PF-transform: using Galois connections to structure relational algebra

J.N. Oliveira

Dept. Informática,
Universidade do Minho
Braga, Portugal

DI/UM, 2008 (updated: Dec. 2009, Nov. 2010)

Why Galois connections?

We motivate this subject by placing some very general questions:

- Why is **programming**, or **systems design** “difficult”?
- Is there a generic skill, or competence, that one such acquire to become a “good programmer”?

What **makes** programming difficult?

- **Technology** (mess) — don’t fall in the trap: simply **abstract** from it!
- **Requirements** — again abstract from these as much as possible — write formal models or specs

Specifications:

- What is it that makes the specification of a problem hard to fulfill?

Problems = Easy + Hard

Superlatives in problem statements, eg.

- *"... the smallest such number"*
- *"... the longest such list"*
- *"... the best approximation"*

suggest two layers in specifications:

- the **easy** layer — **broad** class of solutions (eg. a *prefix* of a list)
- the **difficult** layer — requires one **particular** such solution regarded as **optimal** in some sense (eg. "shortest with maximal density").

Example

Requirements for **whole division** $x \div y$:

- Write a program which computes number z which, multiplied by y, approximates x.
- Check your program with the following test data:
  $x, y, z = 7, 2, 1$
  $x, y, z = 7, 2, 2$
- Ups! Forgot to tell that I want the **largest** such number (sorry!):
  $x, y, z = 7, 2, 3$

Deriving the algorithm... from what?

... *where is the formal specification of* $x \div y$*?*

Example

Requirements for **whole division** $x \div y$:

- Write a program which computes number z which, multiplied by y, approximates x.
- Check your program with the following test data:
  $x, y, z = 7, 2, 1$
  $x, y, z = 7, 2, 2$
- Ups! Forgot to tell that I want the **largest** such number (sorry!):
  $x, y, z = 7, 2, 3$

Deriving the algorithm... from what?

... where is the formal specification of $x \div y$*?*

Example — writing a spec

First version (literal):

$$x \div y = \langle \bigvee\ z\ ::\ z \times y \leq x \rangle \tag{1}$$

Second version (involved):

$$z = x \div y\ \equiv\ \langle \exists\ r\ :\ 0 \leq r < y :\ x = z \times y + r \rangle \tag{2}$$

Third version (clever!):

$$z \times y \leq x\ \equiv\ z \leq x \div y \qquad (y > 0) \tag{3}$$

— a Galois connection.

Why (3) is better than (1,2)

It captures the requirements:

- It is a solution: $x \div y$ multiplied by y approximates x

  $$(x \div y) \times y \leq x$$

  (let $z := x \div y$ in (3) and simplify)

- It is the best solution because it provides the **largest** such number:

  $$z \times y \leq x \;\Rightarrow\; z \leq x \div y \qquad\qquad (y > 0)$$

  (the $\Rightarrow$ part of $\equiv$).

Main advantage:

Highly calculational! See the next example.

Proving $(n \div m) \div d = n \div (d \times m)$

$$
\begin{array}{ll}
 & q \leq (n \div m) \div d \\
\equiv & \qquad \{ \ \text{“al-djabr” (3)} \ \} \\
 & q \times d \leq n \div m \\
\equiv & \qquad \{ \ \text{“al-djabr” (3)} \ \} \\
 & (q \times d) \times m \leq n \\
\equiv & \qquad \{ \ \times \text{ is associative } \} \\
 & q \times (d \times m) \leq n \\
\equiv & \qquad \{ \ \text{“al-djabr” (3)} \ \} \\
 & q \leq n \div (d \times m) \\
:: & \qquad \{ \text{ indirection } \} \\
 & (n \div m) \div d = n \div (d \times m)
\end{array}
$$

(Generic) indirect equality

Note the use of **indirect equality** rule

$$(q \leq x \equiv q \leq y) \equiv (x = y)$$

valid for $\leq$ **any** partial order.

Exercise 1: Derive from (3) the two *cancellation* laws

$$q \leq (q \times d) \div d \tag{4}$$

$$(n \div d) \times d \leq n \tag{5}$$

and *reflexion* law:

$$n \div d \geq 1 \equiv d \leq n \tag{6}$$

□

Galois connections

$n \div d$ is an example of operation involved in a **Galois** connection:

$$\underbrace{q \times d}_{f\ q} \leq n \quad \equiv \quad q \leq \underbrace{n \div d}_{g\ n}$$

In general, for **preorders** $(A, \leq)$ and $(B, \sqsubseteq)$ and

$$\begin{tikzcd}
(A, \leq) \arrow[r, bend left, "g"] & (B, \sqsubseteq) \arrow[l, bend left, "f"]
\end{tikzcd} \tag{7}$$

(f, g) are *Galois connected* iff...

Galois adjoints

$$\underbrace{f}_{\text{lower adjoint}}\ b \leq a \quad \equiv \quad b \sqsubseteq \underbrace{g}_{\text{upper adjoint}}\ a$$

that is

$$f^{\circ} \cdot \leq \quad = \quad \sqsubseteq \cdot g$$

Remarks:

- Galois (connected) adjoints enjoy a number of interesting **generic** properties
- *Very elegant* — **calculational** — way of performing *equational* reasoning (including *logical* deduction)

Basic properties

Cancellation:

$$(f \cdot g)a \leq a \quad \text{and} \quad b \sqsubseteq (g \cdot f)b$$

Distribution (in case of lattice structures):

$$\begin{array}{rcl} f(a \sqcup a') & = & (f\ a) \vee (f\ a') \\ g(b \wedge b') & = & (g\ b) \sqcap (g\ b') \end{array}$$

Conversely,

- If f distributes over $\sqcup$ then it has an upper adjoint g ($f^{\#}$)
- If g distributes over $\wedge$ then it has a lower adjoint f ($g^{\flat}$)

Other properties

If (f, g) are Galois connected,

- f (g) **uniquely** determines g (f) — thus the $\_^\flat$, $\_^\sharp$ notations
- f and g are **monotonic**
- (g, f) are also Galois connected — just **reverse** the orderings
- $f = f \cdot g \cdot f$ and $g = g \cdot f \cdot g$

etc

Summary

$(f\ b) \leq a \equiv b \sqsubseteq (g\ a)$		
Description	$f = g^\flat$	$g = f^\sharp$
Definition	$f\ b = \bigwedge\{a : b \sqsubseteq g\ a\}$	$g\ a = \bigsqcup\{b : f\ b \leq a\}$
Cancellation	$f(g\ a) \leq a$	$b \sqsubseteq g(f\ b)$
Distribution	$f(b \sqcup b') = (f\ b) \vee (f\ b')$	$g(a' \sqcap a) = (g\ a') \sqcap (g\ a)$
Monotonicity	$b \sqsubseteq b' \Rightarrow f\ b \leq f\ b'$	$a \leq a' \Rightarrow g\ a \sqsubseteq g\ a'$

In the sequel we will re-interpret the relational operators we've seen so far as Galois adjoints.

Examples

Not only

$$\underbrace{(d\times)\,q}_{f\ q} \leq n \quad\equiv\quad q \leq \underbrace{n(\div d)}_{g\ n}$$

but also the two shunting rules,

$$\underbrace{(h\cdot)X}_{f\ X} \subseteq Y \quad\equiv\quad X \subseteq \underbrace{(h^{\circ}\cdot)Y}_{g\ Y}$$

$$\underbrace{X(\cdot h^{\circ})}_{f\ X} \subseteq Y \quad\equiv\quad X \subseteq \underbrace{Y(\cdot h)}_{g\ Y}$$

as well as *converse*,

$$\underbrace{X^{\circ}}_{f\ X} \subseteq Y \quad\equiv\quad X \subseteq \underbrace{Y^{\circ}}_{g\ Y}$$

and so and so forth — see the next two slides.

Converse

$(f\ X) \subseteq Y \equiv X \subseteq (g\ Y)$			
Description	$f = g^{\flat}$	$g = f^{\sharp}$	**Obs.**
converse	$(\_)^{\circ}$	$(\_)^{\circ}$	$bR^{\circ}a \equiv aRb$

Thus:

Cancellation $(R^{\circ})^{\circ} = R$

Monotonicity $R \subseteq S \equiv R^{\circ} \subseteq S^{\circ}$

Distributions $(R \cap S)^{\circ} = R^{\circ} \cap S^{\circ}, (R \cup S)^{\circ} = R^{\circ} \cup S^{\circ}$

Example of calculation from the GC

Converse involution:

$$(R^\circ)^\circ \;=\; R \qquad (8)$$

Indirect proof of (8):

$$
\begin{array}{ll}
 & (R^\circ)^\circ \subseteq Y \\
\equiv & \quad \{ \ {}^\circ\text{-universal} \quad X^\circ \subseteq Y \;\equiv\; X \subseteq Y^\circ \quad \text{for } X := R^\circ \ \} \\
 & R^\circ \subseteq Y^\circ \\
\equiv & \quad \{ \ {}^\circ\text{-monotonicity} \ \} \\
 & R \subseteq Y \\
:: & \quad \{ \ \text{indirection} \ \} \\
 & (R^\circ)^\circ = R
\end{array}
$$

Functions

$(f\ X) \subseteq Y \equiv X \subseteq (g\ Y)$			
Description	$f = g^\flat$	$g = f^\sharp$	**Obs.**
shunting rule	$(h\cdot)$	$(h^\circ\cdot)$	NB: h is a function
"converse" shunting rule	$(\cdot h^\circ)$	$(\cdot h)$	NB: h is a function

Consequences:

Functional equality: $h \subseteq g \equiv \quad h = k \quad \equiv h \supseteq k$

Functional division: $h^\circ \cdot R = h \setminus R$

Question: what does $h \setminus R$ mean?

Relational division

$(f\ X) \subseteq Y \equiv X \subseteq (g\ Y)$			
Description	$f = g^\flat$	$g = f^\sharp$	**Obs.**
left-division	$(R\cdot)$	$(R \setminus\ )$	left-factor
right-division	$(\cdot R)$	$(\ / R)$	right-factor

that is,

$$R \cdot X \subseteq Y \equiv X \subseteq R \setminus Y \tag{9}$$
$$X \cdot R \subseteq Y \equiv X \subseteq Y / R \tag{10}$$

Immediate: $(R\cdot)$ and $(\cdot R)$ distribute over union:

$$\begin{aligned} R \cdot (S \cup T) &= (R \cdot S) \cup (R \cdot T) \\ (S \cup T) \cdot R &= (S \cdot R) \cup (T \cdot R) \end{aligned}$$

Some intuition about relational division operators follows.

Relational (left) division

Left division abstracts a (pointwise) universal quantification

$$\begin{tikzcd} A \arrow[dr, "R"'] & & C \arrow[ll, "R \backslash S"'] \arrow[dl, "S"] \\ & B \arrow[u, phantom, "\subseteq"] & \end{tikzcd}$$

$$a(R \setminus S)c \equiv \langle \forall\ b : b\ R\ a : b\ S\ c \rangle \quad (11)$$

Example:

$b\ R\ a$ = flight b carries passenger a

$b\ S\ c$ = flight b belongs to air-company c

$a\ (R \setminus S)\ c$ = passenger a is faithful to company c, that is, (s)he only flies company c.

Relational (right) division

By taking converses we arrive at $S \,/\, R = (R^\circ \setminus S^\circ)^\circ$:

$$
\begin{array}{ll}
 & X \subseteq S \,/\, R \\
\equiv & \qquad \{ \text{ Galois connection } ((\cdot R), (/R)) \text{ } \} \\
 & X \cdot R \subseteq S \\
\equiv & \qquad \{ \text{ converses } \} \\
 & R^\circ \cdot X^\circ \subseteq S^\circ \\
\equiv & \qquad \{ \text{ Galois connection } ((R\cdot), (R\setminus)) \text{ } \} \\
 & X^\circ \subseteq R^\circ \setminus S^\circ \\
\equiv & \qquad \{ \text{ converses } \} \\
 & X \subseteq (R^\circ \setminus S^\circ)^\circ \\
:: & \qquad \{ \text{ indirection } \} \\
 & S \,/\, R = (R^\circ \setminus S^\circ)^\circ
\end{array}
$$

Relational (right) division

Therefore:

$$
\begin{aligned}
& c(S \,/\, R)a \\
\equiv & \qquad \{ \text{ above } \} \\
& a(R^\circ \setminus S^\circ)c \\
\equiv & \qquad \{ \ (11) \ \} \\
& \langle \forall\ b \ : \ b\ R^\circ a : \ b\ S^\circ c \rangle \\
\equiv & \qquad \{ \ \text{converses} \ \} \\
& \langle \forall\ b \ : \ a\ R\ b : \ c\ S\ b \rangle
\end{aligned}
$$

$$
\begin{tikzcd}
 & a \arrow[dl, "S/R"'] \\
c & b \arrow[l, "S"] \arrow[u, "R"']
\end{tikzcd}
$$

Domain and range

$(f\ X) \subseteq Y \equiv X \subseteq (g\ Y)$			
Description	$f = g^\flat$	$g = f^\sharp$	**Obs.**
domain	δ	$(\top\cdot)$	lower $\subseteq$ restricted to coreflexives
range	ρ	$(\cdot\top)$	lower $\subseteq$ restricted to coreflexives

Thus

$$\delta\, R \subseteq \Phi \quad \equiv \quad R \subseteq \top \cdot \Phi \tag{12}$$

$$\rho\, R \subseteq \Phi \quad \equiv \quad R \subseteq \Phi \cdot \top \tag{13}$$

etc.

Domain and split

The following fact holds:

$$\langle R, S\rangle^\circ \cdot \langle X, Y\rangle \quad = \quad (R^\circ \cdot X) \cap (S^\circ \cdot Y)$$

Corollary:

$$\delta\, R \quad = \quad \ker \langle id, R\rangle$$

Another consequence of the fact above:

$$\ker R \subseteq \ker (S \cdot R) \quad \Leftarrow \quad S \text{ entire}$$

Corollary:

$$\ker R \quad \subseteq \quad \ker (f \cdot R)$$

Appendix I

Handling Hoare triples in relation algebra

We finally show to handle **Hoare triples** such as

$$\{p\}P\{q\} \tag{14}$$

in pointfree, relation algebra. First we spell out the meaning of (14):

$$\langle \forall\ s\ :\ p\ s :\ \langle \forall\ s'\ :\ s \xrightarrow{P} s'\ :\ q\ s' \rangle \rangle \tag{15}$$

Then (recording the meaning of program P as relation $[\![P]\!]$ on program states) we PF-transform (15) into

$$\Phi_p \subseteq [\![P]\!] \setminus (\Phi_q \cdot \top) \tag{16}$$

thanks to (11) and then to...

Relationship with Hoare Logic

$$[\![P]\!] \cdot \Phi_p \subseteq \Phi_q \cdot \top \tag{17}$$

thanks to (9). By putting (17) and the meaning of $\Phi_q \xleftarrow{f} \Phi_p$,

$$f \cdot \Phi_p \subseteq \Phi_p \cdot \top \tag{18}$$

we realize both share the same scheme,

$$R \cdot \Phi \subseteq \Psi \cdot \top \tag{19}$$

which is equivalent to

$$R \cdot \Phi \subseteq \Psi \cdot R \tag{20}$$

(tell why) and which one can condense into notation

$$\Phi \xrightarrow{R} \Psi \tag{21}$$

Relationship with Hoare Logic

All in all

- Notation (21) can be regarded as the **type assertion** that, if fed with values (or starting on states) "of type Φ" computation P yields results (changes to states) "of type Ψ" (if it terminates).
- We see that functional *predicative types* and Hoare Logic are one and the same device: a way to **type** computations, be them specified as (allways terminating, deterministic) functions or encoded into (possibly non-terminating, non-deterministic) programs.

Appendix II

"Al-djabr" calculation of algorithms

The next slides show how the well-known algorithm implementing whole division,

$$n \div d \quad = \quad \mathit{if} \quad n < d \quad \mathit{then} \quad 0 \quad \mathit{else} \quad (n-d) \div d + 1$$

can be inferred from "al-djabr" rule (3) via indirect equality, in two parts:

1. case $n \geq d$
2. case $n < d$

.

Calculation of $n \div d$ case $n \geq d$

$$
\begin{array}{ll}
 & q \leq n \div d \\
\equiv & \quad \{ \text{ rule (3) assuming } d > 0 \text{ } \} \\
 & q \times d \leq n \\
\equiv & \quad \{ \text{ cancellation } \} \\
 & q \times d - d \leq n - d \\
\equiv & \quad \{ \text{ distribution law } \} \\
 & (q-1) \times d \leq n - d \\
\equiv & \quad \{ \text{ (3) again, assuming } n \geq d \text{ } \} \\
 & q - 1 \leq (n-d) \div d \\
\equiv & \quad \{ \text{ trading } -1 \text{ to the right } \} \\
 & q \leq (n-d) \div d + 1
\end{array}
$$

Calculation of $n \div d$ case $n < d$

That is, every natural number q which is at most $n \div d$ (for $n \geq d$) is also at most $(n-d) \div d + 1$ and vice versa. We conclude that the two expressions are the same

$$n \div d = (n-d) \div d + 1 \tag{22}$$

for $n \geq d$. For $n < d$, we reason in the same style:

$$
\begin{array}{ll}
 & q \leq n \div d \\
\equiv & \quad \{ \ (3) \text{ and transitivity, since } n < d \ \} \\
 & q \times d \leq n \wedge q \times d < d \\
\equiv & \quad \{ \ \text{since } d \neq 0 \ \} \\
 & q \times d \leq n \wedge q \leq 0 \\
\equiv & \quad \{ \ q \leq 0 \text{ entails } q \times d \leq n, \text{ since } 0 \leq n \ \} \\
 & q \leq 0
\end{array}
$$

If-then-else's — eventually!

So, in case $n < d$, we have

$$q \leq n \div d \quad \equiv \quad q \leq 0$$

By indirect equality, we get, for this case

$$n \div d \quad \equiv \quad 0$$

In other words, we have calculated the **then** and **else**-parts of the algorithm:

$$n \div d \quad = \quad \mathit{if} \quad n < d \quad \mathit{then} \quad 0 \quad \mathit{else} \quad (n - d) \div d + 1$$

Appendix III

Modular law

Dedekind's rule, also known as the **modular law**:

$$R \cdot S \cap T \quad \subseteq \quad R \cdot (S \cap R^\circ \cdot T) \tag{23}$$

cf. analogy with $ab + c \quad \leq \quad a(b + a^{-1}c)$. Dually (apply converses and rename):

$$(R \cdot S) \cap T \quad \subseteq \quad (R \cap (T \cdot S^\circ)) \cdot S \tag{24}$$

Symmetrical equivalent statement:

$$(R \cdot S) \cap T \quad \subseteq \quad (R \cap (T \cdot S^\circ)) \cdot (S \cap (R^\circ \cdot T)) \tag{25}$$

$=$ "weak right-distribution of meet over composition".