On the “pointfree transform”: from *description* to *calculation* and back

ALFA **LERNET** ***kick-ff meeting,***

U. Minho, Braga

J.N. Oliveira

DI/U.Minho Braga, Portugal

Problem-solving strategy

Software technology is becoming a mature discipline in its (however late) adoption of the **universal problem solving** strategy (UPS) which one is taught at school:

- **understand** your problem
- build a mathematical **model** of it
- **reason** in such a model
- upgrade your model, if necessary
- **calculate** a final solution and implement it.

School maths UPS example

The problem:

My three children were born at a 3 year interval rate. Altogether, they are as old as me. I am 48. How old are they?

The model:

$$x + (x + 3) + (x + 6) \;=\; 48$$

The calculation:

School maths UPS example

The calculation:

$$3x + 9 = 48$$

$\equiv$ { "al-djabr" rule }

$$3x = 48 - 9$$

$\equiv$ { "al-hatt" rule }

$$x = 16 - 3$$

The solution:

$$\begin{array}{rcl} x & = & 13 \\ x+3 & = & 16 \\ x+6 & = & 19 \end{array}$$

UPS sophistication

Only the underlying mathematics changes:

- from simple **arithmetics** at primary school to
- systems of **linear** equations, then to
- **differential/integral** equations
- eventually: **software** calculi

Useful calculation rules are forever, eg.:

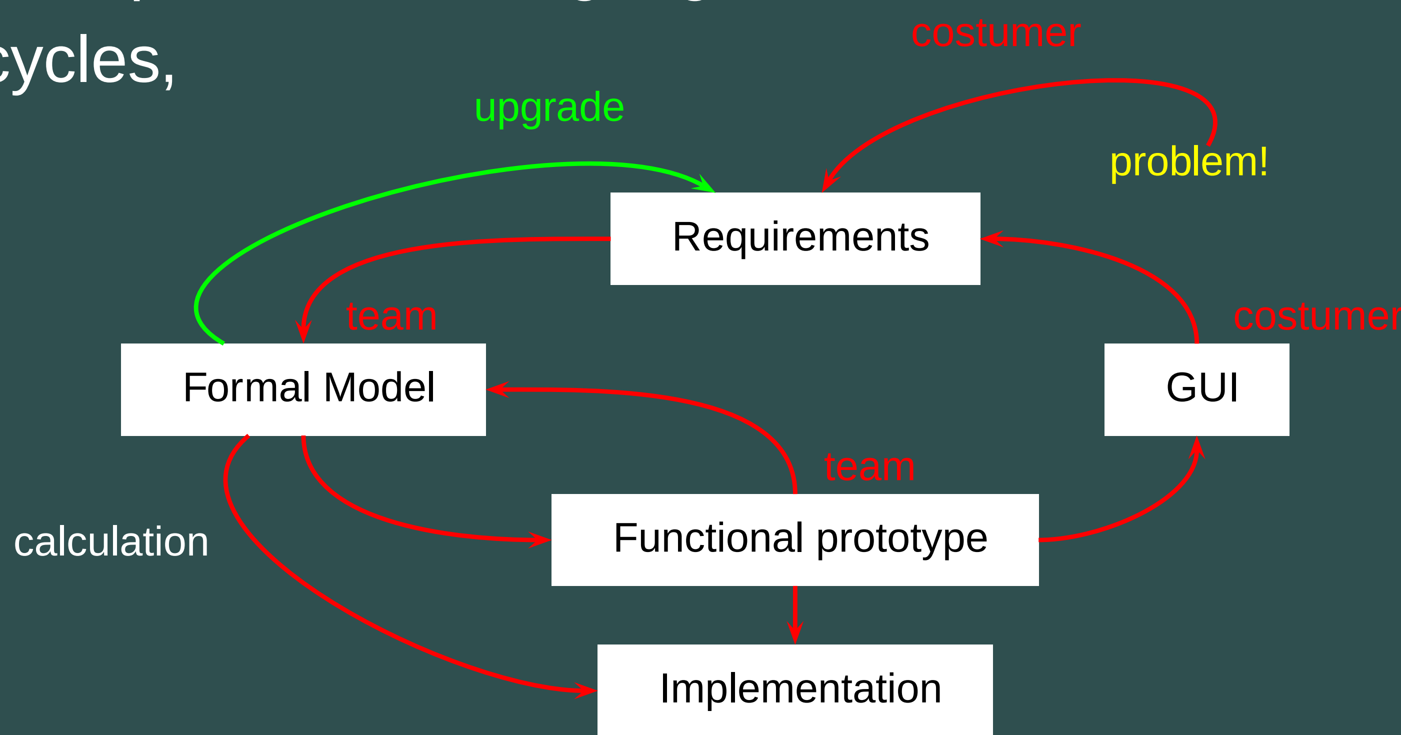
$$x - z \leq y \quad \equiv \quad x \leq z + y$$

cf. Al-Khowarizm's al-jabr rule (9c)

Galois connections (19c), etc

Formal methods and the UPS

- Formal Methods are 40 years old
- Formal specification languages, refinement calculi, life-cycles,



However: how much of all this is to be left for posterity?

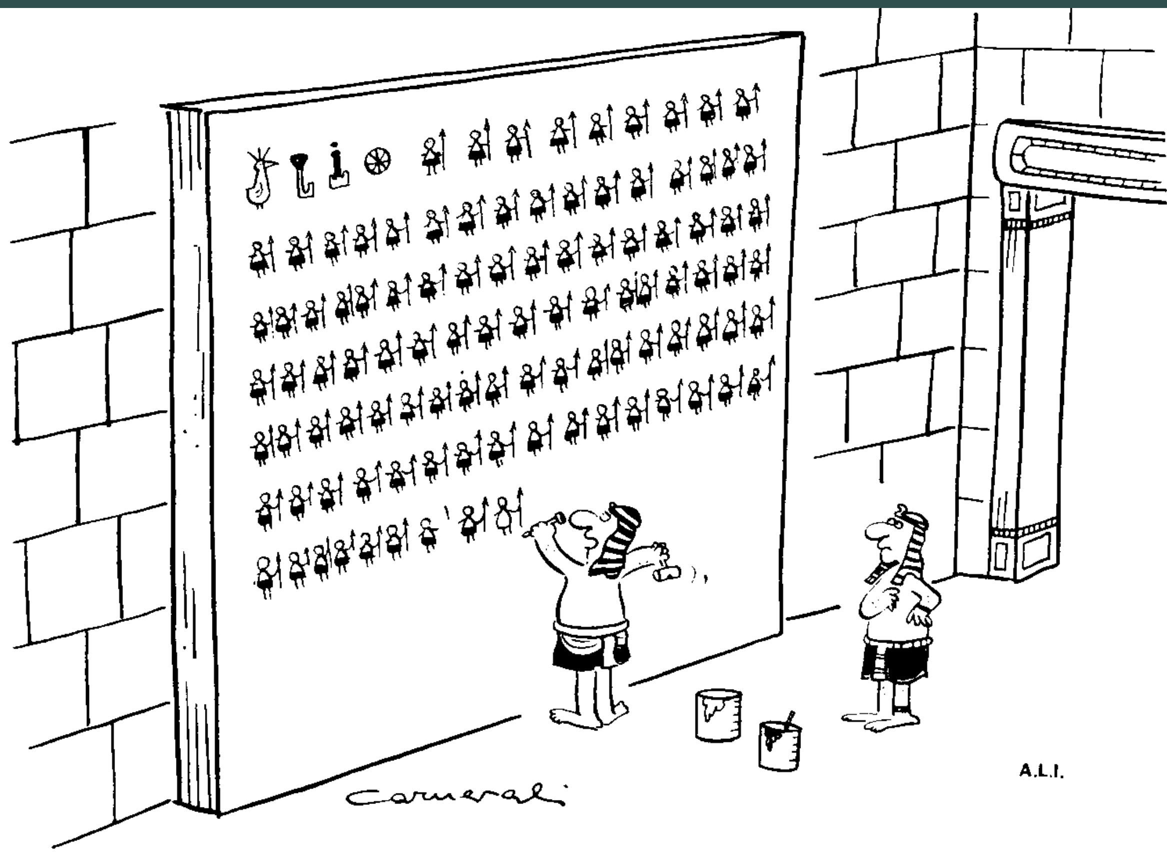
UPS challenges

A “notation problem”:

- mathematical modelling requires descriptive notations, therefore:
  - intuitive
  - domain-specific
- calculations require elegant notations, therefore:
  - simple and compact
  - generic
  - cryptic, otherwise uneasy to manipulate

Recall Dijsktra’s definition : elegant $\equiv$ simple and remarkably effective

Why formal / elegant notations?



Are you sure there isn't a simpler means of writing
'The Pharaoh had 10,000 soldiers?'

Trend for notation economy

Well-known throughout the history of maths — a kind of “natural language **implosion**” — particularly visible in the syncopated phase (16c), eg.

.40.p̃.2.ce. son yguales a .20.co

(P. Nunes, Coimbra, 1567) for nowadays $40 + 2x^2 = 20x$, or

B 3 in A quad - D plano in A + A cubo æquatur Z solido

(F. Viète, Paris, 1591) for nowadays $3BA^2 - DA + A^3 = Z$

Descriptive vs cryptic PLs

Attempts of the past:

- COBOL - the natural language description dream
- PASCAL - almost there for algorithmic description, not so good in recursive data abstraction
- APL - too cryptic for descriptive purposes
- Backus' FP - cryptic but pretty close on the calculation side

Currently

- JAVA, C$^{\{++,\#,\ldots\}}$ - powerful but, how to we reason about these?
- HASKELL - elegant and powerful (if not misused)
- VDM/Z - formal abstract modelling at last, but too set-theoretic (proofs grow exponentially complex)

Not enough:

Maths notations often require **transforms** for calculation purposes, eg. the Laplace transform:

Laplace transform

t-space | s-space

Given problem

$$y'' + 4y' + 3y = 0$$
$$y(0) = 3$$
$$y'(0) = 1$$

→

Subsidiary equation

$$s^2Y + 4sY + 3Y = 3s + 13$$

↓

Solution of subs. equation

$$Y = \frac{-2}{s+3} + \frac{5}{s+1}$$

←

Solution of given problem

$$y(t) = -2e^{-3t} + 5e^{-t}$$

A transform for set-theory

An old idea:

$$[\![\text{sets, predicates}]\!] \; = \; \text{pointfree binary relations}$$

Calculus of binary relations $B \xleftarrow{R} A$:

- 1860 - introduced by De Morgan, embryonic
- 1870 - Peirce fi nds interesting equational laws
- 1941 - Tarski's school, cf. A Formalization of Set Theory without Variables
- 1980's - coreflexive models of sets (Freyd and Scedrov, Eindhoven MPC group and others)

Binary Relations

- Arrow $B \xleftarrow{R} A$ denotes a binary relation to B (target) from A (source).
- bRa means that pair (b, a) is in R.
- "R at most S" **ordering**:
  $R \subseteq S \;\equiv\; \langle \forall\, a, b \;::\; bRa \Rightarrow bSa \rangle$
- **Converse** of R — R° such that $a(R^\circ)b$ iff bRa.
- **Composition** — $b(R \cdot S)c$ wherever

$$\langle \exists\, a \;::\; bRa \;\wedge\; aSc \rangle$$

- **Identity**: id such that $R \cdot id = id \cdot R = R$

Sets and predicates

The meaning of a predicate ϕ is the coreflexive relation $[\![\phi]\!] \subseteq id$ such that $b[\![\phi]\!]a \equiv (b = a) \ \wedge \ (\phi \ a)$.

Example:

$$[\![n! \leq 1]\!] \quad =$$

The meaning of a set $S \subseteq A$ is the meaning of its characteristic predicate $[\![\lambda a.a \in S]\!]$, that is,

$$b[\![S]\!]a \quad \equiv \quad (b = a) \ \wedge \ a \in S$$

Uppercase Φ will abbreviate ϕ. Of course, $\Phi^\circ = \Phi$.

Useful “al-djabr” rules

Most of them are Galois connections, eg.:

$$f \cdot R \subseteq S \;\equiv\; R \subseteq f^\circ \cdot S$$

$$R \cdot f^\circ \subseteq S \;\equiv\; R \subseteq S \cdot f$$

$$T \cdot R \subseteq S \;\equiv\; R \subseteq T \setminus S$$

where $T \setminus S$ pointfree-transforms another kind of universally quantified implication:

$$b(T \setminus S)a \;\equiv\; \langle \forall\, x \;::\; x\, T\, b \Rightarrow x\, S\, a \rangle$$

Illustration

Remainder of talk overviews recent work on the pointfree-transform applied to two different problem domains:

- **Database** theory — functional and multi-valued dependences
- Operation **refinement** — Groves factorization of the satisfaction relation (joint work with Ph.D. student C. Rodrigues)

Need to develop the extended composition and inclusion rules which follow.

“Guarded” composition

Given

- binary relations $B \xleftarrow{R} A$ and $A \xleftarrow{S} C$
- predicate $2 \xleftarrow{\phi} A$ (ie. coreflexive Φ)
- $b \in B$ and $c \in C$

Then

$$\langle \exists\, a \;:\; \phi\, a : \; b\, R\, a \;\wedge\; a\, S c \rangle$$

pointfree-transforms to

$$b(R \cdot \Phi \cdot S)c$$

“Guarded ‘at most”’

Given

- binary relations $B \xleftarrow{R,S} A$
- predicates $2 \xleftarrow{\psi} A$ and $2 \xleftarrow{\phi} B$ (ie., coreflexives Ψ and Φ, respectively)

Then

$$\langle \forall\, b, a \;:\; (\phi\; b) \;\wedge\; (\psi\; a) : \; b\; R\; a \Rightarrow b\; S\; a\rangle$$

pointfree-transforms to

$$\Phi \cdot R \cdot \Psi^{\circ} \subseteq S$$

Example 1: FDs in RDB theory

Given relational table

$$T = \begin{array}{|c|c|c|c|c|} \hline \ldots & x & \ldots & y & \ldots \\ \hline \ldots & a & \ldots & b & \ldots \\ \ldots & b & \ldots & b & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \hline \end{array}$$

this is said to satisfy functional dependency $x \rightarrow y$ iff all pairs of tuples $t, t' \in T$ which "agree" on x also "agree" on y, that is,

$$\langle \forall\ t, t' \ :\ t, t' \in T : \quad t[x] = t'[x] \quad \Rightarrow \quad t[y] = t'[y] \ \rangle$$

Standard FD theory

Inference rules for FD reasoning based on

- Armstrong axioms for computing the closure of a set of FDs

However,

- formula

$$\langle \forall\, t, t' \;:\; t, t' \in T : \quad t[x] = t'[x] \;\; \Rightarrow \;\; t[y] = t'[y] \;\rangle$$

— with its logical implication inside a “two-dimensional” universal quantification — is not particularly agile.

We want to write less and. . . “let the symbols work”!

The role of functions

From **Database Systems: The Complete Book** by Garcia-Molina, Ullman and Widom (2002), p. 87:

> **What Is "Functional" About Functional Dependencies?**
>
> $A_1A_2\cdots A_n \rightarrow B$ *is called a "functional dependency" because in principle there is a function that takes a list of values [...] and produces a unique value (or no value at all) for* B *[...] However, this function is not the usual sort of function that we meet in mathematics, because there is no way to compute it from first principles. [...] Rather, the function is only computed by lookup in the relation [...]*

However,

- No advantage is taken of the rich calculus of functions

In fact, functions are everywhere in FD theory:

- as attributes and as the FDs themselves

Functions in one slide

A function f is a binary relation such that

Pointwise	Pointfree	
"Left" Uniqueness		
$b\ f\ a \wedge b'\ f\ a \;\Rightarrow\; b = b'$	$\mathit{img}\ f \;\subseteq\; id$	(f is simple)
Leibniz principle		
$a = a' \;\Rightarrow\; f\ a = f\ a'$	$id \;\subseteq\; \mathit{ker}\ f$	(f is entire)

equivalent to GCs

$$f \cdot R \subseteq S \;\equiv\; R \subseteq f^{\circ} \cdot S$$

$$R \cdot f^{\circ} \subseteq S \;\equiv\; R \subseteq S \cdot f$$

(NB: $\mathit{ker}\ f = \mathit{img}\ f^{\circ} = f^{\circ} \cdot f$ measures f's injectivity).

Pointfree-transform

Since **attribute** sets are (projection) **functions**,

- transform $(x\ t) = (x\ t')$ into $t(\textit{ker}\ x)t'$ etc
- thanks to the "guarded 'at most´ rule", for $\Phi = \Psi = [\![T]\!]$, $R = \textit{ker}\ x, S = \textit{ker}\ y$ transform

$$\langle \forall\ t, t' \ :\ t, t' \in T : \quad (x\ t) = (x\ t') \quad \Rightarrow \quad (y\ t) = (y\ t')\ \rangle$$

into

$$[\![T]\!] \cdot (\textit{ker}\ x) \cdot [\![T]\!] \subseteq \textit{ker}\ y$$

and then to…

Pointfree, generic FDs

. . . and then to

$$y \leq x \cdot [\![T]\!]$$

where $\leq$ is the "injectivity" preorder:

$$R \leq S \quad \equiv \quad \mathit{ker}\ S \subseteq \mathit{ker}\ R$$

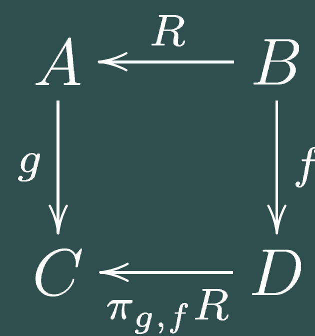
Going generic:

$$f \xrightarrow{R} g \quad \equiv \quad g \ \leq \ f \cdot R^{\circ}$$

Reasoning

Pointfree “al-jabr” rules show the above to be equivalent to

$f \overset{R}{\to} g \quad \equiv \quad$ projection $\pi_{g,f} R$ is simple,
where $\pi_{g,f} R = g \cdot R \cdot f^{\circ}$.



What is this useful for?

- **Armstrong** axioms for free
- Good cornerstone for RDB theory to follow, cf. eg. the more general **multi-valued** dependences

MVD standard definition

n-ary relation T is said to satisfy the multi-valued dependency (MVD) $x \twoheadrightarrow y$ iff, for any two tuples $t, t' \in T$ which "agree" on x there exists a tuple $t'' \in T$ which "agrees" with t on xy and "agrees" with t' on $S - xy$, that is,

$$\langle \forall\, t, t' \;:\; t, t' \in T : \quad t[x] = t'[x] \quad \rangle$$

$$\Downarrow$$

$$\langle \exists\, t'' \;:\; t'' \in T : \quad t[xy] = t''[xy] \wedge t''[S - xy] = t'[S - xy] \quad \rangle$$

	x	y	$S - xy$
t	α	β	γ
t''	α	β	γ'
t'	α	β'	γ'

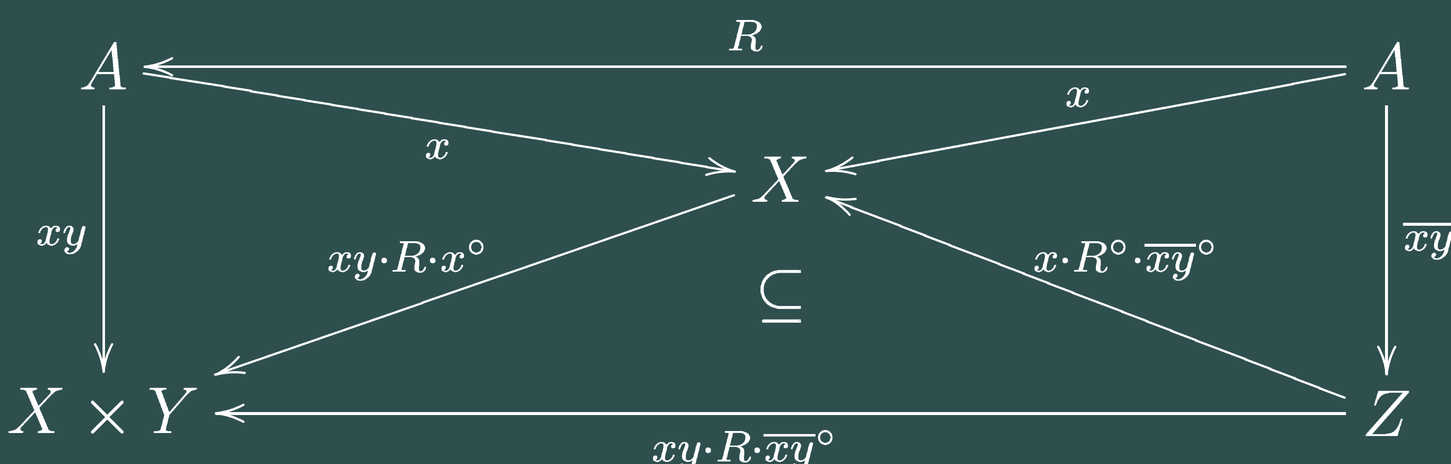
MVDs pointfree-transformed

Thanks to the “guarded rules” we obtain, in sequence:

$$x \overset{R}{\twoheadrightarrow} y \quad \equiv \quad \textit{ker}\ (x \cdot R^\circ) \subseteq (\textit{ker}\ xy) \cdot R \cdot \textit{ker}\ \overline{xy}$$

$$\equiv \quad (xy \cdot R \cdot x^\circ) \cdot (x \cdot R^\circ \cdot \overline{xy}^\circ) \quad \subseteq \quad xy \cdot R \cdot \overline{xy}^\circ$$

$$\equiv \quad (\pi_{xy,x} R) \cdot (\pi_{x,\overline{xy}} R^\circ) \quad \subseteq \quad \pi_{xy,\overline{xy}} R$$



Lossless decomposition

We are pretty close to one of the main results in RDB theory, the theorem of **lossless decomposition** of MVDs: $x \overset{T}{\twoheadrightarrow} y$ holds iff T decomposes losslessly into two relations with schemata xy and $x\overline{yx}$, respectively:

$$x \overset{T}{\twoheadrightarrow} y \qquad \equiv \qquad (\pi_{y,x}T) \bowtie (\pi_{\overline{yx},x}T) = \pi_{y\overline{yx},x}T$$

Maier [4] proves this in "implication-first" logic style, in two parts — if + only if — involving existential and universal quantifications over no less than six tuple variables $t, t_1, t_2, t'_1, t'_2, t_3$:

Lossless decomposition (Maier)

Theorem 7.1 Let r be a relation on scheme R, and let X, Y, and Z be subsets of R such that $Z = R - (X\,Y)$. Relation r satisfies the MVD $X \twoheadrightarrow Y$ if and only if r decomposes losslessly onto the relation schemes $R_1 = X\,Y$ and $R_2 = X\,Z$.

Proof: Suppose the MVD holds. Let $r_1 = \pi_{R_1}(r)$ and $r_2 = \pi_{R_2}(r)$. Let t be a tuple in $r_1 \bowtie r_2$. There must be a tuple $t_1 \in r_1$ and a tuple $t_2 \in r_2$ such that $t(X) = t_1(X) = t_2(X)$, $t(Y) = t_1(Y)$, and $t(Z) = t_2(Z)$. Since r_1 and r_2 are projections of r, there must be tuples t_1' and t_2' in r with $t_1(X\,Y) = t_1'(X\,Y)$ and $t_2(X\,Z) = t_2'(X\,Z)$. The MVD $X \twoheadrightarrow Y$ implies that t must be in r, since r must contain a tuple t_3 with $t_3(X) = t_1'(X)$, $t_3(Y) = t_1'(Y)$, and $t_3(Z) = t_2'(Z)$, which is a description of t.

Suppose now that r decomposes losslessly onto R_1 and R_2. Let t_1 and t_2 be tuples in r such that $t_1(X) = t_2(X)$. Let r_1 and r_2 be defined as before. Relation r_1 contains a tuple $t_1' = t_1(X\,Y)$ and relation r_2 contains a tuple $t_2' = t_2(X\,Z)$. Since $r = r_1 \bowtie r_2$, r contains a tuple t such that $t(X\,Y) = t_1(X\,Y)$ and $t(X\,Z) = t_2(X\,Z)$. Tuple t is the result of joining t_1' and t_2'. Hence t_1 and t_2 cannot be used in a counterexample to $X \twoheadrightarrow Y$, hence r satisfies $X \twoheadrightarrow Y$.

Our (pointfree) proof

A sequence of equivalences (for details see my draft paper *First Steps in Pointfree Functional Dependency Theory*) :

$$(\pi_{y,x} T) \bowtie (\pi_{\overline{yx},x} T) = \pi_{y\overline{yx},x} T$$

$\equiv$ { unfold defi nitions }

$$\langle y \cdot T \cdot x^\circ, \overline{yx} \cdot T \cdot x^\circ \rangle \quad = \quad y\overline{yx} \cdot T \cdot x^\circ$$

$\equiv$ { since $\langle R, S \rangle \cdot T \subseteq \langle R \cdot T, S \cdot T \rangle$ holds by monotonicity }

$$\langle y \cdot T \cdot x^\circ, \overline{yx} \cdot T \cdot x^\circ \rangle \quad \subseteq \quad y\overline{yx} \cdot T \cdot x^\circ$$

$\equiv$ { "split twist" rule ; converses }

$$\langle y \cdot T \cdot x^\circ, id \rangle \cdot (x \cdot T^\circ \cdot \overline{yx}^\circ) \quad \subseteq \quad \langle y, x \cdot T^\circ \rangle \cdot \overline{yx}^\circ$$

Pointfree proof

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

$$\langle y \cdot T \cdot x^\circ, id\rangle \cdot x \cdot x^\circ \cdot x \cdot T^\circ \cdot \overline{yx}^\circ \quad \subseteq \quad \langle y, x \cdot T^\circ\rangle \cdot \overline{yx}^\circ$$

$\equiv$ { $\langle R, S\rangle \cdot \Phi = \langle R, S \cdot \Phi\rangle$ for $\Phi := x \cdot x^\circ$ and $\Phi := T^\circ$ }

$$\langle y \cdot T \cdot x^\circ, x \cdot x^\circ\rangle \cdot x \cdot T^\circ \cdot \overline{yx}^\circ \quad \subseteq \quad \langle y, x\rangle \cdot T^\circ \cdot \overline{yx}^\circ$$

$\equiv$ { $\langle R, f\rangle \cdot f^\circ \quad = \quad \langle R \cdot f^\circ, f \cdot f^\circ\rangle$; above rule for $\Phi = T$ }

$$(\langle y, x\rangle \cdot T \cdot x^\circ) \cdot (x \cdot T^\circ \cdot \overline{yx}^\circ) \quad \subseteq \quad \langle y, x\rangle \cdot T \cdot \overline{yx}^\circ$$

$\equiv$ { definition }

$$x \overset{T}{\twoheadrightarrow} y$$

Example 2: ASM refinement

ASM (=abstract state machines) refi nement ordering:

Machine $\mathcal{P}A \xleftarrow{R} A$ implements machine $\mathcal{P}A \xleftarrow{S} A$ —written $S \vdash R$ iff

$$\langle \forall\, a \;:\; (S\; a) \supset \emptyset : \; \emptyset \subset (R\, a) \subseteq (S\; a) \rangle$$

where $S\; a$ means the set of states reachable (in machine S) from state a.

Going pointfree

ASM machines above are power-transposes of state-transition binary relations. Why not work with these directly? We obtain alternative pointwise definition

$$S \vdash R \;\equiv\; \langle \forall\, a \,:\, a \in \textit{dom}\, S : \; a \in \textit{dom}\, R \;\wedge\; (\langle \forall\, b \,:\, b\, R\, a : \; b\, S$$

the pointfree-transform of which is

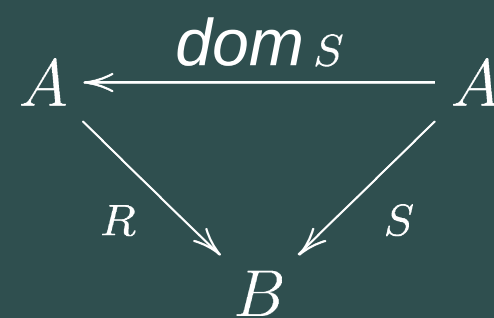
$$S \vdash R \;\equiv\; \textit{dom}\, S \subseteq \textit{dom}\, R \cap (R \setminus S)$$

that is

$$S \vdash R \;\equiv\; (\textit{dom}\, S \subseteq \textit{dom}\, R) \;\wedge\; R \cdot \textit{dom}\, S \subseteq S$$

Generalization

Easy to see that the source and target types of both S, R don't need to be the same:



So, pointfree-transformed $S \vdash R$ covers other refinement situations such that eg. a **VDM** implicit specification S refined by some function f:

$$S \vdash f \quad \equiv \quad \textit{dom}\, S \subseteq f^{\circ} \cdot S$$

In fact, from this — back to points — we obtain, in classical "VDM-speak"

$$\forall a.\text{pre-}S(a) \Rightarrow \text{post-}S(f\ a, a)$$

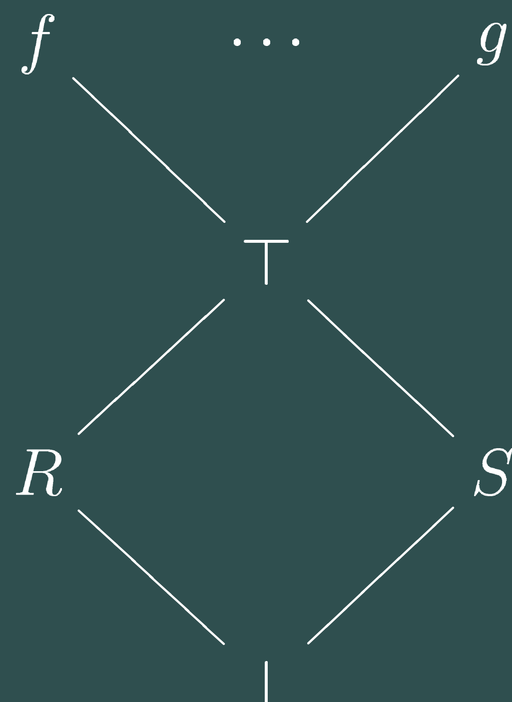
$\vdash$ = "meet of opposites"

Although consensual, the $\vdash$ ordering hosts a kind of *conflict* in its definition:

- *dom* $S \subseteq$ *dom* R (=R more defined than S) suggests R "larger" than S
- $R \cdot$ *dom* $S \subseteq S$ (=R more deterministic than S) suggests R "smaller" than S

Can $\vdash$ be "factored" into such two (kind of "anti-symmetric") sub-orderings?

Funny shaped semi-lattice



$$
\begin{aligned}
(R \sqcap S)a \;&=\; if\ (R\ a) = \emptyset \vee (S\ a) = \emptyset\ then\ \emptyset \\
&\quad else\ (R\ a) \cup (S\ a)
\end{aligned}
$$

Guessing $\vdash_{pre}$ and $\vdash_{post}$

Refine but do not increase definition:

$$S \vdash_{post} R \;\equiv\; S \vdash R \;\wedge\; \textit{dom}\, R \subseteq \textit{dom}\, S$$

Refine but do not increase determinism:

$$S \vdash_{pre} R \equiv S \vdash R \;\wedge\; S \subseteq R \cdot \textit{dom}\, S$$

Easy calculations simplify these to:

$$\vdash_{post} \;=\; \subseteq^{\circ} \cap \textit{ker dom}$$

$$S \vdash_{pre} R \;\equiv\; R \cdot \textit{dom}\, S = S$$

Relax / constrain rules

The following equivalences stem from the pointfree definitions:

- Refine S by "relaxing" it:

$$S \vdash_{pre} S \cup R \;\equiv\; R \cdot \mathit{dom}\, S \subseteq S$$

- Refine S by "constraining" it:

$$S \vdash_{post} S \cap R \;\equiv\; \mathit{dom}\, S = \mathit{dom}\,(R \cap S)$$

$\vdash \subseteq \vdash_{pre} \cdot \vdash_{post}$ calculation

$$
\begin{array}{ll}
& S \vdash R \\
\equiv & \quad \{ \text{ definition } \} \\
& R \cdot \mathit{dom}\, S \subseteq S \;\wedge\; \mathit{dom}\, S \subseteq \mathit{dom}\, R \\
\equiv & \quad \{ \;\cup\; \} \\
& R \cdot \mathit{dom}\, S \subseteq S \;\wedge\; (\mathit{dom}\, S) \cup (\mathit{dom}\, R) = \mathit{dom}\, R \\
\equiv & \quad \{ \text{ relax rule ; } \mathit{dom} \text{ is a lower-adjoint } \} \\
& (S \vdash_{pre} S \cup R) \;\wedge\; \mathit{dom}\, (S \cup R) = \mathit{dom}\, R \\
\equiv & \quad \{ \text{ constrain rule, since } R = R \cap (S \cup R) \; \} \\
& ((S \vdash_{pre} S \cup R) \;\wedge\; (S \cup R) \vdash_{post} R \cap (S \cup R) \\
\equiv & \quad \{ \text{ relax rule and } R = R \cap (S \cup R) \; \} \\
& (S \vdash_{pre} S \cup R) \;\wedge\; (S \cup R) \vdash_{post} R \\
\Rightarrow & \quad \{ \text{ composition } \} \\
& S (\vdash_{pre} \cdot \vdash_{post}) R
\end{array}
$$

Groves factorization

Very straightforward. A similar calculation leads to

$$\vdash \quad \subseteq \quad \vdash_{post} \cdot \vdash_{pre}$$

and since

$$\vdash_{post} \cdot \vdash_{pre} \quad \subseteq \quad \vdash$$
$$\vdash_{pre} \cdot \vdash_{post} \quad \subseteq \quad \vdash$$

hold by monotonicity, we've altogether calculated Groves factorization

$$\vdash_{pre} \cdot \vdash_{post} \quad = \quad \vdash \quad = \quad \vdash_{post} \cdot \vdash_{pre}$$

Calculating $R \sqcap T$

Instead of "inventing"

$$
\begin{array}{rcl}
(R \sqcap T)a & = & if\ (R\ a) = \emptyset \vee (T\ a) = \emptyset\ then\ \emptyset \\
& & else\ (R\ a) \cup (T\ a)
\end{array}
$$

and proving that it satisfies (universal property) equation

$$S \vdash R \sqcap T \quad \equiv \quad S \vdash R\ \wedge\ S \vdash T$$

at point-level, we calculate its **unique** solution

$$R \sqcap T \quad = \quad (R \cup T) \cdot \mathit{dom}\, R \cdot \mathit{dom}\, T$$

at pointfree-transform level:

Calculating $R \sqcap T$

$S \vdash R \sqcap T$

$\equiv$ { equation to be solved }

$S \vdash R \ \wedge \ S \vdash T$

$\equiv$ { definition of $\vdash$ twice; composition of coreflexives }

dom $S \subseteq (R \setminus S \cap T \setminus S) \cap$ *dom* $R \cdot$ *dom* T

$\equiv$ { property of relational division }

dom $S \subseteq (R \cup T) \setminus S \cap$ *dom* $R \cdot$ *dom* T

$\equiv$ { $(R \cdot \Phi \setminus S) \cap \Phi \quad = \quad (R \setminus S) \cap \Phi$ for $R, \Phi :=$ $R \cup T$, *dom* $R \cdot$ *dom* T }

dom $S \subseteq ((R \cup T) \cdot$ *dom* $R \cdot$ *dom* $T) \setminus S \cap$ (*dom* $R \cdot$ *dom* T)

$\equiv$ { *dom* $((R \cup T) \cdot$ *dom* $R \cdot$ *dom* $T) =$ *dom* $R \cdot$ *dom* T ; definition }

$S \vdash ((R \cup T) \cdot$ *dom* $R \cdot$ *dom* $T)$

:: { indirect equality}

$R \sqcap T = (R \cup T) \cdot$ *dom* $R \cdot$ *dom* T

Related work

- Boudriga et al [2] formulate $S \vdash R$ relationally but reason at pointwise level
- Lindsay Groves [3] postulates and then proves the above decomposition in the context of the Z schema calculus, requiring the extra notion of *compatible* relations, which complicates proofs unnecessarily.
- Our goal is to apply this factorization to the refinement of "components as coalgebras" — eg. (monadic) machines (=objects) of type $B \times (M\ A)^I \longleftarrow A$ — cf. Barbosa and Meng research [5] .

Summary

- Invest in **perennial** reasoning strategies
- Shift from “implication first” maths to “let the symbols work” maths
- Rôle of **transforms**, **abstract** notation and abstract patterns (easier to spot al-jabr rules)
- Stimulate **elegance** in mathematics (it is effective!)
- Learn with the other engineering disciplines
- Recommended reading: **Backhouse’s** draft text book [1].

References

[1] R.C. Backhouse. *Mathematics of Program Construction*. Univ. of Nottingham, 2004. Draft of book in preparation. 608 pages.

[2] Noureddine Boudriga, Fathi Elloumi, and Ali Mili. On the lattice of specifications: Applications to a specification methodology. *Formal Asp. Comput.*, 4(6):544–571, 1992.

[3] Lindsay Groves. Refinement and the Z schema calculus. *ENTCS*, 70(3), 2002. The BCS FACS Refinement Workshop, Denmark July 2002.

[4] D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1983. ISBN 0-914894-42-0.

[5] Sun Meng and L.S. Barbosa. On refinement of generic state-based software components. In C. Rettray, S. Maharaj, and C. Shankland, editors, *10th Int. Conf. Algebraic Methods and Software Technology (AMAST)*, pages 506–520, Stirling, August 2004. Springer Lect. Notes Comp. Sci. (3116). Best Student Co-authored Paper Award.