Metaphorisms in Programming

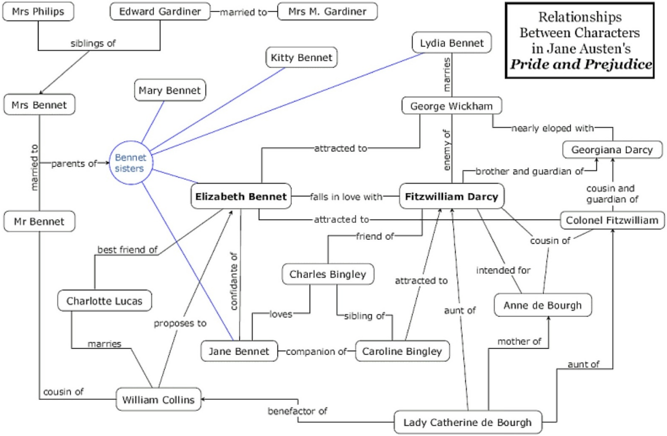
J.N. Oliveira

RAMiCS 2015
Braga
September 2015

INESC TEC & University of Minho

Grant PTDC/EIA-CCO/122240/2010

Everything is a relation



(Source: Wikipedia, **Pride and Prejudice**, by Jane Austin, 1813.)

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 building block of semantics
- *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:

$$
\begin{tikzcd}
P \arrow[loop left, "chngt'"] \arrow[dr, "corrupt"'] & & D \arrow[ll, "corrupt^{\circ} \cdot dirty"'] \arrow[dl, "dirty"] \arrow[loop right, "chng"] \\
& \mathbb{B} &
\end{tikzcd}
$$

$dirty\ (chng\ x) = False$ 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"

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

where functions $f : \mathbf{T} \to A$ and $g : \mathbf{V} \to A$ extract the common **attribute** (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**

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

— 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"...)

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

$$compiler = code\_generator \cdot pretty\_printer^\circ$$

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

$$sort = ordered \cdot (bag^\circ \cdot bag)$$

$$\begin{tikzcd} aaaabbc \arrow[dr, "bag"'] & & abacaba \arrow[dl, "bag"] \\ & a4b2c1 & \end{tikzcd}$$

?

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):

$$M = (f^\circ \cdot g) \upharpoonright R \qquad \begin{tikzcd} & \mathbf{T} & \\ \mathbf{T} \arrow[ur, "R"] \arrow[dr, "f"'] & & \mathbf{V} \arrow[ul, "M"'] \arrow[ll, "f^\circ \cdot g"'] \arrow[dl, "g"] \\ & A & \end{tikzcd} \tag{2}$$

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

Text formatting metaphorism

Formatted text is a sequence of text lines,

$$\begin{tikzcd}
{[String]} \arrow[dr, "(\gg\!= words)"'] & & String \arrow[ll, "Format"'] \arrow[dl, "words"] \\
& {[String]} &
\end{tikzcd}$$

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,

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

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* of 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

$$M = (f^\circ \cdot f) \restriction R \qquad (4)$$

— 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 $f\ y' = f\ x$ (this is the **optimization**) recalling $S \restriction R = S \cap R \mathbin{/} S^\circ$, that is:
  $$\langle \forall\ y'\ :\ f\ y' = f\ x :\ y\ R\ y' \rangle$$

— recall *Programming from Galois connections*, RAMiCS 2011 (Mu and Oliveira, 2012).

Inductive metaphorisms

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

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

$(\!|h|\!)$ *expresses a* **fold** *or* **catamorphism** *over algebra* h.

In this case, for surjective $f = (\!|k|\!)$

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

holds, meaning that metaphorism $M = \mathbf{ker}\ f \upharpoonright R$ can be implemented by calculating $M = (\!|\mathbf{ker}\ f \cdot \mathsf{in}|\!) \upharpoonright R$ — cf. "greedy" theorems, etc

Calculating metaphorisms

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

$$M = (\mathbf{ker}\ (\!|k|\!)) \upharpoonright R = ((\!|k|\!)^\circ \upharpoonright R) \cdot (\!|k|\!) \tag{6}$$

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).

NB: the (functional) composition of a **fold** followed by an **unfold** has been known as a **metamorphism** (Martin Erwig).

AoP, pp.154–155

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

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

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

$$\mathit{flatten} \ = \ (\![\mathit{nil}, \mathit{join}]\!),$$

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

In outline, the derivation of quicksort is

$$\begin{array}{ll}
& \mathit{ordered} \cdot \mathit{perm} \\
\supseteq & \{\text{since } \mathit{flatten} \text{ is a function}\} \\
& \mathit{ordered} \cdot \mathit{flatten} \cdot \mathit{flatten}^\circ \cdot \mathit{perm} \\
= & \{\text{claim: } \mathit{ordered} \cdot \mathit{flatten} = \mathit{flatten} \cdot \mathit{inordered} \text{ (see below)}\} \\
& \mathit{flatten} \cdot \mathit{inordered} \cdot \mathit{flatten}^\circ \cdot \mathit{perm} \\
= & \{\text{converses}\} \\
& \mathit{flatten} \cdot (\mathit{perm} \cdot \mathit{flatten} \cdot \mathit{inordered})^\circ \\
\supseteq & \{\text{fusion, for an appropriate definition of } \mathit{split}\} \\
& \mathit{flatten} \cdot (\![\mathit{nil}, \mathit{split}^\circ]\!)^\circ.
\end{array}$$

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

$$\mathit{inordered} \ = \ (\![\mathit{null}, \mathit{fork} \cdot \mathit{check}]\!)$$

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

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

The relation *intree* is the membership test for trees. Introducing $\mathsf{F}f = f \times \mathit{id} \times f$ for brevity, the proviso for the fusion step in the above calculation is

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

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

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

We now reason:

$$\begin{array}{ll}
& \mathit{perm} \cdot \mathit{flatten} \cdot \mathit{fork} \cdot \mathit{check} \\
= & \{\text{catamorphisms, since } \mathit{flatten} = (\![\mathit{nil}, \mathit{join}]\!)\} \\
& \mathit{perm} \cdot \mathit{join} \cdot \mathsf{F}\,\mathit{flatten} \cdot \mathit{check} \\
= & \{\text{claim: } \mathsf{F}\,\mathit{flatten} \cdot \mathit{check} = \mathit{check}' \cdot \mathsf{F}\,\mathit{flatten}\} \\
& \mathit{perm} \cdot \mathit{join} \cdot \mathit{check}' \cdot \mathsf{F}\,\mathit{flatten} \\
= & \{\text{claim: } \mathit{perm} \cdot \mathit{join} = \mathit{perm} \cdot \mathit{join} \cdot \mathsf{F}\,\mathit{perm}\} \\
& \mathit{perm} \cdot \mathit{join} \cdot \mathsf{F}\,\mathit{perm} \cdot \mathit{check}' \cdot \mathsf{F}\,\mathit{flatten} \\
= & \{\text{claim: } \mathsf{F}\,\mathit{perm} \cdot \mathit{check}' = \mathit{check}' \cdot \mathsf{F}\,\mathit{perm}\text{; functors}\} \\
& \mathit{perm} \cdot \mathit{join} \cdot \mathit{check}' \cdot \mathsf{F}(\mathit{perm} \cdot \mathit{flatten}) \\
\supseteq & \{\text{taking } \mathit{split} \subseteq \mathit{check}' \cdot \mathit{join}^\circ \cdot \mathit{perm}\} \\
& \mathit{split}^\circ \cdot \mathsf{F}(\mathit{perm} \cdot \mathit{flatten}).
\end{array}$$

Formal proofs of the three claims are left as exercises. In words, *split* is defined by the rule that if $(y, a, z) = \mathit{split}\ x$, 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:

$$\mathit{split} \ = \ (\![\mathit{base}, \mathit{step}]\!) \cdot \mathit{embed}.$$

The fusion conditions are:

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

These conditions are satisfied by taking

$$\begin{array}{rcl}
\mathit{base}\ a & = & ([\,], a, [\,]) \\
\mathit{step}\,(a, (x, b, y)) & = & \begin{cases} ([a] +\!\!+ x, b, y), & \text{if } aRb \\ (x, b, [a] +\!\!+ y), & \text{otherwise.} \end{cases}
\end{array}$$

Finally, appeal to the hylomorphism theorem gives that $X = \mathit{flatten} \cdot (\![\mathit{nil}, \mathit{split}^\circ]\!)^\circ$ 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 $\llparenthesis h \rrparenthesis \cdot \llparenthesis h \rrparenthesis^\circ = id$ of a **surjective** fold over **another** datatype $\mathbf{W}$:

$$
\begin{tikzcd}
 & \mathbf{W} \arrow[dl, "\llparenthesis h \rrparenthesis"'] & & & \\
\mathbf{T} & & \mathbf{T} \arrow[ll, "id = img\ \llparenthesis h \rrparenthesis"] \arrow[ul, "\llparenthesis h \rrparenthesis^\circ"'] \arrow[dr, "\llparenthesis k \rrparenthesis"'] & & \mathbf{T} \arrow[ll, "\mathbf{ker}\ \llparenthesis k \rrparenthesis"'] \arrow[dl, "\llparenthesis k \rrparenthesis"] \\
 & & & A &
\end{tikzcd}
\tag{7}
$$

Typically, choose polynomial $\mathbf{W}$ of degree higher than $\mathbf{T}$, 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 $\mathbf{T}$ to type $\mathbf{W}$:

$$
\begin{tikzcd}
 & \mathbf{W} \arrow[dl, "(|h|)"'] \arrow[drr, "(|k|)\cdot(|h|)"'] & & \mathbf{T} \arrow[ll, "N=(|h|)^{\circ}\cdot\mathbf{ker}\ (|k|)"'] \arrow[dl, "(|k|)"] \\
\mathbf{T} & & & \\
 & & A &
\end{tikzcd}
\tag{8}
$$

$\mathbf{W}$ is the (**virtual**) data type chosen to command a **divide & conquer** algorithmic implementation.

The aim is to convert N into an **unfold**, say $[\!(X)\!]$ so that we get a hylomorphism 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:

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

Special case of optimization (shrinking)

Kernels are reflexive and therefore entire. Thus:

$$M = \mathbf{ker}\ (\!|k|\!) \upharpoonright (p? \cdot \top) = p? \cdot \mathbf{ker}\ (\!|k|\!)$$

Example — the **sorting metaphorism**:

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

Equivalence $Perm = \mathbf{ker}\ bag$ is the metaphor

Function bag computes the bag of elements of a finite list.

Pointwise:

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

Sorting example (details)

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

  $$inord\ (a, (x, y)) = x +\!\!+ [a] +\!\!+ y$$

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

Shrinking metaphorisms into hylomorphisms

$$
\begin{array}{ll}
& M = q? \cdot id \cdot \mathbf{ker}\ (|k|) \\
\Leftrightarrow & \qquad \{\ (|h|) \cdot (|h|)^\circ = id \text{ for surjective } (|h|)\ \} \\
& M = q? \cdot (|h|) \cdot (|h|)^\circ \cdot \mathbf{ker}\ (|k|) \\
\Leftrightarrow & \qquad \{\ \text{switch to } p? \text{ such that } (|h|) \cdot p? = q? \cdot (|h|)\ \} \\
& M = (|h|) \cdot \underbrace{p? \cdot (|h|)^\circ \cdot \mathbf{ker}\ (|k|)}_{[\![(X)]\!]}
\end{array}
$$

Unfold $[\![(X)]\!]$ is the (relational) **divide** step in $\mathbf{T} \xleftarrow{(|h|)} \mathbf{W} \xleftarrow{[\![(X)]\!]} \mathbf{T}$

$[\![(X)]\!]$ is in fact a new **metaphorism**, now between **W** and **T**.

$$
\begin{tikzcd}
\mathbf{W} \arrow[dr, "(|k|)\cdot(|h|)"'] & & \mathbf{T} \arrow[ll, "{[\![(X)]\!]}"'] \arrow[dl, "(|k|)"] \\
& A &
\end{tikzcd}
$$

Weakest preconditions

Before proceeding to calculating **divide** step $[\![X]\!]$, we observe that $p?$ such that

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

holds is the **weakest pre-condition** for $(\!|h|\!)$ to ensure $q?$ on its output, that is $(q \cdot f)?$ — recall the standard GC:

$$\underbrace{\rho\,(f \cdot p?)}_{\mathbf{sp}(f,p)} \subseteq q? \quad \Leftrightarrow \quad f \cdot p? \subseteq q? \cdot f \quad \Leftrightarrow \quad p? \subseteq \underbrace{\delta\,(q? \cdot f)}_{\mathbf{wp}(f,q)} \tag{10}$$

NB:

$$\delta\,(q? \cdot f) = (q \cdot f)? \tag{11}$$

Weakest precondition algebra

The following $\mathbf{wp}(\cdot,\cdot)$-universal property,

$$f \cdot p? = q? \cdot f \quad \Leftrightarrow \quad p = q \cdot f \tag{12}$$

enables a "logic-free" calculation of weakest preconditions.

So, given f and post-condition q, replacing $q? \cdot f$ by $f \cdot p?$ is always possible (cf. **existence**) and such p is **unique**.

Also, for some q:

$$\mathbf{ker}\ f \cdot p? = p? \cdot \mathbf{ker}\ f \quad \Leftarrow \quad p = q \cdot f \tag{13}$$

Relational proofs for (12) and (13) follow.

Proof of (12)

Step ($\Rightarrow$):

$$
\begin{array}{ll}
 & p = q \cdot f \\
\Leftrightarrow & \quad \{ \text{ unfolding } p? = (q \cdot f)? \ \} \\
 & p? \subseteq (q \cdot f)? \ \wedge \ (q \cdot f)? \subseteq p? \\
\Leftrightarrow & \quad \{ \ sp \dashv wp \text{ GC } (10) \ ; \ (11) \ \} \\
 & f \cdot p? \subseteq q? \cdot f \ \wedge \ \delta\,(q? \cdot f) \subseteq p? \\
\Leftarrow & \quad \{ \ f \cdot p? = q? \cdot f \text{ assumed (twice) } \} \\
 & \delta\,(f \cdot p?) \subseteq p? \\
\Leftrightarrow & \quad \{ \text{ domain GC ; shunting test } p? \ \} \\
 & f \cdot p? \subseteq \top \\
\Leftrightarrow & \quad \{ \text{ trivia } \} \\
 & \mathit{true}
\end{array}
$$

$\Box$

Proof of (12)

Step ($\Leftarrow$):

$$
\begin{array}{ll}
& f \cdot p? = q? \cdot f \;\Leftarrow\; p = q \cdot f \\
\Leftrightarrow & \qquad \{\ (11)\ ;\ \text{substitution, i.e. cancellation (10)}\ \} \\
& q? \cdot f \;\subseteq\; f \cdot (q \cdot f)? \\
\Leftrightarrow & \qquad \{\ R = R \cdot \delta\, R\ ;\ (11)\ \} \\
& q? \cdot f \cdot \delta\,(q? \cdot f) \;\subseteq\; f \cdot (\delta\, q? \cdot f) \\
\Leftrightarrow & \qquad \{\ \text{domain } \delta\,(q? \cdot f)\ \} \\
& q? \cdot f \;\subseteq\; f \\
\Leftarrow & \qquad \{\ \text{monotonicity of } (\cdot f)\ \} \\
& q? \;\subseteq\; id \\
\Leftrightarrow & \qquad \{\ q? \text{ is a test}\ \} \\
& true
\end{array}
$$

$\square$

Proof of (13)

$$
\begin{array}{ll}
& \mathbf{ker}\ f \cdot p? \\
= & \qquad \{\ \mathbf{ker}\ f = f^\circ \cdot f\ ;\ (12)\ \text{since}\ p = q \cdot f\ \ \} \\
& f^\circ \cdot q? \cdot f \\
= & \qquad \{\ \text{converses}\ ;\ \text{partial identities}\ \ \} \\
& (q? \cdot f)^\circ \cdot f \\
= & \qquad \{\ \text{again}\ (12)\ \ \} \\
& (f \cdot p?)^\circ \cdot f \\
= & \qquad \{\ \text{converses}\ ;\ \text{kernels}\ \ \} \\
& p? \cdot \mathbf{ker}\ f
\end{array}
$$

$\square$

Shrinking metaphorisms into hylomorphisms

In the case of

$$(\!| h |\!) \cdot p? = q? \cdot (\!| h |\!) \qquad \begin{tikzcd} \mathbf{W} \arrow[d, "(\!| h |\!)"'] & \mathbf{W} \arrow[l, "p?"'] \arrow[d, "(\!| h |\!)"] \\ \mathbf{T} & \mathbf{T} \arrow[l, "q?"] \end{tikzcd} \tag{14}$$

above, test $p? : \mathbf{W} \leftarrow \mathbf{W}$ will be of shape

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

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

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

Shifting the metaphor

Given h and q, solving

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

for p amounts to finding conditions for reducing both $(\!|h|\!) \cdot p?$ and $q? \cdot (\!|h|\!)$ to the same fold $(\!|R|\!)$ over $\mathbf{W}$, thanks to relational **fold-fusion**:

$$Q \cdot (\!|S|\!) = (\!|R|\!) \quad \Leftarrow \quad Q \cdot S = R \cdot \mathbf{F}\ Q \qquad (15)$$

a) Reducing one side (15) :

$$q? \cdot (\!|h|\!) = (\!|R|\!) \quad \Leftarrow \quad q? \cdot h = R \cdot (\mathbf{F}\ q?) \qquad (16)$$

b) Reducing the other side:

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

Shifting the metaphor

$$
\begin{array}{cl}
\Leftrightarrow & \quad \{ \text{ inline } p? = (\!|\mathsf{in}_{\mathbf{W}} \cdot w?|\!) \ \} \\
& (\!|h|\!) \cdot (\!|\mathsf{in}_{\mathbf{W}} \cdot w?|\!) = (\!|R|\!) \\
\Leftarrow & \quad \{ \text{ fusion (15) } \} \\
& (\!|h|\!) \cdot \mathsf{in}_{\mathbf{W}} \cdot w? = R \cdot \mathbf{F}\ (\!|h|\!) \\
\Leftrightarrow & \quad \{ \text{ cancellation: } (\!|h|\!) \cdot \mathsf{in}_{\mathbf{W}} = h \cdot \mathbf{F}\ (\!|h|\!) \ \} \\
& h \cdot \mathbf{F}\ (\!|h|\!) \cdot w? = R \cdot \mathbf{F}\ (\!|h|\!) \\
\Leftrightarrow & \quad \{ \text{ switch to } r? \text{ such that } \mathbf{F}\ (\!|h|\!) \cdot w? = r? \cdot \mathbf{F}\ (\!|h|\!) \ \} \\
& h \cdot r? \cdot \mathbf{F}\ (\!|h|\!) = R \cdot (\mathbf{F}\ (\!|h|\!)) \\
\Leftarrow & \quad \{ \text{ Leibniz } \} \\
& h \cdot r? = R
\end{array}
$$

Shifting the metaphor

Thus $R = h \cdot r?$ ensures proviso (14), which we can replace in the other proviso — side condition of fusion step (16), getting

$$q? \cdot h = h \cdot r? \cdot \mathbf{F}\ q? \qquad \begin{tikzcd} \mathbf{T} \arrow[d, "q?"'] & & \mathbf{F}\ \mathbf{T} \arrow[ll, "h"'] \arrow[d, "\mathbf{F}\ q?"] \\ \mathbf{T} & \mathbf{F}\ \mathbf{T} \arrow[l, "h"] & \mathbf{F}\ \mathbf{T} \arrow[l, "r?"] \end{tikzcd} \tag{17}$$

while not forgetting:

$$\mathbf{F}\ (|h|) \cdot w? = r? \cdot \mathbf{F}\ (|h|) \qquad \begin{tikzcd} \mathbf{F}\ \mathbf{W} \arrow[d, "\mathbf{F}\ (|h|)"'] & \mathbf{F}\ \mathbf{W} \arrow[l, "w?"'] \arrow[d, "\mathbf{F}\ (|h|)"] \\ \mathbf{F}\ \mathbf{T} & \mathbf{F}\ \mathbf{T} \arrow[l, "r?"] \end{tikzcd} \tag{18}$$

Example (sorting)

Clearly, condition r on $\mathbf{F}\ \mathbf{T}$ in proviso (17),

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

is the weakest precondition for h to maintain q.

Let us calculate r for the **sorting** metaphorism, where (recall) q checks for ordered lists, $(\!|h|\!) = \mathit{flatten}$, i.e.

$$\begin{array}{l} \mathit{flatten}\ \mathit{empty} = \mathit{nil} \\ \mathit{flatten}\ (\mathit{fork}\ (a, (x, y)) = \mathit{inord}\ (a, (\mathit{flatten}\ x, \mathit{flatten}\ y)) \\ \quad \mathbf{where}\ \mathit{inord}\ (a, (x, y)) = x +\!\!+ [a] +\!\!+ y \end{array}$$

and thus

$$\begin{array}{rcl} h & = & [\mathit{nil}, \mathit{inord}] \\ \mathbf{F}\ f & = & \mathit{id} + \mathit{id} \times (f \times f) \end{array}$$

Example (sorting)

We get:

$$
\begin{array}{ll}
& q? \cdot [nil, inord] = [nil, inord] \cdot r? \cdot (id + id \times (q? \times q?)) \\
\Leftrightarrow & \quad \{ \text{ switch to } s \text{ such that } r? = id + s?\text{; coproducts } \} \\
& [q? \cdot nil, q? \cdot inord] = [nil, inord \cdot s? \cdot (id \times (q? \times q?))] \\
\Leftrightarrow & \quad \{ \text{ the empty list is trivially ordered } \} \\
& q? \cdot inord = inord \cdot s? \cdot (id \times (q? \times q?)) \\
\Leftrightarrow & \quad \{ \text{ universal property (12) } \} \\
& (q \cdot inord)? = s? \cdot (id \times (q? \times q?))
\end{array}
$$

Example — quicksort

Knowing q and *inord* we easily spot s going pointwise:

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

Altogether:

$$
\begin{array}{l}
\mathit{inord}\ (a, (x, y)) = x +\!\!+ [a] +\!\!+ y \\
\textbf{pre}\ (a, (x, y)) = \langle \forall\ b\ :\ b\ \epsilon_{\mathbf{T}}\ x :\ b \leqslant a \rangle \wedge \langle \forall\ b\ :\ b\ \epsilon_{\mathbf{T}}\ y :\ a \leqslant b \rangle
\end{array}
$$

Calculating the **divide** step

Assuming (17) and (18), let us calculate X:

$$
\begin{array}{ll}
 & p? \cdot (\!|h|\!)^\circ \cdot \mathbf{ker}\ (\!|k|\!) = [\![X]\!] \\
\Leftrightarrow & \qquad \{ \text{ converses } \} \\
 & \mathbf{ker}\ (\!|k|\!) \cdot (\!|h|\!) \cdot p? = (\!|X^\circ|\!) \\
\Leftrightarrow & \qquad \{ \ (\!|h|\!) \cdot p? = q? \cdot (\!|h|\!) \text{ assumed — cf. (14) } \} \\
 & \mathbf{ker}\ (\!|k|\!) \cdot q? \cdot (\!|h|\!) = (\!|X^\circ|\!) \\
\Leftarrow & \qquad \{ \text{ fusion (15) ; functor } \mathbf{F} \ \} \\
 & \mathbf{ker}\ (\!|k|\!) \cdot q? \cdot h = X^\circ \cdot \mathbf{F}\ \mathbf{ker}\ (\!|k|\!) \cdot \mathbf{F}\ q?
\end{array}
$$

We are far for having a closed formula for X — how do we get rid of term $\mathbf{F}\ \mathbf{ker}\ (\!|k|\!) \cdot \mathbf{F}\ q?$?

Calculating the **divide** step

Removing $\mathbf{F}\ q?$ first:

$$
\begin{array}{ll}
 & \mathbf{ker}\ (\!|k|\!) \cdot q? \cdot h = X^{\circ} \cdot \mathbf{F}\ \mathbf{ker}\ (\!|k|\!) \cdot \mathbf{F}\ q? \\
\Leftrightarrow & \quad \{\ \text{proviso (17):}\ q? \cdot h = h \cdot r? \cdot \mathbf{F}\ q?\ \} \\
 & \mathbf{ker}\ (\!|k|\!) \cdot h \cdot r? \cdot \mathbf{F}\ q? = X^{\circ} \cdot \mathbf{F}\ \mathbf{ker}\ (\!|k|\!) \cdot \mathbf{F}\ q? \\
\Leftarrow & \quad \{\ \text{Leibniz}\ \} \\
 & \mathbf{ker}\ (\!|k|\!) \cdot h \cdot r? = X^{\circ} \cdot \mathbf{F}\ \mathbf{ker}\ (\!|k|\!)
\end{array}
$$

Next, we'll get rid of $\mathbf{F}\ \mathbf{ker}\ (\!|k|\!)$.

This will require equivalence $\mathbf{ker}\ (\!|k|\!)$ to be a **congruence** for algebra h, see the next slide.

Auxiliary results

Theorem

Let R *be a* **congruence** *for algebra* $h : \mathbf{F}\ A \to A$*, that is*

$$h \cdot (\mathbf{F}\ R) \subseteq R \cdot h \qquad i.e. \qquad y\ (\mathbf{F}\ R)\ x \Rightarrow (h\ y)\ R\ (h\ x) \tag{19}$$

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

$$R \cdot h = R \cdot h \cdot (\mathbf{F}\ R) \tag{20}$$

(Proof in the appendix.) $\square$

Example: for $f = (\!|k|\!)$, $\mathbf{ker}\ f$ is congruence for **initial** algebra in, since $\mathbf{ker}\ f = (\!|\mathbf{ker}\ f \cdot \mathsf{in}|\!)$ is an instance of (20), cf.
$\mathbf{ker}\ f \cdot \mathsf{in} = \mathbf{ker}\ f \cdot \mathsf{in} \cdot (\mathbf{F}\ \mathbf{ker}\ f)$.

Example: "same parity as" is a congruence for *succ*.

Calculating the **divide** step

We move on:

$$
\begin{array}{ll}
& \mathbf{ker}\ (|k|) \cdot h \cdot r? = X^\circ \cdot \mathbf{F}\ \mathbf{ker}\ (|k|) \\
\Leftrightarrow & \quad \{\ (20)\ \} \\
& \mathbf{ker}\ (|k|) \cdot h \cdot (\mathbf{F}\ \mathbf{ker}\ (|k|)) \cdot r? = X^\circ \cdot \mathbf{F}\ \mathbf{ker}\ (|k|)
\end{array}
$$

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

$$
\mathbf{F}\ (\mathbf{ker}\ (|k|)) \cdot r? = r? \cdot \mathbf{F}\ (\mathbf{ker}\ (|k|)) \tag{21}
$$

whereby we get (after cancellation, converses):

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

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

Example (sorting)

Let us calculate (relational) **coalgebra**

$$
\begin{array}{l}
X : \mathbf{T} \rightarrow 1 + \mathbf{T} \times (\mathbf{T} \times \mathbf{T}) \\
X = (id + s?) \cdot (bag \cdot [nil, inord])^\circ \cdot bag
\end{array}
\tag{22}
$$

for the **sorting** metaphor:

$$
\begin{array}{ll}
& X = (id + s?) \cdot (bag \cdot [nil, inord])^\circ \cdot bag \\
\Leftrightarrow & \quad \{ \text{ take converses and let } X^\circ = [X1^\circ, X2^\circ] \ \} \\
& [X1^\circ, X2^\circ] = bag^\circ \cdot (bag \cdot [nil, inord]) \cdot (id + s?) \\
\Leftrightarrow & \quad \{ \ bag^\circ \cdot bag = Perm; \text{ coproducts } \ \} \\
& [X1^\circ, X2^\circ] = [Perm \cdot nil, Perm \cdot inord \cdot s?] \\
\Leftrightarrow & \quad \{ \ Perm \cdot nil = nil; \text{ converses } \ \} \\
& \left\{ \begin{array}{l} X1 = nil^\circ \\ X2 = s? \cdot inord^\circ \cdot Perm \end{array} \right.
\end{array}
$$

Example — quicksort

$\Leftrightarrow$ { go pointwise }

$$\left\{ \begin{array}{l} ()\ X1\ x \Leftrightarrow x = [\,] \\ (a,(y,z))\ X2\ x \Leftrightarrow (a,(y,z))\ (s? \cdot inord^{\circ} \cdot Perm)\ x \end{array} \right.$$

where the second line unfolds to:

$$(a,(y,z))\ X2\ x \Leftrightarrow s\ (a,(y,z)) \wedge (y +\!\!+ [a] +\!\!+ z)\ Perm\ x$$

Note the free choice of "pivot" a provided s holds.

NB: We still need to check the other side conditions — not difficult but not immediate (see the paper).

Wrapping up

Quicksort derivation takes longer than 2 pages...

Generic calculation of the refinement of a **metaphorism** 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 (19), (20) concerning congruences:

$$
\begin{array}{ll}
& R \cdot h = R \cdot h \cdot (\mathbf{F}\ R) \\
\Leftrightarrow & \quad \{\ R \cdot h\ \subseteq\ R \cdot h \cdot (\mathbf{F}\ R) \text{ holds by } id \subseteq \mathbf{F}\ R\text{, since } id \subseteq R\ \} \\
& R \cdot h \cdot (\mathbf{F}\ R)\ \subseteq\ R \cdot h \\
\Leftrightarrow & \quad \{\ \text{lower } R \text{ can be cancelled, see below}\ \} \\
& h \cdot (\mathbf{F}\ R)\ \subseteq\ R \cdot h
\end{array}
$$

$\Box$

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

Then (next slide):

Annex

For any suitably typed relations X and Y:

$$
\begin{array}{ll}
& R \cdot X \subseteq R \cdot Y \\
\Leftrightarrow & \quad \{ \text{ inline } R = \mathbf{ker}\ k_R \ \} \\
& \mathbf{ker}\ k_R \cdot X \subseteq \mathbf{ker}\ k_R \cdot Y \\
\Leftrightarrow & \quad \{ \ \mathbf{ker}\ k_R = k_R^\circ \cdot k_R \text{ ; shunting } \} \\
& k_R \cdot k_R^\circ \cdot k_R \cdot X \subseteq k_R \cdot Y \\
\Leftrightarrow & \quad \{ \ f \cdot f^\circ \cdot f = f \text{ (difunctionality) } \} \\
& k_R \cdot X \subseteq k_R \cdot Y \\
\Leftrightarrow & \quad \{ \text{ shunting ; } R = \mathbf{ker}\ k_R \ \} \\
& X \subseteq R \cdot Y
\end{array}
$$

$\square$

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.

S.-C. Mu and J.N. Oliveira. Programming from Galois connections. *JLAP*, 81(6):680–704, 2012. doi: 10.1016/j.jlap.2012.05.003. .

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`.