PF-transform: using Galois connections to structure relational algebra

J.N. Oliveira

Dept. Informática,
Universidade do Minho
Braga, Portugal

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

\[
\begin{array}{c|c}
7 & 2 \\
\hline
1 & 3 \\
\end{array}
\]

\[2 \times 3 + 1 = 7, \text{ “ie.”} \quad 3 = 7 \div 2\]

However

\[
\begin{array}{c|c}
7 & 2 \\
\hline
3 & 2 \\
\end{array}
\]

\[2 \times 2 + 3 = 7 \quad \wedge \quad 2 \neq 7 \div 2\]

\[
\begin{array}{c|c}
7 & 2 \\
\hline
5 & 1 \\
\end{array}
\]

\[2 \times 1 + 5 = 7 \quad \wedge \quad 1 \neq 7 \div 2\]

That is: for some \( r \),

\[
\begin{array}{c|c}
n & d \\
\hline
r & q \\
\end{array}
\]

\[q = n \div d \equiv d \times q + r = n\]

provided \( q \) is the largest such \( q \) (\( r \) smallest)
Example — specifying $x \div y$

First version (literal):

$$x \div y = \langle \bigvee z :: z \times y \leq x \rangle$$  \hspace{1cm} (203)

Second version (involved):

$$z = x \div y \equiv \langle \exists r : 0 \leq r < y : x = z \times y + r \rangle$$  \hspace{1cm} (204)

Third version (clever!):

$$z \times y \leq x \equiv z \leq x \div y \hspace{1cm} (y > 0)$$  \hspace{1cm} (205)

— a so-called Galois connection, as we shall soon see.
Why (205) is better than (203,204)

Equivalence (205),

\[ z \times y \leq x \equiv z \leq x \div y \quad (y > 0) \]

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:
  
  \[ z \times y \leq x \Rightarrow z \leq x \div y \quad (y > 0) \]

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

\[ z \times y \leq x \equiv z \leq x \div y \quad (y > 0) \]

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

\[
q \leq (n \div m) \div d \\
\equiv \{ (205) \} \\
q \times d \leq n \div m \\
\equiv \{ (205) \} \\
(q \times d) \times m \leq n \\
\equiv \{ \times \text{ is associative} \} \\
q \times (d \times m) \leq n \\
\equiv \{ (205) \} \\
q \leq n \div (d \times m) \\
:: \{ \text{indirection (206)} \} \\
(n \div m) \div d = n \div (d \times m)\]
(Generic) indirect equality

Note the use of the (generic) indirect equality rule

\[
\langle \forall q :: q \leq x \equiv q \leq y \rangle \equiv (x = y)
\]  

(206)

valid for any partial order \( \leq \).

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

\[
q \leq (q \times d) \div d
\]

\[
(n \div d) \times d \leq n
\]

and reflexion law:

\[
n \div d \geq 1 \equiv d \leq n
\]  

(207)
Galois connections

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

\[ z \times y \leq x \iff z \leq x \div y \]

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

\[ (A, \leq) \xrightarrow{g} (B, \sqsubseteq) \quad (B, \sqsubseteq) \xrightarrow{f} (A, \leq) \]  \hspace{1cm} (208)

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

\[ f \quad b \leq a \quad \equiv \quad b \sqsubseteq g \quad a \quad \text{(209)} \]

lower adjoint \quad \quad \quad \quad \quad upper adjoint

that is

\[ f^\circ \cdot \leq = \sqsubseteq \cdot g \quad \text{(210)} \]

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

$$f \vdash g$$  \hspace{1cm} (211)

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

For preorders in

\[(A, \leq) \quad \text{and} \quad (B, \sqsubseteq)\]

(212)

the two *cancellation* laws hold:

\((f \cdot g)a \leq a \quad \text{and} \quad b \sqsubseteq (g \cdot f)b\)  \hspace{1cm} (213)

— recall exercise 95 for the case of whole division.

*Distribution* laws

\[
f(b \sqcup b') = (f \cdot b) \lor (f \cdot b') \hspace{1cm} (214)
\]

\[
g(a \land a') = (g \cdot a) \land (g \cdot a') \hspace{1cm} (215)
\]
Basic properties

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

\[ b \sqcup b' \sqsubseteq x \equiv b \sqsubseteq x \land b' \sqsubseteq x \quad (216) \]

and infima

\[ x \sqsubseteq b \sqcap b' \equiv x \sqsubseteq b \land x \sqsubseteq b' \quad (217) \]

exist. (Similarly for \( A, \leq, \lor, \land \).)

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 $^\flat$, $^\#$ 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
(f b) ≤ a ≡ b ⊑ (g a)

<table>
<thead>
<tr>
<th>Description</th>
<th>f = g♭</th>
<th>g = f♯</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>f b = {a : b ⊑ g a}</td>
<td>g a = {b : f b ≤ a}</td>
</tr>
<tr>
<td>Cancellation</td>
<td>f(g a) ≤ a</td>
<td>b ⊑ g(f b)</td>
</tr>
<tr>
<td>Distribution</td>
<td>f(b ⊔ b') = (f b) ∨ (f b')</td>
<td>g(a' ∧ a) = (g a') ∩ (g a)</td>
</tr>
<tr>
<td>Monotonicity</td>
<td>b ⊑ b' ⇒ f b ≤ f b'</td>
<td>a ≤ a' ⇒ g a ⊑ g a'</td>
</tr>
</tbody>
</table>

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,

\[ \text{simple, generic and highly calculational} \]

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

\[ \text{elegant} \equiv \text{simple and remarkably effective}. \]

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

Not only

\[(d \times) q \leq n \equiv q \leq n(\div d)\]

but also the two **shunting rules**,\n
\[(h \cdot) X \subseteq Y \equiv X \subseteq (h^\circ \cdot) Y\]

\[X(\cdot h^\circ) \subseteq Y \equiv X \subseteq Y(\cdot h)\]

as well as **converse**,\n
\[X^\circ \subseteq Y \equiv X \subseteq Y^\circ\]

and so and so forth — are **adjoints** of GCs: see the next slides.
Converse

\[(f X) \subseteq Y \equiv X \subseteq (g Y)\]

<table>
<thead>
<tr>
<th>Description</th>
<th>(f = g^\flat)</th>
<th>(g = f^\sharp)</th>
<th>Obs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>converse</td>
<td>((_)^\circ)</td>
<td>((_)^\circ)</td>
<td>(bR^\circ a \equiv aRb)</td>
</tr>
</tbody>
</table>

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:

\[(R^\circ)^\circ = R\]  \hspace{1cm} (218)

Indirect proof of (218):

\[(R^\circ)^\circ \subseteq Y\]

\[\equiv \{ \circ\text{-universal } X^\circ \subseteq Y \equiv X \subseteq Y^\circ \text{ for } X := R^\circ \} \]

\[R^\circ \subseteq Y^\circ\]

\[\equiv \{ \circ\text{-monotonicity } \}\]

\[R \subseteq Y\]

\[:: \{ \text{indirection } \}\]

\[(R^\circ)^\circ = R\]
Motivation  E+H split  Galois connections  Application I — Hoare Logic  Application II — Optimization calculus  Application III —

Functions

\[(f \ X) \subseteq Y \equiv X \subseteq (g \ Y)\]

<table>
<thead>
<tr>
<th>Description</th>
<th>(f = g^\flat)</th>
<th>(g = f^\sharp)</th>
<th>Obs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>shunting rule</td>
<td>((h \cdot))</td>
<td>((h^\circ \cdot))</td>
<td>NB: (h) is a function</td>
</tr>
<tr>
<td>“converse” shunting rule</td>
<td>((\cdot h^\circ))</td>
<td>((\cdot h))</td>
<td>NB: (h) is a function</td>
</tr>
</tbody>
</table>

Consequences:

Functional equality: \[h \subseteq g \equiv h = k \equiv h \supseteq k\]

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 \textbf{number} which multiplied by \( y \) approximates \( x \),

\[ Z \cdot Y \subseteq X \equiv Z \subseteq X/Y \quad (219) \]

means that \( X/Y \) is the largest \textbf{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:

\[
\begin{align*}
& c(X/Y) a \\
& \equiv \{ \text{introduce points } C <^c 1 \text{ and } A <^a 1 \} \\
& x(c^\circ \cdot (X/Y) \cdot a)x \\
& \equiv \{ \text{one-point (12)} \} \\
& x' = x \Rightarrow x'(c^\circ \cdot (X/Y) \cdot a)x
\end{align*}
\]

Proceed by going pointfree:
We reason

\[ \text{id} \subseteq c^\circ \cdot (X/Y) \cdot a \]

\[ \equiv \{ \text{shunting rules (Galois connections)} \} \]

\[ c \cdot a^\circ \subseteq X/Y \]

\[ \equiv \{ \text{rule (219) — Galois connection} \} \]

\[ c \cdot a^\circ \cdot Y \subseteq X \]

\[ \equiv \{ \text{now shunt } c \text{ back to the right} \} \]

\[ a^\circ \cdot Y \subseteq c^\circ \cdot X \]

\[ \equiv \{ \text{back to points via (47)} \} \]

\[ \langle \forall b : a Y b : c X b \rangle \]
In summary:

\[ c \left( \frac{X}{Y} \right) a \equiv \langle \forall \ b : a \ Y \ b : c \ X \ b \rangle \]

Example:

\[ a \ Y \ b = \text{passenger } a \ \text{chooses flight } b \]
\[ c \ X \ b = \text{company } c \ \text{operates flight } b \]
\[ c \left( \frac{X}{Y} \right) a = \text{company } c \ \text{is the only one trusted by passenger } a, \ \text{that is, } a \ \text{only flies } c. \]
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

\[ X \cdot f = X / f^\circ \]  \hspace{1cm} (221)

\[ X / \bot = \top \]  \hspace{1cm} (222)

\[ \top / Y = \top \]  \hspace{1cm} (223)

and check their pointwise meaning. □

Exercise 100: Define

\[ X \setminus Y = (Y^\circ / X^\circ)^\circ \]  \hspace{1cm} (224)

and infer:

\[ a(R \setminus S)c \equiv \langle \forall b : b R a : b S c \rangle \]  \hspace{1cm} (225)

\[ R \cdot X \subseteq Y \equiv X \subseteq R \setminus Y \]  \hspace{1cm} (226)

□
Relational division

\[(f \ X) \subseteq Y \equiv X \subseteq (g \ Y)\]

<table>
<thead>
<tr>
<th>Description</th>
<th>(f = g^\flat)</th>
<th>(g = f^#)</th>
<th>Obs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>right-division</td>
<td>((\cdot R))</td>
<td>((/ R))</td>
<td>right-factor</td>
</tr>
<tr>
<td>left-division</td>
<td>((R \cdot))</td>
<td>((R \backslash))</td>
<td>left-factor</td>
</tr>
</tbody>
</table>

that is,

\[
X \cdot R \subseteq Y \equiv X \subseteq Y / R \quad (227)
\]

\[
R \cdot X \subseteq Y \equiv X \subseteq R \backslash Y \quad (228)
\]

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

\[
R \cdot (S \cup T) = (R \cdot S) \cup (R \cdot T)
\]

\[
(S \cup T) \cdot R = (S \cdot R) \cup (T \cdot R)
\]

\((\backslash R)\) and \((/ R)\) are monotonic and distribute over \(\cap\).
## Domain and range

\[(f \ X) \subseteq Y \equiv X \subseteq (g \ Y)\]

<table>
<thead>
<tr>
<th>Description</th>
<th>f = g♭</th>
<th>g = f♯</th>
<th>Obs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>δ</td>
<td>(⊤·)</td>
<td>lower ⊆ restricted to coreflexives</td>
</tr>
<tr>
<td>range</td>
<td>ρ</td>
<td>(·⊤)</td>
<td>lower ⊆ restricted to coreflexives</td>
</tr>
</tbody>
</table>

Thus the universal properties of domain and range

\[\delta R \subseteq \Phi \equiv R \subseteq \top \cdot \Phi\]
\[\rho R \subseteq \Phi \equiv R \subseteq \Phi \cdot \top\]

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

\[\delta (S \cup R) = \delta S \cup \delta R\]
\[\top \cdot (\Phi \cap \Psi) = \top \cdot \Phi \cap \top \cdot \Psi\]

hold — similarly for ρ and (·⊤).
Other operators

\[(f \ X) \subseteq Y \equiv X \subseteq (g \ Y)\]

<table>
<thead>
<tr>
<th>Description</th>
<th>(f = g^\flat)</th>
<th>(g = f^\sharp)</th>
<th>Obs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>implication</td>
<td>((R \cap))</td>
<td>((R \Rightarrow))</td>
<td>(b(R \Rightarrow X)a \equiv bRa \Rightarrow bXa)</td>
</tr>
<tr>
<td>difference</td>
<td>((-R))</td>
<td>((R \cup))</td>
<td></td>
</tr>
</tbody>
</table>

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 $A \subseteq \mathcal{P}A$ 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 $\subseteq \setminus \subseteq$ of the previous exercise is reflexive and transitive. □

Exercise 104: Prove that equality

$$(R \setminus S) \cdot f = R \setminus (S \cdot f)$$

holds. □
Exercises

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

Exercise 106: The relational $overriding$ combinator,

$$R \uparrow S = S \cup R \cap \perp/S^\circ$$  \hspace{1cm} (230)

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

$$X \subseteq R \uparrow S \equiv X - S \subseteq R \land \delta (X - S) \cdot \delta S = \perp$$  \hspace{1cm} (231)
Binary adjoints

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

$$\bigcup(R, S) \subseteq X \equiv (R, S)(\subseteq \times \subseteq)(X, X)$$

or even as

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

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

$$\bigcup \vdash \Delta$$

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

$$\Delta \vdash \bigcap$$

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)

$Ff \vdash Fg$

**Splitting** (lattices)

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

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

$\Delta \vdash \sqcap$ \hspace{1cm} (232)

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

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

\[ \{p\} P \{q\} \]  
(233)

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

\[ \langle \forall s : p \ s : \langle \forall s' : s \overset{P}{\rightarrow} s' : q \ s' \rangle \rangle \]  
(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:

$$\langle \forall s : p \ s : \langle \forall s' : s'[[P]]s : q s' \rangle \rangle \equiv \begin{cases} \text{coreflexives} \end{cases} \langle \forall s : s\PhiPs : \langle \forall s' : s'[[P]]s : s'\Phi_q s' \rangle \rangle \equiv \begin{cases} \top; \text{coreflexives} \end{cases} \langle \forall s, s'' : s\PhiPs'' : \langle \forall s' : s'[[P]]s : s'(\Phi_q \cdot \top)s'' \rangle \rangle \equiv \begin{cases} \text{recall (225) and remove variables} \end{cases} \Phi_p \subseteq [[P]] \setminus (\Phi_q \cdot \top)$$
Handling Hoare triples in relation algebra

Finally:

\[
\Phi_p \subseteq \llbracket P \rrbracket \setminus (\Phi_q \cdot \top)
\]

\[
\equiv \{ \text{GC of division (228)} \}
\]

\[
\llbracket P \rrbracket \cdot \Phi_p \subseteq \Phi_q \cdot \top
\]

\[
\equiv \{ (118) \}
\]

\[
\llbracket P \rrbracket \cdot \Phi_p \subseteq \Phi_q \cdot \llbracket P \rrbracket
\]

Comparing this with the meaning of contract \( \Phi_q \leftarrow^f \Phi_p \) — recall (143) — we realize that they are the same in case \( \llbracket P \rrbracket \) 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

\[
[P] \cdot \Phi_p \subseteq \Phi_q \cdot [P] 
\]

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

\[
[P] \cdot \Phi_p \subseteq \Phi_q \cdot \top 
\]
Hoare triples are GCs

In turn, (236) is equivalent to

$$\Phi_p \subseteq \llbracket P \rrbracket \setminus (\Phi_q \cdot \top) \cap id$$

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

$$\rho(\llbracket P \rrbracket \cdot \Phi_p) \subseteq \Phi_q$$

Thus we have the following Galois connection for Hoare triples, where $P$, $\Phi$ and $\Psi$ abbreviate $\llbracket P \rrbracket$, $\Phi_p$ and $\Phi_q$, respectively:

$$\rho(P \cdot \Phi) \subseteq \Psi \equiv \Phi \subseteq P \setminus (\Psi \cdot \top) \cap id \quad (237)$$

Adjoint functions $f$ and $g$ are known as predicate transformers.
Hoare triples are GCs

The usual notation for \( g \psi \) is \( P \triangleleft \psi \) — the \textbf{weakest} (liberal) \textbf{pre-condition} (WP) for \( \psi \) to hold on the outputs of \( P \).

Dually, \( f \Phi = \rho (P \cdot \Phi) \) is known as the \textbf{strongest post-condition} (SP) holding on all outputs of \( P \) restricted by \( \Phi \) 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

\[
\Phi \xrightarrow{R} \Psi \equiv R \cdot \Phi \subseteq \Psi \cdot R \tag{238}
\]
\[
\equiv \Phi \subseteq R \triangleleft \Psi \tag{239}
\]

which extends \textbf{functional contracts} to arbitrary relations.
Exercises

Exercise 107: Prove

\[ \text{id} \leftarrow_R^R \Phi \equiv \text{TRUE} \equiv \Phi \leftarrow_R \bot \]  \hspace{1cm} (240)

Exercise 108: Prove the special cases:

- WP of a function \( f \):
  \[ f \Downarrow \Phi_q = \lambda a. q(f a) \]  \hspace{1cm} (241)

- SP of a function \( f \):
  \[ \rho(f \cdot \Phi_p) = \lambda b. b \in \{ f a | p a \} \]  \hspace{1cm} (242)

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

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

\[ [P; Q] = [Q] \cdot [P] \]

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

\[
\{p\} P \{q\} , \{q\} Q \{s\} \\
\{p\} P; Q \{s\}
\]

is an instance of the following relational typing rule:

\[
\Psi \xleftarrow{R \cdot S} \Phi \iff \Psi \xleftarrow{R} \Upsilon \land \Upsilon \xleftarrow{S} \Phi \quad (243)
\]
Exercises

Exercise 110: Prove the “trading rule”:

\[ \gamma \leftarrow^R \phi \cdot \psi \equiv \gamma \leftarrow^{R \cdot \phi} \psi \]  

(244)

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

\[ \psi_1 \cdot \psi_2 \leftarrow^R \phi \equiv \psi_1 \leftarrow^R \phi \land \psi_2 \leftarrow^R \phi \]  

(245)

in Hoare logic. Then prove (245). □
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:

\[
\Upsilon \cdot \Psi \leftarrow^R \Phi \equiv \Upsilon \leftarrow^R \Phi \land \Psi \leftarrow^R \Phi
\]

\[
\equiv \{ \text{WPs (239) three times} \}
\]

\[
\Phi \subseteq R \setminus (\Upsilon \cdot \Psi) \equiv \Phi \subseteq R \setminus \Upsilon \land \Phi \subseteq R \setminus \Psi
\]

\[
\equiv \{ \text{coreflexives (112) ; meet-universal (64)} \}
\]

\[
\langle \forall \Phi :: \Phi \subseteq R \setminus (\Upsilon \cdot \Psi) \equiv \Phi \subseteq (R \setminus \Upsilon) \cap (R \setminus \Psi) \rangle
\]

\[
\equiv \{ \text{meet of correflexives; indirect equality (69)} \}
\]

\[
R \setminus (\Upsilon \cdot \Psi) = (R \setminus \Upsilon) \cdot (R \setminus \Psi)
\]
WP calculus

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

\[(S \cdot R) \bullet \Phi = R \bullet (S \bullet \Phi)\]  \hspace{1cm} (246)

into a contract:

\[R \bullet (S \bullet \phi) = (S \cdot R) \bullet \phi\]

\[\equiv \quad \{ \text{indirect equality (69)} \} \]

\[\psi \subseteq R \bullet (S \bullet \phi) \equiv \psi \subseteq (S \cdot R) \bullet \phi\]

\[\equiv \quad \{ (239) \text{ twice} \} \]

\[\phi \leftarrow R \psi \quad \equiv \quad \phi \leftarrow (S \cdot R) \psi\] \hspace{1cm} (247)

The outcome, still involving the $\bullet$ operator, is an advantageous replacement for (243), since it is an equivalence.
Exercises

Exercise 112: Show that \( \rho R \leq^R \delta R \) holds. However, WP \( R \downarrow (\rho R) = id \) rather than \( \delta R \). Explain why. □

Exercise 113: Show that \( \rho R \leq^R \delta R \) holds. However, WP \( R \downarrow (\rho R) = id \) rather than \( \delta R \). Explain why. □

Exercise 114: The two “shunting” rules for \( S \) a simple relation,

\[
S \cdot R \subseteq Q \equiv (\delta S) \cdot R \subseteq S^\circ \cdot Q \tag{248}
\]
\[
R \cdot S^\circ \subseteq Q \equiv R \cdot \delta S \subseteq Q \cdot S \tag{249}
\]

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

\[
R \cdot \Phi \subseteq S \equiv R \cdot \Phi \subseteq S \cdot \Phi
\]
\[
\Phi \cdot R \subseteq S \equiv \Phi \cdot R \subseteq \Phi \cdot S
\]

referred to earlier on as the closure properties (113) and (114), respectively; (b) prove either (248) or (249) by cyclic implication (vulg. “ping-pong”). □
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. \( \mathbb{N}_0 \leq \mathbb{N}_0 \).

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 \textit{take} function available in Haskell, for instance:

\[
\text{length } y_s \leq n \land y_s \preceq x_s \equiv y_s \preceq \text{take } n x_s \tag{250}
\]

Here the ordering on outputs is the \textbf{prefix} relation (\preceq) on lists.

For each \( n \), term \( \text{length } y_s \leq n \land y_s \preceq x_s \) tells which outputs \( y_s \) are candidates for \( \text{take } n x_s \).

But only one of these is acceptable — the \textbf{longest} such prefix, which is \textbf{optimal} with respect to the prefix ordering.
Exercise

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

- Show that

\[
\text{take} \left( \text{length} \; \text{xs} \right) \; \text{xs} = \text{xs}
\]

holds.

- Resort to indirect equality over \(\preceq\) in proving

\[
\text{take} \; n \; \left( \text{take} \; m \; \text{xs} \right) = \text{take} \; \left( \text{min} \; n \; m \right) \; \text{xs}
\]

where \textit{min}, the minimum of two natural numbers, is given by the obvious Galois connection.

\[\Box\]
Optimization in an abstract setting

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

\[
\begin{align*}
z\left(\div y\right)x & \equiv z \times y \leq x \land \langle \forall z' : z' \times y \leq x : z \geq z' \rangle \\
\equiv & \left\{ \text{define } z R x = z \times y \leq x \right\} \\
\left( z \times y \leq x \land \langle \forall z' : z' \times y \leq x : z \geq z' \rangle \right) & \equiv z \left(\div y\right)x
\end{align*}
\]

In summary:

\[
\left(\div y\right) = R \cap \geq / R^o \text{ where } R = \left(\times y\right)^o \cdot \leq
\]

(251)
Optimization in an abstract setting

Generalization: given any relation $B \leftarrow^R A$ and an optimization criterion $B \leftarrow^S B$ on its outputs, define a new relational combinator $R \uparrow S$ (read: $R$ optimized by $S$, or $R$ “shrunk” by $S$) as follows:

$$R \uparrow S = \underbrace{R \cap S / R^\circ}_{\text{easy}} \underbrace{\null}_{\text{hard}}$$

(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:

\[ X \subseteq R \upharpoonright S \iff X \subseteq R \land X \cdot R^\circ \subseteq S \]  

(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 \( b'Xa \land bRa \), then \( b'Sb \) 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 applications to software design.
Basic properties of $R \upharpoonright S$

Chaotic optimization:

$$R \upharpoonright \top = R$$  \hspace{1cm} (254)

Impossible optimization:

$$R \upharpoonright \bot = \bot$$  \hspace{1cm} (255)

“Brute force” determinization:

$$R \upharpoonright id = \text{largest deterministic fragment of } R$$  \hspace{1cm} (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:

$$R \upharpoonright S \text{ is simple} \iff S \text{ is anti-symmetric} \quad (257)$$

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

$$R \upharpoonright S = R \equiv \text{img } R \subseteq S \quad (258)$$

Thus (functions)

$$f \upharpoonright S = f \iff S \text{ is reflexive} \quad (259)$$
Basic properties of $R \upharpoonright S$

Pre-condition fusion:

\[(R \upharpoonright S) \cdot \Phi = (R \cdot \Phi) \upharpoonright S\]  \hfill (260)

Two function fusion rules

\[(R \upharpoonright S) \cdot f = (R \cdot f) \upharpoonright S\]  \hfill (261)
\[(f \cdot R) \upharpoonright S = f \cdot (R \upharpoonright S_f)\]  \hfill (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.
\[\square\]
Basic properties of \( R \uparrow S \)

Union:

\[
(R \cup S) \uparrow Q = (R \uparrow Q) \cap Q/S^\circ \cup (S \uparrow Q) \cap Q/R^\circ
\] (263)

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

\[
(p \rightarrow R, T) \uparrow S = p \rightarrow (R \uparrow S), (p \uparrow S)
\] (264)

the distribution over alternatives (77),

\[
[R, S] \uparrow U = [R \uparrow U, S \uparrow U]
\] (265)

and the “function competition” rule:

\[
(f \cup g) \uparrow S = (f \cap S \cdot g) \cup (g \cap S \cdot f)
\] (266)

since \( S/g^\circ = S \cdot g \).
“Function competition” rule

With points:

\[ y((f \cup g) \upharpoonright S)x \equiv \begin{cases} 
  y = f \ x \land (f \ x)S(g \ x) \\
  \lor \\
  y = g \ x \land (g \ x)S(f \ x) 
\end{cases} \]

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.

\[ y = abs \ x \equiv \begin{cases} 
  y = x \land x \geq -x \land y = -x \land -x \geq x \\
  \lor \\
  y = x \land x \geq 0 \land y = -x \land 0 \geq x 
\end{cases} \]
**R | S on data**

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

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

\[
\begin{array}{|c|c|c|}
\hline
\text{Examiner} & \text{Mark} & \text{Student} \\
\hline
\text{Smith} & 10 & \text{John} \\
\text{Smith} & 11 & \text{Mary} \\
\text{Smith} & 15 & \text{Arthur} \\
\text{Wood} & 12 & \text{John} \\
\text{Wood} & 11 & \text{Mary} \\
\text{Wood} & 15 & \text{Arthur} \\
\hline
\end{array}
\]

and wishing to “choose the best mark”, project over Mark, Student and optimize over the \( \geq \) ordering on Mark (next slide):
Relational shrinking can also be used for induction-free reasoning about sequences (lists), welcome in Alloy where no explicit recursion is available.

Example of $R \upharpoonright S$ in list-processing: given a sequence $A \leftarrow S \leftarrow \IN$, 

$$A \leftarrow \operatorname{nub} S \leftarrow \IN \triangleq (S^\circ \upharpoonright \leq)^\circ$$

removes all duplicates while keeping the first instances. (Data in $\IN$ could be regarded as “time stamps”.)
Galois connections (211) as optimization problems

\[ f^\circ \cdot (\leq) = (\square) \cdot g \]

\[ \equiv \quad \{ \text{ping-pong} \} \]

\[ (\square) \cdot g \subseteq f^\circ \cdot (\leq) \land f^\circ \cdot (\leq) \subseteq (\square) \cdot g \]

\[ \equiv \quad \{ \text{converses} \} \]

\[ (\square) \cdot g \subseteq f^\circ \cdot (\leq) \land (f^\circ \cdot (\leq))^\circ \subseteq g^\circ \cdot (\sqcup) \]

\[ \equiv \quad \{ \text{since } f \text{ is monotonic (see exercise 119 below)} \} \]

\[ g \subseteq f^\circ \cdot (\leq) \land g \cdot (f^\circ \cdot (\leq))^\circ \subseteq (\sqcup), \]

“easy” \quad “hard”

\[ \equiv \quad \{ \text{universal property (253)} \} \]

\[ g \subseteq (f^\circ \cdot (\leq)) \restriction (\square) \quad (267) \]
Galois connections as optimization problems

Comments:

• Given the two orderings ($\leq$) and ($\sqsubseteq$) 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 $g \subseteq (f^\circ \cdot (\leq)) \uparrow (\sqsubseteq)$ expresses an optimization by ($\sqsubseteq$), 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

\[(\sqsubseteq) \cdot f^\circ \subseteq f^\circ \cdot (\leq),\] (268)

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

\[g \subseteq f^\circ \cdot (\leq) \equiv (\sqsubseteq) \cdot g \subseteq f^\circ \cdot (\leq)\] (269)

Suggestion: do a “ping-pong” proof. \[\square\]
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):

\[ R \uparrow S \] algebra greatly simplifies and generalizes the calculation of programs from such specifications. (Notably, there is no need for power transpose.)
Folds ($k\alpha\tau\alpha$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:

$$X = (|R|) \equiv X \cdot in = R \cdot F X \quad (270)$$

(Read $(|R|)$ as “fold $R$” or “κατα $R$”.)
Relational folds

It is very easy to show that

\[(\text{in}) = id\]  \hfill (271)

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

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

\[f[\ ] = [\ ] \]
\[f(\text{cons}(a, x)) = \text{cons}(a, f x)\]

Now suppose we have \(R = [\text{nil}, \text{cons} \cup \text{nil}]\) in (270). What is the meaning of \([\text{nil}, \text{cons} \cup \text{nil}]\)?
Unfolding $X = ([\text{nil}, \text{cons} \cup \text{nil}])$ we get

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

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

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

$$y \ X \ (a : x) \equiv y = [\ ] \lor (\exists \ x' : \ x' \ X \ x : \ y = a : x')$$

Thus $([\text{nil}, \text{cons} \cup \text{nil}])$ is the prefix relation:

$$(\preceq) = ([\text{nil}, \text{cons} \cup \text{nil}])$$
The “Greedy” theorem

\[(|R| \uparrow S|) \subseteq (|R|) \uparrow S \iff S^\circ \leftarrow^R F S^\circ\]  \hspace{1cm} (272)

for $S$ transitive. (\textbf{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 \textit{msp} problem ("maximum sum prefix"), whose spec

\[
\begin{align*}
\text{msp} &:: \text{[Int]} \leftarrow \text{[Int]} \\
y \text{ msp } x &\equiv y \text{ is a prefix of } x \text{ that yields the maximum sum}
\end{align*}
\]

translates into (\(\leq = ([\text{nil}, \text{cons} \cup \text{nil}])\) is the prefix ordering)

\[
y \text{ msp } x \quad \Rightarrow \quad y \leq x \land \langle \forall z : z \leq x : \text{sum } y \geq \text{sum } z \rangle
\]

which in turn PF-transforms into

\[
\text{msp} \quad \subseteq \quad \leq \uparrow \geq_{\text{sum}}
\]

(\textbf{NB:} not a GC, it is nevertheless a good example to understand greedy programming.)
Example of greedy programming

We calculate:

\[ msp \subseteq \leq \uparrow \geq_{\text{sum}} \]

\[ \equiv \{ \text{definition of prefix ordering} \} \]

\[ msp \subseteq ([\text{nil}, \text{cons} \cup \text{nil}]) \uparrow \geq_{\text{sum}} \]

\[ \Leftarrow \{ \text{greedy theorem (272)} \} \]

\[ msp \subseteq ([\text{nil}, \text{cons} \cup \text{nil}] \uparrow \geq_{\text{sum}}) \]

\[ \equiv \{ \text{junc-rule (265) ; determinism of nil} \} \]

\[ msp \subseteq ([\text{nil}, (\text{cons} \cup \text{nil}) \uparrow \geq_{\text{sum}}]) \]

\[ \equiv \{ \text{function competition rule (266)} \} \]

\[ msp \subseteq ([\text{nil}, (\text{cons} \cap \geq_{\text{sum}} \cdot \text{nil}) \cup (\text{nil} \cap \geq_{\text{sum}} \cdot \text{cons})]) \]

(Side condition ignored for brevity.)
Example of greedy programming

Let $R$ abbreviate the inductive step

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

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

$$y = [\ ] \land 0 \geq a + \text{sum} \ x \lor y = a : x \land a + \text{sum} \ x \geq 0$$

The case $a + \text{sum} \ x = 0$ 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,

$$msp = (\preceq \uparrow \geq_{\text{sum}}) \uparrow \preceq$$

(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:

\[
\begin{align*}
\text{msp} \; [] &= [] \\
\text{msp}(a:s) &= \text{let } x = \text{msp} \; s \\
&\quad \text{in if } \text{sum } x > -a \text{ then } a:x \text{ else } []
\end{align*}
\]

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 \((x \div y)\), \textit{take} etc end up being anamorphisms.
