CSI - A Calculus for Information Systems (2024/25)

Class 1 — About FM

Global picture

Concerning software ‘engineering’:

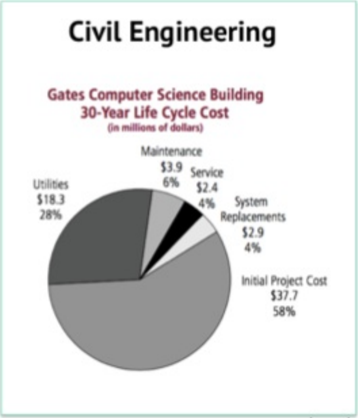
Software

- **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

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

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?

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

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

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

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

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

$$x - z \leq y \;\equiv\; x \leq y + z$$

all-hatt

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

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:

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

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

$sorted = \ldots marks \ldots$

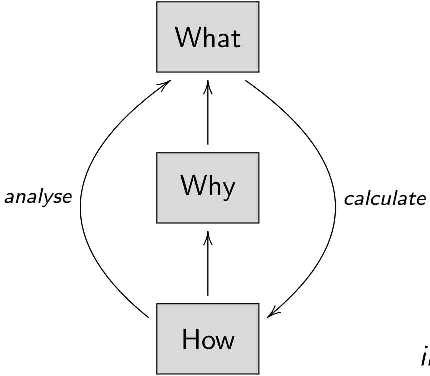
$bag = \ldots.$

But,

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

Yes — Wait and see :-)

FM — scientific software design

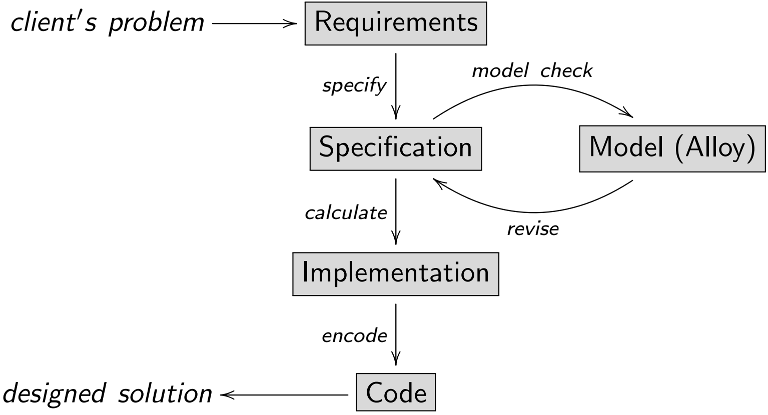


specification (*model*)

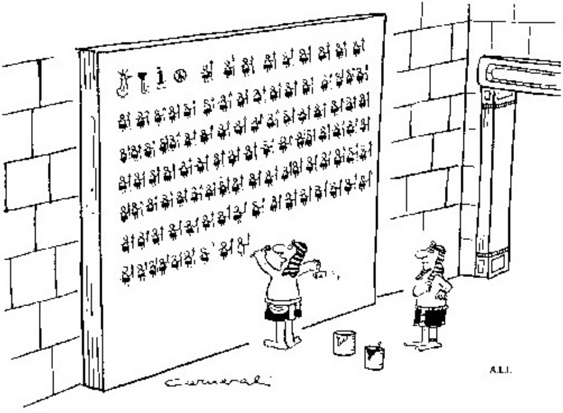
justification

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 [5]

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

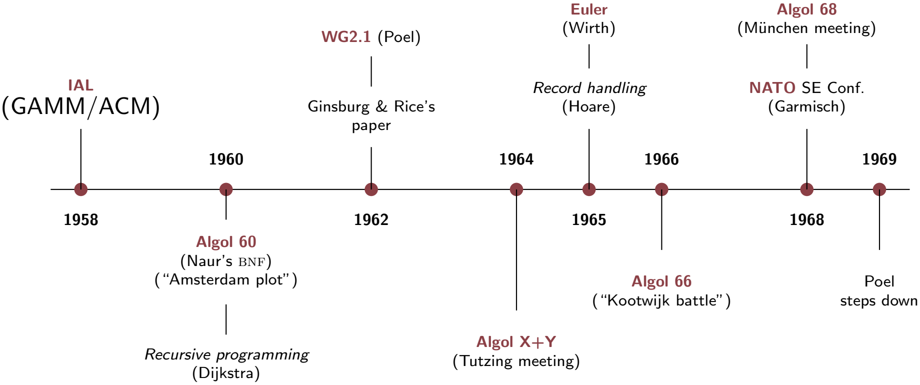
Tools:

- Alloy 6
- Coq
- Frama-C
- NuSMV
- Overture

Resources:

- Formal Methods Europe
- Formal Methods wiki (Oxford)

60+ years ago (1958-)



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

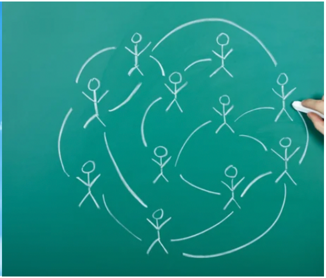
`pre`...? `choice`...?

- Non-determinism
- Parallelism
- Abstraction

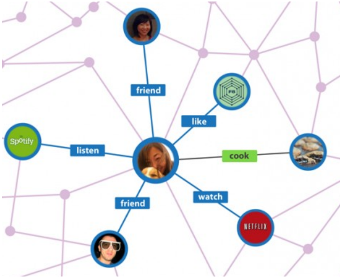
Functions not enough!

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

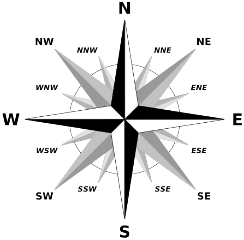
Software is too **wide** — from assembly to quantum programming

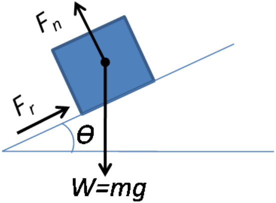
Can you think of a **unified** theory able to express and reason about software **in general**?

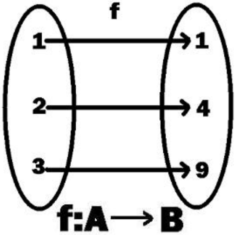
Put in another way:

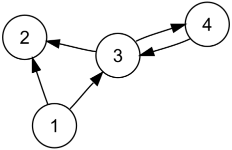
Is there a “lingua franca” for the software sciences?

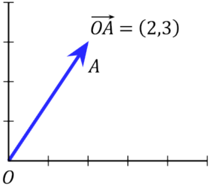
Check the pictures...



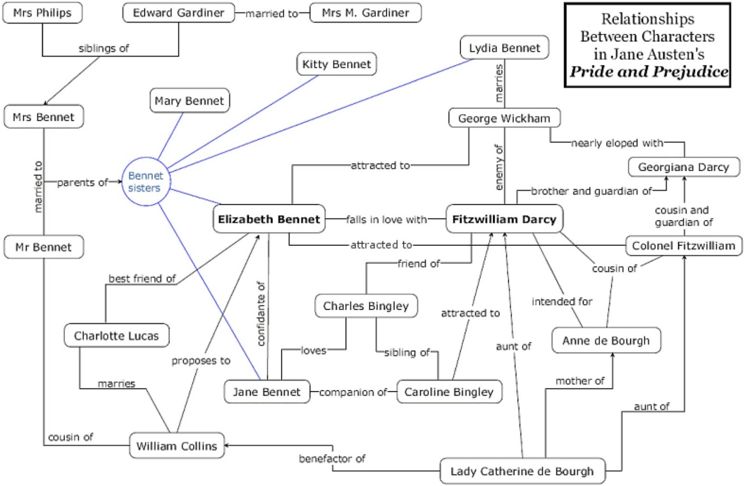






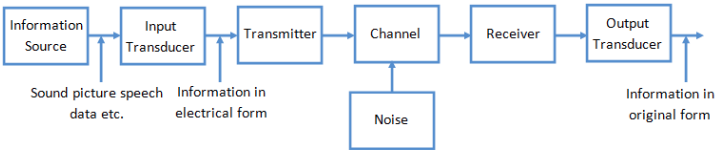


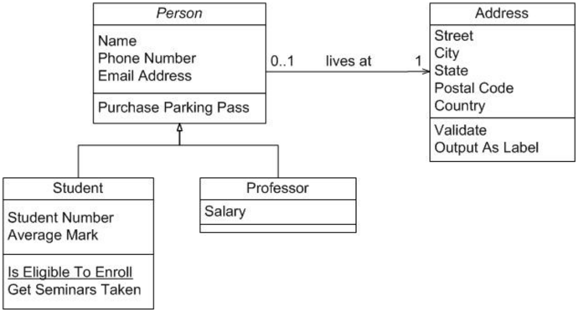
Check the pictures



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

Check the pictures





Check the pictures

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

Arrows everywhere

Arrows! A (graphical) device **common** to describing (many) **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 descriptions may have a **quantitative** side too.

Going Relational

Relation algebra

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

Question:

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

Relation algebra

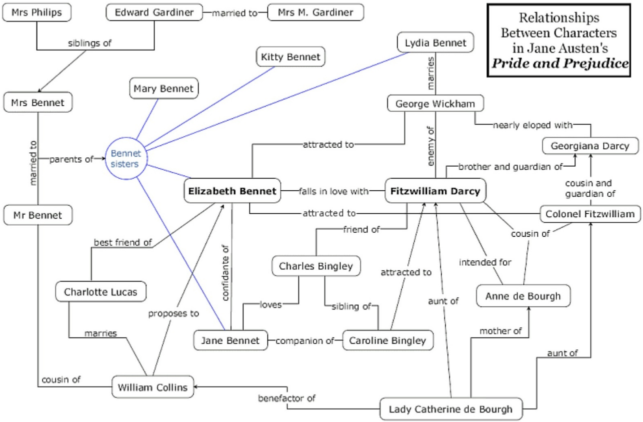
Historically, predicate logic was **not** the first one proposed:

- 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.)

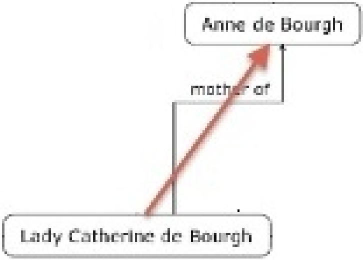
Arrow notation for relations

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

$$mother\_of : People \rightarrow People$$

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



Pairs

Consider assertions

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

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, one might have written instead:

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

What are $(\leqslant)$, *isMotherOf*, $(1+)$?

- They can 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. $\mathtt{Catherine}$ *isMotherOf* $\mathtt{Anne}$ *— and in maths — eg.* $0 \leqslant \pi$ *— extends to arbitrary* $B \xleftarrow{R} A$ *: we write*

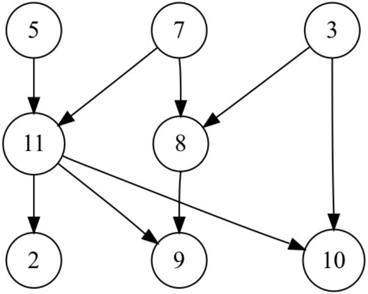
$$b \; R \; a$$

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

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

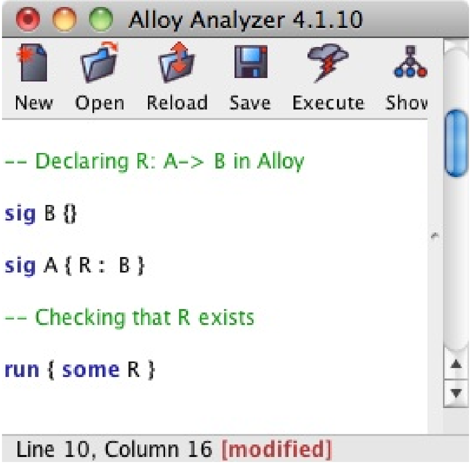
In this case $A = B = \{1..11\}$. Relations $A \xleftarrow{R} A$ over a single type A are also referred to as (directed) **graphs**.

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** [4] in this course.



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.

Home work

- How would you address this problem?
- Try an write an Alloy for it (sig's only)

NB: You can seek help from ChatGPT — but please be critical...

```
abstract sig Item {}
one sig Fox, Goose, Beans extends Item {}

abstract sig Location {}
one sig InitialBank, FarBank extends Location {}

sig Boat {
    passengers: set Item
}

// Predicates to define the constraints
pred farmerCanCross[boat: Boat] {
    // Farmer must be on the boat
    Fox in boat.passengers or Goose in boat.passengers or Bea
}

pred foxAndGooseSafe[boat: Boat] {
    // Fox and Goose cannot be left alone together
    Fox in boat.passengers implies not (Goose in boat.passeng
}
```

Propositio de homine et capra et lvpo

Data types:

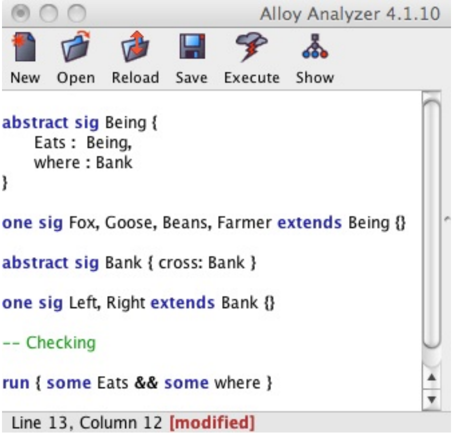
$$
\begin{aligned}
Being &= \{Farmer, Fox, Goose, Beans\} && (2) \\
Bank &= \{Left, Right\} && (3)
\end{aligned}
$$

Relations:

$$
\begin{array}{ccccc}
Being & \xrightarrow{Eats} & Being & & \\
 & & {\scriptstyle where}\downarrow & & \\
 & & Bank & \xrightarrow{cross} & Bank
\end{array}
\qquad (4)
$$

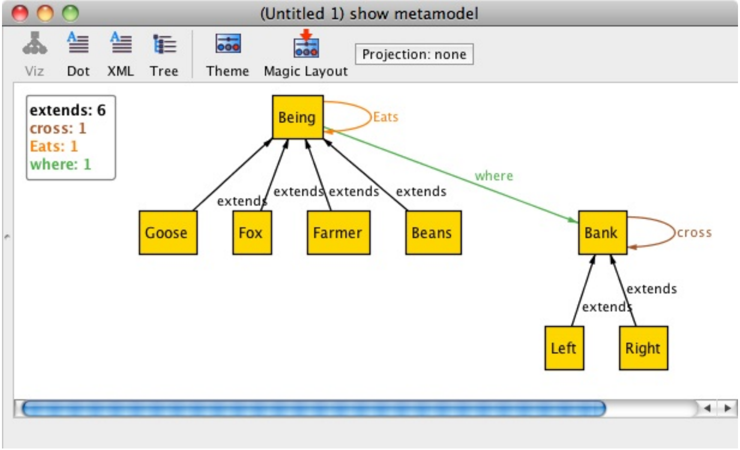
PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Specification source written in Alloy:



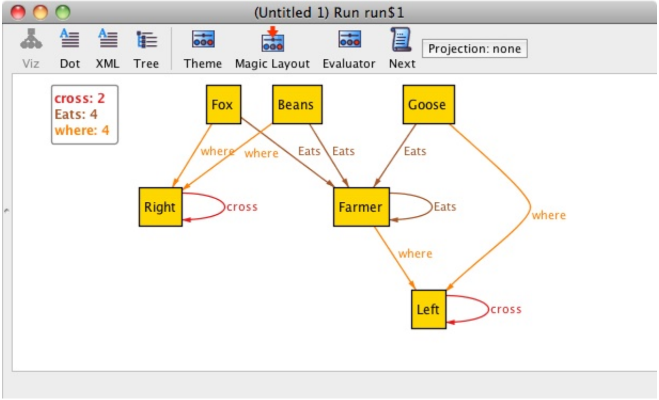
Propositio de homine et capra et lvpo

Diagram of specification (model) given by Alloy:



Propositio de homine et capra et lvpo

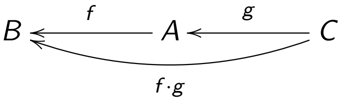
Diagram of instance of the model given by Alloy:



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

Composition

Recall **function composition** (aside).



(5)

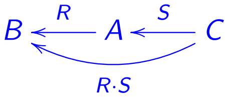
$$b = f(g\ c)$$

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:



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

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

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

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]

$$
\begin{array}{ll}
b(f \cdot g)c & \equiv \langle \exists\ a\ ::\ b\ f\ a \wedge a\ g\ c \rangle \\
& \equiv \qquad \{\ a\ g\ c \text{ means } a = g\ c\ (1)\ \} \\
& \quad \langle \exists\ a\ ::\ a = g\ c \wedge b\ f\ a \rangle \\
& \equiv \qquad \{\ \exists\text{-trading } (221)\ ;\ b\ f\ a \text{ means } b = f\ a\ (1)\ \} \\
& \quad \langle \exists\ a\ :\ a = g\ c :\ b = f\ a \rangle \\
& \equiv \qquad \{\ \exists\text{-one point rule } (225)\ \} \\
& \quad b = f(g\ c)
\end{array}
$$

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

[1] Check the appendix on predicate calculus.

Class 2 — The "Zoo" of Binary Relations

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,

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

transitive

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

and **antisymmetric**:

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

Therefore:

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

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

$$id\ a \;=\; a \tag{13}$$

Clearly, for every R,

$$\perp \;\subseteq\; R \;\subseteq\; \top \tag{14}$$

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

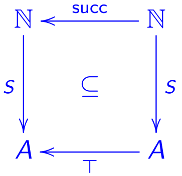
$$
\begin{array}{ll}
 & R \\
\subseteq & \quad \{ \ .... \ \} \\
 & S \\
\subseteq & \quad \{ \ .... \ \} \\
 & R \\
:: & \quad \{ \ \text{"ping-pong"} \ (11) \ \} \\
 & R = S
\end{array}
$$

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

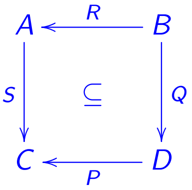
$S \cdot \mathsf{succ} \subseteq \top \cdot S$ depicted by diagram



(onde $\mathsf{succ}\ n = n + 1$) means that numbers are assigned to students sequentially. □

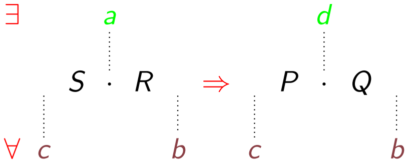
Diagrams ("magic squares")

Pointfree:



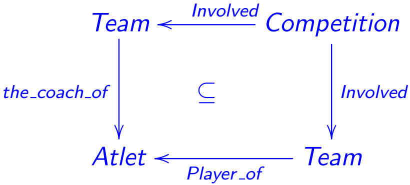
$$S \cdot R \subseteq P \cdot Q$$

Pointwise:



Exercises

Exercise 2: Consider sports competitions involving teams which have atlets (players) and coaches. Follow the rule of the previous slide and spell out the logical meaning of the following *magic square*:



Then express this meaning in natural language, avoiding reading completely through the logic obtained in the previous step. □

Exercises

Exercise 3: Use (6) and (8) and predicate calculus to show that

$$R \cdot id \;=\; R \;=\; id \cdot R \tag{15}$$

$$R \cdot \bot \;=\; \bot \;=\; \bot \cdot R \tag{16}$$

hold and that composition is associative:

$$R \cdot (S \cdot T) = (R \cdot S) \cdot T \tag{17}$$

$\square$

Exercise 4: 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). $\square$

(**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,

$$a(R^\circ)b \quad \equiv \quad b\ R\ a \tag{18}$$

Note that converse commutes with composition

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

and with itself:

$$(R^\circ)^\circ = R \tag{20}$$

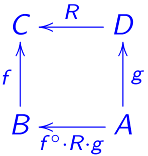
Converse captures the **passive voice**: *Catherine eats the apple* — $R = (eats)$ — is the same as *the apple is eaten by Catherine* — $R^\circ = (is\ eaten\ by)$.

Function converses

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

$$(f\ b)R(g\ a) \quad \equiv \quad b(f^\circ \cdot R \cdot g)a \tag{21}$$

cf. diagram:



Therefore (tell why):

$$b(f^\circ \cdot g)a \quad \equiv \quad f\ b = g\ a \tag{22}$$

Let us see an example of using these rules.

PF-transform at work

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

$$
\begin{array}{ll}
 & f \text{ is } \textbf{injective} \\
\equiv & \qquad \{ \text{ recall definition from discrete maths } \} \\
 & \langle \forall\, y, x \;:\; (f\ y) = (f\ x) :\; y = x \rangle \\
\equiv & \qquad \{ \text{ (22) for } f = g \ \} \\
 & \langle \forall\, y, x \;:\; y(f^\circ \cdot f)x :\; y = x \rangle \\
\equiv & \qquad \{ \text{ (21) for } R = f = g = id \ \} \\
 & \langle \forall\, y, x \;:\; y(f^\circ \cdot f)x :\; y(id)x \rangle \\
\equiv & \qquad \{ \text{ go pointfree (8) i.e. drop } y, x \ \} \\
 & f^\circ \cdot f \subseteq id
\end{array}
$$

The other way round

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

$$
\begin{array}{ll}
& id \subseteq f \cdot f^\circ \\
\equiv & \quad \{ \text{ relational inclusion (8) } \} \\
& \langle \forall\ y, x\ :\ y(id)x :\ y(f \cdot f^\circ)x \rangle \\
\equiv & \quad \{ \text{ identity relation ; composition (6) } \} \\
& \langle \forall\ y, x\ :\ y = x :\ \langle \exists\ z\ ::\ y\ f\ z \wedge z\ f^\circ x \rangle \rangle \\
\equiv & \quad \{ \text{ } \forall\text{-one point (224) ; converse (18) } \} \\
& \langle \forall\ x\ ::\ \langle \exists\ z\ ::\ x\ f\ z \wedge x\ f\ z \rangle \rangle \\
\equiv & \quad \{ \text{ trivia ; function } f \ \} \\
& \langle \forall\ x\ ::\ \langle \exists\ z\ ::\ x = f\ z \rangle \rangle \\
\equiv & \quad \{ \text{ recalling definition from maths } \} \\
& f \text{ is } \textbf{surjective}
\end{array}
$$

Why id (really) matters

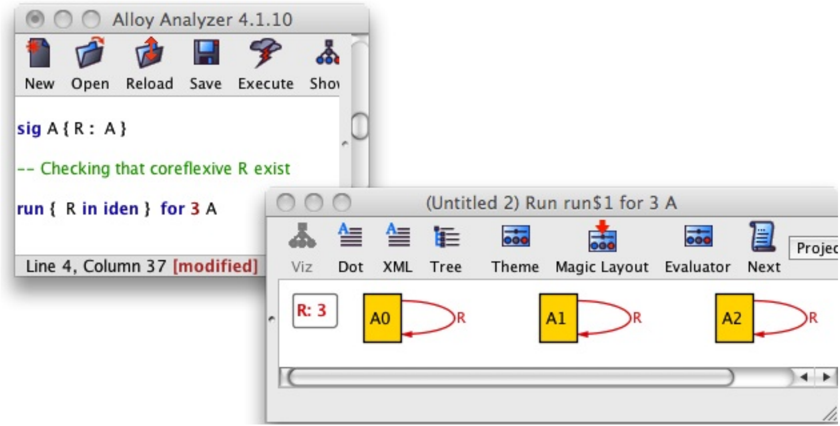
Terminology:

- 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
$A \xleftarrow{\ker R} A$ $\ker R \stackrel{\mathrm{def}}{=} R^\circ \cdot R$	$B \xleftarrow{\mathrm{img}\ R} B$ $\mathrm{img}\ R \stackrel{\mathrm{def}}{=} R \cdot R^\circ$

Alloy: checking for coreflexive relations



Kernels of functions

Meaning of $\ker f$:

$$
\begin{array}{ll}
& a'(\ker f)a \\
\equiv & \{ \text{ substitution } \} \\
& a'(f^\circ \cdot f)a \\
\equiv & \{ \text{ rule (22) } \} \\
& f\ a' = f\ a
\end{array}
$$

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

Exercise 5: Let K be a nonempty data domain, $k \in K$ and $\underline{k}$ be the *"everywhere* k*"* function:

$$
\begin{array}{rcl}
\underline{k} & : & A \to K \\
\underline{k}\ a & = & k
\end{array} \qquad (23)
$$

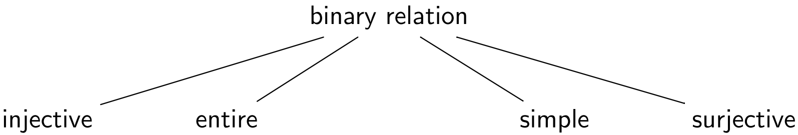
Compute which relations are defined by the following expressions:

$$
\ker \underline{k}, \quad \underline{b} \cdot \underline{c}^\circ, \quad \mathrm{img}\ \underline{k} \qquad (24)
$$

□

Binary relation taxonomy

Topmost criteria:



Definitions:

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

(25)

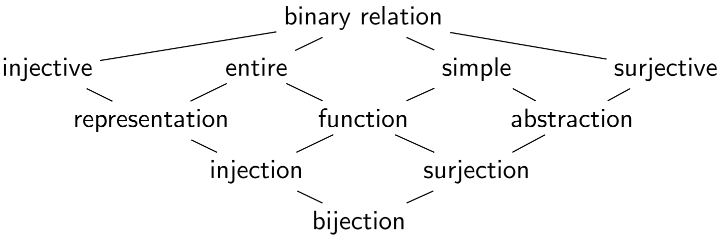
Facts:

$$\ker (R^\circ) = \operatorname{img} R \quad (26)$$

$$\operatorname{img} (R^\circ) = \ker R \quad (27)$$

Binary relation taxonomy

The whole picture:



(28)

Exercise 6: Resort to (26,27) and (25) 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

A lone -> B	A -> some B	A -> lone B	A some -> B
injective	entire	simple	surjective

A lone -> some B	A -> one B	A some -> lone B
representation	function	abstraction

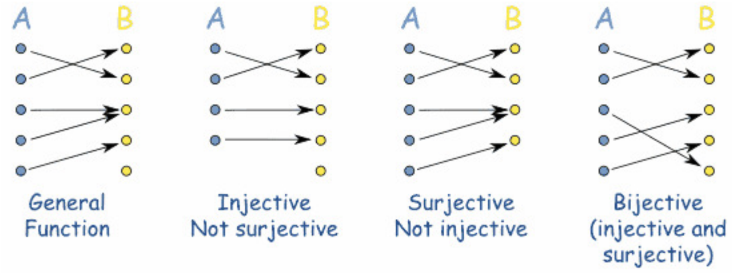
A lone -> one B	A some -> one B
injection	surjection

A one -> one B
bijection

(Courtesy of Alcino Cunha.)

Exercises

Exercise 7: 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 8: Prove the following fact

$$\textit{A function } f \textit{ is a bijection } \textbf{iff} \textit{ its converse } f^{\circ} \textit{ is a function} \qquad (29)$$

by completing:

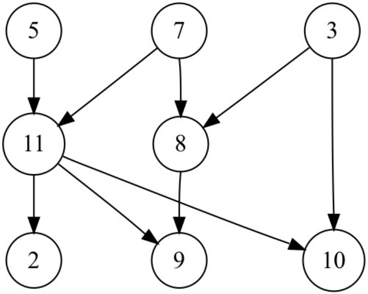
$$
\begin{array}{ll}
 & f \text{ and } f^{\circ} \text{ are functions} \\
\equiv & \quad \{ \ \dots \ \} \\
 & (id \subseteq \ker f \wedge \operatorname{img} f \subseteq id) \wedge (id \subseteq \ker (f^{\circ}) \wedge \operatorname{img} (f^{\circ}) \subseteq id) \\
\equiv & \quad \{ \ \dots \ \} \\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

□

Taxonomy using matrices

Recall that 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

Taxonomy using matrices

- **entire** — at least one 1 in every column (30)
- **surjective** — at least one 1 in every row (31)
- **simple** — at most one 1 in every column (32)
- **injective** — at most one 1 in every row (33)
- **bijective** — exactly one 1 in evey column and every row. (34)

PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Exercise 9: 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 10: 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 11: Relation $where : Being \to Bank$ 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**. □

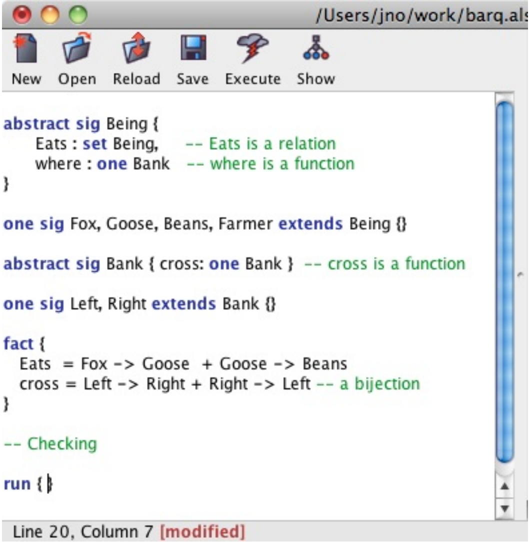
Exercise 12: There are only two **constant** functions (23) in the type $Being \longrightarrow Bank$ of $where$. Identify them and explain their role in the puzzle. □

Exercise 13: Two functions f and g are bijections iff $f^\circ = g$, recall (29). 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.)



Class 3 — Functions

Functions in one slide

As seen before, 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:

$$f \cdot R \subseteq S \equiv R \subseteq f^{\circ} \cdot S \tag{35}$$

$$R \cdot f^{\circ} \subseteq S \equiv R \subseteq S \cdot f \tag{36}$$

Equality rule:

$$f \subseteq g \equiv f = g \equiv f \supseteq g \tag{37}$$

Rule (37) follows from (35,36) by "cyclic inclusion" (next slide).

Proof of functional equality rule (37)

$f \subseteq g$

$\equiv$ { identity }

$f \cdot id \subseteq g$

$\equiv$ { shunting on f }

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

$\equiv$ { shunting on g }

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

$\equiv$ { converses; identity }

$g \subseteq f$

Then:

$f = g$

$\equiv$ { cyclic inclusion (11)

$f \subseteq g \wedge g \subseteq f$

$\equiv$ { aside }

$f \subseteq g$

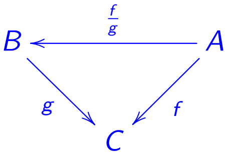
$\equiv$ { aside }

$g \subseteq f$

$\square$

Dividing functions

$$\frac{f}{g} = g^\circ \cdot f \qquad cf. \tag{38}$$



Exercise 14: Check the properties:

$$\frac{f}{id} = f \tag{39}$$

$$\frac{f \cdot h}{g \cdot k} = k^\circ \cdot \frac{f}{g} \cdot h \tag{40}$$

$$\frac{f}{f} = \ker f \tag{41}$$

$$\left(\frac{f}{g}\right)^\circ = \frac{g}{f} \tag{42}$$

□

Exercise 15: Infer $id \subseteq \ker f$ (f is total) and $\operatorname{img} f \subseteq id$ (f is simple) from the shunting rules (35) or (36). □

Dividing functions

By (21) we have:

$$b \frac{f}{g} a \quad \equiv \quad g\ b = f\ a \tag{43}$$

How useful is this? Think of the following sentence:

Mary lives where John was born.

By (43), this can be expressed by a division:

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

In general,

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

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 some: $\ker R$ 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 $id \subseteq R$	(44)
coreflexive:	iff $R \subseteq id$	(45)

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

transitive:	iff $R \cdot R \subseteq R$	(46)
symmetric:	iff $R \subseteq R^\circ (\equiv R = R^\circ)$	(47)
anti-symmetric:	iff $R \cap R^\circ \subseteq id$	(48)
irreflexive:	iff $R \cap id = \bot$	
connected:	iff $R \cup R^\circ = \top$	(49)

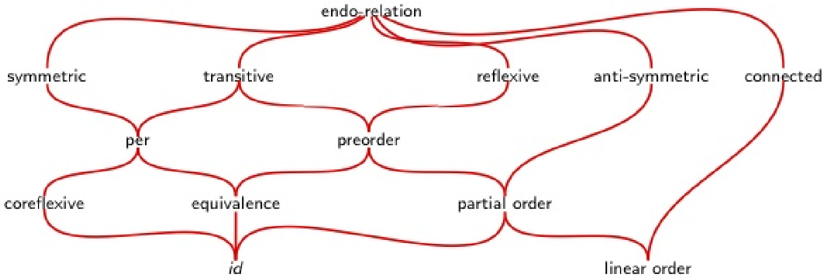
where, in general, for R, S of the same type:

$$b\ (R \cap S)\ a \equiv b\ R\ a \wedge b\ S\ a \tag{50}$$

$$b\ (R \cup S)\ a \equiv b\ R\ a \vee b\ S\ a \tag{51}$$

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: $y\ IsBrotherOf\ x$.

[3] Kernels of functions are always equivalence relations, see exercise 23.

Exercises

Exercise 16: Consider the relation

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

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

Exercise 17: Check which of the following properties,

transitive, symmetric, anti-symmetric, connected

hold for the relation *Eats* of exercise 10. □

Exercises

Useful:

A difunctional relation that is reflexive and symmetric necessarily is an equivalence relation.

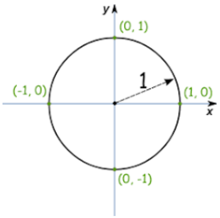
Exercise 18: Let $\mathbb{R} \xleftarrow{R} \mathbb{R}$ be the binary relation that defines the unit circunference,

$$y \; R \; x \quad \stackrel{\mathrm{def}}{=} \quad y^2 + x^2 = 1 \tag{52}$$

that is,

$$R = \frac{(1-) \cdot sq}{sq} \tag{53}$$

where $sq : \mathbb{R} \to \mathbb{R}$ and $(1-) : \mathbb{R} \to \mathbb{R}$ are the functions $y = x^2$ e $y = 1 - x$, respectively.



Without using (52), show that R is **symmetric**. □

Exercises

Exercise 19: A relation R is said to be **co-transitive** or **dense** iff the following holds:

$$\langle \forall\, b, a \;:\; b\; R\; a : \langle \exists\, c \;:\; b\; R\; c : \; c\; R\; a \rangle\rangle \tag{54}$$

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

Exercise 20: Expand criteria (46) to (49) to pointwise notation. □

Exercises

Exercise 21: The teams (T) of a football league play games (G) at home or away, and every game takes place in some date:

$$\begin{array}{ccccc} T & \xleftarrow{\ home\ } & G & \xrightarrow{\ away\ } & T \\ & & \downarrow{\scriptstyle date} & & \\ & & D & & \end{array}$$

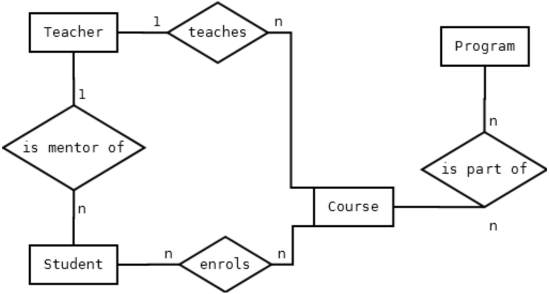
Moreover, *(a) No team can play two games on the same date; (b) All teams play against each other but not against themselves; (c) For each home game there is another game away involving the same two teams.* Show that

$$id \subseteq \frac{away}{home} \cdot \frac{away}{home} \tag{55}$$

captures one of the requirements above (which?) and that (55) amounts to forcing $home \cdot away^{\circ}$ to be symmetric. □

Formalizing ER diagrams

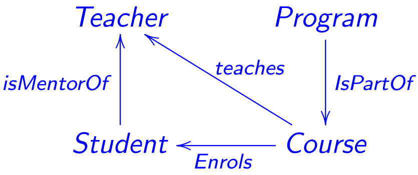
So-called "**Entity-Relationship**" (ER) diagrams are commonly used to capture relational information, e.g.[4]



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



(56)

Exercise 22: Looking at diagram (56),

- Specify, in the relational pointfree style, the property: *mentors of students necessarily are among their teachers*.
- Why is

$$\frac{teaches}{isMentorOf} \subseteq Enrols$$

  inadequate as answer to the previous question?

□

Class 4 – Meet and Join

Meet and join

Recall **meet** (intersection) and **join** (union), introduced by (50) and (51), respectively.

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

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

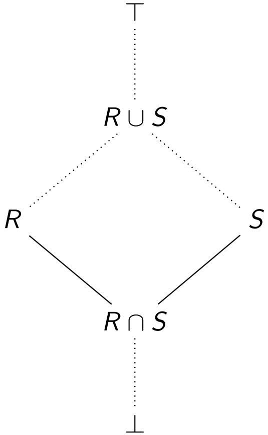
$$X \subseteq R \cap S \equiv X \subseteq R \wedge X \subseteq S \qquad (57)$$

$$R \cup S \subseteq X \equiv R \subseteq X \wedge S \subseteq X \qquad (58)$$

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

Taking (57) as example:

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

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

$$\left\{ \begin{array}{l} R \cap S \subseteq R \\ R \cap S \subseteq S \end{array} \right. \qquad (59)$$

Right cancellation ($X := R$ or $X := S$):

$$R = R \cap S \equiv R \subseteq S \qquad (60)$$

Indirect equality

Universal properties such as e.g. (57,58) blend nicely with the so-called **indirect equality** way of proving relation equality:

Indirect equality *rules:*

$$R = S \equiv \langle \forall\ X :: (X \subseteq R \equiv X \subseteq S)\rangle \tag{61}$$

$$\equiv \langle \forall\ X :: (R \subseteq X \equiv S \subseteq X)\rangle \tag{62}$$

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.

$$\left\{\begin{array}{ll} & X \subseteq R \\ \equiv & \quad \{ \ \ldots \ \} \\ & X \subseteq \ldots \\ \equiv & \quad \{ \ \ldots \ \} \\ & X \subseteq S \\ :: & \quad \{ \text{ indirect equality (61) } \} \\ & R = S \\ \Box & \end{array}\right.$$

How universal properties help

$R \cap \top = R$.

Why?

Again recall (57)

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

and use **indirect equality** (aside):

$$\begin{array}{ll} & X \subseteq R \cap \top \\ \equiv & \{ \text{ universal property (57) } \} \\ & \left\{ \begin{array}{l} X \subseteq R \\ X \subseteq \top \end{array} \right. \\ \equiv & \{ \top \text{ is above anything } \} \\ & X \subseteq R \\ :: & \{ \text{ indirect equality (61) } \} \\ & R \cap \top = R \end{array}$$

How universal properties help

Other expected properties of meet and join can also be inferred by **indirect equality**.

Take **associativity**

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

as example and follow the reasonig aside.

$$
\begin{array}{ll}
& X \subseteq (R \cap S) \cap Q \\
\equiv & \qquad \{ \cap\text{-universal (57) twice} \} \\
& (X \subseteq R \wedge X \subseteq S) \wedge X \subseteq Q \\
\equiv & \qquad \{ \wedge \text{ is associative} \} \\
& X \subseteq R \wedge (X \subseteq S \wedge X \subseteq Q) \\
\equiv & \qquad \{ \cap\text{-universal (57) twice} \} \\
& X \subseteq R \cap (S \cap Q) \\
:: & \qquad \{ \text{indirection (61)} \} \\
& (R \cap S) \cap Q = R \cap (S \cap Q) \\
\Box &
\end{array}
$$

Distributivity

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

$$R \cdot (S \cup Q) \quad = \quad (R \cdot S) \cup (R \cdot Q) \tag{63}$$

$$(S \cup Q) \cdot R \quad = \quad (S \cdot R) \cup (Q \cdot R) \tag{64}$$

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

$$(S \cap Q) \cdot R = (S \cdot R) \cap (Q \cdot R) \quad \Leftarrow \quad \left\{ \begin{array}{c} Q \cdot \mathrm{img}\ R \subseteq Q \\ \vee \\ S \cdot \mathrm{img}\ R \subseteq S \end{array} \right. \tag{65}$$

$$R \cdot (Q \cap S) = (R \cdot Q) \cap (R \cdot S) \quad \Leftarrow \quad \left\{ \begin{array}{c} (\ker\ R) \cdot Q \subseteq Q \\ \vee \\ (\ker\ R) \cdot S \subseteq S \end{array} \right. \tag{66}$$

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:*

$$CanEat \subseteq SameBank \cdot \underline{Farmer} \tag{67}$$

Propositio de homine et capra et lvpo

By (35), "starvation" property (67) converts to:

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

In this version, (67) can be depicted as a 'magic square':

$$\begin{array}{ccc} \mathit{Being} & \xleftarrow{\mathit{CanEat}} & \mathit{Being} \\ {\scriptstyle \mathit{where}}\downarrow & \subseteq & \downarrow{\scriptstyle \underline{\mathit{Farmer}}} \\ \mathit{Bank} & \xleftarrow[\mathit{where}]{} & \mathit{Being} \end{array} \qquad (68)$$

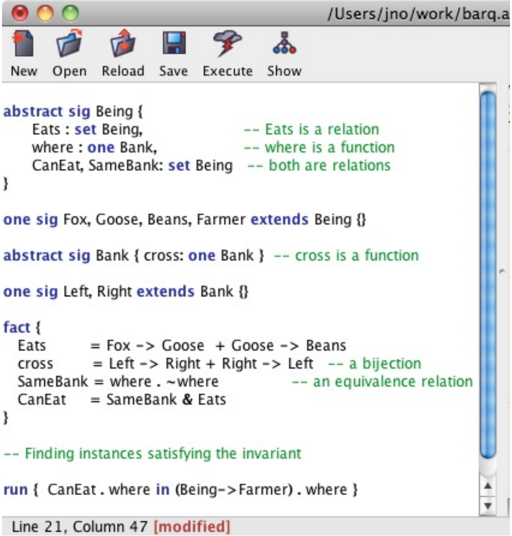
This "reads" in a nice way:

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

PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Properties which — such as (68) — are desirable and must **always hold** are called **invariants**.

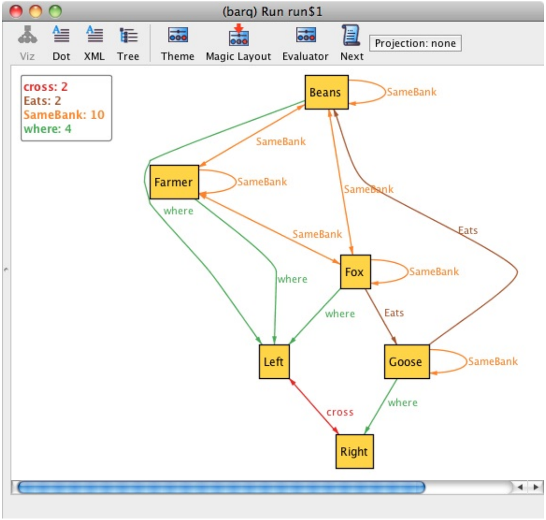
See aside the 'starvation' invariant (68) written in **Alloy**.



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

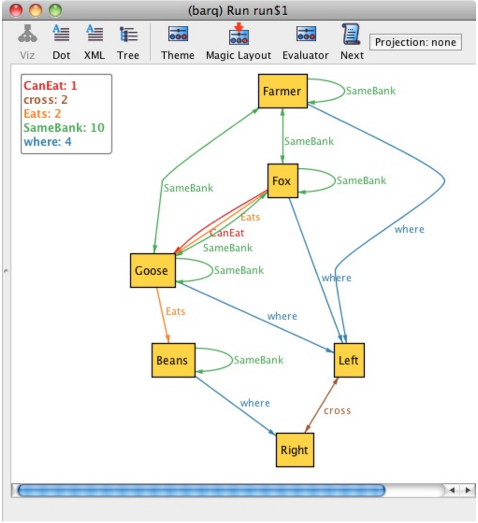


Propositio de homine et capra et lvpo

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 $SameBank = \ker\ where$. Then *SameBank* is an **equivalence relation** by the exercise below.

Exercise 23: Knowing that property

$$f \cdot f^{\circ} \cdot f = f \tag{69}$$

holds for every function f (to be shown later), prove that $\ker f = \frac{f}{f}$ (41) 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.

Monotone reasoning

Monotonicity

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

$$R \subseteq S \quad \Rightarrow \quad R^\circ \subseteq S^\circ \tag{70}$$

$$R \subseteq S \land U \subseteq V \quad \Rightarrow \quad R \cdot U \subseteq S \cdot V \tag{71}$$

$$R \subseteq S \land U \subseteq V \quad \Rightarrow \quad R \cap U \subseteq S \cap V \tag{72}$$

$$R \subseteq S \land U \subseteq V \quad \Rightarrow \quad R \cup U \subseteq S \cup V \tag{73}$$

etc hold.

Transitivity, recall:

$$R \subseteq S \land S \subseteq Q \Rightarrow R \subseteq Q$$

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**

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

  and then proceed with $R \subseteq M$.

Proofs by $\subseteq$-transitivity

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

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

and then proceed with $M \subseteq S$.

Example

Composition of simple $A \xrightarrow{S} B$ and $B \xrightarrow{R} C$ is simple:

$$
\begin{array}{ll}
& \mathrm{img}\,(R \cdot S) \subseteq id \\
\equiv & \quad \{ \mathrm{img}\, R = R \cdot R^{\circ};\ \text{converses (19)} \} \\
& R \cdot S \cdot S^{\circ} \cdot R^{\circ} \subseteq id \\
\Leftarrow & \quad \{ S \text{ is simple, } S \cdot S^{\circ} \subseteq id;\ \text{rule B} \} \\
& R \cdot R^{\circ} \subseteq id \\
\Leftarrow & \quad \{ R \text{ is simple, } R \cdot R^{\circ} \subseteq id;\ \text{rule B} \} \\
& id \subseteq id \\
\equiv & \quad \{ R \subseteq R \text{ always holds} \} \\
& true
\end{array}
$$

Example

Proof of shunting rule (35):

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

Thus the equivalence in (35) is established by circular implication.

Exercises (monotonicity and transitivity)

Exercise 24: 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 25: Prove that relational **composition** preserves **all** relational classes in the taxonomy of (28). □

Meaning of $f \cdot r = id$

On the one hand,

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

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

---

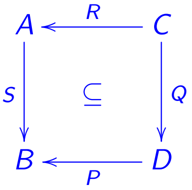
f *is an* **abstraction** *and* r *is a* **representation**

Exercise 26: Why are π_1 and π_2 **surjective**? And why are i_1 and i_2 **injective**? Why are isomorphisms **bijections**? □

Exploring “magic squares”

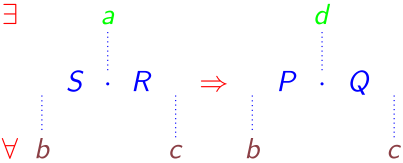
'Magic square'

Recall



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

... i.e. the pointwise:



Converse magic squares

$$
\begin{array}{ccc} A & \xleftarrow{\;R\;} & C \\ {\scriptstyle P}\big\downarrow & \subseteq & \big\downarrow{\scriptstyle Q} \\ B & \xleftarrow[\;S\;]{} & D \end{array}
\quad\equiv\quad
\begin{array}{ccc} A & \xleftarrow{\;P^\circ\;} & B \\ {\scriptstyle R^\circ}\big\downarrow & \subseteq & \big\downarrow{\scriptstyle S^\circ} \\ C & \xleftarrow[\;Q^\circ\;]{} & D \end{array}
\qquad (74)
$$

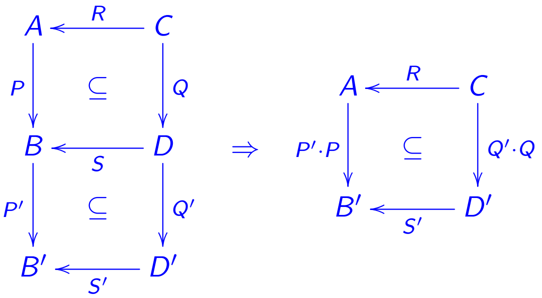
Magic square compositionality

Magic squares compose, not only **horizontally**

$$\begin{array}{ccccc} A & \xleftarrow{\;R\;} & C & \xleftarrow{\;R'\;} & C' \\ {\scriptstyle P}\downarrow & \subseteq & \downarrow{\scriptstyle Q} & \subseteq & \downarrow{\scriptstyle Q'} \\ B & \xleftarrow[\;S\;]{} & D & \xleftarrow[\;S'\;]{} & D' \end{array} \quad \Rightarrow \quad \begin{array}{ccc} A & \xleftarrow{\;R\cdot R'\;} & C' \\ {\scriptstyle P}\downarrow & \subseteq & \downarrow{\scriptstyle Q'} \\ B' & \xleftarrow[\;S\cdot S'\;]{} & D' \end{array} \qquad (75)$$

Magic square compositionality

… but also **vertically**:



(76)

Exercise 27: Prove (75) and (76). □

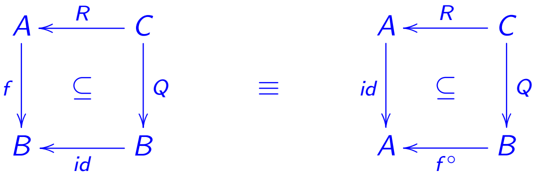
Exercise 28: Use (76) to prove that the compostion of monotonic functions is a monotonic function. □

Shunting rules as magic squares

Recall *shunting rule* (35)

$$f \cdot R \subseteq Q \quad \equiv \quad R \subseteq f^\circ \cdot Q$$

and compare with:

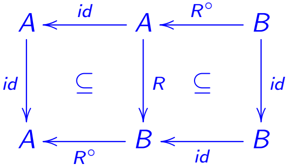


Exercise 29: Draw the magic squares of the other shunting rule (36).

□

Exercises

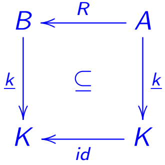
Exercise 30: What do the following magic squares tell us about relation R?



□

Exercise 31:

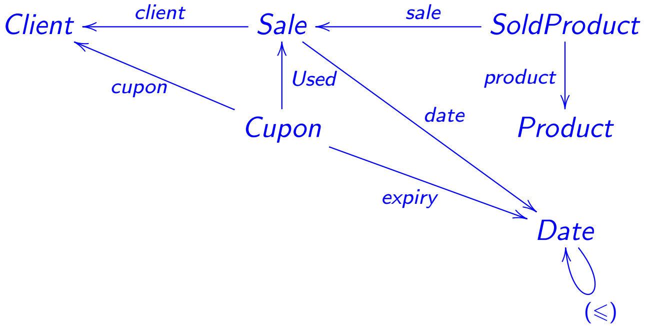
The square aside captures an important property of constant functions. Apply (graphically) the shunting rules and conclude that $\ker \underline{k} = \top$.



(77)

□

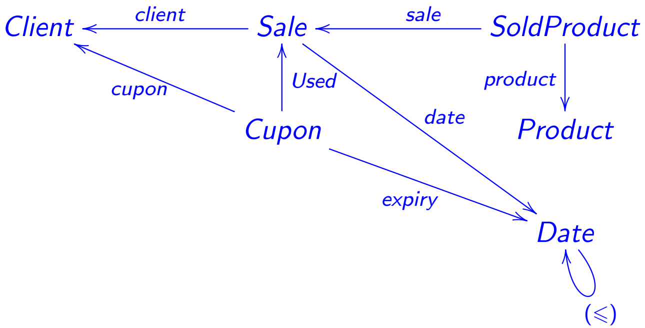
"D. Acácia grocery"



Find "magic square" for property:

Coupons cannot be used beyond their expiry date.

"D. Acácia grocery"

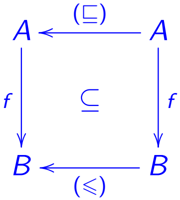


Find "magic square" for property:

Coupons can only be used by clients who own them.

‘Magic square’ patterns

Now consider the special case



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

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

‘Magic square’ patterns

Do we need...

$\exists \quad a \qquad\qquad b'$

$$f \cdot (\sqsubseteq) \Rightarrow (\leqslant) \cdot f$$

$\forall \quad b \qquad a' \qquad b \qquad a'$

as before?

‘Magic square’ patterns

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

$$
\begin{array}{ll}
& f \cdot (\sqsubseteq) \ \subseteq \ (\leqslant) \cdot f \\
\equiv & \quad \{ \ (35) \ \} \\
& (\sqsubseteq) \ \subseteq \ f^{\circ} \cdot (\leqslant) \cdot f \\
\equiv & \quad \{ \ (21) \ \} \\
& \langle \forall \ a, a' \ : \ a \sqsubseteq a' : \ f \ a \leqslant f \ a' \rangle
\end{array}
$$

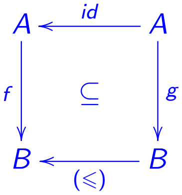
In summary,

$$f \cdot (\sqsubseteq) \ \subseteq \ (\leqslant) \cdot f \qquad (78)$$

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

'Magic square' patterns

Now consider yet another special case:



$$f \subseteq (\leqslant) \cdot g \qquad (79)$$

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

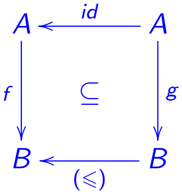
$$\langle \forall\ a\ ::\ f\ a \leqslant g\ a \rangle$$

meaning that

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

'Magic square' patterns

Now consider yet another special case:



$$f \subseteq (\leqslant) \cdot g \qquad (79)$$

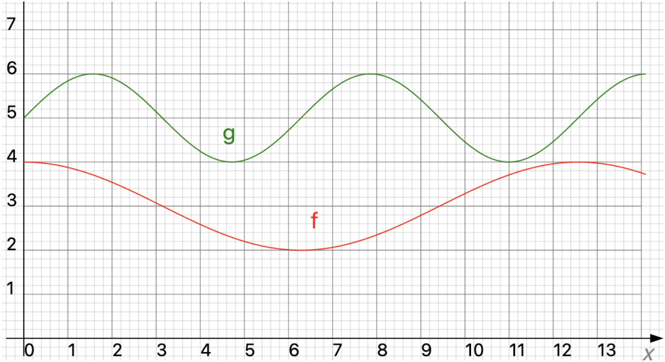
Likewise, $f \subseteq (\leqslant) \cdot g$ will unfold to

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

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 32:

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 33: As generalization of exercise 1, draw the most general "magic square" that accommodates relational assertion:

$$M \cdot R^\circ \subseteq \top \cdot M \tag{80}$$

□

Exercise 34: Type the following relational assertions

$$M \cdot N^\circ \subseteq \bot \tag{81}$$

$$M \cdot N^\circ \subseteq id \tag{82}$$

$$M^\circ \cdot \top \cdot N \subseteq > \tag{83}$$

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

Exercises

Exercise 35: Let $bag : A^* \to \mathbb{N}_0{}^A$ be the function that, given a finite sequence (list) indicates the number of occurrences of its elements, for instance,

$$bag\ [a, b, a, c]\ a = 2$$
$$bag\ [a, b, a, c]\ b = 1$$
$$bag\ [a, b, a, c]\ c = 1$$

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

$$S = \frac{bag}{bag} \cap \frac{true}{ordered} \tag{84}$$

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

Exercises

Exercise 36: Prove the **union simplicity** rule:

$$M \cup N \text{ is simple} \quad \equiv \quad M, N \text{ are simple and } M \cdot N^\circ \subseteq id \tag{85}$$

□

Exercise 37: Derive from (85) the corresponding rule for **injective** relations. □

Exercise 38: Explain in your own words the following equalities:

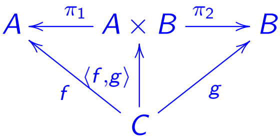
$$1 \xleftarrow{\top} 1 \;=\; 1 \xleftarrow{!} 1 \;=\; id \tag{86}$$

□

Class 5 — Pairs and sums

Relational pairing

Recall:



$$\langle f, g \rangle\ c = (f\ c, g\ c) \qquad (87)$$

Clearly:

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

$$\equiv \qquad \{\ \langle f, g \rangle\ c = (f\ c, g\ c)\ (87)\ ;\ \text{equality of pairs}\ \}$$

$$\left\{ \begin{array}{l} a = f\ c \\ b = g\ c \end{array} \right.$$

$$\equiv \qquad \{\ y = f\ x\ \equiv\ y\ f\ x\ \}$$

$$\left\{ \begin{array}{l} a\ f\ c \\ b\ g\ c \end{array} \right.$$

Relational pairing

That is:

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

This suggests the generalization

$$(a, b)\ \langle R, S\rangle\ c\ \equiv\ a\ R\ c \wedge b\ S\ c \tag{88}$$

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

$$R \times S = \langle R \cdot \pi_1, S \cdot \pi_2\rangle \tag{89}$$

(89) unfolds to the pointwise:

$$(b, d)(R \times S)(a, c)\quad \equiv\quad b\ R\ a \wedge d\ S\ c \tag{90}$$

Relational pairing example (in matrix layout)

Example — given relations

$where^\circ$ =

	Left	*Right*
Fox	1	0
Goose	0	1
Beans	0	1

and $cross$ =

	Left	*Right*
Left	0	1
Right	1	0

pairing them up evaluates to:

$\langle where^\circ, cross \rangle$ =

	Left	*Right*
(*Fox*, *Left*)	0	0
(*Fox*, *Right*)	1	0
(*Goose*, *Left*)	0	1
(*Goose*, *Right*)	0	0
(*Beans*, *Left*)	0	1
(*Beans*, *Right*)	0	0

Exercises

Exercise 39: Show that

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

PF-transforms to:

$$\langle R, S\rangle \ = \ \pi_1^\circ \cdot R \cap \pi_2^\circ \cdot S \qquad (91)$$

Then infer universal property

$$X \subseteq \langle R, S\rangle \quad \equiv \quad \pi_1 \cdot X \subseteq R \ \wedge \ \pi_2 \cdot X \subseteq S \qquad (92)$$

from (91) via indirect equality (61). □

Exercise 40: What can you say about (92) in case X, R and S are functions? □

Exercises

Exercise 41: 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). $\square$

Exercise 42: Derive from

$$
\langle R, S \rangle^{\circ} \cdot \langle X, Y \rangle = (R^{\circ} \cdot X) \cap (S^{\circ} \cdot Y) \quad (93)
$$

the following properties:

$$
\ker \langle R, S \rangle = \ker R \cap \ker S \quad (94)
$$

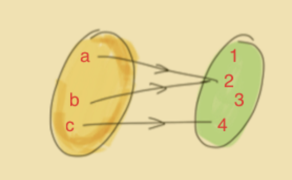
$\square$

Injectivity preorder

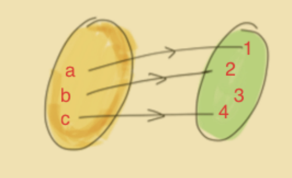
$\ker R = R^\circ \cdot R$ *measures* the level of **injectivity** of R according to the preorder $(\leqslant)$ defined by

$$R \leqslant S \quad \equiv \quad \ker S \subseteq \ker R \qquad (95)$$

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

$$! \leqslant f \leqslant id$$

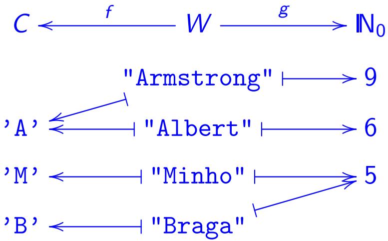
Also easy to show:

$$id \leqslant f \quad \equiv \quad f \text{ is injective} \qquad (96)$$

Exercise 43: 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.



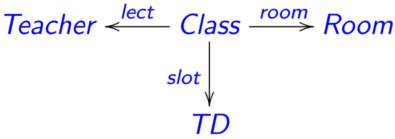
□

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



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:

$$
\begin{array}{ll}
& room \leqslant \langle lect, slot \rangle \\
\equiv & \{ \ (95) \ \} \\
& \ker \langle lect, slot \rangle \ \subseteq \ \ker room \\
\equiv & \{ \ (94) \ ; (41) \ \} \\
& \dfrac{lect}{lect} \cap \dfrac{slot}{slot} \subseteq \dfrac{room}{room} \\
\equiv & \{ \ \text{going pointwise, for all } c_1, c_2 \in Class \ \} \\
& \left\{ \begin{array}{l} lect\ c_1 = lect\ c_2 \\ slot\ c_1 = slot\ c_2 \end{array} \right. \Rightarrow room\ c_1 = room\ c_2
\end{array}
$$

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:

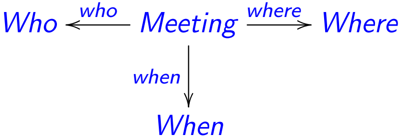
- $room$ is **dependent** on $lect$ and $slot$, i.e.
- $lect$ and $slot$ **determine** $room$.

Generalization: the "agenda design pattern"

Nobody can be in different places at the same time

$$where \leqslant \langle who, when \rangle$$

in the context of the generic data model:



Exercise 44: Do $who \leqslant \langle where, when \rangle$ and $when \leqslant \langle who, where \rangle$ 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***.*

Relational injectivity

Universal property:

$$\langle R, S\rangle \leqslant X \quad\equiv\quad R \leqslant X \wedge S \leqslant X \tag{97}$$

Cancellation of (97) means that *pairing* always *increases injectivity*:

$$R \leqslant \langle R, S\rangle \quad \text{and} \quad S \leqslant \langle R, S\rangle. \tag{98}$$

(98) unfolds to $\ker \langle R, S\rangle \ \subseteq\ (\ker R) \cap (\ker S)$, confirming (94).

Injectivity **shunting law**:

$$R \cdot g \leqslant S \quad\equiv\quad R \leqslant S \cdot g^{\circ} \tag{99}$$

Exercises

Exercise 45: $\langle R, id \rangle$ is always *injective* — why? □

Exercise 46: Let f and g be given such that

$$f \cdot g \cdot f = f \tag{100}$$

holds. Show that:

$$f \cdot g = id \quad \Leftarrow \quad f \text{ surjective} \tag{101}$$

$$g \cdot f = id \quad \Leftarrow \quad f \text{ injective} \tag{102}$$

Hint: recall (37) among other required laws. □

Relation pairing continued

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

$$\langle R, S\rangle \cdot Q = \langle R \cdot Q, S \cdot Q\rangle \quad \Leftarrow \quad \left\{ \begin{array}{l} R \cdot \mathrm{img}\ Q \ \subseteq \ R \\ \qquad \vee \\ S \cdot \mathrm{img}\ Q \ \subseteq \ S \end{array} \right. \tag{103}$$

However, the **absorption** law

$$(R \times S) \cdot \langle P, Q\rangle = \langle R \cdot P, S \cdot Q\rangle \tag{104}$$

holds unconditionally.

Exercises

Exercise 47: Recalling (29), prove that

$$swap = \langle \pi_2, \pi_1 \rangle \tag{105}$$

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

Exercise 48: Derive from the laws of pairing studied thus far the following facts about relational product:

$$id \times id = id \tag{106}$$

$$(R \times S) \cdot (P \times Q) = (R \cdot P) \times (S \cdot Q) \tag{107}$$

□

Exercise 49: Show that (103) holds. Suggestion: recall (65). From this infer that no side-condition is required for T simple. □

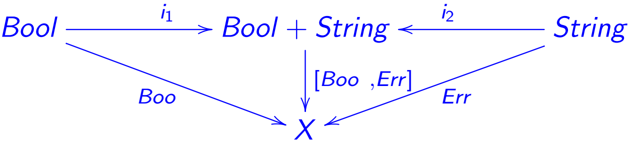
Class 6 — Sums

Relational sums

Example (Haskell):

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

PF-transforms to

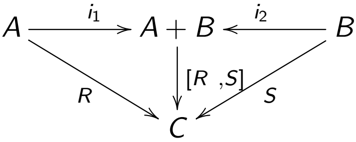


(108)

where

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

cf.



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

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:

$$[R\ ,S] = X \quad \equiv \quad R = X \cdot i_1 \ \wedge\ S = X \cdot i_2 \qquad (109)$$

Thus $[i_1\ ,i_2] = id$ — solve (109) for R and S when $X = id$, etc etc.

Divide and conquer

The property for sums (coproducts) corresponding to (93) for products is:

$$[R\ , S] \cdot [Q\ , U]^\circ \quad = \quad (R \cdot Q^\circ) \cup (S \cdot U^\circ) \tag{110}$$

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

Exercise 50: Show that:

$$\mathrm{img}\ [R\ , S] \quad = \quad \mathrm{img}\ R \cup \mathrm{img}\ S \tag{111}$$

$$\mathrm{img}\ i_1 \cup \mathrm{img}\ i_2 \quad = \quad id \tag{112}$$

□

Exercises

Exercise 51: 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:

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

Show that the relation

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

is a function. □

Exercises

Exercise 52: Consider the following definition of a relation $A \xleftarrow{R} A^*$,

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

where

$$\mathsf{in} = [\mathsf{nil}\ , \mathsf{cons}] \quad (113)$$

$$\mathsf{nil}\ \_ = [\,] \quad (114)$$

$$\mathsf{cons}\ (h, t) = h : t \quad (115)$$

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

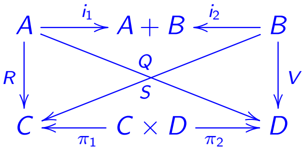
(b) Based on this, spell out the meaning of $a\ R\ x$ in you own words. □

$+$ meets $\times$

The **exchange law**

$$[\langle R, S\rangle , \langle Q, V\rangle] \;=\; \langle [R \;, Q], [S \;, V]\rangle \tag{116}$$

holds for all relations as in diagram



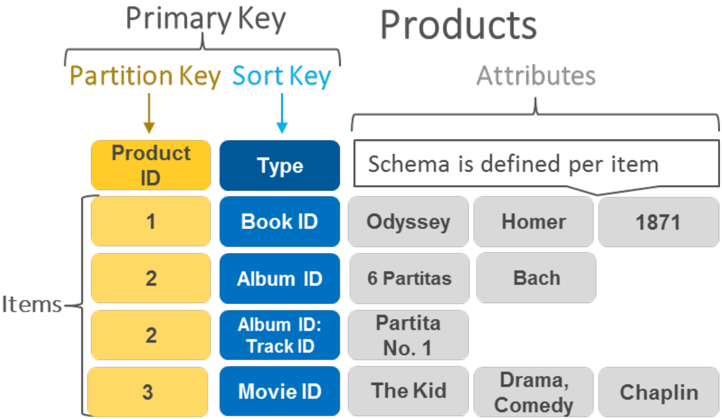
and the **fusion** law

$$\langle R, S\rangle \cdot f \;=\; \langle R \cdot f, S \cdot f\rangle \tag{117}$$

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

Exercise 53: Relying on both (109) and (117) prove (116). □

On key-value (KV) data models



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**.

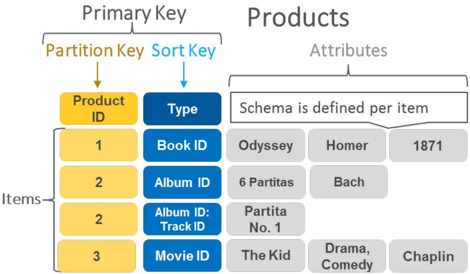
No-SQL, **columnar** *database trend.*

Example above:

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

On key-value data models

"Schema is defined per item"...



In this example:

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

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

Magic square sums

Exercise 54: Prove (118) below.

$$
\begin{array}{c}
\begin{array}{ccc}
A & \xleftarrow{R} & C \\
{\scriptstyle P}\downarrow & \subseteq & \downarrow{\scriptstyle Q} \\
B & \xleftarrow[S]{} & D
\end{array} \\
+ \\
\begin{array}{ccc}
A' & \xleftarrow{R'} & C' \\
{\scriptstyle P'}\downarrow & \subseteq & \downarrow{\scriptstyle Q'} \\
B' & \xleftarrow[S]{} & D'
\end{array}
\end{array}
\quad = \quad
\begin{array}{ccc}
A + A' & \xleftarrow{R+R'} & C + C' \\
{\scriptstyle P+P'}\downarrow & \subseteq & \downarrow{\scriptstyle Q+Q'} \\
B + B' & \xleftarrow[S+S']{} & D + D'
\end{array}
\qquad (118)
$$

□

Class 7 — Relational division (and so on)

Relation division — motivation

Recall the algorithm of **whole** (**integer**) division:

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

How does one **specify** such an algorithm?

Back to the primary school desk

The **whole division** algorithm

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

However

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

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

That is:

$$\begin{array}{c|c} x & y \\ \hline \ldots & x \div y \end{array} \qquad z \times y \leqslant x \Rightarrow z \leqslant x \div y$$

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

Back to the primary school desk

On the other hand,

$$z \leqslant x \div y \Rightarrow z \times y \leqslant x$$

For instance:

$$2 \leqslant 7 \div 2 \Rightarrow 2 \times 2 = 4 \leqslant 7$$

Altogether:

$$\begin{array}{c|c} x & y \\ \hline \ldots & x \div y \end{array} \qquad z \times y \leqslant x \;\equiv\; z \leqslant x \div y$$

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

Note the **equivalence**.

Back to the primary school desk

On the other hand,

$$z \leqslant x \div y \Rightarrow z \times y \leqslant x$$

For instance:

$$2 \leqslant 7 \div 2 \Rightarrow 2 \times 2 = 4 \leqslant 7$$

Altogether:

$$\begin{array}{c|c} x & y \\ \hline \ldots & x \div y \end{array} \qquad z \times y \leqslant x \;\equiv\; z \leqslant x \div y$$

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

Note the **equivalence**.

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** *that multiplied by* y *approximates* x,

---

also

$$Z \cdot Y \subseteq X \;\equiv\; Z \subseteq X / Y \qquad (119)$$

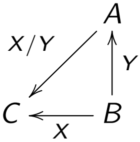
means that X / Y is the largest **relation** which pre-composed with Y approximates X.

What is the pointwise meaning of X / Y?

We reason:

First, the types:

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



Next, the calculation:

$$\begin{array}{cl} & c\ (X/Y)\ a \\ \equiv & \{\ \text{introduce points } C \xleftarrow{\underline{c}} 1 \text{ and } A \xleftarrow{\underline{a}} 1\ \} \\ & x(\underline{c}^\circ \cdot (X/Y) \cdot \underline{a})x \\ \equiv & \{\ \text{one-point (224)}\ \} \\ & x' = x \;\Rightarrow\; x'(\underline{c}^\circ \cdot (X/Y) \cdot \underline{a})x \end{array}$$

We proceed by going pointfree:

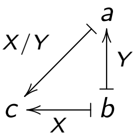
We reason

$$
\begin{array}{ll}
& id \subseteq \underline{c}^\circ \cdot (X / Y) \cdot \underline{a} \\
\equiv & \quad \{ \text{ shunting rules } \} \\
& \underline{c} \cdot \underline{a}^\circ \subseteq X / Y \\
\equiv & \quad \{ \text{ universal property (119) } \} \\
& \underline{c} \cdot \underline{a}^\circ \cdot Y \subseteq X \\
\equiv & \quad \{ \text{ now shunt } \underline{c} \text{ back to the right } \} \\
& \underline{a}^\circ \cdot Y \subseteq \underline{c}^\circ \cdot X \\
\equiv & \quad \{ \text{ go back to points via (21) } \} \\
& \langle \forall\ b\ :\ a\ Y\ b :\ c\ X\ b \rangle
\end{array}
$$

Outcome

In summary:

$$c\ (X/Y)\ a \ \equiv\ \langle \forall\ b\ :\ a\ Y\ b :\ c\ X\ b \rangle \qquad (120)$$



Example:

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

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

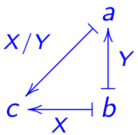
$c\ (X/Y)\ a$ = company c is the only one trusted by passenger a, that is, a **only flies** c.

Pattern X / Y

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

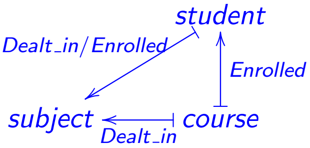
---

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



For instance,

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



Pointwise meaning in full

The full pointwise encoding of

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

is:

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

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

$$\langle \forall\ b\ :\ \langle \exists\ a\ :\ \phi :\ \psi \rangle :\ \gamma \rangle \ \equiv \ \langle \forall\ a\ :\ \phi :\ \langle \forall\ b\ :\ \psi :\ \gamma \rangle \rangle$$

recognizable as the **splitting** rule (232) of the Eindhoven calculus.

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

Exercises

Exercise 55: Prove the equalities

$$X \cdot f = X / f^\circ \tag{121}$$

$$X / \bot = \top \tag{122}$$

$$X / id = X \tag{123}$$

and check their pointwise meaning. □

Exercise 56: Define

$$X \setminus Y = (Y^\circ / X^\circ)^\circ \tag{124}$$

and infer:

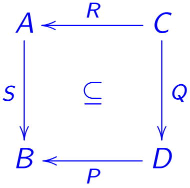
$$a(R \setminus S)c \equiv \langle \forall\ b\ :\ b\ R\ a :\ b\ S\ c \rangle \tag{125}$$

$$R \cdot X \subseteq Y \equiv X \subseteq R \setminus Y \tag{126}$$

□

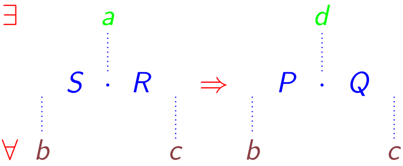
Patterns in diagrams (again!)

Back to our good old "squares":



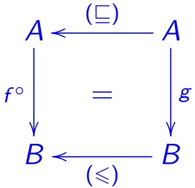
$$S \cdot R \subseteq P \cdot Q$$

... i.e. the pointwise:



Patterns in diagrams - very special case

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



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

$$f\ b \sqsubseteq a \ \equiv \ b \leqslant g\ a \qquad (127)$$

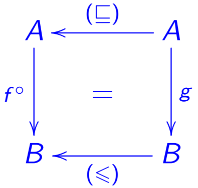
In this very special situation, f and g in

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

$$f \vdash g \qquad (128)$$

Patterns in diagrams - very special case

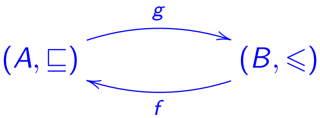
Again assuming two preorders $(\sqsubseteq)$ and $(\leqslant)$:



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

$$f\ b \sqsubseteq a \ \equiv\ b \leqslant g\ a \qquad (127)$$

In this very special situation, f and g in

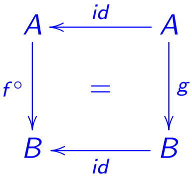


are said to be **Galois connected** (GC) and we write

$$f \vdash g \qquad (128)$$

Patterns in diagrams - even more special case

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



$$f^\circ = g$$

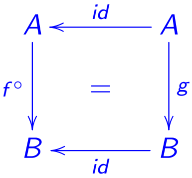
$$f\ b = a \equiv b = g\ a \qquad (129)$$

That is to say,

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

Patterns in diagrams - even more special case

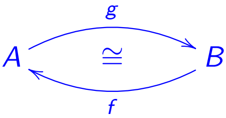
Preorders $(\sqsubseteq)$ and $(\leqslant)$ are the **identity**:



$$f^\circ = g$$

$$f\ b = a \ \equiv \ b = g\ a \qquad (129)$$

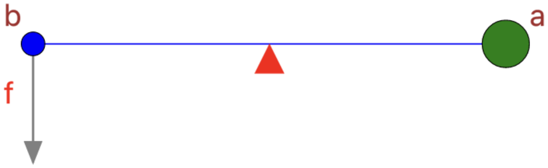
That is to say,



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

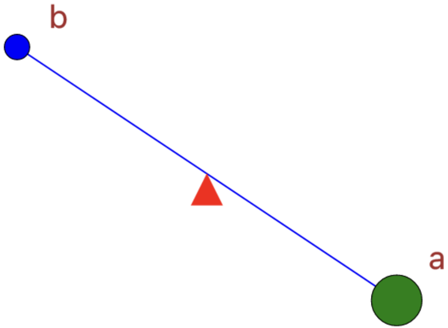
GC — mechanics analogy

Stability:



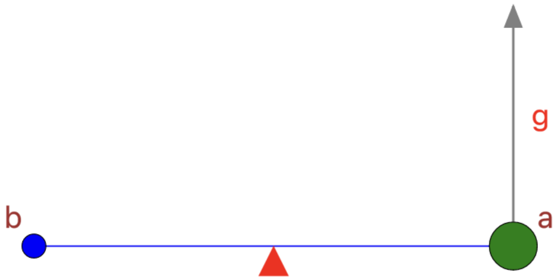
GC — mechanics analogy

Instability:



GC — mechanics analogy

Stability restored:



"*Restauratio*" rule (Middle Ages).

Example of GC

Integer division GC:

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

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* $z \times y \leqslant x$.

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 is 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:

$$\mathsf{length}\ (\mathsf{take}\ n\ xs) \leqslant n \ \wedge\ \mathsf{take}\ n\ xs \ \preceq\ xs \tag{130}$$

But this is not **enough** — (silly) implementation $\mathsf{take}\ n\ xs = [\,]$ meets (130)!

Superlatives...

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

...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 (130)

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

- Then (from above) ys should be a prefix of take n xs:

$$\text{length } ys \leqslant n \wedge ys \ \preceq \ xs \ \Rightarrow \ ys \ \preceq \ \text{take } n \ xs \qquad (131)$$

Final touch

So we have two clauses,

a **easy** *one (130)*

and

a **hard** *one (131).*

Interestingly, (130) can be derived from (131) itself,

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

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

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

$$\mathit{length}\ ys \leqslant n \ \wedge\ ys \preceq xs \quad \equiv \quad ys \preceq \text{take}\ n\ xs \qquad (132)$$

— 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 57: Solely relying on specification (132) use indirect equality to prove that

$$take\ (length\ xs)\ xs = xs \tag{133}$$

$$take\ 0\ xs = [\,] \tag{134}$$

$$take\ n\ [\,] = [\,] \tag{135}$$

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 58: Derive from (127) 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

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

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

Example of calculation from the GC

Converse involution (cancellation):

$$(R^\circ)^\circ = R \tag{136}$$

Proof of (136):

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

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,

$$X \cdot R \subseteq Y \equiv X \subseteq Y \mathbin{/} R \qquad (137)$$

$$R \cdot X \subseteq Y \equiv X \subseteq R \setminus Y \qquad (138)$$

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

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

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

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

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	$(\_ - R)$	$(R \cup)$	$b\ (X - R)\ a \equiv \begin{cases} b\ X\ a \\ \neg\ (b\ R\ a) \end{cases}$

Thus the universal properties of implication and difference,

$$R \cap X \subseteq Y \quad \equiv \quad X \subseteq R \Rightarrow Y \tag{139}$$

$$X - R \subseteq Y \quad \equiv \quad X \subseteq R \cup Y \tag{140}$$

are GCs — etc, etc

Exercise 60: Show that $R \cap (R \Rightarrow Y) \subseteq Y$ ("modus ponens") holds. □

Class 8 — How predicates become relations

How predicates become relations

Recall from (38) 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.

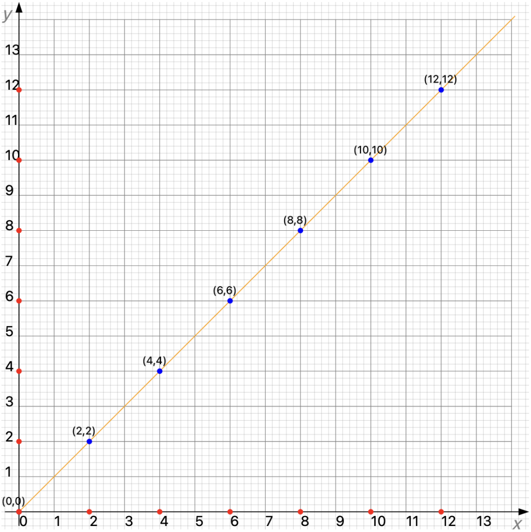
Now define:

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

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

Exercise 61: 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 (65) and

$$\underline{k} \cdot R \ \subseteq \ \underline{k} \tag{143}$$

Then:

$$R \cdot \Phi_p \quad = \quad R \cap \top \cdot \Phi_p \tag{144}$$

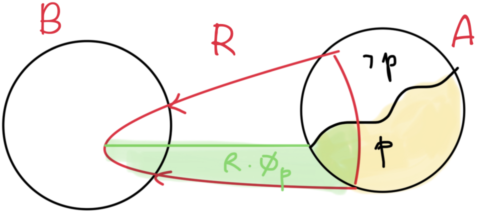
$$\Phi_q \cdot R \quad = \quad R \cap \Phi_q \cdot \top \tag{145}$$

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

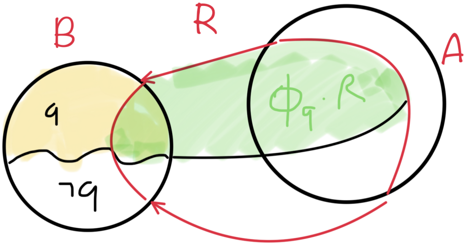
Exercise 62: Why does (143) hold? □

Relational restrictions

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



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



Exercises

Exercise 63: Show that $R - S \subseteq R$, $R - \bot = R$ and $R - R = \bot$ hold. □

Exercise 64: Prove the distributive property:

$$g^{\circ} \cdot (R \cap S) \cdot f \quad = \quad g^{\circ} \cdot R \cdot f \cap g^{\circ} \cdot S \cdot f \tag{146}$$

Then show that

$$g^{\circ} \cdot \Phi_p \cdot f \quad = \quad \frac{f}{g} \cap \frac{true}{p \cdot g} \tag{147}$$

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

Exercise 65: Infer

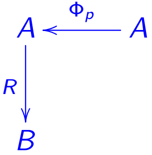
$$\Phi_q \cdot \Phi_p = \Phi_q \cap \Phi_p \tag{148}$$

from properties (144) and (145). □

Contracts

More on pre/post relational restrictions

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

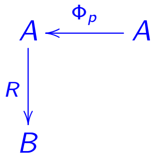


... and those in a **post** restriction

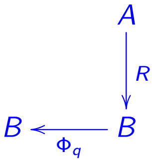
we immediately realize they fit together into a "magic" square...

More on pre/post relational restrictions

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



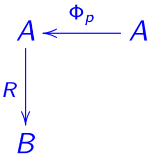
... and those in a **post** restriction



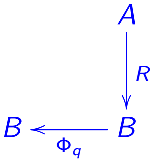
we immediately realize they fit together into a "magic" square...

More on pre/post relational restrictions

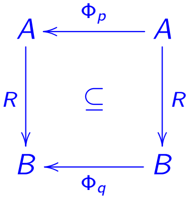
Looking at the types in a **pre** restriction



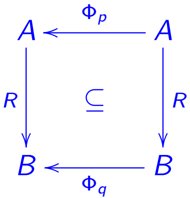
... and those in a **post** restriction



we immediately realize they fit together into a "magic" square...



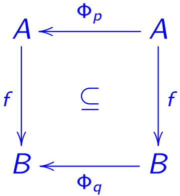
"Magic squares" — again!



$$R \cdot \Phi_p \subseteq \Phi_q \cdot R \tag{149}$$

What does this mean?

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



$$f \cdot \Phi_p \subseteq \Phi_q \cdot f \tag{150}$$

Contracts

By shunting, (150) is the same as $\Phi_p \subseteq f^\circ \cdot \Phi_q \cdot f$, therefore meaning:

$$\langle \forall\ a\ :\ p\ a :\ q\ (f\ a) \rangle \qquad (151)$$

by exercise 61.

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

$$p \xrightarrow{f} q \qquad (152)$$

— a notation that generalizes the type of f. **Important**: thanks to (145), (150) 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.

$$
\begin{array}{ll}
 & f \cdot \Phi_p \subseteq \Phi_q \cdot f \\
\equiv & \quad \{ \text{ see above (145) } \} \\
 & f \cdot \Phi_p \subseteq \Phi_q \cdot \top \\
\equiv & \quad \{ \text{ shunting (35); (142) } \} \\
 & \Phi_p \subseteq f^\circ \cdot \frac{true}{q} \\
\equiv & \quad \{ \text{ (40) } \} \\
 & \Phi_p \subseteq \frac{true}{q \cdot f} \\
\equiv & \quad \{ \Phi_p \subseteq id \text{ ; (57) } \} \\
 & \Phi_p \subseteq id \cap \frac{true}{q \cdot f} \\
\equiv & \quad \{ \text{ (141) } \} \\
 & \Phi_p \subseteq \Phi_{q \cdot f}
\end{array}
$$

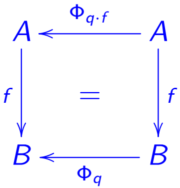
We conclude that $q \cdot f$ is such a **weakest** pre-condition.

Perfect magic square

The special situation of a weakest precondition is nicely captured by the universal property:

$$f \cdot \Phi_p = \Phi_q \cdot f \quad \equiv \quad p = q \cdot f \tag{153}$$

that is, the "perfect square":



$$f \cdot \Phi_{q \cdot f} = \Phi_q \cdot f \tag{154}$$

Weakest pre-conditions

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

Exercise 66: Calculate the weakest pre-condition $\text{WP}(f, q)$ 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 67: 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 (152), we say that q is an **invariant** of f — meaning that the "truth value" of q remains unchanged by execution of f.

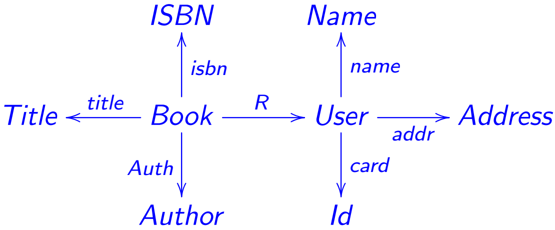
More generally, invariant q is **preserved** by function f provided contract $p \xrightarrow{f} q$ holds and $p \Rightarrow q$, that is, $\Phi_p \subseteq \Phi_q$.

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* $\Phi_{(p \wedge q)} = \Phi_p \cdot \Phi_q$.

Class 9 - Weakest precondition calculation

Library loan example



$u\ R\ b$ 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:

$$\mathrm{img}\ R \subseteq id \tag{155}$$

- no book without an author:

$$id \subseteq \ker Auth \tag{156}$$

- no two users with the same card Id:

$$\ker card \subseteq id \tag{157}$$

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:

$$return\ S\ R = R - S \tag{158}$$

$$borrow\ S\ R = S \cup R \tag{159}$$

Clearly, these operations only change the *books-on-loan* relation R, which is conditioned by **invariant**:

$$inv\ R = \mathrm{img}\ R \ \subseteq\ id$$

Library loan example

The question is, then: given

$$inv\ R = \mathrm{img}\ R \ \subseteq\ id \tag{160}$$

are the following "types"

$$inv \xleftarrow{return\ S} inv \tag{161}$$

$$inv \xleftarrow{borrow\ S} inv \tag{162}$$

ok?

We check (161,162) below.

Library loan example

Checking (161):

$$
\begin{array}{ll}
& inv\ (return\ S\ R) \\
\equiv & \quad \{\ \text{inline definitions}\ \} \\
& \mathrm{img}\ (R - S)\ \subseteq\ id \\
\Leftarrow & \quad \{\ \text{since } \mathrm{img} \text{ is monotonic}\ \} \\
& \mathrm{img}\ R\ \subseteq\ id \\
\equiv & \quad \{\ \text{definition}\ \} \\
& inv\ R \\
\Box &
\end{array}
$$

So, for all R, $inv\ R \Rightarrow inv\ (return\ S\ R)$ holds — invariant inv is preserved.

Library loan example

Checking (161):

$$
\begin{array}{cl}
 & inv\ (return\ S\ R) \\
\equiv & \quad \{ \text{ inline definitions } \} \\
 & \mathrm{img}\ (R - S)\ \subseteq\ id \\
\Leftarrow & \quad \{ \text{ since } \mathrm{img} \text{ is monotonic } \} \\
 & \mathrm{img}\ R\ \subseteq\ id \\
\equiv & \quad \{ \text{ definition } \} \\
 & inv\ R \\
\Box &
\end{array}
$$

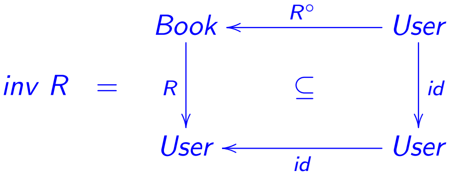
So, for all R, $inv\ R \Rightarrow inv\ (return\ S\ R)$ holds — invariant inv is preserved.

Library loan example

At this point note that (161) 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 63) and "smaller than injective is injective" (exercise 24), it is immediate that inv (160) is preserved.

To see this better, unfold and draw definition (160):



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:

$$
\begin{array}{ll}
& inv\ (borrow\ S\ R) \\
\equiv & \{\ \text{inline definitions}\ \} \\
& \mathrm{img}\ (S \cup R) \subseteq id \\
\equiv & \{\ \text{exercise 36}\ \} \\
& \mathrm{img}\ R \subseteq id \wedge \mathrm{img}\ S \subseteq id \wedge S \cdot R^{\circ} \subseteq id \\
\equiv & \{\ \text{definition of } inv\ \} \\
& inv\ R \wedge \underbrace{\mathrm{img}\ S \subseteq id \wedge S \cdot R^{\circ} \subseteq id}_{\textsc{wp}(borrow\ S, inv)}
\end{array}
$$

Library loan example (Alloy)

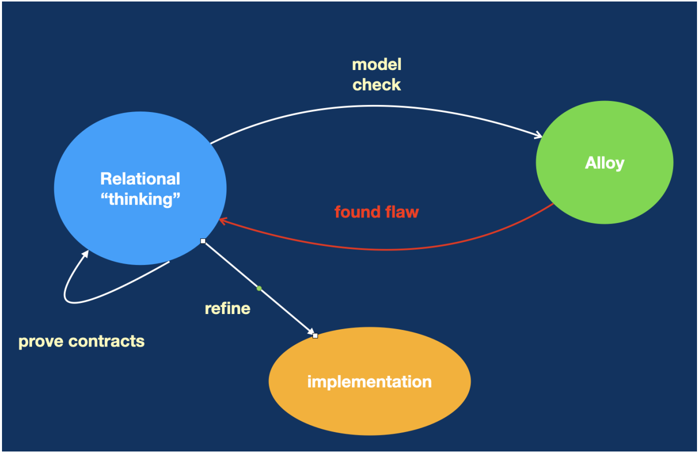
In practice, our proposed **workflow** does not rush to the **calculation** of the **weakest precondition** of a **contract**.

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 play any role in *borrow*, we comment them all, obtaining a much smaller model:

sig $Book$ $\{ R : \text{lone } User \}$
sig $User$ $\{ \}$

fun $borrow$
  $[S, R : Book \rightarrow \text{lone } User] :$
    $Book \rightarrow \text{lone } User$ $\{$
  $R + S$
$\}$

Library loan example (Alloy)

Next, we single out the **invariant**, making it explicit as a **predicate**:

sig *Book* { *R* : *User* }

sig *User* { }

pred *inv* {
  R **in** *Book* $\rightarrow$ lone *User*
}

fun *borrow*
  [*S*, *R* : *Book* $\rightarrow$ *User*] :
    Book $\rightarrow$ *User* {
  $R + S$
}

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 $Book$

$$\text{sig } Time\ \{ r : Book \rightarrow User \}$$
$$\text{sig } Book\ \{ \}$$
$$\text{sig } User\ \{ \}$$

and re-writing inv accordingly (aside).

```
pred inv [t : Time] {
    t · r in Book → lone User
}
```

Note how $r : Time \rightarrow (Book \rightarrow User)$ is a **function** — it yields, for each $t \in Time$, the relation $Book \xrightarrow{r\ t} User$.

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:

$$
\begin{array}{l}
assert\ contract\ \{ \\
\quad \mathrm{all}\ t, t' : Time, S : Book \rightarrow User\ | \\
\quad\quad inv\ [t]\ and\ t' \cdot r = borrow\ [t \cdot r, S] \Rightarrow inv\ [t'] \\
\}
\end{array}
$$

Once we check this, for instance running

$$\mathrm{check}\ contract\ for\ 3\ but\ exactly\ 2\ Time$$

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:

$$\begin{array}{l} \text{pred } \mathit{pre}\ [t : \mathit{Time}, S : \mathit{Book} \rightarrow \mathit{User}]\ \{ \\ \quad S\ \textbf{in}\ \mathit{Book} \rightarrow \text{lone}\ \mathit{User} \\ \quad {\sim}S \cdot (t \cdot r)\ \textbf{in}\ \text{iden} \\ \} \end{array}$$

The fact that this yields no more counter-examples 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.

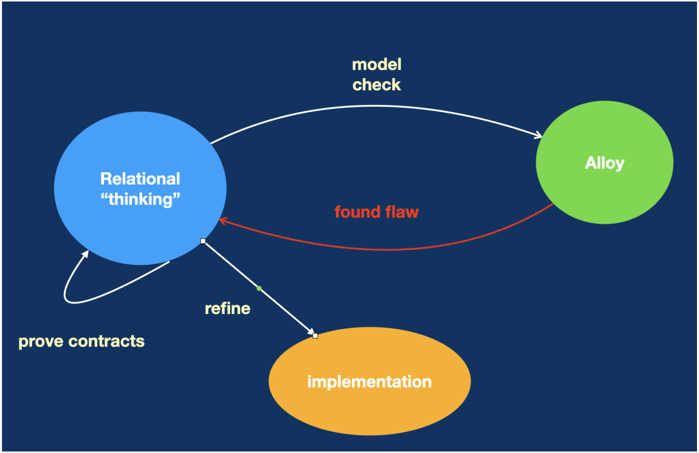
Library loan example (Alloy)

Note that pre-conditioned $borrow\ S \cdot \Phi_{pre}$ is no 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:

$$
\begin{array}{l}
\text{pred } borrow\ [t, t' : Time, S : Book \rightarrow User]\ \{ \\
\quad \text{-- pre-condition} \\
\quad S\ \textbf{in}\ Book \rightarrow \text{lone } User \\
\quad \sim S \cdot (t \cdot r)\ \textbf{in}\ \text{iden} \\
\quad \text{-- post-condition} \\
\quad t' \cdot r = t \cdot r + S \\
\}
\end{array}
$$

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].

Relational contracts

Relation *borrow* $S \cdot \Phi_{pre}$ above invites us to go back to (149) and define

$$p \xrightarrow{R} q \quad \equiv \quad R \cdot \Phi_p \subseteq \Phi_q \cdot R \tag{163}$$

thus generalizing functional contracts (150) to arbitrary relations, meaning:

$$\langle \forall\ a\ :\ p\ a :\ \langle \forall\ b\ :\ b\ R\ a :\ q\ b \rangle \rangle \tag{164}$$

— see the exercise below.

Exercise 68: Sow that an alternative way of stating (163) is

$$p \xrightarrow{R} q \quad \equiv \quad R \cdot \Phi_p \subseteq \Phi_q \cdot \top \tag{165}$$

Then derive (164) from (165). □

Exercises

Exercise 69: In Alloy, wherever on writes

$$\text{sig } A \, \{ \text{disj } R, S : \text{set } B \}$$

the keyword disj enforces the following invariant:

No $b \in B$ is related by both R and S to the same $a \in A$:

$$\langle \forall\ a, b \ :\ b\ R\ a : \ \langle \forall\ b' \ :\ b'\ S\ a : \ b' \neq b \rangle \rangle \tag{166}$$

Show that (166) is nothing but (167) below:

$$S \cdot R^{\circ} \cap id \ \subseteq\ \bot \tag{167}$$

□

Two distinguished coreflexives: domain and range

Remember...

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

How about intersecting both with id?

$$\delta R = \ker R \cap id \tag{168}$$

$$\rho R = \mathrm{img}\ R \cap id \tag{169}$$

Two 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

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

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

Dually,

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

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

Distinguished coreflexives: domain and range

Universal properties:

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

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:

$$\delta X \ \subseteq\ Y \equiv X \subseteq \top \cdot Y \tag{170}$$

$$\rho R \ \subseteq\ Y \equiv R \subseteq Y \cdot \top \tag{171}$$

Distinguished coreflexives: domain and range

Some facts about domain and range:

$$R = R \cdot \delta R \tag{172}$$

$$R = \rho R \cdot R \tag{173}$$

$$\delta(R \cdot S) = \delta(\delta R \cdot S) \tag{174}$$

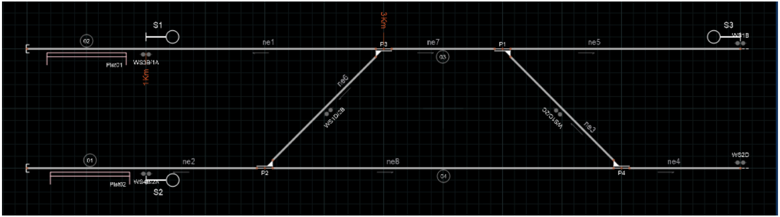
$$\rho(R \cdot S) = \rho(R \cdot \rho S) \tag{175}$$

$$\top \cdot \delta R = \top \cdot R \tag{176}$$

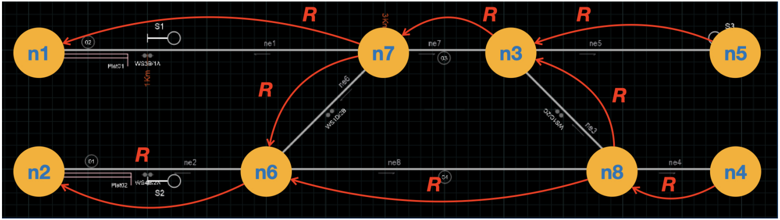
$$\rho R \cdot \top = R \cdot \top \tag{177}$$

$$\delta R \subseteq \delta S \equiv R \subseteq \top \cdot S \tag{178}$$

Case study: railway topology



Case study: railway topology



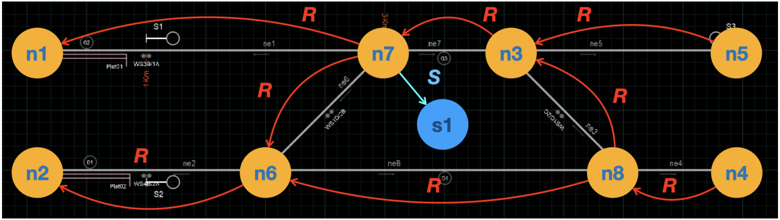
$$Sw \xleftarrow{S} N \xleftarrow{R} N \xrightarrow{P} Sl$$

where

Sw – *switches* ('agulhas')

Sl – *signals* ('sinais')

Case study: railway topology

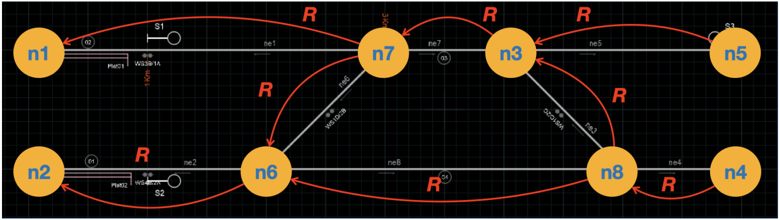


$$Sw \xleftarrow{S} N \xleftarrow{R} N \xrightarrow{P} Sl$$

Switches:

$$switchOk(S, R, P) \;=\; \delta S \subseteq R^{\circ} \cdot (\neq) \cdot R$$

Case study: railway topology



$$Sw \xleftarrow{S} N \xleftarrow{R} N \xrightarrow{P} Sl$$

Add a switch:

$$addSwitch\ (s, n)\ (S, R, P) = (S \cup \underline{s} \cdot \underline{n}^{\circ}, R, P)$$

Case study: railway topology

$$switchOk\ (addSwitch\ (s,n)\ (S,R,P))$$

$$\equiv \quad \{ \ \dots\dots\dots \ \}$$

$$\delta(S \cup \underline{s} \cdot \underline{n}^\circ) \ \subseteq \ R^\circ \cdot (\neq) \cdot R$$

$$\equiv \quad \{ \ \dots\dots\dots \ \}$$

$$switchOk\ (S,R,P) \wedge \underline{n} \cdot \top \cdot \underline{n}^\circ \ \subseteq \ R^\circ \cdot (\neq) \cdot R$$

$$\equiv \quad \{ \ \dots\dots\dots \ \}$$

$$switchOk\ (S,R,P) \wedge \top \ \subseteq \ \underline{n}^\circ \cdot R^\circ \cdot (\neq) \cdot R \cdot \underline{n}$$

$$\equiv \quad \{ \ \dots\dots\dots \ \}$$

$$switchOk\ (S,R,P) \wedge \langle \exists\ n_1, n_2\ :\ n_1 \neq n_2 :\ n\ R^\circ\ n_1 \wedge n_2\ R\ n \rangle$$

$$\equiv \quad \{ \ \dots\dots\dots \ \}$$

$$switchOk\ (S,R,P) \wedge \underbrace{\langle \exists\ n_1, n_2\ :\ n_1 \neq n_2 :\ n_1\ R\ n \wedge n_2\ R\ n \rangle}_{WP}$$

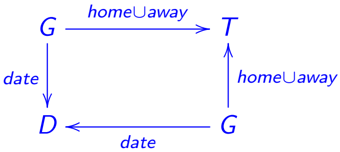
Exercise 21 (continued)

Exercise 70: Recalling exercise 21, 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:

$$\ker\ (home \cup away) - id \xrightarrow{date} Ok$$



□

Class 10 - 10-4-2025

Difunctions

Difunctional relations

A relation R is said to be **difunctional** or *regular* wherever $R \cdot R^\circ \cdot R = R$ holds, which amounts to $R \cdot R^\circ \cdot R \subseteq R$ since the converse inclusion always holds.

Exercise 71: Is the following relation **difunctional** ?

R	a_1	a_2	a_3	a_4	a_5
b_1	0	0	1	0	1
b_2	0	0	0	0	0
b_3	0	1	0	0	0
b_4	0	1	0	1	0
b_5	0	0	0	1	0

(179)

Justify your answer. □

Exercise 72: Use (172) — or (173) — to show that $R \subseteq R \cdot R^\circ \cdot R$ always holds. □

Difunctional relations

Some intuition about R being **difunctional**:

R is such that, wherever two inputs have a common image, then they have exactly the same set of images.

That is: **columns** in the matrix representation of R are either the same or disjoint.

Useful:

A difunctional relation that is *reflexive* and *symmetric* necessarily is an *equivalence* relation.

Exercises

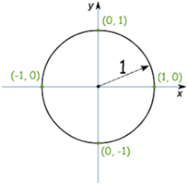
Exercise 73: Find the smallest difunctional relation that contains R (179) of exercise 71. $\square$

Exercise 74: Show that $\frac{f}{g}$ is difunctional. $\square$

Exercise 75: Use the difunctionality of $\ker f = \frac{f}{f}$ to show that $\ker f$ is an **equivalence** relation. $\square$

Exercise 76: Show that the unit circunference

$y\ R\ x \ \equiv\ y^2 + x^2 = 1$ is difunctional. **Hint**: recall exercise 18.



$\square$

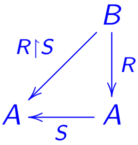
Shrinking and overriding

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

$$X \subseteq R \upharpoonright S \quad \equiv \quad X \subseteq R \ \wedge \ X \cdot R^\circ \subseteq S \tag{180}$$

cf. diagram:



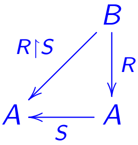
Property (180) states that $R \upharpoonright S$ is the largest part of R such that, if it yields an output for an input x, this must be a 'maximum, with respect to S, among all possible outputs of x by R.

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

$$X \subseteq R \upharpoonright S \quad \equiv \quad X \subseteq R \ \wedge \ X \cdot R^\circ \subseteq S \tag{180}$$

cf. diagram:



Property (180) 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.

Exercises

Exercise 77: Show, by indirect equality, that (180) is equivalent to:

$$R \upharpoonright S \;=\; R \cap S / R^{\circ} \tag{181}$$

$\square$

Exercise 78: Show that not only the injectivity but also the simplicity of a given relation R is preserved by its shrinking $R \upharpoonright S$ by any other relation S on its output type. $\square$

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 $S = \pi_2 \cdot R$ is the relation

$$Mark \xleftarrow{\pi_2 \cdot R} Student \;=\; \left(\begin{array}{c|c} Mark & Student \\ \hline 10 & John \\ 11 & Mary \\ 12 & John \\ 15 & Arthur \end{array}\right)$$

and

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

Properties of shrinking

Two *fusion* rules:

$$(S \cdot f) \upharpoonright R \quad = \quad (S \upharpoonright R) \cdot f \qquad (182)$$

$$(f \cdot S) \upharpoonright R \quad = \quad f \cdot (S \upharpoonright (f^\circ \cdot R \cdot f)) \qquad (183)$$

"Chaotic optimization":

$$R \upharpoonright \top \quad = \quad R \qquad (184)$$

"Impossible optimization":

$$R \upharpoonright \bot \quad = \quad \bot \qquad (185)$$

"Brute force" determinization:

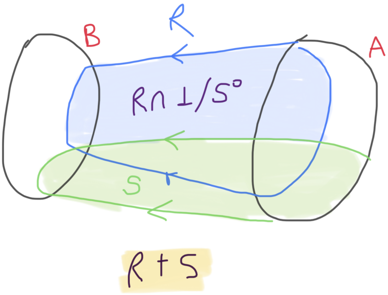
$$R \upharpoonright id \quad = \quad \text{largest simple fragment of } R \qquad (186)$$

Relation overriding

The relational **overriding** combinator

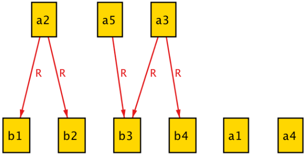
$$R \dagger S \;=\; S \cup R \cap \bot / S^{\circ} \qquad (187)$$

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:

$P = \top \dagger R =$

	a_1	a_2	a_3	a_4	a_5
b_1					
b_2					
b_3					
b_4					

Exercise on relation overriding

And now this other one:

$Q = R \dagger (\underline{b_4} \cdot \underline{a_2}^\circ) =$

	a_1	a_2	a_3	a_4	a_5
b_1					
b_2					
b_3					
b_4					

□

Exercise 79: (a) Show that $\bot \dagger S = S$, $R \dagger \bot = R$ and $R \dagger R = R$ hold. (b) Infer the universal property:

$$X \subseteq R \dagger S \quad \equiv \quad X - S \ \subseteq \ R \wedge (X - S) \cdot S^\circ = \bot \tag{188}$$

□

Propositio de homine et capra et lvpo

Recalling the Alcuin puzzle (4)

$$\begin{array}{ccccc} Being & \xrightarrow{Eats} & Being & & \\ & & \downarrow{\scriptstyle where} & & \\ & & Bank & \xrightarrow{cross} & Bank \end{array}$$

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

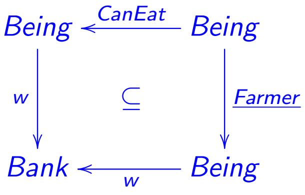
$$carry(where, who) = where \dagger (cross \cdot where \cdot \Phi_{who}) \qquad (189)$$

In **Alloy** syntax:

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

Invariants versus contracts

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



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

Recalling (189),

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

we also define:

$$trip\ b\ w = carry\ (w, b) \qquad (190)$$

Invariants versus contracts

Then the **contract**

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

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

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*.

Exercises

Exercise 80: Show that

$$R \dagger f = f$$

holds, arising from (188,140) — where f is a function, of course. □

Exercise 81: Function *move* (189) could have been defined by

$$move = where_{who}^{cross}$$

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

$$R_p^f \quad = \quad R \dagger (f \cdot R \cdot \Phi_p) \tag{191}$$

Prove the equalities: $R_p^{id} = R$, $R_{false}^f = R$ and $R_{true}^f = f \cdot R$.

□

Class 11 — 8-5-2025

Relators

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

$$\begin{array}{ccc} A & \cdots\cdots & \mathcal{G}A \\ {\scriptstyle R}\downarrow & & \downarrow{\scriptstyle \mathcal{G}R} \\ B & \cdots\cdots & \mathcal{G}B \end{array} \tag{192}$$

from $\mathcal{G}A$ to $\mathcal{G}B$ which obeys the following properties:

$$\mathcal{G}id = id \tag{193}$$

$$\mathcal{G}\ (R \cdot S) = (\mathcal{G}\ R) \cdot (\mathcal{G}\ S) \tag{194}$$

$$\mathcal{G}(R^\circ) = (\mathcal{G}\ R)^\circ \tag{195}$$

and is **monotonic**:

$$R \subseteq S \quad \Rightarrow \quad \mathcal{G}R \subseteq \mathcal{G}S \tag{196}$$

“Functorial squares””

For any relator $\mathcal{G}$:

$$
\begin{array}{ccc}
A & \xleftarrow{R} & C \\
{\scriptstyle P}\downarrow & \subseteq & \downarrow{\scriptstyle Q} \\
B & \xleftarrow[S]{} & D
\end{array}
\quad\Rightarrow\quad
\begin{array}{ccc}
\mathcal{G}\ A & \xleftarrow{\mathcal{G}\ R} & \mathcal{G}\ C \\
{\scriptstyle \mathcal{G}\ P}\downarrow & \subseteq & \downarrow{\scriptstyle \mathcal{G}\ Q} \\
\mathcal{G}\ B & \xleftarrow[\mathcal{G}\ S]{} & \mathcal{G}\ D
\end{array}
\qquad (197)
$$

This follows from the monotonicity (196) and functoriality (194) of $\mathcal{G}$.

Relators: *"Maybe"* example

$$
\begin{array}{ccc}
A & \cdots\cdots & \mathcal{G}A = 1 + A \\
\downarrow R & & \downarrow \mathcal{G}R = id + R \\
B & \cdots\cdots & \mathcal{G}B = 1 + B
\end{array}
$$

(Read $1 + A$ as "maybe A")

Unfolding $\mathcal{G}R = id + R$:

$$
\begin{array}{ll}
& y(id + R)x \\
\equiv & \quad \{ \text{ unfolding the sum, cf. } id + R = [i_1 \cdot id\ , i_2 \cdot R] \ \} \\
& y(i_1 \cdot i_1^\circ \cup i_2 \cdot R \cdot i_2^\circ)x \\
\equiv & \quad \{ \text{ relational union (51); image } \} \\
& y(\mathrm{img}\ i_1)x \vee y(i_2 \cdot R \cdot i_2^\circ)x \\
\equiv & \quad \{ \text{ let } NIL \text{ be } \underline{\text{the}} \text{ inhabitant of the singleton type } \} \\
& y = x = i_1 NIL \vee \langle \exists\ b, a\ :\ y = i_2\ b \wedge x = i_2\ a :\ b\ R\ a \rangle
\end{array}
$$

Relators: R^* example

Take $\mathcal{G}\ X = X^*$.

Then, for some $B \xleftarrow{R} A$, relator $B^\star \xleftarrow{R^\star} A^\star$ is the relation

$$R^* = [\mathsf{nil}\ , \mathsf{cons} \cdot (R \times R^*)] \cdot \mathsf{out} \tag{198}$$

where

$$\begin{aligned}
&\mathsf{out} = \mathsf{in}^\circ \\
&\mathsf{in} = [\mathsf{nil}\ , \mathsf{cons}] \\
&\mathsf{nil}\ _ = [\,] \\
&\mathsf{cons}\ (h, t) = h : t.
\end{aligned}$$

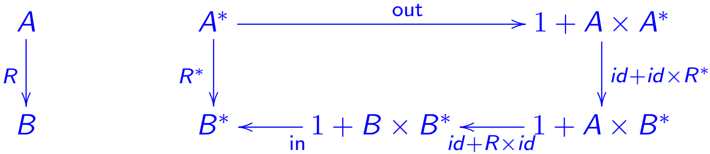
What does (198) mean?

Relators: R^* example

To unfold

$$R^* = [\mathsf{nil}\ , \mathsf{cons} \cdot (R \times R^*)] \cdot \mathsf{out}$$

look at this diagram first:



About R^*

Then:

$$R^* \cdot \mathsf{in} = [\mathsf{nil}\ , \mathsf{cons} \cdot (R \times R^*)]$$

$$\equiv \qquad \{\ \mathsf{in} = [\mathsf{nil}\ , \mathsf{cons}]\ \text{etc}\ \}$$

$$\left\{ \begin{array}{l} R^* \cdot \mathsf{nil} = \mathsf{nil} \\ R^* \cdot \mathsf{cons} = \mathsf{cons} \cdot (R \times R^*) \end{array} \right.$$

that is:

$$\left\{ \begin{array}{l} y\ R^*\ [\,] \ \equiv\ y = [\,] \\ y\ R^*\ (h : t) \ \equiv\ \langle \exists\ b, x\ :\ y = (b : x) :\ b\ R\ a \wedge x\ R^*\ t \rangle \end{array} \right.$$

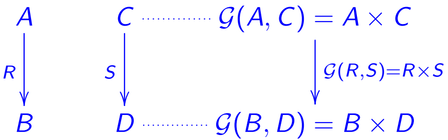
In case $R := f$, $R^* = \mathsf{map}\ f$.

Exercises

Exercise 82: Show that $\bot^* \neq \bot$ because, in fact, $\bot^* = \mathsf{nil} \cdot \mathsf{nil}^\circ$.
Hint: recall the divide & conquer rule (110), among others. □

Exercise 83: 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 84: Show that (Kronecker) product



is a (binary) relator. □

Theorems for free

Parametric polymorphism by example

Function

$$countBits : \mathbb{N}_0 \leftarrow Bool^\star$$
$$countBits\ [\,] = 0$$
$$countBits(b{:}bs) = 1 + countBits\ bs$$

and

$$countNats : \mathbb{N}_0 \leftarrow \mathbb{N}^\star$$
$$countNats\ [\,] = 0$$
$$countNats(b{:}bs) = 1 + countNats\ bs$$

are both subsumed by **generic** (parametric):

$$count : (\forall a)\ \mathbb{N}_0 \leftarrow a^\star$$
$$count\ [\,] = 0$$
$$count(a{:}as) = 1 + count\ as$$

Parametric polymorphism: why?

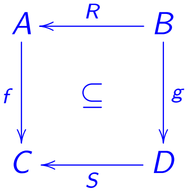
- Less code (**specific** solution = **generic** solution + **customization**)
- Intellectual reward
- Last but not least, quotation from *Theorems for free!*, by Philip Wadler [8]:

  > *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!

Back to magic squares

A very common square with 2 **functions**:



$$f \cdot R \subseteq S \cdot g \qquad (199)$$

It captures a **higher-order relation** on functions:

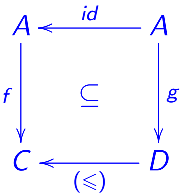
$$f\ S^R\ g \ \equiv\ f \cdot R \subseteq S \cdot g \qquad (200)$$

In words:

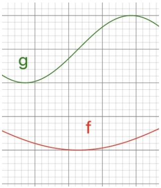
"R-related inputs are mapped to S-related outputs".

Recall (79)...

Let $R := id$, $S := (\leqslant)$:



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



This square captures the $(\leqslant)$-pointwise-ordering of functions:

$$f\ (\leqslant)^{id}\ g \quad \equiv \quad \langle \forall\ a\ ::\ f\ a \leqslant g\ a \rangle$$

In words:

"The same input is mapped to $(\leqslant)$*-related outputs".*

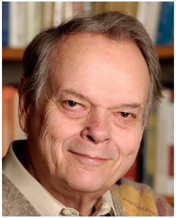
"Higher-order" squares

Because of their role in *free theorems*,
these squares will be referred to as
Reynolds squares:

$$\begin{array}{ccc} A & \xleftarrow{R} & B \\ {\scriptstyle f}\downarrow & \subseteq & \downarrow{\scriptstyle g} \\ C & \xleftarrow[S]{} & D \end{array}$$

that is to say,

$$\frac{A \xleftarrow{R} B \qquad C \xleftarrow{S} D}{C^A \xleftarrow{S^R} D^B}$$



J.C. Reynolds
(1935–2013)

Thus one is lead to **relational exponentials** S^R such that e.g.

$$(S^R)^\circ = (S^\circ)^{(R^\circ)} \tag{201}$$

$$id^{id} = id \tag{202}$$

etc. **NB**: We often write $S \leftarrow R$ or $R \rightarrow S$ instead of S^R when exponents get too nested.

"Higher-order" squares

Functions-only Reynolds squares:

$$f\ (h \rightarrow k)\ g \quad \equiv \quad f \cdot h = k \cdot g \tag{203}$$

In case of h° instead of h,

$$f\ (h^\circ \rightarrow k)\ g \ \equiv \ f \cdot h^\circ \subseteq k \cdot g \tag{204}$$

we get a **higher-order function**:

$$(h^\circ \rightarrow k)\ g = k \cdot g \cdot h \tag{205}$$

Exercise 85: Prove (203) and (205). □

"Higher-order" squares

Then:

$$(id \to k)\ g = k \cdot g \tag{206}$$
$$(h^\circ \to id)\ g = g \cdot h \tag{207}$$

cf. **covariant** and **contravariant** exponentials.

In fully pointfree notation, the exponentials (206,207) become

$$k^{id} = (k\cdot)$$
$$id^{(h^\circ)} = (\cdot h)$$

Then, by (201):

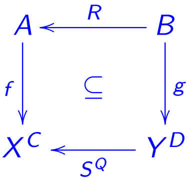
$$id^{h} = (\cdot h)^\circ \tag{208}$$

and so and so forth.

☞ Rich construction! ☺

Higher-order Reynolds squares

Exponential relations S^R can involve other exponentials, for instance $(S^Q)^R$ i.e. $R \to S^Q$:



$$f \; (R \to S^Q) \; g$$

Let us unfold this, assuming all fresh variables universally quantified:

Higher-order Reynolds squares

$$f\ (R \rightarrow S^Q)\ g \qquad (209)$$

$\equiv$ { Reynolds square (199) }

$$f \cdot R \subseteq S^Q \cdot g$$

$\equiv$ { shunting (35) followed by "nice rule" (21) }

$$a\ R\ b \Rightarrow (f\ a)\ S^Q\ (g\ b)$$

$\equiv$ { (199) again }

$$a\ R\ b \Rightarrow ((f\ a) \cdot Q \subseteq S \cdot (g\ b))$$

$\equiv$ { (35) followed by (21) again }

$$a\ R\ b \Rightarrow c\ Q\ d \Rightarrow (f\ a\ c)\ S\ (g\ b\ d) \qquad (210)$$

Currying / uncurrying (relationally)

$$a\ R\ b \Rightarrow c\ Q\ d \Rightarrow (f\ a\ c)\ S\ (g\ b\ d)$$

$\equiv$ { uncurrying f and g }

$$a\ R\ b \wedge c\ Q\ d \Rightarrow \widehat{f}\ (a, c)\ S\ \widehat{g}\ (b, d)$$

$\equiv$ { relational product (90) }

$$(a, c)\ (R \times Q)\ (c, d) \Rightarrow \widehat{f}\ (a, c)\ S\ \widehat{g}\ (b, d)$$

$\equiv$ { go pointfree by shunting (35), cf "nice rule" (21) }

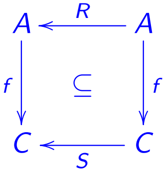
$$(R \times Q) \subseteq \widehat{f}^{\circ} \cdot S \cdot \widehat{g}$$

$\equiv$ { Reynolds square (199) }

$$\widehat{f}\ (R \times Q \rightarrow S)\ \widehat{g}$$

Relational types

$S^R \cap id$ captures all Reynolds squares (199) in which $f = g$:



$$f \cdot R \subseteq S \cdot f \qquad (211)$$

In this case we often abbreviate $f\ (R \rightarrow S)\ f$ to $f : R \rightarrow S$, meaning that f has **relational type** $R \rightarrow S$.

Note how type variables A and C in $f : A \rightarrow C$ are straightforwardly replaced by relations R and S in $f : R \rightarrow S$.

☞ **Types** *"are"* **relations** *[7]*.

Free theorems

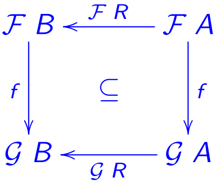
Let a **parametric** function $f : \mathcal{F}\ X \rightarrow \mathcal{G}\ X$ be given.

Its **free theorem** *states that* f *has* **relational type**

$$f : \mathcal{F}\ R \rightarrow \mathcal{G}\ R \qquad (212)$$

for any R *relating its parameters.*

The square captured by (212) is:



This extends to multi-parametric f, as will be shown briefly.

First example (id)

The simplest of all polymorphic functions is:

$$id : A \rightarrow A$$

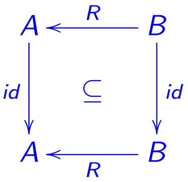
So $\mathcal{F}\ A = \mathcal{G}\ A = A$ in (212), thus we have relational type

$$id : R \rightarrow R$$

which means the *free theorem* (FT)

$$id \cdot R \ \subseteq \ R \cdot id$$

cf:

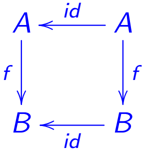


First example (id)

In case R is a function f, the FT theorem boils down to id's **natural** property:

$$id \cdot f \;=\; f \cdot id$$

cf.



which can be read alternatively as stating that id is the **unit** of composition.

Second example (*reverse*)

In this example the target function is:

$$reverse : A^* \rightarrow A^*$$

So $\mathcal{F}\ A = \mathcal{G}\ A = A^*$ in (212), and thus we have relational type

$$reverse : R^* \rightarrow R^*$$

where $R^\star$ is given by (198), leading to the FT:

$$reverse \cdot R^* \ \subseteq\ R^* \cdot reverse \qquad (213)$$

In case R is a function r, the FT theorem boils down to *reverse*'s **natural** property:

$$reverse \cdot r^\star \ =\ r^\star \cdot reverse$$

Second example (*reverse*)

However, the interest of (213) goes far beyond the case of functions, for instance:

$$
\begin{array}{ll}
 & \mathit{reverse} \cdot \bot^* \subseteq \bot^* \cdot \mathit{reverse} \\
\equiv & \qquad \{ \text{ exercise 82 ; nil} = \underline{[\,]} \ \} \\
 & \mathit{reverse} \cdot \underline{[\,]} \cdot \underline{[\,]}^\circ \subseteq \underline{[\,]} \cdot \underline{[\,]}^\circ \cdot \mathit{reverse} \\
\equiv & \qquad \{ \text{ shunting + converses + constant functions } \} \\
 & \underline{\mathit{reverse}\ [\,]} \cdot \underline{\mathit{reverse}\ [\,]}^\circ \subseteq \underline{[\,]} \cdot \underline{[\,]}^\circ \\
\equiv & \qquad \{ \ \underline{a} \cdot \underline{b}^\circ \subseteq \underline{c} \cdot \underline{d}^\circ \equiv \left\{ \begin{array}{l} a = c \\ b = d \end{array} \right. \ \} \\
 & \mathit{reverse}\ [\,] = [\,]
\end{array}
$$

Second example (*reverse*)

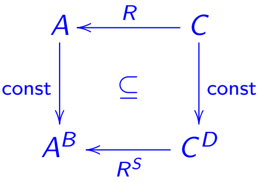
In the opposite direction, let $R := \top$ in (213):

$$
\begin{array}{cl}
& reverse \cdot \top^* \ \subseteq\ \top^* \cdot reverse \\
\equiv & \qquad \{ \text{ it can be proved that } \top^* = \frac{\mathsf{length}}{\mathsf{length}} \ \} \\
& reverse \cdot \dfrac{\mathsf{length}}{\mathsf{length}} \ \subseteq\ \dfrac{\mathsf{length}}{\mathsf{length}} \cdot reverse \\
\equiv & \qquad \{ \text{ shunting} + (40) \ \} \\
& \dfrac{\mathsf{length}}{\mathsf{length}} \ \subseteq\ \dfrac{\mathsf{length} \cdot reverse}{\mathsf{length} \cdot reverse} \\
\equiv & \qquad \{ \text{ pointwise } \} \\
& \mathsf{length}\ a = \mathsf{length}\ b \Rightarrow \mathsf{length}\ (reverse\ a) = \mathsf{length}\ (reverse\ b)
\end{array}
$$

Third example (constant functions)

Example: Haskell constant function $\mathsf{const} : a \rightarrow b \rightarrow a$, that is $\mathsf{const} : A \rightarrow A^B$.

By (212), const has **relational type** $R \rightarrow R^S$, that is:



$$\mathsf{const} \cdot R \;\subseteq\; R^S \cdot \mathsf{const} \qquad (214)$$

Pointwise equivalent, recall (209,210):

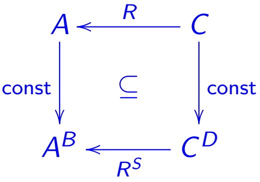
$$a\ R\ c \Rightarrow b\ S\ d \Rightarrow (\mathsf{const}\ a\ b)\ R\ (\mathsf{const}\ c\ d)$$

for all a, b, c, d.

Third example (constant functions)

Example: Haskell constant function $\mathsf{const} : a \to b \to a$, that is $\mathsf{const} : A \to A^B$.

By (212), const has **relational type** $R \to R^S$, that is:



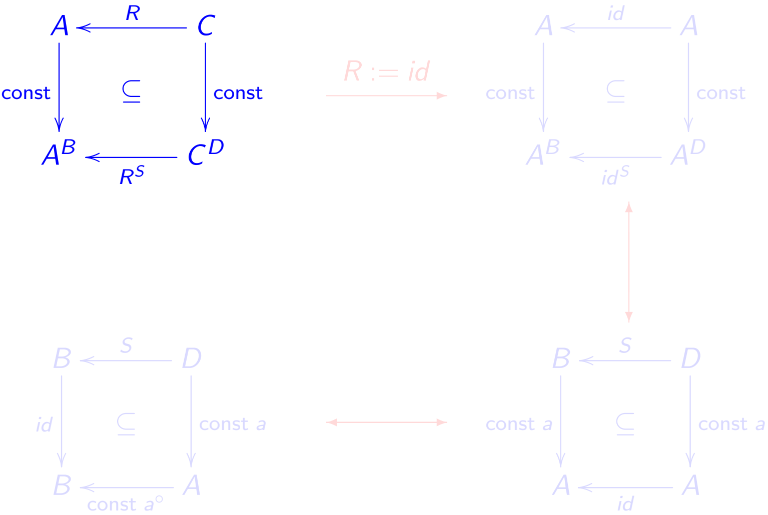
$$\mathsf{const} \cdot R \ \subseteq \ R^S \cdot \mathsf{const} \qquad (214)$$

Pointwise equivalent, recall (209,210):

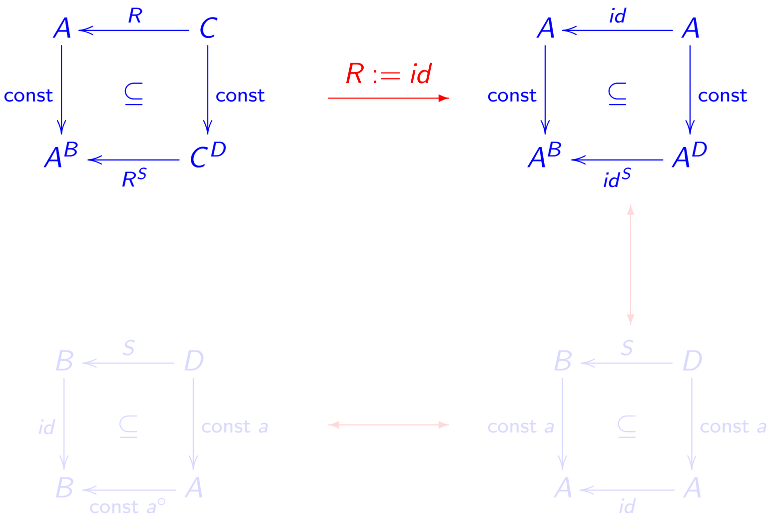
$$a\ R\ c \Rightarrow b\ S\ d \Rightarrow (\mathsf{const}\ a\ b)\ R\ (\mathsf{const}\ c\ d)$$

for all a, b, c, d.

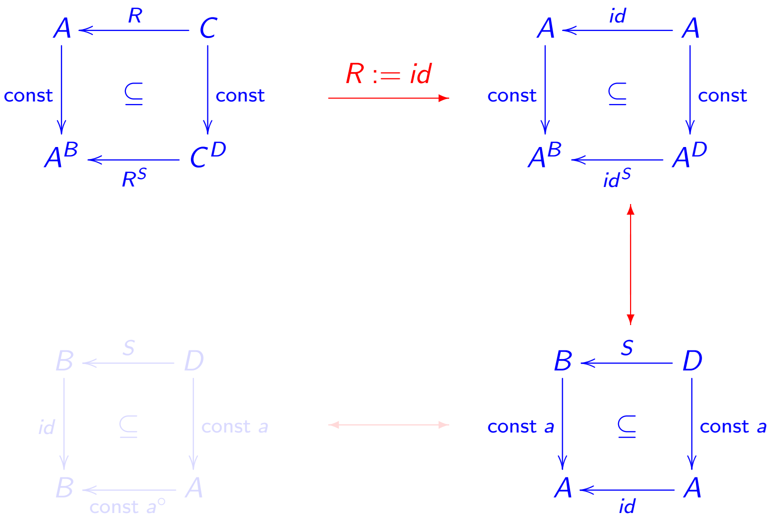
Third example (constant functions)



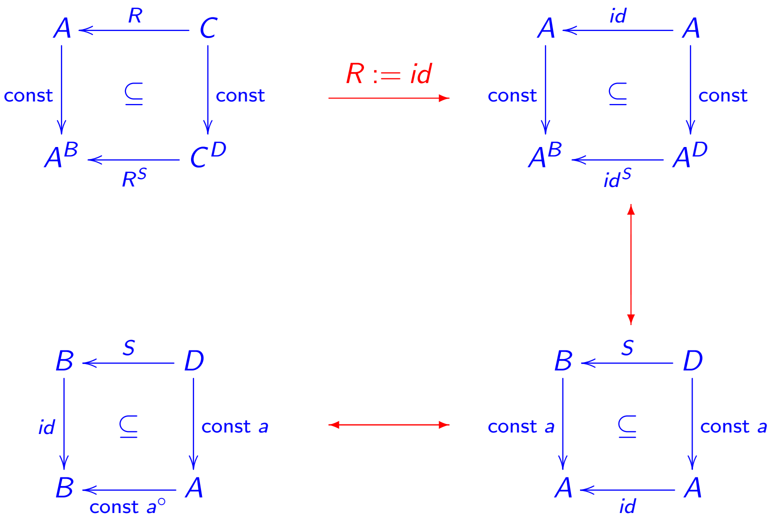
Third example (constant functions)



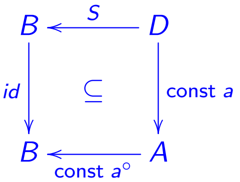
Third example (constant functions)



Third example (constant functions)



Third example (constant functions)



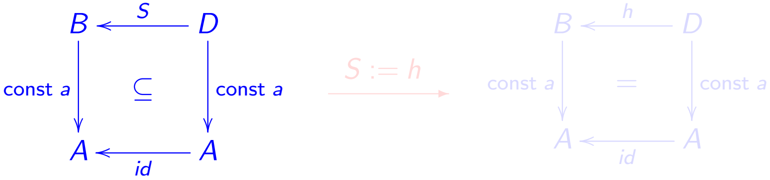
$S \subseteq (\text{const } a)^\circ \cdot (\text{const } a)$

So $(\text{const } a)^\circ \cdot \text{const } a$ is the largest possible S, i.e. the **top relation** $\top$:

$$(\text{const } a)^\circ \cdot (\text{const } a) = \top \tag{215}$$

Thus no other function can be **less injective** than const a.

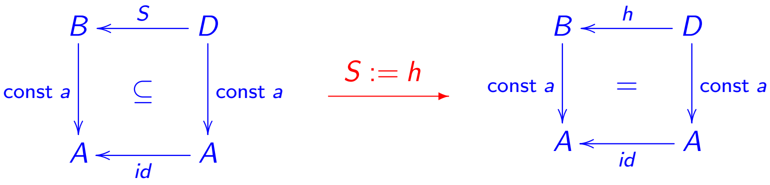
Third example (constant functions)



$$\text{const } a \cdot h = \text{const } a$$

$$h \cdot (\text{const } c) = \text{const } (h\ c)$$

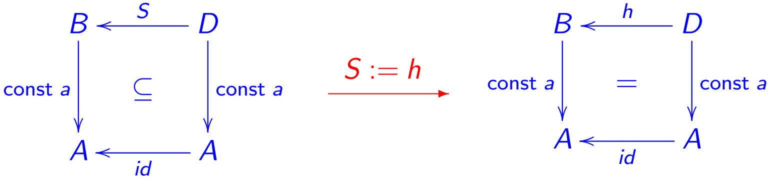
Third example (constant functions)



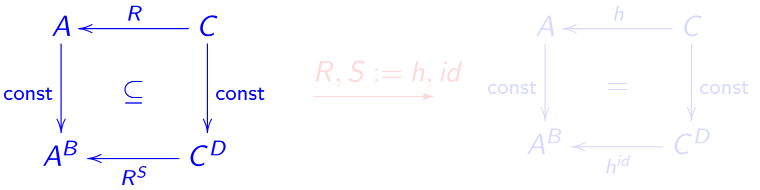
$$\text{const } a \cdot h = \text{const } a$$

$$h \cdot (\text{const } c) = \text{const } (h\ c)$$

Third example (constant functions)

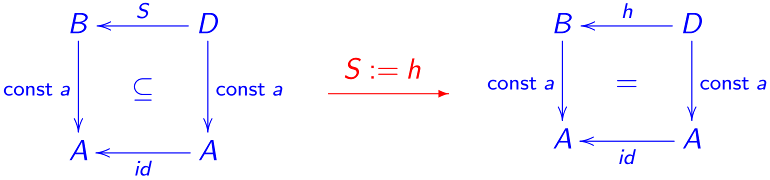


$$\text{const}\ a \cdot h = \text{const}\ a$$

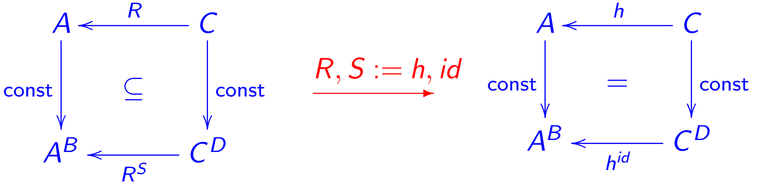


$$h \cdot (\text{const}\ c) = \text{const}\ (h\ c)$$

Third example (constant functions)



$$\mathsf{const}\ a \cdot h = \mathsf{const}\ a$$



$$h \cdot (\mathsf{const}\ c) = \mathsf{const}\ (h\ c)$$

Fourth example: filtering

Type in Haskell:

$$\mathsf{filter} :: (a \to \mathbb{B}) \to [a] \to [a]$$

Because $\mathbb{B}$ is not parametric, the **relational type** of filter is

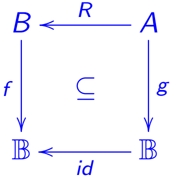
$$\mathsf{filter} : id^R \to {R^*}^{R^*}$$

and its FT square is:

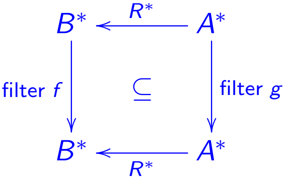
$$\begin{array}{ccc} \mathbb{B}^B & \xleftarrow{\;id^R\;} & \mathbb{B}^A \\ {\scriptstyle \mathsf{filter}}\downarrow & \subseteq & \downarrow{\scriptstyle \mathsf{filter}} \\ {B^*}^{B^*} & \xleftarrow[{R^*}^{R^*}]{} & {A^*}^{A^*} \end{array} \qquad (216)$$

FT square decomposition for filter

Thus we see that (216) unfolds into two squares: first



which is sufficient for:



Altogether:

$$f \cdot R \subseteq g \Rightarrow (\text{filter } f) \cdot R^* \subseteq R^* \cdot (\text{filter } g) \tag{217}$$

Exercises

Exercise 86: From (217) infer

$$\mathsf{filter}\ p\ [\,] = [\,]$$

by an argument similar to what was done for *reverse* earlier on. □

Exercise 87: 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: $serial : (a \rightarrow \mathbb{N}) \rightarrow a^{\star} \rightarrow a^{\star}$.
Calculate the FT of *serial*. □

Fifth example: Sorting

Type in Haskell:

$sort :: Ord\ a \Rightarrow [a] \rightarrow [a]$

Because of *Ord*, the relational type of *sort* is not $sort : R^* \rightarrow R^*$ (as was the caso of *reverse*) but rather:

$sort : (R \rightarrow id^R) \rightarrow (R^* \rightarrow R^*)$

cf:

class $Eq\ a \Rightarrow Ord\ a$ **where**
  $compare :: a \rightarrow a \rightarrow Ordering$

Fifth example: Sorting

Haskell:

$$\textbf{class } Eq\ a \Rightarrow Ord\ a \textbf{ where}$$
$$\quad compare :: a \to a \to Ordering$$

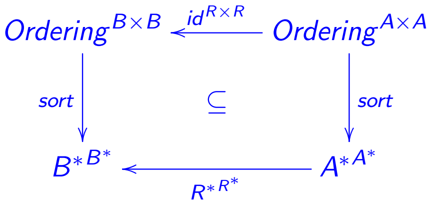
We can uncurry *compare* and work with:

$$sort : (R \times R \to id) \to (R^* \to R^*)$$

that is, we have the **relational type**:

$$sort : id^{R \times R} \to {R^*}^{R^*}$$

Fifth example: Sorting



(218)

means

$$sort \cdot (id^{R \times R}) \subseteq R^{*R^*} \cdot sort$$

that is

$$id^{R \times R} \subseteq sort^{\circ} \cdot R^{*R^*} \cdot sort$$

that is (adding variables):

$$f\ (id^{R \times R})\ g \Rightarrow (sort\ f)\ R^{*R^*}\ (sort\ g)$$

Sorting — FT square

Again we see that (218) unfolds into two squares: first

$$
\begin{array}{ccc}
B \times B & \xleftarrow{R \times R} & A \times A \\
f \downarrow & \subseteq & \downarrow g \\
Ordering & \xleftarrow[id]{} & Ordering
\end{array}
$$

which is sufficient for:

$$
\begin{array}{ccc}
B^* & \xleftarrow{R^*} & A^* \\
sort\ f \downarrow & \subseteq & \downarrow sort\ g \\
B^* & \xleftarrow[R^*]{} & A^*
\end{array}
$$

Altogether:

$$
f \cdot (R \times R) \ \subseteq \ g \Rightarrow (sort\ f) \cdot R^* \ \subseteq \ R^* \cdot (sort\ g)
$$

Sorting — FT square

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 88: 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. □

Exercise 89: Consider the following function from Haskell's Prelude:

$$\begin{array}{l} \mathit{findIndices} :: (a \rightarrow \mathbb{B}) \rightarrow [a] \rightarrow [\mathbb{Z}] \\ \mathit{findIndices}\ p\ xs = [i \mid (x, i) \leftarrow \mathsf{zip}\ xs\ [0 \mathinner{..}], p\ x] \end{array}$$

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. □

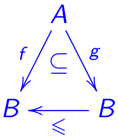
Exercise 90: Choose arbitrary functions from Haskell's Prelude and calculate their FT. □

Exercises

Exercise 91: Recalling (79), 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 $f \stackrel{.}{\leqslant} g$. In symbols:

$$f \stackrel{.}{\leqslant} g \quad = \quad f \subseteq (\leqslant) \cdot g$$

cf. diagram



Show that implication

$$f \stackrel{.}{\leqslant} g \quad \Rightarrow \quad (map\ f) \stackrel{.}{\leqslant}^{\star} (map\ g) \qquad (219)$$

follows from the *FT* of the function $map\ :\ (a \to b) \to a^{\star} \to b^{\star}$. □

Exercise 92: Infer the FT of the McCarthy conditional:

$$(\_) \to (\_), (\_) : (A \to \mathbb{B}) \to (A \to B) \to (A \to B) \to (A \to B)$$

□

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 93: Infer the FT of the following function, written in Haskell syntax,

$$\mathbf{while} :: (a \rightarrow \mathbb{B}) \rightarrow (a \rightarrow a) \rightarrow (a \rightarrow b) \rightarrow a \rightarrow b$$
$$\mathbf{while}\ p\ f\ g\ x = \mathbf{if}\ \neg\ (p\ x)\ \mathbf{then}\ g\ x\ \mathbf{else}\ \mathbf{while}\ p\ f\ g\ (f\ x)$$

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:

$$\langle\forall\ k\ :\ \phi \wedge \varphi :\ \gamma\rangle \quad = \quad \langle\forall\ k\ :\ \phi :\ \varphi \Rightarrow \gamma\rangle \tag{220}$$

$$\langle\exists\ k\ :\ \phi \wedge \varphi :\ \gamma\rangle \quad = \quad \langle\exists\ k\ :\ \phi :\ \varphi \wedge \gamma\rangle \tag{221}$$

de Morgan:

$$\neg\langle\forall\ k\ :\ \phi :\ \gamma\rangle \quad = \quad \langle\exists\ k\ :\ \phi :\ \neg\ \gamma\rangle \tag{222}$$

$$\neg\langle\exists\ k\ :\ \phi :\ \gamma\rangle \quad = \quad \langle\forall\ k\ :\ \phi :\ \neg\ \gamma\rangle \tag{223}$$

One-point:

$$\langle\forall\ k\ :\ k = e :\ \gamma\rangle \quad = \quad \gamma[k := e] \tag{224}$$

$$\langle\exists\ k\ :\ k = e :\ \gamma\rangle \quad = \quad \gamma[k := e] \tag{225}$$

Background — Eindhoven quantifier calculus

Nesting:

$$\langle \forall\ a, b\ :\ \phi \wedge \varphi :\ \gamma \rangle \ = \ \langle \forall\ a\ :\ \phi :\ \langle \forall\ b\ :\ \varphi :\ \gamma \rangle \rangle \tag{226}$$

$$\langle \exists\ a, b\ :\ \phi \wedge \varphi :\ \gamma \rangle \ = \ \langle \exists\ a\ :\ \phi :\ \langle \exists\ b\ :\ \varphi :\ \gamma \rangle \rangle \tag{227}$$

Rearranging-∀:

$$\langle \forall\ k\ :\ \phi \vee \varphi :\ \gamma \rangle \ = \ \langle \forall\ k\ :\ \phi :\ \gamma \rangle \wedge \langle \forall\ k\ :\ \varphi :\ \gamma \rangle \tag{228}$$

$$\langle \forall\ k\ :\ \phi :\ \gamma \wedge \varphi \rangle \ = \ \langle \forall\ k\ :\ \phi :\ \gamma \rangle \wedge \langle \forall\ k\ :\ \phi :\ \varphi \rangle \tag{229}$$

Rearranging-∃:

$$\langle \exists\ k\ :\ \phi :\ \gamma \vee \varphi \rangle \ = \ \langle \exists\ k\ :\ \phi :\ \gamma \rangle \vee \langle \exists\ k\ :\ \phi :\ \varphi \rangle \tag{230}$$

$$\langle \exists\ k\ :\ \phi \vee \varphi :\ \gamma \rangle \ = \ \langle \exists\ k\ :\ \phi :\ \gamma \rangle \vee \langle \exists\ k\ :\ \varphi :\ \gamma \rangle \tag{231}$$

Splitting:

$$\langle \forall\ j\ :\ \phi :\ \langle \forall\ k\ :\ \varphi :\ \gamma \rangle \rangle \ = \ \langle \forall\ k\ :\ \langle \exists\ j\ :\ \phi :\ \varphi \rangle :\ \gamma \rangle \tag{232}$$

$$\langle \exists\ j\ :\ \phi :\ \langle \exists\ k\ :\ \varphi :\ \gamma \rangle \rangle \ = \ \langle \exists\ k\ :\ \langle \exists\ j\ :\ \phi :\ \varphi \rangle :\ \gamma \rangle \tag{233}$$

References

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.
Prentice-Hall, 1980.
IBSN 0138218846.

J. Voigtländer.
Free theorems simply, via dinaturality, 2019.
arXiv cs.PL 1908.07776.

P.L. Wadler.
Theorems for free!

In *4th Int. Symp. on Functional Programming Languages and Computer Architecture*, pages 347–359, London, Sep. 1989. ACM.