Programming from Metaphorisms

José Nuno Oliveira[a]

[a] *High Assurance Software Laboratory*
(http://haslab.uminho.pt)
INESC TEC and University of Minho
Braga, Portugal

Abstract

This paper presents a study of the *metaphorism* pattern of relational specification, showing how it 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 studied in this paper is a strategy known as *change of virtual data structure*. By framing metaphorisms in the class of (inductive) *regular* relations, sufficient conditions are given for such implementations to be calculated using relation algebra.

The strategy is illustrated with examples including the derivation of the *quicksort* and *mergesort* algorithms, showing what they have in common and what makes them different from the very start of development.

Keywords: Programming from specifications, Algebra of programming, Weakest precondition calculus.

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

(attributed to Mark Twain)

1. Context

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?

A synergy between metaphors in cognitive linguistics and some relational patterns common in the field of formal specification, termed *metaphorisms*, was suggested in our earlier conference paper [1], which the current paper extends by framing the approach into the study of the wider class of inductive *regular* *relations* [2]. In particular, an algebra useful for reasoning about such specification patterns is developed, whose ubiquity is already observed by Jaoua et al. [2]:

> *We have found regular relations to be very general; in particular, [...] most [...] specifications we encounter in practice are regular.*

Metaphorisms are regular relations represented by symmetric divisions of inductive functions (aka. *folds* or *catamorphisms*) restricted by other regular relations expressing some kind of optimization. After introducing the metaphor and metaphorism concepts and their underlying algebra, this paper presents a generic process of implementing metaphorisms towards *divide & conquer* program strategies based on implicit, *virtual* data structures.

Related work. This paper follows our previous line of research [3] in investigating relational specification patterns which involve the *shrinking* combinator for controlling vagueness and non-determinism. It also relates to the work on representation changers [4] and on the relational algebra of programming in general [5, 6]. Our calculation of sufficient conditions for implementing metaphorisms via change of virtual data-structure, illustrated with the *quicksort* and *mergesort* algorithms, can be regarded as a generalization and expansion of the derivation of *quicksort* by Bird and de Moor [5], where it is given in a rather brief and terse style.

Interest in so-called *regular* relations dates back to at least the work by Riguet in the late 1940's [7]. Their use as specification devices was pioneered by Jaoua et al. [2] and Mili et al. [6] in the early 1990's. Shortly afterwards, Hutton's PhD thesis [8] presents a number of program derivations in which such relations are in evidence. Rectangular relations, a special case of regular relations, have also been studied in [7, 9]. Interestingly, Jaoua et al. [2] already acknowledge that *it is common for specifications to be written as the intersection of an equivalence relation with a rectangular relation*, which is precisely the specification pattern at focus in the current paper.

Metaphorisms can also be regarded as relational generalizations of so-called *metamorphisms* [10, 11]. *Virtual data structures* have been studied mainly from the perspective of *deforestation* [12, 13]. Their role in structuring *divide and conquer* algorithms is commonly accepted but less worked out in a formal context.[1] Sorting is addressed from this perspective in Bird & de Moor's textbook [5], which also stresses algorithm classification through synthesis in the spirit of [14]. In the same vein, the role of intermediate, virtual types in classifying and cataloguing specifications in software repositories has also been emphasized [15].

[1] In the words of Swierstra and de Moor [12] *"virtual data structures (...) play the role of a catalyst in the development of programs, in the sense that in the final program they have been transformed away"*.

2. Introduction

Programming theory has been structured around concepts such as *syntax*, *semantics*, *generative grammar* and so on, that 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. the 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 [16]. 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 dirty*, the same that would enable one to say that somebody might need to *clean his/her reputation*, for instance.

In his *Philosophy of Rhetoric* [17], 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. soiling). The *flow of meaning* is from vehicle to tenor, through the (as a rule left unspecified) common attribute. A sketchy characterization of this construction in the form of a "cospan"

$$\mathsf{T} \xrightarrow{\;g\;} A \xleftarrow{\;f\;} \mathsf{V} \qquad (1)$$

is given in [18]. Functions $f : \mathsf{V} \rightarrow A$ and $g : \mathsf{T} \rightarrow A$, the "witnesses" of the metaphor, extract 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[2] $\mathsf{T} \xleftarrow{g^\circ \cdot f} \mathsf{V}$ 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.

A wide class of optimization criteria can be characterized by so called *regular*, or *rational* relations.[3] First, some intuition about what *regularity* means in this context: a regular relation is such that, wherever two inputs have a common

[2]Given a binary relation R, writing $b\ R\ a$ (read: "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 the *passive voice*; compare e.g. *John loves Mary* with *Mary is loved by John*: $(loves)^\circ = (is\ loved\ by)$.

[3]Also called *difunctional* or *uniform* — see e.g. [7, 2, 8, 5].

image, then they have *exactly the same* set of images. In other words, the image sets of two different inputs are either disjoint or the same. As a counterexample, take following relation, represented as matrix with inputs taken from set $\{a_1, .., a_5\}$ and outputs delivered into set $\{b_1, .., b_5\}$:

R	a_1	a_2	a_3	a_4	a_5
b_1	0	0	1	0	1
b_2	0	0	0	0	0
b_3	0	1	0	0	0
b_4	0	1	0	1	0
b_5	0	0	0	1	0

(2)

Concerning inputs a_3 and a_5, regularity holds; but sets $\{b_3, b_4\}$ and $\{b_4, b_5\}$ — the images of a_2 and a_4, respectively — are neither disjoint nor the same: so R isn't regular. (It will become so if e.g. b_4 is dropped from both image sets or one of b_3 or b_5 is replaced for the other in the corresponding image set.)

These relations are also called *rational* because they can be represented by "fractions" of the form $\frac{f}{g}$, where f and g are functions and notation $\frac{R}{S}$ expresses the so-called *symmetric division* [19, 20] of two relations R and S. As detailed in the sequel, it can be easily shown that $g^\circ \cdot f = \frac{f}{g}$, meaning that metaphors (1) are rational relations.

This paper is organized in two main parts. In the first part we develop an *algebra of metaphors* expressed as rational relations and address the combination of metaphors with another class of rational relations (called *rectangular*) used to express requirements on the "tenor" ("output") side of metaphors. The second part focusses on metaphors $\frac{f}{g}$ where V and T are inductive (recursive) types and f and g are morphisms which extract a common view of such types. That is, f and g become *catamorphisms* [5], also known as *folds* [21].

We use the word *metaphorism* [1] to refer to the specification pattern just described. 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.[4] Other examples could have been given:

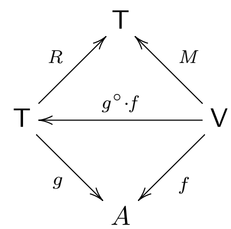
- Change of base of numeric representation — the number represented in the source is the same represented by the result, cf. the 'representation changers' studied by Hutton and Meijer [4].
- Source code refactoring — the meaning of the source program is preserved, the target code being better styled wrt. coding conventions and best practices.
- Gaussian elimination — it transforms a system of linear equations into a triangular system that has the same set of roots.

[4] 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.

- 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 $g^\circ \cdot f$ 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 that "shrinks" $g^\circ \cdot f$,

$$M = (g^\circ \cdot f) \upharpoonright R \qquad (3)$$

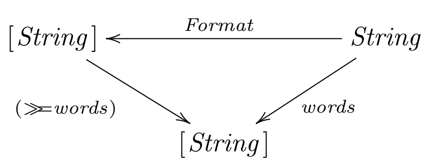


using the "shrinking" operator proposed by Mu and Oliveira [3] for reducing non-determinism. By unfolding the meaning of this relational operator, the relationship (3) established by M is the following:

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

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

A *metaphorism* $M = (g^\circ \cdot f) \upharpoonright R$ therefore involves two functions and an optimization criterion. In the text formatting metaphorism, for instance,



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. (Monadic function ≫=*words* promotes *words* from strings to lists of strings.[5]) Formatting consists in (re)introducing white space evenly throughout the output text lines. For economy of presentation, the diagram omits the optimization part in

$$Format = ((\gg\!= words)^\circ \cdot words) \upharpoonright R \qquad (4)$$

where relation $R : [String] \leftarrow [String]$ should capture the intended formatting criterion on lines of text, e.g. evenly spaced lines better than unevenly spaced ones, and so on.

Formally, nothing precludes f and g from being the same attribute function, in which case types V and T are also the same. Although less interesting from a

[5]Technically, (≫=*words*) is termed the *Kleisli lifting* (or *extension*) of function *words* [22].

strictly (cognitive) metaphorical perspective, metaphorisms of this instance of (3) are very common in programming — take *sorting* as example, where V and T are inhabited by finite sequences of the same (ordered) 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.[6] Because such a tree is not visible from outside, some authors refer to it as a *virtual* data structure [12].

Contribution. This paper addresses a generic process of implementing metaphorisms that introduces *divide & conquer* strategies through implicit, virtual data structures. In particular, it

- introduces the relational notions of *metaphor* and *metaphorism* and develops their algebra based on rational relations, including *divide & conquer* factorization laws (Sections 4 and 5);
- gives results for implementing metaphorisms as hylomorphisms [5] (Sections 6 and 7), of which two examples are given: *quicksort* (Sect. 8) and *mergesort* (Sect. 9).

The paper also includes Sections 11 and 12, which conclude and discuss future work, respectively. Proofs of some auxiliary results are given in Appendix A.

3. Relation algebra preliminaries

Functions. A function $f : X \to Y$ is a special case of a relation, such that $y\ f\ x \;\Leftrightarrow\; y = f\ x$.[7] The equality sign forces f to be totally defined and deterministic. (We read $y = f\ x$ saying "y is *the* result — not *a* result — of applying f to x".) This makes (total) functions quite rich in relational algebra. For instance, any function f satisfies not only the *shunting rules*

$$f \cdot R \subseteq S \quad\Leftrightarrow\quad R \subseteq f^\circ \cdot S \tag{5}$$

$$R \cdot f^\circ \subseteq S \quad\Leftrightarrow\quad R \subseteq S \cdot f \tag{6}$$

where R, S are arbitrary (suitably typed) binary relations, but also

$$b(g^\circ \cdot R \cdot f)a \qquad\Leftrightarrow\qquad (g\ b)R(f\ a) \tag{7}$$

[6]Similar patterns can be found in other *divide & conquer* algorithms.

[7]Following a widespread convention, functions (i.e. total and deterministic relations) will be denoted by lowercase characters (e.g. f, g) or identifiers starting with a lowercase characters, while uppercase letters are reserved to arbitrary relations.

In order to save parentheses in relational expressions, we adopt the following precedence rules: (a) unary operators take precedence over binary ones ; (b) composition binds tighter than any other binary operator; (c) intersection binds tighter than union; (d) division binds tighter than intersection.

a rule which helps moving variables outwards in expressions. For R the identity function $id\ x = x$, (7) instantiates to $b(g^\circ \cdot f)a \ \Leftrightarrow \ g\ b = f\ a$, that is, to metaphor (1).

Given $k \in K$, $\underline{k} : X \to K$ denotes the polymorphic *constant* function which always yields k as result: $\underline{k} \cdot f = \underline{k}$, for every f. Predicates are functions of type $X \to \mathbb{B}$, where $\mathbb{B} = \{\mathrm{T}, \mathrm{F}\}$ is the set of truth values. The constant predicates

$$true = \underline{\mathrm{T}} \quad , \quad false = \underline{\mathrm{F}} \tag{8}$$

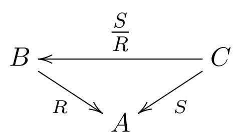
are used in the sequel. Notation

$$! : X \to 1 \tag{9}$$

is chosen to describe the unique (constant) function of its type, where 1 denotes the singleton type.

Symmetric division. Given two arbitrary relations R and S typed as in the diagram below, define the *symmetric division* $\frac{S}{R}$ [20] of S by R by:

$$b\ \frac{S}{R}\ c \ \Leftrightarrow \ \langle \forall\ a\ ::\ a\ R\ b \ \Leftrightarrow \ a\ S\ c \rangle \tag{10}$$



That is, $b\ \frac{S}{R}\ c$ means that b and c are related to exactly the same outputs (in A) by R and by S. Another way of writing (10) is $b\ \frac{S}{R}\ c \Leftrightarrow \{a \mid a\ R\ b\} = \{a \mid a\ S\ c\}$ which is the same as

$$b\ \frac{S}{R}\ c \quad \Leftrightarrow \quad \Lambda R\ b = \Lambda S\ c \tag{11}$$

where Λ is the *power transpose* [5] operator which maps a relation $Q : Y \leftarrow X$ to the set valued function $\Lambda Q : X \to \mathsf{P}\ Y$ such that $\Lambda Q\ x = \{y \mid y\ Q\ x\}$. Another way to define $\frac{S}{R}$ is [20]

$$\frac{S}{R} \quad = \quad R \setminus S \cap R^\circ / S^\circ \tag{12}$$

which factors symmetric division into the two asymmetric divisions $R \setminus S$ and R / S which can be defined by Galois connections:

$$R \cdot X \subseteq S \ \Leftrightarrow \ X \subseteq R \setminus S \tag{13}$$

$$X \cdot R \subseteq S \ \Leftrightarrow \ X \subseteq S / R \tag{14}$$

Pointwise, $b\ (P\ /\ Q)\ a$ means $\forall\ x\ :\ a\ Q\ x :\ b\ P\ x$ (right division) and $b\ (P \setminus Q)\ a$ means $\forall\ x\ :\ x\ P\ b :\ x\ Q\ a$ (left division). Note that, by (13, 14), (12) is equivalent to the universal property:

$$X \subseteq \frac{S}{R} \quad \Leftrightarrow \quad R \cdot X \subseteq S \wedge S \cdot X^\circ \subseteq R \tag{15}$$

From the definitions above a number of standard properties arise [20]:

$$\left(\frac{S}{R}\right)^\circ = \frac{R}{S} \tag{16}$$

$$\frac{S}{R} \cdot \frac{Q}{S} \subseteq \frac{Q}{R} \tag{17}$$

$$f^\circ \cdot \frac{S}{R} \cdot g = \frac{S \cdot g}{R \cdot f} \tag{18}$$

$$id \subseteq \frac{R}{R} \tag{19}$$

Thus $\frac{R}{R}$ is always an *equivalence relation*, for any given R. Furthermore,

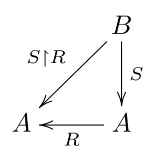
$$R = \frac{R}{R} \quad \Leftrightarrow \quad R \text{ is an equivalence relation} \tag{20}$$

holds.[8] Finally note that, even in the case of functions, (17) remains an inclusion:

$$\frac{f}{g} \cdot \frac{h}{f} \subseteq \frac{h}{g} \tag{21}$$

Relation shrinking [3]. Given relations $S : A \leftarrow B$ and $R : A \leftarrow A$, define $S \upharpoonright R : A \leftarrow B$, pronounced "$S$ shrunk by R", by

$$X \subseteq S \upharpoonright R \quad \Leftrightarrow \quad X \subseteq S \ \wedge \ X \cdot S^\circ \subseteq R \quad \text{cf. diagram:} \tag{22}$$



This states that $S \upharpoonright R$ is the largest part of S such that, if it yields an output for an input x, it must be a maximum, with respect to R, among all possible outputs of x by S. By indirect equality, (22) is equivalent to the closed definition:

$$S \upharpoonright R = S \cap R/S^\circ \tag{23}$$

Among the properties of shrinking [3] we single out two *fusion* rules

$$(S \cdot f) \upharpoonright R = (S \upharpoonright R) \cdot f \tag{24}$$

$$(f \cdot S) \upharpoonright R = f \cdot (S \upharpoonright (f^\circ \cdot R \cdot f)) \tag{25}$$

that will prove useful in the sequel. Putting universal properties (15,22) together we get, by indirect equality,

$$\frac{R}{g} = g^\circ \cdot (R \upharpoonright id) \tag{26}$$

$$\frac{f}{R} = (R \upharpoonright id)^\circ \cdot f \tag{27}$$

[8] This is proved by Riguet on page 134 of [7], where the symmetric division $\frac{R}{R}$ is denoted by *noy.*(R), for "noyaux" of R ("noyaux" means "kernel"). For those readers not wishing to delve into the notation of Riguet [7] we give a simple proof of (20) in Appendix A based on the laws of relation division.

capturing a relationship between shrinking and symmetric division: knowing that $R \upharpoonright id$ is nothing but the deterministic fragment of R, we see how the *vagueness* of arbitrary R replacing either f or g in $\frac{f}{g}$ is forced to shrink.

Recursive relations. Later in the paper we shall need a number of standard constructions in relation algebra that are briefly introduced next. (For the many details omitted please see e.g. the textbook by Bird and de Moor [5].)

Let F be a *relator* [23], that is, a mathematical construction such as, for any type A, type $\mathsf{F}\ A$ is defined and for any relation $R : B \leftarrow A$, relation $\mathsf{F}\ R : \mathsf{F}\ B \leftarrow \mathsf{F}\ A$ is defined such that $\mathsf{F}\ id = id$, $\mathsf{F}\ R^\circ = (\mathsf{F}\ R)^\circ$ and $\mathsf{F}\ (R \cdot S) = (\mathsf{F}\ R) \cdot (\mathsf{F}\ S)$.

Any relation $R : A \leftarrow \mathsf{F}\ A$ is said to be a (relational) F-*algebra*. Special cases include functional F-algebras and, among these, those that are isomorphisms. Within these, the so-called *initial* F-algebras, say $\mathsf{in}_\mathsf{F} : \mathsf{T} \leftarrow \mathsf{F}\ \mathsf{T}$, are such that, given any other F-algebra $R : A \leftarrow \mathsf{F}\ A$, there is a unique relation of type $A \leftarrow \mathsf{T}$, usually written $(\!|R|\!)$, such that $(\!|R|\!) \cdot \mathsf{in}_\mathsf{F} = R \cdot \mathsf{F}\ (\!|R|\!)$ holds. Type T (often denoted by μ_F to express its relationship with the base relator F) is also referred to as *initial*. The meaning of such relations $(\!|R|\!)$, usually referred to as *catamorphisms*, or *folds*, is captured by the *universal property*:

$$X = (\!|R|\!) \quad \Leftrightarrow \quad X \cdot \mathsf{in}_\mathsf{F} = R \cdot (\mathsf{F}\ X) \tag{28}$$

The base F captures the recursive pattern of type T (which we write as μ_F). For instance, for T the datatype of finite lists over a given type A one has

$$\begin{cases} \mathsf{F}\ X = 1 + A \times X \\ \mathsf{F}\ f = id + id \times f \end{cases} \tag{29}$$

This instance is relevant for the examples that come later in this paper.

Given F-algebras $R : A \leftarrow \mathsf{F}\ A$ and $S : B \leftarrow \mathsf{F}\ B$, the composition $H = (\!|R|\!) \cdot (\!|S|\!)^\circ$, of type $A \leftarrow B$, is usually referred to as a *hylomorphism* [5]. H is the least fixpoint of the relational equation $X = R \cdot (\mathsf{F}\ X) \cdot S^\circ$. The intermediate type μ_F generated by $(\!|S|\!)^\circ$ and consumed by $(\!|R|\!)$ is known as the *virtual data structure* [12] of the hylomorphism. The opposite composition $(\!|S|\!)^\circ \cdot (\!|R|\!)$, for suitably typed S and R, is sometimes termed a *metamorphism* [11].

Two properties stem from (28) that prove particularly useful in calculations about $(\!|R|\!)$, namely *fusion*

$$S \cdot (\!|R|\!) = (\!|Q|\!) \quad \Leftarrow \quad S \cdot R = Q \cdot \mathsf{F}\ S \tag{30}$$

and *cancellation* (cf. above):

$$(\!|R|\!) \cdot \mathsf{in}_\mathsf{F} \quad = \quad R \cdot \mathsf{F}\ (\!|R|\!) \tag{31}$$

Fusion is particularly helpful in the sense of finding a sufficient condition on S, R and Q for merging $S \cdot (\!|R|\!)$ into $(\!|Q|\!)$. In the words of Bird and de Moor [5], law (30) *is probably the most useful tool in the arsenal of techniques for program derivation.* The remainder of this paper will give further evidence of this statement.

4. On the algebra of metaphors

Metaphors as symmetric divisions. Substituting $S, R := f, g$ in (15) and using the shunting rules (5,6) we obtain, by indirect equality:

$$\frac{f}{g} \quad = \quad g^{\circ} \cdot f \tag{32}$$

So, a metaphor $g^{\circ} \cdot f$ (3) can be expressed as a symmetric division. On the other hand, moving the variables of (11) outwards by use of (7), we obtain the following *power transpose cancellation* rule:[9]

$$\frac{\Lambda S}{\Lambda R} \quad = \quad \frac{S}{R} \tag{33}$$

Read from right to left, this shows a way of converting arbitrary symmetric divisions into metaphors.

Hereafter we will adopt $\frac{f}{g}$ as our canonical notation for metaphors. This has the advantage of suggesting an analogy with *rational numbers* [10] which makes calculation rules easy to understand and memorize. From (16) we immediately get that *converses of metaphors* are metaphors:[11]

$$\left(\frac{f}{g}\right)^{\circ} = \frac{g}{f} \tag{34}$$

Moreover, $\frac{f}{id} = f$ and $g^{\circ} = \frac{id}{g}$, consistent with id being the unit of composition, $R \cdot id = R = id \cdot R$. As expected,

$$\frac{id}{g} \cdot \frac{f}{id} = \frac{f}{g}$$

holds, a corollary of the more general:[12]

$$\frac{id}{g} \cdot \frac{h}{k} \cdot \frac{f}{id} = \frac{h \cdot f}{k \cdot g} \tag{35}$$

If id plays the role of the multiplicative identity 1 in the rational number analogy, what is the counterpart of number 0? It is the empty relation $\perp$, which is represented by any $\frac{f}{g}$ such that $g^{\circ} \cdot f$ is empty, that is, $g\ y \neq f\ x$ for any choice of x and y. (Any two relations R and S such that $R^{\circ} \cdot S \subseteq \perp$ are

[9] This rule is nothing but another way of stating exercise 4.48 proposed by Bird and de Moor [5]. Note that ΛR is always a (total) function.

[10] This analogy was first noted by Jaoua et al. [2] where functions, rational relations and arbitrary relations are paralleled with integers, rationals and reals, respectively.

[11] As (the perception of) time predates money in human evolution, it is reasonable to guess that metaphor TIME IS MONEY might have started the other way around, by its *converse* MONEY IS TIME, although this is highly speculative of course.

[12] Equality (35) can be regarded as a generalization of Proposition 4.5 given by Jaoua et al. [2].

said to be *range disjoint.*) For instance, $\frac{true}{false} = \bot$, where *true* and *false* are the constant functions yielding the corresponding truth values (8). In general,

$$a \neq b \;\Leftrightarrow\; \frac{\underline{a}}{\underline{b}} = \bot$$

where $\underline{a}$ and $\underline{b}$ denote constant functions. The opposite situation of $a = b$ above leads to $\frac{\underline{a}}{\underline{a}} = \top$, where $y \top x$ holds for all y, x. The canonical presentation of this largest possible metaphor is

$$\frac{!_A}{!_B} = \top_{B \leftarrow A} \tag{36}$$

recall (9).

Intersecting metaphors. In the words of C.S. Peirce, $y \top x$ simply means that "y is coexistent with x" [24]. Suggestively, the symbol chosen by Peirce to denote $\top$ is ∞. Although semantically poor, this metaphor surely holds about any x and y related by any other metaphor. In a sense, it can be regarded as the starting point for any metaphorical relationship, obtained by some form of refinement. Metaphor conjunction is one way of doing such refinement,

$$\frac{f}{g} \cap \frac{h}{k} = \frac{f \vartriangle h}{g \vartriangle k} \tag{37}$$

where the *pairing* of two functions, say $f \vartriangle h$, is defined by $(f \vartriangle h)\ x = (f\ x, h\ x)$.[13] As an example of the intersection rule consider

$$\frac{true}{q} \cap \frac{p}{true} = \frac{true \vartriangle p}{q \vartriangle true}$$

where p and q are predicates and *true* is the everywhere true predicate already introduced. It is easy to show that $y\ \frac{true}{q}\ x = q\ y$ and $y\ \frac{p}{true}\ x = p\ x$ hold; so the intersection should mean $q\ y \wedge p\ x$. In fact:

$$
\begin{array}{ll}
 & y\ \dfrac{true \vartriangle p}{q \vartriangle true}\ x \\
\Leftrightarrow & \quad \{ \text{ pointwise meaning of } \frac{f}{g} \text{ and } f \vartriangle g \ \} \\
 & (q\ y, \mathrm{T}) = (\mathrm{T}, p\ x) \\
\Leftrightarrow & \quad \{ \text{ equality of pairs ; predicate logic } \} \\
 & q\ y \wedge p\ x
\end{array}
$$

□

[13] The fact that metaphors are preserved by intersection, captured by (37), follows immediately from a more general law of relation algebra [5]: $(R \vartriangle S)^\circ \cdot (P \vartriangle Q) = R^\circ \cdot P \cap S^\circ\ Q$, where pairing is extended to arbitrary relations in the expected way: $(y, z)\ (R \vartriangle S)\ x \;\Leftrightarrow\; (y\ R\ x) \wedge (z\ S\ x)$. Note how these laws include what are normally regarded as the two key benefits of the calculus of relations: *converse functions as specifications and intersection of specifications* [25, 26] .

We will focus on the particular metaphoric pattern $\frac{f}{g} \cap \frac{true}{p} = \frac{f^{\vartriangle} true}{g^{\vartriangle} p}$ later in this paper:

$$y\,(\frac{f}{g} \cap \frac{true}{p})\,x \quad \Leftrightarrow \quad (g\;y = f\;x) \;\wedge\; p\;y$$

In this relational specification pattern, outputs y preserve some common information wrt. inputs x with the additional ingredient of satisfying post-condition p.

Rectangular metaphors. A relation R is said to be *rectangular* iff $R = R \cdot \top \cdot R$ holds [7, 9]. Note that $R \subseteq R \cdot \top \cdot R$ always holds: $b\;R\;a$ implies that there exist a', b' such that $b\;R\;a'$ and $b'\;R\;a$. Metaphors of the form $\frac{\underline{a}}{f}$ (meaning $f\;x = a$ for some given a) are rectangular, as the following calculation shows:

$$\frac{\underline{a}}{f} \cdot \top \cdot \frac{\underline{a}}{f} = \frac{\underline{a}}{f}$$

$\Leftrightarrow$ { since $R \subseteq R \cdot \top \cdot R$ always holds }

$$\frac{\underline{a}}{f} \cdot \top \cdot \frac{\underline{a}}{f} \subseteq \frac{\underline{a}}{f}$$

$\Leftarrow$ { monotonicity of composition (by f°) }

$$\underline{a} \cdot \top \cdot \frac{\underline{a}}{f} \subseteq \underline{a}$$

$\Leftrightarrow$ { shunting (5) ; $\frac{\underline{a}}{\underline{a}} = \top$ }

$$\top \cdot \frac{\underline{a}}{f} \subseteq \top$$

$\Leftrightarrow$ { any relation is at most $\top$ }

$$true$$

$\Box$

As rectangularity is preserved by converse, $\frac{f}{\underline{a}}$ is also rectangular.

Kernel metaphors. In keeping with the analogy between fractions of integers and *fractions of functions* one might wish the equality $\frac{f}{f} = id$ to hold, but this only happens for f injective.[14] As seen in Sect. 3, metaphor $\frac{f}{f}$ is an *equivalence relation* and therefore reflexive:

$$id \subseteq \frac{f}{f} \tag{38}$$

[14] Morphisms such that $\frac{S}{S} = id$ are referred to as *straight* by Freyd and Scedrov [20] and generically underlie the proof strategy known as *indirect equality* [5].

It is known as the *kernel* of f and it "measures" the injectivity of f, as defined by the preorder

$$f \leqslant g \;\Leftrightarrow\; \frac{g}{g} \subseteq \frac{f}{f} \tag{39}$$

where $f \leqslant g$ means that f is *less injective* than g.[15] Clearly, $! \leqslant f \leqslant id$ for any f (36,38). The following alternative way of stating (39)

$$f \leqslant g \quad\Leftrightarrow\quad \langle \exists\, k \;::\; f = k \cdot g \rangle \tag{40}$$

is given by Gibbons [21].

Every equivalence relation $A \xleftarrow{R} A$ is representable by a kernel metaphor, and canonically by

$$R = \frac{\Lambda R}{\Lambda R} \tag{41}$$

where the power transpose ΛR maps each element of A into its *equivalence class.* (41) follows immediately from (20) and (33).

Weakest preconditions. Given a predicate p, we define the metaphor $p?$ by

$$p? = id \cap \frac{true}{p} \tag{42}$$

This is called the *partial identity*[16] for p in the sense that

$$y\;(p?)\;x \;\Leftrightarrow\; (p\;y) \wedge y = x$$

holds. That is, $p?$ is the fragment of id where p holds. Note that $p? = id \cap \frac{p}{true}$ by converses. The rectangular metaphor $\frac{true}{p}$ can be recovered from $p?$ by

$$p? \cdot \top = \frac{true}{p} \tag{43}$$

(Conversely, $\top \cdot p? = \frac{p}{true}$.) It is easy to show that

$$f \cap \frac{true}{q} = q? \cdot f$$
$$f \cap \frac{p}{true} = f \cdot p?$$

hold.[17] Thus, q (resp. p) work as *post* (resp. *pre*) conditions for function f. The particular situation in which $q? \cdot f = f \cdot p?$ holds captures a *weakest/strongest* pre/post-condition relationship expressed by the following universal property:

$$f \cdot p? = q? \cdot f \quad\Leftrightarrow\quad p = q \cdot f \tag{44}$$

---

[15] See e.g. [27, 21]. This injectivity preorder is the converse of the *determination order* of [28].

[16] *Partial identities* are also known as *coreflexives*, *monotypes* or *tests* [29, 20, 30].

[17] Check (A.3) and (A.4) in Appendix A.

Condition $p = q \cdot f$ is equivalent to $p = wp\ (f, q)$, the weakest precondition (WP) for the outputs of f to fall within q. Property (44) enables a "logic-free" calculation of weakest preconditions, as we shall soon see: given f and post-condition q, there exists a unique (weakest) precondition p such that $q? \cdot f$ can be replaced by $f \cdot p?$. Moreover:

$$\frac{f}{f} \cdot p? = p? \cdot \frac{f}{f} \quad \Leftarrow \quad p \leqslant f \tag{45}$$

where $\leqslant$ denotes the injectivity preorder on functions (39,40). Relational proofs for (44) and (45) are given in Appendix A.

Products of metaphors. Metaphors can also be combined pairwise, leading to metaphors on pairs.[18] This situation is captured by the *product rule*,

$$\frac{f}{g} \times \frac{h}{k} = \frac{f \times h}{g \times k} \tag{46}$$

telling that the product of two metaphors is a metaphor. Relational (Kronecker) product is defined as expected, $(x, y)\ (R \times S)\ (a, b) \Leftrightarrow x\ R\ a \wedge y\ S\ b$, which, in the case of functions, becomes $(f \times g)\ (a, b) = (f\ a, g\ b)$. Both pairing and product can be written pointfree,

$$R \times S = R \cdot \pi_1 \vartriangle S \cdot \pi_2 \tag{47}$$

$$R \vartriangle S = \pi_1^\circ \cdot R \cap \pi_2^\circ \cdot S \tag{48}$$

where $\pi_1\ (a, b) = a$ and $\pi_2\ (a, b) = b$ are the standard projections. The proof of (46) follows:

$$\begin{array}{cl} & \frac{f}{g} \times \frac{h}{k} \\ = & \{\ (47)\ ;\ (35)\ \} \\ & \frac{f \cdot \pi_1}{g} \vartriangle \frac{h \cdot \pi_2}{k} \\ = & \{\ (48)\ ;\ (35)\ \} \\ & \frac{f \cdot \pi_1}{g \cdot \pi_1} \cap \frac{h \cdot \pi_2}{k \cdot \pi_2} \\ = & \{\ (37)\ \} \\ & \frac{f \cdot \pi_1 \vartriangle g \cdot \pi_1}{h \cdot \pi_2 \vartriangle k \cdot \pi_2} \end{array}$$

[18] By referring to the *quixotic* plot of a couple of politicians in some particular situation, one might wish to suggest that one of them behaved like Don Quixote *and* the other like Sancho Panza.

$=$ { (47) twice }

$$\frac{f \times g}{h \times k}$$

□

Functorial metaphors. Metaphor product rule (46) can be regarded as an instance of a more general result: any relator F [19] distributes over a metaphor $\frac{f}{g}$:

$$\mathsf{F}\,\frac{f}{g} = \frac{\mathsf{F}\,f}{\mathsf{F}\,g} \tag{49}$$

This result follows immediately from standard properties of relators, $\mathsf{F}\,(R \cdot S) = (\mathsf{F}\,R) \cdot (\mathsf{F}\,S)$ and $\mathsf{F}\,(R^\circ) = (\mathsf{F}\,R)^\circ$. Rule (46) corresponds to $\mathsf{F}\,(R, S) = R \times S$, where F is binary. Thus

$$\frac{f}{g} + \frac{h}{k} = \frac{f + h}{g + k} \tag{50}$$

also holds, where direct sum $R + S$ is the same as $[i_1 \cdot R\,, i_2 \cdot S]$, where i_1 and i_2 are the standard *injections* associated to a datatype sum, $A \xrightarrow{i_1} A + B \xleftarrow{i_2} B$ and

$$C \xleftarrow{[R\,,S]} A + B \tag{51}$$

denotes the junction $R \cdot i_1^\circ \cup R \cdot i_2^\circ$ of relations $C \xleftarrow{R} A$ and $C \xleftarrow{S} B$. By (40), one has

$$[f\,, g] \leqslant f + g \tag{52}$$

since $[f\,, g] = [id, id] \cdot (f + g)$ by coproduct laws. Moreover,

$$f + g \leqslant h + k \;\Leftrightarrow\; f \leqslant h \wedge g \leqslant k \tag{53}$$

holds by coproduct laws too, since $\frac{h+k}{h+k} \subseteq \frac{f+g}{f+g}$ is equivalent to $\frac{h}{h} + \frac{k}{k} \subseteq \frac{f}{f} + \frac{g}{g}$ by (50).

Difunctionality and uniformity. A relation R is said to be *difunctional* [7, 9] or *regular* [2] wherever $R \cdot R^\circ \cdot R = R$ holds, which amounts to $R \cdot R^\circ \cdot R \subseteq R$ since the converse inclusion always holds.

Metaphors are difunctional because every symmetric division is so, as is easy to check by application of laws (17) and (16). The fact that every function f is difunctional can be expressed by $f \cdot \frac{f}{f} = f$.

[19] Recall from Sect. 3 that a relator F is an (endo)functor F that preserves converses.

A relation R is said to be *uniform* [2] if and only if $\Lambda R \leqslant R$, where preorder (39) is extended to arbitrary relations

$$R \leqslant S \quad \Leftrightarrow \quad S^\circ \cdot S \subseteq R^\circ \cdot R \tag{54}$$

as in [27]. Metaphors are uniform relations, because a relation is uniform iff it is difunctional (regular), as the following calculation shows:

$$
\begin{array}{ll}
 & R \text{ is uniform} \\
\Leftrightarrow & \quad \{ \text{ definition above } \} \\
 & \Lambda R \leqslant R \\
\Leftrightarrow & \quad \{ \text{ (54) ; (32) } \} \\
 & R^\circ \cdot R \subseteq \dfrac{\Lambda R}{\Lambda R} \\
\Leftrightarrow & \quad \{ \ \Lambda \text{ cancellation (33) } \} \\
 & R^\circ \cdot R \subseteq \dfrac{R}{R} \\
\Leftrightarrow & \quad \{ \text{ universal property (15) of symmetric division } \} \\
 & R \cdot R^\circ \cdot R \subseteq R \\
\Leftrightarrow & \quad \{ \text{ definition above } \} \\
 & R \text{ is difunctional}
\end{array}
$$

$\Box$

Note how step $R^\circ \cdot R \subseteq \frac{R}{R}$ above captures the intuition about a regular (i.e. uniform, difunctional) relation R, as given in the introduction, recall (2): $a_1\ (R^\circ \cdot R)\ a_2$ tells that a_1 and a_2 have *some* common image; $a_1\ \frac{R}{R}\ a_2$ tells that they have *exactly the same* image sets.

Functional metaphors. When is a metaphor $\frac{f}{g}$ a function? Shunting rules (5,6) are equivalent to saying that f is total (or *entire*) — $id \subseteq f^\circ f$, which we have already seen in fraction notation (38) — and deterministic (or *simple*) — $f \cdot f^\circ \subseteq id$. We shall use notation $\rho f = f \cdot f^\circ$ for the *range* (of output values) of f.

By (5,6), checking the totality of $\frac{f}{g}$ — $id \subseteq \frac{g}{f} \cdot \frac{f}{g}$ — amounts to $\rho f \subseteq \rho g$: the attribute value of any given vehicle is the attribute value of some tenor. In case g is surjective ($\rho g = id$), $\frac{f}{g}$ is total for any f. For determinism to hold, $\rho \frac{f}{g} = \frac{f}{g} \cdot \frac{g}{f} \subseteq id$, rule (21) offers the sufficient condition $\frac{g}{g} = id$, that is, g injective suffices.

For total metaphors, the inclusion $h \subseteq \frac{f}{g}$ has at least a functional solution h, which can be calculated using the rule

$$h \subseteq \frac{f}{g} \;\Leftrightarrow\; g \cdot h = f \tag{55}$$

that relies on the useful law of *function equality*

$$f \subseteq g \Leftrightarrow f = g \Leftrightarrow f \supseteq g \tag{56}$$

itself a follow-up of shunting rules (5,6).

5. Divide & conquer metaphors

"Shrinking" metaphors. Thus far we have not taken into account the *shrinking* part of (3), which we now write using fraction notation:

$$M = \frac{f}{g} \upharpoonright R \tag{57}$$

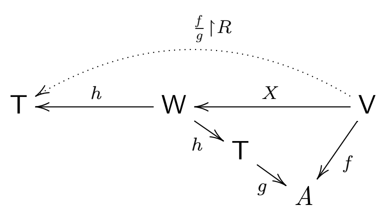
By law (24) one gets:

$$\frac{f}{g} \upharpoonright R = (\frac{id}{g} \upharpoonright R) \cdot f$$

Below we will show that this equality is an instance of a more general result that underlies more elaborate metaphor transformations that prove useful in the sequel. The main idea of such transformations is to split a $\mathsf{T} \longleftarrow \mathsf{V}$ metaphor in two parts mediated by an intermediate type, say W in

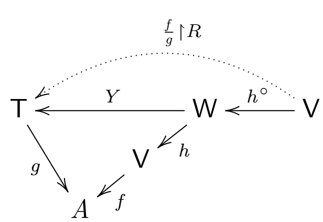
$$\mathsf{T} \longleftarrow \mathsf{W} \longleftarrow \mathsf{V}$$

which is intended to gain control of the "pipeline". This can be done in two ways. Suppose there is a surjection $h : \mathsf{W} \to \mathsf{T}$ onto the tenor side, that is, $\rho\, h = h \cdot h^\circ = id$. Then the splitting can be expressed as in the following diagram



(58)

provided one can find a relation X such that $h \cdot X = \frac{f}{g} \upharpoonright R$. Alternatively, we can imagine *surjection* h onto the vehicle side, say $h : \mathsf{W} \to \mathsf{V}$ in



(59)

and try and find relation Y such that $Y \cdot h^\circ = \frac{f}{g} \upharpoonright R$.

Note how intermediate type W is a *representation* of T or V in, respectively, (58) and (59), h acting as a typical data refinement *abstraction* function.[20] Anticipating that the two-stage schemas of (58) and (59) are intended to specify *divide & conquer* implementations of the original metaphor, let us calculate *conquer* step Y in the first place:

$$\frac{f}{g} \upharpoonright R$$

$= \quad \{$ identity of composition $\}$

$$(\frac{f}{g} \upharpoonright R) \cdot id$$

$= \quad \{$ h assumed to be a surjection, $\rho\, h = h \cdot h^\circ = id$ $\}$

$$(\frac{f}{g} \upharpoonright R) \cdot h \cdot h^\circ$$

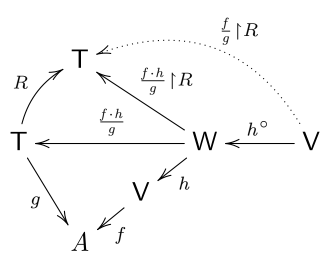
$= \quad \{$ law (24) $\}$

$$\underbrace{(\frac{f \cdot h}{g} \upharpoonright R)}_{Y} \cdot h^\circ$$

Clearly, in this refinement strategy, the optimization of the starting metaphor goes into the *conquer* stage, where it optimizes a richer metaphor between tenor T and W, the new vehicle. *Divide* step h° is just a representation of the original vehicle V into the new vehicle W (59). Altogether:

$$\frac{f}{g} \upharpoonright R \quad = \quad (\frac{f \cdot h}{g} \upharpoonright R) \cdot h^\circ \qquad \text{for } h \text{ surjective} \tag{60}$$

In a diagram, completing (59):



Dually, it is to be expected that the derivation of X in (58) will yield an optimized *divide* step where most of the work goes, running h as *conquer* step

[20] Following the usual terminology [31], by an *abstraction* we mean a *simple* (ie. functional) and *surjective* relation. In this paper all abstractions are total (entire), that is, they are functions. In symbols, α is an abstraction *function* iff $id \subseteq \alpha^\circ \cdot \alpha$ and $id = \alpha \cdot \alpha^\circ$.

to abstract from W, the new tenor, to T, the old tenor. Due to the asymmetry of *shrinking*, the inference of X is less immediate, calling for definition (23):

$$\frac{f}{g} \upharpoonright R$$
$$= \qquad \{ \text{ (23) ; converse of a metaphor (34) } \}$$
$$\frac{f}{g} \cap R \,/\, \frac{g}{f}$$
$$= \qquad \{ \text{ } h \text{ assumed to be a surjection, } \rho\, h = h \cdot h^\circ = id \text{ } \}$$
$$h \cdot h^\circ \cdot (\frac{f}{g} \cap R \,/\, \frac{g}{f})$$
$$= \qquad \{ \text{ injective } h^\circ \text{ distributes over } \cap \text{ ; (18) } \}$$
$$h \cdot (\frac{f}{g \cdot h} \cap h^\circ \cdot R \,/\, \frac{g}{f})$$
$$= \qquad \{ \text{ (14) ; shunting (6) } \}$$
$$h \cdot \underbrace{(\frac{f}{g \cdot h} \cap h^\circ \cdot (R \,/\, g) \cdot f)}_{X}$$

Clearly, the choice of some intermediate w by X tells where the optimization has moved to, as detailed below by rendering X in pointwise notation:

$$\begin{array}{l} w\ X\ v \ \Leftrightarrow \\ \quad \mathbf{let}\ a = f\ v \\ \quad \mathbf{in}\ (g\ (h\ w) = a) \wedge \langle \forall\ t\ :\ a = g\ t :\ (h\ w)\ R\ t \rangle \end{array}$$

In words:

> *Given vehicle v, X will select those w that represent tenors $(h\ w)$ with the same attribute (a) as vehicle v, and that are best among all other tenors t exhibiting the same attribute a.*

Altogether:

$$\frac{f}{g} \upharpoonright R \quad = \quad h \cdot (\frac{f}{g \cdot h} \cap h^\circ \cdot (R \,/\, g) \cdot f) \qquad \text{for } h \text{ surjective} \tag{61}$$

A calculation similar to that showing $\frac{f}{g}$ difunctional above, will show that $R \,/\, g$ being difunctional is sufficient for factor $h^\circ \cdot (R \,/\, g) \cdot f$ in (61) to be so.

Post-conditioned metaphors. Let us finally consider the following pattern of metaphor shrinking

$$\frac{f}{g} \upharpoonright \frac{true}{q} \tag{62}$$

indicating that only the outputs satisfying q are regarded as good enough. That is, q acts as a *post-condition* on $\frac{f}{g}$. An example of (62) is the metaphor

$$Sort = \frac{bag}{bag} \upharpoonright \frac{true}{ordered}$$

where *bag* is the function that extracts the bag (multiset) of elements of a finite list and *ordered* the predicate that checks whether a finite list is ordered according to some predefined criterion. Clearly, $\mathsf{V} = \mathsf{T}$ in this example.

The laws developed above for metaphor shrinking can be instantiated for this pattern and reasoned about. Alternatively, it can easily be shown that (62) reduces to

$$\frac{f}{g} \cap q? \cdot \top \tag{63}$$

provided $\frac{f}{g}$ is entire (total), which is surely the case wherever $f = g$, as we have seen. This follows from this law of the shrinking operator proved in Appendix A:

$$S \upharpoonright (q? \cdot \top) = q? \cdot S \quad \Leftarrow \quad S \text{ is entire} \tag{64}$$

Specifications of the form $\frac{f}{f} \cap q? \cdot \top$ are intersections of an *equivalence relation with a rectangular relation*, a common specification pattern already identified by Jaoua et al. [2]. As intersections of rational relations are rational (regular) relations, pattern $\frac{f}{f} \cap q? \cdot \top$ is rational. By (A.3), (63) further reduces to

$$q? \cdot \frac{f}{g}$$

a pattern to be referred to as a *postconditioned metaphor*. Sorting thus is one such metaphor,

$$Sort = ordered? \cdot Perm \quad \textbf{where} \quad Perm = \frac{bag}{bag} \tag{65}$$

where $y\ Perm\ x$ means that y is a *permutation* of x.

Understandably, the *divide & conquer* versions of a postconditioned metaphor are easier to calculate than in the generic cases above, because one can take advantage of WP laws such as e.g. (44). Corresponding to (61), one gets

$$q? \cdot \frac{f}{g} = h \cdot p? \cdot \frac{f}{g \cdot h} \qquad \text{for } h \text{ surjective and } p = q \cdot h \tag{66}$$

since:

$$q? \cdot id \cdot \frac{f}{g}$$

$$= \qquad \{ \ h \text{ assumed surjective } \}$$

$$q? \cdot h \cdot h^\circ \cdot \frac{f}{g}$$

$$= \quad \{ \text{ switch to WP } p \text{ (44), cf. } q? \cdot h = h \cdot p? \ \}$$

$$h \cdot \underbrace{p? \cdot \frac{f}{g \cdot h}}_{X}$$

The counterpart of (60) is even more immediate:

$$q? \cdot \frac{f}{g} = \underbrace{q? \cdot \frac{f \cdot h}{g}}_{Y} \cdot h^\circ \qquad \text{for } h \text{ surjective} \tag{67}$$

6. Metaphorisms

Thus far, types T, V and W have been left uninterpreted. We want now to address metaphors in which these are inductive (tree-like) types specified by initial algebras, say $\mathsf{T} \xleftarrow{\mathsf{in_F}} \mathsf{F\,T}$, $\mathsf{W} \xleftarrow{\mathsf{in_G}} \mathsf{G\,W}$ and $\mathsf{V} \xleftarrow{\mathsf{in_H}} \mathsf{H\,V}$, assuming such algebras exist for functors F, G and H, respectively. Moreover, f, g and h become folds (catamorphisms) over such initial types, recall Sect. 3. We shall refer to such metaphors involving catamorphisms over inductive types as *metaphorisms* [1].

To facilitate linking each type with its functor, we shall adopt the familiar notation μ_F instead of T, μ_G instead of W and μ_H instead of V. The popular notation $(\!|R|\!)$ will be used to express folds over such types, recall (28). Also useful in the sequel is the fact that inductive predicates can be expressed by folds too, in the form of partial identities:[21]

$$(\!|R|\!) \subseteq id \quad \Leftarrow \quad R \subseteq \mathsf{in_F} \tag{68}$$

Our first example of metaphorism calculation by fusion (30) is the derivation of a simple (functional) *representation changer* [4]:

> *A representation changer is a function that converts a concrete representation of an abstract value into a different concrete representation of that value.*

Metaphorisms of the form $\frac{k \cdot (\!|y|\!)}{(\!|y|\!)}$ are representation changers, in which the change of representation consists in picking an attribute of the vehicle, extracted by $(\!|y|\!)$, changing its value by applying k and then mapping the new attribute value back to the tenor, which in this case is of the same type as the vehicle.

---

[21] Property (28) establishes $(\!|R|\!)$ as the unique fixpoint of the equation $X = R \cdot (\mathsf{F}\ X) \cdot \mathsf{in_F}^\circ$, and therefore the *least prefix point* too: $(\!|R|\!) \subseteq X \Leftarrow R \cdot (\mathsf{F}\ X) \cdot \mathsf{in_F}^\circ \subseteq X$ [5]. From this, one can easily infer (68), that fold is a monotonic operator, etc.

Theorem 1. *Representation changer* $\frac{k \cdot (\!|y|\!)}{(\!|y|\!)}$ *is refined by a functional implementation* $(\!|x|\!)$ *provided, for some* z *such that* $k \cdot y = z \cdot \mathsf{F}\ k$ *holds,* $x \subseteq \frac{z \cdot \mathsf{F}\ (\!|y|\!)}{(\!|y|\!)}$ *also holds. Proof: the proof relies on (double) fusion (30):*

$$
\begin{array}{ll}
& (\!|x|\!) \subseteq \dfrac{k \cdot (\!|y|\!)}{(\!|y|\!)} \\
\Leftrightarrow & \quad \{ \ \textit{(55)} \ \} \\
& (\!|y|\!) \cdot (\!|x|\!) = k \cdot (\!|y|\!) \\
\Leftrightarrow & \quad \{ \ \textit{fuse } k \cdot (\!|y|\!) \textit{ into } (\!|z|\!) \textit{ assuming } k \cdot y = z \cdot \mathsf{F}\ k \ \textit{(30)} \ \} \\
& (\!|y|\!) \cdot (\!|x|\!) = (\!|z|\!) \\
\Leftarrow & \quad \{ \ \textit{fusion (30) again} \ \} \\
& (\!|y|\!) \cdot x = z \cdot \mathsf{F}\ (\!|y|\!) \\
\Leftrightarrow & \quad \{ \ \textit{metaphors (55)} \ \} \\
& x \ \subseteq \ \dfrac{z \cdot \mathsf{F}\ (\!|y|\!)}{(\!|y|\!)}
\end{array}
$$

□

Comparing the top and bottom lines of the calculation above we see that the "banana brackets" of $(\!|x|\!)$ have disappeared. This condition, together with the intermediate assumption $k \cdot y = z \cdot \mathsf{F}\ k$, are sufficient for the refinement to take place.

The example of application of this theorem given below is a quite simple one, its purpose being mainly to illustrate the calculational style which will be followed in the rest of the paper to derive programs from metaphorisms. Let the initial algebra for finite lists be denoted by the familiar

$$\mathsf{in_F} = [\mathsf{nil}, \mathsf{cons}] \tag{69}$$

where $\mathsf{nil}\ \_ = [\,]$ is the constant function which yields the empty list and $\mathsf{cons}\ (a, s) = a : s$ adds a to the front of s. The underlying functor is $\mathsf{F}\ f = id + id \times f$, recall (29). Let $\mathsf{add}\ (x, y) = x + y$ denote natural number addition and $k = (b+)$ be the unary function that adds b to its argument. Define $y = [\mathsf{zero}, \mathsf{add}]$ where zero is the everywhere-0 constant function. So $\mathsf{sum} = (\!|y|\!)$ is the function which sums all elements of a list.

The intended change of representation between a vehicle v and tenor t is specified by $\mathsf{sum}\ t = b + \mathsf{sum}\ v$. Clearly, $(b+) \cdot [\mathsf{zero}, \mathsf{add}] = z \cdot (id + id \times (b+))$ has solution $z = [\underline{b}, \mathsf{add}]$, since $b + 0 = b$ and $b + (h + t) = h + (b + t)$. Knowing z, our aim is to solve $x \subseteq \frac{z \cdot \mathsf{F}\ (\!|y|\!)}{(\!|y|\!)}$ for $x = [x_1, x_2]$, helped by the following law

$$[x, y] \ \subseteq \ \frac{[g, h]}{f} \ \Leftrightarrow \ x \ \subseteq \ \frac{g}{f} \wedge y \ \subseteq \ \frac{h}{f} \tag{70}$$

easy to infer by coproduct and metaphor algebra.

Applied to our example, this yields $x_1 \subseteq \frac{b}{(\![y]\!)}$ and $x_2 \subseteq \frac{\mathsf{add}\cdot(id\times(\![y]\!))}{(\![y]\!)}$, the latter equivalent to $(\![y]\!) \cdot x_2 = \mathsf{add} \cdot (id \times (\![y]\!))$. From this we get $x_2 = \mathsf{cons}$ by cancellation (31). On the other hand, x_1 is necessarily a constant function $\underline{w}$ such that $(\![y]\!)\ w = b$. The simplest choice for w is the singleton list $[\,b\,]$. We therefore obtain the following functional solution for the given metaphor, unfolding $r = (\![x]\!)$ to pointwise notation:

$$\begin{array}{rcl} r\,[\,] & = & [\,b\,] \\ r\,(a:t) & = & a + r\ t \end{array}$$

7. Shrinking metaphorisms into hylomorphisms

This section focusses on metaphorisms that are equivalence relations over inductive data types. Let $\mu\mathsf{F} \xleftarrow{\mathsf{in}_\mathsf{F}} \mathsf{F}\,\mu\mathsf{F}$, and let $A \xleftarrow{k} \mathsf{F}\,A$ be given, so that $A \xleftarrow{(\![k]\!)} \mu\mathsf{F}$. It turns out that not only is $R = \frac{(\![k]\!)}{(\![k]\!)}$ itself a relational fold

$$R = (\![R \cdot \mathsf{in}_\mathsf{F}]\!)$$

of type $\mu\mathsf{F} \longleftarrow \mu\mathsf{F}$, but also it is a *congruence* for the algebra in_F.[22] This follows from the following theorem.

Theorem 2 (F-congruences). *Let R be a congruence for an algebra $h : \mathsf{F}\,A \to A$ of functor F, that is*

$$h \cdot (\mathsf{F}\,R) \subseteq R \cdot h \qquad i.e. \qquad y\ (\mathsf{F}\,R)\ x \Rightarrow (h\ y)\ R\ (h\ x) \tag{71}$$

hold and R is an equivalence relation. Then (71) is equivalent to:

$$R \cdot h = R \cdot h \cdot (\mathsf{F}\,R) \tag{72}$$

For the particular case $h = \mathsf{in}_\mathsf{F}$, (72) is equivalent to:

$$R = (\![R \cdot \mathsf{in}_\mathsf{F}]\!) \tag{73}$$

For R presented as a kernel metaphor $R = \frac{f}{f}$, (71) is also equivalent to

$$f \cdot h \leqslant \mathsf{F}\,f \tag{74}$$

where $\leqslant$ is the injectivity preorder (39). (Proof: see Appendix A.)
□

A standard result in algebraic specification states that if a function f defined on an initial algebra is a fold then $\frac{f}{f}$ is a congruence [32, 21]. Although not

[22] The *Perm* equivalence relation is an example of this, recall (65).

strictly necessary, we give below a proof that frames this result in Theorem 2 by making $R = \frac{\llparenthesis k \rrparenthesis}{\llparenthesis k \rrparenthesis}$ in (73) and calculating:

$$
\begin{array}{cl}
& \dfrac{\llparenthesis k \rrparenthesis}{\llparenthesis k \rrparenthesis} = \llparenthesis \dfrac{\llparenthesis k \rrparenthesis}{\llparenthesis k \rrparenthesis} \cdot \mathsf{in}_{\mathsf{F}} \rrparenthesis \\
\Leftrightarrow & \qquad \{ \text{ universal property (28) ; metaphor algebra (49) } \} \\
& \dfrac{\llparenthesis k \rrparenthesis \cdot \mathsf{in}_{\mathsf{F}}}{\llparenthesis k \rrparenthesis} = \dfrac{\llparenthesis k \rrparenthesis \cdot \mathsf{in}_{\mathsf{F}}}{\llparenthesis k \rrparenthesis} \cdot \dfrac{\mathsf{F}\ \llparenthesis k \rrparenthesis}{\mathsf{F}\ \llparenthesis k \rrparenthesis} \\
\Leftrightarrow & \qquad \{ \text{ cancellation (31) ; } f \cdot \frac{f}{f} = f \ \} \\
& \dfrac{\llparenthesis k \rrparenthesis \cdot \mathsf{in}_{\mathsf{F}}}{\llparenthesis k \rrparenthesis} = \dfrac{k \cdot \mathsf{F}\ \llparenthesis k \rrparenthesis}{\llparenthesis k \rrparenthesis} \\
\Leftarrow & \qquad \{ \text{ Leibniz } \} \\
& \llparenthesis k \rrparenthesis \cdot \mathsf{in}_{\mathsf{F}} = k \cdot \mathsf{F}\ \llparenthesis k \rrparenthesis \\
\Leftrightarrow & \qquad \{ \text{ universal property (28) } \} \\
& \mathit{true}
\end{array}
$$

□

For example, in the case $R = \mathit{Perm}$ (65), (73) instantiates to

$$\mathit{Perm} \cdot \mathsf{in}_{\mathsf{F}} = \mathit{Perm} \cdot \mathsf{in}_{\mathsf{F}} \cdot (\mathsf{F}\ \mathit{Perm})$$

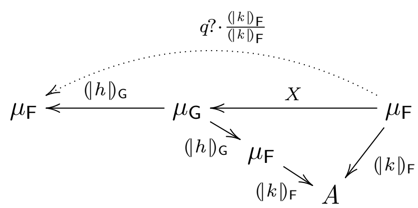
whose useful part is

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

In words, this 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 main usefulness of (72,73) is that the inductive definition of a kernel equivalence relation generated by a fold is such that the recursive branch (the F term) can be added or removed where convenient, as shown in the sequel.

To appreciate relational fold fusion (30) and Theorem 2 at work in metaphorism refinement let us consider metaphorisms of the postconditioned form $M = q? \cdot \frac{\llparenthesis k \rrparenthesis}{\llparenthesis k \rrparenthesis}$ instantiating diagram (58) for inductive types μ_{F} and μ_{G}:[23]



[23] Since there are folds over different types in the diagram we tag each of them with the corresponding functor.

As before, this assumes a (surjective) abstraction function $(\!|h|\!)_{\mathsf{G}} : \mu_{\mathsf{G}} \rightarrow \mu_{\mathsf{F}}$ ensuring that every inhabitant of μ_{F} can be represented by at least one inhabitant of the intermediate type μ_{G}. By direct application of (66) we obtain the equation

$$q? \cdot \frac{(\!|k|\!)_{\mathsf{F}}}{(\!|k|\!)_{\mathsf{F}}} = (\!|h|\!)_{\mathsf{G}} \cdot \underbrace{p? \cdot \frac{(\!|k|\!)_{\mathsf{F}}}{(\!|k|\!)_{\mathsf{F}} \cdot (\!|h|\!)_{\mathsf{G}}}}_{X} \tag{75}$$

provided $q? \cdot (\!|h|\!)_{\mathsf{G}} = (\!|h|\!)_{\mathsf{G}} \cdot p?$ — recall (44). Our main goals are, therefore:

- to find p such that

  $$(\!|h|\!)_{\mathsf{G}} \cdot p? = q? \cdot (\!|h|\!)_{\mathsf{G}} \tag{76}$$

  holds, where q is given;

- to convert $X = p? \cdot \frac{(\!|k|\!)_{\mathsf{F}}}{(\!|k|\!)_{\mathsf{F}} \cdot (\!|h|\!)_{\mathsf{G}}}$, of type $\mu_{\mathsf{G}} \leftarrow \mu_{\mathsf{F}}$ (75), into the converse of a fold, $X = (\!|Z^{\circ}|\!)^{\circ}$, for some $Z : \mathsf{G}\ \mu_{\mathsf{F}} \leftarrow \mu_{\mathsf{F}}$.

In general, we shall use notation $[\!(R)\!]$ to abbreviate the expression $(\!|R^{\circ}|\!)^{\circ}$. In case Z above happens to be a function g, the original metaphorism (whose recursion is F-shaped) will be converted into a so-called *hylomorphism* [5]

$$(\!|h|\!)_{\mathsf{G}} \cdot [\!(g)\!]_{\mathsf{G}}$$

whose recursion is G-shaped, thus carrying a "change of virtual data-structure".

Shifting the metaphor. For the purposes of our calculations in this paper it is enough to consider the partial identities (coreflexives) $p?$ (resp. $q?$) in (76) on inductive type μ_{G} (resp. μ_{F}) generated by constraining the initial algebra in_{G} (resp. in_{F}),

$$\begin{aligned} p? &= (\!|\ \mu_{\mathsf{G}} \xleftarrow{\mathsf{in}_{\mathsf{G}}} \mathsf{G}\ \mu_{\mathsf{G}} \xleftarrow{w?} \mathsf{G}\ \mu_{\mathsf{G}}\ |\!) \\ q? &= (\!|\ \mu_{\mathsf{F}} \xleftarrow{\mathsf{in}_{\mathsf{F}}} \mathsf{F}\ \mu_{\mathsf{F}} \xleftarrow{t?} \mathsf{F}\ \mu_{\mathsf{F}}\ |\!) \end{aligned}$$

for suitable pre-conditions w and t — recall (68).

The calculation of (76) proceeds by fusion (30), aiming to reduce both $(\!|h|\!)_{\mathsf{G}} \cdot p?$ and $q? \cdot (\!|h|\!)_{\mathsf{G}}$ to some relational fold $(\!|R|\!)_{\mathsf{G}}$ over μ_{G}. On the right hand side, fusion yields

$$q? \cdot (\!|h|\!)_{\mathsf{G}} = (\!|R|\!)_{\mathsf{G}} \quad \Leftarrow \quad q? \cdot h = R \cdot (\mathsf{G}\ q?) \tag{77}$$

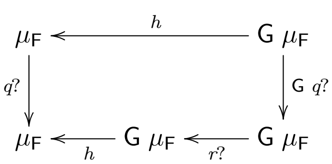
On the other side:

$$\begin{aligned} & (\!|h|\!)_{\mathsf{G}} \cdot p? = (\!|R|\!)_{\mathsf{G}} \\ \Leftrightarrow \quad & \{ \text{ inline } p? = (\!|\mathsf{in}_{\mathsf{G}} \cdot w?|\!)_{\mathsf{G}} \ \} \end{aligned}$$

$$\begin{array}{cl}
& (|h|)_{\mathsf{G}} \cdot (|\mathsf{in}_{\mathsf{G}} \cdot w?|)_{\mathsf{G}} = (|R|)_{\mathsf{G}} \\
\Leftarrow & \quad \{ \text{ fusion (30) } \} \\
& (|h|)_{\mathsf{G}} \cdot \mathsf{in}_{\mathsf{G}} \cdot w? = R \cdot \mathsf{G}\ (|h|)_{\mathsf{G}} \\
\Leftrightarrow & \quad \{ \text{ cancellation: } (|h|)_{\mathsf{G}} \cdot \mathsf{in}_{\mathsf{G}} = h \cdot \mathsf{G}\ (|h|)_{\mathsf{G}} \text{ (31) } \} \\
& h \cdot \mathsf{G}\ (|h|)_{\mathsf{G}} \cdot w? = R \cdot \mathsf{G}\ (|h|)_{\mathsf{G}} \\
\Leftrightarrow & \quad \{ \text{ switch to } r? \text{ such that } \mathsf{G}\ (|h|)_{\mathsf{G}} \cdot w? = r? \cdot \mathsf{G}\ (|h|)_{\mathsf{G}} \text{ holds (44) } \} \\
& h \cdot r? \cdot \mathsf{G}\ (|h|)_{\mathsf{G}} = R \cdot \mathsf{G}\ (|h|)_{\mathsf{G}} \\
\Leftarrow & \quad \{ \text{ Leibniz } \} \\
& h \cdot r? = R
\end{array}$$

Thus $R = h \cdot r?$ ensures proviso (76). By replacing R in the other proviso — the side condition of fusion step (77) — one obtains

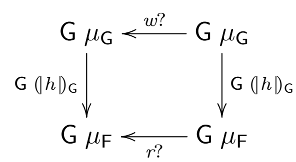
$$q? \cdot h = h \cdot r? \cdot \mathsf{G}\ q?$$



(78)

that has to be ensured together with the other assumption above:

$$\mathsf{G}\ (|h|)_{\mathsf{G}} \cdot w? = r? \cdot \mathsf{G}\ (|h|)_{\mathsf{G}}$$



(79)

Let us summarize these calculations in the form of a theorem.

Theorem 3. *Let* $\mu_{\mathsf{G}} \xrightarrow{(|h|)_{\mathsf{G}}} \mu_{\mathsf{F}}$ *be an abstraction of inductive type* $\mu_{\mathsf{F}} \xleftarrow{\mathsf{in}_{\mathsf{F}}} \mathsf{F}\ \mu_{\mathsf{F}}$ *by another inductive type* $\mu_{\mathsf{G}} \xleftarrow{\mathsf{in}_{\mathsf{G}}} \mathsf{G}\ \mu_{\mathsf{G}}$ *, and* $q? = (|\mathsf{in}_{\mathsf{F}} \cdot t?|)_{\mathsf{F}}$ *be a partial identity representing an inductive predicate over* μ_{F}*.*

To calculate the weakest precondition p *for* $(|h|)_{\mathsf{G}}$ *to ensure* q *on its output, say* $p? = (|\mathsf{in}_{\mathsf{G}} \cdot w?|)_{\mathsf{G}}$*, it suffices to find a predicate* r *on* $\mathsf{G}\ \mu_{\mathsf{F}}$ *such that (78) and (79) hold.*

□

Note how condition r on $\mathsf{G}\ \mu_{\mathsf{F}}$ in proviso (78) is the weakest precondition for algebra h to maintain q, while (79) establishes w as the weakest precondition for the recursive branch $\mathsf{G}\ (|h|)_{\mathsf{G}}$ to ensure r on its output.

Calculating the "divide" step. Armed with side conditions (78) and (79), our final aim is to calculate $X = [\![Z]\!]$ in (75):

$$\underbrace{p? \cdot \frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F} \cdot (\![h]\!)_\mathsf{G}}}_{X} = [\![Z]\!]$$

$$\Leftrightarrow \quad \{ \text{ converses ; } [\![Z]\!] = (\![Z^\circ]\!)^\circ \ \}$$

$$\frac{(\![k]\!)_\mathsf{F} \cdot (\![h]\!)_\mathsf{G}}{(\![k]\!)_\mathsf{F}} \cdot p? = (\![Z^\circ]\!)$$

$$\Leftrightarrow \quad \{ \ (\![h]\!)_\mathsf{G} \cdot p? = q? \cdot (\![h]\!)_\mathsf{G} \text{ assumed — cf. (76) } \}$$

$$\frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F}} \cdot q? \cdot (\![h]\!)_\mathsf{G} = (\![Z^\circ]\!)$$

$$\Leftarrow \quad \{ \text{ fusion (30) ; functor } \mathsf{G} \ \}$$

$$\frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F}} \cdot q? \cdot h = Z^\circ \cdot \mathsf{G} \, \frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F}} \cdot \mathsf{G} \, q?$$

$$\Leftrightarrow \quad \{ \text{ proviso (78): } q? \cdot h = h \cdot r? \cdot \mathsf{G} \, q? \ \}$$

$$\frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F}} \cdot h \cdot r? \cdot \mathsf{G} \, q? = Z^\circ \cdot \mathsf{G} \, \frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F}} \cdot \mathsf{G} \, q?$$

$$\Leftarrow \quad \{ \text{ Leibniz } \}$$

$$\frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F}} \cdot h \cdot r? = Z^\circ \cdot \mathsf{G} \, \frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F}} \tag{80}$$

We are still far from having a closed formula for Z. Can we get rid of term $\mathsf{G} \, \frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F}}$ from the right hand side? This is where Theorem 2 plays a role, enabling such a cancellation provided we ensure that equivalence $\frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F}}$ is a *congruence* for algebra h, which (by virtue of Theorem 2) amounts to ensuring $(\![k]\!)_\mathsf{F} \cdot h \leqslant \mathsf{G} \, (\![k]\!)_\mathsf{F}$. In words: $(\![k]\!)_\mathsf{F} \cdot h$ should be *no more injective* (54) than the recursive branch $\mathsf{G} \, (\![k]\!)_\mathsf{F}$. It turns out that we shall need yet another similar injectivity clause involving r in the sequel. Altogether:

$$(\![k]\!)_\mathsf{F} \cdot h \quad \leqslant \quad \mathsf{G} \, (\![k]\!)_\mathsf{F} \tag{81}$$

$$r \quad \leqslant \quad \mathsf{G} \, (\![k]\!)_\mathsf{F} \tag{82}$$

Below we resume the calculation of (80) assuming (81) and (82):

$$\frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F}} \cdot h \cdot r? = Z^\circ \cdot \mathsf{G} \, \frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F}}$$

$$\Leftrightarrow \quad \{ \ (72) \ \}$$

$$\frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F}} \cdot h \cdot \mathsf{G} \, \frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F}} \cdot r? = Z^\circ \cdot \mathsf{G} \, \frac{(\![k]\!)_\mathsf{F}}{(\![k]\!)_\mathsf{F}}$$

$$\Leftrightarrow \qquad \{ \text{ (49) first, then (45) thanks to (82) } \}$$

$$\frac{(|k|)_{\mathsf{F}}}{(|k|)_{\mathsf{F}}} \cdot h \cdot r? \cdot \frac{\mathsf{G}\ (|k|)_{\mathsf{F}}}{\mathsf{G}\ (|k|)_{\mathsf{F}}} = Z^{\circ} \cdot \frac{\mathsf{G}\ (|k|)_{\mathsf{F}}}{\mathsf{G}\ (|k|)_{\mathsf{F}}}$$

$$\Leftarrow \qquad \{ \text{ drop } \tfrac{\mathsf{G}\ (|k|)_{\mathsf{F}}}{\mathsf{G}\ (|k|)_{\mathsf{F}}} \text{ (Leibniz) } \}$$

$$\frac{(|k|)_{\mathsf{F}}}{(|k|)_{\mathsf{F}}} \cdot h \cdot r? = Z^{\circ} \tag{83}$$

Taking converses, we get

$$Z = r? \cdot \frac{(|k|)_{\mathsf{F}}}{(|k|)_{\mathsf{F}} \cdot h} \tag{84}$$

from (83) — another metaphorism, of the expected type $\mathsf{G}\ \mu_{\mathsf{F}} \longleftarrow \mu_{\mathsf{F}}$.

Summing up, note how the original metaphorism $q? \cdot \frac{(|k|)_{\mathsf{F}}}{(|k|)_{\mathsf{F}}}$ gets converted into a hylomorphism whose *divide* step is another metaphorism (84). Recall that r acts as WP for algebra h to maintain q and that w is the WP for the recursive branch $\mathsf{G}\ (|h|)_{\mathsf{G}}$ to ensure r.

Altogether, the "outer" metaphor which we started from (involving only μ_{F}) disappears and gives place to an "inner" metaphor between inductive types μ_{G} and μ_{F} (the divide step), whereby the optimization is internalized. This "inner" metaphor is more interesting, as we can see by looking at an example of all this reasoning. Before this, we close this section with the checklist of all provisos that have to be verified for Z (84) to exist:

- (78) — establishes r as the weakest precondition for G-algebra h to *maintain* q as an invariant.
- (79) — establishes w as the weakest precondition for the recursive branch $\mathsf{G}\ (|h|)_{\mathsf{G}}$ to ensure r as post-condition.
- (81) — $(|k|)_{\mathsf{F}} \cdot h$ should be no more injective than the recursive branch $\mathsf{G}\ (|k|)_{\mathsf{F}}$.
- (82) — inputs undistinguishable by $\mathsf{G}\ (|k|)_{\mathsf{F}}$ should also be undistinguishable by predicate r.

8. Example: Quicksort

This section shows how the derivation of *quicksort* as given e.g. by Bird and de Moor [5] corresponds to the implementation strategy for metaphorisms given above, under the following instantiations:

- The starting metaphorism is (65) where *Perm* is the list permutation relationship.

- μ_{F} is the usual finite list datatype with constructors (say) nil and cons, that is, $\mathsf{in}_{\mathsf{F}} = [\mathsf{nil}, \mathsf{cons}]$ with base $\mathsf{F}\, f = id + id \times f$ (29).

- μ_{G} is the binary tree data type whose base is $\mathsf{G}\, f = id + id \times f^2$ and whose initial algebra is (say) $\mathsf{in}_{\mathsf{G}} = [\mathsf{empty}, \mathsf{node}]$. (We use abbreviation f^2 for $f \times f$.)

- $(\!|k|\!)_{\mathsf{F}} = bag$, the function which converts a list into the bag (multiset) of its elements.

- $(\!|h|\!)_{\mathsf{G}} = \mathit{flatten}$, for $h = [\mathsf{nil}, inord]$ where $inord\ (a, (x, y)) = x +\!\!+ [\,a\,] +\!\!+ y$; that is, *flatten* is the binary tree into finite list (inorder) traversal surjection.

- $q = ordered$ (65), that is, $q? = (\!|[\mathsf{nil}, \mathsf{cons}] \cdot (id + mn?)|\!)$, for $mn\ (x, xs) = \langle \forall\ x'\ :\ x' \in_{\mu_{\mathsf{F}}} xs :\ x' \geqslant x \rangle$ where $\in_{\mu_{\mathsf{F}}}$ 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. 7, we first have to search for some predicate r that, following (78), should be the weakest precondition for $[\mathsf{nil}, inord]$ to preserve ordered lists ($q?$). We calculate:

$$
\begin{array}{ll}
& q? \cdot [\mathsf{nil}, inord] = [\mathsf{nil}, inord] \cdot r? \cdot (id + id \times (q? \times q?)) \\
\Leftrightarrow & \quad \{\ \text{switch to } s \text{ such that } r? = id + s?\text{; coproducts}\ \} \\
& [q? \cdot \mathsf{nil}, q? \cdot inord] = [\mathsf{nil}, inord \cdot s? \cdot (id \times (q? \times q?))] \\
\Leftrightarrow & \quad \{\ \text{the empty list is trivially ordered}\ \} \\
& q? \cdot inord = inord \cdot s? \cdot (id \times (q? \times q?)) \\
\Leftrightarrow & \quad \{\ \text{universal property (44)}\ \} \\
& s? \cdot (id \times (q? \times q?)) = (q \cdot inord)?
\end{array}
$$

Knowing the definitions of q and $inord$, we easily infer s by going pointwise:

$$
\begin{array}{ll}
& q\ (x +\!\!+ [\,a\,] +\!\!+ y) \\
\Leftrightarrow & \quad \{\ \text{pointwise definition of ordered lists}\ \} \\
& \left\{ \begin{array}{l} (q\ x) \wedge (q\ y) \\ \underbrace{\langle \forall\ b\ :\ b \in_{\mu_{\mathsf{F}}} x :\ b \leqslant a \rangle \wedge \langle \forall\ b\ :\ b \in_{\mu_{\mathsf{F}}} y :\ a \leqslant b \rangle}_{s\ (a,(x,y))} \end{array} \right.
\end{array}
\qquad (85)
$$

Knowing s and thus r, we go back to (84) to calculate the (relational) coalgebra that shall control the *divide* part, still letting $r? = id + s?$,

$$
\begin{array}{l}
Z : 1 + \mu_{\mathsf{F}} \times (\mu_{\mathsf{F}} \times \mu_{\mathsf{F}}) \leftarrow \mu_{\mathsf{F}} \\
Z = (id + s?) \cdot \frac{bag}{bag \cdot [\mathsf{nil}, inord]}
\end{array}
\qquad (86)
$$

as follows:

$$
\begin{array}{ll}
 & Z = (id + s?) \cdot \dfrac{bag}{bag \cdot [\mathsf{nil}, inord]} \\
\Leftrightarrow & \quad \{ \text{ let } Z^\circ = [Z_1^\circ, Z_2^\circ] \ \} \\
 & [Z_1^\circ, Z_2^\circ]^\circ = (id + s?) \cdot \dfrac{bag}{bag \cdot [\mathsf{nil}, inord]} \\
\Leftrightarrow & \quad \{ \text{ take converses } \} \\
 & [Z_1^\circ, Z_2^\circ] = \dfrac{bag \cdot [\mathsf{nil}, inord]}{bag} \cdot (id + s?) \\
\Leftrightarrow & \quad \{ \ Perm\ (65) \text{ ; coproducts } \} \\
 & [Z_1^\circ, Z_2^\circ] = [Perm \cdot \mathsf{nil}, Perm \cdot inord \cdot s?] \\
\Leftrightarrow & \quad \{ \text{ coproducts; } Perm \cdot \mathsf{nil} = \mathsf{nil}\text{; converses } \} \\
 & \left\{ \begin{array}{l} Z_1 = \mathsf{nil}^\circ \\ Z_2 = s? \cdot inord^\circ \cdot Perm \end{array} \right. \\
\Leftrightarrow & \quad \{ \text{ go pointwise } \} \\
 & \left\{ \begin{array}{l} \_\ Z_1\ x \Leftrightarrow x = [\,] \\ (a, (y, z))\ Z_2\ x \Leftrightarrow (a, (y, z))\ (s? \cdot inord^\circ \cdot Perm)\ x \end{array} \right.
\end{array}
$$

The second clause of the bottom line just above unfolds to:

$$(a, (y, z))\ Z_2\ x \Leftrightarrow s\ (a, (y, z)) \wedge (y +\!\!+ [\,a\,] +\!\!+ z)\ Perm\ x$$

In words, $y\ Z\ x$ has the following meaning: either $x = [\,]$ and Z yields the unique inhabitant of singleton type 1 (cf. Z_1) or x is non-empty and Z splits a permutation of x into two halves y and z separated by a "pivot" a, all subject to s calculated above (85). Note the free choice of "pivot" a provided s holds. In the standard version, a is the head of x and Z_2 is rendered deterministic as follows (Haskell notation):

$$x_2\ (h : t) = (h, ([\,a \mid a \leftarrow t, a \leqslant h\,], [\,a \mid a \leftarrow t, a > h\,]))$$

It is easy to show that the particular partition chosen in this standard version meets predicate s. But there is, still, a check-list of proofs to discharge.

Ensuring bi-ordered (virtual) intermediate trees. This corresponds to (79) of the check-list which, instantiated for this exercise, is:

$$\mathsf{G}\ \mathit{flatten} \cdot (id + x?) = (id + s?) \cdot \mathsf{G}\ \mathit{flatten}$$

Letting $w? = id + x?$, the goal is to find weakest precondition x that is basically s "passed along" $\mathsf{G}\ \mathit{flatten}$ from lists to trees:[24]

$$(id \times \mathit{flatten}^2) \cdot x? = s? \cdot (id \times \mathit{flatten}^2)$$

$$\Leftrightarrow \quad \{ \ (44) \ \}$$

$$x = s \cdot (id \times \mathit{flatten}^2)$$

$$\Leftrightarrow \quad \{ \ \text{go pointwise} \ \}$$

$$x\ (a, (t_1, t_2)) = s\ (a, (\mathit{flatten}\ t_1, \mathit{flatten}\ t_2))$$

$$\Leftrightarrow \quad \{ \ \text{definition of } s \ \}$$

$$x\ (a, (t_1, t_2)) = \left\{ \begin{array}{l} \langle \forall\ b\ :\ b\ \epsilon_{\mu \mathsf{F}}\ (\mathit{flatten}\ t_1) :\ b \leqslant a \rangle \\ \langle \forall\ b\ :\ b\ \epsilon_{\mu \mathsf{F}}\ (\mathit{flatten}\ t_2) :\ a \leqslant b \rangle \end{array} \right.$$

$$\Leftrightarrow \quad \{ \ \text{define } \epsilon_{\mu \mathsf{G}} = \epsilon_{\mu \mathsf{F}} \cdot \mathit{flatten} \ \}$$

$$x\ (a, (t_1, t_2)) = \langle \forall\ b\ :\ b\ \epsilon_{\mu \mathsf{G}}\ t_1 :\ b \leqslant a \rangle \wedge \langle \forall\ b\ :\ b\ \epsilon_{\mu \mathsf{G}}\ t_2 :\ a \leqslant b \rangle)$$

In words, x in $p? = (\!| \mathsf{in}_{\mathsf{G}} \cdot w? |\!)_{\mathsf{G}} = (\!| \mathsf{in}_{\mathsf{G}} \cdot (id + x?) |\!)_{\mathsf{G}}$ ensures that the first part of the implementation, controlled by the *divide step* coalgebra Z calculated above (86) yields trees which are *bi-ordered*. Trees with this property are known as *binary search trees* [33].

Preserving the metaphor. Next we consider side condition (81) of the check-list, which instantiates to:

$$bag \cdot [\mathsf{nil}, \mathit{inord}] \leqslant id + id \times bag^2$$

$$\Leftarrow \quad \{ \ \text{coproducts; (52)} \ \}$$

$$bag \cdot \mathsf{nil} + bag \cdot \mathit{inord} \leqslant id + id \times bag^2$$

$$\Leftrightarrow \quad \{ \ (53)\ ;\ \text{any } f \leqslant id\ [27] \ \}$$

$$bag \cdot \mathit{inord} \leqslant id \times bag^2$$

$$\Leftrightarrow \quad \{ \ (40) \ \}$$

$$\langle \exists\ k\ ::\ bag \cdot \mathit{inord} = k \cdot (id \times bag^2) \rangle$$

That k exists arises from the fact that bag is a homomorphism between the monoid of lists and that of bags: algebra k will join two bags and a singleton bag in the same way as $\mathit{inord}\ (a, (x, y))$ yields $x +\!\!+ [\,a\,] +\!\!+ y$, at list level.

Down to the multiset level. Finally, we have to check the last assumption (82) of the ckeck-list. By (40) and (44), this amounts to finding u such that $\mathsf{G}\ bag \cdot r? =$

[24] As before, we abbreviate $\mathit{flatten} \times \mathit{flatten}$ by $\mathit{flatten}^2$ for economy of notation.

$u? \cdot \mathsf{G}\ bag$:

$$
\begin{array}{ll}
& \mathsf{G}\ bag \cdot r? = u? \cdot \mathsf{G}\ bag \\
\Leftrightarrow & \quad \{\ \mathsf{G}\ R = id + id \times R^2\ ;\ r? = id + s?;\ \text{let}\ u? = id + v?\ \} \\
& (id + id \times bag^2) \cdot (id + s?) = (id + v?) \cdot (id + id \times bag^2) \\
\Leftrightarrow & \quad \{\ \text{coproducts}\ \} \\
& (id \times bag^2) \cdot s? = v? \cdot (id \times bag^2) \\
\Leftrightarrow & \quad \{\ (44)\ \} \\
& s = v \cdot (id \times bag^2) \\
\Leftrightarrow & \quad \{\ \text{go pointwise}\ \} \\
& s\ (a, (x, y)) = v\ (a, (bag\ x, bag\ y)) \\
\Leftrightarrow & \quad \{\ \text{unfold}\ s\ \} \\
& v\ (a, (bag\ x, bag\ y)) = \left\{ \begin{array}{l} \langle \forall\ b\ :\ b \in_{\mu\mathsf{F}} x :\ b \leqslant a \rangle \\ \langle \forall\ b\ :\ b \in_{\mu\mathsf{F}} y :\ a \leqslant b \rangle \end{array} \right. \\
\Leftrightarrow & \quad \{\ \text{assume}\ \in_{\mathsf{B}}\ \text{such that}\ \in_{\mu\mathsf{F}} = \in_{\mathsf{B}} \cdot bag\ \} \\
& v\ (a, (bag\ x, bag\ y)) = \left\{ \begin{array}{l} \langle \forall\ b\ :\ b \in_{\mathsf{B}} (bag\ x) :\ b \leqslant a \rangle \\ \langle \forall\ b\ :\ b \in_{\mathsf{B}} (bag\ y) :\ a \leqslant b \rangle \end{array} \right. \\
\Leftarrow & \quad \{\ \text{substitution}\ \} \\
& v\ (a, (b_1, b_2)) = \left\{ \begin{array}{l} \langle \forall\ b\ :\ b \in_{\mathsf{B}} b_1 :\ b \leqslant a \rangle \\ \langle \forall\ b\ :\ b \in_{\mathsf{B}} b_2 :\ a \leqslant b \rangle \end{array} \right.
\end{array}
$$

Thus we have found post-condition u ensured by $id \times bag^2$ with s as weakest-precondition.

Finally, multiset membership $\in_{\mathsf{B}} = \in \cdot\ support$ can be obtained by taking multiset *supports*, whereby we land in standard set membership ($\in$). Altogether, we have relied on 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 — the *invariant* part of the sorting metaphor.

9. Example: Mergesort

In a landmark paper on algorithm classification and synthesis [14], Darlington carries out a derivation of sorting algorithms that places *quicksort* and *mergesort* in different branches of a derivation tree. In this section we give a calculation of mergesort which shows precisely where they differ, given that both are *divide & conquer* algorithms.

The fact that mergesort relies on a different kind of tree, called *leaf tree* and based on a different base functor, say $\mathsf{K}\, f = id + f^2$, is not the main difference. This resides chiefly in the division of work of *mergesort* which, contrary to *quicksort*, does almost everything in the *conquer* step. In our setting,

> while *quicksort* follows generic metaphorism refinement plan (58),
> *mergesort* follows plan (59), recall Sect. 5.

With no further detours we go back to (67), the instance of (59) which fits the sorting metaphorism, to obtain

$$q? \cdot \frac{bag}{bag} = \underbrace{q? \cdot \frac{bag \cdot tips}{bag}}_{(\!|X|\!)_{\mathsf{K}}} \cdot tips^{\circ}$$

where $tips = (\!|t|\!)_{\mathsf{K}}$ is the fold which converts a leaf tree into a sequence of leaves, in the obvious way: $t = [singl\,, conc]$, where $singl\ a = [\,a\,]$ and $conc\ (x, y) = x +\!\!+\, y$.[25]

Divide step $tips^{\circ}$ can be refined into a function using standard "converse of a function" theorems, see e.g. [25, 34]. Our aim is to calculate X, the K-algebra that shall control the *conquer* step. We reason:

$$
\begin{array}{ll}
& (\!|X|\!)_{\mathsf{K}} = q? \cdot \dfrac{bag}{bag} \cdot (\!|t|\!)_{\mathsf{K}} \\
\Leftarrow & \quad \{\ \text{fusion (30) ; functor } \mathsf{K}\ \} \\
& q? \cdot \dfrac{bag}{bag} \cdot t = X \cdot (\mathsf{K}\ q?) \cdot \mathsf{K}\ \dfrac{bag}{bag} \\
\Leftarrow & \quad \{\ \text{(72) assuming } \tfrac{bag}{bag} \text{ is a } \mathsf{K}\text{-congruence for algebra } t \text{ ; Leibniz}\ \} \\
& q? \cdot \dfrac{bag}{bag} \cdot t = X \cdot \mathsf{K}\ q?
\end{array}
$$

Next, we head for a functional implementation $x \subseteq X$:

$$
\begin{array}{ll}
& x \cdot \mathsf{K}\ q? \subseteq q? \cdot \dfrac{bag}{bag} \cdot t \\
\Leftarrow & \quad \{\ \text{cancel } q? \text{ assuming } x \cdot \mathsf{K}\ q? = q? \cdot x \text{ (44)}\ \} \\
& x\ \subseteq\ \dfrac{bag \cdot t}{bag}
\end{array}
$$

Again we obtain solution $x : \mathsf{K}\ \mu_{\mathsf{F}} \to \mu_{\mathsf{F}}$ as a metaphor implementation, essentially requiring that x preserves the bag of elements of the lists involved, as in *quicksort*. Note that the surjectivity of *bag* allows for a total solution x, whose standard implementation is the well-known *list merge* function that

[25] Note that the trivial case of empty list is treated separately from this scheme.

merges two ordered lists into an ordered list. This behaviour is in fact required by the last assumption above, $x \cdot \mathsf{K}\ q? = q? \cdot x$. The other assumption, that $\frac{bag}{bag}$ is a congruence for algebra t, amounts to (recall Theorem 2):

$$
\begin{array}{ll}
 & bag \cdot t \leqslant \mathsf{K}\ bag \\
\Leftarrow & \quad \{\ t = [singl, conc];\ \text{coproduct injectivity (52,53)}\ ;\ \mathsf{K}\, f = id + f^2\ \} \\
 & \left\{ \begin{array}{l} bag \cdot singl \leqslant id \\ bag \cdot conc \leqslant bag^2 \end{array} \right. \\
\Leftrightarrow & \quad \{\ \text{any } f \leqslant id\ \} \\
 & bag \cdot conc \leqslant bag^2 \\
\Leftrightarrow & \quad \{\ (40)\ \} \\
 & \langle \exists\ k\ ::\ bag\ (x +\!\!+\ y) = k\ (bag\ x, bag\ y) \rangle \\
\Leftrightarrow & \quad \{\ \text{same argument as in quicksort}\ \} \\
 & true
\end{array}
$$

□

Summing up, the workload inversion in *mergesort*, compared to *quicksort*, can be felt right at the start of the derivation, by grafting the (range of the) virtual tree representation at the front rather than at the rear of the pipeline.

10. Example: minimum height trees

Our last example addresses a metaphorism $\frac{(\!|f|\!)}{(\!|g|\!)} \upharpoonright R$ in which R is an optimization preorder. It is adapted from [25] where it is labelled *tree with minimum height*. Rephrased in our setting, the problem to be addressed is that of *reshaping a binary tree so as to minimize its height*:[26]

$$
\frac{tips}{tips} \upharpoonright \leqslant_{height} \tag{87}
$$

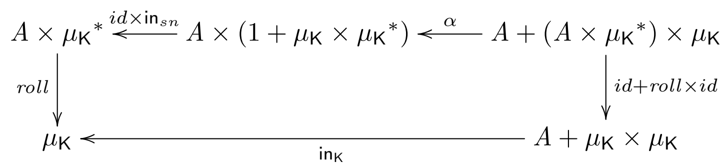
Recall that $tips = (\!|[singl, conc]|\!)_{\mathsf{K}}$ converts a tree into the sequence of its leaves.[27] Heights of trees are calculated by function $height = (\!|[id, ht]|\!)_{\mathsf{K}}$ where $ht\ (a, b) = (a \sqcup b) + 1$, for $a \sqcup b = b \ \Leftrightarrow\ a \leqslant b$. Finally, $\leqslant_{height}$ is a shorthand for $height^{\circ} \cdot (\leqslant) \cdot height$, the preorder that ranks trees according to their height.

By rule (24), (87) is the same as $(tips^{\circ} \upharpoonright \leqslant_{height}) \cdot tips$. Bird et al. [25] show the advantage of handling $tips^{\circ}$ using a different format for trees known as *left*

[26] That is to say, leaves are numbers representing heights of subtrees, and the problem is to assemble the subtrees into a single tree of minimum height.

[27] Note that *tips* is called *flatten* in [25]. Recall $\mathsf{K}\ f = id + f^2$ and $\mathsf{in}_{\mathsf{K}} = [\mathsf{leaf}, \mathsf{fork}]$ from Sect. 9.

spine.[28] Let $\mu\mathsf{K} \xleftarrow{\mathsf{in_K}} A + \mu\mathsf{K}^2$ be our datatype of trees with leaves of type A. The corresponding left spine datatype is $S = A \times \mu\mathsf{K}^*$ and it is isomorphic to $\mu\mathsf{K}$. This isomorphism, termed *roll* as in [25], is depicted below in the form of a diagram, where α is obvious and $\mathsf{in}_{sn} = [\mathsf{nil}, \mathsf{snoc}]$ is the "snoc" variant of initial algebra of lists:[29]

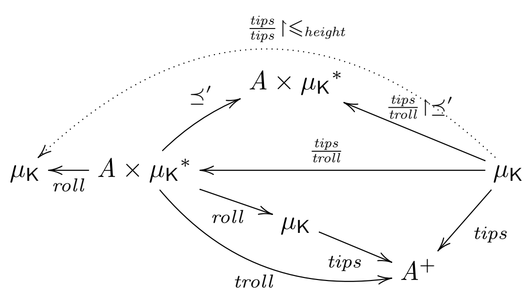


To obtain $roll^\circ$ one just has to reverse all arrows in the diagram, since they are all isomorphisms.

The left spine representation is introduced as in the previous examples:

$$
\begin{array}{ll}
& (tips^\circ \upharpoonright \leqslant_{height}) \cdot tips \\
= & \quad \{\ roll \cdot roll^\circ = id\ \} \\
& ((roll \cdot roll^\circ \cdot tips^\circ) \upharpoonright \leqslant_{height}) \cdot tips \\
= & \quad \{\ \text{by (25) abbreviating } \preceq' = (\leqslant)_{height \cdot roll} \text{ and } troll = tips \cdot roll\ \} \\
& roll \cdot (troll^\circ \upharpoonright \preceq') \cdot tips
\end{array}
$$

Altogether, we are lead to a metaphorism between binary trees and left spines, post processed by *roll*:



The hard bit above is $troll^\circ \upharpoonright \preceq'$, to be addressed in two steps: first, we convert $troll^\circ = (\!|S|\!)$ for some S using the *converse of a function* theorem by Bird and de Moor [5]. Then we use the *greedy theorem* of shrinking [3] to refine $(\!|S|\!) \upharpoonright \preceq'$ into $(\!|S \upharpoonright \preceq'|\!)$. For easy reference, we quote both theorems below from their sources.

[28] In essence, left spines offer a bottom-up access to trees, in this case more convenient than the standard top-down traversal.

[29] Recall $\mathsf{snoc}\ (x, xs) = xs \mathbin{+\!\!+} [x]$ from [5].

Theorem 4 (Converse of a function). *Let* $\mathsf{F}\ \mathsf{T} \xrightarrow{\mathsf{in_T}} \mathsf{T}$ *and* $A \xrightarrow{f} \mathsf{T}$ *be given. Then* $f^\circ = (\!|R|\!)$ *provided* $R : A \leftarrow \mathsf{F}\ A$ *is surjective and such that* $f \cdot R \subseteq \mathsf{in_T} \cdot \mathsf{F}\, f$. *(Proof: see theorem 6.4 in [5].)*
□

Theorem 5 (Greedy shrinking). $(\!|S \upharpoonright R|\!) \subseteq (\!|S|\!) \upharpoonright R$ *provided* R *is transitive and* S *is monotonic with respect to* R°, *that is,* $S \cdot (\mathsf{F}\ R^\circ) \subseteq R^\circ \cdot S$. *(Proof: see theorem 1 in [3].)*
□

In our case, $\mathsf{in_T} : \mathsf{F}\ A^+ \rightarrow A^+$ (non-empty lists) where $\mathsf{F}\ X = A + A \times X$, assuming A fixed. ($A = \mathbb{Z}$ in [25].) We aim at $troll^\circ = (\!|[R, Q]|\!)$, where $R : A \times {\mu_\mathsf{K}}^* \leftarrow A$ and $Q : A \times {\mu_\mathsf{K}}^* \leftarrow A \times (A \times {\mu_\mathsf{K}}^*)$. While $R = id \vartriangle \mathsf{nil}$ is immediate, Q is constrained by theorem 4

$$Q \subseteq \frac{\mathsf{cons} \cdot (id \times troll)}{troll} \tag{88}$$

and by theorem 5:

$$Q \cdot (id \times (\preceq')^\circ) \subseteq (\preceq')^\circ \cdot Q \tag{89}$$

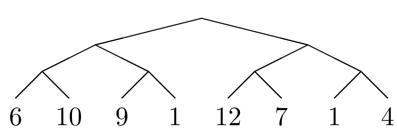
Solutions will be of the form $(\!|[\mathsf{one} \vartriangle \mathsf{nil}, Q \upharpoonright \preceq']|\!)$ by the following properties of shrinking [3]:

$$[S,T] \upharpoonright R = [S \upharpoonright R, T \upharpoonright R]$$
$$f \upharpoonright R = f \ \Leftarrow\ R \text{ is reflexive}$$

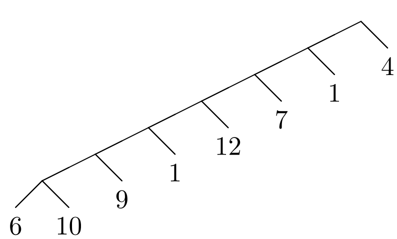
The following property of *troll*

$$troll \cdot (id \times \mathsf{cons} \cdot (\mathsf{leaf} \times id)) = \mathsf{cons} \cdot (id \times troll)$$

naively suggests solution $Q = id \times \mathsf{cons} \cdot (\mathsf{leaf} \times id)$ which, however, does not work: for input tree



with height 15, $(\!|id \times \mathsf{cons} \cdot (\mathsf{leaf} \times id)|\!) \cdot tips$ would generate output tree



with height 17, worsening the input rather than improving it. Still, because

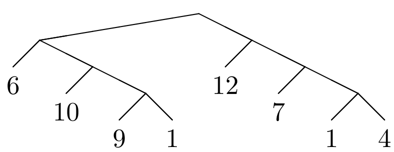
$$\mathsf{head} \cdot troll = \pi_1$$

one may stick to the pattern $Q = \pi_1 \vartriangle U \cdot (id \times (\mathsf{leaf} \times id))$, for some $U : \mu_{\mathsf{K}}{}^* \leftarrow A \times (\mu_{\mathsf{K}} \times \mu_{\mathsf{K}}{}^*)$.

It turns out that finding U such that Q satisfies (88,89) is not easy [25]. The solution is a strategy allowed by the monotonicity of shrinking on the optimization relation: if $P \subseteq R$ then $S \upharpoonright P \subseteq S \upharpoonright R$ and therefore $(\!| S \upharpoonright P |\!) \subseteq (\!| S \upharpoonright R |\!)$.[30] At this point we can pick the refinement $U = minsplit$ given in [25]:

$$\begin{array}{l}
minsplit\ (a, (x, [\,])) = [x] \\
minsplit\ (a, (x, y : xs)) \\
\quad \mid (a \sqcup height\ x) < height\ y = x : y : xs \\
\quad \mid \mathsf{otherwise} = minsplit\ (a, (\mathsf{fork}\ (x, y), xs))
\end{array}$$

For the input tree given above, this refinement will yield



with (minimum) height 14.

To follow the reasoning by Bird et al. [25] that leads to the above solution note that, although shrinking is not used there, it is in a sense implicit. Take $S \upharpoonright R$ and apply the power-transpose to S, obtaining $(\in \cdot \Lambda S) \upharpoonright R$. Since ΛS is a function we can use (24) to get $(\in \upharpoonright R) \cdot \Lambda S$. The relation $\in \upharpoonright R : A \leftarrow \mathsf{P}\ A$, which picks minimal elements of a set according to criterion R is written $min\ R$ in [25]. So expressions of the form $(min\ R \cdot \Lambda S)$ in that paper express the same as $S \upharpoonright R$ in the current paper.

The ordering on left spines found in [25] to ensure the monotonicity of Q is $\sqsubseteq_{lspinecosts}$ where $lspinecosts\ (a, ts) = [(height \cdot roll)\ (a, x) \mid x \leftarrow pref\ ts]$, $pref\ ts$ lists the prefixes of ts in length-decreasing order, and $[a_1 \mathinner{..} a_m] \sqsubseteq [b_1 \mathinner{..} b_n] \Leftrightarrow m \leqslant n \wedge \langle \forall\ i\ :\ i \leqslant m\ :\ a_i \leqslant b_i \rangle$. As examples, let s be the left spine of the first, balanced tree given above as example, and s' be that of the last, minimal-height one. We have $lspinecosts\ s = [15, 12, 11, 6]$ while $lspinecosts\ s' = [14, 12, 6]$, meaning $s' \sqsubseteq_{lspinecosts} s$. That $\sqsubseteq_{lspinecosts} \subseteq \preceq' = \leqslant_{height \cdot roll}$ holds follows immediately from $\mathsf{head} \cdot lspinecosts = height \cdot roll$.

11. Conclusions

This paper identifies a pattern of relational specification, termed *metaphorism*, in which some kernel information of the input is preserved at the same

[30]The *refined greedy theorem* by Bird et al. [25] corresponds to this use of theorem 5.

time some form of optimization takes place towards the output of an algorithmic process. Text processing, sorting and representation changers are given as examples of metaphorisms.

Metaphorisms expose the *variant/invariant* duality essential to program correctness 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*).

At the heart of relational specifications of this kind the paper identifies the occurrence of *metaphors* characterized as *symmetric divisions* [20, 35] of functions. This makes it possible to regard them as *rational* (regular) relations [2] and develop an algebra of metaphors that contains much of what is needed for refining metaphorisms into recursive programs.

In particular, the kind of metaphorism refinement studied in the paper is known as *changing the virtual data structure*, whereby algorithms are restructured in a *divide & conquer* fashion. The paper gives sufficient conditions for such implementations to be calculated in general and gives the derivation of *quicksort* and *mergesort* as examples. The former can be regarded as a generalization of the reasoning about the same algorithm given by Bird and de Moor [5].

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 governs the implementation. Where exactly this inner metaphor is located depends on the overall refinement plan.

The *quicksort* example shows how the outer metaphor, relating lists which permute each other, gives place to an inner metaphor located in the *divide* step that relates lists to binary search trees. This provides technical evidence for *quicksort* being usually classified as a "*Hard Split, Easy Join*" [36] algorithm: indeed, the "metaphor shift" calculated in Sect. 8 shows the workload passing along the conquer layer towards the *divide* one, eventually landing into the *coalgebra* which governs the *"hard" divide* process. Conversely, the inner metaphor in the case of *mergesort* goes into the *algebra* of the *conquer* step, explaining why this is regarded as a "*Easy Split, Hard Join*" algorithm by Howard [36]. As seen in the paper, this has to do with *where* the virtual data structure is placed, either at the front or at the rear of the starting metaphorism.

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(ism) is worth a thousand axioms?

12. Future work

The research reported in this paper falls into the area of investigating how to manage or refine specification vagueness (non-determinism) by means of the "shrinking" combinator proposed elsewhere [3, 37]. The interplay between this combinator and metaphors (as symmetric divisions) has room for further research. Can $\frac{f}{g}$ be generalized to some $\frac{R}{S}$ and still retain metaphors' ability to *equate objects of incompatible orders* [38]? Facts (26), (27) and (33) point towards such a generalization. This relates to another direction for possible genericity: metaphorisms as given in this paper call for a *division allegory* [20] such as that of binary relations. Can this be generalized? Gibbons [21] asks a similar question and suggests *regular* categories as the right abstraction. It will be interesting to generalize metaphorisms in a similar, categorial way.

Another direction for future research is to generalize *shrinking* in metaphorisms to *thinning* [5]. A notation similar to $S \upharpoonright R$ can be adopted for thinning,

$$S \downharpoonright R = \in \setminus S \cap (\in^\circ \cdot R) \,/\, S^\circ$$

where $S \downharpoonright R$ is a set-valued relation: $x\ (S \downharpoonright R)\ a$ holds for exactly those x such that $x \subseteq \Lambda S\ a$ and x is lower-bounded with respect to R. $S \upharpoonright R$ corresponds to that part of $S \downharpoonright R$ whose outputs are singletons containing minima, $\eta \cdot (S \upharpoonright R) \subseteq S \downharpoonright R$ where $\eta\ b = \{\, b \,\}$. For R a preorder one has:

$$S \upharpoonright R = (\in \upharpoonright R) \cdot (S \downharpoonright R)$$

So preorder shrinking can be expressed by thinning. Not surprisingly, shrinking and thinning share similar laws, namely (24), cf. $(S \cdot f) \downharpoonright R = (S \downharpoonright R) \cdot f$. Thus, refining metaphorisms under thinning can also follow the *converse of a function* strategy enabled by theorem 4. Moreover, the thinning counterpart to the greedy theorem, $(\!|(S \cdot \mathsf{F} \in) \downharpoonright R|\!) \subseteq (\!|S|\!) \downharpoonright R$ suggests similar refinement processes. Whether thinning offers genuinely new opportunities for metaphorism reasoning as compared to shrinking is a subject for future research.

Acknowledgements

The origin of this paper is not computer science. When reading Leonard Bernstein's *The Unanswered Question* [38] the author's attention was driven by the *triangular formations* mentioned by the maestro in his metaphorical analysis of music:

> *(...) in music as in poetry, the A and B of a metaphor must both relate to some X-factor (...) such as rhythm or (...) harmonic progressions. You see there still that triangular formation of A, B and X to be reckoned with.*

Bernstein's "triangles" inspired the "cospans" of [1] and of this paper (1), ventured earlier in [18]. Music has to do with sequences of sound events and several of its stylistic features can be described by metaphorisms. It was the generalization of these to arbitrary finite sequences that suggested the sorting examples and then the theory behind this paper.

The author is indebted to his colleague and linguist Álvaro Iriarte for inviting him to contribute to the 2013 *Humanities and Sciences* colloquium where [18] was presented. This was followed by several interesting coffee-time conversations in which Álvaro eventually pointed to Lakoff's work [16]. Reading this classic changed the author's perception of natural language for ever.

On the technical side, comments and suggestions by Ali Jaoua and Ali Mili are gratefully acknowledged. Last but not least, the author thanks the detailed suggestions and comments made by the anonymous referees which helped to improve the original manuscript.

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 Trust (POCI-01-0145-FEDER-016826).

References

[1] J. Oliveira, Metaphorisms in Programming, in: RAMiCS 2015, vol. 9348 of *LNCS*, Springer-Verlag, 171–190, 2015.

[2] A. Jaoua, A. Mili, N. Boudriga, J. L. Durieux, Regularity of Relations: A Measure of Uniformity, TCS 79 (2) (1991) 323–339.

[3] S.-C. Mu, J. Oliveira, Programming from Galois connections, JLAP 81 (6) (2012) 680–704.

[4] G. Hutton, E. Meijer, Back to Basics: Deriving Representation Changers Functionally, JFP 6 (1) (1996) 181–188.

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

[6] A. Mili, J. Desharnais, F. Mili, Computer Program Construction, Oxford University Press, 1994.

[7] J. Riguet, Relations binaires, fermetures, correspondences de Galois, Bulletin de la Société Mathématique de France 76 (1948) 114–155.

[8] G. Hutton, Between Functions and Relations in Calculating Programs, Ph.D. thesis, University of Glasgow, research Report FP-93-5, 1992.

[9] G. Schmidt, Rectangles, Fringes, and Inverses, in: R. Berghammer, B. Möller, G. Struth (Eds.), RelMiCS/AKA 2008, Springer-Verlag, ISBN 978-3-540-78913-0, 352–366, 2008.

[10] M. Erwig, Categorical Programming with Abstract Data Types, in: 7th AMAST, LNCS 1548, 406–421, 1998.

[11] J. Gibbons, Metamorphisms: Streaming representation-changers, SCP 65 (2007) 108–139.

[12] D. Swierstra, O. de Moor, Virtual data structures, in: B. Möller, H. Partsch, S. Schuman (Eds.), Formal Program Development, vol. 755 of *LNCS*, Springer-Verlag, ISBN 978-3-540-57499-6, 355–371, 1993.

[13] A. Takano, E. Meijer, Shortcut Deforestation in Calculational Form, in: S. Peyton-Jones (Ed.), FPLCA, ACM, 306–313, 1995.

[14] J. Darlington, A synthesis of several sorting algorithms, Acta Informatica 11 (1978) 1–30.

[15] J. Oliveira, On the Design of a "Periodic Table" of VDM Specifications, in: Proc. of the VDM'02 workshop, held in conjunction with FME'02 in Copenhagen, invited talk, 2002.

[16] G. Lakoff, M. Johnson, Metaphors we live by, University of Chicago Press, Chicago, ISBN 978-0-226-46800-6, 1980.

[17] I. Richards, The Philosophy of Rhetoric, Oxford University Press, 1936.

[18] J. Oliveira, On the *'A'* that links the *'M'*s of Maths, Music and Maps, contributed talk to the 2013 CEHUM Autumn Colloquium XV (Maths and Comp. Science Panel), U. Minho, Braga, 21-23 Nov. 2013, 2013.

[19] R. Berghammer, G. Schmidt, H. Zierer, Symmetric quotients and domain constructions, IPL 33 (3) (1989) 163–168, ISSN 0020-0190.

[20] P. Freyd, A. Scedrov, Categories, Allegories, vol. 39 of *Mathematical Library*, North-Holland, 1990.

[21] J. Gibbons, Kernels, in a Nut Shell, JLAMP 85 (5, Part 2) (2016) 921–930.

[22] E. Moggi, Notions of computation and monads, Information and Computation 93 (1) (1991) 55 – 92.

[23] R. Backhouse, P. de Bruin, P. Hoogendijk, G. Malcolm, T. Voermans, J. van der Woude, Polynomial relators, in: AMAST'91, Springer-Verlag, 303–362, 1992.

[24] R. Maddux, The Origin Of Relation Algebras In The Development And Axiomatization Of The Calculus Of Relations, Studia Logica 50 (1991) 421–455.

[25] R. Bird, J. Gibbons, S. Mu, Algebraic Methods for Optimization Problems, in: Algebraic and Coalgebraic Methods in the Mathematics of Program Construction, vol. 2297 of *LNCS*, Springer, 281–308, 2002.

[26] D. Naumann, Towards patterns for heaps and imperative lambdas, JLAMP 85 (5, Part 2) (2016) 1038–1056.

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

[28] F. Bancilhon, N. Spyratos, Update Semantics of Relational Views, ACM TDS 6 (4) (1981) 557–575.

[29] H. Doornbos, R. Backhouse, J. van der Woude, A Calculational Approach to Mathematical Induction, TCS 179 (1–2) (1997) 103–135, ISSN 0304-3975.

[30] D. Kozen, Kleene Algebra with Tests, ACM ToPLAS 19 (3) (1997) 427–443.

[31] J. Oliveira, Transforming Data by Calculation, in: GTTSE'07, vol. 5235 of *LNCS*, Springer-Verlag, 134–195, , 2008.

[32] H. Ehrig, B. Mahr, Fundamentals of Algebraic Specification 1: Equations and Initial Semantics, Springer-Verlag, New-York, 1985.

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

[34] S.-C. Mu, R. Bird, Theory and applications of inverting functions as folds, SCP 51 (1–2) (2004) 87–116, ISSN 0167-6423.

[35] G. Schmidt, T. Ströhlein, Relations and Graphs: Discrete Mathematics for Computer Scientists, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1993.

[36] B. Howard, Another Iteration on Darlington's 'A Synthesis of Several Sorting Algorithms', Tech. Rep. KSU CIS 94-8, Department of Computing and Information Sciences, Kansas State University, 1994.

[37] J. Oliveira, M. Ferreira, Alloy Meets the Algebra of Programming: A Case Study, IEEE Trans. Soft. Eng. 39 (3) (2013) 305–326, .

[38] L. Bernstein, The Unanswered Question (Six Talks at Harvard), Harvard University Press, ISBN 9780674920019, 1981.

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

[40] J. Oliveira, Pointfree Foundations for (Generic) Lossless Decomposition, Technical Report TR-HASLab:3:2011, HASLab/INESC TEC & U. Minho, URL `https://repositorium.sdum.uminho.pt/handle/1822/24648`, 2011.

Appendix A. Auxiliary lemmas and proofs

Lemma 6. *Given predicate q and function f,*

$$(q \cdot f)? = \delta\,(q? \cdot f) \qquad \text{(A.1)}$$

holds, where

$$\delta\, R = id \cap R^{\circ} \cdot R \tag{A.2}$$

is the domain *of R.*
Proof:

$$
\begin{array}{ll}
& (q \cdot f)? \\
= & \quad \{ \;\textit{(42)}\; \} \\
& id \cap \dfrac{true}{q \cdot f} \\
= & \quad \{ \;\textit{since } \frac{f}{f} \textit{ is reflexive (38)}\; \} \\
& id \cap \dfrac{f}{f} \cap \dfrac{true \cdot f}{q \cdot f} \\
= & \quad \{ \;\textit{(37) ; products}\; \} \\
& id \cap \dfrac{(id \vartriangle true) \cdot f}{(id \vartriangle q) \cdot f} \\
= & \quad \{ \;\textit{(32) ; (37)}\; \} \\
& id \cap f^{\circ} \cdot (id \cap \dfrac{true}{q}) \cdot f \\
= & \quad \{ \;\textit{(42)}\; \} \\
& id \cap f^{\circ} \cdot q? \cdot f \\
= & \quad \{ \;\textit{(A.2)}\; \} \\
& \delta\, (q? \cdot f)
\end{array}
$$

□

The rest of this appendix provides proofs of results left pending in the main text.

Proof of property (20). **Part (⇒)** — $\frac{R}{R}$ is always an equivalence relation, recall Sect. 3. **Part (⇐)** — assume that R is an equivalence relation. Then:

$$
\begin{array}{ll}
& R = R \setminus R \\
\Leftrightarrow & \quad \{ \text{ since } R \subseteq R \setminus R \text{ just states that } R \text{ is transitive (13) } \} \\
& R \setminus R \subseteq R \\
\Leftarrow & \quad \{ \text{ since } R \cdot (R \setminus R) \subseteq R \text{ by (13) } \} \\
& R \setminus R \subseteq R \cdot (R \setminus R)
\end{array}
$$

$$\begin{array}{ll}
\Leftarrow & \quad \{ \text{ composition is monotone } \} \\
& id \subseteq R \\
\Leftrightarrow & \quad \{ \ R \text{ is reflexive } \} \\
& true
\end{array}$$

Then:

$$\begin{array}{ll}
& R = \dfrac{R}{R} \\
\Leftrightarrow & \quad \{ \text{ (12) ; } R^\circ \mathbin{/} R^\circ = (R \setminus R)^\circ \ \} \\
& R = R \setminus R \cap (R \setminus R)^\circ \\
\Leftrightarrow & \quad \{ \ R = R \setminus R \text{ above } \} \\
& R = R \cap R^\circ \\
\Leftrightarrow & \quad \{ \text{ since } R \text{ is symmetric: } R = R^\circ \ \} \\
& true
\end{array}$$

$\square$

Proof of property (44). **Part (⇒)** — show that $p = q{\cdot}f$ follows from $f{\cdot}p? = q?{\cdot}f$:

$$\begin{array}{ll}
& p = q \cdot f \\
\Leftrightarrow & \quad \{ \text{ bijection between predicates and partial identities } \} \\
& p? = (q \cdot f)? \\
\Leftrightarrow & \quad \{ \text{ (A.1) ; } f \cdot p? = q? \cdot f \text{ assumed } \} \\
& p? = \delta\,(f \cdot p?) \\
\Leftrightarrow & \quad \{ \ \delta\,(R \cdot S) = \delta\,(\delta\,R \cdot S) \ \} \\
& p? = \delta\,(\delta\,f \cdot p?) \\
\Leftrightarrow & \quad \{ \ \delta\,f = id \ \} \\
& p? = \delta\,(p?) \\
\Leftrightarrow & \quad \{ \text{ domain of a partial identity is itself } \} \\
& true
\end{array}$$

$\square$

Part ($\Leftarrow$) — show that $f \cdot p? = q? \cdot f$ holds assuming $p = q \cdot f$:

$$
\begin{array}{ll}
& f \cdot p? = q? \cdot f \\
\Leftrightarrow & \quad \{ \text{ substitution } p := q \cdot f ;\ (\text{A.1}) \ \} \\
& f \cdot \delta\,(q? \cdot f) = q? \cdot f \\
\Leftarrow & \quad \{ \ \subseteq\text{-antisymmetry, since } \delta\,(q? \cdot f) \subseteq f^{\circ} \cdot q? \cdot f \text{ and } f \cdot f^{\circ} \subseteq id \ \} \\
& q? \cdot f \subseteq f \cdot \delta\,(q? \cdot f) \\
\Leftrightarrow & \quad \{ \ R = R \cdot \delta\, R \ \} \\
& q? \cdot f \cdot \delta\,(q? \cdot f) \subseteq f \cdot \delta\,(q? \cdot f) \\
\Leftarrow & \quad \{ \text{ monotonicity of composition } \} \\
& q? \subseteq id \\
\Leftrightarrow & \quad \{ \ q? \text{ is a partial identity } \} \\
& true
\end{array}
$$

□

Proof of property (45). By (39), $p \leqslant. f$ is equivalent to the existence of some q such that $p = q \cdot f$ holds, which in turn is equivalent to $f \cdot p? = q? \cdot f$ by (44). Then:

$$
\begin{array}{ll}
& \frac{f}{f} \cdot p? \\
= & \quad \{ \text{ metaphors (32) ; (44) } \} \\
& f^{\circ} \cdot q? \cdot f \\
= & \quad \{ \text{ converses ; partial identities } \} \\
& (q? \cdot f)^{\circ} \cdot f \\
= & \quad \{ \text{ again (44) and (32) } \} \\
& p? \cdot \frac{f}{f}
\end{array}
$$

□

Proof of property (64). Our strategy is indirect equality carried over the universal property of the shrinking operator (22):

$$
\begin{array}{ll}
& X \subseteq S \upharpoonright (q? \cdot \top) \\
\Leftrightarrow & \quad \{ \text{ (22) ; (36) } \} \\
& X \subseteq S \wedge X \cdot S^{\circ} \subseteq q? \cdot \frac{!}{!}
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow & \{ \text{ shunting (6) ; converses } \} \\
& X \subseteq S \wedge X \cdot (! \cdot S)^{\circ} \subseteq q? \cdot !^{\circ} \\
\Leftrightarrow & \{ \text{ assume } S \text{ entire, so } ! \cdot S = ! \ \} \\
& X \subseteq S \wedge X \cdot !^{\circ} \subseteq q? \cdot !^{\circ} \\
\Leftrightarrow & \{ \text{ shunting (6) ; (36) } \} \\
& X \subseteq S \wedge X \subseteq q? \cdot \top \\
\Leftrightarrow & \{ \text{ (A.4) below } \} \\
& X \subseteq q? \cdot S \\
:: & \{ \text{ indirect equality } \} \\
& S \upharpoonright (q? \cdot \top) = q? \cdot S
\end{array}
$$

□

The proof relies on a well-known property of partial identities, given below together with its converse version:

$$R \cdot p? \quad = \quad R \cap \top \cdot p? \tag{A.3}$$

$$q? \cdot R \quad = \quad R \cap q? \cdot \top \tag{A.4}$$

see e.g. [39].

Proof of Theorem 2. Equality (73) follows immediately from (72) by fold-cancellation (31). Next we show the equivalence between (72) and (71):

$$
\begin{array}{ll}
& R \cdot h = R \cdot h \cdot (\mathsf{F}\ R) \\
\Leftrightarrow & \{ \ R \cdot h \ \subseteq \ R \cdot h \cdot (\mathsf{F}\ R) \text{ holds by } id \subseteq \mathsf{F}\ R\text{, since } id \subseteq R \ \} \\
& R \cdot h \cdot (\mathsf{F}\ R) \ \subseteq \ R \cdot h \\
\Leftrightarrow & \{ \ (R\cdot) \text{ is a closure operation, see (A.5) below } \} \\
& h \cdot (\mathsf{F}\ R) \ \subseteq \ R \cdot h
\end{array}
$$

□

The last step relies on the fact that composition with equivalence relations is a *closure* operation:

$$R \cdot S \subseteq R \cdot Q \quad \Leftrightarrow \quad S \subseteq R \cdot Q \tag{A.5}$$

This fact is used elsewhere [40] to reason about functional dependencies in databases. Below we rephrase its proof using the power transpose ΛR which maps objects to their R-equivalence classes (41):

$$
\begin{array}{ll}
& R \cdot S \;\subseteq\; R \cdot Q \\
\Leftrightarrow & \quad \{ \;\; R = \frac{\Lambda R}{\Lambda R} \; (41) \;\; \} \\
& \dfrac{\Lambda R}{\Lambda R} \cdot S \;\subseteq\; \dfrac{\Lambda R}{\Lambda R} \cdot Q \\
\Leftrightarrow & \quad \{ \;\; \frac{\Lambda R}{\Lambda R} = \Lambda R^{\circ} \cdot \Lambda R \; (32) \text{ ; shunting (5)} \;\; \} \\
& \Lambda R \cdot \Lambda R^{\circ} \cdot \Lambda R \cdot S \;\subseteq\; \Lambda R \cdot Q \\
\Leftrightarrow & \quad \{ \; f \cdot f^{\circ} \cdot f = f \text{ (difunctionality)} \; \} \\
& \Lambda R \cdot S \;\subseteq\; \Lambda R \cdot Q \\
\Leftrightarrow & \quad \{ \;\; \text{shunting (5) ; } \Lambda R^{\circ} \cdot \Lambda R = \frac{\Lambda R}{\Lambda R} = R \; (41) \; \} \\
& S \;\subseteq\; R \cdot Q
\end{array}
$$

□

Finally, the proof that (74) is equivalent to (71) for the special case $R = \frac{f}{f}$:

$$
\begin{array}{ll}
& h \cdot (\mathsf{F}\, \dfrac{f}{f}) \;\subseteq\; \dfrac{f}{f} \cdot h \\
\Leftrightarrow & \quad \{ \;\; \text{metaphor algebra: (32) etc} \;\; \} \\
& \mathsf{F}\, \dfrac{f}{f} \;\subseteq\; \dfrac{f \cdot h}{f \cdot h} \\
\Leftrightarrow & \quad \{ \;\; \text{injectivity preorder (39) ; relator } \mathsf{F} \text{ (49)} \;\; \} \\
& f \cdot h \leqslant \mathsf{F}\, f
\end{array}
$$

□