Why point-freeness matters

J.N. Oliveira

Departamento de Informática, Escola de Engenharia
Universidade do Minho

CIC'06 — U. Minho, Braga, Oct. 11-13

A “rendez vous” workshop

Three institutions

- CWI
- UM
- UNL

Three towns

- Amsterdam
- Braga
- Lisbon

Three “cultures”?

Workshop “motto”

Share R&D experiences and learn more about each other

On the UM "flavour"

Who and how

- TFM — Teory and Formal Methods Group
- Emphasis on "correct by construction"
- So-called *pointfree (PF) flavour* ...

When and why

- John Backus pointed the way (1978) — FP and parallelism
- JNO's PhD thesis (1984) — FP for dataflow reasoning
- JMV's f-NDP notation (1987) — nondeterministic FP for software design
- Bird-Meertens-Backhouse approach (now) — do it by calculation in the PF-style

However

- Pointfree? functions? relations? monads? coalgebras?

A notation conflict

Purpose of formal modelling

Identify *properties* of real-world situations which, once expressed in maths, become abstract *models* which can be queried and reasoned about.

This often raises a kind of

Notation conflict

between

- *descriptiveness* — ie., adequacy to describe domain-specific objects and properties, and
- *compactness* — as required by algebraic reasoning and solution calculation.

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 $3BA^2 - DA + A^3 = Z$

Back to the school desk

(where it all started for any of us...)

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?

Back to the school desk

(where it all started for any of us...)

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

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

Back to the school desk

(where it all started for any of us...)

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

$$\begin{array}{cl} & 3x + 9 = 48 \\ \equiv & \quad \{ \text{ "al-djabr" rule } \} \\ & 3x = 48 - 9 \\ \equiv & \quad \{ \text{ "al-hatt" rule } \} \\ & x = 16 - 3 \end{array}$$

Back to the school desk

The solution

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

Comments

- Simple problem
- Simple notation
- Questions: *"al-djabr" rule ? "al-hatt" rule ?*

Back to the school desk

The solution

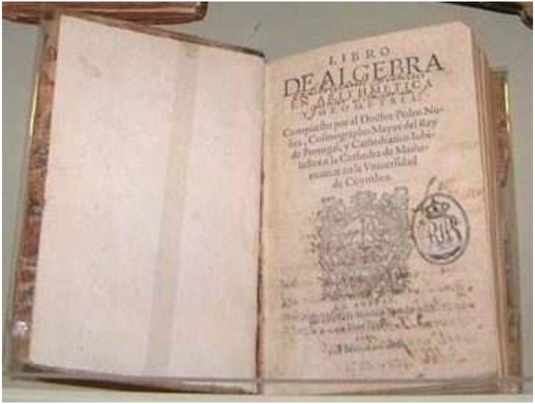
$$\begin{aligned} x &= 13 \\ x + 3 &= 16 \\ x + 6 &= 19 \end{aligned}$$

Comments

- Simple problem
- Simple notation
- Questions: *"al-djabr" rule ? "al-hatt" rule ?*

Have a look at Pedro Nunes (1502-1578) *Libro de Algebra en Arithmetica y Geometria* (published in The Netherlands in 1567)

Libro de Algebra en Arithmetica y Geometria (1567)



(...) ho inuẽtor desta arte foy hum Mathematico Mouro, cujo nome era Gebre, & ha em alguãs Liuarias hum pequeno tractado Arauigo, que contem os capitulos de q̃ usamos
(fol. a ij r)

Reference to *On the calculus of al-gabr and al-muqâbala* [1] by Abû Abd Allâh Muhamad B. Mûsâ Al-Huwârizmî, a famous 9c Persian mathematician.

[1]Original title: *Kitâb al-muhtasar fi hisab al-gabr wa-almuqâbala*.

Calculus of al-gabr and al-muqâbala

al-djabr

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

Calculus of al-gabr and al-muqâbala

al-djabr

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

al-hatt

$$x * z \leq y \quad \equiv \quad x \leq y * z^{-1} \qquad (z > 0)$$

Calculus of al-gabr and al-muqâbala

al-djabr

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

al-hatt

$$x * z \leq y \quad \equiv \quad x \leq y * z^{-1} \qquad (z > 0)$$

al-muqâbala

Ex:

$$4x^2 + 3 = 2x^2 + 2x + 6 \quad \equiv \quad 2x^2 = 2x + 3$$

Verdict

Pedro Nunes *libro de algebra*, 1567, fol 270r.

(...) De manera, que
quien sabe por Algebra,
sabe <u>scientificamente</u>.

(in this way, who knows by Algebra knows scientifically)

Thus — already in the 16c —

$e = m + c$

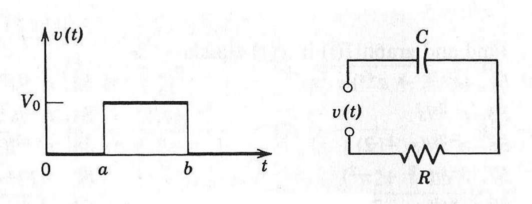
engineering $=$ <u>*model*</u> *first, then* <u>*calculate*</u> ...

Not enough

More demanding problems, eg. electrical circuits:

The problem

Predict $i(t)$ for RC-circuit

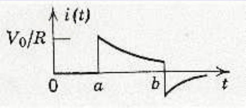


The model

$$
\begin{aligned}
v(t) &= Ri(t) + \frac{1}{C}\int_0^t i(\tau)d\tau \\
v(t) &= V_0(u(t-a) - u(t-b)) \qquad (b > a)
\end{aligned}
$$

High-school example

The solution



Calculation?

Physicists and engineers overcome difficulties in calculating integral/differential equations by changing the "mathematical space", for instance by moving (temporarily) from the time-space to the s-space in the *Laplace transformation*.

Laplace transform

$f(t)$ is transformed into $(\mathcal{L}\ f)s = \int_0^\infty e^{-st} f(t)dt$

High-school example

Laplace-transformed RC-circuit model

$$RI(s) + \frac{I(s)}{sC} = \frac{V_0}{s}(e^{-as} - e^{-bs})$$

Algebraic solution for $I(s)$

$$I(s) = \frac{\frac{V_0}{R}}{s + \frac{1}{RC}}(e^{-as} - e^{-bs})$$

Back to the t-space

$$i(t) = \begin{cases} 0 & if \quad t < a \\ (\frac{V_0 e^{-\frac{a}{RC}}}{R})e^{-\frac{t}{RC}} & if \quad a < t < b \\ (\frac{V_0 e^{-\frac{a}{RC}}}{R} - \frac{V_0 e^{-\frac{b}{RC}}}{R})e^{-\frac{t}{RC}} & if \quad t > b \end{cases}$$

(after some algebraic manipulation)

Quoting Kreyszig's book, p.242

"(...) The Laplace transformation is a method for solving differential equations (...) [which] consists of three main steps:

1st step. *The given "hard" problem is transformed into a "simple" equation (subsidiary equation).*

2nd step. *The subsidiary equation is solved by* **purely algebraic** *manipulations.*

3rd step. *The solution of the subsidiary equation is transformed back to obtain the solution of the given problem.*

In this way the Laplace transformation reduces the problem of solving a differential equation to an **algebraic problem***".*

Question

Notations and calculi used to describe software artifacts include

- Naive set theory
- Predicate calculus
- Temporal/modal logic calculi
- Lambda calculus

Is there a "Laplace transform" applicable to these?

Perhaps there is, cf. syntactic analogy

$$\langle \int x \ : \ 0 < x < 10 : \ x^2 - x \rangle$$

$$\langle \forall\, x \ : \ 0 < x < 10 : \ x^2 \geq x \rangle$$

Laplace transform: t-space $\longleftrightarrow$ s-space

$(\mathcal{L}\ f)s = \int_0^\infty e^{-st} f(t)dt$, eg.

$f(t)$	$\mathcal{L}(f)$
1	$\frac{1}{s}$
t	$\frac{1}{s^2}$
t^n	$\frac{n!}{s^{n+1}}$
e^{at}	$\frac{1}{s-a}$
etc	$\ldots$

An "s-space equivalent" for logical quantification

The pointfree ($\mathcal{PF}$) transform

ϕ	$\mathcal{PF}\ \phi$
$\langle \exists\ a\ ::\ b\ R\ a \wedge a\ S\ c \rangle$	$b(R \cdot S)c$
$\langle \forall\ a, b\ :\ b\ R\ a :\ b\ S\ a \rangle$	$R \subseteq S$
$\langle \forall\ a\ ::\ a\ R\ a \rangle$	$id \subseteq R$
$\langle \forall\ x\ :\ x\ R\ b :\ x\ S\ a \rangle$	$b(R \setminus S)a$
$\langle \forall\ c\ :\ b\ R\ c :\ a\ S\ c \rangle$	$a(S\ /\ R)b$
$b\ R\ a \wedge c\ S\ a$	$(b, c)\langle R, S \rangle a$
$b\ R\ a \wedge d\ S\ c$	$(b, d)(R \times S)(a, c)$
$b\ R\ a \wedge b\ S\ a$	$b\ (R \cap S)\ a$
$b\ R\ a \vee b\ S\ a$	$b\ (R \cup S)\ a$
$(f\ b)\ R\ (g\ a)$	$b(f^\circ \cdot R \cdot g)a$
TRUE	$b\ \top\ a$
FALSE	$b\ \bot\ a$

What are R, S, id ?

A transform for logic and set-theory

An old idea

$$\mathcal{PF}(\text{sets, predicates}) = \text{pointfree binary relations}$$

Calculus of binary relations

- 1860 - introduced by De Morgan, embryonic
- 1870 - Peirce finds 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)

Unifying approach

Everything is a (binary) relation

Binary Relations

Arrow notation

Arrow $B \xleftarrow{R} A$ denotes a binary relation to B (target) from A (source).

Identity of composition

id such that $R \cdot id = id \cdot R = R$

Converse

Converse of R — R° such that $a(R^{\circ})b$ iff $b\ R\ a$.

Ordering

"$R \subseteq S$ — the "R is at most S" — the obvious $R \subseteq S$ **ordering**.

Binary Relations

Pointwise meaning

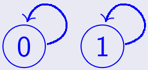
$b\ R\ a$ means that pair $\langle b, a \rangle$ is in R, eg.

$$\begin{array}{ccc} 1 & \leq & 2 \\ \texttt{John} & \mathit{IsFatherOf} & \texttt{Mary} \\ 3 & = (1+) & 2 \end{array}$$

Reflexive and coreflexive relations

- Reflexive relation: $id \subseteq R$
- Coreflexive relation: $R \subseteq id$

Sets

Are represented by coreflexives, eg. set $\{0, 1\}$ is 

Back to "quien sabe por Algebra, sabe scientificamente"

Useful "al-djabr" rules, as those (nowadays) christened as **Galois connections**

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

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

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

or **closure** rules, eg. (for Φ coreflexive),

$$\Phi \cdot R \subseteq S \quad \equiv \Phi \cdot R \subseteq \Phi \cdot S$$

Back to basics

Which areas of computing have nowadays well-established, widespread theories taught in undergraduate courses ?

- Parsers and compilers
- Relational databases
- Automata, labelled transition systems

Let us see examples of *Why point-freeness matters* in these areas.

Example: Bisimulations

Definition 1 (orig. Milner, as in the Wikipedia):

A bisimulation is a simulation between two LTS such that its converse is also a simulation, where a simulation between two LTS $(X, \Lambda, \rightarrow_X)$ and $(Y, \Lambda, \rightarrow_Y)$ is a relation $R \subseteq X \times Y$ such that, if $(p, q) \in R$, then for all α in Λ, and for all $p' \in S$, $p \xrightarrow{\alpha} p'$ implies that there is a q' such that $q \xrightarrow{\alpha} q'$ and $(p', q') \in R$:

$$
\begin{tikzcd}
p \arrow[d, "\alpha"'] & q \arrow[l, "R"'] \arrow[d, "\alpha"] \\
p' & q' \arrow[l, "R"]
\end{tikzcd}
$$

Typical example of classical, descriptive definition.

Example: Bisimulations

Definition 2 (by Aczel & Mendler):

Given two coalgebras $c : X \rightarrow F(X)$ and $d : Y \rightarrow F(Y)$ an F-bisimulation is a relation $R \subseteq X \times Y$ which can be extended to a coalgebra ρ such that projections π_1 and π_2 lift to F-comorphisms, as expressed by

$$\begin{tikzcd}
 & R \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] \arrow[d, "\rho"'] & \\
X \arrow[d, "c"'] & \mathsf{F}\, R \arrow[dl, "\mathsf{F}\,\pi_1"'] \arrow[dr, "\mathsf{F}\,\pi_2"] & Y \arrow[d, "d"] \\
\mathsf{F}\, X & & \mathsf{F}\, Y
\end{tikzcd}$$

Simpler and generic (coalgebraic)

Example: Bisimulations

Definition 3 (by Bart Jacobs):

A bisimulation for coalgebras $c : X \to F(X)$ and $d : Y \to F(Y)$ is a relation $R \subseteq X \times Y$ which is "closed under c and d":

$$(x, y) \in R \quad \Rightarrow \quad (c(x), d(y)) \in Rel(F)(R).$$

for all $x \in X$ and $y \in Y$.

Coalgebraic, even simpler

Question: are these "the same" definition?

We will check the equivalence of these definitions by PF-transformation and (kind of) PF-pattern matching

Bisimulations PF-transformed

Let us implode the outermost $\forall$ in Jacobs definition by PF-transformation:

$$
\begin{array}{ll}
 & \langle \forall\, x, y \;::\; x\, R\, y \Rightarrow (c\ x)\ Rel(F)(R)\ (d\ y) \rangle \\
\equiv & \quad \{\ \text{PF-transform rule } (f\ b)R(g\ a) \equiv b(f^{\circ} \cdot R \cdot g)a\ \} \\
 & \langle \forall\, x, y \;::\; x\, R\, y \Rightarrow x(c^{\circ} \cdot Rel(F)(R) \cdot d)y \rangle \\
\equiv & \quad \{\ \text{drop variables (PF-transform of inclusion)}\ \} \\
 & R \subseteq c^{\circ} \cdot Rel(F)(R) \cdot d \\
\equiv & \quad \{\ \text{introduce relator ; ``al-djabr'' rule}\ \} \\
 & c \cdot R \subseteq (\mathsf{F}\, R) \cdot d \\
\equiv & \quad \{\ \text{introduce Reynolds combinator}\ \} \\
 & c(\mathsf{F}\, R \leftarrow R)d
\end{array}
$$

About Reynolds arrow

"Reynolds arrow combinator" is a relation on functions

$$f(R \leftarrow S)g \equiv f \cdot S \subseteq R \cdot g$$

cf. diagram

$$\begin{tikzcd} B \arrow[d, "f"'] & A \arrow[l, "S"'] \arrow[d, "g"] \arrow[dl, phantom, "\subseteq"] \\ C & D \arrow[l, "R"] \end{tikzcd}$$

useful in expressing properties of functions — namely *monotonicity*

$$B \xleftarrow{f} A \text{ is monotonic} \equiv f(\leq_B \leftarrow \leq_A)f$$

lifting

$$f \mathrel{\dot{\leq}} g \equiv f(\leq \leftarrow id)f$$

polymorphism (free theorem):

$$\mathsf{G}\,A \xleftarrow{f} \mathsf{F}\,A \text{ is polymorphic} \equiv \langle \forall\ R\ ::\ f(\mathsf{G}\,R \leftarrow \mathsf{F}\,R)f \rangle$$

etc

Why Reynolds arrow matters?

Useful and manageable PF-properties

For example

$$
\begin{aligned}
id \leftarrow id &= id & (1)\\
(R \leftarrow S)^\circ &= R^\circ \leftarrow S^\circ & (2)\\
R \leftarrow S \subseteq V \leftarrow U &\Leftarrow R \subseteq V \wedge U \subseteq S & (3)\\
(R \leftarrow V) \cdot (S \leftarrow U) &\subseteq (R \cdot S) \leftarrow (V \cdot U) & (4)
\end{aligned}
$$

recalled from Roland's *"On a relation on functions"* (1990)

Immediately useful, eg. (1) ensures *id* as bisimulation between a given coalgebra and itself (next slide):

Why Reynolds arrow matters

Calculation

$$
\begin{array}{cl}
 & c(\mathsf{F}\ id \leftarrow id)d \\
\equiv & \qquad \{\ \text{relator F preserves the identity}\ \} \\
 & c(id \leftarrow id)d \\
\equiv & \qquad \{\ (1)\ \} \\
 & c\ (id)\ d \\
\equiv & \qquad \{\ id\ x = x\ \} \\
 & c = d
\end{array}
$$

Too simple and obvious, even *without* Reynolds arrow in the play. What about the equivalence between Jacobs and Aczel-Mendler's definition?

Why Reynolds arrow matters

Roland and Kevin Backhouse (2004) developed a number of properties of $S \leftarrow R$ to which we add the following:

$$\begin{array}{c} \text{pair } (r,\ s) \text{ is a tabulation} \\ \Downarrow \\ (r \cdot s^\circ) \leftarrow (f \cdot g^\circ) \quad = \quad (r \leftarrow f) \cdot (s \leftarrow g)^\circ \end{array} \tag{5}$$

Tabulations

A pair of functions $\begin{tikzcd} & C \arrow[dl, "r"'] \arrow[dr, "s"] & \\ A & & B \end{tikzcd}$ is a tabulation iff $r^\circ \cdot r \cap s^\circ \cdot s = id$.

Example: π_1 and π_2 form a tabulation, as we very easily check: (next slide)

Why Reynolds arrow matters

$$
\begin{array}{cl}
 & \pi_1^\circ \cdot \pi_1 \cap \pi_2^\circ \cdot \pi_2 = id \\
\equiv & \qquad \{ \text{ go pointwise, where } \cap \text{ is conjunction } \} \\
 & (b,a)(\pi_1^\circ \cdot \pi_1)(y,x) \wedge (b,a)(\pi_2^\circ \cdot \pi_2)(y,x) \equiv (b,a) = (y,x) \\
\equiv & \qquad \{ \text{ rule } (f\ \ b)R(g\ \ a) \equiv b(f^\circ \cdot R \cdot g)a \text{ twice } \} \\
 & \pi_1(b,a) = \pi_1(y,x) \wedge \pi_2(b,a) = \pi_2(y,x) \equiv (b,a) = (y,x) \\
\equiv & \qquad \{ \text{ trivia } \} \\
 & b = y \wedge a = x \quad \equiv \quad (b,a) = (y,x)
\end{array}
$$

NB: it is a standard result that every R can be factored in a tabulation $R = f \cdot g^\circ$, eg. $R = \pi_1 \cdot \pi_2^\circ$.

Jacobs $\equiv$ Aczel & Mendler

$$
\begin{array}{cl}
 & c(\mathsf{F}\,R \leftarrow R)d \\
\equiv & \qquad \{ \text{ tabulate } R = \pi_1 \cdot \pi_2^{\circ} \ \} \\
 & c(\mathsf{F}(\pi_1 \cdot \pi_2^{\circ}) \leftarrow (\pi_1 \cdot \pi_2^{\circ}))d \\
\equiv & \qquad \{ \text{ relator commutes with composition and converse } \} \\
 & c(((\mathsf{F}\,\pi_1) \cdot (\mathsf{F}\,\pi_2)^{\circ}) \leftarrow (\pi_1 \cdot \pi_2^{\circ}))d \\
\equiv & \qquad \{ \ (5) \ \} \\
 & c((\mathsf{F}\,\pi_1 \leftarrow \pi_1) \cdot ((\mathsf{F}\,\pi_2)^{\circ} \leftarrow \pi_2^{\circ}))d \\
\equiv & \qquad \{ \ (2) \ \} \\
 & c((\mathsf{F}\,\pi_1 \leftarrow \pi_1) \cdot (\mathsf{F}\,\pi_2 \leftarrow \pi_2)^{\circ})d \\
\equiv & \qquad \{ \text{ go pointwise (composition) } \} \\
 & \langle \exists\ a\ ::\ c(\mathsf{F}\,\pi_1 \leftarrow \pi_1)a \wedge d(\mathsf{F}\,\pi_2 \leftarrow \pi_2)a \rangle
\end{array}
$$

cf.

$$
\begin{tikzcd}
X \arrow[dddd, "c"'] & & Y \arrow[ll, "R"'] \arrow[dddd, "d"] \\
 & Z \arrow[ul, "\pi_1"] \arrow[ur, "\pi_2"'] \arrow[d, "a"] & \\
 & \mathsf{F}\,Z \arrow[ddl, "\mathsf{F}\,\pi_1"] \arrow[ddr, "\mathsf{F}\,\pi_2"'] & \\
 & & \\
\mathsf{F}\,X & & \mathsf{F}\,Y \arrow[ll, "\mathsf{F}\,R"]
\end{tikzcd}
$$

Why Reynolds arrow matters

Meaning of $\langle \exists\ a\ ::\ c(\mathsf{F}\,\pi_1 \leftarrow \pi_1)a \wedge d(\mathsf{F}\,\pi_2 \leftarrow \pi_2)a \rangle$:

there exists a coalgebra a whose carrier is the "graph" of bisimulation R and which is such that projections π_1 and π_2 lift to the corresponding coalgebra morphisms.

Comments:

- One-slide-long proofs are easier to grasp — for a (longer) bi-implication proof of the above see Backhouse & Hoogendijk's paper on *dialgebras* (1999)
- Elegance of the calculation lies in the synergy brought about by Reynolds arrow (to the best of our knowledge, such a synergy is new in the literature)
- Rule (5) does most of the work — its proof is an example of generic, stepwise PF-reasoning (cf. last talk this afternoon)

Invariants

Fact $c(\mathsf{F}\, id \leftarrow id)c$ above already tells us that id is a (trivial) F-invariant for coalgebra c. In general:

F-invariants

An F-invariant Φ is a *coreflexive* bisimulation between a coalgebra and itself:

$$c(\mathsf{F}\,\Phi \leftarrow \Phi)c \tag{6}$$

Invariants bring about *modalities*:

$$\begin{array}{rcl} c(\mathsf{F}\,\Phi \leftarrow \Phi)c & \equiv & c \cdot \Phi \subseteq \mathsf{F}\,\Phi \cdot c \\ & \equiv & \qquad \{ \text{ “al-djabr” rule } \} \\ & & \Phi \subseteq \underbrace{c^\circ \cdot (\mathsf{F}\,\Phi) \cdot c}_{\bigcirc_c \Phi} \end{array}$$

since we define the *“next time X holds”* modal operator as

$$\bigcirc_c X \stackrel{\mathrm{def}}{=} c^\circ \cdot (\mathsf{F}\,X) \cdot c$$

Invariants and projections

Elsewhere we have derived Galois connection

$$\pi_{g,f} R \subseteq S \quad \equiv \quad R \subseteq g^\circ \cdot S \cdot f \tag{7}$$

in order to get (for free) properties of lower adjoint $\pi_{g,f}$ in the context of multi-valued dependency reasoning (database theory).

Interesting enough, this time we reuse an instance of such a connection, ie. *"al-djabr" rule*

$$\pi_c \Phi \subseteq \Psi \quad \equiv \quad \Phi \subseteq \bigcirc_c \Psi \tag{8}$$

(within coreflexives) to obtain (again for free) properties — now — of the upper adjoint $\bigcirc_c$:

Invariants and projections

As as upper adjoint in a Galois connection,

- $\bigcirc_c$ is **monotonic** — thus simple proofs such as

$$
\begin{array}{cl}
 & \Phi \text{ is an invariant} \\
\equiv & \qquad \{ \text{ PF-definition of invariant } \} \\
 & \Phi \subseteq \bigcirc_c \Phi \\
\Rightarrow & \qquad \{ \text{ monotonicity } \} \\
 & \bigcirc_c \Phi \subseteq \bigcirc_c (\bigcirc_c \Phi) \\
\equiv & \qquad \{ \text{ PF-definition of invariant } \} \\
 & \bigcirc_c \Phi \text{ is an invariant}
\end{array}
$$

- $\bigcirc_c$ **distributes** over conjunction, that is PF-equality

$$\bigcirc_c (\Phi \cdot \Psi) \quad = \quad (\bigcirc_c \Phi) \cdot (\bigcirc_c \Psi)$$

holds, etc

What about Milner's original definition?

Milner's definition is recovered via

- the power-transpose relating binaru relations and set-valued functions,

$$f = \Lambda R \quad \equiv \quad R = \in \cdot f \tag{9}$$

  where $A \xleftarrow{\in} \mathcal{P}A$ is the membership relation.

- the powerset relator:

$$\mathcal{P}R \quad = \quad (\in \backslash (R \cdot \in)) \cap ((\in^\circ \cdot R)/(\in^\circ)) \tag{10}$$

  which unfolds to an elaborate pointwise formula:

$$Y(\mathcal{P}R)X \quad \equiv \quad \langle \forall\ a \ :\ a \in Y :\ \langle \exists\ b \ :\ b \in X :\ a\ R\ b \rangle\rangle \wedge \ldots etc$$

Follow up

- Further modal operators, for instance $\Box\Psi$ — *henceforth* Ψ — usually defined as *the largest invariant at most* Ψ:

$$\Box\Psi \quad = \quad \langle \bigcup \Phi \ :: \ \Phi \subseteq \Psi \cap \bigcirc_c \Phi \rangle$$

  which shrinks to a greatest (post)fix-point

$$\Box\Psi \quad = \quad \langle \nu \ \Phi \ :: \ \Psi \cdot \bigcirc_c \Phi \rangle$$

  where meet (of coreflexives) is replaced by composition, as this paves the way to agile reasoning

- Properties calculated by PF-fixpoint calculation
- etc

Summary

- Pointfree / pointwise dichotomy: PF is for reasoning in-the-large, PW is for the small
- As in the 9c and 16c, "al-djabr" rules are forever
- Back to basics: need for computer science theory "refactoring"
- Rôle of PF-patterns: clear-cut expression of complex logic structures once expressed in less symbols
- Rôle of PF-patterns: much easier to spot synergies among different theories
- Coalgebraic approach in a relational setting: a win-win approach while putting together coalgebras (functions) + relators (relations).
- Other exercises — refinement and database theories