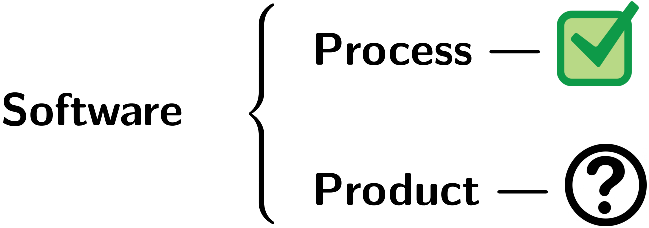
CSI - A Calculus for Information Systems (2022/23)

Class 1 — About FM

Global picture

Concerning software ‘engineering’:



Software

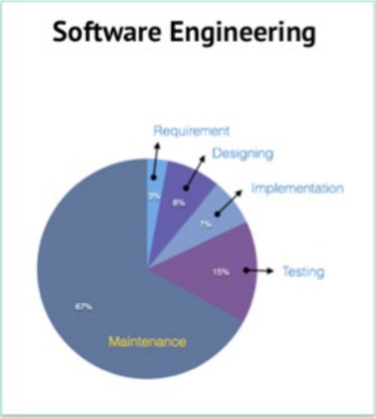
Process — ✓

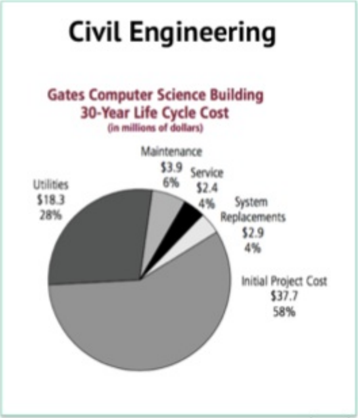
Product — ?

Formal methods provide an answer to the question mark above.

Global picture

Concerning software ‘engineering’:





Credits: Zhenjiang Hu, NII, Tokyop JP

Have you ever used a FM?

Of course you have! Check this:

A problem

My three children
were born at a 3 year
interval rate.
Altogether, they are
as old as me. I am 48.
How old are they?

A model

$$x + (x + 3) + (x + 6) = 48$$

— maths description of the problem.

Some calculations

$$\begin{array}{cl} & 3x + 9 = 48 \\ \equiv & \quad \{ \text{"al-djabr" rule} \} \\ & 3x = 48 - 9 \\ \equiv & \quad \{ \text{"al-hatt" rule} \} \\ & x = 16 - 3 \end{array}$$

The solution

$$\begin{array}{rcl} x & = & 13 \\ x + 3 & = & 16 \\ x + 6 & = & 19 \end{array}$$

Have you ever used a FM?

Of course you have! Check this:

A problem

My three children
were born at a 3 year
interval rate.
Altogether, they are
as old as me. I am 48.
How old are they?

A model

$$x + (x + 3) + (x + 6) = 48$$

— maths description of the problem.

Some calculations

$$\begin{array}{ll} & 3x + 9 = 48 \\ \equiv & \qquad \{ \text{"al-djabr" rule} \} \\ & 3x = 48 - 9 \\ \equiv & \qquad \{ \text{"al-hatt" rule} \} \\ & x = 16 - 3 \end{array}$$

The solution

$$\begin{array}{rcl} x & = & 13 \\ x + 3 & = & 16 \\ x + 6 & = & 19 \end{array}$$

Have you ever used a FM?

Of course you have! Check this:

A problem

My three children were born at a 3 year interval rate. Altogether, they are as old as me. I am 48. How old are they?

A model

$$x + (x+3) + (x+6) = 48$$

— maths description of the problem.

Some calculations

$$\begin{array}{ll} & 3x + 9 = 48 \\ \equiv & \qquad \{ \text{ "al-djabr" rule } \} \\ & 3x = 48 - 9 \\ \equiv & \qquad \{ \text{ "al-hatt" rule } \} \\ & x = 16 - 3 \end{array}$$

The solution

$$\begin{array}{rcl} x & = & 13 \\ x+3 & = & 16 \\ x+6 & = & 19 \end{array}$$

Have you ever used a FM?

Of course you have! Check this:

A problem

My three children were born at a 3 year interval rate. Altogether, they are as old as me. I am 48. How old are they?

A model

$$x + (x + 3) + (x + 6) = 48$$

— maths description of the problem.

Some calculations

$$\begin{array}{ll} & 3x + 9 = 48 \\ \equiv & \{ \text{"al-djabr" rule} \} \\ & 3x = 48 - 9 \\ \equiv & \{ \text{"al-hatt" rule} \} \\ & x = 16 - 3 \end{array}$$

The solution

$$\begin{array}{rcl} x & = & 13 \\ x + 3 & = & 16 \\ x + 6 & = & 19 \end{array}$$

Have you ever used a FM?

"Al-djabr" rule ? "al-hatt" rule ?

al-djabr

$$x - \enclose{circle}{z} \leq y \quad \equiv \quad x \leq y + \enclose{circle}{z}$$

all-hatt

$$x * \enclose{circle}{z} \leq y \quad \equiv \quad x \leq y * \enclose{circle}{z^{-1}} \qquad (z > 0)$$

These rules that you have used so many times were discovered by Persian mathematicians, notably by Al-Huwarizmi (9c AD).

NB: "**algebra**" stems from "**al-djabr**" and "**algarismo**" from **Al-Huwarizmi**.

Software problems

Now, suppose the **problem** was

> *Please write a program to list the students of my class ordered by their marks.*

Is there a mathematical **model** for this problem?

Yes, of course there is — see aside:

$$sort \subseteq \frac{bag}{bag} \cap \frac{true}{sorted}$$

where

$$sorted = \ldots marks \ldots$$

$$bag = \ldots.$$

But,

- what do $X \cap Y$, $\frac{f}{g}$ ... mean here?
- Is there an "**algebra**" for such symbols?

Yes — Wait and see :-)

Software problems

Now, suppose the **problem** was

Please write a program to list the students of my class ordered by their marks.

Is there a mathematical **model** for this problem?

Yes, of course there is — see aside:

$$sort \subseteq \frac{bag}{bag} \cap \frac{true}{sorted}$$

where

$$sorted = \ldots marks \ldots$$

$$bag = \ldots.$$

But,

- what do $X \cap Y$, $\frac{f}{g}$ ... mean here?
- Is there an "**algebra**" for such symbols?

Yes — Wait and see :-)

Software problems

Now, suppose the **problem** was

> *Please write a program to list the students of my class ordered by their marks.*

Is there a mathematical **model** for this problem?

Yes, of course there is — see aside:

$$
\begin{aligned}
&sort \subseteq \frac{bag}{bag} \cap \frac{true}{sorted}\\
&\quad \textbf{where}\\
&\qquad sorted = \ldots marks \ldots\\
&\qquad bag = ...
\end{aligned}
$$

But,

- what do $X \cap Y$, $\frac{f}{g}$ ... mean here?
- Is there an "**algebra**" for such symbols?

Yes — Wait and see :-)

Software problems

Now, suppose the **problem** was

> *Please write a program to list the students of my class ordered by their marks.*

Is there a mathematical **model** for this problem?

Yes, of course there is — see aside:

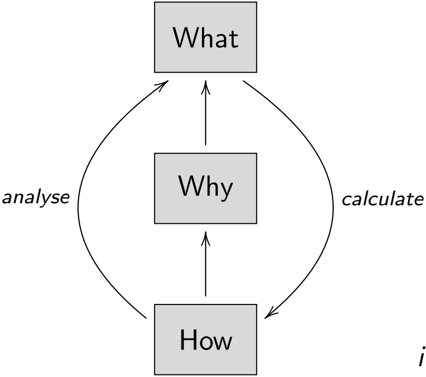
$$
\begin{array}{l}
sort \;\subseteq\; \frac{bag}{bag} \cap \frac{true}{sorted} \\
\quad \textbf{where} \\
\qquad sorted = \ldots marks \ldots \\
\qquad bag = \ldots.
\end{array}
$$

But,

- what do $X \cap Y$, $\frac{f}{g}$ ... mean here?
- Is there an "**algebra**" for such symbols?

Yes — Wait and see :-)

FM — scientific software design



What

Why

How

analyse

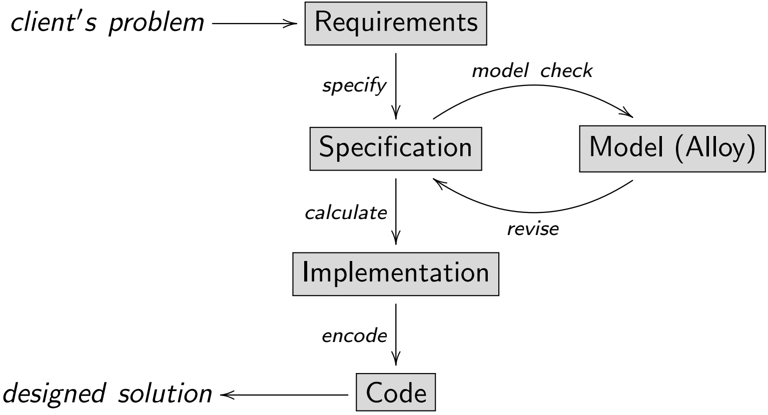
calculate

specification (*model*)

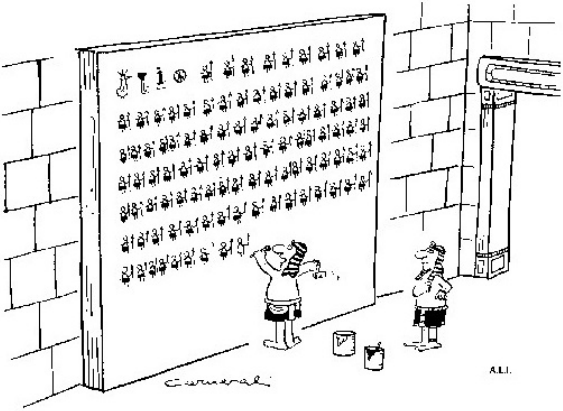
justification

implementation (*program*)

FM — simplified life-cycle



Notation matters!



Are you sure there isn't a simpler means of writing 'The Pharaoh had 10,000 soldiers?'

Credits: Cliff B. Jones 1980 [4]

Well-known FM notations / tools / resources

Just a sample, as there are many — follow the links (in alphabetic order):

Notations:

- Alloy
- B-Method
- JML
- mCRL2
- SPARK-Ada
- TLA+
- VDM
- Z

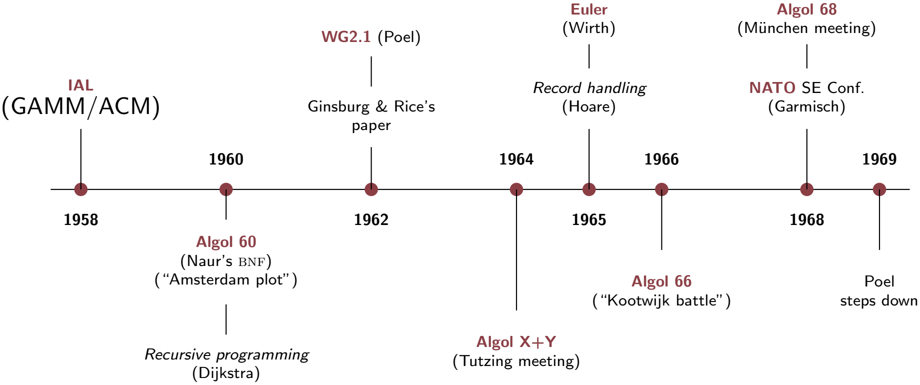
Tools:

- Alloy 4
- Coq
- Frama-C
- NuSMV
- Overture

Resources:

- Formal Methods Europe
- Formal Methods wiki (Oxford)

60+ years ago (1958-)



1958 — **IAL** (GAMM/ACM)

1960 — **Algol 60** (Naur's BNF) ("Amsterdam plot") — *Recursive programming* (Dijkstra)

1962 — Ginsburg & Rice's paper — **WG2.1** (Poel)

1964 — **Algol X+Y** (Tutzing meeting)

1965 — *Record handling* (Hoare) — **Euler** (Wirth)

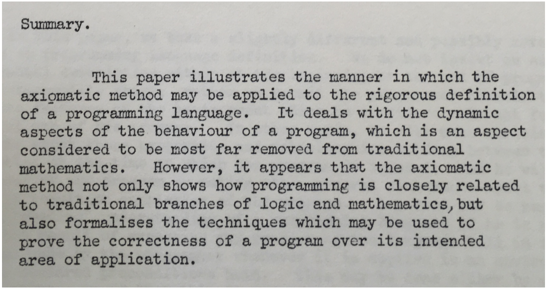
1966 — **Algol 66** ("Kootwijk battle")

1968 — **NATO** SE Conf. (Garmisch) — **Algol 68** (München meeting)

1969 — Poel steps down

Hoare Logic — “turning point” (1968)

Floyd-Hoare logic for **program correctness** dates back to 1968:



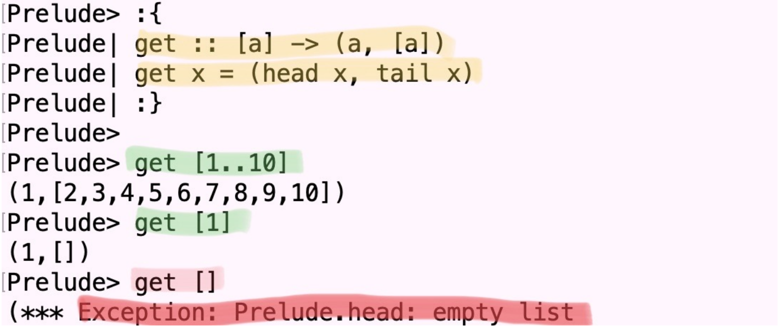
Summary.

This paper illustrates the manner in which the axiomatic method may be applied to the rigorous definition of a programming language. It deals with the dynamic aspects of the behaviour of a program, which is an aspect considered to be most far removed from traditional mathematics. However, it appears that the axiomatic method not only shows how programming is closely related to traditional branches of logic and mathematics, but also formalises the techniques which may be used to prove the correctness of a program over its intended area of application.

(ADB/IFIP/1164;1456)

Inv/pre/post

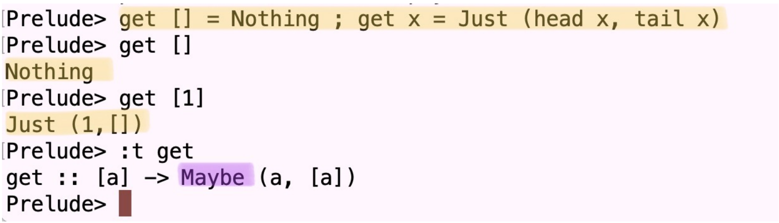
Starting where (pure) **functions** stop:



```
[Prelude> :{
[Prelude| get :: [a] -> (a, [a])
[Prelude| get x = (head x, tail x)
[Prelude| :}
[Prelude>
[Prelude> get [1..10]
(1,[2,3,4,5,6,7,8,9,10])
[Prelude> get [1]
(1,[])
[Prelude> get []
(*** Exception: Prelude.head: empty list
```

Inv/pre/post

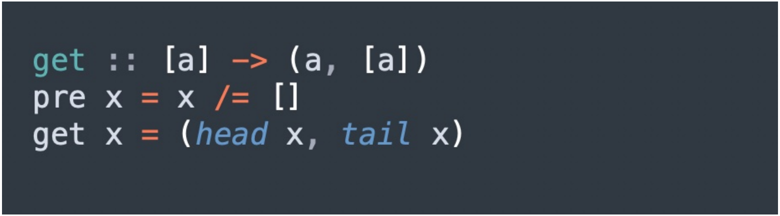
Error handling...



```
Prelude> get [] = Nothing ; get x = Just (head x, tail x)
Prelude> get []
Nothing
Prelude> get [1]
Just (1,[])
Prelude> :t get
get :: [a] -> Maybe (a, [a])
Prelude> 
```

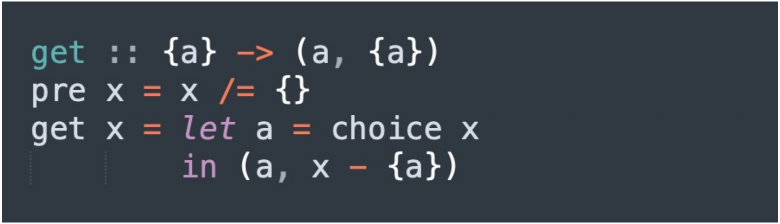
Inv/pre/post

Pre-conditions?



```
get :: [a] -> (a, [a])
pre x = x /= []
get x = (head x, tail x)
```

Not everything is a **list**, a **tree** or a **stream**...



```
get :: {a} -> (a, {a})
pre x = x /= {}
get x = let a = choice x
        in (a, x - {a})
```

Inv/pre/post

`pre`...? `choice`...?

- Non-determinism
- Parallelism
- Abstraction

Inv/pre/post

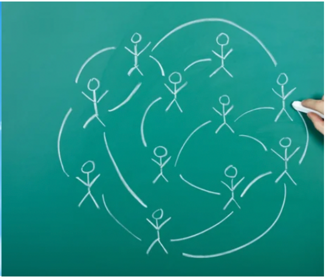
`pre`...? `choice`...?

- Non-determinism
- Parallelism
- Abstraction

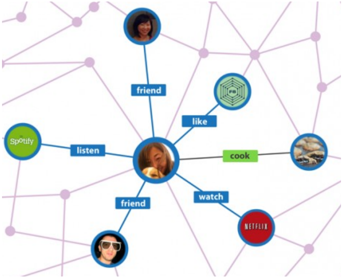
Functions not enough!

Solution?

Relations *(which extend functions)*



Is “everything” a relation?



How to “dematerialize“ them?

Software is pre-science — **formal** but not fully **calculational**

Software is too diverse — many approaches, lack of unity

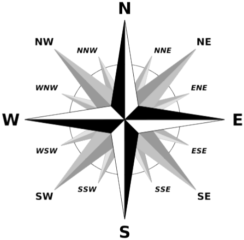
Software is too wide a concept — from assembly to quantum programming

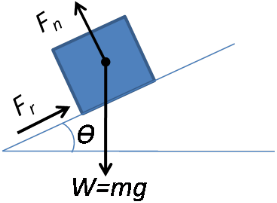
Can you think of a **unified** theory able to express and reason about software *in general*?

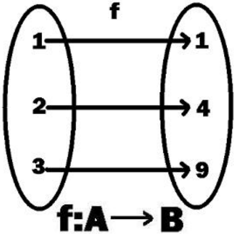
Put in another way:

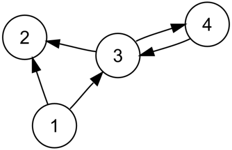
> *Is there a “lingua franca” for the software sciences?*

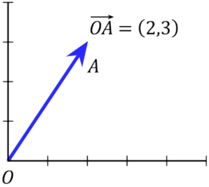
Check the pictures...



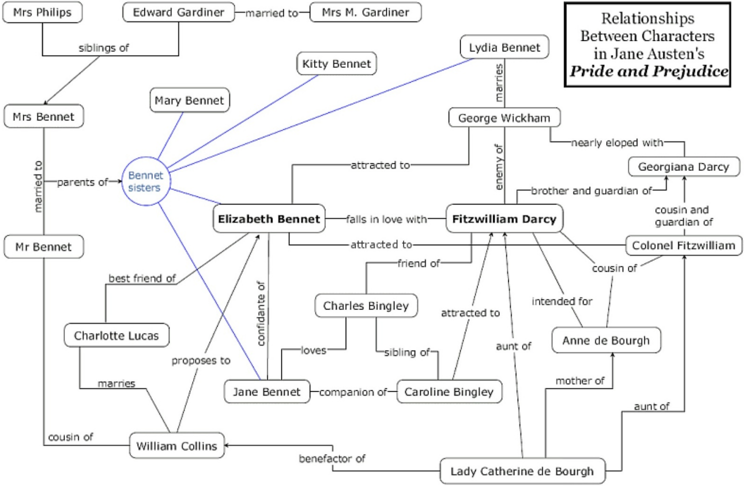






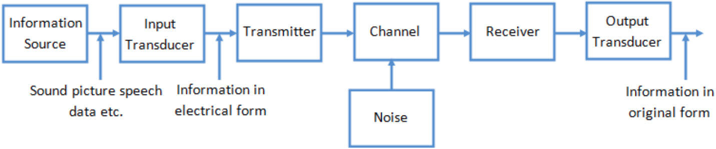


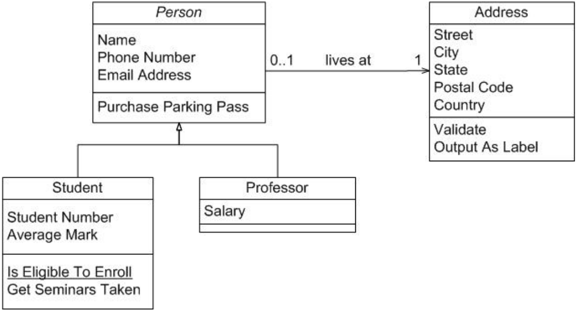
Check the pictures



(Wikipedia: **Pride and Prejudice**, by Jane Austin, 1813.)

Check the pictures


Information Source
Input Transducer
Transmitter
Channel
Receiver
Output Transducer
Sound picture speech data etc.
Information in electrical form
Noise
Information in original form



Check the pictures

Which **graphical** device have you found **common** to **all** pictures?

Arrows everywhere

Arrows! Thus we identify a (graphical) ingredient **common** to describing (several) **different** fields of human activity.

For this ingredient to be able to support a **generic** theory of systems, mind the remarks:

- We need a **generic** notation able to cope with very distinct problem domains, e.g. **process** theory versus **database** theory, for instance.
- Notation is not enough — we need to **reason** and **calculate** about software.
- Semantics-rich **diagram** representations are welcome.
- System description may have a **quantitative** side too.

Class 2 — Going Relational

Relation algebra

In previous courses you may have used **predicate logic**, **finite automata**, **grammars** etc to capture the meaning of real-life problems.

Question:

Is there a unified formalism for **formal modelling***?*

Relation algebra

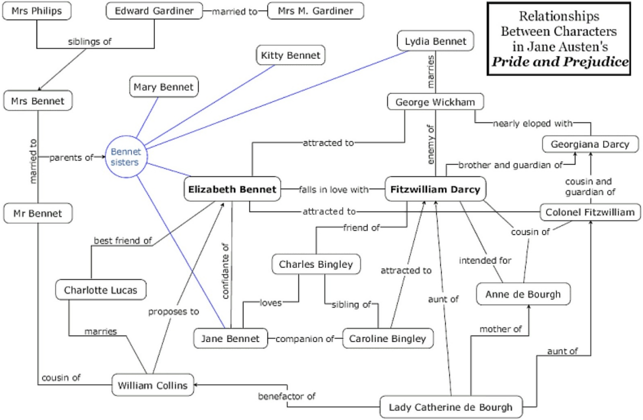
Historically, predicate logic was **not** the first to be proposed:

- Augustus de Morgan (1806-71) — recall *de Morgan* laws — proposed a **Logic of Relations** as early as 1867.
- Predicate logic appeared later.

Perhaps de Morgan was right in the first place: in real life, "everything is a **relation**"...

Everything is a relation...

... as diagram



shows. (Wikipedia: **Pride and Prejudice**, by Jane Austin, 1813.)

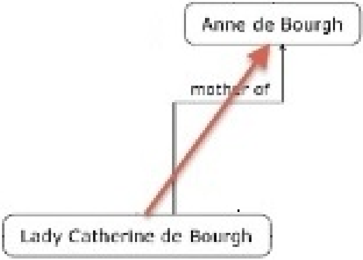
Arrow notation for relations

The picture is a collection of **relations** — vulg. a **semantic network** — elsewhere known as a (binary) **relational system**.

However, in spite of the use of **arrows** in the picture (aside) not many people would write

$$mother\_of : People \to People$$

as the **type** of **relation** $mother\_of$.



Pairs

Consider assertions

$$\begin{array}{rcl} 0 & \leqslant & \pi \\ \texttt{Catherine} & \mathit{isMotherOf} & \texttt{Anne} \\ 3 & = (1+) & 2 \end{array}$$

They are statements of fact concerning various kinds of object — real numbers, people, natural numbers, etc

They involve **two** such objects, that is, **pairs**

$$\begin{array}{c} (0, \pi) \\ (\texttt{Catherine}, \texttt{Anne}) \\ (3, 2) \end{array}$$

respectively.

Sets of pairs

So, we might have written instead:

$$\begin{array}{rcl} (0, \pi) & \in & \leqslant \\ (\texttt{Catherine}, \texttt{Anne}) & \in & \mathit{isMotherOf} \\ (3, 2) & \in & (1+) \end{array}$$

What are $(\leqslant)$, $\mathit{isMotherOf}$, $(1+)$?

- they could be regarded as **sets of pairs**
- better: they should be regarded as **binary relations**.

Therefore,

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

$$A \xrightarrow{R} B \ .$$

Notation

Infix notation

The usual infix notation used in natural language — eg. Catherine isMotherOf Anne — and in maths — eg. $0 \leqslant \pi$ — extends to arbitrary $B \xleftarrow{R} A$: we write

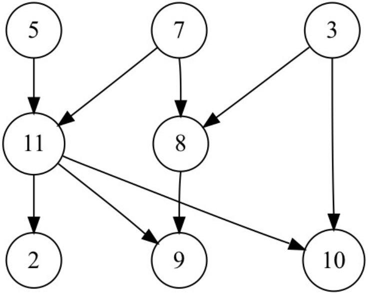
$$b\ R\ a$$

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

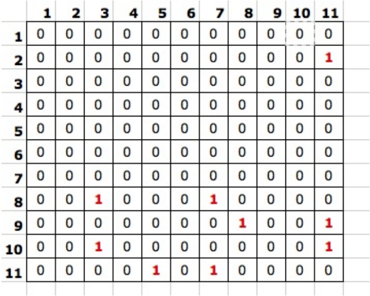
Binary relations are matrices

Binary relations can be regarded as Boolean **matrices**, eg.

Relation R:



Matrix M:



	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0
8	0	0	1	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	1	0	0	1
10	0	0	1	0	0	0	0	0	0	0	1
11	0	0	0	0	1	0	1	0	0	0	0

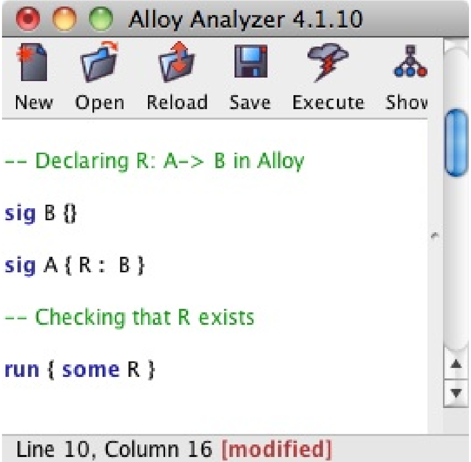
In this case $A = B = \{1..11\}$. Relations $A \xleftarrow{R} A$ over a single type are also referred to as (directed) **graphs**.

Alloy: where "everything is a relation"

Declaring binary relation $A \xrightarrow{R} B$ is **Alloy** (aside).

Alloy is a tool designed at MIT (`http://alloy.mit.edu/alloy`)

We shall be using **Alloy** [3] in this course.



```
-- Declaring R: A-> B in Alloy

sig B {}

sig A { R : B }

-- Checking that R exists

run { some R }
```

Line 10, Column 16 [modified]

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,

$$b\ f\ a \text{ literally means } b = f\ a \tag{1}$$

Therefore, we generalize

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

Exercise

Taken from Propositiones ad acuendos iuuenes ("Problems to Sharpen the Young"), by abbot Alcuin of York († 804):

> XVIII. Propositio de homine et capra et lvpo.
> *Homo quidam debebat ultra fluuium transferre lupum, capram, et fasciculum cauli. Et non potuit aliam nauem inuenire, nisi quae duos tantum ex ipsis ferre ualebat. Praeceptum itaque ei fuerat, ut omnia haec ultra illaesa omnino transferret. Dicat, qui potest, quomodo eis illaesis transire potuit?*

Exercise

> XVIII. Fox, goose and bag of beans puzzle. *A farmer goes to market and purchases a fox, a goose, and a bag of beans. On his way home, the farmer comes to a river bank and hires a boat. But in crossing the river by boat, the farmer could carry only himself and a single one of his purchases - the fox, the goose or the bag of beans. (If left alone, the fox would eat the goose, and the goose would eat the beans.) Can the farmer carry himself and his purchases to the far bank of the river, leaving each purchase intact?*

Identify the main **types** and **relations** involved in the puzzle and draw them in a diagram.

Propositio de homine et capra et lvpo

Data types:

$$Being = \{Farmer, Fox, Goose, Beans\} \quad (2)$$

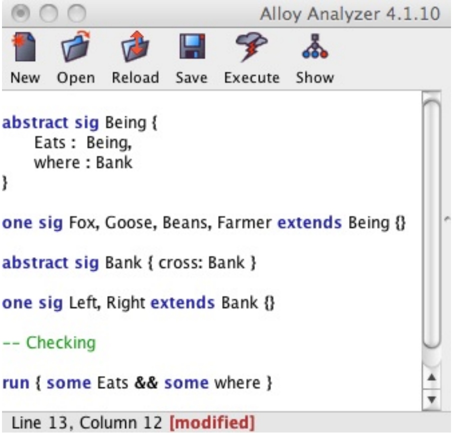
$$Bank = \{Left, Right\} \quad (3)$$

Relations:

$$\begin{tikzcd} Being \arrow[r, "Eats"] & Being \arrow[d, "where"'] & \\ & Bank \arrow[r, "cross"] & Bank \end{tikzcd} \quad (4)$$

Propositio de homine et capra et lvpo

Specification source written in Alloy:

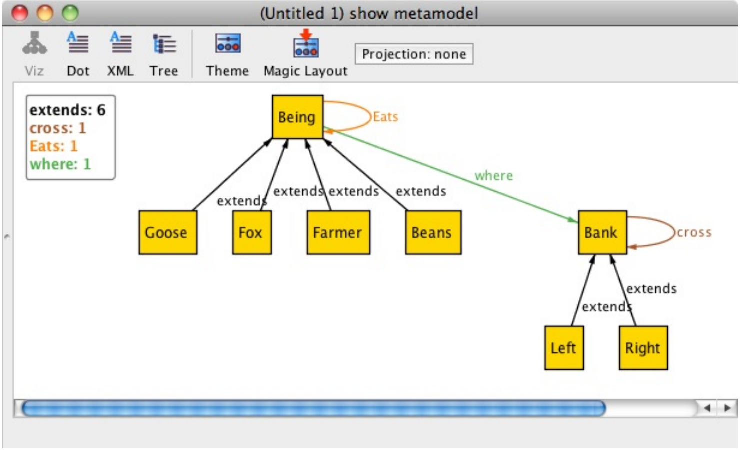


```
abstract sig Being {
     Eats :  Being,
     where : Bank
}

one sig Fox, Goose, Beans, Farmer extends Being {}

abstract sig Bank { cross: Bank }

one sig Left, Right extends Bank {}

-- Checking

run { some Eats && some where }
```

Line 13, Column 12 [modified]

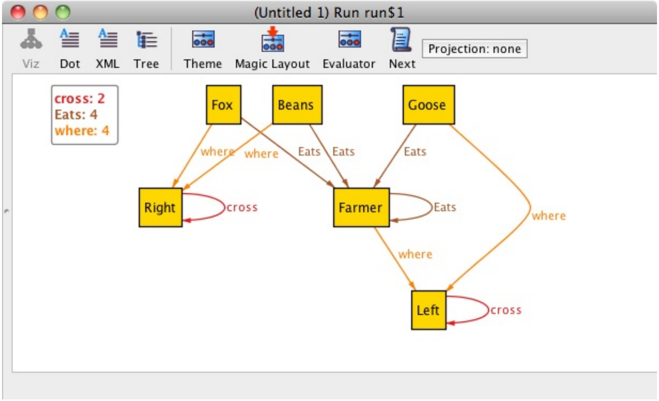
Propositio de homine et capra et lvpo

Diagram of specification (model) given by Alloy:



Propositio de homine et capra et lvpo

Diagram of instance of the model given by Alloy:



Silly instance, why? — specification too **loose**...

Composition

Recall **function composition** (aside).

$$\begin{tikzcd} B & A \arrow[l, "f"'] & C \arrow[l, "g"'] \arrow[ll, bend left, "f \cdot g"] \end{tikzcd} \qquad (5)$$

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

We extend $f \cdot g$ to relational composition $R \cdot S$ in the obvious way:

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

Composition

That is:

$$\begin{tikzcd} B & A \arrow[l, "R"'] & C \arrow[l, "S"'] \arrow[ll, bend left, "R\cdot S"] \end{tikzcd}$$

$$b(R \cdot S)c \quad \equiv \quad \langle \exists\ a\ ::\ b\ R\ a\ \wedge\ a\ S\ c \rangle \tag{6}$$

Example: $Uncle = Brother \cdot Parent$, *that expands to*

$$u\ Uncle\ c \quad \equiv \quad \langle \exists\ p\ ::\ u\ Brother\ p \wedge p\ Parent\ c \rangle$$

Note how this rule *removes* $\exists$ when applied from right to left.

Notation $R \cdot S$ is said to be **point-free** (no variables, or points).

Check generalization

Back to functions, (6) becomes[1]

$$
\begin{array}{rcl}
b(f \cdot g)c & \equiv & \langle \exists\ a\ ::\ b\ f\ a \wedge a\ g\ c \rangle \\
& \equiv & \qquad \{\ a\ g\ c \text{ means } a = g\ c\ (1)\ \} \\
& & \langle \exists\ a\ ::\ b\ f\ a \wedge a = g\ c \rangle \\
& \equiv & \qquad \{\ \exists\text{-trading (197) ; } b\ f\ a \text{ means } b = f\ a\ (1)\ \} \\
& & \langle \exists\ a\ :\ a = g\ c\ :\ b = f\ a \rangle \\
& \equiv & \qquad \{\ \exists\text{-one point rule (201)}\ \} \\
& & b = f(g\ c)
\end{array}
$$

So, we easily recover what we had before (5).

[1] Check the appendix on predicate calculus.

Relation inclusion

Relation inclusion generalizes function equality:

Equality *on functions*

$$f = g \quad \equiv \quad \langle \forall\ a\ ::\ f\ a = g\ a \rangle \tag{7}$$

generalizes to **inclusion** *on relations:*

$$R \subseteq S \quad \equiv \quad \langle \forall\ b, a\ :\ b\ R\ a :\ b\ S\ a \rangle \tag{8}$$

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

Inclusion is **typed**:

For $R \subseteq S$ to hold both R and S need to be of the same **type***, say* $B \xleftarrow{R,S} A$.

Relation inclusion

$R \subseteq S$ is a partial order, that is, it is

reflexive,

$$id \ \subseteq \ R \tag{9}$$

transitive

$$R \ \subseteq \ S \wedge S \ \subseteq \ Q \Rightarrow R \ \subseteq \ Q \tag{10}$$

and **antisymmetric**:

$$R \subseteq S \ \wedge \ S \subseteq R \quad \equiv \quad R = S \tag{11}$$

Therefore:

$$R = S \ \equiv \ \langle \forall \ b, a \ :: \ b \ R \ a \ \equiv \ b \ S \ a \rangle \tag{12}$$

Relational equality

Both (12) and (11) establish **relation equality**, resp. in PW/PF fashion.

Rule (11) is also called "ping-pong" or **cyclic inclusion**, often taking the format

$$
\begin{array}{lll}
 & R & \\
\subseteq & & \{ \; .... \; \} \\
 & S & \\
\subseteq & & \{ \; .... \; \} \\
 & R & \\
:: & & \{ \; \text{"ping-pong" (11)} \; \} \\
 & R = S &
\end{array}
$$

Indirect relation equality

Most often we prefer an *indirect* way of proving relation equality:

Indirect equality *rules:*

$$R = S \equiv \langle \forall\, X :: (X \subseteq R \equiv X \subseteq S) \rangle \tag{13}$$

$$\equiv \langle \forall\, X :: (R \subseteq X \equiv S \subseteq X) \rangle \tag{14}$$

Compare with eg. equality of sets in discrete maths:

$$A = B \equiv \langle \forall\, a :: a \in A \equiv b \in B \rangle$$

Indirect relation equality

The typical layout is e.g.

$$\left\{\begin{array}{ll} & X \subseteq R \\ \equiv & \quad \{ \;\ldots\; \} \\ & X \subseteq \ldots \\ \equiv & \quad \{ \;\ldots\; \} \\ & X \subseteq S \\ :: & \quad \{ \text{ indirect equality (13) } \} \\ & R = S \\ \Box & \end{array}\right.$$

Special relations

Every type $B \longleftarrow A$ has its

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

Every type $A \longleftarrow A$ has the

- **identity** relation $A \xleftarrow{id} A$ which is nothing but function
$$id\ a = a \tag{15}$$

Clearly, for every R,

$$\perp \subseteq R \subseteq \top \tag{16}$$

Diagrams

Assertions of the form $X \subseteq Y$ where X and Y are relation compositions can be represented graphically by square-shaped diagrams, see the following exercise.

Exercise 1: Let $a\ S\ n$ mean: *"student a is assigned number n"*. Using (6) and (8), check that assertion

$S \cdot \geqslant\ \subseteq\ \top \cdot S$ depicted by diagram

$$\begin{tikzcd}
\mathbb{N} \arrow[d, "s"'] & \mathbb{N} \arrow[l, "\geqslant"'] \arrow[d, "s"] \arrow[dl, phantom, "\subseteq"] \\
A & A \arrow[l, "\top"]
\end{tikzcd}$$

means that numbers are assigned to students sequentially. □

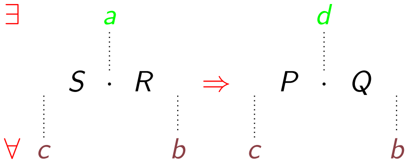
Diagrams

Pointfree:

$$\begin{tikzcd}
A \arrow[d, "S"'] & B \arrow[l, "R"'] \arrow[d, "Q"] \arrow[dl, phantom, "\subseteq"] \\
C & D \arrow[l, "P"]
\end{tikzcd}$$

$$S \cdot R \subseteq P \cdot Q$$

Pointwise:



Exercises

Exercise 2: Use (6) and (8) and predicate calculus to show that

$$R \cdot id \;=\; R \;=\; id \cdot R \tag{17}$$

$$R \cdot \bot \;=\; \bot \;=\; \bot \cdot R \tag{18}$$

hold and that composition is associative:

$$R \cdot (S \cdot T) = (R \cdot S) \cdot T \tag{19}$$

□

Exercise 3: Use (7), (8) and predicate calculus to show that

$$f \subseteq g \quad \equiv \quad f = g$$

holds (moral: for functions, inclusion and equality coincide). □

(**NB**: see the appendix for a compact set of rules of the predicate calculus.)

Converses

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

$$a(R^\circ)b \equiv b\ R\ a \tag{20}$$

Note that converse commutes with composition

$$(R \cdot S)^\circ = S^\circ \cdot R^\circ \tag{21}$$

and with itself:

$$(R^\circ)^\circ = R \tag{22}$$

Converse captures the **passive voice**: *Catherine eats the apple* — $R = (eats)$ — is the same as *the apple is eaten by Catherine* — $R^\circ = (is\ eaten\ by)$.

Function converses

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

$$(f\ \ b)R(g\ \ a) \quad \equiv \quad b(f^\circ \cdot R \cdot g)a \tag{23}$$

cf. diagram:

$$\begin{tikzcd}
C & D \arrow[l, "R"'] \\
B \arrow[u, "f"] & A \arrow[u, "g"'] \arrow[l, "f^\circ \cdot R \cdot g"]
\end{tikzcd}$$

Therefore (tell why):

$$b(f^\circ \cdot g)a \quad \equiv \quad f\ \ b = g\ \ a \tag{24}$$

Let us see an example of using these rules.

Class 3 — The “Zoo” of Binary Relations

PF-transform at work

Transforming a well-known PW-formula into PF notation:

$$
\begin{array}{ll}
 & f \text{ is } \textbf{injective} \\
\equiv & \qquad \{ \text{ recall definition from discrete maths } \} \\
 & \langle \forall\, y,x \;:\; (f\ y) = (f\ x) :\; y = x \rangle \\
\equiv & \qquad \{ \text{ (24) for } f = g \ \} \\
 & \langle \forall\, y,x \;:\; y(f^\circ \cdot f)x :\; y = x \rangle \\
\equiv & \qquad \{ \text{ (23) for } R = f = g = id \ \} \\
 & \langle \forall\, y,x \;:\; y(f^\circ \cdot f)x :\; y(id)x \rangle \\
\equiv & \qquad \{ \text{ go pointfree (8) i.e. drop } y,x \ \} \\
 & f^\circ \cdot f \subseteq id
\end{array}
$$

The other way round

Now check what $id \subseteq f \cdot f^\circ$ means:

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

$$\equiv \qquad \{ \text{ relational inclusion (8) } \}$$

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

$$\equiv \qquad \{ \text{ identity relation ; composition (6) } \}$$

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

$$\equiv \qquad \{ \text{ } \forall\text{-one point (200) ; converse (20) } \}$$

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

$$\equiv \qquad \{ \text{ trivia ; function } f \text{ } \}$$

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

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

f is **surjective**

Why *id* (really) matters

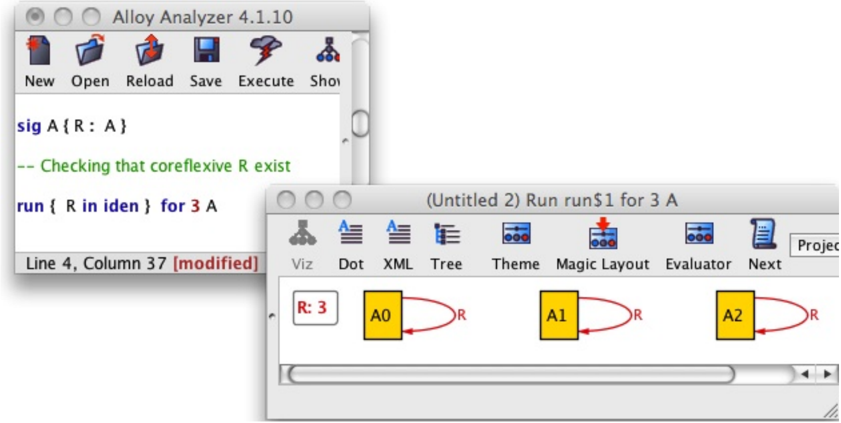
Terminology:

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

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

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

Alloy: checking for coreflexive relations



Alloy Analyzer 4.1.10

```
sig A { R : A }

-- Checking that coreflexive R exist

run { R in iden } for 3 A
```

Line 4, Column 37 [modified]

(Untitled 2) Run run$1 for 3 A

Kernels of functions

Meaning of $\ker f$:

$$\begin{array}{cl} & a'(\ker f)a \\ \equiv & \{\ \text{substitution}\ \} \\ & a'(f^\circ \cdot f)a \\ \equiv & \{\ \text{rule (24)}\ \} \\ & f\ a' = f\ a \end{array}$$

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

Exercise 4: Let K be a nonempty data domain, $k \in K$ and $\underline{k}$ be the *"everywhere* k*"* function:

$$\begin{array}{rcl} \underline{k} & : & A \longrightarrow K \\ \underline{k}\,a & = & k \end{array} \qquad (25)$$

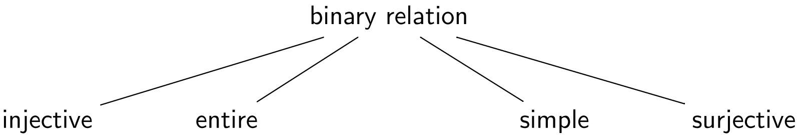
Compute which relations are defined by the following expressions:

$$\ker \underline{k}, \quad \underline{b} \cdot \underline{c}^\circ, \quad \operatorname{img} \underline{k} \qquad (26)$$

□

Binary relation taxonomy

Topmost criteria:



binary relation

injective

entire

simple

surjective

Definitions:

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

(27)

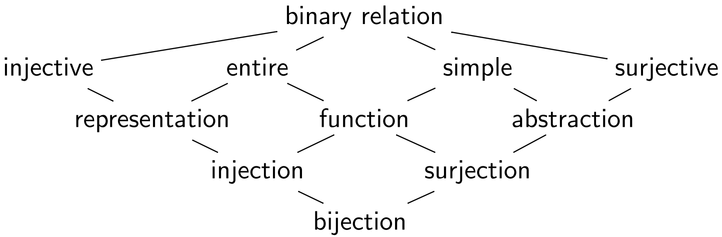
Facts:

$$\ker (R^\circ) = \operatorname{img} R \tag{28}$$

$$\operatorname{img} (R^\circ) = \ker R \tag{29}$$

Binary relation taxonomy

The whole picture:



binary relation

injective entire simple surjective

representation function abstraction

injection surjection

bijection

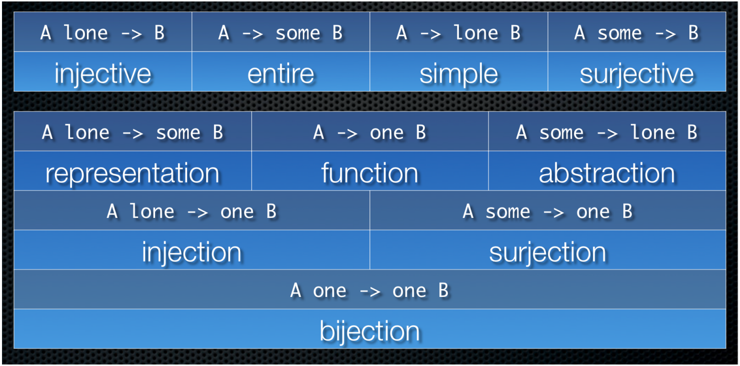
(30)

Exercise 5: Resort to (28,29) and (27) to prove the following rules of thumb:

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

□

The same in Alloy

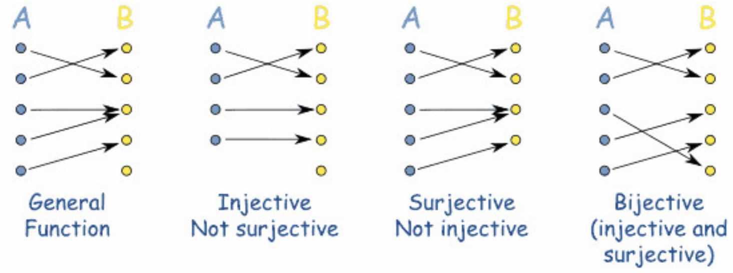


<table>
<tr><td colspan="3"><code>A lone -> B</code></td><td colspan="3"><code>A -> some B</code></td><td colspan="3"><code>A -> lone B</code></td><td colspan="3"><code>A some -> B</code></td></tr>
<tr><td colspan="3">injective</td><td colspan="3">entire</td><td colspan="3">simple</td><td colspan="3">surjective</td></tr>
<tr><td colspan="4"><code>A lone -> some B</code></td><td colspan="4"><code>A -> one B</code></td><td colspan="4"><code>A some -> lone B</code></td></tr>
<tr><td colspan="4">representation</td><td colspan="4">function</td><td colspan="4">abstraction</td></tr>
<tr><td colspan="6"><code>A lone -> one B</code></td><td colspan="6"><code>A some -> one B</code></td></tr>
<tr><td colspan="6">injection</td><td colspan="6">surjection</td></tr>
<tr><td colspan="12"><code>A one -> one B</code></td></tr>
<tr><td colspan="12">bijection</td></tr>
</table>

(Courtesy of Alcino Cunha.)

Exercises

Exercise 6: Label the items (uniquely) in these drawings[2]



and compute, in each case, the **kernel** and the **image** of each relation. Why are all these relations **functions**? □

[2] Credits: `http://www.matematikaria.com/unit/injective-surjective-bijective.html`.

Exercises

Exercise 7: Prove the following fact

$$\textit{A relation } f \textit{ is a bijection } \textbf{iff} \textit{ its converse } f^{\circ} \textit{ is a function} \tag{31}$$

by completing:

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

□

Propositio de homine et capra et lvpo

Exercise 8: Let relation $Bank \xrightarrow{cross} Bank$ (4) be defined by:

$$\begin{array}{rcl} Left & cross & Right \\ Right & cross & Left \end{array}$$

It therefore is a bijection. Why? □

Exercise 9: Check which of the following properties,

simple, entire,
injective,
surjective,
reflexive,
coreflexive

Eats	*Fox*	*Goose*	*Beans*	*Farmer*
Fox	0	1	0	0
Goose	0	0	1	0
Beans	0	0	0	0
Farmer	0	0	0	0

hold for relation *Eats* (4) above ("food chain" $Fox > Goose > Beans$).
□

Propositio de homine et capra et lvpo

Exercise 10: Relation $where : Being \rightarrow Bank$ should obey the following constraints:

- *everyone is somewhere in a bank*
- *no one can be in both banks at the same time*.

Express such constraints in relational terms. Conclude that *where* should be a **function**. □

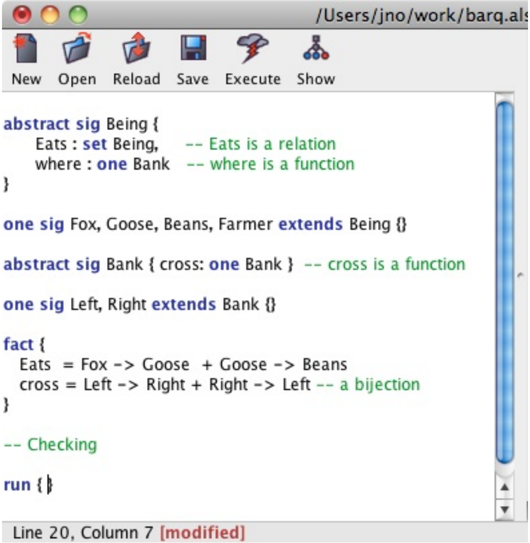
Exercise 11: There are only two **constant** functions (25) in the type $Being \longrightarrow Bank$ of *where*. Identify them and explain their role in the puzzle. □

Exercise 12: Two functions f and g are bijections iff $f^\circ = g$, recall (31). Convert $f^\circ = g$ to point-wise notation and check its meaning. □

Propositio de homine et capra et lvpo

Adding detail to the previous **Alloy** model (aside)

(More about Alloy syntax and semantics later.)



```
abstract sig Being {
    Eats : set Being,      -- Eats is a relation
    where : one Bank       -- where is a function
}

one sig Fox, Goose, Beans, Farmer extends Being {}

abstract sig Bank { cross: one Bank }  -- cross is a function

one sig Left, Right extends Bank {}

fact {
  Eats  = Fox -> Goose  + Goose -> Beans
  cross = Left -> Right + Right -> Left -- a bijection
}

-- Checking

run {}
```

Line 20, Column 7 [modified]

Functions in one slide

Recapitulating: a **function** f is a binary relation such that

Pointwise	Pointfree	
“Left” Uniqueness		
$b\ f\ a \wedge b'\ f\ a \quad \Rightarrow \quad b = b'$	$\mathrm{img}\ f \quad \subseteq \quad id$	(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 $f(a)$.

Functions, relationally

(The following properties of any function f are **extremely** useful.)

Shunting rules:

$$f \cdot R \subseteq S \quad \equiv \quad R \subseteq f^\circ \cdot S \tag{32}$$

$$R \cdot f^\circ \subseteq S \quad \equiv \quad R \subseteq S \cdot f \tag{33}$$

Equality rule:

$$f \subseteq g \;\equiv\; f = g \;\equiv\; f \supseteq g \tag{34}$$

Rule (34) follows from (32,33) by “cyclic inclusion” (next slide).

Proof of functional equality rule (34)

$$
\begin{array}{ll}
 & f \subseteq g \\
\equiv & \qquad \{ \text{ identity } \} \\
 & f \cdot id \subseteq g \\
\equiv & \qquad \{ \text{ shunting on } f \ \} \\
 & id \subseteq f^{\circ} \cdot g \\
\equiv & \qquad \{ \text{ shunting on } g \ \} \\
 & id \cdot g^{\circ} \subseteq f^{\circ} \\
\equiv & \qquad \{ \text{ converses; identity } \} \\
 & g \subseteq f
\end{array}
$$

Then:

$$
\begin{array}{ll}
 & f = g \\
\equiv & \qquad \{ \text{ cyclic inclusion (11)} \\
 & f \subseteq g \wedge g \subseteq f \\
\equiv & \qquad \{ \text{ aside } \} \\
 & f \subseteq g \\
\equiv & \qquad \{ \text{ aside } \} \\
 & g \subseteq f \\
\Box &
\end{array}
$$

Dividing functions

$$\frac{f}{g} = g^\circ \cdot f \qquad cf. \qquad \begin{tikzcd} B \arrow[dr, "g"'] & & A \arrow[ll, "\frac{f}{g}"'] \arrow[dl, "f"] \\ & C & \end{tikzcd} \tag{35}$$

Exercise 13: Check the properties:

$$\frac{f}{id} = f \tag{36}$$

$$\frac{f}{f} = \mathrm{ker}\ f \tag{38}$$

$$\frac{f \cdot h}{g \cdot k} = k^\circ \cdot \frac{f}{g} \cdot h \tag{37}$$

$$\left(\frac{f}{g}\right)^\circ = \frac{g}{f} \tag{39}$$

□

Exercise 14: Infer $id \subseteq \mathrm{ker}\ f$ (f is total) and $\mathrm{img}\ f \subseteq id$ (f is simple) from the shunting rules (32) or (33). □

Dividing functions

By (23) we have:

$$b \,\frac{f}{g}\, a \quad \equiv \quad g\ b = f\ a \tag{40}$$

How useful is this? Think of the following sentence:

Mary lives where John was born.

By (40), this can be expressed by a division:

$$\mathit{Mary}\ \frac{\mathit{birthplace}}{\mathit{residence}}\ \mathit{John} \ \equiv\ \mathit{residence}\ \mathit{Mary} = \mathit{birthplace}\ \mathit{John}$$

In general,

$b\,\frac{f}{g}\,a$ *means "the g of b is the f of a".*

Class 4 — On Endo-Relations

Endo-relations

A relation $A \xrightarrow{R} A$ whose input and output types coincide is called an

---

endo-relation.

This special case of relation is gifted with an extra **taxonomy** and many **applications**.

We have already seen them: $\mathrm{ker}\ R$ and $\mathrm{img}\ R$ are **endo-relations**.

Graphs, orders, the identity, equivalences and so on are all **endo-relations** as well.

Taxonomy of endo-relations

Besides

reflexive: iff $id \subseteq R$ (41)

coreflexive: iff $R \subseteq id$ (42)

an endo-relation $A \xleftarrow{R} A$ can be

transitive: iff $R \cdot R \subseteq R$ (43)

symmetric: iff $R \subseteq R^\circ (\equiv R = R^\circ)$ (44)

anti-symmetric: iff $R \cap R^\circ \subseteq id$ (45)

irreflexive: iff $R \cap id = \bot$

connected: iff $R \cup R^\circ = \top$ (46)

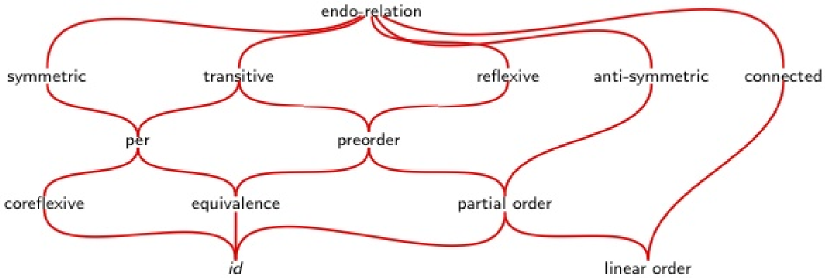
where, in general, for R, S of the same type:

$$b\ (R \cap S)\ a \quad \equiv \quad b\ R\ a \wedge b\ S\ a \tag{47}$$

$$b\ (R \cup S)\ a \quad \equiv \quad b\ R\ a \vee b\ S\ a \tag{48}$$

Taxonomy of endo-relations

Combining these criteria, endo-relations $A \xleftarrow{R} A$ can further be classified as



Taxonomy of endo-relations

In summary:

- **Preorders** are reflexive and transitive orders.
  Example: $age\ y \leqslant age\ x$.
- **Partial** orders are anti-symmetric preorders
  Example: $y \subseteq x$ where x and y are sets.
- **Linear** orders are connected partial orders
  Example: $y \leqslant x$ in $\mathbb{N}$
- **Equivalences** are symmetric preorders
  Example: $age\ y = age\ x$. [3]
- **Pers** are partial equivalences
  Example: $y\ \mathit{IsBrotherOf}\ x$.

[3] Kernels of functions are always equivalence relations, see exercise 21.

Exercises

Exercise 15: Consider the relation

$$b \; R \; a \;\; \equiv \;\; \textit{team } b \textit{ is playing against team } a \textit{ at the moment}$$

Is this relation: reflexive? irreflexive? transitive? anti-symmetric? symmetric? connected? □

Exercise 16: Check which of the following properties,

transitive, symmetric, anti-symmetric, connected

hold for the relation *Eats* of exercise 9. □

Exercises

Exercise 17: A relation R is said to be **co-transitive** or **dense** iff the following holds:

$$\langle \forall\ b, a\ :\ b\ R\ a :\ \langle \exists\ c\ :\ b\ R\ c :\ c\ R\ a \rangle\rangle \tag{49}$$

Write the formula above in PF notation. Find a relation (eg. over numbers) which is co-transitive and another which is not. □

Exercise 18: Expand criteria (43) to (46) to pointwise notation. □

Exercises

Exercise 19: The teams (T) of a football league play games (G) at home or away, and every game takes place in some date:

$$\begin{tikzcd}
T & G \arrow[l, "home"'] \arrow[r, "away"] \arrow[d, "date"'] & T \\
 & D &
\end{tikzcd}$$

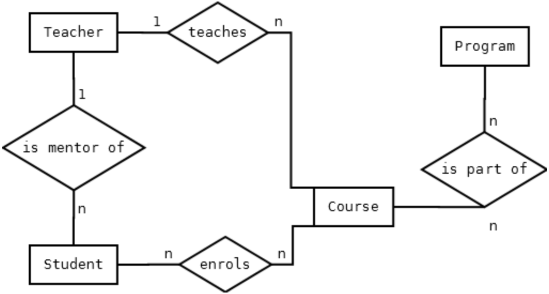
Moreover, *(a) No team can play two games on the same date; (b) All teams play against each other but not against themselves; (c) For each home game there is another game away involving the same two teams.* Show that

$$id \subseteq \frac{away}{home} \cdot \frac{away}{home} \tag{50}$$

captures one of the requirements above (which?) and that (50) amounts to forcing $home \cdot away^\circ$ to be symmetric. □

Formalizing ER diagrams

So-called “**Entity-Relationship**” (ER) diagrams are commonly used to capture relational information, e.g.[4]



ER-diagrams can be **formalized** in $A \xrightarrow{R} B$ notation, see e.g. the following relational algebra (RA) diagram.

[4] Credits: `https://dba.stackexchange.com/questions`.

Exercise

$$\begin{tikzcd}
Teacher & Program \arrow[d, "IsPartOf"] \\
Student \arrow[u, "isMentorOf"] & Course \arrow[ul, "teaches"'] \arrow[l, "Enrols"]
\end{tikzcd} \tag{51}$$

Exercise 20: Looking at diagram (51),

- Specify the property: *mentors of students necessarily are among their teachers*.
- Specify the relation R between students and teachers such that $t\ R\ s$ means: t *is the mentor of* s *and also teaches one of her/his courses*.

□

Meet and join

Recall **meet** (intersection) and **join** (union), introduced by (47) and (48), respectively.

They lift pointwise conjunction and disjunction, respectively, to the pointfree level.

Their meaning is nicely captured by the following **universal** properties:

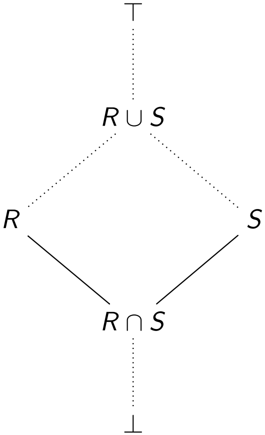
$$X \subseteq R \cap S \equiv X \subseteq R \wedge X \subseteq S \tag{52}$$

$$R \cup S \subseteq X \equiv R \subseteq X \wedge S \subseteq X \tag{53}$$

NB: recall the generic notions of **greatest lower bound** and **least upper bound**, respectively.

In summary

Type $B \longleftarrow A$ forms a lattice:



“top”

join, lub (“least upper bound”)

meet, glb (“greatest lower bound”)

“bottom”

How universal properties help

Using (52) i.e.

$$X \subseteq R \cap S \equiv \left\{ \begin{array}{lcl} X & \subseteq & R \\ X & \subseteq & S \end{array} \right.$$

as example, similarly for (53).

Cancellation

($X := R \cap S$):

$$\left\{ \begin{array}{lcl} R \cap S & \subseteq & R \\ R \cap S & \subseteq & S \end{array} \right. \quad (54)$$

$R \cap \top = R$ why? Use indirect equality

$$\begin{array}{ll} & X \subseteq R \cap \top \\ \equiv & \quad \{ \text{ universal property } \} \\ & \left\{ \begin{array}{lcl} X & \subseteq & R \\ X & \subseteq & \top \end{array} \right. \\ \equiv & \quad \{ \ \top \text{ is above anything } \} \\ & X \subseteq R \\ :: & \quad \{ \text{ indirect equality } \} \\ & R \cap \top = R \end{array}$$

How universal properties help

Meet and join have other expected properties, e.g. **associativity**

$$(R \cap S) \cap T = R \cap (S \cap T)$$

again proved aside by indirect equality.

$$
\begin{array}{ll}
 & X \subseteq (R \cap S) \cap T \\
\equiv & \qquad \{ \text{ $\cap$-universal (52) twice } \} \\
 & (X \subseteq R \wedge X \subseteq S) \wedge X \subseteq T \\
\equiv & \qquad \{ \ \wedge \text{ is associative } \} \\
 & X \subseteq R \wedge (X \subseteq S \wedge X \subseteq T) \\
\equiv & \qquad \{ \text{ $\cap$-universal (52) twice } \} \\
 & X \subseteq R \cap (S \cap T) \\
:: & \qquad \{ \text{ indirection (13) } \} \\
 & (R \cap S) \cap T = R \cap (S \cap T) \\
\square &
\end{array}
$$

Distributivity

As we will prove later, **composition** distributes over **union**

$$R \cdot (S \cup T) \;=\; (R \cdot S) \cup (R \cdot T) \tag{55}$$

$$(S \cup T) \cdot R \;=\; (S \cdot R) \cup (T \cdot R) \tag{56}$$

while distributivity over **intersection** is side-conditioned:

$$(S \cap Q) \cdot R = (S \cdot R) \cap (Q \cdot R) \;\Leftarrow\; \left\{ \begin{array}{c} Q \cdot \mathrm{img}\ R \subseteq Q \\ \vee \\ S \cdot \mathrm{img}\ R \subseteq S \end{array} \right. \tag{57}$$

$$R \cdot (Q \cap S) = (R \cdot Q) \cap (R \cdot S) \;\Leftarrow\; \left\{ \begin{array}{c} (\mathrm{ker}\ R) \cdot Q \subseteq Q \\ \vee \\ (\mathrm{ker}\ R) \cdot S \subseteq S \end{array} \right. \tag{58}$$

Propositio de homine et capra et lvpo

Back to our running example, we specify:

Being at the same bank:

$$SameBank \;=\; \ker where \;=\; \frac{where}{where}$$

Risk of somebody eating somebody else:

$$CanEat \;=\; SameBank \cap Eats$$

Then

"Starvation" is ensured by $Farmer$ *present at the same bank:*

$$CanEat \;\subseteq\; SameBank \cdot \underline{Farmer} \tag{59}$$

Propositio de homine et capra et lvpo

By (32), “starvation” property (59) converts to:

$$\mathit{where} \cdot \mathit{CanEat} \quad \subseteq \quad \mathit{where} \cdot \underline{\mathit{Farmer}}$$

In this version, (59) can be depicted as a diagram:

$$\begin{tikzcd}
\mathit{Being} \arrow[d, "\mathit{where}"'] \arrow[dr, phantom, "\subseteq"] & \mathit{Being} \arrow[l, "\mathit{CanEat}"'] \arrow[d, "\underline{\mathit{Farmer}}"] \\
\mathit{Bank} & \mathit{Being} \arrow[l, "\mathit{where}"]
\end{tikzcd} \tag{60}$$

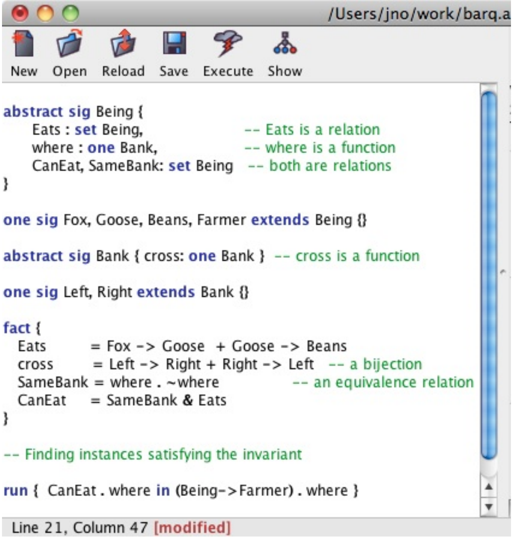
which “reads” in a nice way:

where (somebody) *CanEat* (somebody else) (that’s)
where (the) *Farmer* (is).

Propositio de homine et capra et lvpo

Properties which — such as (60) — are desirable and must **always hold** are called **invariants**.

See aside the ‘starvation’ invariant (60) written in **Alloy**.



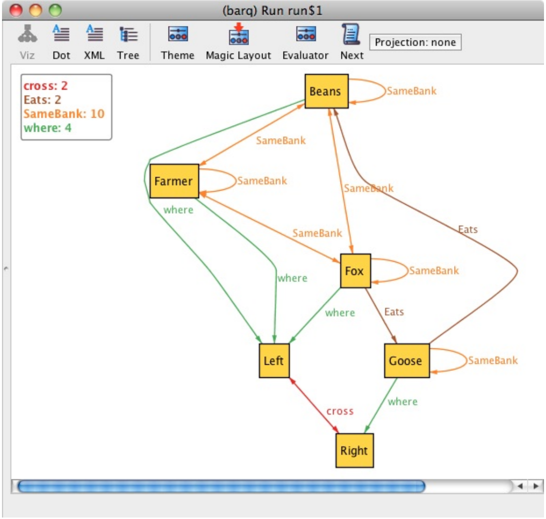
```
abstract sig Being {
    Eats : set Being,                    -- Eats is a relation
    where : one Bank,                    -- where is a function
    CanEat, SameBank: set Being          -- both are relations
}

one sig Fox, Goose, Beans, Farmer extends Being {}

abstract sig Bank { cross: one Bank }  -- cross is a function

one sig Left, Right extends Bank {}

fact {
  Eats       = Fox -> Goose  + Goose -> Beans
  cross      = Left -> Right + Right -> Left   -- a bijection
  SameBank = where . ~where              -- an equivalence relation
  CanEat     = SameBank & Eats
}

-- Finding instances satisfying the invariant

run {  CanEat . where in (Being->Farmer) . where }
```

Line 21, Column 47 [modified]

Propositio de homine et capra et lvpo

Carefully observe instance of such an invariant (aside):

- *SameBank* is an **equivalence** — exactly the **kernel** of *where*
- *Eats* is simple but not transitive
- *cross* is a **bijection**
- *CanEat* is empty
- etc


(barq) Run run$1

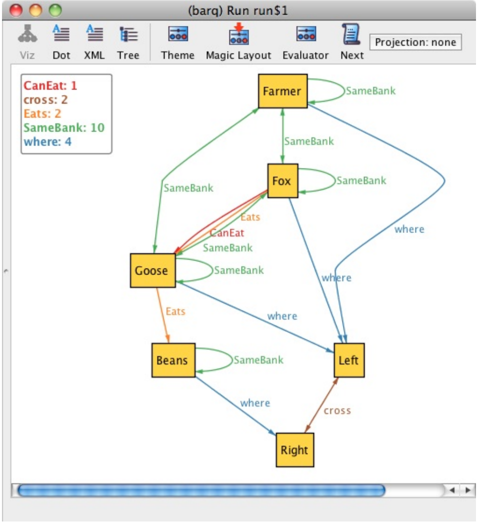
Projection: none

cross: 2
Eats: 2
SameBank: 10
where: 4

PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Another instance of the same invariant, in which:

- *CanEat* is **not** empty (*Fox* can eat *Goose*!)
- but *Farmer* is on the same bank :-)



Why is *SameBank* an equivalence?

Recall that $\mathit{SameBank} = \ker\ \mathit{where}$. Then $\mathit{SameBank}$ is an **equivalence relation** by the exercise below.

Exercise 21: Knowing that property

$$f \cdot f^{\circ} \cdot f = f \tag{61}$$

holds for every function f, prove that $\ker f = \frac{f}{f}$ (38) is an **equivalence** relation. □

Equivalence relations expressed in this way are captured in natural language by the textual pattern

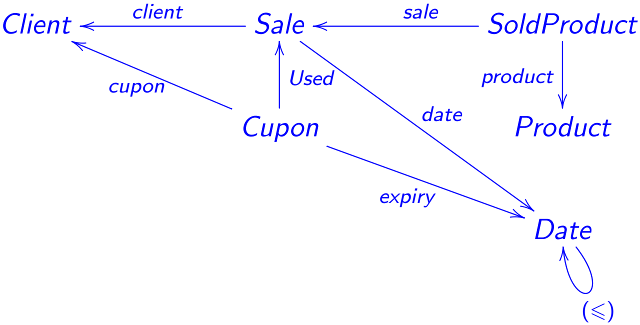
---

$a(\ker f)b$ *means* *"a and b have the same f"*

---

which is very common in requirements.

“D. Acácia grocery”



Specify the property:

Coupons cannot be used beyond their expiry date.

"D. Acácia grocery"

$$\begin{tikzcd}
Client & Sale \arrow[l, "client"'] \arrow[ddr, "date"] & SoldProduct \arrow[l, "sale"'] \arrow[d, "product"'] \\
 & Cupon \arrow[u, "Used"'] \arrow[ul, "cupon"] \arrow[dr, "expiry"'] & Product \\
 & & Date \arrow[loop below, "(\leqslant)"]
\end{tikzcd}$$

Specify the property:

Coupons can only be used by clients who own them.

Class 5 — Design patterns (relationally)

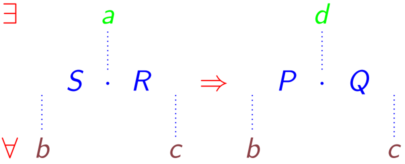
Patterns in diagrams

Recall

$$\begin{tikzcd} A \arrow[d, "S"'] \arrow[dr, phantom, "\subseteq"] & C \arrow[l, "R"'] \arrow[d, "Q"] \\ B & D \arrow[l, "P"] \end{tikzcd}$$

$$S \cdot R \subseteq P \cdot Q$$

... i.e. the pointwise:



Patterns in diagrams

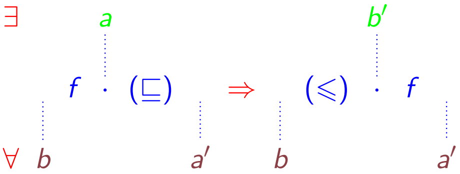
Now consider the special case

$$\begin{tikzcd} A \arrow[d, "f"'] & A \arrow[l, "(\sqsubseteq)"'] \arrow[d, "f"] \\ B \arrow[ur, phantom, "\subseteq"] & B \arrow[l, "(\leqslant)"] \end{tikzcd} \qquad f \cdot (\sqsubseteq) \subseteq (\leqslant) \cdot f$$

where $(\sqsubseteq)$ and $(\leqslant)$ are preorders.

Patterns in diagrams

Do we need...



as before?

Patterns in diagrams

No — for **functions** things are much easier:

$$\begin{array}{cl} & f \cdot (\sqsubseteq) \;\subseteq\; (\leqslant) \cdot f \\ \equiv & \qquad \{ \;\; (32) \;\; \} \\ & (\sqsubseteq) \;\subseteq\; f^{\circ} \cdot (\leqslant) \cdot f \\ \equiv & \qquad \{ \;\; (23) \;\; \} \\ & \langle \forall\, a, a' \;:\; a \sqsubseteq a' \;:\; f\ a \leqslant f\ a' \rangle \end{array}$$

In summary,

$$f \cdot (\sqsubseteq) \;\subseteq\; (\leqslant) \cdot f \qquad (62)$$

states that f is a **monotonic** function.

Patterns in diagrams

Now consider yet another special case:

$$\begin{tikzcd} A \arrow[d, "f"'] \arrow[dr, phantom, "\subseteq"] & A \arrow[l, "id"'] \arrow[d, "g"] \\ B & B \arrow[l, "(\leqslant)"] \end{tikzcd}$$

$$f \subseteq (\leqslant) \cdot g$$

Likewise, $f \subseteq (\leqslant) \cdot g$ will unfold to

$$\langle \forall\ a\ ::\ f\ a \leqslant g\ a \rangle$$

meaning that

f is **pointwise-smaller** *than g wrt.* $(\leqslant)$.

Patterns in diagrams

Now consider yet another special case:

$$\begin{tikzcd} A \arrow[d, "f"'] \arrow[dr, phantom, "\subseteq"] & A \arrow[l, "id"'] \arrow[d, "g"] \\ B & B \arrow[l, "(\leqslant)"] \end{tikzcd}$$

$f \subseteq (\leqslant) \cdot g$

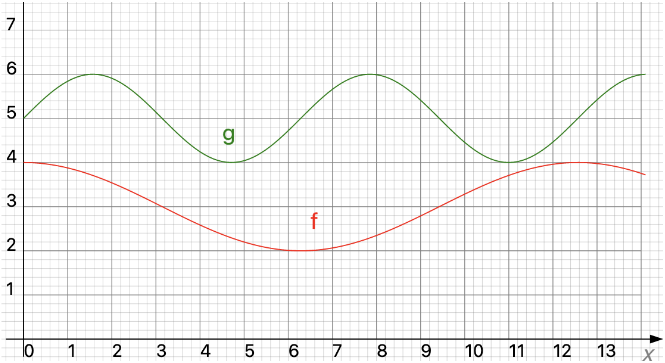
Likewise, $f \subseteq (\leqslant) \cdot g$ will unfold to

$$\langle \forall\ a\ ::\ f\ a \leqslant g\ a \rangle$$

meaning that

f *is* **pointwise-smaller** *than* g *wrt.* $(\leqslant)$.

$$f \ \dot{\leqslant}\ g$$



Usual abbreviation: $f \ \dot{\leqslant}\ g \ \equiv \ f \subseteq (\leqslant) \cdot g$.

Relational patterns: the pre-order $f^\circ \cdot (\leqslant) \cdot f$

Given a **preorder** $(\leqslant)$, a function f function taking values on the carrier set of $(\leqslant)$, define

$$(\leqslant_f) = f^\circ \cdot (\leqslant) \cdot f$$

It is easy to show that:

$$b \ \leqslant_f \ a \ \equiv \ (f \ b) \leqslant (f \ a)$$

That is, we compare **objects** a and b with respect to their **attribute** f.

Exercise 22:

1. Show that $(\leqslant_f)$ is a **preorder.**
2. Show that $(\leqslant_f)$ is not (in general) a total order even in the case $(\leqslant)$ is so.

□

Exercises

Exercise 23: Show that $1 \xleftarrow{\top} 1 = 1 \xleftarrow{!} 1 = id$. □

Exercise 24: As generalization of exercise 1, draw the most general type diagram that accommodates relational assertion:

$$M \cdot R^{\circ} \subseteq \top \cdot M \tag{63}$$

□

Exercise 25: Type the following relational assertions

$$M \cdot N^{\circ} \subseteq \bot \tag{64}$$

$$M \cdot N^{\circ} \subseteq id \tag{65}$$

$$M^{\circ} \cdot \top \cdot N \subseteq > \tag{66}$$

and check their pointwise meaning. Confirm your intuitions by repeating this exercise in Alloy. □

Exercise

Exercise 26: Let $bag : A^* \to \mathbb{N}^A$ be the function that, given a finite sequence (list) indicates the number of occurrences of its elements, for instance,

$$
\begin{array}{l}
bag\ [a, b, a, c]\ a = 2 \\
bag\ [a, b, a, c]\ b = 1 \\
bag\ [a, b, a, c]\ c = 1
\end{array}
$$

Let $ordered : A^* \to \mathbb{B}$ be the obvious predicate assuming a **total** order predefined in A. Finally, let $true = \underline{\mathsf{True}}$. Having defined

$$S = \frac{bag}{bag} \cap \frac{true}{ordered} \tag{67}$$

identify the type of S and, going pointwise and simplifying, tell which operation is specified by S. □

Monotonicity

All relational combinators studied so far are $\subseteq$-monotonic, namely:

$$R \subseteq S \quad \Rightarrow \quad R^\circ \subseteq S^\circ \tag{68}$$

$$R \subseteq S \wedge U \subseteq V \quad \Rightarrow \quad R \cdot U \subseteq S \cdot V \tag{69}$$

$$R \subseteq S \wedge U \subseteq V \quad \Rightarrow \quad R \cap U \subseteq S \cap V \tag{70}$$

$$R \subseteq S \wedge U \subseteq V \quad \Rightarrow \quad R \cup U \subseteq S \cup V \tag{71}$$

etc hold.

Exercise 27: Prove the **union simplicity** rule:

$$M \cup N \text{ is simple} \quad \equiv \quad M,\ N \text{ are simple and } M \cdot N^\circ \subseteq id \tag{72}$$

Derive from (72) the corresponding rule for **injective** relations. □

Proofs by $\subseteq$-transitivity

Wishing to prove $R \subseteq S$, the following rules are of help by relying on a "mid-point" M (analogy with interval arithmetics):

- Rule A: **lowering the upper side**

$$
\begin{array}{ll}
& R \subseteq S \\
\Leftarrow & \qquad \{ \ M \subseteq S \text{ is known ; transitivity of } \subseteq (10) \ \} \\
& R \subseteq M
\end{array}
$$

  and then proceed with $R \subseteq M$.

- Rule B: **raising the lower side**

$$
\begin{array}{ll}
& R \subseteq S \\
\Leftarrow & \qquad \{ \ R \subseteq M \text{ is known; transitivity of } \subseteq \ \} \\
& M \subseteq S
\end{array}
$$

  and then proceed with $M \subseteq S$.

Example

Proof of shunting rule (32):

$$
\begin{array}{ll}
 & R \subseteq f^\circ \cdot S \\
\Leftarrow & \quad \{ \ id \subseteq f^\circ \cdot f \ ; \text{raising the lower-side} \ \} \\
 & f^\circ \cdot f \cdot R \subseteq f^\circ \cdot S \\
\Leftarrow & \quad \{ \ \text{monotonicity of } (f^\circ \cdot) \ \} \\
 & f \cdot R \subseteq S \\
\Leftarrow & \quad \{ \ f \cdot f^\circ \subseteq id \ ; \text{lowering the upper-side} \ \} \\
 & f \cdot R \subseteq f \cdot f^\circ \cdot S \\
\Leftarrow & \quad \{ \ \text{monotonicity of } (f \cdot) \ \} \\
 & R \subseteq f^\circ \cdot S
\end{array}
$$

Thus the equivalence in (32) is established by circular implication.

Exercises (monotonicity and transitivity)

Exercise 28: Prove the following rules of thumb:

- **smaller** than injective (simple) is injective (simple)
- **larger** than entire (surjective) is entire (surjective)
- $R \cap S$ is injective (simple) provided one of R or S is so
- $R \cup S$ is entire (surjective) provided one of R or S is so.

□

Exercise 29: Prove that relational **composition** preserves **all** relational classes in the taxonomy of (30). □

Meaning of $f \cdot r = id$

On the one hand,

$$
\begin{array}{ll}
& f \cdot r = id \\
\equiv & \qquad \{ \text{ equality of functions } \} \\
& f \cdot r \subseteq id \\
\equiv & \qquad \{ \text{ shunting } \} \\
& r \subseteq f^\circ
\end{array}
$$

Since f is simple:

- f° is injective
- and so is r, because "smaller than injective is injective".

Meaning of $f \cdot r = id$

On the other hand,

$$
\begin{array}{ll}
& f \cdot r = id \\
\equiv & \qquad \{ \text{ equality of functions } \} \\
& id \subseteq f \cdot r \\
\equiv & \qquad \{ \text{ shunting } \} \\
& r^\circ \subseteq f
\end{array}
$$

Since r is entire:

- r° is surjective
- and so is f because “larger that surjective is surjective”.

Meaning of $f \cdot r = id$

We conclude that

*f is **surjective** and r is **injective** wherever $f \cdot r = id$ holds.*

Since both are functions, we furthermore conclude that

*f is an **abstraction** and r is a **representation***

Exercise 30: Why are π_1 and π_2 **surjective** and i_1 and i_2 **injective**? Why are isomorphisms **bijections**? □

Class 6 — Pairs and sums

Relational pairing

Recall:

$$\begin{tikzcd} A & A \times B \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & B \\ & C \arrow[ul, "f"] \arrow[u, "\langle f,g \rangle"] \arrow[ur, "g"'] & \end{tikzcd}$$

$$\langle f,g\rangle\, c = (f\ c, g\ c) \qquad (73)$$

Clearly:

$$
\begin{array}{ll}
 & (a,b) = \langle f,g\rangle\ c \\
\equiv & \quad \{\ \langle f,g\rangle\ c = (f\ c, g\ c)\ (73)\ ;\ \text{equality of pairs}\ \} \\
 & \left\{\begin{array}{l} a = f\ c \\ b = g\ c \end{array}\right. \\
\equiv & \quad \{\ y = f\ x\ \equiv\ y\ f\ x\ \} \\
 & \left\{\begin{array}{l} a\ f\ c \\ b\ g\ c \end{array}\right.
\end{array}
$$

Relational pairing

That is:

$$(a, b)\ \langle f, g\rangle\ c\ \equiv\ a\ f\ c \wedge b\ g\ c$$

This suggests the generalization

$$(a, b)\ \langle R, S\rangle\ c\ \equiv\ a\ R\ c \wedge b\ S\ c \tag{74}$$

from which one immediately derives the ('Kronecker') **product**:

$$R \times S = \langle R \cdot \pi_1, S \cdot \pi_2\rangle \tag{75}$$

(75) unfolds to the pointwise:

$$(b, d)(R \times S)(a, c) \quad \equiv \quad b\ R\ a \wedge d\ S\ c \tag{76}$$

Relational pairing example (in matrix layout)

Example — given relations

$$\mathit{where}^\circ \quad = \quad \begin{array}{r|cc} & \mathit{Left} & \mathit{Right} \\ \hline \mathit{Fox} & 1 & 0 \\ \mathit{Goose} & 0 & 1 \\ \mathit{Beans} & 0 & 1 \end{array} \qquad \mathit{and} \qquad \mathit{cross} \quad = \quad \begin{array}{r|cc} & \mathit{Left} & \mathit{Right} \\ \hline \mathit{Left} & 0 & 1 \\ \mathit{Right} & 1 & 0 \end{array}$$

pairing them up evaluates to:

$$\langle \mathit{where}^\circ, \mathit{cross} \rangle \quad = \quad \begin{array}{r|cc} & \mathit{Left} & \mathit{Right} \\ \hline (\mathit{Fox}, \mathit{Left}) & 0 & 0 \\ (\mathit{Fox}, \mathit{Right}) & 1 & 0 \\ (\mathit{Goose}, \mathit{Left}) & 0 & 1 \\ (\mathit{Goose}, \mathit{Right}) & 0 & 0 \\ (\mathit{Beans}, \mathit{Left}) & 0 & 1 \\ (\mathit{Beans}, \mathit{Right}) & 0 & 0 \end{array}$$

Exercises

Exercise 31: Show that

$$(b,c)\langle R,S\rangle a \quad\equiv\quad b\ R\ a \wedge c\ S\ a$$

PF-transforms to:

$$\langle R,S\rangle \;=\; \pi_1^\circ \cdot R \cap \pi_2^\circ \cdot S \tag{77}$$

Then infer universal property

$$X \subseteq \langle R,S\rangle \quad\equiv\quad \pi_1 \cdot X \subseteq R \;\wedge\; \pi_2 \cdot X \subseteq S \tag{78}$$

from (77) via indirect equality (13). □

Exercise 32: What can you say about (78) in case X, R and S are functions? □

Exercises

Exercise 33: Unconditional distribution laws

$$\begin{array}{rcl} (P \cap Q) \cdot S & = & (P \cdot S) \cap (Q \cdot S) \\ R \cdot (P \cap Q) & = & (R \cdot P) \cap (R \cdot Q) \end{array}$$

will hold provide one of R or S is simple and the other injective. Tell which (justifying). □

Exercise 34: Derive from

$$\langle R, S\rangle^{\circ} \cdot \langle X, Y\rangle \quad = \quad (R^{\circ} \cdot X) \cap (S^{\circ} \cdot Y) \tag{79}$$

the following properties:

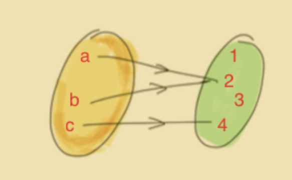
□
$$\ker \langle R, S\rangle \ = \ \ker R \cap \ker S \tag{80}$$

Injectivity preorder

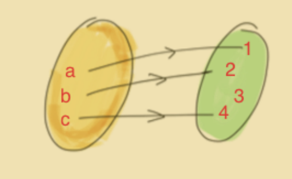
$\ker R = R^\circ \cdot R$ *measures* the level of **injectivity** of R according to the preorder $(\leqslant)$ defined by

$$R \leqslant S \quad \equiv \quad \ker S \subseteq \ker R \tag{81}$$

telling that R is *less injective* or *more defined* (entire) than S — for instance:



$\leqslant$



Injectivity preorder

Restricted to *functions*, $(\leqslant)$ is *universally* bounded by

$$! \leqslant f \leqslant id$$

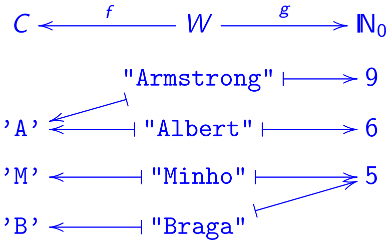
Also easy to show:

$$id \leqslant f \quad \equiv \quad f \text{ is injective} \tag{82}$$

Exercise 35: Let f and g be the two functions depicted on the right.

Check the assertions:

1. $f \leqslant g$
2. $g \leqslant f$
3. Both hold
4. None holds.



$$\begin{tikzcd} C & W \arrow[l, "f"'] \arrow[r, "g"] & \mathbb{N}_0 \end{tikzcd}$$

□

The specification pattern $h \leqslant \langle f, g \rangle$

As illustration of the use of this ordering in formal specification, suppose one writes

$$room \leqslant \langle lect, slot \rangle$$

in the context of the data model

$$\begin{tikzcd}
Teacher & Class \arrow[l, "lect"'] \arrow[r, "room"] \arrow[d, "slot"'] & Room \\
& TD &
\end{tikzcd}$$

where TD abbreviates time and date.

The specification pattern $h \leqslant \langle f, g \rangle$

What are we telling about this model by writing

$room \leqslant \langle lect, slot \rangle$?

Unfolding it:

$$
\begin{array}{cl}
 & room \leqslant \langle lect, slot \rangle \\
\equiv & \{\ (81)\ \} \\
 & \ker \langle lect, slot \rangle \ \subseteq\ \ker room \\
\equiv & \{\ (80)\ ;\ (38)\ \} \\
 & \frac{lect}{lect} \cap \frac{slot}{slot} \ \subseteq\ \frac{room}{room} \\
\equiv & \{\ \text{going pointwise, for all } c_1, c_2 \in Class\ \} \\
 & \left\{ \begin{array}{l} lect\ c_1 = lect\ c_2 \\ slot\ c_1 = slot\ c_2 \end{array} \right. \Rightarrow room\ c_1 = room\ c_2
\end{array}
$$

The specification pattern $h \leqslant \langle f, g \rangle$

That is, $room \leqslant \langle lect, slot \rangle$ imposes that

> *a given lecturer cannot be in two different rooms at the same time.*

(Think of c_1 and c_2 as classes shared by different courses, possibly of different degrees.)

In the standard terminology of database theory this is called a **functional dependency**, meaning that:

- *room* is **dependent** on *lect* and *slot*, i.e.
- *lect* and *slot* **determine** *room*.

Generalization: the “agenda design pattern”

Nobody can be in different places at the same time

$$\mathit{where} \leqslant \langle \mathit{who}, \mathit{when} \rangle$$

in the context of the generic data model:

$$\begin{tikzcd}
\mathit{Who} & \mathit{Meeting} \arrow[l, "\mathit{who}"'] \arrow[r, "\mathit{where}"] \arrow[d, "\mathit{when}"'] & \mathit{Where} \\
& \mathit{When} &
\end{tikzcd}$$

Exercise 36: Do $\mathit{who} \leqslant \langle \mathit{where}, \mathit{when} \rangle$ and $\mathit{when} \leqslant \langle \mathit{who}, \mathit{where} \rangle$ express reasonable facts? □

The specification pattern $h \leqslant \langle f, g \rangle$

Let $h := id$ in this pattern:

Two functions f and g are said to be **complementary** *wherever $id \leqslant \langle f, g \rangle$.*

For instance:

π_1 and π_2 are complementary since $\langle \pi_1, \pi_2 \rangle = id$ by $\times$-reflection.

Informal interpretation:

Non-injective *f and g compensate each other's lack of injectivity so that their pairing is* **injective**.

Universal property

$$\langle R, S\rangle \leqslant X \quad \equiv \quad R \leqslant X \wedge S \leqslant X \tag{83}$$

Cancellation of (83) means that *pairing* always *increases injectivity*:

$$R \leqslant \langle R, S\rangle \quad \text{and} \quad S \leqslant \langle R, S\rangle . \tag{84}$$

(84) unfolds to $\ker \langle R, S\rangle \ \subseteq \ (\ker R) \cap (\ker S)$, confirming (80).

Injectivity **shunting law**:

$$R \cdot g \leqslant S \quad \equiv \quad R \leqslant S \cdot g^{\circ} \tag{85}$$

Exercise 37: $\langle R, id\rangle$ is always *injective* — why? □

Relation pairing continued

The **fusion**-law of relation pairing requires a side condition:

$$\begin{array}{l} \langle R, S\rangle \cdot T = \langle R \cdot T, S \cdot T\rangle \\ \qquad \Leftarrow \; R \cdot (\mathrm{img}\ T) \subseteq R \lor S \cdot (\mathrm{img}\ T) \subseteq S \end{array} \tag{86}$$

The **absorption** law

$$(R \times S) \cdot \langle P, Q\rangle = \langle R \cdot P, S \cdot Q\rangle \tag{87}$$

holds unconditionally.

Exercises

Exercise 38: Recalling (31), prove that

$$swap = \langle \pi_2, \pi_1 \rangle \tag{88}$$

is a bijection. (Assume property $(R \cap S)^\circ = R^\circ \cap S^\circ$.) □

Exercise 39: Derive from the laws of pairing studied thus far the following facts about relational product:

$$id \times id = id \tag{89}$$

$$(R \times S) \cdot (P \times Q) = (R \cdot P) \times (S \cdot Q) \tag{90}$$

□

Exercise 40: Show that (86) holds. Suggestion: recall (57). From this infer that no side-condition is required for T simple. □

Exercises

Exercise 41:

Consider the adjacency relation A defined by clauses:
(a) A is symmetric;
(b) $id \times (1+) \cup (1+) \times id \;\subseteq\; A$

	$(y+1, x)$	
$(y, x-1)$	(y, x)	$(y, x+1)$
	$(y-1, x)$	

Show that A is **neither** transitive nor reflexive.

NB: consider $(1+) : \mathbb{Z} \to \mathbb{Z}$ a bijection, i.e. $\mathsf{pred} = (1+)^{\circ}$ is a function.

□

Relational sums

Example (Haskell):

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

PF-transforms to

$$\begin{tikzcd}
Bool \arrow[r, "i_1"] \arrow[dr, "Boo"'] & Bool + String \arrow[d, "{[Boo\ ,Err]}"] & String \arrow[l, "i_2"'] \arrow[dl, "Err"] \\
& X &
\end{tikzcd} \qquad (91)$$

where

$$[R\ ,S] \;=\; (R \cdot i_1^\circ) \cup (S \cdot i_2^\circ) \qquad \text{cf.} \qquad \begin{tikzcd}
A \arrow[r, "i_1"] \arrow[dr, "R"'] & A + B \arrow[d, "{[R\ ,S]}"] & B \arrow[l, "i_2"'] \arrow[dl, "S"] \\
& C &
\end{tikzcd}$$

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

Relational sums

From $[R\ , S]\ =\ (R \cdot i_1^\circ) \cup (S \cdot i_2^\circ)$ above one easily infers, by indirect equality,

$$[R\ , S] \subseteq X \quad \equiv \quad R \subseteq X \cdot i_1\ \wedge\ S \subseteq X \cdot i_2$$

(check this).

It turns out that inclusion can be strengthened to equality, and therefore **relational coproducts** have exactly the same properties as functional ones, stemming from the universal property:

$$[R\ , S] = X \quad \equiv \quad R = X \cdot i_1\ \wedge\ S = X \cdot i_2 \tag{92}$$

Thus $[i_1\ , i_2] = id$ — solve (92) for R and S when $X = id$, etc etc.

Divide and conquer

The property for sums (coproducts) corresponding to (79) for products is:

$$[R\ ,S]\cdot[T\ ,U]^\circ \quad = \quad (R\cdot T^\circ)\cup(S\cdot U^\circ) \tag{93}$$

NB: This *divide-and-conquer* rule is essential to **parallelizing** relation composition by **block** decomposition.

Exercise 42: Show that:

$$\mathrm{img}\ [R\ ,S] \quad = \quad \mathrm{img}\ R\cup\mathrm{img}\ S \tag{94}$$

$$\mathrm{img}\ i_1\cup\mathrm{img}\ i_2 \quad = \quad id \tag{95}$$

□

Exercises

Exercise 43: The type declaration

$$\textbf{data}\ \mathit{Maybe}\ a = \mathsf{Nothing} \mid \mathsf{Just}\ a$$

in Haskell corresponds, as is known, to the declaration of the isomorphism:

$$\begin{array}{l} \mathsf{in} : 1 + A \rightarrow \mathit{Maybe}\ A \\ \mathsf{in} = [\underline{\mathsf{Nothing}}\ , \mathsf{Just}] \end{array}$$

Show that the relation

$$R = i_1 \cdot \underline{\mathsf{Nothing}}^{\circ} \cup i_2 \cdot \mathsf{Just}^{\circ}$$

is a function. □

Exercises

Exercise 44: Consider the following definition of a relation $A \xleftarrow{R} A^*$,

$$R \cdot \mathsf{in} = [\bot\ , \pi_1 \cup R \cdot \pi_2]$$

where

$$\mathsf{in} = [\mathsf{nil}\ , \mathsf{cons}] \tag{96}$$

$$\mathsf{nil}\ \_ = [\,] \tag{97}$$

$$\mathsf{cons}\ (h, t) = h : t \tag{98}$$

(a) Rely on the co-product laws to derive (formally) the *pointwise* definition of R.

(b) Based on this, spell out the meaning of $a\ R\ x$ in you own words. □

$+$ meets $\times$

The **exchange law**

$$[\langle R, S\rangle , \langle T, V\rangle] \;=\; \langle [R , T], [S , V]\rangle \tag{99}$$

holds for all relations as in diagram

$$\begin{tikzcd}
A \arrow[r, "i_1"] \arrow[d, "R"'] \arrow[rrd, "S"] & A + B & B \arrow[l, "i_2"'] \arrow[d, "V"] \arrow[lld, "T"'] \\
C & C \times D \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & D
\end{tikzcd}$$

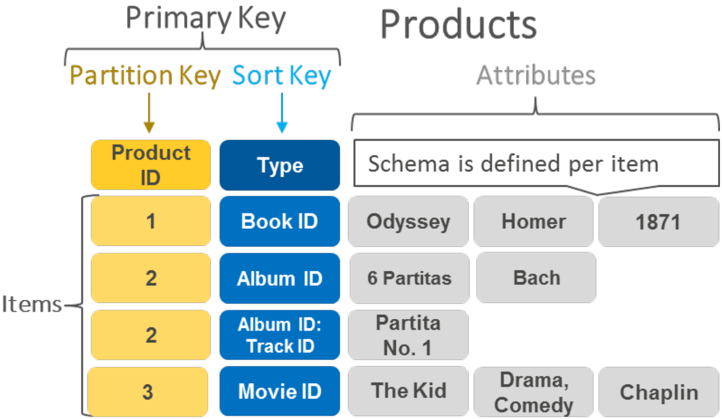
and the **fusion** law

$$\langle R, S\rangle \cdot f \;=\; \langle R \cdot f, S \cdot f\rangle \tag{100}$$

also holds, where f is a function. (Why?)

Exercise 45: Relying on both (92) and (100) prove (99). □

On key-value (KV) data models



Product ID	Type	Schema is defined per item		
1	Book ID	Odyssey	Homer	1871
2	Album ID	6 Partitas	Bach	
2	Album ID: Track ID	Partita No. 1		
3	Movie ID	The Kid	Drama, Comedy	Chaplin

On key-value data models

Simple relations abstract what is currently known as the **key-value-pair** (**KV**) data model in modern databases

E.g. Hbase, Amazon DynamoDB etc

In each such relation $K \xrightarrow{S} V$, K is said to be the **key** and V the **value**.

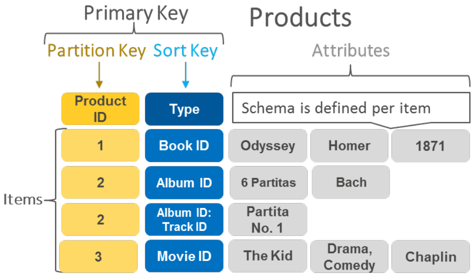
No-SQL, **columnar** *database trend.*

Example above:

$$\underbrace{PartitionKey \times SortKey}_{K} \to \underbrace{Type \times \ldots}_{V}$$

On key-value data models

"Schema is defined per item"...



Primary Key

Partition Key Sort Key

Products

Attributes

Items

Product ID	Type	Schema is defined per item		
1	Book ID	Odyssey	Homer	1871
2	Album ID	6 Partitas	Bach	
2	Album ID: Track ID	Partita No. 1		
3	Movie ID	The Kid	Drama, Comedy	Chaplin

In this example:

$$V = \mathit{Title} \times (1 + \mathit{Author} \times (1 + \mathit{Date} \times \ldots))$$

This shows the expressiveness of **products** and **coproducts** in data modelling.

Class 7 — Relational division

Relational division

In the same way

$$z \times y \leqslant x \;\equiv\; z \leqslant x \div y$$

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

$$Z \cdot Y \subseteq X \;\equiv\; Z \subseteq X / Y \qquad (101)$$

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

What is the pointwise meaning of X / Y?

We reason:

First, the types of

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

$$\begin{tikzcd}
 & A \arrow[dl, "X/Y"'] \\
C & B \arrow[l, "X"] \arrow[u, "Y"']
\end{tikzcd}$$

Next, the calculation:

$$\begin{array}{ll}
 & c\ (X/Y)\ a \\
\equiv & \qquad \{ \ \text{introduce points } C \xleftarrow{\underline{c}} 1 \text{ and } A \xleftarrow{\underline{a}} 1 \ \} \\
 & x(\underline{c}^{\circ} \cdot (X/Y) \cdot \underline{a})x \\
\equiv & \qquad \{ \ \text{one-point (200)} \ \} \\
 & x' = x \ \Rightarrow \ x'(\underline{c}^{\circ} \cdot (X/Y) \cdot \underline{a})x
\end{array}$$

Proceed by going pointfree:

We reason

$$
\begin{array}{ll}
 & id \ \subseteq \ \underline{c}^{\circ} \cdot (X / Y) \cdot \underline{a} \\
\equiv & \quad \{ \text{ shunting rules } \} \\
 & \underline{c} \cdot \underline{a}^{\circ} \ \subseteq \ X / Y \\
\equiv & \quad \{ \text{ universal property (101) } \} \\
 & \underline{c} \cdot \underline{a}^{\circ} \cdot Y \ \subseteq \ X \\
\equiv & \quad \{ \text{ now shunt } \underline{c} \text{ back to the right } \} \\
 & \underline{a}^{\circ} \cdot Y \ \subseteq \ \underline{c}^{\circ} \cdot X \\
\equiv & \quad \{ \text{ back to points via (23) } \} \\
 & \langle \forall \, b \ : \ a \ Y \ b : \ c \ X \ b \rangle
\end{array}
$$

Outcome

In summary:

$$c\ (X/Y)\ a \ \equiv\ \langle \forall\ b\ :\ a\ Y\ b :\ c\ X\ b \rangle \qquad (102)$$

$$\begin{tikzcd}
 & a \arrow[dl, maps to, "X/Y"'] \\
c & b \arrow[u, maps to, "Y"'] \arrow[l, maps to, "X"]
\end{tikzcd}$$

Example:

$a\ Y\ b$ = passenger a chooses 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.

Pattern $X \,/\, Y$

Informally, $c\ (X \,/\, Y)\ a$ captures the linguistic pattern

---

a **only** *Y those b's*
such that *c X b.*

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

For instance,

> **Students** *enrolled*
> *in* **courses** *only*
> *dealing with*
> *particular* **subjects**

$$\begin{tikzcd}
& student \arrow[dl, mapsto, "Dealt\_in/Enrolled"'] \\
subject & course \arrow[l, mapsto, "Dealt\_in"] \arrow[u, mapsto, "Enrolled"']
\end{tikzcd}$$

Pointwise meaning in full

The full pointwise encoding of

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

is:

$$\begin{array}{ll} & \langle \forall\ c, b \ :\ \langle \exists\ a \ :\ cZa :\ aYb \rangle :\ cXb \rangle \\ \equiv & \\ & \langle \forall\ c, a \ :\ cZa :\ \langle \forall\ b \ :\ aYb :\ cXb \rangle \rangle \end{array}$$

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 (208) of the Eindhoven calculus.

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

Exercises

Exercise 46: Prove the equalities

$$X \cdot f = X / f^{\circ} \tag{103}$$

$$X / \bot = \top \tag{104}$$

$$X / id = X \tag{105}$$

and check their pointwise meaning. □

Exercise 47: Define

$$X \setminus Y = (Y^{\circ} / X^{\circ})^{\circ} \tag{106}$$

and infer:

$$a(R \setminus S)c \equiv \langle \forall\, b : b\, R\, a : b\, S\, c \rangle \tag{107}$$

$$R \cdot X \subseteq Y \equiv X \subseteq R \setminus Y \tag{108}$$

□

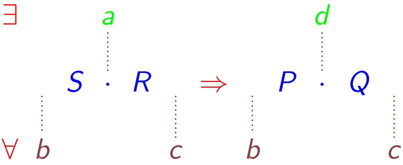
Patterns in diagrams (again!)

Back to our good old "rectangle":

$$\begin{tikzcd}
A \arrow[d, "S"'] \arrow[dr, phantom, "\subseteq"] & C \arrow[l, "R"'] \arrow[d, "Q"] \\
B & D \arrow[l, "P"]
\end{tikzcd}$$

$$S \cdot R \subseteq P \cdot Q$$

... i.e. the pointwise:



Patterns in diagrams - very special case

Again assuming two preorders $(\sqsubseteq)$ and $(\leqslant)$:

$$\begin{tikzcd}
A & A \arrow[l, "(\sqsubseteq)"'] \arrow[d, "g"] \\
B \arrow[from=u, "f^\circ"'] \arrow[ur, phantom, "="] & B \arrow[l, "(\leqslant)"]
\end{tikzcd}$$

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

$$f\ b \sqsubseteq a \ \equiv\ b \leqslant g\ a \qquad (109)$$

In this very special situation, f and g in

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

are said to be **Galois connected** (GC) and we write

$$f \vdash g \qquad (110)$$

Patterns in diagrams - very special case

Again assuming two preorders $(\sqsubseteq)$ and $(\leqslant)$:

$$\begin{tikzcd}
A & A \arrow[l, "(\sqsubseteq)"'] \arrow[d, "g"] \\
B & B \arrow[l, "(\leqslant)"]
\arrow[from=1-1, to=2-1, "f^{\circ}"']
\arrow[from=1-1, to=2-2, phantom, "="]
\end{tikzcd}$$

$$f^{\circ} \cdot (\sqsubseteq) = (\leqslant) \cdot g$$

$$f\ b \sqsubseteq a \ \equiv\ b \leqslant g\ a \qquad (109)$$

In this very special situation, f and g in

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

are said to be **Galois connected** (GC) and we write

$$f \vdash g \qquad (110)$$

Patterns in diagrams - even more special case

Preorders $(\sqsubseteq)$ and $(\leqslant)$ are the **identity**:

$$\begin{tikzcd}
A & A \arrow[l, "id"'] \arrow[d, "g"] \\
B & B \arrow[l, "id"] \\
\arrow[from=1-1, to=2-1, "f^{\circ}"'] \arrow[from=1-1, to=2-2, phantom, "="]
\end{tikzcd}$$

$$f^{\circ} = g$$

$$f\ b = a \ \equiv\ b = g\ a \qquad (111)$$

That is to say,

$$\begin{tikzcd}
A \arrow[r, bend left, "g"] \arrow[r, phantom, "\cong"] & B \arrow[l, bend left, "f"]
\end{tikzcd}$$

Isomorphisms *are special cases of* **Galois connections**.

Patterns in diagrams - even more special case

Preorders $(\sqsubseteq)$ and $(\leqslant)$ are the **identity**:

$$\begin{tikzcd}
A \arrow[d, "f^\circ"'] \arrow[dr, phantom, "="] & A \arrow[l, "id"'] \arrow[d, "g"] \\
B & B \arrow[l, "id"]
\end{tikzcd}$$

$$f^\circ = g$$

$$f\ b = a \ \equiv \ b = g\ a \qquad (111)$$

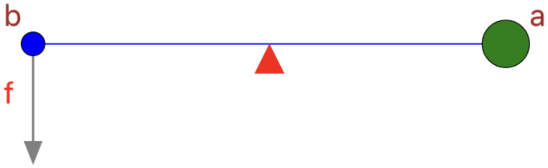
That is to say,

$$\begin{tikzcd}
A \arrow[r, bend left, "g"] \arrow[r, phantom, "\cong"] & B \arrow[l, bend left, "f"]
\end{tikzcd}$$

Isomorphisms *are special cases of* **Galois connections**.

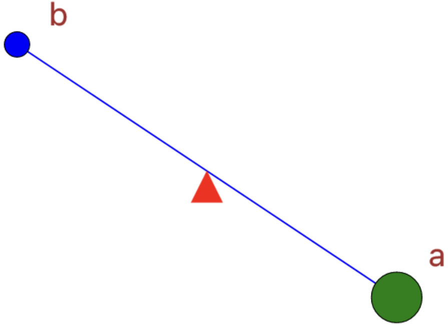
GC — mechanics analogy

Stability:



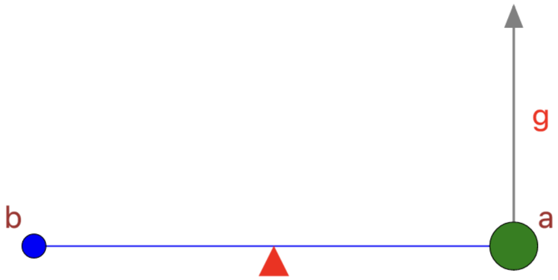
GC — mechanics analogy

Instability:



GC — mechanics analogy

Stability restored:



"Restauratio" rule (Middle Ages).

Example of GC

Integer division:

$$z \times y \leqslant x \;\equiv\; z \leqslant x \div y$$

that is:

$$z \underbrace{\times y}_{f} \leqslant x \;\equiv\; z \leqslant x \underbrace{\div y}_{g}$$

So:

$$(\times y) \vdash (\div y)$$

Principle:

Difficult $(\div y)$ *explained by* **easy** $(\times y)$.

GCs

Interpreting:

$$f^\circ \cdot (\sqsubseteq) = (\leqslant) \cdot g, \textit{ ie.}$$

$$f\ b \sqsubseteq a \ \equiv\ b \leqslant g\ a, \textit{ ie.}$$

$$f \vdash g$$

- $f\ b$ is the **smallest** a such that $b \leqslant g\ a$ holds.
- $g\ a$ is the **largest** b such that $f\ b \sqsubseteq a$ holds.

Thus $z \times y \leqslant x \ \equiv\ z \leqslant x \div y$ reads like this:

$x \div y$ is the largest z such that $z \times y \leqslant x$.

Yes! (back to the primary school desk)

The **whole division** algorithm

$$\begin{array}{r|l} 7 & 2 \\ \cline{2-2} 1 & 3 \end{array} \qquad 2 \times 3 + 1 = 7 \quad , \text{“i.e.“} \qquad 3 = 7 \div 2$$

However

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

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

That is:

$$\begin{array}{r|c} x & y \\ \cline{2-2} \ldots & x \div y \end{array} \qquad z \times y \leqslant x \Rightarrow z \leqslant x \div y$$

$x \div y$ **largest** z
such that
$z \times y \leqslant x$.

GCs as specifications

Thus:

$z \times y \leqslant x \;\equiv\; z \leqslant x \div y$ *is a* **specification** *of* $x \div y$

How does it relate to its **implementation**, e.g.

$$
\begin{array}{l}
x \div y = \\
\quad \textbf{if } x < y \textbf{ then } 0 \\
\quad \textbf{else } 1 + (x - y) \div y
\end{array}
$$

?

It's a long story. For the moment, let us appreciate the power of the GC concept.

GCs as specifications

Consider the following **requirements** about the take function in Haskell:

> take *n xs should yield the* **longest** *possible* **prefix** *of xs not exceeding n in* **length**.

Warming up examples:

$$\mathsf{take}\ 2\ [10, 20, 30] = [10, 20]$$
$$\mathsf{take}\ 20\ [10, 20, 30] = [10, 20, 30]$$
$$\ldots$$

How do we write a formal **specification** for these requirements?

Specifying functions on lists

Clearly,

- $\mathsf{take}\ n\ xs$ is a **prefix** of xs — specify this as e.g.

  $$\mathsf{take}\ n\ xs\ \preceq\ xs$$

  where $\preceq$ denotes the **prefix** partial order.
- the length of $\mathsf{take}\ n\ xs$ cannot exceed n — easy to specify:

  $$\mathsf{length}\ (\mathsf{take}\ n\ xs) \leqslant n$$

Altogether:

$$\mathsf{length}\ (\mathsf{take}\ n\ xs) \leqslant n\ \wedge\ \mathsf{take}\ n\ xs\ \preceq\ xs \tag{112}$$

But this is not **enough** — (silly) implementation $\mathsf{take}\ n\ xs = [\,]$ meets (112)!

Superlatives...

The crux is how to formally specify the **superlative** in

> *...*$\mathsf{take}\ n\ xs$ *should yield the* **longest possible** *prefix...*

This is the **hard** part but there is a standard method to follow:

- think of an arbitrary list ys also satisfying (112)

$$\mathsf{length}\ ys \leqslant n \ \wedge\ ys \preceq xs$$

- Then (from above) ys should be a prefix of $\mathsf{take}\ n\ xs$:

$$\mathsf{length}\ ys \leqslant n \wedge ys \preceq xs \ \Rightarrow\ ys \preceq \mathsf{take}\ n\ xs \qquad (113)$$

Final touch

So we have two clauses,

a **easy** *one (112)*

and

a **hard** *one (113).*

Interestingly, (112) can be derived from (113) itself,

$$\mathsf{length}\ ys \leqslant n \wedge ys \preceq xs \Leftarrow ys \preceq \mathsf{take}\ n\ xs$$

by letting $ys := \mathsf{take}\ n\ xs$ and simplifying.

So a single line is enough to **formally specify** *take*:

$$length\ ys \leqslant n \ \wedge\ ys \preceq xs \quad \equiv \quad ys \preceq \mathsf{take}\ n\ xs \tag{114}$$

— a **GC**.

Reasoning about specifications (GCs)

One of the advantages of **formal specification** is that one may **quest** the specification (aka **model**) to derive useful properties of the design **before the implementation phase**.

GCs + **indirect equality** (on partial orders) yield much in this process — see the following exercise.

Exercise 48: Solely relying on specification (114) use indirect equality to prove that

$$take\ (length\ xs)\ xs = xs \tag{115}$$

$$take\ 0\ xs = [\,] \tag{116}$$

$$take\ n\ [\,] = [\,] \tag{117}$$

hold. □

GCs: many properties for free

$(f\ \ b) \leqslant a \equiv b \sqsubseteq (g\ \ a)$

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

Exercise 49: Derive from (109) that both f and g are monotonic. □

Remark on GCs

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

$$\underbrace{z\ (\times y)}_{f\ z} \leqslant x \quad \equiv \quad z \leqslant \underbrace{x\ (\div y)}_{g\ n}$$

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	$(\_)^\circ$	$(\_)^\circ$	$b\ R^\circ\ a \equiv a\ R\ b$

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

Example of calculation from the GC

Converse involution (cancellation):

$$(R^{\circ})^{\circ} \;=\; R \tag{118}$$

Proof of (118):

$$\begin{array}{cl}
 & (R^{\circ})^{\circ} = R \\
\equiv & \quad \{ \text{ antisymmetry ("ping-pong") } \} \\
 & (R^{\circ})^{\circ} \;\subseteq\; R \wedge R \;\subseteq\; (R^{\circ})^{\circ} \\
\equiv & \quad \{ \text{ }^{\circ}\text{-universal } \; X^{\circ} \;\subseteq\; Y \;\;\equiv\;\; X \subseteq Y^{\circ} \;\text{ twice } \} \\
 & R^{\circ} \;\subseteq\; R^{\circ} \wedge R^{\circ} \;\subseteq\; R^{\circ} \\
\equiv & \quad \{ \text{ reflexivity (twice) } \} \\
 & \textsc{True}
\end{array}$$

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,

$$X \cdot R \subseteq Y \equiv X \subseteq Y \,/\, R \tag{119}$$

$$R \cdot X \subseteq Y \equiv X \subseteq R \setminus Y \tag{120}$$

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

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

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

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

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	$(\_ - R)$	$(R \cup\ )$	$b\ (X - R)\ a \equiv \left\{\begin{array}{l} b\ X\ a \\ \neg\ (b\ R\ a) \end{array}\right.$

Thus the universal properties of implication and difference,

$$R \cap X \subseteq Y \quad \equiv \quad X \subseteq R \Rightarrow Y \tag{121}$$

$$X - R \subseteq Y \quad \equiv \quad X \subseteq R \cup Y \tag{122}$$

are GCs — etc, etc

Exercise 51: Show that $R \cap (R \Rightarrow Y) \subseteq Y$ ("modus ponens") holds and that $R - R = \bot - R = \bot$. □

Relation shrinking

Given relations $R : A \leftarrow B$ and $S : A \leftarrow A$, define $R \upharpoonright S : A \leftarrow B$, pronounced “$R$ shrunk by S”, by

$$X \subseteq R \upharpoonright S \quad \equiv \quad X \subseteq R \ \wedge \ X \cdot R^{\circ} \subseteq S \tag{123}$$

cf. diagram:

$$\begin{tikzcd}
 & B \arrow[dl, "R\upharpoonright S"'] \arrow[d, "R"] \\
A & A \arrow[l, "S"]
\end{tikzcd}$$

Property (123) states that $R \upharpoonright S$ is the largest part of R such that, if it yields an output for an input x, this must be a ‘maximum, with respect to S, among all possible outputs of x by R.

Exercise 52: Show, by indirect equality, that (123) is equivalent to:

$$R \upharpoonright S \ = \ R \cap S / R^{\circ} \tag{124}$$

□

Relation shrinking

Given relations $R : A \leftarrow B$ and $S : A \leftarrow A$, define $R \upharpoonright S : A \leftarrow B$, pronounced "$R$ shrunk by S", by

$$X \subseteq R \upharpoonright S \quad \equiv \quad X \subseteq R \ \wedge \ X \cdot R^{\circ} \subseteq S \tag{123}$$

cf. diagram:

$$\begin{tikzcd}
 & B \arrow[dl, "R \upharpoonright S"'] \arrow[d, "R"] \\
A & A \arrow[l, "S"]
\end{tikzcd}$$

Property (123) states that $R \upharpoonright S$ is the largest part of R such that, if it yields an output for an input x, this must be a 'maximum, with respect to S, among all possible outputs of x by R.

Exercise 53: Show, by indirect equality, that (123) is equivalent to:

$$R \upharpoonright S \ = \ R \cap S / R^{\circ} \tag{124}$$

□

Relation shrinking

Example Given

$$Examiner \times Mark \xleftarrow{R} Student \;=\; \left(\begin{array}{c|c|c} Examiner & Mark & Student \\ \hline Smith & 10 & John \\ Smith & 11 & Mary \\ Smith & 15 & Arthur \\ Wood & 12 & John \\ Wood & 11 & Mary \\ Wood & 15 & Arthur \end{array}\right)$$

suppose we wish to choose the best mark for each student.

Relation shrinking

Then $S = \pi_1 \cdot R$ is the relation

$$\mathit{Mark} \xleftarrow{\pi_1 \cdot R} \mathit{Student} \;=\; \left(\begin{array}{c|c} \mathit{Mark} & \mathit{Student} \\ \hline 10 & \mathit{John} \\ 11 & \mathit{Mary} \\ 12 & \mathit{John} \\ 15 & \mathit{Arthur} \end{array}\right)$$

and

$$\mathit{Mark} \xleftarrow{S \upharpoonright (\geqslant)} \mathit{Student} \;=\; \left(\begin{array}{c|c} \mathit{Mark} & \mathit{Student} \\ \hline 11 & \mathit{Mary} \\ 12 & \mathit{John} \\ 15 & \mathit{Arthur} \end{array}\right)$$

Properties of shrinking

Two *fusion* rules:

$$(S \cdot f) \upharpoonright R \quad = \quad (S \upharpoonright R) \cdot f \tag{125}$$

$$(f \cdot S) \upharpoonright R \quad = \quad f \cdot (S \upharpoonright (f^{\circ} \cdot R \cdot f)) \tag{126}$$

"Chaotic optimization":

$$R \upharpoonright \top \quad = \quad R \tag{127}$$

"Impossible optimization":

$$R \upharpoonright \bot \quad = \quad \bot \tag{128}$$

"Brute force" determinization:

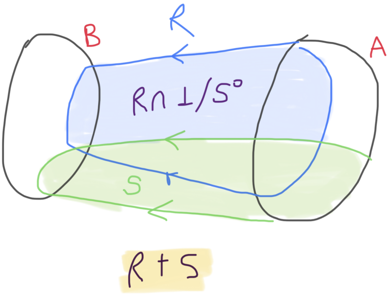
$$R \upharpoonright id \quad = \quad \text{largest simple fragment of } R \tag{129}$$

Relation overriding

The relational **overriding** combinator

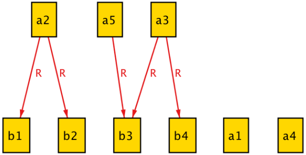
$$R \dagger S \;=\; S \cup R \cap \bot / S^\circ \tag{130}$$

yields 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".



Exercise on relation overriding

Let $R : A \to B$ be given as in the picture, where $A = \{ a_1, a_2, a_3, a_4, a_5 \}$ and $B = \{ b_1, b_2, b_3, b_4 \}$:



Represent as a Boolean matrix the following relation overriding:

$$P = \top \dagger R = \begin{array}{c|c|c|c|c|c} & a_1 & a_2 & a_3 & a_4 & a_5 \\ \hline b_1 & 0 & 0 & 0 & 0 & 0 \\ \hline b_2 & 0 & 0 & 0 & 0 & 0 \\ \hline b_3 & 0 & 0 & 0 & 0 & 0 \\ \hline b_4 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Exercise on relation overriding

And now this other one:

$$Q = R \dagger (\underline{b_4} \cdot \underline{a_2}^\circ) = \begin{array}{c|c|c|c|c|c|}
 & a_1 & a_2 & a_3 & a_4 & a_5 \\ \hline
b_1 & 0 & 0 & 0 & 0 & 0 \\ \hline
b_2 & 0 & 0 & 0 & 0 & 0 \\ \hline
b_3 & 0 & 0 & 0 & 0 & 0 \\ \hline
b_4 & 0 & 0 & 0 & 0 & 0 \\ \hline
\end{array}$$

□

Exercise 54: (a) Show that $\bot \dagger S = S$, $R \dagger \bot = R$ and $R \dagger R = R$ hold. (b) Infer the universal property:

$$X \subseteq R \dagger S \quad \equiv \quad X - S \ \subseteq \ R \wedge (X - S) \cdot S^\circ = \bot \tag{131}$$

□

Class 8 — Programming from GCs

Back to take

In exercise 48 we inferred

$$take\ 0\ xs = [\,]$$
$$take\ n\ [\,] = [\,]$$

from the specification of take (114).

The remaining case is, by pattern matching

$$\mathsf{take}\ (n+1)\ (h : xs) \tag{132}$$

Can this be inferred from (114) too?

Let us unfold (132) and see what happens. **NB**: We will need the following fact about list-prefixing:

$$s \preceq (h : t)\ \equiv\ s = [\,] \vee \langle \exists\, s' \ :\ s = (h : s') :\ s' \preceq t \rangle \tag{133}$$

Back to take

In exercise 48 we inferred

$$take\ 0\ xs = [\,]$$
$$take\ n\ [\,] = [\,]$$

from the specification of take (114).

The remaining case is, by pattern matching

$$\mathsf{take}\ (n+1)\ (h : xs) \tag{132}$$

Can this be inferred from (114) too?

Let us unfold (132) and see what happens. **NB**: We will need the following fact about list-prefixing:

$$s \preceq (h : t) \equiv s = [\,] \vee \langle \exists\ s' \ :\ s = (h : s') :\ s' \preceq t \rangle \tag{133}$$

Back to take

$$ys \preceq \mathsf{take}\ (n+1)\ (h : xs)$$

$$\equiv \qquad \{\ \ \text{GC (114) ; prefix (133)}\ \ \}$$

$$\mathsf{length}\ ys \leqslant n+1 \wedge (ys = [\,] \vee \langle \exists\ ys'\ :\ ys = (h : ys') :\ ys' \preceq xs \rangle)$$

$$\equiv \qquad \{\ \ \text{distribution ; length } [\,] \leqslant n+1\ \ \}$$

$$ys = [\,] \vee \langle \exists\ ys'\ :\ ys = (h : ys') :\ \mathsf{length}\ ys \leqslant n+1 \wedge ys' \preceq xs \rangle$$

$$\equiv \qquad \{\ \ \mathsf{length}\ (h : t) = 1 + \mathsf{length}\ t\ \ \}$$

$$ys = [\,] \vee \langle \exists\ ys'\ :\ ys = (h : ys') :\ \mathsf{length}\ ys' \leqslant n \wedge ys' \preceq xs \rangle$$

$$\equiv \qquad \{\ \ \text{GC (114)}\ \ \}$$

$$ys = [\,] \vee \langle \exists\ ys'\ :\ ys = (h : ys') :\ ys' \preceq \mathsf{take}\ n\ xs \rangle$$

$$\equiv \qquad \{\ \ \text{fact (133)}\ \ \}$$

$$ys \preceq h : \mathsf{take}\ n\ xs$$

$$:: \qquad \{\ \text{indirect equality over list prefixing } (\preceq)\ \}$$

$$\mathsf{take}\ (n+1)\ (h : xs) = h : \mathsf{take}\ n\ xs$$

Back to take

Altogether, we've calculated the **implementation** of take

```
take 0 _             = []
take _ []            = []
take(n+1) (h:xs) = h:take n xs
```

from its **specification**

$$\mathit{length}\ \mathit{ys} \leqslant n \ \wedge\ \mathit{ys} \preceq \mathit{xs} \quad \equiv \quad \mathit{ys} \preceq \mathsf{take}\ n\ \mathit{xs}$$

(a GC), by indirect equality.

A clear illustration of the **FM** golden triad:

- specification — **what** the program should do;
- implementation — **how** the program does it;
- justification — **why** the program does it (CbC in this case).

Back to take

Altogether, we've calculated the **implementation** of take

```
take 0 _          = []
take _ []         = []
take(n+1) (h:xs) = h:take n xs
```

from its **specification**

$$\mathit{length}\ \mathit{ys} \leqslant n \ \wedge\ \mathit{ys} \preceq \mathit{xs} \quad \equiv \quad \mathit{ys} \preceq \mathsf{take}\ n\ \mathit{xs}$$

(a GC), by indirect equality.

A clear illustration of the **FM** golden triad:

- specification — **what** the program should do;
- implementation — **how** the program does it;
- justification — **why** the program does it (CbC in this case).

Exercise

Exercise 55: Follow the **specification method** of the previous example to formally specify the requirements

> *The function* takeWhile p xs *should yield the longest prefix of* xs *such that all* x *in such a prefix satisfy predicate* p*.*

and

> *The function* filter p xs *should yield the longest sublist of* xs *such that all* x *in such a sublist satisfy predicate* p*.*

NB: assume the existence of the sublist ordering $ys \sqsubseteq xs$ such that e.g. `"ab"` $\sqsubseteq$ `"acb"` holds but `"ab"` $\sqsubseteq$ `"bca"` **does not** hold. □

Putting (more) relational combinators together

We define the **lexicographic chaining** of two (endo) relations $A \xleftarrow{R;S} A$ as follows,

$$R\,;S \quad = \quad R \cap (R^\circ \Rightarrow S) \tag{134}$$

recalling (135):

$$R \cap X \subseteq Y \quad \equiv \quad X \subseteq (R \Rightarrow Y)$$

Thus:

$$b\,(R\,;S)\,a \quad \equiv \quad b\,R\,a \wedge (a\,R\,b \Rightarrow b\,S\,a)$$

Exercise 56: Show by indirect equality that (134) is the same as the universal property

$$X \;\subseteq\; R\,;S \equiv X \;\subseteq\; R \wedge X \cap R^\circ \;\subseteq\; S \tag{135}$$

□

Putting (more) relational combinators together

We define **relational projection** as follows:

$$\pi_{g,f} R \overset{\mathrm{def}}{=} g \cdot R \cdot f^{\circ} \qquad (136)$$

$$\begin{tikzcd} B \arrow[d, "g"'] & A \arrow[l, "R"'] \arrow[d, "f"] \\ C & D \arrow[l, "\pi_{g,f} R"] \end{tikzcd}$$

By indirect equality we obtain:

$$\pi_{g,f} R \subseteq X \equiv R \subseteq g^{\circ} \cdot X \cdot f \qquad (137)$$

— that is,

$$\boxed{(f\ X) \subseteq Y \equiv X \subseteq (g\ Y)}$$

Description	$f = g^{\flat}$	$g = f^{\sharp}$	**Obs.**
projection	$(\pi_{g,f}\ \_)$	$(g^{\circ} \cdot \_ \cdot f)$	

Putting (more) relational combinators together

Thus:

Projection $\pi_{g,f} R$ *is the smallest relation which, wherever* b *is* R*-related to* a*, relates* $(g\ b)$ *to* $(f\ a)$*.*

Regarding relations as **sets of pairs**, we have

$$\pi_{g,f} R \quad \overset{\text{def}}{=} \quad \{(g\ b, f\ a) \mid (b, a) \in R\} \tag{138}$$

NB: This generalizes the homonymous SQL projection operator, in the context of which functions f and g are regarded as **attributes**.

Relations as functions — the power transpose

Implicit in how e.g. **Alloy** works is the fact that **relations** can be represented by **functions**. Let $A \xrightarrow{R} B$ be a relation in

$$\Lambda R : A \to \mathcal{P}\ B$$
$$\Lambda R\ a = \{\, b \mid b\ R\ a \,\}$$

such that:

$$\Lambda R = f \quad \equiv \quad \langle \forall\ b, a\ ::\ b\ R\ a\ \equiv\ b \in f\ a \rangle$$

That is (universal property):

$$\begin{tikzcd} A \to \mathcal{P}\ B \arrow[r, bend left, "(\in\cdot)"] \arrow[r, phantom, "\cong"] & A \to B \arrow[l, bend left, "\Lambda"] \end{tikzcd}$$

$$f = \Lambda R \quad \equiv \quad \in \cdot f = R \tag{139}$$

In words: any **relation** can be represented by set-valued **function**.

Relations as functions — the "Maybe" transpose

Let $A \xrightarrow{S} B$ be a **simple** relation. Define the function

$$\Gamma S : A \to B + 1$$

such that:

$$\Gamma S = f \quad \equiv \quad \langle \forall\ b, a\ ::\ b\ S\ a\ \equiv\ (i_1\ b) = f\ a \rangle$$

That is:

$$\begin{tikzcd} A \to B + 1 \arrow[rr, bend left, "(i_1^\circ \cdot)"] \arrow[rr, phantom, "\cong"] & & A \to B \arrow[ll, bend left, "\Gamma"] \end{tikzcd} \qquad f = \Gamma S \quad \equiv \quad S = i_1^\circ \cdot f \quad (140)$$

In words: simple **relations** can be represented by "pointer"-valued **functions**.

"Maybe" transpose in action (Haskell)

(Or how **data** becomes **functional**.)

For finite relations, and assuming these represented **extensionally** as lists of pairs, the function

$$mT = \mathit{flip}\ \mathit{lookup} :: \mathit{Eq}\ a \Rightarrow [(a,b)] \to (a \to \mathit{Maybe}\ b)$$

implements the "Maybe"-transpose

$$\begin{tikzcd} A \to B + 1 \arrow[r, bend left, "(i_1^\circ \cdot)"] \arrow[r, phantom, "\cong"] & A \to B \arrow[l, bend left, "\Gamma"] \end{tikzcd} \qquad f = \Gamma S \quad \equiv \quad S = i_1^\circ \cdot f$$

in Haskell.

Data "functionalization"

Inspired by (140), we may implement

$$\mathsf{Just}^{\circ} \cdot mT$$

in Haskell,

$$\begin{array}{l}
pap :: Eq\ a \Rightarrow [(a, t)] \rightarrow a \rightarrow t \\
pap\ m = unJust \cdot (mT\ \ m)\ \textbf{where}\ unJust\ (\mathsf{Just}\ a) = a
\end{array}$$

which converts a list of **key-value pairs** into a *partial function*.

NB: pap abbreviates "partial application".

In this way, the **columnar** *approach to data processing can be made* **functional**.

Class 9 — Predicates become relations

How predicates become relations

Recall from (35) the notation

$$\frac{f}{g} \;=\; g^{\circ} \cdot f$$

and, given **predicate** $\mathbb{B} \xleftarrow{p} A$, the relation $A \xleftarrow{\frac{true}{p}} X$, where *true* is the everywhere-True **constant** function.

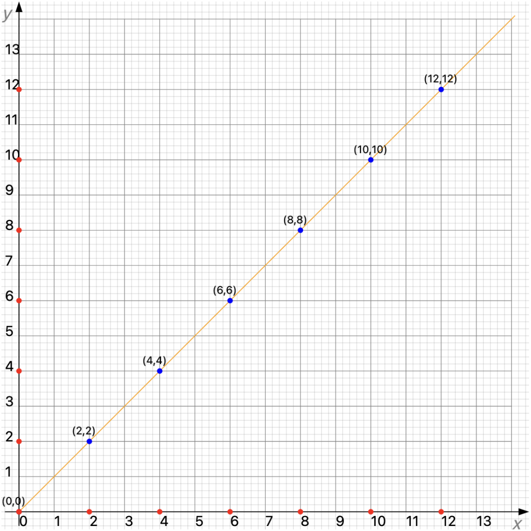
Now define:

$$\Phi_p = id \cap \frac{true}{p} \tag{141}$$

Clearly, Φ_p is the **coreflexive** relation which **represents** predicate p as a binary relation — see the following exercise.

Exercise 57: Show that $y \; \Phi_p \; x \;\equiv\; y = x \wedge p \; x$ □

Φ_{even}



Predicates become relations

Moreover,

$$\Phi_p \cdot \top = \frac{true}{p} \tag{142}$$

thanks to distributive property (57) and

$$\underline{k} \cdot R \subseteq \underline{k}$$

Then:

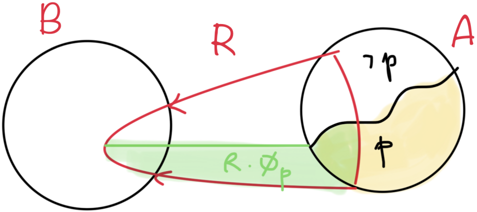
$$\Phi_p \cdot R = R \cap \Phi_p \cdot \top \tag{143}$$

$$R \cdot \Phi_q = R \cap \top \cdot \Phi_q \tag{144}$$

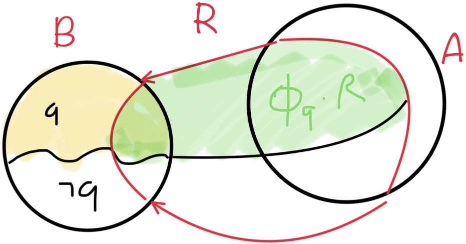
These are called **post** and **pre** *restrictions* of R.

Relational restrictions

Pre *restriction* $R \cdot \Phi_p$:



Post *restriction* $\Phi_q \cdot R$:



Distinguished coreflexives: domain and range

Do you remember...

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

How about intersecting both with id?

$$\delta R = \ker R \cap id \tag{145}$$

$$\rho R = \mathrm{img}\ R \cap id \tag{146}$$

Distinguished coreflexives: domain and range

Clearly:

$$a' \; \delta R \; a \;\equiv\; a' = a \wedge \langle \exists \; b \;:\; b \; R \; a' :\; b \; R \; a \rangle$$

that is

$$\delta R = \Phi_p \text{ **where** } p \; a = \langle \exists \; b \;::\; b \; R \; a \rangle$$

Thus δR captures all a which R **reacts** *to.*

Dually,

$$\rho R = \Phi_q \text{ **where** } q \; b = \langle \exists \; a \;::\; b \; R \; a \rangle$$

Thus ρR captures all b which R **hits** *as target.*

Distinguished coreflexives: domain and range

As was to be expected:

$$(f\ X) \subseteq Y \equiv X \subseteq (g\ Y)$$

Description	f	g	**Obs.**
domain	δ	$(\top\cdot)$	left $\subseteq$ restricted to coreflexives
range	ρ	$(\cdot\top)$	left $\subseteq$ restricted to coreflexives

Spelling out these GC:

$$\delta X \ \subseteq \ Y \equiv X \subseteq \top \cdot Y \tag{147}$$

$$\rho R \ \subseteq \ Y \equiv R \subseteq Y \cdot \top \tag{148}$$

Propositio de homine et capra et lvpo

Recalling the data model (4)

$$\begin{tikzcd}
Being \arrow[r, "Eats"] & Being \arrow[d, "where"'] & \\
 & Bank \arrow[r, "cross"] & Bank
\end{tikzcd}$$

we specify the move of *Being*s to the other bank is an example of relational restriction and overriding:

$$carry(where, who) = where \dagger (cross \cdot where \cdot \Phi_{who}) \qquad (149)$$

In **Alloy** syntax:

```
fun carry[where: Being -> one Bank,
          who:   set Being]: Being -> one Bank
    { where ++ (who <: where).cross }
```

Exercises

Exercise 58: Prove the distributive property:

$$g^\circ \cdot (R \cap S) \cdot f \quad = \quad g^\circ \cdot R \cdot f \cap g^\circ \cdot S \cdot f \tag{150}$$

Then show that

$$g^\circ \cdot \Phi_p \cdot f \quad = \quad \frac{f}{g} \cap \frac{true}{p \cdot g} \tag{151}$$

holds (both sides of the equality mean $g\ b = f\ a \wedge p\ (g\ b)$). □

Exercise 59: Infer

$$\Phi_q \cdot \Phi_p = \Phi_q \cap \Phi_p \tag{152}$$

from properties (144) and (143). □

Exercise 60: Derive (138) from (136). □

Exercises

Exercise 61: (a) From (135) infer:

$$\bot \Rightarrow R \quad = \quad \top \tag{153}$$

$$R \Rightarrow \top \quad = \quad \top \tag{154}$$

(b) via indirect equality over (134) show that

$$\top \,;\, S \quad = \quad S \tag{155}$$

holds for any S and that, for R symmetric, we have:

$$R \,;\, R \quad = \quad R \tag{156}$$

□

Exercise 62: Show that $R - S \subseteq R$, $R - \bot = R$ and $R - R = \bot$ hold. □

Exercises

Exercise 63: Let students in a course have two numeric marks,

$$\mathbb{N} \xleftarrow{mark1} Student \xrightarrow{mark2} \mathbb{N}$$

and define the **preorders**:

$$\begin{aligned} \leqslant_{mark1} &= mark1^\circ \cdot \leqslant \cdot mark1 \\ \leqslant_{mark2} &= mark2^\circ \cdot \leqslant \cdot mark2 \end{aligned}$$

Spell out in pointwise notation the meaning of lexicographic ordering

$$\leqslant_{mark1} ; \leqslant_{mark2}$$

□

Exercises

Exercise 64: Show that

$$R \dagger f = f$$

holds, arising from (131,122) — where f is a function, of course. □

Exercise 65: Function *move* (149) could have been defined by

$$move = where^{cross}_{who}$$

using the following (generic) **selective update** operator:

$$R^f_p \quad = \quad R \dagger (f \cdot R \cdot \Phi_p) \tag{157}$$

Prove the equalities: $R^{id}_p = R$, $R^f_{false} = R$ and $R^f_{true} = f \cdot R$.
□

Exercises

Exercise 66: A relation R is said to satisfy **functional dependency** (FD) $g \rightarrow f$, written $g \xtwoheadrightarrow{R} f$ wherever projection $\pi_{f,g} R$ (136) is **simple**.

1. Recalling (81), prove the equivalence:

$$g \xtwoheadrightarrow{R} f \quad \equiv \quad f \leqslant g \cdot R^{\circ} \tag{158}$$

2. Show that (158) trivially holds wherever g is injective and R is simple, for all (suitably typed) f.

3. Prove the **composition rule** of FDs:

$$h \xtwoheadleftarrow{S \cdot R} g \quad \Leftarrow \quad h \xtwoheadleftarrow{S} f \quad \wedge \quad f \xtwoheadleftarrow{R} g \tag{159}$$

□

Class 10 — Contracts

Back to pre/post relational restrictions

Looking at the types in a **pre** restriction

$$\begin{tikzcd} A \arrow[d, "R"'] & A \arrow[l, "\Phi_p"'] \\ B & \end{tikzcd}$$

... and those in a **post** restriction

$$\begin{tikzcd} & A \arrow[d, "R"] \\ B & B \arrow[l, "\Phi_q"] \end{tikzcd}$$

we immediately realize they fit together into a “magic” square...

$$\begin{tikzcd} A \arrow[d, "R"'] \arrow[dr, phantom, "\subseteq"] & A \arrow[l, "\Phi_p"'] \arrow[d, "R"] \\ B & B \arrow[l, "\Phi_q"] \end{tikzcd}$$

Back to pre/post relational restrictions

Looking at the types in a **pre** restriction

$$\begin{tikzcd} A \arrow[d, "R"'] & A \arrow[l, "\Phi_p"'] \\ B & \end{tikzcd}$$

... and those in a **post** restriction

$$\begin{tikzcd} & A \arrow[d, "R"] \\ B & B \arrow[l, "\Phi_q"] \end{tikzcd}$$

we immediately realize they fit together into a “magic” square...

$$\begin{tikzcd} A \arrow[d, "R"'] & A \arrow[l, "\Phi_p"'] \arrow[d, "R"] \\ B \arrow[ur, phantom, "\subseteq"] & B \arrow[l, "\Phi_q"] \end{tikzcd}$$

Back to pre/post relational restrictions

Looking at the types in a **pre** restriction

$$\begin{tikzcd}
A \arrow[d, "R"'] & A \arrow[l, "\Phi_p"'] \\
B &
\end{tikzcd}$$

... and those in a **post** restriction

$$\begin{tikzcd}
& A \arrow[d, "R"] \\
B & B \arrow[l, "\Phi_q"]
\end{tikzcd}$$

we immediately realize they fit together into a “magic” square...

$$\begin{tikzcd}
A \arrow[d, "R"'] \arrow[dr, phantom, "\subseteq"] & A \arrow[l, "\Phi_p"'] \arrow[d, "R"] \\
B & B \arrow[l, "\Phi_q"]
\end{tikzcd}$$

Our good old "square" (again!!)

$$\begin{tikzcd}
A \arrow[d, "R"'] & A \arrow[l, "\Phi_p"'] \arrow[d, "R"] \arrow[dl, phantom, "\subseteq"] \\
B & B \arrow[l, "\Phi_q"]
\end{tikzcd}$$

$$R \cdot \Phi_p \subseteq \Phi_q \cdot R$$

What does this mean?

Let us see this for the (simpler) case in which R is a function f:

$$\begin{tikzcd}
A \arrow[d, "f"'] & A \arrow[l, "\Phi_p"'] \arrow[d, "f"] \arrow[dl, phantom, "\subseteq"] \\
B & B \arrow[l, "\Phi_q"]
\end{tikzcd}$$

$$f \cdot \Phi_p \subseteq \Phi_q \cdot f \qquad (160)$$

Contracts

By shunting, (160) is the same as $\Phi_p \subseteq f^\circ \cdot \Phi_q \cdot f$, therefore meaning:

$$\langle \forall\ a\ :\ p\ a :\ q\ (f\ a) \rangle \tag{161}$$

by exercise 57.

In words:

For all inputs a such that **condition** *$p\ a$ holds, the output $f\ a$ satisfies* **condition** *q.*

In software design, this is known as a (functional) **contract**, which we shall write

$$p \xrightarrow{f} q \tag{162}$$

— a notation that generalizes the type of f. **Important**: thanks to (143), (160) can also be written: $f \cdot \Phi_p \ \subseteq\ \Phi_q \cdot \top$.

Weakest pre-conditions

Note that more than one (**pre**) condition p may ensure (**post**) condition q on the outputs of f.

Indeed, contract $false \xrightarrow{f} q$ always holds, but pre-condition $false$ is useless (“**too strong**”).

The weaker p, the better. Now, is there a **weakest** such p?

See the calculation aside.

$$\left\{ \begin{array}{cl}
 & f \cdot \Phi_p \subseteq \Phi_q \cdot f \\
\equiv & \qquad \{ \text{ see above (143) } \} \\
 & f \cdot \Phi_p \subseteq \Phi_q \cdot \top \\
\equiv & \qquad \{ \text{ shunting (32); (142) } \} \\
 & \Phi_p \subseteq f^\circ \cdot \frac{true}{q} \\
\equiv & \qquad \{ \text{ (37) } \} \\
 & \Phi_p \subseteq \frac{true}{q \cdot f} \\
\equiv & \qquad \{ \Phi_p \subseteq id \text{ ; (52) } \} \\
 & \Phi_p \subseteq id \cap \frac{true}{q \cdot f} \\
\equiv & \qquad \{ \text{ (141) } \} \\
 & \Phi_p \subseteq \Phi_{q \cdot f}
\end{array} \right.$$

We conclude that $q \cdot f$ is such a **weakest** pre-condition.

Weakest pre-conditions

Notation $\mathrm{WP}(f,q) = q \cdot f$ is often used for **weakest** pre-conditions.

Exercise 67: Calculate the weakest pre-condition $\mathrm{WP}(f,q)$ for the following function / post-condition pairs:

- $f\ x = x^2 + 1$, $q\ y = y \leqslant 10$ (in $\mathbb{R}$)
- $f = \mathbb{N} \xrightarrow{\mathsf{succ}} \mathbb{N}$, $q = even$
- $f\ x = x^2 + 1$, $q\ y = y \leqslant 0$ (in $\mathbb{R}$)

□

Exercise 68: Show that $q \xleftarrow{g \cdot f} p$ holds provided $r \xleftarrow{f} p$ and $q \xleftarrow{g} r$ hold. □

Invariants versus contracts

In case **contract**

$$q \xrightarrow{f} q$$

holds (162), we say that q is an **invariant** of f — meaning that the "truth value" of q remains unchanged by execution of f.

More generally, invariant q is **preserved** by function f provided contract $p \xrightarrow{f} q$ holds and $p \Rightarrow q$, that is, $\Phi_p \subseteq \Phi_q$.

Some pre-conditions are weaker than others:

We shall say that w is the **weakest** *pre-condition for f to preserve* **invariant** *q wherever $\mathrm{WP}(f, q) = w \wedge q$, where $\Phi_{(p \wedge q)} = \Phi_p \cdot \Phi_q$.*

Invariants versus contracts

Recalling the Alcuin puzzle, let us define the **starvation** invariant as a predicate on the state of the puzzle, passing the *where* function as a parameter w:

$$\begin{tikzcd}
\mathit{Being} \arrow[d, "w"'] \arrow[dr, phantom, "\subseteq"] & \mathit{Being} \arrow[l, "\mathit{CanEat}"'] \arrow[d, "\underline{\mathit{Farmer}}"] \\
\mathit{Bank} & \mathit{Being} \arrow[l, "w"]
\end{tikzcd}$$

$$\mathit{starving}\ w \quad = \quad w \cdot \mathit{CanEat} \ \subseteq\ w \cdot \underline{\mathit{Farmer}}$$

Recalling (149),

$$\mathit{carry}(\mathit{where}, \mathit{who}) = \mathit{where} \dagger (\mathit{cross} \cdot \mathit{where} \cdot \Phi_{\mathit{who}})$$

we also define:

$$\mathit{trip}\ b\ w = \mathit{carry}\ (w, b) \tag{163}$$

Invariants versus contracts

Then the **contract**

$$\mathit{starving} \xrightarrow{\mathit{trip}\ b} \mathit{starving}$$

would mean that the function $\mathit{trip}\ b$ — that should carry b to the other bank of the river — always preserves the invariant: $\mathrm{WP}(\mathit{trip}\ b, \mathit{starving}) = \mathit{starving}$.

Things are not that easy, however: there is a need for a **pre-condition** ensuring that b is on the Farmer's bank and is *the right being to carry* !

Let us see a simpler example first.

Library loan example

$$\begin{tikzcd}
 & \mathit{ISBN} & \mathit{Name} & \\
\mathit{Title} & \mathit{Book} \arrow[l, "title"'] \arrow[u, "isbn"'] \arrow[r, "R"] \arrow[d, "Auth"] & \mathit{User} \arrow[u, "name"'] \arrow[r, "addr"'] \arrow[d, "card"'] & \mathit{Address} \\
 & \mathit{Author} & \mathit{Id} &
\end{tikzcd}$$

$u\ R\ b$ means "book b currently on loan to library user u".

Desired properties:

- *same book not on loan to more than one user;*
- *no book with no authors;*
- *no two users with the same card* Id.

NB: lowercase arrow labels denote functions, as usual.

Library loan example

Encoding of desired properties:

- no book on loan to more than one user:

  $$Book \xrightarrow{R} User \quad \textit{is } \textbf{simple}$$

- no book without an author:

  $$Book \xrightarrow{Auth} Author \quad \textit{is } \textbf{entire}$$

- no two users with the same card Id:

  $$User \xrightarrow{card} Id \quad \textit{is } \textbf{injective}$$

NB: as all other arrows are functions, they are simple+entire.

Library loan example

Encoding of desired properties as relational **invariants**:

- no book on loan to more than one user:
$$\mathrm{img}\ R \ \subseteq\ id \tag{164}$$
- no book without an author:
$$id \ \subseteq\ \ker\ Auth \tag{165}$$
- no two users with the same card Id:
$$\ker\ card \ \subseteq\ id \tag{166}$$

Library loan example

Now think of two operations on $User \xleftarrow{R} Book$, one that **returns** books to the library and another that **records** new borrowings:

$$return\ S\ R = R - S \tag{167}$$

$$borrow\ S\ R = S \cup R \tag{168}$$

Clearly, these operations only change the *books-on-loan* relation R, which is conditioned by invariant

$$inv\ R = \mathrm{img}\ R \ \subseteq\ id \tag{169}$$

The question is, then: are the following "types"

$$inv \xleftarrow{return\ S} inv \tag{170}$$

$$inv \xleftarrow{borrow\ S} inv \tag{171}$$

ok? We check (170,171) below.

Library loan example

Checking (170):

$$
\begin{array}{ll}
& \mathit{inv}\ (\mathit{return}\ S\ R) \\
\equiv & \qquad \{\ \text{inline definitions}\ \} \\
& \mathrm{img}\ (R - S)\ \subseteq\ \mathit{id} \\
\Leftarrow & \qquad \{\ \text{since } \mathrm{img}\ \text{ is monotonic}\ \} \\
& \mathrm{img}\ R\ \subseteq\ \mathit{id} \\
\equiv & \qquad \{\ \text{definition}\ \} \\
& \mathit{inv}\ R \\
\Box &
\end{array}
$$

So, for all R, $\mathit{inv}\ R \Rightarrow \mathit{inv}\ (\mathit{return}\ S\ R)$ holds — invariant inv is preserved.

Library loan example

At this point note that (170) was checked only as a *warming-up* exercise — we don't need to worry about it! Why?

As $R - S$ is smaller than R (exercise 62) and "smaller than injective is injective" (exercise 28), it is immediate that inv (169) is preserved.

To see this better, unfold and draw definition (169):

$$inv\ R \quad = \quad \begin{tikzcd} Book \arrow[d, "R"'] & User \arrow[l, "R^\circ"'] \arrow[d, "id"] \\ User \arrow[ur, phantom, "\subseteq"] & User \arrow[l, "id"] \end{tikzcd}$$

As R is on the lower-path of the square, it can always get smaller.

Library loan example

This “rule of thumb” does not work for *borrow S* because, in general, $R \subseteq borrow\ S\ R$.

So R gets bigger, not smaller, and we have to check the contract:

$$
\begin{array}{ll}
& inv\ (borrow\ S\ R) \\
\equiv & \qquad \{\ \text{inline definitions}\ \} \\
& \mathrm{img}\ (S \cup R) \subseteq id \\
\equiv & \qquad \{\ \text{exercise 27}\ \} \\
& \mathrm{img}\ R \subseteq id \wedge \mathrm{img}\ S \subseteq id \wedge S \cdot R^{\circ} \subseteq id \\
\equiv & \qquad \{\ \text{definition of } inv\ \} \\
& inv\ R \wedge \underbrace{\mathrm{img}\ S \subseteq id \wedge S \cdot R^{\circ} \subseteq id}_{\textsc{wp}(borrow\ S, inv)}
\end{array}
$$

Library loan example (Alloy)

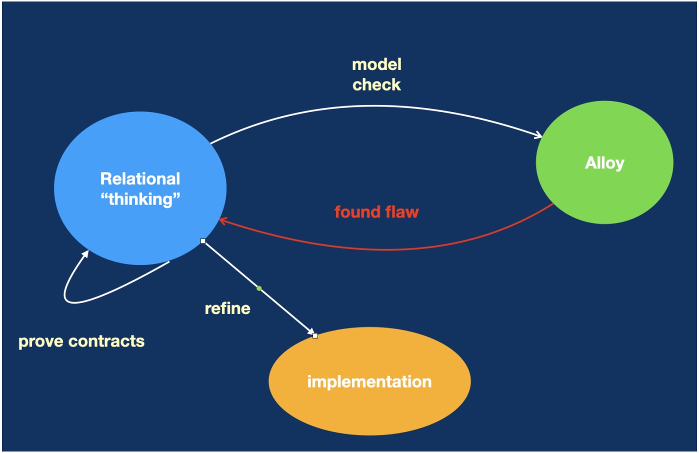
In practice, our proposed **workflow** does not go immediately to the **calculation** of the **weakest precondition** of a **contract**.

We **model-check** the **contract** first, in order to save the process from childish errors:

> *What is the point in trying to prove something that a model checker can easily tell is a nonsense?*

This follows a systematic process, illustrated next.

Relation Algebra + Alloy round-trip



Library loan example (Alloy)

First we write the Alloy model of what we have thus far:

```
sig Book {
   title : one Title,
   isbn : one ISBN,
   Auth : some Author,
   R : lone User
}
sig User {
   name : one Name,
   add : some Address,
   card : one Id
}
sig Title, ISBN, Author,
   Name, Address, Id { }
```

```
fact {
   card .˜ card in iden
         -- card is injective
}
fun borrow
      [S, R : Book → lone User] :
         Book → lone User {
      R + S
}
fun return
      [S, R : Book → lone User] :
         Book → lone User {
      R − S
}
```

Library loan example (Alloy)

As we have seen, *return* is no problem, so we focus on *borrow*.

Realizing that most attributes of *Book* and *User* don't matter wrt. checking *borrow*, we comment them all, obtaining a much smaller model:

$$
\begin{array}{l}
\mathrm{sig}\ \mathit{Book}\ \{ R : \mathrm{lone}\ \mathit{User} \} \\
\mathrm{sig}\ \mathit{User}\ \{\ \} \\
\mathrm{fun}\ \mathit{borrow} \\
\quad [S, R : \mathit{Book} \rightarrow \mathrm{lone}\ \mathit{User}] : \\
\quad\quad \mathit{Book} \rightarrow \mathrm{lone}\ \mathit{User}\ \{ \\
\quad R + S \\
\}
\end{array}
$$

Next, we single out the **invariant**, making it explicit as a predicate (aside).

$$
\begin{array}{l}
\mathrm{sig}\ \mathit{Book}\ \{ R : \mathit{User} \} \\
\mathrm{sig}\ \mathit{User}\ \{\ \} \\
\mathrm{pred}\ \mathit{inv}\ \{ \\
\quad R\ \mathbf{in}\ \mathit{Book} \rightarrow \mathrm{lone}\ \mathit{User} \\
\} \\
\mathrm{fun}\ \mathit{borrow} \\
\quad [S, R : \mathit{Book} \rightarrow \mathit{User}] : \\
\quad\quad \mathit{Book} \rightarrow \mathit{User}\ \{ \\
\quad R + S \\
\}
\end{array}
$$

Library loan example (Alloy)

In the step that follows, we make the model **dynamic**, in the sense that we need at least two instances of relation R — one before *borrow* is applied and the other after.

We introduce *Time* as a way of recording such two moments, pulling R out of *Book*

$$\text{sig } \mathit{Time}\ \{ r : \mathit{Book} \rightarrow \mathit{User} \}$$
$$\text{sig } \mathit{Book}\ \{ \}$$
$$\text{sig } \mathit{User}\ \{ \}$$

and re-writing *inv* accordingly (aside).

$$\begin{array}{l} \text{pred } \mathit{inv}\ [t : \mathit{Time}]\ \{ \\ \quad t \cdot r \textbf{ in } \mathit{Book} \rightarrow \text{lone } \mathit{User} \\ \} \end{array}$$

Note how $r : \mathit{Time} \rightarrow (\mathit{Book} \rightarrow \mathit{User})$ is a **function** — it yields, for each $t \in \mathit{Time}$, the relation $\mathit{Book} \xrightarrow{r\ t} \mathit{User}$.

Library loan example (Alloy)

This makes it possible to express contract $inv \xrightarrow{borrow\ S} inv$ in terms of $t \in Time$,

$$\langle \forall\ t, t' \ :\ inv\ t \wedge r\ t' = borrow\ S\ (r\ t) :\ inv\ t' \rangle$$

i.e. in Alloy:

$$\begin{array}{l} assert\ contract\ \{ \\ \quad \text{all}\ t, t' : Time, S : Book \to User\ | \\ \qquad inv\ [t]\ and\ t' \cdot r = borrow\ [t \cdot r, S] \Rightarrow inv\ [t'] \\ \} \end{array}$$

Once we check this, for instance running

$$\text{check}\ contract\ for\ 3\ but\ exactly\ 2\ Time$$

we shall obtain counter-examples. (These were expected...)

Library loan example (Alloy)

The counter-examples will quickly tell us what the problems are, guiding us to add the following pre-condition to the contract:

$$\begin{array}{l}\text{pred } \mathit{pre}\,[t : \mathit{Time}, S : \mathit{Book} \to \mathit{User}]\ \{\\ \quad S\ \textbf{in}\ \mathit{Book} \to \text{lone}\ \mathit{User}\\ \quad {\sim}S \cdot (t \cdot r)\ \textbf{in}\ \text{iden}\\ \}\end{array}$$

The fact that this does not yield counter-examples anymore does not tell us that

- *pre* is enough in general
- *pre* is weakest.

This we have to prove by calculation — as we have seen before.

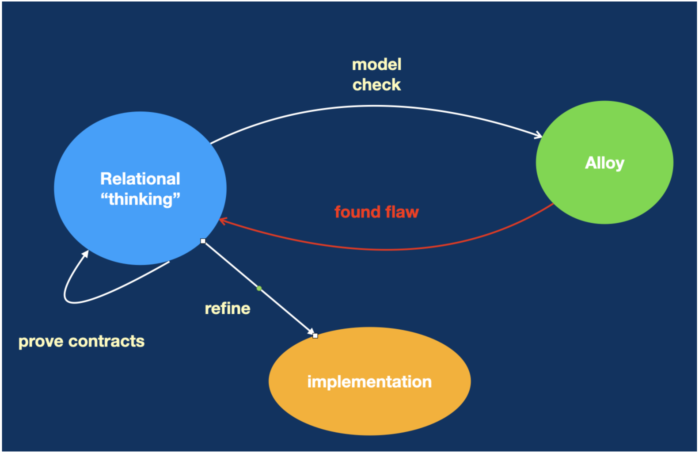
Library loan example (Alloy)

Note that pre-conditioned $borrow\ S \cdot \Phi_{pre}$ is not longer a **function**, because it is not **entire** anymore.

We can encode such a relation in Alloy in an easy-to-read way, as a predicate structured in two parts — pre-condition and post-condition:

```
pred borrow [t, t' : Time, S : Book → User] {
  -- pre-condition
  S in Book → lone User
  ~S · (t · r) in iden
  -- post-condition
  t' · r = t · r + S
}
```

Alloy + Relation Algebra round-trip



Summary

- The Alloy + Relation Algebra round-trip enables us to take advantage of the best of the two verification strategies.
- Diagrams of **invariants** help in detecting which **contracts** don't need to be checked.
- Functional specifications are good as starting point but soon evolve towards becoming relations, comparable to the **methods** of an OO programming language.
- Time was added to the model just to obtain more than one "state". In general, *Time* will be **linearly ordered** so that the **traces** of the model can be reasoned about.[5]

[5] In Alloy, just declare: open util/ordering[Time].

Library loan example revisited

More detailed data model of our **library** with **invariants** captured by diagram

$$\begin{tikzcd}
\mathit{ISBN} \arrow[d, "M"'] \arrow[dr, phantom, "\supseteq"] & \mathit{ISBN} \times \mathit{UID} \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, "R"] \arrow[dr, phantom, "\subseteq"] & \mathit{UID} \arrow[d, "N"] \\
\mathit{Title} \times \mathit{Publisher} \arrow[r, "\top"'] & \mathit{Date} & \mathit{Name} \times \mathit{Address} \times \mathit{Phone} \arrow[l, "\top"]
\end{tikzcd} \tag{172}$$

where

- M — records **books** on loan, identified by ISBN;
- N — records library **users** (identified by user id's in UID);

(both simple) and

- R — records **loan** dates.

Library loan example revisited

The two squares in the diagram impose bounds on R:

- Non-existing **books** cannot be on loan (left square);
- Only known **users** can take books home (right square).

(**NB:** in the database terminology these are known as **integrity constraints**.)

Exercise 69: Add variables to both squares in (172) so that the same conditions are expressed pointwise. Then show that the conjunction of the two squares means the same as assertion

$$R^\circ \quad \subseteq \quad \langle M^\circ \cdot \top, N^\circ \cdot \top \rangle \tag{173}$$

and draw this in a diagram. □

Library loan example revisited

Exercise 70: Consider implementing M, R and N as **files** in a relational **database**. For this, think of **operations** on the database such as, for example, that which records new loans (K):

$$borrow(K,(M,R,N)) \quad = \quad (M, R \cup K, N) \tag{174}$$

It can be checked that the **pre-condition**

$$\text{pre-}borrow(K,(M,R,N)) \quad = \quad R \cdot K^{\circ} \subseteq id$$

is necessary for maintaining (172) (why?) but it is not enough. Calculate — for a rectangle in (172) of your choice — the corresponding clause to be added to pre-*borrow*. □

Library loan example revisited

Exercise 71: The operations that **buy** new books

$$buy(X, (M, R, N)) \quad = \quad (M \cup X, R, N) \tag{175}$$

and **register** new users

$$register(Y, (M, R, N)) \quad = \quad (M, R, N \cup Y) \tag{176}$$

don't need any **pre-conditions**. Why? (Hint: compute their WP.) □

NB: see annex on proofs by $\subseteq$-monotonicity for a strategy generalizing the exercise above.

Relational contracts

Finally, let the following definition

$$p \xrightarrow{R} q \quad \equiv \quad R \cdot \Phi_p \subseteq \Phi_q \cdot R \qquad (177)$$

generalize functional contracts (160) to arbitrary relations, meaning:

$$\langle \forall\ b, a \ :\ b\ R\ a :\ p\ a \Rightarrow q\ b \rangle \qquad (178)$$

— see the exercise below.

Exercise 72: Sow that an alternative way of stating (177) is

$$p \xrightarrow{R} q \quad \equiv \quad R \cdot \Phi_p \subseteq \Phi_q \cdot \top \qquad (179)$$

□

Exercise 19 (continued)

Exercise 73: Recalling exercise 19, let the following relation specify that two dates are at least one week apart in time:

$$d\ Ok\ d' \quad \equiv \quad \mid d - d' \mid > 1\ week$$

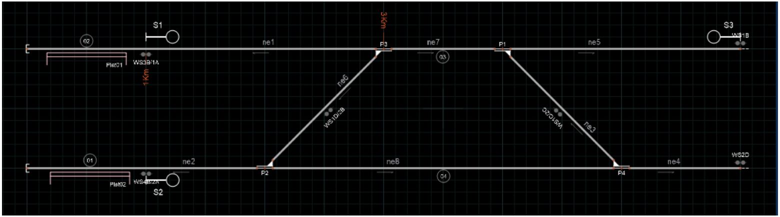
Looking at the type diagram below right, say in your own words the meaning of the invariant specified by the relational type (**??**) statement below, on the left:

$$\ker\ (home \cup away) - id \xrightarrow{date} Ok$$

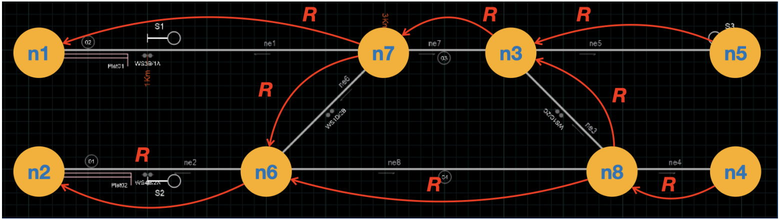
$$\begin{tikzcd} G \arrow[r, "home \cup away"] \arrow[d, "date"'] & T \\ D & G \arrow[l, "date"] \arrow[u, "home \cup away"'] \end{tikzcd}$$

□

Case study: railway topologies



Case study: railway topologies



$$\begin{tikzcd} Sw & N \arrow[l, "S"'] & N \arrow[l, "R"'] \arrow[r, "P"] & Sl \end{tikzcd}$$

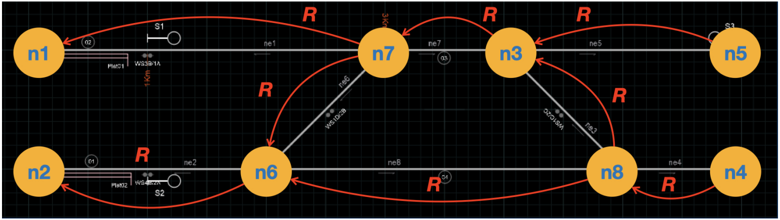
where

Sw – *switches* (‘*agulhas*’)

Sl – *signals* (‘*sinais*’)

□

Case study: railway topologies

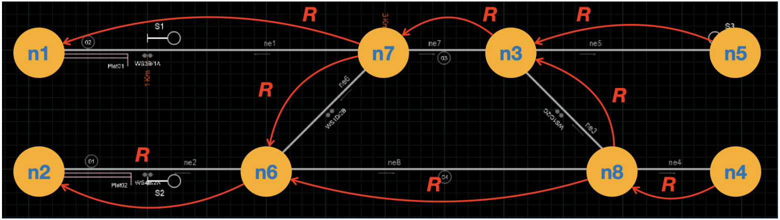


$$Sw \xleftarrow{S} N \xleftarrow{R} N \xrightarrow{P} Sl$$

Switches:

$$switchOk(S, R, P) \quad = \quad \delta S \ \subseteq \ R^{\circ} \cdot (\neq) \cdot R$$

Case study: railway topologies



$$\begin{tikzcd} Sw & N \arrow[l, "S"'] & N \arrow[l, "R"'] \arrow[r, "P"] & Sl \end{tikzcd}$$

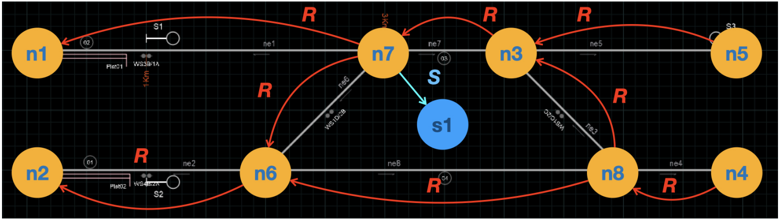
Add a switch:

$$addSwitch\ (s, n)\ (S, R, P) = (S \cup \underline{s} \cdot \underline{n}^{\circ}, R, P)$$

Case study: railway topologies

$$
\begin{array}{ll}
 & \mathit{switchOk}\ (\mathit{addSwitch}\ (s, n)\ (S, R, P)) \\
\equiv & \qquad \{\ \ \ldots\ldots\ldots\ \ \} \\
 & \delta(S \cup \underline{s} \cdot \underline{n}^{\circ})\ \subseteq\ R^{\circ} \cdot (\neq) \cdot R \\
\equiv & \qquad \{\ \ \ldots\ldots\ldots\ \ \} \\
 & \mathit{switchOk}\ (S, R, P) \wedge \underline{n} \cdot \top \cdot \underline{n}^{\circ}\ \subseteq\ R^{\circ} \cdot (\neq) \cdot R \\
\equiv & \qquad \{\ \ \ldots\ldots\ldots\ \ \} \\
 & \mathit{switchOk}\ (S, R, P) \wedge \top\ \subseteq\ \underline{n}^{\circ} \cdot R^{\circ} \cdot (\neq) \cdot R \cdot \underline{n} \\
\equiv & \qquad \{\ \ \ldots\ldots\ldots\ \ \} \\
 & \mathit{switchOk}\ (S, R, P) \wedge \langle \exists\ n_1, n_2\ \ :\ \ n_1 \neq n_2 :\ \ n\ R^{\circ}\ n_1 \wedge n_2\ R\ n \rangle \\
\equiv & \qquad \{\ \ \ldots\ldots\ldots\ \ \} \\
 & \mathit{switchOk}\ (S, R, P) \wedge \underbrace{\langle \exists\ n_1, n_2\ \ :\ \ n_1 \neq n_2 :\ \ n_1\ R\ n \wedge n_2\ R\ n \rangle}_{\mathit{WP}}
\end{array}
$$

Case study: railway topologies



$$\begin{tikzcd} Sw & N \arrow[l, "S"'] & N \arrow[l, "R"'] \arrow[r, "P"] & Sl \end{tikzcd}$$

Switches:

$$switchOk(S, R, P) \quad = \quad \delta S \ \subseteq \ R^{\circ} \cdot (\neq) \cdot R$$

Class 11 — Theorems for free

Parametric polymorphism by example

Function

$$\mathit{countBits} : \mathbb{N}_0 \leftarrow \mathit{Bool}^\star$$
$$\mathit{countBits}\ [\,] = 0$$
$$\mathit{countBits}(b{:}bs) = 1 + \mathit{countBits}\ bs$$

and

$$\mathit{countNats} : \mathbb{N}_0 \leftarrow \mathbb{N}^\star$$
$$\mathit{countNats}\ [\,] = 0$$
$$\mathit{countNats}(b{:}bs) = 1 + \mathit{countNats}\ bs$$

are both subsumed by **generic** (parametric):

$$\mathit{count} : (\forall a)\ \mathbb{N}_0 \leftarrow a^\star$$
$$\mathit{count}\ [\,] = 0$$
$$\mathit{count}(a{:}as) = 1 + \mathit{count}\ as$$

Parametric polymorphism: why?

- Less code (**specific** solution = **generic** solution + **customization**)
- Intellectual reward
- Last but not least, quotation from *Theorems for free!*, by Philip Wadler [6]:

> *From the type of a polymorphic function we can derive a theorem that it satisfies.* (...) *How useful are the theorems so generated? Only time and experience will tell* (...)

- No doubt: free theorems are **very** useful!

Polymorphic type signatures

Polymorphic function signature:

$$f \quad : \quad t$$

where t is a functional type, according to the following "grammar" of types:

$$t \quad ::= \quad t' \leftarrow t''$$

$$t \quad ::= \quad \mathcal{F}(t_1, \ldots, t_n) \qquad \text{type constructor } \mathcal{F}$$

$$t \quad ::= \quad v \qquad \text{type } \textit{variables } v\text{, cf. } \textit{polymorphism}$$

What does it mean for f to be **parametrically** polymorphic?

Free theorem of type t

Let

- V be the set of type variables involved in type t
- $\{R_v\}_{v\in V}$ be a V-indexed family of relations (f_v in case all such R_v are functions).
- R_t be a relation defined inductively as follows:

$$\begin{aligned} R_{t:=v} &= R_v && (180)\\ R_{t:=\mathcal{F}(t_1,\ldots,t_n)} &= \mathcal{F}(R_{t_1},\ldots,R_{t_n}) && (181)\\ R_{t:=t'\leftarrow t''} &= R_{t'} \leftarrow R_{t''} && (182)\end{aligned}$$

Questions: What does $\mathcal{F}$ in the RHS of (181) mean? What kind of relation is $R_{t'} \leftarrow R_{t''}$? See next slides.

Background: relators

Parametric datatype $\mathcal{G}$ is said to be a **relator** [2] wherever, given a relation from A to B, $\mathcal{G}R$ extends R to $\mathcal{G}$-structures: it is a relation

$$\begin{tikzcd} A \arrow[r, dotted, no head] \arrow[d, "R"'] & \mathcal{G}A \arrow[d, "\mathcal{G}R"] \\ B \arrow[r, dotted, no head] & \mathcal{G}B \end{tikzcd} \tag{183}$$

from $\mathcal{G}A$ to $\mathcal{G}B$ which obeys the following properties:

$$\mathcal{G}id = id \tag{184}$$

$$\mathcal{G}\ (R \cdot S) = (\mathcal{G}\ R) \cdot (\mathcal{G}\ S) \tag{185}$$

$$\mathcal{G}(R^\circ) = (\mathcal{G}\ R)^\circ \tag{186}$$

and is monotonic:

$$R \subseteq S \quad \Rightarrow \quad \mathcal{G}R \subseteq \mathcal{G}S \tag{187}$$

Relators: *“Maybe”* example

$$\begin{tikzcd}
A \arrow[r, dotted, no head] \arrow[d, "R"'] & \mathcal{G}A = 1 + A \arrow[d, "\mathcal{G}R = id + R"] \\
B \arrow[r, dotted, no head] & \mathcal{G}B = 1 + B
\end{tikzcd}$$

(Read $1 + A$ as “maybe A”)

Unfolding $\mathcal{G}R = id + R$:

$$
\begin{array}{ll}
& y(id + R)x \\
\equiv & \qquad \{ \text{ unfolding the sum, cf. } id + R = [i_1 \cdot id \ , i_2 \cdot R] \ \} \\
& y(i_1 \cdot i_1^\circ \cup i_2 \cdot R \cdot i_2^\circ)x \\
\equiv & \qquad \{ \text{ relational union (48); image } \} \\
& y(\mathrm{img}\ i_1)x \vee y(i_2 \cdot R \cdot i_2^\circ)x \\
\equiv & \qquad \{ \text{ let } NIL \text{ be } \underline{\text{the}} \text{ inhabitant of the singleton type } \} \\
& y = x = i_1 NIL \vee \langle \exists\ b, a \ : \ y = i_2\ b \wedge x = i_2\ a : \ b\ R\ a \rangle
\end{array}
$$

Relators: R^* example

Take $\mathcal{F}X = X^\star$.

Then, for some $B \xleftarrow{R} A$, relator $B^\star \xleftarrow{R^\star} A^\star$ is the relation

$$R^* = [\mathsf{nil}\ , \mathsf{cons} \cdot (R \times R^*)] \cdot \mathsf{out} \tag{188}$$

Why? Look at this diagram:

$$\begin{tikzcd}
A \arrow[d, "R"'] & A^* \arrow[rr, "\mathsf{out}"] \arrow[d, "R^*"'] & & 1 + A \times A^* \arrow[d, "id + id \times R^*"] \\
B & B^* & 1 + B \times B^* \arrow[l, "\mathsf{in}"'] & 1 + A \times B^* \arrow[l, "id + R \times id"']
\end{tikzcd}$$

NB: $\mathsf{in} = [\mathsf{nil}\ , \mathsf{cons}]$ where $\mathsf{nil}\ \_ = [\,]$ and $\mathsf{cons}\ (h, t) = h : t$.

Relators: R^* example

Take $\mathcal{F}X = X^\star$.

Then, for some $B \xleftarrow{R} A$, relator $B^\star \xleftarrow{R^\star} A^\star$ is the relation

$$R^* = [\mathsf{nil}\ , \mathsf{cons} \cdot (R \times R^*)] \cdot \mathsf{out} \tag{188}$$

Why? Look at this diagram:

$$\begin{tikzcd}
A \arrow[d, "R"'] & A^* \arrow[rr, "\mathsf{out}"] \arrow[d, "R^*"'] & & 1 + A \times A^* \arrow[d, "id+id\times R^*"] \\
B & B^* & 1 + B \times B^* \arrow[l, "\mathsf{in}"] & 1 + A \times B^* \arrow[l, "id+R\times id"]
\end{tikzcd}$$

NB: $\mathsf{in} = [\mathsf{nil}\ , \mathsf{cons}]$ where $\mathsf{nil}\ \_ = [\,]$ and $\mathsf{cons}\ (h, t) = h : t$.

About R^*

Then:

$$
\begin{array}{ll}
 & R^* \cdot \mathsf{in} = [\mathsf{nil}\ , \mathsf{cons} \cdot (R \times R^*)] \\
\equiv & \qquad \{\ \mathsf{in} = [\mathsf{nil}\ , \mathsf{cons}]\ \text{etc}\ \ \} \\
 & \left\{ \begin{array}{l} R^* \cdot \mathsf{nil} = \mathsf{nil} \\ R^* \cdot \mathsf{cons} = \mathsf{cons} \cdot (R \times R^*) \end{array} \right.
\end{array}
$$

that is:

$$
\left\{ \begin{array}{l} y\ R^*\ [\,]\ \equiv\ y = [\,] \\ y\ R^*\ (h : t)\ \equiv\ \langle \exists\ b, x\ :\ y = (b : x) :\ b\ R\ a \wedge x\ R^*\ t \rangle \end{array} \right.
$$

In case $R := f$, $R^* = \mathsf{map}\ f$.

Exercises

Exercise 74: Inspect the meaning of properties (184) and (186) for the list relator R^* defined above. □

Exercise 75: Show that the *identity* relator $\mathcal{I}$, which is such that $\mathcal{I}\ R \quad = \quad R$ and the *constant* relator $\mathcal{K}$ (for a given data type K) which is such that $\mathcal{K}\ R \quad = \quad id_K$ are indeed relators. □

Exercise 76: Show that (Kronecker) product

$$\begin{tikzcd}
A \arrow[d, "R"'] & C \arrow[d, "S"'] \arrow[r, dash, dotted] & \mathcal{G}(A,C) = A \times C \arrow[d, "\mathcal{G}(R,S)=R\times S"] \\
B & D \arrow[r, dash, dotted] & \mathcal{G}(B,D) = B \times D
\end{tikzcd}$$

is a (binary) relator. □

Background: “Reynolds arrow” operator

The following relation on functions

$$f(R \leftarrow S)g \quad \equiv \quad f \cdot S \subseteq R \cdot g \qquad \begin{tikzcd} A \arrow[d, "f"'] & B \arrow[l, "S"'] \arrow[d, "g"] \\ C & D \arrow[l, "R"] \end{tikzcd} \qquad (189)$$

is another instance of our ”magic rectangle”.

That is to say,

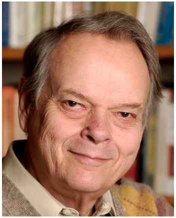
$$\frac{A \xleftarrow{S} B \qquad C \xleftarrow{R} D}{C^A \xleftarrow{R \leftarrow S} D^B}$$

For instance, $f(id \leftarrow id)g \;\equiv\; f = g$ that is, $id \leftarrow id \;=\; id$

Free theorem (FT) of type t

The *free theorem* (FT) of type t is the following (remarkable) result due to J. Reynolds [5], advertised by P. Wadler [6] and re-written by Backhouse [1] in the pointfree style:

> *Given any function $\theta : t$, and V as above, then $\theta\ R_t\ \theta$ holds, for any relational instantiation of type variables in V.*



J.C. Reynolds
(1935–2013)

Note that this theorem

- is a result about t
- holds **independently** of the actual definition of θ.
- holds about any polymorphic function of type t

First example (id)

The target function:

$$\theta = id : a \leftarrow a$$

Calculation of $R_{t=a\leftarrow a}$:

$$\begin{array}{ll} & R_{a\leftarrow a} \\ \equiv & \qquad \{ \text{ rule } \quad R_{t=t'\leftarrow t''} \quad = \quad R_{t'} \leftarrow R_{t''} \quad \} \\ & R_a \leftarrow R_a \end{array}$$

Calculation of FT (R_a abbreviated to R):

$$\begin{array}{ll} & id(R \leftarrow R)id \\ \equiv & \qquad \{ \ (189) \ \} \\ & id \cdot R \subseteq R \cdot id \end{array}$$

First example (*id*)

In case R is a function f, the FT theorem boils down to *id*'s **natural** property:

$$id \cdot f \;=\; f \cdot id$$

cf.

$$\begin{tikzcd}
a & a \arrow[l, "id"'] \arrow[d, "f"] \\
b & b \arrow[l, "id"]
\arrow[from=1-1, to=2-1, "f"']
\end{tikzcd}$$

which can be read alternatively as stating that *id* is the **unit** of composition.

Second example (*reverse*)

The target function: $\theta = reverse : a^\star \leftarrow a^\star$.

Calculation of $R_{t=a^\star \leftarrow a^\star}$:

$$\begin{array}{ll}
 & R_{a^\star \leftarrow a^\star} \\
\equiv & \qquad \{ \text{ rule } \quad R_{t=t' \leftarrow t''} \quad = \quad R_{t'} \leftarrow R_{t''} \quad \} \\
 & R_{a^\star} \leftarrow R_{a^\star} \\
\equiv & \qquad \{ \text{ rule } \quad R_{t=\mathcal{F}(t_1,\ldots,t_n)} \quad = \quad \mathcal{F}(R_{t_1},\ldots,R_{t_n}) \quad \} \\
 & {R_a}^\star \leftarrow {R_a}^\star
\end{array}$$

where $s\ R^\star s'$ is given by (188). The calculation of FT follows.

Second example (*reverse*)

The FT itself will predict (R_a abbreviated to R):

$$
\begin{array}{ll}
& reverse(R^\star \leftarrow R^\star)reverse \\
\equiv & \qquad \{ \text{ definition } \; f(R \leftarrow S)g \;\; \equiv \;\; f \cdot S \subseteq R \cdot g \;\; \} \\
& reverse \cdot R^\star \subseteq R^\star \cdot reverse
\end{array}
$$

In case R is a function r, the FT theorem boils down to *reverse*'s **natural** property:

$$reverse \cdot r^\star \;=\; r^\star \cdot reverse$$

that is,

$$reverse \; [\; r \;\; a \mid a \leftarrow l \,] \;=\; [\; r \;\; b \mid b \leftarrow reverse \;\; l \,]$$

Second example (*reverse*)

Further calculation (back to R):

$$
\begin{array}{ll}
& reverse \cdot R^\star \subseteq R^\star \cdot reverse \\
\equiv & \qquad \{ \text{ shunting rule (32) } \} \\
& R^\star \subseteq reverse^\circ \cdot R^\star \cdot reverse \\
\equiv & \qquad \{ \text{ going pointwise (8, 23) } \} \\
& \langle \forall\ s, r\ ::\ s\ R^\star r \Rightarrow (reverse\ s) R^\star (reverse\ r) \rangle
\end{array}
$$

An instance of this pointwise version of *reverse*-FT will state that, for example, *reverse* will respect element-wise orderings ($R := <$):

Third example: FT of *sort*

Our next example calculates the FT of

$$sort : a^\star \leftarrow a^\star \leftarrow (Bool \leftarrow (a \times a))$$

where the first parameter stands for the chosen ordering relation, expressed by a binary predicate:

$$
\begin{array}{ll}
& sort(R_{(a^\star \leftarrow a^\star) \leftarrow (Bool \leftarrow (a \times a))})sort \\
\equiv & \qquad \{ \ (181, 180, 182); \text{ abbreviate } R_a := R \ \} \\
& sort((R^\star \leftarrow R^\star) \leftarrow (R_{Bool} \leftarrow (R \times R)))sort \\
\equiv & \qquad \{ \ R_{t:=Bool} = id \text{ (constant relator) — cf. exercise 75 } \} \\
& sort((R^\star \leftarrow R^\star) \leftarrow (id \leftarrow (R \times R)))sort
\end{array}
$$

Third example: FT of *sort*

$$
\begin{array}{ll}
 & sort((R^{\star} \leftarrow R^{\star}) \leftarrow (id \leftarrow (R \times R)))sort \\
\equiv & \qquad \{ \ (189) \ \} \\
 & sort \cdot (id \leftarrow (R \times R)) \ \subseteq \ (R^{\star} \leftarrow R^{\star}) \cdot sort \\
\equiv & \qquad \{ \ \text{shunting } (32) \ \} \\
 & (id \leftarrow (R \times R)) \ \subseteq \ sort^{\circ} \cdot (R^{\star} \leftarrow R^{\star}) \cdot sort \\
\equiv & \qquad \{ \ \text{introduce variables } f \text{ and } g \ (8,\ 23) \ \} \\
 & f(id \leftarrow (R \times R))g \ \Rightarrow \ (sort\ f)(R^{\star} \leftarrow R^{\star})(sort\ g) \\
\equiv & \qquad \{ \ (189) \text{ twice} \ \} \\
 & f \cdot (R \times R) \subseteq g \ \Rightarrow \ (sort\ f) \cdot R^{\star} \subseteq R^{\star} \cdot (sort\ g)
\end{array}
$$

Third example: FT of *sort*

Case $R := r$:

$$f \cdot (r \times r) = g \quad \Rightarrow \quad (sort\ f) \cdot r^{\star} = r^{\star} \cdot (sort\ g)$$

$$\equiv \qquad \{ \ \text{introduce variables} \ \}$$

$$\left\langle \begin{array}{c} \forall\ a, b\ :: \\ f(r\ a, r\ b) = g(a, b) \end{array} \right\rangle \Rightarrow \left\langle \begin{array}{c} \forall\ l\ :: \\ (sort\ f)(r^{\star}\ l) = r^{\star}(sort\ g\ l) \end{array} \right\rangle$$

Denoting predicates f, g by infix orderings $\leqslant, \preceq$:

$$\left\langle \begin{array}{c} \forall\ a, b\ :: \\ r\ a \leqslant r\ b \equiv a \preceq b \end{array} \right\rangle \Rightarrow \left\langle \begin{array}{c} \forall\ l\ :: \\ sort\ (\leqslant)(r^{\star}\ l) = r^{\star}(sort\ (\preceq)\ l) \end{array} \right\rangle$$

That is, for r monotonic and injective,

$$sort\ (\leqslant)\ [\ r\ a \mid a \leftarrow l\ ]$$

is always the same list as

$$[\ r\ a \mid a \leftarrow sort\ (\preceq)\ l\ ]$$

Exercises

Exercise 77: Let C be a nonempty data domain and let and $c \in C$. Let $\underline{c}$ be the *"everywhere* c" function, recall (25). Show that the free theorem of $\underline{c}$ reduces to

$$\langle \forall\, R \ ::\ R \subseteq \top \rangle \tag{190}$$

□

Exercise 78: Calculate the free theorem associated with the projections $A \xleftarrow{\pi_1} A \times B \xrightarrow{\pi_2} B$ and instantiate it to (a) functions; (b) coreflexives. Introduce variables and derive the corresponding pointwise expressions. □

Exercises

Exercise 79: Consider higher order function `const: a -> b -> a` such that, given any x of type a, produces the constant function $\mathit{const}\ x$. Show that the equalities

$$\begin{aligned} \mathit{const}(f\ x) &= f \cdot (\mathit{const}\ x) && (191)\\ (\mathit{const}\ x) \cdot f &= \mathit{const}\ x && (192)\\ (\mathit{const}\ x)^{\circ} \cdot (\mathit{const}\ x) &= \top && (193) \end{aligned}$$

arise as corollaries of the *free theorem* of const. □

Exercises

Exercise 80: The following is a well-known Haskell function

$$\mathrm{filter} :: (a \to \mathbb{B}) \to [a] \to [a]$$

Calculate the free theorem associated with its type

$$\mathit{filter} : a^\star \leftarrow a^\star \leftarrow (\mathit{Bool} \leftarrow a)$$

and instantiate it to the case where all relations are functions. □

Exercise 81: In many sorting problems, data are sorted according to a given *ranking* function which computes each datum's numeric rank (eg. students marks, credits, etc). In this context one may parameterize sorting with an extra parameter f ranking data into a fixed numeric datatype, eg. the integers: $\mathit{serial} : (a \to \mathbb{N}) \to a^\star \to a^\star$.
Calculate the FT of serial. □

Exercises

Exercise 82: Consider the following function from Haskell's Prelude:

$$\begin{array}{l} \mathit{findIndices} :: (a \rightarrow \mathbb{B}) \rightarrow [a] \rightarrow [\mathbb{Z}] \\ \mathit{findIndices}\ p\ xs = [i \mid (x, i) \leftarrow \mathsf{zip}\ xs\ [0\,..],\ p\ x] \end{array}$$

which yields the indices of elements in a sequence `xs` which satisfy `p`. For instance, $\mathit{findIndices}\ (< 0)\ [1, -2, 3, 0, -5] = [1, 4]$. Calculate the FT of this function. □

Exercise 83: Choose arbitrary functions from Haskell's Prelude and calculate their FT. □

Exercises

Exercise 84: Wherever two equally typed functions f, g such that $f\ a \leqslant g\ a$, for all a, we say that f is *pointwise at most* g and write $f \overset{.}{\leqslant} g$. In symbols:

$$f \overset{.}{\leqslant} g \quad = \quad f \subseteq (\leqslant) \cdot g \qquad \text{cf. diagram} \qquad (194)$$

$$\begin{tikzcd} & A \arrow[dl, "f"'] \arrow[dr, "g"] & \\ B & \subseteq & B \arrow[ll, "\leqslant"] \end{tikzcd}$$

Show that implication

$$f \overset{.}{\leqslant} g \quad \Rightarrow \quad (map\ f) \overset{.}{\leqslant}^{\star} (map\ g) \qquad (195)$$

follows from the *FT* of the function $map\ :\ (a \rightarrow b) \rightarrow a^{\star} \rightarrow b^{\star}$. □

Automatic generation of free theorems (Haskell)

See the interesting site in Janis Voigtlaender's home page:

http://www-ps.iai.uni-bonn.de/ft

Relators in our calculational style are implemented in this automatic generator by structural *lifting*.

Exercise 85: Infer the FT of the following function, written in Haskell syntax,

$$\begin{array}{l} \mathbf{while} :: (a \rightarrow \mathbb{B}) \rightarrow (a \rightarrow a) \rightarrow (a \rightarrow b) \rightarrow a \rightarrow b \\ \mathbf{while}\ p\ f\ g\ x = \mathbf{if}\ \neg\ (p\ x)\ \mathbf{then}\ g\ x\ \mathbf{else}\ \mathbf{while}\ p\ f\ g\ (f\ x) \end{array}$$

which implements a generic `while`-loop. Derive its corollary for functions and compare your result with that produced by the tool above. □

Background — Eindhoven quantifier calculus

Trading:

$$\langle \forall\ k\ :\ R \wedge S :\ T \rangle = \langle \forall\ k\ :\ R :\ S \Rightarrow T \rangle \tag{196}$$

$$\langle \exists\ k\ :\ R \wedge S :\ T \rangle = \langle \exists\ k\ :\ R :\ S \wedge T \rangle \tag{197}$$

de Morgan:

$$\neg \langle \forall\ k\ :\ R :\ T \rangle = \langle \exists\ k\ :\ R :\ \neg T \rangle \tag{198}$$

$$\neg \langle \exists\ k\ :\ R :\ T \rangle = \langle \forall\ k\ :\ R :\ \neg T \rangle \tag{199}$$

One-point:

$$\langle \forall\ k\ :\ k = e :\ T \rangle = T[k := e] \tag{200}$$

$$\langle \exists\ k\ :\ k = e :\ T \rangle = T[k := e] \tag{201}$$

Background — Eindhoven quantifier calculus

Nesting:

$$\langle \forall\ a, b : R \wedge S : T \rangle = \langle \forall\ a : R : \langle \forall\ b : S : T \rangle\rangle \tag{202}$$
$$\langle \exists\ a, b : R \wedge S : T \rangle = \langle \exists\ a : R : \langle \exists\ b : S : T \rangle\rangle \tag{203}$$

Rearranging-$\forall$:

$$\langle \forall\ k : R \vee S : T \rangle = \langle \forall\ k : R : T \rangle \wedge \langle \forall\ k : S : T \rangle \tag{204}$$
$$\langle \forall\ k : R : T \wedge S \rangle = \langle \forall\ k : R : T \rangle \wedge \langle \forall\ k : R : S \rangle \tag{205}$$

Rearranging-$\exists$:

$$\langle \exists\ k : R : T \vee S \rangle = \langle \exists\ k : R : T \rangle \vee \langle \exists\ k : R : S \rangle \tag{206}$$
$$\langle \exists\ k : R \vee S : T \rangle = \langle \exists\ k : R : T \rangle \vee \langle \exists\ k : S : T \rangle \tag{207}$$

Splitting:

$$\langle \forall\ j : R : \langle \forall\ k : S : T \rangle\rangle = \langle \forall\ k : \langle \exists\ j : R : S \rangle : T \rangle \tag{208}$$
$$\langle \exists\ j : R : \langle \exists\ k : S : T \rangle\rangle = \langle \exists\ k : \langle \exists\ j : R : S \rangle : T \rangle \tag{209}$$

References

K. Backhouse and R.C. Backhouse.
Safety of abstract interpretations for free, via logical relations and Galois connections.
SCP, 15(1–2):153–196, 2004.

R.C. Backhouse, P. de Bruin, P. Hoogendijk, G. Malcolm, T.S. Voermans, and J. van der Woude.
Polynomial relators.
In *AMAST'91*, pages 303–362. Springer-Verlag, 1992.

D. Jackson.
Software Abstractions: Logic, Language, and Analysis.
The MIT Press, Cambridge Mass., 2012.
Revised edition, ISBN 0-262-01715-2.

C.B. Jones.
Software Development — A Rigorous Approach.
Series in Computer Science. Prentice-Hall International, Upper Saddle River, NJ, USA, 1980.
C.A.R. Hoare (series editor).

J.C. Reynolds.
Types, abstraction and parametric polymorphism.
Information Processing 83, pages 513–523, 1983.

P.L. Wadler.
Theorems for free!
In *4th International Symposium on Functional Programming Languages and Computer Architecture*, pages 347–359, London, Sep. 1989. ACM.