

Programming from Metaphorisms

J.N. Oliveira

IFIP WG2.1 MEETING #73
GOTHENBURG
AUGUST 2015



INESC TEC & University of Minho

Grant PTDC/EIA-CCO/122240/2010

Metaphorism < metaphor



Cognitive linguistics versus Chomskian **generative** linguistics

- Information science is based on Chomskian **generative grammars**
- Semantics is a “quotient” of syntax
- **Cognitive linguistics** has emerged meanwhile
- Emphasis on conceptual **metaphors** — the basic semantics building block
- *Metaphors we live by* (Lakoff and Johnson, 1980).

Metaphors we live by



A cognitive metaphor is a device whereby the meaning of an idea (concept) is carried by another, e.g.

*She **counterattacked** with a **winning** argument*

— the underlying metaphor is **ARGUMENT IS WAR**.

Metaphor **TIME IS MONEY** underlies everyday phrases such as e.g.:

*You are **wasting** my time*

***Invest** your time in something else.*



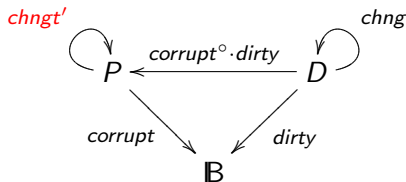
Metaphoric language

Attributed to Mark Twain:

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

(‘No jobs for the boys’ in metaphorical form).

Metaphor structure, where P = politician and D = diaper:



$dirty(chng\ x) = False$ induces $chngt'$ over P , and so on.



Formal metaphors

In his *Philosophy of Rhetoric*, Richards (1936) finds three kernel ingredients in a metaphor, namely

- a **tenor** (e.g. *politicians*)
- a **vehicle** (e.g. *diapers*)
- an implicit, shared **attribute**.

Formally, we have a “cospan”



where functions $f : \mathbf{T} \rightarrow \mathbf{A}$ and $g : \mathbf{V} \rightarrow \mathbf{A}$ extract the common **attribute** (\mathbf{A}) from **tenor** (\mathbf{T}) and **vehicle** (\mathbf{V}).

Formal metaphors



The cognitive, æsthetic, or witty power of a **metaphor** is obtained by *hiding* A , thereby establishing a *composite*, **binary relationship**

$$\mathbf{T} \xleftarrow{f \circ g} V$$

— the “**T** is V ” metaphor — which leaves A implicit.

Mathematics terminology is inherently metaphoric, cf. e.g. (in our field)

- polynomial functor
- vector addition

(algebraic structure sharing) and so is **computing** terminology in general

- ... **stack**, **queue**, **pipe**, **memory**, **driver**, ...

in a true cognitive sense.



“Metaphoric” software science?

Two flavours in (applied) **linguistics**,

- **generative** (grammars, parsing)
- **cognitive** (“metaphors we live by” ...)

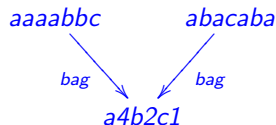
Parallel to software science — further to **hylomorphisms** with pattern $f \cdot g^\circ$, e.g. context-free languages, compilers:

compiler = *code_generator* · *pretty_printer*[◦]

how about “**metaphorisms**” with pattern $f^\circ \cdot g$, e.g. sorting:

sort = *ordered* · (*bag*[◦] · *bag*)

?



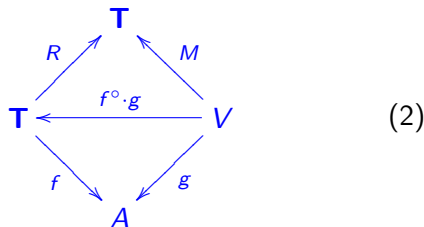


Metaphorical specifications

In the field of program specification, many problem statements are indeed **metaphorical** in such a formal sense.

Such “**metaphorisms**” are input-output relationships in which some hidden information is **preserved** (the **invariant** part), subject to some form of optimization (the **variant** part):

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

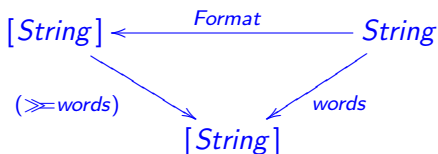


Shrinking $(\cdot \upharpoonright \cdot)$ reduces the **vagueness** of relation $f^\circ \cdot g$ in (2) under criterion R , which tells which **T**s are “better”.



Text formatting metaphorism

Formatted text is a sequence of text lines,



such that the original sequence of **words** is preserved when white space is ignored.

Formatting consists in (re)introducing white space evenly throughout the output text lines,

$$\text{Format} = ((\gg\text{words})^\circ \cdot \text{words}) \upharpoonright R \quad (3)$$

as specified by some convenient criterion *R*.

Other metaphorisms



- **Source code refactoring** — the meaning of the source program is preserved, the target code being better styled wrt. coding conventions and best practices.
- **Change of base** (numeric representation) — the numbers represented by the source and the result are the same, cf. the 'representation changers' (Hutton and Meijer, 1996).
- **Sorting** — the bag (multiset) of elements of the source list is preserved, the optimization consisting in obtaining an ordered output.

etc

Shrunken equivalence relation = metaphorism



Wherever $f = g$ in (2) we get

$$M = (f^\circ \cdot f) \upharpoonright R \quad (4)$$

— a “shrunken” equivalence relation because $f^\circ \cdot f = \mathbf{ker} f$ is an equivalence (the **kernel** of f).

Meaning of $y M x$:

- $f y = f x$ (this is the formal **metaphor**)
- y is “best” among all other y' such that $f y' = f x$, recall $S \upharpoonright R = S \cap R / S^\circ$ (this is the **optimization**), that is:

$$\langle \forall y' : f y' = f x : y R y' \rangle$$

(Recall *Programming from Galois connections*, meeting #68, Rome).



Inductive metaphorisms

From a strict, cognitive point of view, case $f = g$ in (2) leads to “poor metaphors”.

Things become more interesting wherever $f = g = \langle k \rangle$ over an inductive type, say $\mathbf{T} \xleftarrow{\text{in}} \mathbf{F} \mathbf{T}$, that is, $f \cdot \text{in} = k \cdot (\mathbf{F} f)$.

In this case, for surjective $f = \langle k \rangle$

$$\ker f = \langle \ker f \cdot \text{in} \rangle \tag{5}$$

holds, meaning that metaphorism $M = \ker f \upharpoonright R$ can be implemented by calculating $M = \langle \ker f \cdot \text{in} \rangle \upharpoonright R$ — cf. “greedy” theorems, etc



Calculating metaphorisms

Another alternative is to shrink only $(\llbracket k \rrbracket)^\circ$ and then fuse the outcome with $(\llbracket k \rrbracket)$, cf.

$$M = (\mathbf{ker} \llbracket k \rrbracket) \upharpoonright R = ((\llbracket k \rrbracket)^\circ \upharpoonright R) \cdot (\llbracket k \rrbracket) \quad (6)$$

by this law of shrinking: $(S \cdot f) \upharpoonright R = (S \upharpoonright R) \cdot f$.

There are, still, other calculational alternatives that lead to “richer metaphors” which in turn lead to more interesting programs.

These amount to what has elsewhere been known as *changing the virtual data structure* (Swierstra and de Moor, 1993).

AoP, pp.154–155



154

6 / Recursive Programs

Quicksort

The so-called ‘advanced’ sorting algorithms (quicksort, mergesort, heapsort, and so on) all use some form of tree as an intermediate datatype. Here we sketch the development of Hoare’s quicksort (Hoare 1962), which follows the path of selection sort quite closely.

Consider the type *tree* *A* defined by

$$\text{tree } A ::= \text{null} \mid \text{fork} (\text{tree } A, A, \text{tree } A).$$

The function *flatten* : *list* *A* ← *tree* *A* is defined by

$$\text{flatten} = \langle \text{nil}, \text{join} \rangle,$$

where *join* (*x*, *a*, *y*) = *x* + [*a*] + *y*. Thus *flatten* produces a list of the elements in a tree in left to right order.

In outline, the derivation of quicksort is

$$\begin{aligned} & \text{ordered} \cdot \text{perm} \\ \supseteq & \quad \{\text{since } \text{flatten} \text{ is a function}\} \\ & \text{ordered} \cdot \text{flatten} \cdot \text{flatten}^\circ \cdot \text{perm} \\ = & \quad \{\text{claim: } \text{ordered} \cdot \text{flatten} = \text{flatten} \cdot \text{inordered} \text{ (see below)}\} \\ & \text{flatten} \cdot \text{inordered} \cdot \text{flatten}^\circ \cdot \text{perm} \\ = & \quad \{\text{converses}\} \\ & \text{flatten} \cdot (\text{perm} \cdot \text{flatten} \cdot \text{inordered})^\circ \\ \supseteq & \quad \{\text{fusion, for an appropriate definition of } \text{split}\} \\ & \text{flatten} \cdot \langle \text{nil}, \text{split}^\circ \rangle^\circ. \end{aligned}$$

In quicksort we head for an algorithm expressed as a hylomorphism using trees as an intermediate datatype.

The coreflexive *inordered* on trees is defined by

$$\text{inordered} = \langle \text{null}, \text{fork} \cdot \text{check} \rangle$$

where the coreflexive *check* holds for (*x*, *a*, *y*) if

$$(\forall b : b \text{intree } z \Rightarrow bRa) \wedge (\forall b : b \text{intree } y \Rightarrow aRb).$$

The relation *mtree* is the membership test for trees. Introducing $Ff = f \times \text{id} \times f$ for brevity, the proviso for the fusion step in the above calculation is

6.6 / Sorting by selection

155

To establish this condition we need the coreflexive *check'* that holds for (*x*, *a*, *y*) if

$$(\forall b : b \text{inlist } x \Rightarrow bRa) \wedge (\forall b : b \text{inlist } y \Rightarrow aRb).$$

Thus *check'* is similar to *check* except for the switch to lists.

We now reason:

$$\begin{aligned} & \text{perm} \cdot \text{flatten} \cdot \text{fork} \cdot \text{check} \\ = & \quad \{\text{catamorphisms, since } \text{flatten} = \langle \text{nil}, \text{join} \rangle\} \\ & \text{perm} \cdot \text{join} \cdot F \text{flatten} \cdot \text{check} \\ = & \quad \{\text{claim: } F \text{flatten} \cdot \text{check} = \text{check}' \cdot F \text{flatten}\} \\ & \text{perm} \cdot \text{join} \cdot \text{check}' \cdot F \text{flatten} \\ = & \quad \{\text{claim: } \text{perm} \cdot \text{join} = \text{perm} \cdot \text{join} \cdot F \text{perm}\} \\ & \text{perm} \cdot \text{join} \cdot F \text{perm} \cdot \text{check}' \cdot F \text{flatten} \\ = & \quad \{\text{claim: } F \text{perm} \cdot \text{check}' = \text{check}' \cdot F \text{perm}; \text{ functors}\} \\ & \text{perm} \cdot \text{join} \cdot \text{check}' \cdot F(\text{perm} \cdot \text{flatten}) \\ \supseteq & \quad \{\text{taking } \text{split} \subseteq \text{check}' \cdot \text{join}^\circ \cdot \text{perm}\} \\ & \text{split}^\circ \cdot F(\text{perm} \cdot \text{flatten}). \end{aligned}$$

Formal proofs of the three claims are left as exercises. In words, *split* is defined by the rule that if (*y*, *a*, *z*) = *split* *x*, then *y* + [*a*] + *z* is a permutation of *x* with *bRa* for all *b* in *y* and *aRb* for all *b* in *z*. As in the case of selection sort, we can implement *split* with a catamorphism on non-empty lists:

$$\text{split} = \langle \text{base}, \text{step} \rangle \cdot \text{embed}.$$

The fusion conditions are:

$$\begin{aligned} & \text{base} \subseteq \text{check}' \cdot \text{join}^\circ \cdot \text{perm} \cdot \text{wrap} \\ & \text{split} \cdot (\text{id} \times \text{check}' \cdot \text{join}) \subseteq \text{check}' \cdot \text{join}^\circ \cdot \text{perm} \cdot \text{cons}. \end{aligned}$$

These conditions are satisfied by taking

$$\begin{aligned} & \text{base } a = (\langle \rangle, a, \langle \rangle) \\ & \text{step } (a, (x, b, y)) = \begin{cases} \langle [a] \rangle + x, b, y, & \text{if } aRb \\ (x, b, [a] + y), & \text{otherwise.} \end{cases} \end{aligned}$$

Finally, appeal to the hylomorphism theorem gives that $X = \text{flatten} \cdot \langle \text{nil}, \text{split}^\circ \rangle^\circ$ is the least solution of the equation



Enriching the metaphor

Changing the virtual data structure amounts to, in the first place, composing the metaphor with a very special hylomorphism: the image $(h) \cdot (h)^\circ = id$ of a **surjective** fold over **another** datatype **W**:

$$\begin{array}{ccccc}
 & \mathbf{W} & & & \\
 & \swarrow & \nwarrow & & \\
 \mathbf{T} & & \mathbf{T} & \xleftarrow{\ker (k)} & \mathbf{T} \\
 & \swarrow & \nwarrow & \searrow & \swarrow \\
 & & & \mathbf{A} & \\
 & \swarrow & \nwarrow & & \\
 \mathbf{T} & & \mathbf{T} & & \mathbf{T}
 \end{array}
 \quad (7)$$

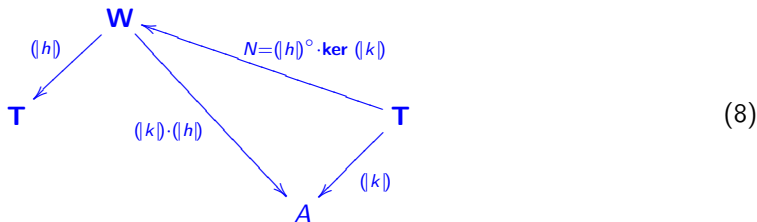
Diagram description: A commutative diagram with nodes **W**, **T**, and **A**. Node **W** is at the top. Node **T** appears three times: one on the left, one in the middle, and one on the right. Node **A** is at the bottom. Arrows: **W** to left **T** labeled (h) ; left **T** to **W** labeled $(h)^\circ$; left **T** to middle **T** labeled $id = img (h)$; middle **T** to **W** labeled $(h)^\circ$; middle **T** to right **T** labeled $\ker (k)$; middle **T** to **A** labeled (k) ; right **T** to **A** labeled (k) .

Typically, choose polynomial **W** of degree higher than **T**, e.g. **binary trees** versus **finite lists**.



Enriching the metaphor

We are heading towards a “richer metaphor” able to shift the “ictus” of algorithmic control from type **T** to type **W**:



W is the (virtual) data type chosen to command the **divide & conquer** algorithmic control.

The aim is to convert N into the converse of a fold, say $(|X|)^{\circ}$ so that we get a hylo-morphism as final implementation.

However, a *metaphorism* = *metaphor* + *optimization*, so we have to consider this too.

Special case of optimization (shrinking)



It may be the case that R in

$$M = (\mathbf{ker} \ (|k|)) \upharpoonright R$$

is of the form $R = p? \cdot \top$ where \top is the *coexists* relation $y \top x = \mathit{true}$ (de Morgan's terminology) and $p?$ is the partial identity (test) which represents predicate p .

Thus $y (p? \cdot \top) x = p y$

It can be shown that:

$$S \upharpoonright (p? \cdot \top) = p? \cdot S \quad \Leftarrow \quad S \text{ is entire} \quad (9)$$

Special case of optimization (shrinking)



Kernels are reflexive and therefore entire. Thus:

$$M = \mathbf{ker} (\downarrow k) \uparrow (p? \cdot \top) = p? \cdot \mathbf{ker} (\downarrow k)$$

Example — the **sorting metaphorism**

$$Sort = (ordered?) \cdot Perm$$

where *Perm* is the metaphor $Perm = \mathbf{ker} bag$ (*bag* computes the bag of elements of a finite list.), that is:

$$y (Sort) x = ordered y \wedge (bag y = bag x)$$



Sorting example (details)

- \mathbf{T} = finite cons-lists, $\text{in}_{\mathbf{T}} = [\text{nil}, \text{cons}]$.
- \mathbf{W} = binary labelled trees, $\mathbf{W} \xleftarrow{\text{in}_{\mathbf{W}} = [\text{empty}, \text{fork}]} \mathbf{F} \mathbf{W}$ where $\mathbf{F} f = \text{id} + \text{id} \times (f \times f)$
- $(\text{!}k) = \text{bag}$ — converts finite lists to bags (multisets of elements).
- $(\text{!}h) = \text{flatten}$, for $h = [\text{nil}, \text{inord}]$ where

$$\text{inord}(a, (x, y)) = x \text{ ++ } [a] \text{ ++ } y$$
 (inorder traversal).
- $q?$ tests for ordered lists, $q? = (([\text{nil}, \text{cons}] \cdot (\text{id} + \text{mn?}))$ where $\text{mn}(x, \text{xs}) = \langle \forall x' : x' \in_{\mathbf{T}} \text{xs} : x' \geq x \rangle$, where $\in_{\mathbf{T}}$ denotes list membership.
(Predicate $\text{mn}(x, \text{xs})$ ensures that list $x : \text{xs}$ is such that x is at most the minimum of xs , if it exists.)

Shrinking metaphorisms into hylomorphisms



$$M = q? \cdot \mathbf{ker} \langle k \rangle$$

$$\Leftrightarrow \{ \text{inline image of surjective } \langle h \rangle \}$$

$$M = \langle h \rangle \cdot \langle h \rangle^\circ \cdot (q?) \cdot \mathbf{ker} \langle k \rangle$$

$$\Leftrightarrow \{ \text{converses ; definition of } \mathbf{ker} \cdot \}$$

$$M = \langle h \rangle \cdot (q? \cdot \langle h \rangle)^\circ \cdot \langle k \rangle^\circ \cdot \langle k \rangle$$

$$\Leftrightarrow \{ \text{hint: assume } p? \text{ such that } \langle h \rangle \cdot p? = q? \cdot \langle h \rangle \}$$

$$M = \langle h \rangle \cdot \underbrace{p? \cdot (\langle k \rangle \cdot \langle h \rangle)^\circ \cdot \langle k \rangle}_{\llbracket X \rrbracket}$$

$\llbracket X \rrbracket$ is the (relational) **divide** step in

$$\mathbf{T} \xleftarrow{\langle h \rangle} \mathbf{W} \xleftarrow{\llbracket X \rrbracket} \mathbf{T}$$

— a new metaphorism, now between \mathbf{W} and \mathbf{T} .

Shrinking metaphorisms into hylomorphisms



Before we proceed to calculating the divide step, let us see how to get $p?$ such that:

$$(h) \cdot p? = q? \cdot (h)$$

This is **weakest pre-condition** for (h) to ensure $q?$ on its output, cf. the general fact

$$f \cdot p? = q? \cdot f \Leftrightarrow p = \mathbf{wp}(f, q) \quad (10)$$

where $\mathbf{wp}(f, q) x = q(f x)$, i.e.

$$\mathbf{wp}(f, q)? = f^\circ \cdot q? \cdot f \cap id = \mathbf{dom}(q? \cdot f) \quad (11)$$

in pointfree notation.

Shrinking metaphorisms into hylomorphisms



In the case of

$$(h) \cdot p? = q? \cdot (h) \quad (12)$$

where $(h) : \mathbf{W} \rightarrow \mathbf{T}$, $p? : \mathbf{W} \rightarrow \mathbf{W}$ will be of shape

$$p? = (\mathbf{W} \xleftarrow{\text{in}_\mathbf{W}} \mathbf{F} \mathbf{W} \xleftarrow{w?} \mathbf{F} \mathbf{W})$$

where $\text{in}_\mathbf{W}$ is the initial algebra of \mathbf{W} , i.e. $(\text{in}_\mathbf{W}) = id$, and so $p? \subseteq id$ by monotonicity since $w? \subseteq id$.

Similarly for $\mathbf{T} \xleftarrow{q?} \mathbf{T} = \text{in}_\mathbf{T} \cdot t?$, for some t .



Shifting the metaphor

We calculate $p = \mathbf{wp}(f, q)$ by (relational) fold-fusion,

$$S \cdot (\llbracket R \rrbracket) = (\llbracket Q \rrbracket) \quad \Leftarrow \quad S \cdot R = Q \cdot \mathbf{F} S \quad (13)$$

aiming at reducing both $(\llbracket h \rrbracket) \cdot p?$ and $q? \cdot (\llbracket h \rrbracket)$ to some fold $(\llbracket R \rrbracket)$ over \mathbf{W} .

Reducing one side:

$$q? \cdot (\llbracket h \rrbracket) = (\llbracket R \rrbracket) \quad \Leftarrow \quad q? \cdot h = R \cdot (\mathbf{F} q?) \quad (14)$$

by (13).

Reducing the other side:

$$\begin{aligned} & (\llbracket h \rrbracket) \cdot p? = (\llbracket R \rrbracket) \\ \Leftrightarrow & \quad \{ \text{inline } p? = (\text{in}_{\mathbf{W}} \cdot w?) \} \end{aligned}$$



Shifting the metaphor

$$\Leftrightarrow \quad \{ \text{inline } p? = (\text{in}_W \cdot w?) \}$$

$$(\!|h)\!| \cdot (\text{in}_W \cdot w?) = (\!|R\!|)$$

$$\Leftarrow \quad \{ \text{fusion (13)} \}$$

$$(\!|h)\!| \cdot \text{in}_W \cdot w? = R \cdot (\mathbf{F} (\!|h)\!|)$$

$$\Leftrightarrow \quad \{ \text{cancellation: } (\!|h)\!| \cdot \text{in}_W = h \cdot \mathbf{F} (\!|h)\!| \}$$

$$h \cdot \mathbf{F} (\!|h)\!| \cdot w? = R \cdot (\mathbf{F} (\!|h)\!|)$$

$$\Leftrightarrow \quad \{ \text{assume } r? \text{ such that } \mathbf{F} (\!|h)\!| \cdot w? = r? \cdot \mathbf{F} (\!|h)\!| \}$$

$$h \cdot r? \cdot (\mathbf{F} (\!|h)\!|) = R \cdot (\mathbf{F} (\!|h)\!|)$$

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

$$h \cdot r? = R$$



Shifting the metaphor

Thus we get $R = h \cdot r?$ which ensures the equality, which we can replace in $q? \cdot h = R \cdot (\mathbf{F} q?)$, the side condition of (14), getting:
 $q? \cdot h = h \cdot r? \cdot (\mathbf{F} q?)$.

Summing up:

Let $(|h|) : \mathbf{W} \rightarrow \mathbf{T}$ abstract type \mathbf{T} by \mathbf{W} , and $q? = (|in_{\mathbf{T}} \cdot t?)$ be a test on \mathbf{T} . To obtain $p = \mathbf{wp}(|h|, q)$, that is, to ensure

$$(|h|) \cdot p? = q? \cdot (|h|)$$

search for the existence of $r? : \mathbf{F} \mathbf{T} \rightarrow \mathbf{F} \mathbf{T}$ such that

$$q? \cdot h = h \cdot r? \cdot (\mathbf{F} q?) \tag{15}$$

$$\mathbf{F} (|h|) \cdot w? = r? \cdot \mathbf{F} (|h|) \tag{16}$$

where $p? = (|in_{\mathbf{W}} \cdot w?)$ and $in_{\mathbf{W}} : \mathbf{F} \mathbf{W} \rightarrow \mathbf{W}$.



Calculating the **divide** step

Now that we can trust $\llbracket X \rrbracket$, let us calculate X :

$$p? \cdot (\llbracket h \rrbracket)^\circ \cdot \mathbf{ker} (\llbracket k \rrbracket) = \llbracket X \rrbracket$$

$$\Leftrightarrow \quad \{ \text{converses} \}$$

$$\mathbf{ker} (\llbracket k \rrbracket) \cdot (\llbracket h \rrbracket) \cdot p? = (\llbracket X^\circ \rrbracket)$$

$$\Leftrightarrow \quad \{ (\llbracket h \rrbracket) \cdot p? = q? \cdot (\llbracket h \rrbracket) \text{ (12)} \}$$

$$\mathbf{ker} (\llbracket k \rrbracket) \cdot q? \cdot (\llbracket h \rrbracket) = (\llbracket X^\circ \rrbracket)$$

$$\Leftarrow \quad \{ \text{fusion (13)} ; \text{functor } \mathbf{F} \}$$

$$\mathbf{ker} (\llbracket k \rrbracket) \cdot q? \cdot h = X^\circ \cdot \mathbf{F} \mathbf{ker} (\llbracket k \rrbracket) \cdot \mathbf{F} q?$$

Still far for having a closed formula for X — need to get rid of $\mathbf{F} \mathbf{ker} (\llbracket k \rrbracket) \cdot \mathbf{F} q?$...

Calculating the **divide** step



Removing **F q?** first:

$$\mathbf{ker} (\lvert k \rvert) \cdot q? \cdot h = X^\circ \cdot \mathbf{F ker} (\lvert k \rvert) \cdot \mathbf{F q?}$$

$$\Leftrightarrow \quad \{ (15): q? \cdot h = h \cdot r? \cdot (\mathbf{F q?}) \}$$

$$\mathbf{ker} (\lvert k \rvert) \cdot h \cdot r? \cdot (\mathbf{F q?}) = X^\circ \cdot \mathbf{F ker} (\lvert k \rvert) \cdot \mathbf{F q?}$$

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

$$\mathbf{ker} (\lvert k \rvert) \cdot h \cdot r? = X^\circ \cdot \mathbf{F ker} (\lvert k \rvert)$$

Next, we'll get rid of **F ker (|k|)**.

This will require equivalence **ker (|k|)** to be a congruence for algebra **h**, see the next slide.



Auxiliary results

Theorem

Let R be a congruence for algebra $h: \mathbf{F} A \rightarrow A$ of functor \mathbf{F} , that is

$$h \cdot (\mathbf{F} R) \subseteq R \cdot h \quad \text{i.e.} \quad y (\mathbf{F} R) x \Rightarrow (h y) R (h x) \quad (17)$$

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

$$R \cdot h = R \cdot h \cdot (\mathbf{F} R) \quad (18)$$

(Proof in the appendix.) \square

Thus initial algebra in is a congruence for kernel $\mathbf{ker} f$ where $f = \langle |k| \rangle$, since $\mathbf{ker} f = \langle |\mathbf{ker} f \cdot \text{in}| \rangle$ is an instance of (18), cf. $\mathbf{ker} f \cdot \text{in} = \mathbf{ker} f \cdot \text{in} \cdot (\mathbf{F} \mathbf{ker} f)$.

EXAMPLE: “same parity as” is a congruence for succ .

Calculating the **divide** step



We move on:

$$\begin{aligned} \ker (\downarrow k) \cdot h \cdot r? &= X^\circ \cdot \mathbf{F} \ker (\downarrow k) \\ \Leftrightarrow \quad \{ (18) \} \\ \ker (\downarrow k) \cdot h \cdot (\mathbf{F} \ker (\downarrow k)) \cdot r? &= X^\circ \cdot \mathbf{F} \ker (\downarrow k) \end{aligned}$$

Annoying: as $r?$ prevents cancellation, we have to assume a final side condition

$$\mathbf{F} (\ker (\downarrow k)) \cdot r? = r? \cdot \mathbf{F} (\ker (\downarrow k)) \quad (19)$$

whereby we get (after cancellation, converses):

$$X = r? \cdot h^\circ \cdot \ker (\downarrow k)$$

— another metaphor, cf. $X = r? \cdot ((\downarrow k) \cdot h)^\circ \cdot (\downarrow k)$



Example — quicksort

Clearly, the main role in the calculation is played by condition r on $\mathbf{F T}$ which, according to (15),

$$q? \cdot h = h \cdot r? \cdot (\mathbf{F} \ q?)$$

which is the weakest precondition for h to maintain q .

Let calculate it for the sorting example, where $h = [\text{nil}, \text{inord}]$, q checks for ordered lists and $\text{inord}(a, (x, y)) = x \# [a] \# y$, and $\mathbf{F} \ f = \text{id} + \text{id} \times (f \times f)$:

$$q? \cdot [\text{nil}, \text{inord}] = [\text{nil}, \text{inord}] \cdot r? \cdot (\text{id} + \text{id} \times (q? \times q?))$$

$$\Leftrightarrow \quad \{ \text{let } r? = \text{id} + \text{wpl?}; \text{ coproducts } \}$$

$$[q? \cdot \text{nil}, q? \cdot \text{inord}] = [\text{nil}, \text{inord} \cdot \text{wpl?} \cdot (\text{id} \times (q? \times q?))]$$

$$\Leftrightarrow \quad \{ \text{the empty list is trivially ordered} \}$$

$$q? \cdot \text{inord} = \text{inord} \cdot \text{wpl?} \cdot (\text{id} \times (q? \times q?))$$



Example — quicksort

Going pointwise:

$$\begin{aligned}
 & q(x \# [a] \# y) \\
 \Leftrightarrow & \quad \{ \text{pointwise definition of ordered lists} \} \\
 & \left\{ \underbrace{(q x) \wedge (q y) \wedge \langle \forall b : b \in_{\mathbf{T}} x : b \leq a \rangle \wedge \langle \forall b : b \in_{\mathbf{T}} y : a \leq b \rangle}_{wpl(a, (x, y))} \right\}
 \end{aligned}$$

Now calculate (relational) coalgebra

$$\begin{aligned}
 X : \mathbf{T} &\rightarrow 1 + \mathbf{T} \times (\mathbf{T} \times \mathbf{T}) \\
 X &= (id + wpl?) \cdot (bag \cdot [nil, inord])^\circ \cdot bag
 \end{aligned} \tag{20}$$

(a metaphor):



Example — quicksort

$$X = (id + wpl?) \cdot (bag \cdot [nil, inord])^\circ \cdot bag$$

$$\Leftrightarrow \quad \{ \text{take converses and let } X^\circ = [X1^\circ, X2^\circ] \}$$

$$[X1^\circ, X2^\circ] = bag^\circ \cdot (bag \cdot [nil, inord]) \cdot (id + wpl?)$$

$$\Leftrightarrow \quad \{ bag^\circ \cdot bag = Perm; Perm.nil = nil; \text{converses} \}$$

$$\begin{cases} X1 = nil^\circ \\ X2 = wpl? \cdot inord^\circ \cdot Perm \end{cases}$$

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

Example — quicksort



$$\begin{cases} X1 = \text{nil}^\circ \\ X2 = \text{wpl?} \cdot \text{inord}^\circ \cdot \text{Perm} \end{cases}$$

\Leftrightarrow { go pointwise }

$$\begin{cases} () X1 x \Leftrightarrow x = [] \\ (a, (y, z)) X2 x \Leftrightarrow (a, (y, z)) (\text{wpl?} \cdot \text{inord}^\circ \cdot \text{Perm}) x \end{cases}$$

where the second case unfolds to:

$$(a, (y, z)) X2 x \Leftrightarrow \text{wpl} (a, (y, z)) \wedge (y \# [a] \# z) \text{Perm} x$$

We still need to check the other side conditions.

Not difficult but not immediate (omitted).

Wrapping up



Quicksort derivation takes longer than 2 pages...

Calculation of the general case in which a **metaphorism** is refined into a **hylomorphism** by *changing the virtual data structure*.

Metaphorism identified as a class of relational specifications.

Currently working out the **text formatting** metaphorism.

Greedy implementation from $M = (f^\circ \cdot g) \upharpoonright R$ where R includes more than one optimization criterion, e.g.

- fixed maximum number of characters per output line
- maximize number of words per line (minimize white space)

Metaphorism — exploratory concept (recent research topic).

Annex



Proof of (17), (18) concerning congruences:

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

$$\Leftrightarrow \{ R \cdot h \subseteq R \cdot h \cdot (\mathbf{F} R) \text{ holds by } \subseteq id (\mathbf{F} R), \text{ since } \subseteq id R \}$$

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

$$\Leftrightarrow \{ \text{lower } R \text{ can be cancelled, see below } \}$$

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

□

Last step can be justified by assuming the function k_R which maps every object to its R -equivalence class — $R = \ker k_R$.

Then (next slide):



Annex

For any suitably typed relations X and Y :

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

$$\Leftrightarrow \{ \text{inline } R = \mathbf{ker} \ k_R \}$$

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

$$\Leftrightarrow \{ \mathbf{ker} \ k_R = k_R^\circ \cdot k_R ; \text{shunting} \}$$

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

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

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

$$\Leftrightarrow \{ \text{shunting} ; R = \mathbf{ker} \ k_R \}$$

$$X \subseteq R \cdot Y$$

□



References

- G. Hutton and E. Meijer. Back to basics: Deriving representation changers functionally. *Journal of Functional Programming*, 6(1): 181–188, 1996.
- G. Lakoff and M. Johnson. *Metaphors we live by*. University of Chicago Press, Chicago, 1980. ISBN 978-0-226-46800-6.
- I.A. Richards. *The Philosophy of Rhetoric*. Oxford University Press, 1936.
- D. Swierstra and O. de Moor. Virtual data structures. In B. Möller, H. Partsch, and S. Schuman, editors, *Formal Program Development*, volume 755 of *LNCS*, pages 355–371. Springer, 1993. ISBN 978-3-540-57499-6. doi: 10.1007/3-540-57499-9_26. URL http://dx.doi.org/10.1007/3-540-57499-9_26.