PF transform: where everything becomes a relation

J.N. Oliveira

Dept. Informática,
Universidade do Minho
Braga, Portugal

DI/UM, 2007 (updated 2009)

Pairs

Consider assertions

$$0 \quad \leq \quad \pi$$
$$\texttt{John} \quad \mathit{IsFatherOf} \quad \texttt{Mary}$$
$$3 \quad = (1+) \quad 2$$

- They are statements of fact concerning various kinds of object — real numbers, people, natural numbers, etc
- They involve *two* such objects, that is, **pairs**

$$(0, \pi)$$
$$(\texttt{John}, \texttt{Mary})$$
$$(3, 2)$$

respectively.

Sets of pairs

So, we might have written

$$\begin{array}{rcl} (0, \pi) & \in & \leq \\ (\texttt{John}, \texttt{Mary}) & \in & \mathit{IsFatherOf} \\ (3, 2) & \in & (1+) \end{array}$$

What are $(\leq)$, *IsFatherOf*, $(1+)$?

- they are **sets of pairs**
- they are **binary relations**

Therefore,

- partial **orders** — eg. $(\leq)$ — are special cases of relations
- **functions** — eg. $\mathit{succ} \triangleq (1+)$ — are special cases of relations

Binary Relations

Binary relations are typed:

Arrow notation

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

A, B are types. Writing $B \xleftarrow{R} A$ means the same as $A \xrightarrow{R} B$.

Infix notation

The usual infix notation used in natural language — eg. `John` *IsFatherOf* `Mary` — and in maths — eg. $0 \leq \pi$ — extends to arbitrary $B \xleftarrow{R} A$: we write

$$b \; R \; a$$

to denote that $(b, a) \in R$.

Binary Relations

Binary relations are typed:

Arrow notation

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

A, B are types. Writing $B \xleftarrow{R} A$ means the same as $A \xrightarrow{R} B$.

Infix notation

The usual infix notation used in natural language — eg. $\texttt{John}\ \mathit{IsFatherOf}\ \texttt{Mary}$ — and in maths — eg. $0 \leq \pi$ — extends to arbitrary $B \xleftarrow{R} A$: we write

$$b\ R\ a$$

to denote that $(b, a) \in R$.

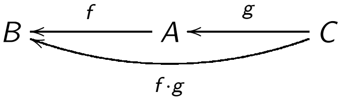
Functions are relations

- Lowercase letters (or identifiers starting by one such letter) will denote special relations known as **functions**, eg. f, g, $succ$, etc.
- We regard function $f : A \longrightarrow B$ as the binary relation which relates b to a iff $b = f\ a$. So, $b\ f\ a$ literally means $b = f\ a$.
- Therefore, we generalize

$$\boxed{\begin{array}{c} B \xleftarrow{\ f\ } A \\ b = f\ a \end{array}} \quad \text{to} \quad \boxed{\begin{array}{c} B \xleftarrow{\ R\ } A \\ b\ R\ a \end{array}}$$

Composition

Recall **function composition**



(1)

$$b = f(g\ c)$$

and extend $f \cdot g$ to $R \cdot S$ in the obvious way:

$$b(R \cdot S)c \quad \equiv \quad \langle \exists\ a\ ::\ b\ R\ a \wedge a\ S\ c \rangle \tag{2}$$

Note how this rule of the PF-transform *removes* $\exists$ when applied from right to left

Check generalization

Back to functions, (2) becomes

$$
\begin{array}{ll}
b(f \cdot g)c & \equiv \quad \langle \exists\, a \;::\; b\ f\ a \wedge a\ g\ c \rangle \\
& \equiv \qquad \{\ a\ g\ c \text{ means } a = g\ c\ \} \\
& \quad\;\; \langle \exists\, a \;::\; b\ f\ a \wedge a = g\ c \rangle \\
& \equiv \qquad \{\ \exists\text{-trading} \;;\; b\ f\ a \text{ means } b = f\ a\ \} \\
& \quad\;\; \langle \exists\, a \;:\; a = g\ c \;:\; b = f\ a \rangle \\
& \equiv \qquad \{\ \text{one-point rule } (\exists)\ \} \\
& \quad\;\; b = f(g\ c)
\end{array}
$$

So, we easily recover what we had before (1).

Inclusion generalizes equality

- **Equality** on functions

$$f = g \quad \equiv \quad \langle \forall\ a\ :\ a \in A :\ f\ a\ =_B\ g\ a \rangle$$

generalizes to **inclusion** on relations:

$$R \subseteq S \quad \equiv \quad \langle \forall\ b, a\ :\ b\ R\ a :\ b\ S\ a \rangle \tag{3}$$

(read $R \subseteq S$ as "R is at most S")

- For $R \subseteq S$ to hold both need to be of the same type, say $B \xleftarrow{R,S} A$
- $R \subseteq S$ is a partial order (reflexive, transitive and anti-symmetric)

Special relations

Every type $B \longleftarrow A$ has its

- *bottom* relation $B \xleftarrow{\perp} A$, which is such that, for all b, a, $b \perp a \equiv \text{FALSE}$
- *topmost* relation $B \xleftarrow{\top} A$, which is such that, for all b, a, $b \top a \equiv \text{TRUE}$

Type $A \longleftarrow A$ has the

- identity relation $A \xleftarrow{id} A$ which is function $id\ a \triangleq a$.

Clearly, for every R,

$$\perp \subseteq R \subseteq \top \tag{4}$$

Exercises

Exercise 1: Resort to PF-transform rule (2) and to the Eindhoven quantifier calculus to show that

$$R \cdot id = R = id \cdot R \tag{5}$$

$$R \cdot \bot = \bot = \bot \cdot R \tag{6}$$

hold and that composition is associative:

$$R \cdot (S \cdot T) = (R \cdot S) \cdot T \tag{7}$$

□

Converses

Every relation $B \xleftarrow{R} A$ has a **converse** $B \xrightarrow{R^\circ} A$ which is such that, for all a, b,

$$a(R^\circ)b \quad \equiv \quad b\ R\ a \tag{8}$$

Note that converse commutes with composition

$$(R \cdot S)^\circ = S^\circ \cdot R^\circ \tag{9}$$

and with itself:

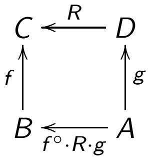
$$(R^\circ)^\circ = R \tag{10}$$

Function converses

Function converses f°, g° etc. always exist (as **relations**) and enjoy the following (very useful) PF-transform property:

$$(f\ b)R(g\ a) \quad \equiv \quad b(f^\circ \cdot R \cdot g)a \tag{11}$$

cf. diagram:



Let us see an example of its use.

PF-transform at work

Transforming a well-known PW-formula:

$$
\begin{array}{ll}
 & f \text{ is injective} \\
\equiv & \quad \{ \text{ recall definition from discrete maths } \} \\
 & \langle \forall\, y, x \;:\; (f\ y) = (f\ x) :\; y = x \rangle \\
\equiv & \quad \{ \text{ introduce } id \text{ (twice) } \} \\
 & \langle \forall\, y, x \;:\; (f\ y) id (f\ x) :\; y(id)x \rangle \\
\equiv & \quad \{ \text{ rule } (f\ b)R(g\ a) \equiv b(f^{\circ} \cdot R \cdot g)a \text{ (11) } \} \\
 & \langle \forall\, y, x \;:\; y(f^{\circ} \cdot id \cdot f)x :\; y(id)x \rangle \\
\equiv & \quad \{ \text{ (5) ; then go pointfree via (3) } \} \\
 & f^{\circ} \cdot f \subseteq id
\end{array}
$$

The other way round

Let us now see what $id \subseteq f \cdot f^\circ$ means:

$$id \subseteq f \cdot f^\circ$$

$\equiv$ { relational inclusion (3) }

$$\langle \forall\ y, x\ :\ y(id)x :\ y(f \cdot f^\circ)x \rangle$$

$\equiv$ { identity relation ; composition (2) }

$$\langle \forall\ y, x\ :\ y = x :\ \langle \exists\ z\ ::\ y\ f\ z \wedge z\ f^\circ x \rangle \rangle$$

$\equiv$ { converse (8) }

$$\langle \forall\ y, x\ :\ y = x :\ \langle \exists\ z\ ::\ y\ f\ z \wedge x\ f\ z \rangle \rangle$$

$\equiv$ { $\forall$-one point ; trivia ; function f }

$$\langle \forall\ x\ ::\ \langle \exists\ z\ ::\ x = f\ z \rangle \rangle$$

$\equiv$ { recalling definition from maths }

f is surjective

Why *id* (really) matters

Terminology:

- Say R *is reflexive* iff $id \subseteq R$
  pointwise: $\langle \forall\ a\ ::\ a\ R\ a \rangle$ (check as homework);
- Say R *is coreflexive* iff $R \subseteq id$
  pointwise: $\langle \forall\ a\ :\ b\ R\ a :\ b = a \rangle$ (check as homework).

Define, for $B \xleftarrow{R} A$:

Kernel of R	**Image** of R
$A \xleftarrow{\ker R} A$	$B \xleftarrow{\mathrm{img}\ R} B$
$\ker R \stackrel{\mathrm{def}}{=} R^\circ \cdot R$	$\mathrm{img}\ R \stackrel{\mathrm{def}}{=} R \cdot R^\circ$

Example: kernels of functions

$$
\begin{array}{ll}
& a'(\ker f)a \\
\equiv & \{ \text{ substitution } \} \\
& a'(f^\circ \cdot f)a \\
\equiv & \{ \text{ PF-transform rule (11) } \} \\
& (f\ a') = (f\ a)
\end{array}
$$

In words: $a'(\ker f)a$ means a' *and* a *"have the same* f*-image"*

Exercise 2: Let C be a nonempty data domain and let and $c \in C$. Let $\underline{c}$ be the *"everywhere* c*"* function:

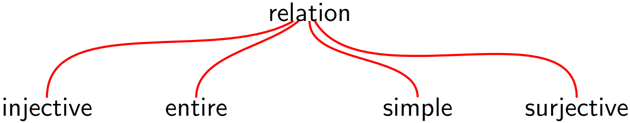
$$
\begin{array}{rcl}
\underline{c} & : & A \longrightarrow C \\
\underline{c}\, a & \triangleq & c
\end{array}
\qquad (12)
$$

Compute which relations are defined by the following PF-expressions:

$$
\ker \underline{c} \quad , \quad \underline{b} \cdot \underline{c}^\circ \quad , \quad \mathrm{img}\, \underline{c} \qquad (13)
$$

Binary relation taxonomy

Topmost criteria:



Definitions:

	Reflexive	*Coreflexive*
$\ker R$	entire R	injective R
$\operatorname{img} R$	surjective R	simple R

(14)

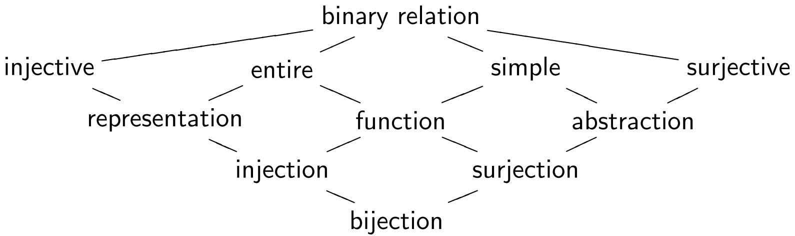
Facts:

$$\ker(R^\circ) = \operatorname{img} R \tag{15}$$

$$\operatorname{img}(R^\circ) = \ker R \tag{16}$$

Binary relation taxonomy

The whole picture:



(17)

Exercise 3: Resort to (15,16) and (14) to prove the following rules of thumb:

- converse of **injective** is **simple** (and vice-versa)
- converse of **entire** is **surjective** (and vice-versa)

□

Functions in one slide

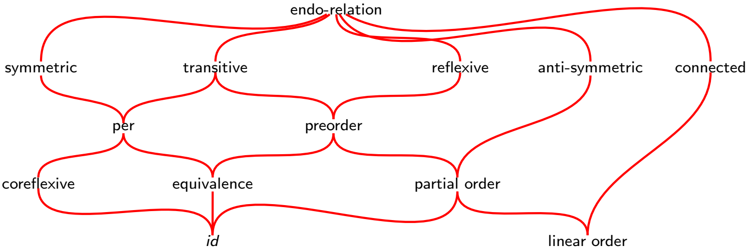
A function f is a binary relation such that

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

NB: Following a widespread convention, functions will be denoted by lowercase characters (eg. f, g, ϕ) or identifiers starting with lowercase characters, and function application will be denoted by juxtaposition, eg. $f\ a$ instead of $f(a)$.

Relation taxonomy — orders

Orders are endo-relations $A \xleftarrow{R} A$ classified as



(Criteria definitions: next slide)

Orders and their taxonomy

Besides

reflexive:	iff $id_A \subseteq R$
coreflexive:	iff $R \subseteq id_A$

an order (or endo-relation) $A \xleftarrow{R} A$ can be

transitive:	iff $R \cdot R \subseteq R$
anti-symmetric:	iff $R \cap R^\circ \subseteq id_A$
symmetric:	iff $R \subseteq R^\circ (\equiv R = R^\circ)$
connected:	iff $R \cup R^\circ = \top$

Orders and their taxonomy

Therefore:

- **Preorders** are reflexive and transitive orders.
  Example: $y\ \mathit{IsAtMostAsOldAs}\ x$
- **Partial** orders are anti-symmetric preorders
  Example: $y \subseteq x$
- **Linear** orders are connected partial orders
  Example: $y \leq x$
- **Equivalences** are symmetric preorders
  Example: $y\ \mathit{Permutes}\ x$
- **Pers** are partial equivalences
  Example: $y\ \mathit{IsBrotherOf}\ x$

Exercises

Exercise 4: Expand all criteria in the previous slides to pointwise notation.

□

Exercise 5: A relation R is said to be *co-transitive* iff the following holds:

$$\langle \forall\ b, a\ :\ b\ R\ a :\ \langle \exists\ c\ :\ b\ R\ c :\ c\ R\ a \rangle \rangle \quad (18)$$

Compute the PF-transform of the formula above. Find a relation (eg. over numbers) which is co-transitive and another which is not.

□

Meet and join

Meet (intersection) and join (union) internalize conjunction and disjunction, respectively,

$$b\ (R \cap S)\ a \quad \equiv \quad b\ R\ a \wedge b\ S\ a \tag{19}$$

$$b\ (R \cup S)\ a \quad \equiv \quad b\ R\ a \vee b\ S\ a \tag{20}$$

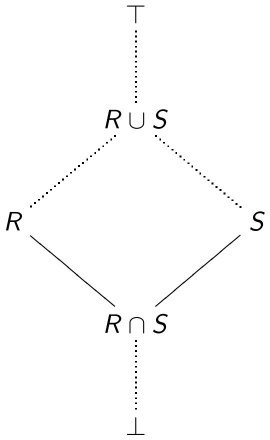
for R, S of the same type. Their meaning is captured by the following **universal** properties:

$$X \subseteq R \cap S \quad \equiv \quad X \subseteq R \ \wedge\ X \subseteq S \tag{21}$$

$$R \cup S \subseteq X \quad \equiv \quad R \subseteq X \ \wedge\ S \subseteq X \tag{22}$$

In summary

Type $B \longleftarrow A$ forms a lattice:



"top"

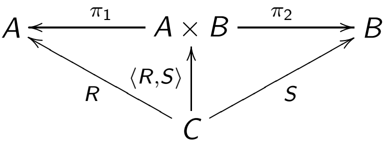
join, lub ("least upper bound")

meet, glb ("greatest lower bound")

"bottom"

All (data structures) **in one** (PF notation)

Products



(23)

where

ψ	$PF\ \psi$
$a\ R\ c \wedge b\ S\ c$	$(a,b)\langle R,S\rangle c$
$b\ R\ a \wedge d\ S\ c$	$(b,d)(R \times S)(a,c)$

(24)

Clearly: $R \times S = \langle R \cdot \pi_1, S \cdot \pi_2 \rangle$

Sums

Example (Haskell):

```
data X = Boo Bool | Err String
```

PF-transforms to

(25)

where

$$[R\ ,S] = (R \cdot i_1^\circ) \cup (S \cdot i_2^\circ)$$ cf.

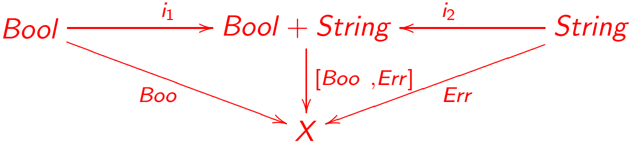
Dually: $R + S = [i_1 \cdot R\ , i_2 \cdot S]$

Sums

Example (Haskell):

```
data X = Boo Bool | Err String
```

PF-transforms to

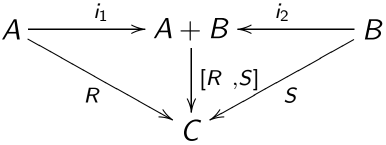


(25)

where

$$[R\ , S] \;=\; (R \cdot i_1^\circ) \cup (S \cdot i_2^\circ)$$

cf.



Dually: $R + S = [i_1 \cdot R\ , i_2 \cdot S]$

Last but not least: relational equality

- **Pointwise** equality:

$$R = S \quad \equiv \quad \langle \forall\, b, a \ :: \ b\ R\ a \equiv b\ S\ a \rangle$$

- **Pointfree** equality:
  - **Cyclic inclusion** ("ping-pong") rule:

$$R = S \quad \equiv \quad R \subseteq S \wedge S \subseteq R \tag{26}$$

  - **Indirect equality** rules [1]:

$$R = S \quad \equiv \quad \langle \forall\, X \ :: \ (X \subseteq R \equiv X \subseteq S) \rangle \tag{27}$$

$$\equiv \quad \langle \forall\, X \ :: \ (R \subseteq X \equiv S \subseteq X) \rangle \tag{28}$$

[1]Cf. [1], p. 82.

Example of indirect proof

$$
\begin{array}{ll}
 & X \subseteq (R \cap S) \cap T \\
\equiv & \qquad \{\ \cap\text{-universal (21)}\ \} \\
 & X \subseteq (R \cap S) \wedge X \subseteq T \\
\equiv & \qquad \{\ \cap\text{-universal (21)}\ \} \\
 & (X \subseteq R \wedge X \subseteq S) \wedge X \subseteq T \\
\equiv & \qquad \{\ \wedge \text{ is associative}\ \} \\
 & X \subseteq R \wedge (X \subseteq S \wedge X \subseteq T) \\
\equiv & \qquad \{\ \cap\text{-universal (21) twice}\ \} \\
 & X \subseteq R \cap (S \cap T) \\
:: & \qquad \{\ \text{indirection}\ \} \\
 & (R \cap S) \cap T = R \cap (S \cap T)
\end{array} \tag{29}
$$

Last but not least: monotonicity

All relational combinators seen so far are $\subseteq$-monotonic, for instance:

$$
\begin{array}{rcl}
R \subseteq S & \Rightarrow & R^\circ \subseteq S^\circ \\
R \subseteq S \wedge U \subseteq V & \Rightarrow & R \cdot U \subseteq S \cdot V \\
R \subseteq S \wedge U \subseteq V & \Rightarrow & R \cap U \subseteq S \cap V \\
R \subseteq S \wedge U \subseteq V & \Rightarrow & R \cup U \subseteq S \cup V
\end{array}
$$

etc

Exercise 6: Prove the following rules of thumb:

- smaller than injective (simple) is injective (simple)
- larger than entire (surjective) is entire (surjective)

□

Exercises

Exercise 7: Show that (11) holds.

□

Exercise 8: Check which of the following hold:

- If relations R and S are simple, then so is $R \cap S$
- If relations R and S are injective, then so is $R \cup S$
- If relations R and S are entire, then so is $R \cap S$

□

Exercise 9: Prove that relational composition preserves *all* relational classes in the taxonomy of (17).

□

Exercises

Exercise 10: Prove the following fact

A function f is a bijection iff its converse f° is a function (30)

by completing:

$$
\begin{array}{cl}
 & f \text{ and } f^\circ \text{ are functions} \\
\equiv & \quad \{ \ \dots \ \} \\
 & (id \subseteq \ker f \wedge \operatorname{img} f \subseteq id) \wedge (id \subseteq \ker f^\circ \wedge \operatorname{img} f^\circ \subseteq id) \\
\equiv & \quad \{ \ \dots \ \} \\
 & \vdots \\
\equiv & \quad \{ \ \dots \ \} \\
 & f \text{ is a bijection}
\end{array}
$$

□

Exercises

Exercise 11: Prove that $swap \triangleq \langle \pi_2, \pi_1 \rangle$ is a bijection.

□

Exercise 12: Let $\leq$ be a preorder and f be a function taking values on the carrier set of $\leq$.

1. Define the pointwise version of relation $\sqsubseteq \;\triangleq\; f^\circ \cdot \leq \cdot f$
2. Show that $\sqsubseteq$ is a preorder.
3. Show that $\sqsubseteq$ is not (in general) a total order even in the case $\leq$ is so.

□

Summary

Rules of the PF-transform seen so far:

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

Background — Eindhoven quantifier calculus

When writing $\forall, \exists$-quantified expressions is useful to know a number of rules which help in reasoning about them. Below we list some of these rules:

- **Trading:**

$$\langle \forall\ i\ :\ R \wedge S :\ T \rangle \ = \ \langle \forall\ i\ :\ R :\ S \Rightarrow T \rangle \tag{31}$$

$$\langle \exists\ i\ :\ R \wedge S :\ T \rangle \ = \ \langle \exists\ i\ :\ R :\ S \wedge T \rangle \tag{32}$$

Background — Eindhoven quantifier calculus

Splitting:

$$\langle \forall\, j \;:\; R :\; \langle \forall\, k \;:\; S :\; T\rangle\rangle \;=\; \langle \forall\, k \;:\; \langle \exists\, j \;:\; R :\; S\rangle :\; T\rangle \quad (33)$$

$$\langle \exists\, j \;:\; R :\; \langle \exists\, k \;:\; S :\; T\rangle\rangle \;=\; \langle \exists\, k \;:\; \langle \exists\, j \;:\; R :\; S\rangle :\; T\rangle \quad (34)$$

One-point:

$$\langle \forall\, k \;:\; k = e :\; T\rangle \;=\; T[k := e] \quad (35)$$

$$\langle \exists\, k \;:\; k = e :\; T\rangle \;=\; T[k := e] \quad (36)$$

Nesting:

$$\langle \forall\, a, b \;:\; R \wedge S :\; T\rangle \;=\; \langle \forall\, a \;:\; R :\; \langle \forall\, b \;:\; S :\; T\rangle\rangle \quad (37)$$

$$\langle \exists\, a, b \;:\; R \wedge S :\; T\rangle \;=\; \langle \exists\, a \;:\; R :\; \langle \exists\, b \;:\; S :\; T\rangle\rangle \quad (38)$$

R. Bird and O. de Moor.
Algebra of Programming.
Series in Computer Science. Prentice-Hall International, 1997.
C.A.R. Hoare, series editor.