PF-transform: using Galois connections to structure relational algebra

J.N. Oliveira

Dept. Informática,
Universidade do Minho
Braga, Portugal

DI/UM, 2008 (updated: 2009-10, 2012, 2014)

Motivation

We motivate this subject by placing some very general questions:

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

Surely that of **abstract modelling**. But, still,

- What is it that makes abstract modelling a challenging task?
- Are there generic conceptual **patterns** that could be used to shorten the path from **problems** to **models**?

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. "longest prefix up to a given length").

Example — back to the primary school desk

The **whole division** algorithm

$$\begin{array}{r|l} 7 & 2 \\ \hline 1 & 3 \end{array} \qquad 2 \times 3 + 1 = 7 \quad , \text{ “ie.”} \qquad 3 = 7 \div 2$$

However

$$\begin{array}{r|l} 7 & 2 \\ \hline 3 & 2 \end{array} \qquad 2 \times 2 + 3 = 7 \quad \wedge \quad 2 \neq 7 \div 2$$

$$\begin{array}{r|l} 7 & 2 \\ \hline 5 & 1 \end{array} \qquad 2 \times 1 + 5 = 7 \quad \wedge \quad 1 \neq 7 \div 2$$

That is: for some r,

$$\begin{array}{r|l} n & d \\ \hline r & q \end{array} \qquad q = n \div d \;\equiv\; d \times q + r = n$$

provided q is the largest such q (r smallest)

Example — specifying $x \div y$

First version (literal):

$$x \div y = \langle \bigvee \; z \; :: \; z \times y \leq x \rangle \tag{203}$$

Second version (involved):

$$z = x \div y \;\equiv\; \langle \exists\, r \;:\; 0 \leq r < y : \; x = z \times y + r \rangle \tag{204}$$

Third version (clever!):

$$z \times y \leq x \;\equiv\; z \leq x \div y \qquad (y > 0) \tag{205}$$

— a so-called **Galois connection**, as we shall soon see.

Why (205) is better than (203,204)

Equivalence (205),

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

captures the requirements in an elegant way:

- 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 (205) and simplify.

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

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

— the $\Rightarrow$ part of the $\equiv$ of (205).

Reasoning

Equivalence (205)

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

is not only simple to write but effective to reason about.

Let us see an example: we want to prove the following equality

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

What about

- using (203)? too many suprema!
- using (204)? too many existential quantifiers!
- using (205)? easy — see the next slide.

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

$$
\begin{array}{cl}
 & q \leq (n \div m) \div d \\
\equiv & \quad \{ \ (205) \ \} \\
 & q \times d \leq n \div m \\
\equiv & \quad \{ \ (205) \ \} \\
 & (q \times d) \times m \leq n \\
\equiv & \quad \{ \ \times \text{ is associative } \} \\
 & q \times (d \times m) \leq n \\
\equiv & \quad \{ \ (205) \ \} \\
 & q \leq n \div (d \times m) \\
:: & \quad \{ \text{ indirection } (206) \ \} \\
 & (n \div m) \div d = n \div (d \times m)
\end{array}
$$

(Generic) indirect equality

Note the use of the (generic) **indirect equality** rule

$$\langle \forall\ q\ ::\ q \leq x \equiv q \leq y \rangle \equiv (x = y) \tag{206}$$

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

Exercise 95: Derive from (205) the two *cancellation* laws

$$\begin{array}{rcl} q & \leq & (q \times d) \div d \\ (n \div d) \times d & \leq & n \end{array}$$

and *reflexion* law:

$$n \div d \geq 1 \quad \equiv \quad d \leq n \tag{207}$$

□

Galois connections

Equivalence (205) is an example of a **Galois** connection:

$$\underbrace{z \times y}_{f\ z} \leq x \quad \equiv \quad z \leq \underbrace{x \div y}_{g\ x}$$

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

$$(A, \leq) \underset{f}{\overset{g}{\rightleftarrows}} (B, \sqsubseteq) \qquad (208)$$

(f, g) are said to be **Galois connected** iff, for all $a \in A$ and $b \in B$...

Galois adjoints

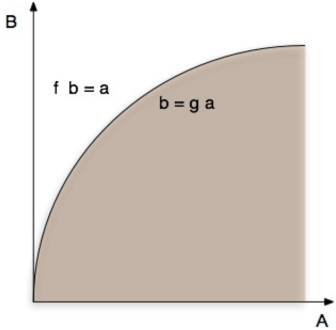
$$\underbrace{f}_{\text{lower adjoint}} b \leq a \equiv b \sqsubseteq \underbrace{g}_{\text{upper adjoint}} a \quad (209)$$

that is

$$f^{\circ} \cdot \leq \; = \; \sqsubseteq \cdot g \quad (210)$$

Graphical interpretation of (210):

- $\sqsubseteq \cdot g$ is the "area" below function g wrt. $\sqsubseteq$
- $f^{\circ} \cdot \leq$ is the "area" above function f wrt. $\leq$
- f and g are such that these areas are the same.



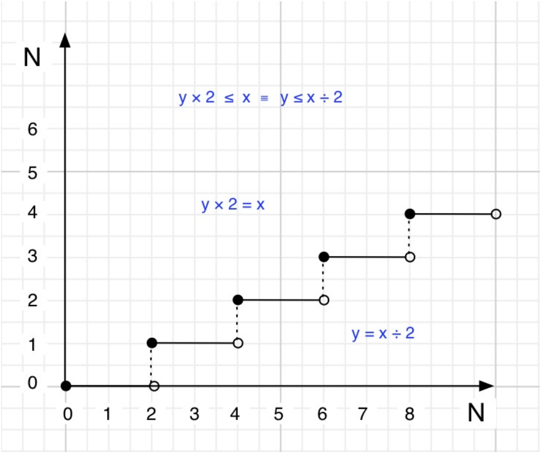
Still whole division

$f = (\times 2)$ is the lower adjoint of $g = (\div 2)$.

The area below $g = (\div 2)$ is the same as the area above $f = (\times 2)$.

$f = (\times 2)$ is not surjective.

$g = (\div 2)$ is not injective.



Adjoints are “nearly” inverses

Easy to observe:

- $g(f\ y) = (y \times 2) \div 2 = y$ — f is indeed a right inverse for g
- $f(g\ 5) = (5 \div 2) \times 2 = 2 \times 2 = 4 \leq 5$ — g is not a right inverse for f, but it provides an **approximation**.

In spite of this asymmetry, the connection enables us to reason about

$$g = (\div y)$$

— the “**hard**” operation — in terms of

$$f = (\times y)$$

— the “**easy**” operation. This is the main advantage of a Galois connection (GC).

Notation

A GC can be expressed by point-wise equivalence (209)

$$f\ x \leq y \ \equiv\ x \sqsubseteq g\ y$$

or by the equivalent relational equality (210),

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

as we have seen.

Abbreviated notation

$$f \vdash g \tag{211}$$

is used instead of (210) wherever the orders are implicit from the context.

Basic properties

For preorders in

$$(A, \leq) \underset{f}{\overset{g}{\rightleftarrows}} (B, \sqsubseteq) \tag{212}$$

the two *cancellation* laws hold:

$$(f \cdot g)a \leq a \quad \text{and} \quad b \sqsubseteq (g \cdot f)b \tag{213}$$

— recall exercise 95 for the case of whole division.

Distribution laws

$$f(b \sqcup b') = (f\ b) \vee (f\ b') \tag{214}$$

$$g(a \wedge a') = (g\ a) \sqcap (g\ a') \tag{215}$$

Basic properties

These hold wherever both preorder are lattices, that is, wherever suprema

$$b \sqcup b' \ \sqsubseteq \ x \ \equiv \ b \sqsubseteq x \ \wedge \ b' \sqsubseteq x \tag{216}$$

and infima

$$x \ \sqsubseteq \ b \sqcap b' \ \equiv \ x \sqsubseteq b \ \wedge \ x \sqsubseteq b' \tag{217}$$

exist. (Similarly for A, $\leq$, $\vee$, $\wedge$.)

Exercise 96: Resort to indirect equality to prove any of (214) or (215).
□

Other properties

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 ($g^{\flat}$)

Moreover, if (f, g) are Galois connected,

- f and g are **monotonic**
- f (g) **uniquely** determines g (f) — thus the $\_^{\flat}$, $\_^{\sharp}$ notations
- (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	$f(b \sqcup b') = (f\ b) \vee (f\ b')$	$g(a' \wedge a) = (g\ a') \sqcap (g\ a)$
Monotonicity	$b \sqsubseteq b' \Rightarrow f\ b \leq f\ b'$	$a \leq a' \Rightarrow g\ a \sqsubseteq g\ a'$

Exercise 97: Derive from (209) that both f and g are monotonic. □

Remark

Galois connections originate from the work of the French mathematician Evariste Galois (1811-1832). Their main advantages,

> *simple, generic and highly calculational*

are welcome in proofs in computing, due to their size and complexity, recall E. Dijkstra:

> *elegant* $\equiv$ *simple and remarkably effective.*

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 — are **adjoints** of GCs: see the next 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	$(R \cap S)^\circ = R^\circ \cap S^\circ, (R \cup S)^\circ = R^\circ \cup S^\circ$

Exercise 98: Why is it that converse-monotonicity can be strengthened to an equivalence? □

Example of calculation from the GC

Converse involution:

$$(R^\circ)^\circ = R \tag{218}$$

Indirect proof of (218):

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

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: $R \cdot h = R / h^\circ$

Question: what does R/S mean?

Relational division

In the same way

$$z \times y \leq x \equiv z \leq x \div y$$

means that $x \div y$ is the largest **number** which multiplied by y approximates x,

$$Z \cdot Y \subseteq X \equiv Z \subseteq X / Y \qquad (219)$$

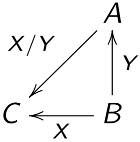
means that X / Y is the largest **relation** which pre-composed Y approximates X.

What is the pointwise meaning of X / Y?

We reason:

First, the types of

$$Z \cdot Y \subseteq X \;\equiv\; Z \subseteq X/Y$$



Next, the calculation:

$$c\ (X/Y)\ a$$

$$\equiv \qquad \{ \text{ introduce points } C \overset{\underline{c}}{\longleftarrow} 1 \text{ and } A \overset{\underline{a}}{\longleftarrow} 1 \ \}$$

$$x(\underline{c}^\circ \cdot (X/Y) \cdot \underline{a})x$$

$$\equiv \qquad \{ \text{ one-point (12) } \}$$

$$x' = x \;\Rightarrow\; x'(\underline{c}^\circ \cdot (X/Y) \cdot \underline{a})x$$

Proceed by going pointfree:

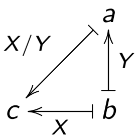
We reason

$$
\begin{array}{ll}
 & id \subseteq \underline{c}^\circ \cdot (X/Y) \cdot \underline{a} \\
\equiv & \quad \{ \text{ shunting rules (Galois connections) } \} \\
 & \underline{c} \cdot \underline{a}^\circ \subseteq X/Y \\
\equiv & \quad \{ \text{ rule (219) — Galois connection } \} \\
 & \underline{c} \cdot \underline{a}^\circ \cdot Y \subseteq X \\
\equiv & \quad \{ \text{ now shunt } \underline{c} \text{ back to the right } \} \\
 & \underline{a}^\circ \cdot Y \subseteq \underline{c}^\circ \cdot X \\
\equiv & \quad \{ \text{ back to points via (47) } \} \\
 & \langle \forall\ b\ :\ a\ Y\ b :\ c\ X\ b \rangle
\end{array}
$$

Outcome

In summary:

$$c\ (X/Y)\ a\ \equiv\ \langle \forall\ b\ :\ a\ Y\ b :\ c\ X\ b \rangle \qquad (220)$$



Example:

$a\ Y\ b$ = passenger a choses flight b

$c\ X\ b$ = company c operates flight b

$c\ (X/Y)\ a$ = company c is the only one trusted by passenger a, that is, a **only flies** c.

Pointwise meaning in full

The full pointwise encoding of Galois connection

$$Z \cdot Y \subseteq X \;\equiv\; Z \subseteq X/Y$$

is:

$$\langle \forall\ c, b\ :\ \langle \exists\ a\ :\ cZa :\ aYb \rangle :\ cXb \rangle \;\equiv\; \langle \forall\ c, a\ :\ cZa :\ \langle \forall\ b\ :\ aYb :\ cXb \rangle\rangle$$

If we drop variables and regard the uppercase letters as denoting Boolean terms dealing without variable c, this becomes

$$\langle \forall\ b\ :\ \langle \exists\ a\ :\ Z :\ Y \rangle :\ X \rangle \;\equiv\; \langle \forall\ a\ :\ Z :\ \langle \forall\ b\ :\ Y :\ X \rangle\rangle$$

recognizable as the **splitting** rule (7) of the Eindhoven calculus.

Put in other words: **existential** quantification is **lower** adjoint of **universal** quantification.

Exercises

Exercise 99: Prove the equalities

$$X \cdot f = X / f^{\circ} \tag{221}$$
$$X / \bot = \top \tag{222}$$
$$\top / Y = \top \tag{223}$$

and check their pointwise meaning. □

Exercise 100: Define

$$X \setminus Y = (Y^{\circ} / X^{\circ})^{\circ} \tag{224}$$

and infer:

$$a(R \setminus S)c \equiv \langle \forall\, b \,:\, b\, R\, a :\, b\, S\, c \rangle \tag{225}$$
$$R \cdot X \subseteq Y \equiv X \subseteq R \setminus Y \tag{226}$$

□

Relational division

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

that is,

$$X \cdot R \subseteq Y \equiv X \subseteq Y / R \qquad (227)$$

$$R \cdot X \subseteq Y \equiv X \subseteq R \setminus Y \qquad (228)$$

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

$$R \cdot (S \cup T) = (R \cdot S) \cup (R \cdot T)$$

$$(S \cup T) \cdot R = (S \cdot R) \cup (T \cdot R)$$

$(\setminus R)$ and $(/R)$ are monotonic and distribute over $\cap$.

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 the universal properties of domain and range

$$\delta\, R \subseteq \Phi \quad \equiv \quad R \subseteq \top \cdot \Phi$$
$$\rho\, R \subseteq \Phi \quad \equiv \quad R \subseteq \Phi \cdot \top$$

— recall (126) and (127) — are Galois connections, and so

$$\delta\,(S \cup R) \quad = \quad \delta\, S \cup \delta\, R$$
$$\top \cdot (\Phi \cap \Psi) \quad = \quad \top \cdot \Phi \cap \top \cdot \Psi$$

hold — similarly for ρ and $(\cdot\top)$.

Other operators

$(f\ X) \subseteq Y \equiv X \subseteq (g\ Y)$			
Description	$f = g^\flat$	$g = f^\sharp$	**Obs.**
implication	$(R\cap)$	$(R\Rightarrow)$	$b(R \Rightarrow X)a \equiv bRa \Rightarrow bXa$
difference	$(\_ - R)$	$(R\cup)$	

Thus the universal properties of implication and difference,

$$R \cap X \subseteq Y \quad \equiv \quad X \subseteq R \Rightarrow Y$$
$$X - R \subseteq Y \quad \equiv \quad X \subseteq R \cup Y$$

are GCs — etc, etc

Exercise 101: Show that $R \cap (R \Rightarrow Y) \subseteq Y$ ("modus ponens") holds and that $R - R = \bot - R = \bot$. □

Exercises

Exercise 102: Let $\mathcal{P}A = \{S : S \subseteq A\}$ and let $A \xleftarrow{\in} \mathcal{P}A$ denote the membership relation $a \in S$, for any a and S. What does the relation $\in \setminus \in$ mean? □

Exercise 103: Show that the relation $\in \setminus \in$ of the previous exercise is reflexive and transitive. □

Exercise 104: Prove that equality

$$(R \setminus S) \cdot f \quad = \quad R \setminus (S \cdot f) \tag{229}$$

holds. □

Exercises

Exercise 105: (a) Show that $R \subseteq \bot / S^\circ \equiv \delta\, R \cap \delta\, S = \bot$; (b) Then use indirect equality to infer the universal property of term $R \cap \bot / S^\circ$ — the largest sub-relation of R whose domain is disjoint of that of S. □

Exercise 106: The relational *overriding* combinator,

$$R \dagger S \;=\; S \cup R \cap \bot / S^\circ \tag{230}$$

means the relation which contains the whole of S and that part of R where S is undefined — read $R \dagger S$ as "R overridden by S". (a) Show that $\bot \dagger S = S$ and that $R \dagger \bot = R$; (b) Infer the universal property:

$$X \subseteq R \dagger S \;\;\equiv\;\; X - S \subseteq R \wedge \delta\,(X - S) \cdot \delta\, S = \bot \tag{231}$$

□

Binary adjoints

Recall the universal property of $\cup$ (65), $R \cup S \ \subseteq \ X \equiv R \subseteq X \ \wedge \ S \subseteq X$, which can be written thus

$$\cup(R, S) \ \subseteq \ X \quad \equiv \quad (R, S)(\subseteq \times \subseteq)(X, X)$$

or even as

$$\cup(R, S) \ \subseteq \ X \quad \equiv \quad (R, S)(\subseteq \times \subseteq)(\Delta X)$$

where $\Delta X = (X, X)$. Clearly,

$$\cup \quad \vdash \quad \Delta$$

Similarly, the universal property of $\cap$ (64) can be captured by

$$\Delta \quad \vdash \quad \cap$$

since $(X, X)(\subseteq \times \subseteq)(R, S) \equiv X \subseteq \cap(R, S)$.

A glimpse of GC (generic) algebra

Assume $f \vdash g$ and $f' \vdash g'$ hold in:

Identity

$$id \vdash id$$

Composition

$$f \cdot f' \vdash g' \cdot g$$

Converse (symmetry)

$$f \vdash g \;\equiv\; g \vdash f$$

Functors (preorders)

$$\mathsf{F}f \vdash \mathsf{F}g$$

Splitting (lattices)

$$\langle f, f' \rangle \vdash \sqcap \cdot (g \times g')$$

In particular, for $f, f' := id$, $g, g' := id$:

$$\triangle \vdash \sqcap \qquad (232)$$

for $\triangle x = (x, x)$.

Application I — Hoare Logic

Handling Hoare triples in relation algebra

As application of the above we show next how to handle **Hoare triples** such as

$$\{p\}P\{q\} \tag{233}$$

in relation algebra. First we spell out the meaning of (233):

$$\langle \forall\ s\ :\ p\ s :\ \langle \forall\ s'\ :\ s \xrightarrow{P} s'\ :\ q\ s' \rangle \rangle \tag{234}$$

that is:

> *if program P is in state s satisfying condition p, and it moves to state s', then s' satisfies q.*

In other words:

> *Condition p holding before P executes is* **sufficient** *for condition q to hold after P executes.*

Handling Hoare triples in relation algebra

Let $[\![P]\!]$ denote the state transition relation of P, that is $s'[\![P]\!]s$ means the same as $s \xrightarrow{P} s'$.

Then (234) re-writes as follows:

$$\langle \forall\ s\ :\ p\ s : \ \langle \forall\ s'\ :\ s'[\![P]\!]s :\ q\ s' \rangle\rangle$$

$\equiv$ { coreflexives }

$$\langle \forall\ s\ :\ s \Phi_p s : \ \langle \forall\ s'\ :\ s'[\![P]\!]s :\ s' \Phi_q s' \rangle\rangle$$

$\equiv$ { $\top$; coreflexives }

$$\langle \forall\ s, s"\ :\ s \Phi_p s" : \ \langle \forall\ s'\ :\ s'[\![P]\!]s :\ s' (\Phi_q \cdot \top) s'' \rangle\rangle$$

$\equiv$ { recall (225) and remove variables }

$$\Phi_p \subseteq [\![P]\!] \setminus (\Phi_q \cdot \top)$$

Handling Hoare triples in relation algebra

Finally:

$$
\begin{array}{ll}
& \Phi_p \subseteq [\![P]\!] \setminus (\Phi_q \cdot \top) \\
\equiv & \quad \{ \text{ GC of division (228) } \} \\
& [\![P]\!] \cdot \Phi_p \subseteq \Phi_q \cdot \top \\
\equiv & \quad \{ \text{ (118) } \} \\
& [\![P]\!] \cdot \Phi_p \subseteq \Phi_q \cdot [\![P]\!]
\end{array}
$$

Comparing this with the meaning of **contract** $\Phi_q \xleftarrow{f} \Phi_p$ — recall (143) — we realize that they are the same in case $[\![P]\!]$ is a function — P deterministic and wholly defined.

Hoare triples are contracts

In summary:

The meaning of Hoare triple $\{p\}P\{q\}$ *is the* **contract**

$$[\![P]\!] \cdot \Phi_p \subseteq \Phi_q \cdot [\![P]\!] \tag{235}$$

where $[\![P]\!]$ *denotes the state transition semantics of* P.

We will write

$$\Phi_p \xrightarrow{P} \Phi_q$$

to mean (235) which, as seen above, is the same as

$$[\![P]\!] \cdot \Phi_p \subseteq \Phi_q \cdot \top \tag{236}$$

Hoare triples are GCs

In turn, (236) is equivalent to

$$\Phi_p \subseteq [\![P]\!] \setminus (\Phi_q \cdot \top) \cap id$$

Thanks to GC (127), (236) is also equivalent to

$$\rho\,([\![P]\!] \cdot \Phi_p) \subseteq \Phi_q$$

Thus we have the following Galois connection for Hoare triples, where P, Φ and Ψ abbreviate $[\![P]\!]$, Φ_p and Φ_q, respectively:

$$\underbrace{\rho\,(P \cdot \Phi)}_{f\ \Phi} \subseteq \Psi \quad \equiv \quad \Phi \subseteq \underbrace{P \setminus (\Psi \cdot \top) \cap id}_{g\ \Psi} \tag{237}$$

Adjoints f and g are known as **predicate transformers**.

Hoare triples are GCs

The usual notation for $g\ \Psi$ is $P \setminus\!\!\!\bullet\ \Psi$ — the **weakest** (liberal) **pre-condition** (WP) for Ψ to hold on the outputs of P.

Dually, $f\ \Phi = \rho\,(P \cdot \Phi)$ is known as the **strongest post-condition** (SP) holding on all outputs of P restricted by Φ on the input.

These concepts are independent of their use in Hoare logic. In general, given a binary relation $B \xleftarrow{R} A$ and coreflexives $A \xleftarrow{\Phi} A$ and $B \xleftarrow{\Psi} B$, we define

$$
\begin{array}{rcll}
\Phi \xrightarrow{R} \Psi & \equiv & R \cdot \Phi \subseteq \Psi \cdot R & (238) \\
 & \equiv & \Phi \subseteq R \setminus\!\!\!\bullet\ \Psi & (239)
\end{array}
$$

which extends **functional contracts** to arbitrary relations.

Exercises

Exercise 107: Prove

$$\square \qquad id \xleftarrow{R} \Phi \equiv \textsc{True} \equiv \Phi \xleftarrow{R} \bot \tag{240}$$

Exercise 108: Prove the special cases:

- WP of a function f:

$$f \setminus\!\!\bullet\, \Phi_q = \lambda a.q(f\ a) \tag{241}$$

- SP of a function f:

$$\rho\,(f \cdot \Phi_p) = \lambda b.b \in \{f\ a \mid p\ a\} \tag{242}$$

NB: recall that (241) has been used several times earlier on in contract calculation. □

Exercises

Exercise 109: The formal meaning of (imperative) code sequential composition is

$$[\![\mathtt{P};\mathtt{Q}]\!] = [\![\mathtt{Q}]\!] \cdot [\![\mathtt{P}]\!]$$

Show that the following rule of the Hoare logic of programs,

$$\frac{\{p\}\mathtt{P}\{q\}\ ,\ \{q\}\mathtt{Q}\{s\}}{\{p\}\mathtt{P};\mathtt{Q}\{s\}}$$

is an instance of the following relational typing rule:

$$\Psi \xleftarrow{R \cdot S} \Phi \quad \Leftarrow \quad \Psi \xleftarrow{R} \Upsilon \ \wedge\ \Upsilon \xleftarrow{S} \Phi \qquad (243)$$

□

Exercises

Exercise 110: Prove the "trading rule":

$$\Upsilon \xleftarrow{R} \Phi \cdot \Psi \quad \equiv \quad \Upsilon \xleftarrow{R \cdot \Phi} \Psi \tag{244}$$

□

Exercise 111: Re-write the following "contract splitting" rule,

$$\Psi_1 \cdot \Psi_2 \xleftarrow{R} \Phi \quad \equiv \quad \Psi_1 \xleftarrow{R} \Phi \ \wedge \ \Psi_2 \xleftarrow{R} \Phi \tag{245}$$

in Hoare logic. Then prove (245). □

WP calculus

Facts (237) and (239) show that whatever one can do in Hoare logic can be done with Dijkstra's WPs.

Let us show an example by converting (245) to WP-calculus:

$$\Upsilon \cdot \Psi \xleftarrow{R} \Phi \quad \equiv \quad \Upsilon \xleftarrow{R} \Phi \ \wedge \ \Psi \xleftarrow{R} \Phi$$

$$\equiv \qquad \{ \text{ WPs (239) three times } \}$$

$$\Phi \subseteq R \mathbin{\backslash\!\!\bullet} (\Upsilon \cdot \Psi) \ \equiv \ \Phi \subseteq R \mathbin{\backslash\!\!\bullet} \Upsilon \wedge \Phi \subseteq R \mathbin{\backslash\!\!\bullet} \Psi$$

$$\equiv \qquad \{ \text{ coreflexives (112) ; meet-universal (64) } \}$$

$$\langle \forall\ \Phi\ ::\ \Phi \subseteq R \mathbin{\backslash\!\!\bullet} (\Upsilon \cdot \Psi) \ \equiv \ \Phi \subseteq (R \mathbin{\backslash\!\!\bullet} \Upsilon) \cap (R \mathbin{\backslash\!\!\bullet} \Psi) \rangle$$

$$\equiv \qquad \{ \text{ meet of correflexives; indirect equality (69) } \}$$

$$R \mathbin{\backslash\!\!\bullet} (\Upsilon \cdot \Psi) \ = \ (R \mathbin{\backslash\!\!\bullet} \Upsilon) \cdot (R \mathbin{\backslash\!\!\bullet} \Psi)$$

WP calculus

A more interesting example is the transformation of the WP-rule for sequential composition

$$(S \cdot R) \mathbin{\backslash\!\!\!\bullet} \Phi \quad = \quad R \mathbin{\backslash\!\!\!\bullet} (S \mathbin{\backslash\!\!\!\bullet} \Phi) \tag{246}$$

into a contract:

$$
\begin{array}{cl}
 & R \mathbin{\backslash\!\!\!\bullet} (S \mathbin{\backslash\!\!\!\bullet} \phi) \quad = \quad (S \cdot R) \mathbin{\backslash\!\!\!\bullet} \phi \\
\equiv & \qquad \{ \text{ indirect equality (69) } \} \\
 & \psi \subseteq R \mathbin{\backslash\!\!\!\bullet} (S \mathbin{\backslash\!\!\!\bullet} \phi) \quad \equiv \quad \psi \subseteq (S \cdot R) \mathbin{\backslash\!\!\!\bullet} \phi \\
\equiv & \qquad \{ \text{ (239) twice } \} \\
 & (S \mathbin{\backslash\!\!\!\bullet} \phi) \xleftarrow{R} \psi \quad \equiv \quad \phi \xleftarrow{(S \cdot R)} \psi
\end{array}
\tag{247}
$$

The outcome, still involving the $\mathbin{\backslash\!\!\!\bullet}$ operator, is an advantageous replacement for (243), since it is an equivalence.

Exercises

Exercise 112: Show that $\rho\, R \xleftarrow{R} \delta\, R$ holds. However, WP $R \mathbin{\backslash\!\!\!\bullet} (\rho\, R) = id$ rather than $\delta\, R$. Explain why. □

Exercise 113: Show that $\rho\, R \xleftarrow{R} \delta\, R$ holds. However, WP $R \mathbin{\backslash\!\!\!\bullet} (\rho\, R) = id$ rather than $\delta\, R$. Explain why. □

Exercise 114: The two "shunting" rules for S a simple relation,

$$S \cdot R \subseteq Q \quad \equiv \quad (\delta\, S) \cdot R \quad \subseteq \quad S^\circ \cdot Q \tag{248}$$

$$R \cdot S^\circ \subseteq Q \quad \equiv \quad R \cdot \delta\, S \quad \subseteq \quad Q \cdot S \tag{249}$$

are "almost" Galois connections. (a) Derive the following variants concerning coreflexives,

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

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

referred to earlier on as the *closure properties* (113) and (114), respectively; (b) prove either (248) or (249) by cyclic implication (vulg.

Application II — Optimization calculus

Programming is optimization

Abstract models are derived from requirements by ignoring unnecessary detail.

This often results in models whose operations are **vague** or **non-deterministic**.

Such operations, often recorded as **pre**/**post** condition pairs, are binary **relations**.

As computers cannot handle vagueness, deriving code for such operations calls for **determinization** — some way to convert such relations into functions.

This process is known as **model refinement**, and it is performed in a stepwise manner; however, how does one control it? What is the **guiding principle** (if any)?

Programming is optimization

Recall (203), one of the definitions given for whole division:

$$x \div y = \langle \bigvee\ z\ ::\ z \times y \leq x \rangle$$

Given some y, term $z \times y \leq x$ denotes a binary relation with input x and output z. But not every output z is acceptable — (203) tells that one wants **the largest** such z.

So there is an **ordering** ($\leq$) on the outputs ($\mathbb{N}_0$) telling what the **optimization** principle should be: *largest* wrt. $\mathbb{N}_0 \xleftarrow{\leq} \mathbb{N}_0$.

Whole division is (perhaps) the first **optimization** problem one solves at school; programmers do it **all the time**, most often unconsciously!

Programming is optimization

Another example is provided by the Galois connection which specifies the *take* function available in Haskell, for instance:

$$\text{length}\ ys \leq n \ \wedge\ ys \preceq xs \quad \equiv \quad ys \preceq take\ n\ xs \tag{250}$$

Here the ordering on outputs is the **prefix** relation ($\preceq$) on lists.

For each n, term $\text{length}\ ys \leq n \ \wedge\ ys \preceq xs$ tells which outputs ys are candidates for $take\ n\ xs$.

But only one of these is acceptable — the **longest** such prefix, which is **optimal** with respect to the prefix ordering.

Exercise

Exercise 115: Before implementing *take* one can start proving properties about this function solely relying on (250):

- Show that

$$take\ (\mathrm{length}\ xs)\ xs = xs$$

  holds.

- Resort to indirect equality over $\preceq$ in proving

$$take\ n\ (take\ m\ xs) \quad = \quad take\ (min\ n\ m)\ xs$$

  where *min*, the minimum of two natural numbers, is given by the obvious Galois connection.

□

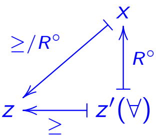
Optimization in an abstract setting

Let us once again go back to (203) and spell out the meaning of its supremum:

$$
\begin{array}{ll}
z(\div y)x & \equiv \quad z \times y \leq x \ \wedge \ \langle \forall\, z' \ :\ z' \times y \leq x :\ z \geq z' \rangle \\
& \equiv \qquad \{ \text{ define } z\ R\ x = z \times y \leq x \ \} \\
& \qquad \underbrace{z \times y \leq x}_{z\ R\ x} \ \wedge \ \underbrace{\langle \forall\, z' \ :\ \underbrace{z' \times y \leq x}_{x\ R^{\circ}\ z'} :\ z \geq z' \rangle}_{z(\geq / R^{\circ})x}
\end{array}
$$

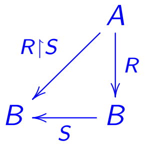
Im summary:

$$(\div y) = R \cap \geq / R^{\circ} \ \textbf{where}\ R = (\times y)^{\circ} \cdot \leq \qquad (251)$$



Optimization in an abstract setting

Generalization: given any relation $B \xleftarrow{R} A$ and an **optimization** criterion $B \xleftarrow{S} B$ on its outputs,



define a new relational combinator $R \upharpoonright S$ (read: R optimized by S, or R "shrunk" by S) as follows:

$$R \upharpoonright S \;=\; \underbrace{R}_{easy} \cap \underbrace{S / R^{\circ}}_{hard} \tag{252}$$

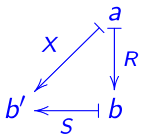
The "hard" term specifies the optimization taking place.

Optimization in an abstract setting

By standard application of **indirect equality** to (252) one obtains the **universal property** of the "shrinking" operator:

$$X \subseteq R \upharpoonright S \quad \equiv \quad X \subseteq R \ \wedge \ X \cdot R^{\circ} \subseteq S \qquad (253)$$

This ensures $R \upharpoonright S$ as the largest sub-relation X of R such that, for all $b', b \in B$, if there exists $a \in A$ such that $b' X a \wedge b R a$, then $b' S b$ holds ("b' better than b").



(253) can be regarded as a GC between the set of all **subrelations** of R and the set of **optimization criteria** on its outputs.

Optimization calculus

Both the definition of $R \upharpoonright S$ and its universal property (253) provide a rich setting for exploiting **generic properties** of **optimization** in this abstract setting.

Below we give a brief account of such algebra, as obtained using relational calculus.

The interested reader is referred to the works by Mu and Oliveira (2012) and Oliveira and Ferreira (2012) for a more complete account of optimization by shrinking, with applicatons to software design.

Basic properties of $R \upharpoonright S$

Chaotic optimization:

$$R \upharpoonright \top \quad = \quad R \tag{254}$$

Impossible optimization:

$$R \upharpoonright \bot \quad = \quad \bot \tag{255}$$

"Brute force" determinization:

$$R \upharpoonright id \quad = \quad \text{largest deterministic fragment of } R \tag{256}$$

Thus $R \upharpoonright id$ is the part of R which cannot be further refined.

Exercise 116: Prove the two first equalities above. □

Basic properties of $R \upharpoonright S$

$R \upharpoonright id$ is the extreme case of the fact which follows:

$$R \upharpoonright S \text{ is simple } \Leftarrow S \text{ is anti-symmetric} \tag{257}$$

Thus anti-symmetric criteria always lead to determinism, possibly at the sacrifice of totality. Clearly: for R simple,

$$R \upharpoonright S = R \quad \equiv \quad \mathsf{img}\, R \subseteq S \tag{258}$$

Thus (functions)

$$f \upharpoonright S = f \quad \Leftarrow \quad S \text{ is reflexive} \tag{259}$$

Basic properties of $R \upharpoonright S$

Pre-condition fusion:

$$(R \upharpoonright S) \cdot \Phi \quad = \quad (R \cdot \Phi) \upharpoonright S \tag{260}$$

Two function fusion rules

$$(R \upharpoonright S) \cdot f \quad = \quad (R \cdot f) \upharpoonright S \tag{261}$$

$$(f \cdot R) \upharpoonright S \quad = \quad f \cdot (R \upharpoonright S_f) \tag{262}$$

where S_f abbreviates $f^\circ \cdot S \cdot f$.

Exercise 117: Show that, for S a preorder, S_f above is also a preorder.
□

Basic properties of $R \upharpoonright S$

Union:

$$(R \cup S) \upharpoonright Q = (R \upharpoonright Q) \cap Q/S^{\circ} \cup (S \upharpoonright Q) \cap Q/R^{\circ} \qquad (263)$$

This has a number of corollaries, namely a **conditional rule**,

$$(p \rightarrow R\ ,\ T) \upharpoonright S \quad = \quad p \rightarrow (R \upharpoonright S)\ ,\ (p \upharpoonright S) \qquad (264)$$

the **distribution** over alternatives (77),

$$[R\ ,S] \upharpoonright U \quad = \quad [R \upharpoonright U\ ,S \upharpoonright U] \qquad (265)$$

and the "**function competition**" rule:

$$(f \cup g) \upharpoonright S \quad = \quad (f \cap S \cdot g) \cup (g \cap S \cdot f) \qquad (266)$$

since $S/g^{\circ} = S \cdot g$.

"Function competition" rule

With points:

$$y((f \cup g) \upharpoonright S)x \equiv \begin{cases} y = f\ x \wedge (f\ x)S(g\ x) \\ \vee \\ y = g\ x \wedge (g\ x)S(f\ x) \end{cases}$$

that is: f (resp. g) "wins" wherever it is better than g (resp. f) wrt. S. For instance,

$$abs = (id \cup sim) \upharpoonright \geq$$

for $sim\ x = -x$, cf.

$$\begin{aligned} y = abs\ x &\equiv y = x \wedge x \geq -x \ \vee \ y = -x \wedge -x \geq x \\ &\equiv y = x \wedge x \geq 0 \ \vee \ y = -x \wedge 0 \geq x \end{aligned}$$

$R \upharpoonright S$ on data

Combinator $R \upharpoonright S$ also makes sense when R and S are finite, relational data structures (eg. tables in a database).

Example of $R \upharpoonright S$ in **data-processing**: given

Examiner	*Mark*	*Student*
Smith	10	*John*
Smith	11	*Mary*
Smith	15	*Arthur*
Wood	12	*John*
Wood	11	*Mary*
Wood	15	*Arthur*

and wishing to "choose the best mark", project over $Mark, Student$ and optimize over the $\geq$ ordering on $Mark$ (next slide):

$R \restriction S$ on data

$$\left(\begin{array}{c|c} \mathit{Mark} & \mathit{Student} \\ \hline 10 & \mathit{John} \\ 11 & \mathit{Mary} \\ 12 & \mathit{John} \\ 15 & \mathit{Arthur} \end{array}\right) \restriction \geq \;=\; \begin{array}{c|c} \mathit{Mark} & \mathit{Student} \\ \hline 11 & \mathit{Mary} \\ 12 & \mathit{John} \\ 15 & \mathit{Arthur} \end{array}$$

Relational shrinking can also be used for induction-free reasoning about sequences (lists), welcome in **Alloy** where no explicit recursion is available.

Example of $R \restriction S$ in **list-processing**: given a sequence $A \xleftarrow{S} \mathbb{N}$,

$$A \xleftarrow{nub\ S} \mathbb{N} \quad \triangleq \quad (S^\circ \restriction \leq)^\circ$$

removes all duplicates while keeping the first instances. (Data in $\mathbb{N}$ could be regarded as "time stamps".)

Galois connections (211) as optimization problems

$$f^\circ \cdot (\leq) \ = \ (\sqsubseteq) \cdot g$$

$\equiv$ { ping-pong }

$$(\sqsubseteq) \cdot g \subseteq f^\circ \cdot (\leq) \wedge f^\circ \cdot (\leq) \ \subseteq \ (\sqsubseteq) \cdot g$$

$\equiv$ { converses }

$$(\sqsubseteq) \cdot g \subseteq f^\circ \cdot (\leq) \wedge (f^\circ \cdot (\leq))^\circ \ \subseteq \ g^\circ \cdot (\sqsupseteq)$$

$\equiv$ { since f is monotonic (see exercise 119 below) }

$$\underbrace{g \ \subseteq \ f^\circ \cdot (\leq)}_{\text{“easy”}} \quad \wedge \quad \underbrace{g \cdot (f^\circ \cdot (\leq))^\circ \ \subseteq \ (\sqsupseteq)}_{\text{“hard”}},$$

$\equiv$ { universal property (253) }

$$g \subseteq (f^\circ \cdot (\leq)) \upharpoonright (\sqsupseteq) \qquad (267)$$

Galois connections as optimization problems

Comments:

- Given the two orderings $(\leq)$ and $(\sqsupseteq)$ and the "easy adjoint" f, implementing the "hard adjoint" amounts to solving the inequation (267) for g.
- We have already seen an instance of this result in (251), for whole division.

Question:

> *Implementations are usually recursive. Where in (267) is the "guideline" for introducing recursion in the calculations ?*

Since $g \subseteq (f^\circ \cdot (\leq)) \upharpoonright (\sqsupseteq)$ expresses an optimization by $(\sqsupseteq)$, it is this ordering which controls the implementation process. How?

Exercises

Assume a generic Galois connection $f^\circ \cdot \leq \;=\; \sqsubseteq \cdot g$ in the following exercises.

Exercise 118: Show that f monotonicity, $x \sqsubseteq y \;\Rightarrow\; f\ x \leq f\ y$, can be written point-free as

$$(\sqsubseteq) \cdot f^\circ \subseteq f^\circ \cdot (\leq), \tag{268}$$

□

Exercise 119: Show that, once (268) is assumed, the following equivalence holds:

$$g \;\subseteq\; f^\circ \cdot (\leq) \;\;\equiv\;\; (\sqsubseteq) \cdot g \;\subseteq\; f^\circ \cdot (\leq) \tag{269}$$

Suggestion: do a "ping-pong" proof. □

Application III — Optimization versus induction

Optimizing over inductive relations

As shown in (Bird and de Moor, 1997) and (Mu and Oliveira, 2012), most often the orderings involved in **program optimization** are **inductive** relations.

- Inductive orderings lead to recursive programs
- "Greedy algorithms" and "dynamic programming" studied in this way in the *Algebra of Programming* book (Bird and de Moor, 1997).
- Complexity of the approach puts many readers off (need for always transposing relations to powerset functions; ...)

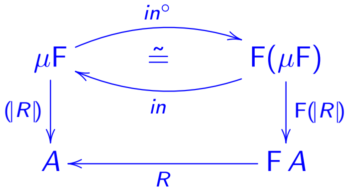
What's new in (Mu and Oliveira, 2012):

> $R \upharpoonright S$ *algebra* **greatly simplifies** *and generalizes the calculation of programs from such specifications. (Notably, there is no need for power transpose.)*

Folds ($\kappa\alpha\tau\alpha$s)

In general, for F a polynomial functor (relator) and initial

$\mu F \xleftarrow{in} F(\mu F)$,



there is a unique solution to equation $X = R \cdot F\,X \cdot in^\circ$ — thus universal property:

$$X = (|R|) \;\equiv\; X \cdot in = R \cdot F\,X \qquad (270)$$

(Read $(|R|)$ as "fold R" or "$\kappa\alpha\tau\alpha$ R".)

Relational folds

It is very easy to show that

$$(\!|in|\!) \quad = \quad id \tag{271}$$

holds — just make $X = id$ in (270) and solve for R (this is known as the **reflexion** property).

Example: $in = [nil, cons]$ for lists. Reflexion (271) means that the function $f = (\![\, nil, cons\,]\!)$ is bound to be the identity, cf.

$$f\ [\,] = [\,]$$
$$f(cons(a, x)) = cons(a, f\ x)$$

Now suppose we have $R = [nil, cons \cup nil]$ in (270). What is the meaning of $(\![\, nil, cons \cup nil\,]\!)$?

Relational folds

Unfolding $X = (\![\, nil, cons \cup nil \,]\!)$ we get

$$X \cdot [nil, cons] = [nil, cons \cup nil] \cdot (id + id \times X)$$

that is, $X \cdot nil = nil$ and $X \cdot cons = (cons \cup nil) \cdot (id \times X)$.

Introducing variables in $X \cdot nil = nil$ we get $y\ X\ [\,] \equiv y = [\,]$ since $nil\ \_ = [\,]$. That is, $[\,]\ X\ [\,] \equiv \textsc{True}$. Doing the same for the other clause we get:

$$y\ X\ (a : x) \ \equiv\ y = [\,] \vee \langle \exists\, x' \ :\ x'\ X\ x :\ \ y = a : x' \rangle$$

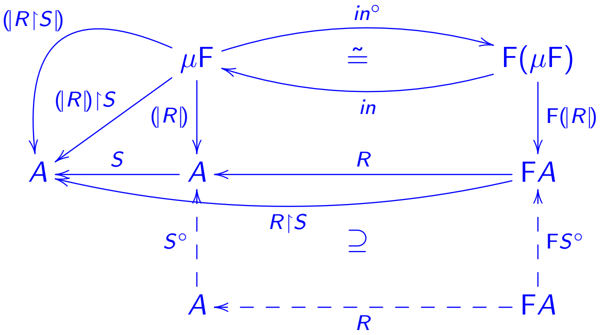
Thus $(\![\, nil, cons \cup nil \,]\!)$ is the **prefix** relation:

$$(\preceq) = (\![\, nil, cons \cup nil \,]\!)$$

The "Greedy" theorem

$$(\!| R \upharpoonright S |\!) \subseteq (\!| R |\!) \upharpoonright S \quad \Leftarrow \quad S^\circ \xleftarrow{\;R\;} \mathsf{F}\, S^\circ \tag{272}$$

for S transitive. (**NB**: $R \xleftarrow{\;X\;} S$ means $X \cdot S \subseteq R \cdot X$) In a diagram, where the side condition is depicted in dashed arrows:



Proof: see (Mu and Oliveira, 2012).

Example of greedy programming

The *msp* problem ("maximum sum prefix"), whose spec

$msp :: [Int] \leftarrow [Int]$
$y\ msp\ x = y\ is\ a\ prefix\ of\ x\ that\ yields\ the\ maximum\ sum$

translates into ($\preceq = (\![nil, cons \cup nil]\!)$ is the prefix ordering)

$$y\ msp\ x \quad \Rightarrow \quad y \preceq x \ \wedge \ \langle \forall\ z\ :\ z \preceq x :\ sum\ y \geq sum\ z \rangle$$

which in turn PF-transforms into

$$msp \quad \subseteq \quad \preceq \upharpoonright \geq_{sum}$$

(**NB:** not a GC, it is nevertheless a good example to understand greedy programming.)

Example of greedy programming

We calculate:

$$
\begin{array}{cl}
 & msp \subseteq \; \preceq \upharpoonright \geq_{sum} \\
\equiv & \quad \{ \text{ definition of prefix ordering } \} \\
 & msp \subseteq \; (\![nil, cons \cup nil]\!) \upharpoonright \geq_{sum} \\
\Leftarrow & \quad \{ \text{ greedy theorem (272) } \} \\
 & msp \subseteq \; (\![[nil, cons \cup nil] \upharpoonright \geq_{sum}]\!) \\
\equiv & \quad \{ \text{ junc-rule (265) ; determinism of } nil \text{ } \} \\
 & msp \subseteq \; (\![nil, (cons \cup nil) \upharpoonright \geq_{sum}]\!) \\
\equiv & \quad \{ \text{ function competition rule (266) } \} \\
 & msp \subseteq \; (\![nil, (cons \cap \geq_{sum} \cdot nil) \cup (nil \cap \geq_{sum} \cdot cons)]\!)
\end{array}
$$

(Side condition ignored for brevity.)

Example of greedy programming

Let R abbreviate the inductive step

$$(nil \cap \geq_{sum} \cdot cons) \cup (cons \cap \geq_{sum} \cdot nil)$$

Then $y\ R\ (a : x)$ means

$$y = [\,] \ \wedge\ 0 \geq a + sum\ x \ \vee\ y = a : x \ \wedge\ a + sum\ x \geq 0$$

The case $a + sum\ x = 0$ is **ambiguous,** in the sense that the algorithm may either stop yielding $y = [\,]$ or yield $y = a : x$, where x is the outcome of the recursive step.

As we still have non-determinism, we need to further shrink what we started from,

$$msp \quad = \quad (\preceq \upharpoonright \geq_{sum}) \upharpoonright \preceq \tag{273}$$

to obtain the function which yields the **shortest** such prefix.

Example of greedy programming

Putting everything together, the overall outcome will be, in Haskell syntax:

```
msp [] = []
msp(a:s) = let x = msp s
           in if sum x > -a then a:x else []
```

See more theorems and examples in (Mu and Oliveira, 2012) covering also optimizations which lead to hylomorphisms and anamorphisms.

It turns out that whole division ($x \div y$), *take* etc end up being anamorphisms.

R. Bird and O. de Moor. *Algebra of Programming*. Series in Computer Science. Prentice-Hall, 1997.

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

J.N. Oliveira and M.A. Ferreira. Alloy meets the algebra of programming: a case study, 2012. To appear in IEEE Transactions on Software Engineering.