq w?$ is equivalent to $f \cdot p? \subseteq q? \cdot f$. We are left with:

$$\begin{array}{ll}
& q? \cdot f \subseteq f \cdot p? \Leftarrow p? = w? \\
\equiv & \quad \{ \text{ substitution } \} \\
& q? \cdot f \subseteq f \cdot w? \\
\equiv & \quad \{ \ R \cdot \mathsf{dom}\, R = R \ \} \\
& (q? \cdot f) \cdot \mathsf{dom}\,(q? \cdot f) \subseteq f \cdot w? \\
\equiv & \quad \{ \ w? = \mathsf{dom}\,(q? \cdot f) \ \} \\
& q? \cdot f \cdot w? \subseteq f \cdot w? \\
\Leftarrow & \quad \{ \ q? \subseteq id; \text{ monotonicity } \} \\
& true \\
\Box &
\end{array}$$

Proof of (36):

$$\begin{array}{ll}
& \mathsf{ker}\, f \cdot p? \\
= & \quad \{ \text{ kernel (38) ; (35) since } p = \mathsf{wp}(f, q) \text{ is assumed } \} \\
& f^{\circ} \cdot q? \cdot f \\
= & \quad \{ \text{ converses ; partial identities } \} \\
& (q? \cdot f)^{\circ} \cdot f \\
= & \quad \{ \text{ again (35) ; converses ; kernel (38) } \} \\
& p? \cdot \mathsf{ker}\, f \\
\Box &
\end{array}$$