

PF transform: where everything becomes a relation

J.N. Oliveira

Dept. Informática,
Universidade do Minho
Braga, Portugal

DI/UM, 2007

Pairs

Consider assertions

$$\begin{array}{ccc} 0 & \leq & \pi \\ \text{John} & \text{IsFatherOf} & \text{Mary} \\ 3 & = (1+) & 2 \end{array}$$

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

$$\begin{array}{c} (0, \pi) \\ (\text{John}, \text{Mary}) \\ (3, 2) \end{array}$$

respectively.

Sets of pairs

So, we might have written

$$\begin{aligned}(0, \pi) &\in \leq \\ (\text{John}, \text{Mary}) &\in \textit{IsFatherOf} \\ (3, 2) &\in (1+)\end{aligned}$$

What are (\leq) , $\textit{IsFatherOf}$, $(1+)$?

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

Therefore,

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

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 $(b, a) \in R$.

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

$$\begin{array}{c} B \xleftarrow{f} A \\ b = f a \end{array}$$

to

$$\begin{array}{c} B \xleftarrow{R} A \\ b R a \end{array}$$

Composition

Recall **function composition**

$$\begin{array}{ccccc}
 B & \xleftarrow{f} & A & \xleftarrow{g} & C \\
 & \searrow \quad \swarrow & & & \\
 & f \cdot g & & &
 \end{array}
 \quad (1)$$

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

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

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

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

Check generalization

Back to functions, (2) becomes

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

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

Inclusion generalizes equality

- **Equality** on functions

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

generalizes to **inclusion** on relations:

$$R \subseteq S \equiv \langle \forall b, a :: b\ R\ a \Rightarrow b\ S\ a \rangle \quad (3)$$

(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)

Exercises

Exercise 1: We want to compare

$$\begin{array}{l} \textit{IsPrefixOf} : (s : A^*) \rightarrow (r : A^*) \\ \textbf{post} \quad \textit{length } r \leq \textit{length } s \wedge \langle \forall i : i \leq \textit{length } r : r\ i = s\ i \rangle \end{array} \quad (4)$$

with

$$\begin{array}{l} \textit{Permutes} : (s : A^*) \rightarrow (r : A^*) \\ \textbf{post} \quad \langle \forall e : e \in \textit{elems } s \cup \textit{elems } r : \textit{count } e\ s = \textit{count } e\ r \rangle \end{array} \quad (5)$$

and with function *tail*, all of type $A^* \longleftarrow A^*$. Check which of the following hold:

- $\textit{tail} \subseteq \textit{IsPrefixOf}$
- $\textit{IsPrefixOf} \subseteq \textit{Permutes}$



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 ,

$$\perp \subseteq R \subseteq \top \tag{6}$$

Exercises

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

$$R \cdot id = R = id \cdot R \quad (7)$$

$$R \cdot \perp = \perp = \perp \cdot R \quad (8)$$

hold and that composition is associative:

$$R \cdot (S \cdot T) = (R \cdot S) \cdot T \quad (9)$$



Converses

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

$$a(R^\circ)b \equiv b R a \quad (10)$$

Note that converse commutes with composition

$$(R \cdot S)^\circ = S^\circ \cdot R^\circ \quad (11)$$

and with itself:

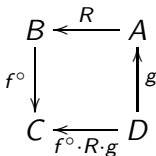
$$(R^\circ)^\circ = R \quad (12)$$

Function converses

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

$$(f \ b)R(g \ a) \equiv b(f^\circ \cdot R \cdot g)a \quad (13)$$

cf. diagram:



Let us see an example of its use.

PF-transform at work

Transforming a well-known PW-formula:

f is injective

$\equiv \{ \text{recall definition from discrete maths} \}$

$\langle \forall y, x :: (f\ y) = (f\ x) \Rightarrow y = x \rangle$

$\equiv \{ \text{introduce } id \text{ (twice)} \}$

$\langle \forall y, x :: (f\ y)id(f\ x) \Rightarrow y(id)x \rangle$

$\equiv \{ \text{rule } (f\ b)R(g\ a) \equiv b(f^\circ \cdot R \cdot g)a \text{ (13)} \}$

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

$\equiv \{ (7) ; \text{then go pointfree via (3)} \}$

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

The other way round

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

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

$$\equiv \{ \text{relational inclusion (3)} \}$$

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

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

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

$$\equiv \{ \text{\(\forall\)-trading ; converse (10)} \}$$

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

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

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

$$\equiv \{ \text{recalling definition from maths} \}$$

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: $\langle \forall a :: b R a \Rightarrow b = a \rangle$ (check as homework).

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

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

Example: kernels of functions

$$\begin{aligned}
 & a'(\ker f)a \\
 \equiv & \quad \{ \text{substitution} \} \\
 & a'(f^\circ \cdot f)a \\
 \equiv & \quad \{ \text{PF-transform rule (13)} \} \\
 & (f a') = (f a)
 \end{aligned}$$

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

Exercise 3: Let C be a nonempty data domain and let $c \in C$. Let \underline{c} be the “everywhere c ” function:

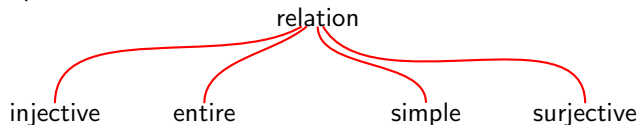
$$\begin{array}{ccc}
 \underline{c} & : & A \longrightarrow C \\
 \underline{c}a & \triangleq & c
 \end{array} \tag{14}$$

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

$$\ker \underline{c} \quad , \quad \underline{b} \cdot \underline{c}^\circ \quad , \quad \text{img } \underline{c} \tag{15}$$

Binary relation taxonomy

Topmost criteria:



Definitions:

	<i>Reflexive</i>	<i>Coreflexive</i>
$\ker R$	entire R	injective R
$\text{img } R$	surjective R	simple R

(16)

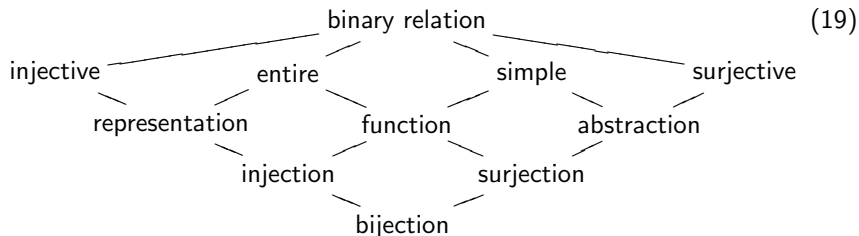
Facts:

$$\ker (R^\circ) = \text{img } R \quad (17)$$

$$\text{img } (R^\circ) = \ker R \quad (18)$$

Binary relation taxonomy

The whole picture:



Exercise 4: Resort to (17,18) and (16) 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

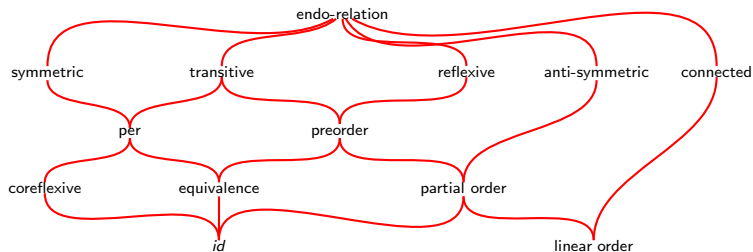
A function f is a binary relation such that

Pointwise	Pointfree	
“Left” Uniqueness		
$b \ f \ a \wedge b' \ f \ a \Rightarrow b = b'$	$\text{img } f \subseteq \text{id}$	(f is simple)
Leibniz principle		
$a = a' \Rightarrow f \ a = f \ a'$	$\text{id} \subseteq \text{ker } f$	(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 $R \cup R^\circ = T$

Orders and their taxonomy

Therefore:

- **Preorders** are reflexive and transitive orders.
Example: $y \text{ } \textit{IsAtMostAsOldAs} \text{ } x$
- **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: $y \text{ } \textit{Permutes} \text{ } x$
- **Pers** are partial equivalences
Example: $y \text{ } \textit{IsBrotherOf} \text{ } x$

Exercises

Exercise 5: Expand all criteria in the previous slides to pointwise notation.



Exercise 6: A relation R is said to be *co-transitive* iff the following holds:

$$\langle \forall b, a : b R a : \langle \exists c : b R c : c R a \rangle \rangle \quad (20)$$

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,

$$b (R \cap S) a \equiv b R a \wedge b S a \quad (21)$$

$$b (R \cup S) a \equiv b R a \vee b S a \quad (22)$$

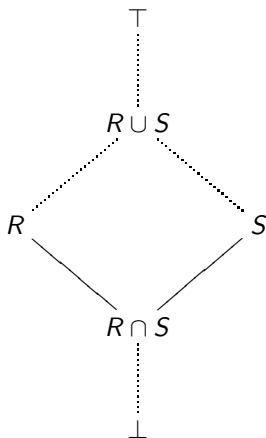
for R, S of the same type. Their meaning is captured by the following **universal** properties:

$$X \subseteq R \cap S \equiv X \subseteq R \wedge X \subseteq S \quad (23)$$

$$R \cup S \subseteq X \equiv R \subseteq X \wedge S \subseteq X \quad (24)$$

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

$$\begin{array}{ccccc}
 & & A & \xleftarrow{\pi_1} & A \times B & \xrightarrow{\pi_2} & B \\
 & & \swarrow R & & \uparrow \langle R, S \rangle & & \searrow S \\
 & & C & & & &
 \end{array}
 \quad (25)$$

where

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

(26)

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

$$\begin{array}{ccccc}
 Bool & \xrightarrow{i_1} & Bool + String & \xleftarrow{i_2} & String \\
 & \searrow \text{Boo} & \downarrow [Boo, Err] & \swarrow Err & \\
 & & X & &
 \end{array} \quad (27)$$

where

$$[R, S] = (R \cdot i_1^\circ) \cup (S \cdot i_2^\circ) \quad \text{cf.} \quad
 \begin{array}{ccccc}
 A & \xrightarrow{i_1} & A + B & \xleftarrow{i_2} & B \\
 & \searrow R & \downarrow [R, S] & \swarrow S & \\
 & & C & &
 \end{array}$$

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

Sums

Example (Haskell):

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

PF-transforms to

$$\begin{array}{ccccc}
 \text{Bool} & \xrightarrow{i_1} & \text{Bool} + \text{String} & \xleftarrow{i_2} & \text{String} \\
 & \searrow \text{Boo} & \downarrow [\text{Boo}, \text{Err}] & \swarrow \text{Err} & \\
 & & X & &
 \end{array} \quad (27)$$

where

$$[R, S] = (R \cdot i_1^\circ) \cup (S \cdot i_2^\circ) \quad \text{cf.} \quad \begin{array}{ccccc} A & \xrightarrow{i_1} & A + B & \xleftarrow{i_2} & B \\ & \searrow R & \downarrow [R, S] & \swarrow S & \\ & & C & & \end{array}$$

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

Last but not least: relational equality

- **Pointwise** equality:

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

- **Pointfree** equality:

- **Cyclic inclusion** (“ping-pong”) rule:

$$R = S \equiv R \subseteq S \wedge S \subseteq R \quad (28)$$

- **Indirect equality** rules¹:

$$R = S \equiv \langle \forall X :: (X \subseteq R \equiv X \subseteq S) \rangle \quad (29)$$

$$\equiv \langle \forall X :: (R \subseteq X \equiv S \subseteq X) \rangle \quad (30)$$

¹Cf. [1], p. 82.

Example of indirect proof

$$\begin{aligned} & X \subseteq (R \cap S) \cap T \\ \equiv & \quad \{ \cap\text{-universal (23)} \} \\ & X \subseteq (R \cap S) \wedge X \subseteq T \\ \equiv & \quad \{ \cap\text{-universal (23)} \} \\ & (X \subseteq R \wedge X \subseteq S) \wedge X \subseteq T \\ \equiv & \quad \{ \wedge \text{ is associative} \} \\ & X \subseteq R \wedge (X \subseteq S \wedge X \subseteq T) \\ \equiv & \quad \{ \cap\text{-universal (23) twice} \} \\ & X \subseteq R \cap (S \cap T) \\ \therefore & \quad \{ \text{indirection} \} \\ & (R \cap S) \cap T = R \cap (S \cap T) \end{aligned} \tag{31}$$

Last but not least: monotonicity

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

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

etc

Exercise 7: Prove the following rules of thumb:

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



Exercises

Exercise 8: Show that (13) holds.



Exercise 9: 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 10: Prove that relational composition preserves *all* relational classes in the taxonomy of (19).



Exercises

Exercise 11: Prove the following fact

A function f is a bijection iff its converse f° is a function (32)

by completing:

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



Exercises

Exercise 12: Prove that $\text{swap} \triangleq \langle \pi_2, \pi_1 \rangle$ is a bijection.



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	$b \perp a$

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:**

$$\langle \forall i : R \wedge S : T \rangle = \langle \forall i : R : S \Rightarrow T \rangle \quad (33)$$

$$\langle \exists i : R \wedge S : T \rangle = \langle \exists i : R : S \wedge T \rangle \quad (34)$$

Background — Eindhoven quantifier calculus

Splitting:

$$\langle \forall j : R : \langle \forall k : S : T \rangle \rangle = \langle \forall k : \langle \exists j : R : S \rangle : T \rangle \quad (35)$$

$$\langle \exists j : R : \langle \exists k : S : T \rangle \rangle = \langle \exists k : \langle \exists j : R : S \rangle : T \rangle \quad (36)$$

One-point:

$$\langle \forall k : k = e : T \rangle = T[k := e] \quad (37)$$

$$\langle \exists k : k = e : T \rangle = T[k := e] \quad (38)$$

Nesting:

$$\langle \forall a, b : R \wedge S : T \rangle = \langle \forall a : R : \langle \forall b : S : T \rangle \rangle \quad (39)$$

$$\langle \exists a, b : R \wedge S : T \rangle = \langle \exists a : R : \langle \exists b : S : T \rangle \rangle \quad (40)$$



R. Bird and O. de Moor.

Algebra of Programming.

Series in Computer Science. Prentice-Hall International, 1997.

C.A.R. Hoare, series editor.