A Relational Approach to Software Specification and Formal Modelling

J.N. Oliveira

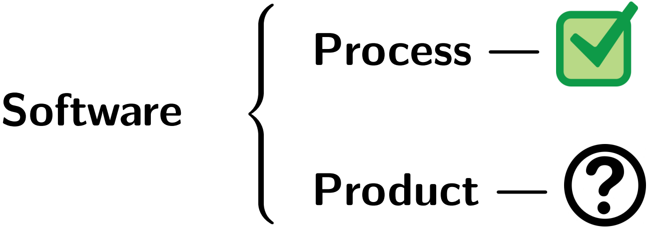
Dept. Informática & HASLAB/Univ. Minho & INESC TEC
Braga, Portugal

(Original slides: 2007 ; this version: 21 Nov 2017)

About FM

Global picture

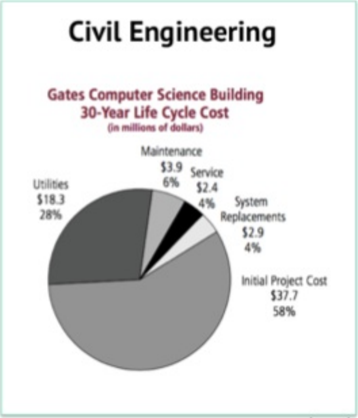
Concerning software ‘engineering’:



Formal methods provide an answer to the question mark above.

Global picture

Concerning software 'engineering':



Credits: Zhenjiang Hu, NII, Tokyop JP

Brief introduction to FM

Science

Science is about understanding how **things** work

Engineering

This is about ensuring that some **desirable things** happen repetitively and **reliably**.

Theodore Von Karman, an aerospace engineer quoted in `http://www.discoverengineering.org`, puts it in this way:

> "**Scientists** *discover the world that exists;* **engineers** *create the world that never was.*"

In both cases

Need for scientific **methods**.

Brief introduction to FM

Science

Science is about understanding how **things** work

Engineering

This is about ensuring that some **desirable things** happen repetitively and **reliably**.

Theodore Von Karman, an aerospace engineer quoted in http://www.discoverengineering.org, puts it in this way:

> "**Scientists** *discover the world that exists;* **engineers** *create the world that never was.*"

In both cases

Need for scientific **methods**.

Brief introduction to FM

Science

Science is about understanding how **things** work

Engineering

This is about ensuring that some **desirable things** happen repetitively and **reliably**.

Theodore Von Karman, an aerospace engineer quoted in http://www.discoverengineering.org, puts it in this way:

> "**Scientists** *discover the world that exists;* **engineers** *create the world that never was.*"

In both cases

Need for scientific **methods**.

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?

Some calculations

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

A model

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

— maths description of the problem.

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?

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?

Of course you have! Check this:

A problem

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

A model

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

— maths description of the problem.

Some calculations

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

The solution

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

Have you ever used a FM?

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

al-djabr

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

all-hatt

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

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

> *I have a class of students. Please write a program to list the students 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

I have a class of students. Please write a program to list the students 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

I have a class of students. Please write a program to list the students 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

I have a class of students. Please write a program to list the students 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

$sorted = \ldots marks \ldots$

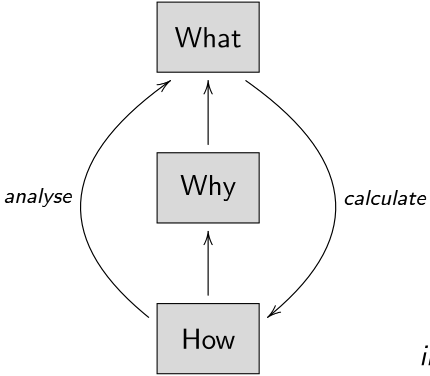
$bag = \ldots.$

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

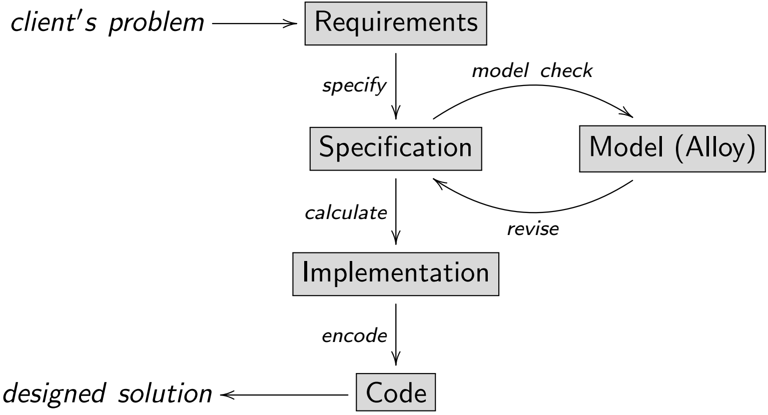


specification (*model*)

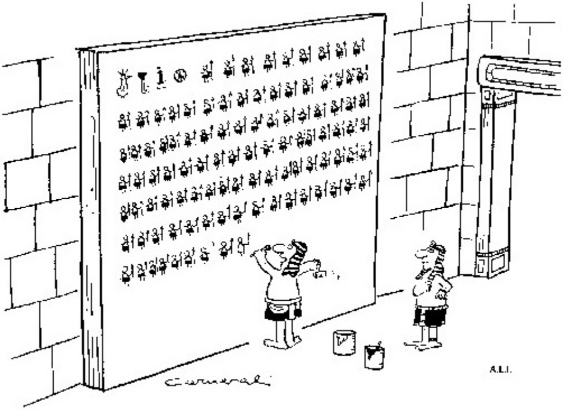
justification

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 4
- Coq
- Frama-C
- NuSMV
- Overture

Resources:

- Formal Methods Europe
- Formal Methods wiki (Oxford)

Basic Relation algebra

Relation algebra

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

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

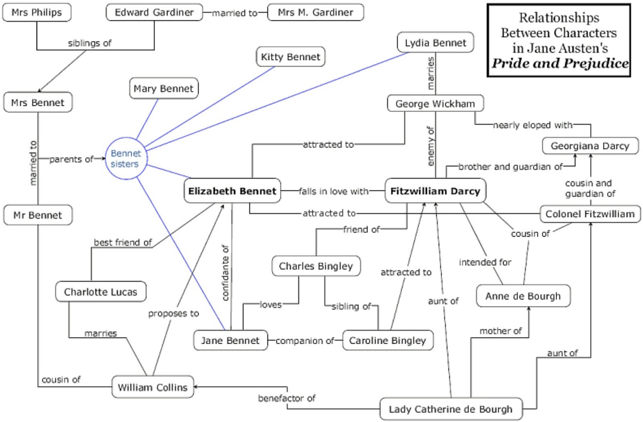
Historically, predicate logic was **not** the first to be proposed:

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

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

Everything is a relation...

... as diagram



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

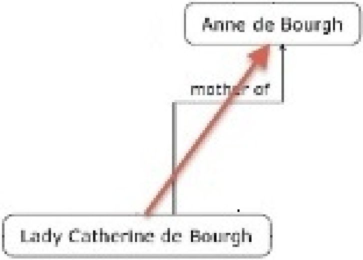
Arrow notation for relations

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

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

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

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

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

Therefore,

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

Binary Relations

Binary relations are typed:

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

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

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 $(b, a) \in R$.

Binary Relations

Binary relations are typed:

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

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

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

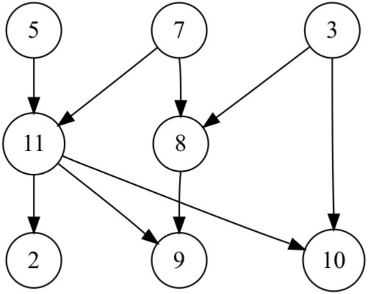
$$b \; R \; a$$

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

Binary relations are matrices

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

Relation R:



Matrix M:

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

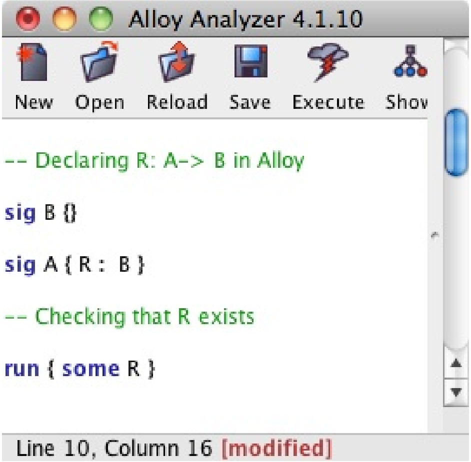
In this case $A = B = \{1..11\}$. Relations $A \xleftarrow{R} A$ over a single type are also referred to as (directed) **graphs**.

Alloy: where "everything is a relation"

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

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

We shall be using **Alloy** [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.

Propositio de homine et capra et lvpo

Data types:

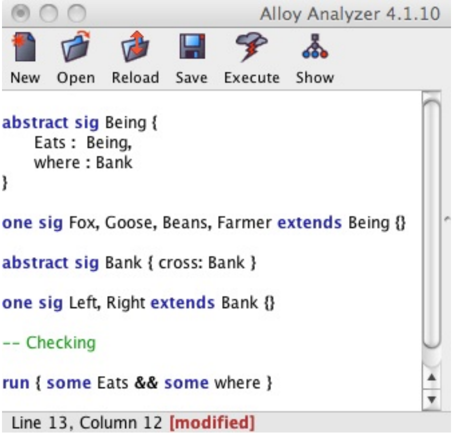
$$
\begin{aligned}
\mathit{Being} &= \{\mathit{Farmer}, \mathit{Fox}, \mathit{Goose}, \mathit{Beans}\} && (2)\\
\mathit{Bank} &= \{\mathit{Left}, \mathit{Right}\} && (3)
\end{aligned}
$$

Relations:

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

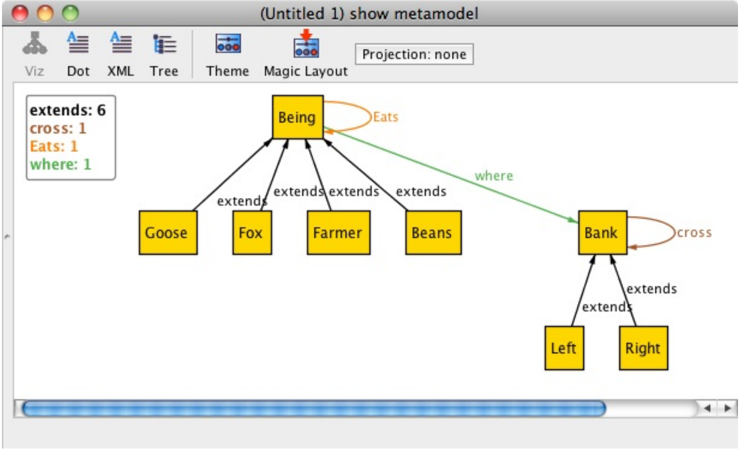
Propositio de homine et capra et lvpo

Specification source written in Alloy:



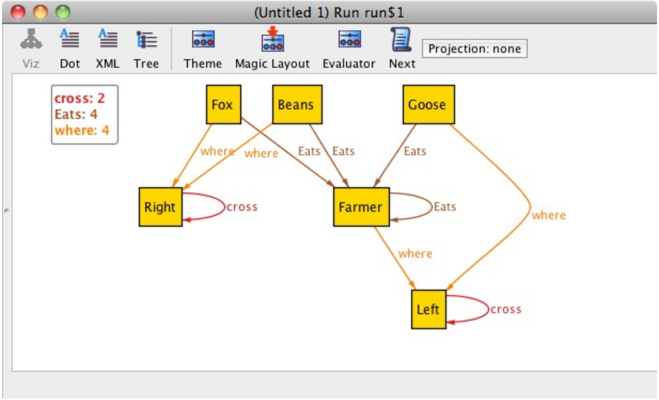
Propositio de homine et capra et lvpo

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).

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

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

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

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

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

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

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

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

Check generalization

Back to functions, (6) becomes[1]

$$
\begin{array}{ll}
b(f \cdot g)c & \equiv \langle \exists\ a\ ::\ b\ f\ a \wedge a\ g\ c \rangle \\
& \equiv \quad \{ \ a\ g\ c \text{ means } a = g\ c \ (1) \ \} \\
& \quad \langle \exists\ a\ ::\ b\ f\ a \wedge a = g\ c \rangle \\
& \equiv \quad \{ \ \exists\text{-trading } (170)\ ;\ b\ f\ a \text{ means } b = f\ a \ (1) \ \} \\
& \quad \langle \exists\ a\ :\ a = g\ c :\ b = f\ a \rangle \\
& \equiv \quad \{ \ \exists\text{-one point rule } (174) \ \} \\
& \quad b = f(g\ c)
\end{array}
$$

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

[1] Check the appendix on predicate calculus.

Relation inclusion

Relation inclusion generalizes function equality:

Equality *on functions*

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

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**:

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

Therefore:

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

Relational equality

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

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

$$
\begin{array}{ll}
 & R \\
\subseteq & \quad \{ \ \ldots. \ \} \\
 & S \\
\subseteq & \quad \{ \ \ldots. \ \} \\
 & R \\
:: & \quad \{ \ \text{“ping-pong”} \ \} \\
 & R = S
\end{array}
$$

Relation equality

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

Indirect equality *rules:*

$$
\begin{array}{rcll}
R = S & \equiv & \langle \forall\ X \ :: \ (X \subseteq R \equiv X \subseteq S) \rangle & (13) \\
 & \equiv & \langle \forall\ X \ :: \ (R \subseteq X \equiv S \subseteq X) \rangle & (14)
\end{array}
$$

The typical layout is e.g.

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

Special relations

Every type $B \longleftarrow A$ has its

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

Every type $A \longleftarrow A$ has the

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

$$id\ a \ = \ a \tag{15}$$

Clearly, for every R,

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

Diagrams

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

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

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

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

means that numbers are assigned to students sequentially. □

Exercises

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

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

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

hold and that composition is associative:

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

$\Box$

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

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

holds (moral: for functions, inclusion and equality coincide). $\Box$

(**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,

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

Note that converse commutes with composition

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

and with itself:

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

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

Function converses

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

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

cf. diagram:

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

Therefore (tell why):

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

Let us see an example of using these rules.

PF-transform at work

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

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

The other way round

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

$$
\begin{array}{ll}
 & id \subseteq f \cdot f^\circ \\
\equiv & \{ \text{ relational inclusion (8) } \} \\
 & \langle \forall\ y, x\ :\ y(id)x :\ y(f \cdot f^\circ)x \rangle \\
\equiv & \{ \text{ identity relation ; composition (6) } \} \\
 & \langle \forall\ y, x\ :\ y = x :\ \langle \exists\ z\ ::\ y\ f\ z \wedge z\ f^\circ x \rangle \rangle \\
\equiv & \{ \ \forall\text{-one point (173) ; converse (20) } \} \\
 & \langle \forall\ x\ ::\ \langle \exists\ z\ ::\ x\ f\ z \wedge x\ f\ z \rangle \rangle \\
\equiv & \{ \text{ trivia ; function } f \ \} \\
 & \langle \forall\ x\ ::\ \langle \exists\ z\ ::\ x = f\ z \rangle \rangle \\
\equiv & \{ \text{ recalling definition from maths } \} \\
 & f \text{ is } \textbf{surjective}
\end{array}
$$

Why id (really) matters

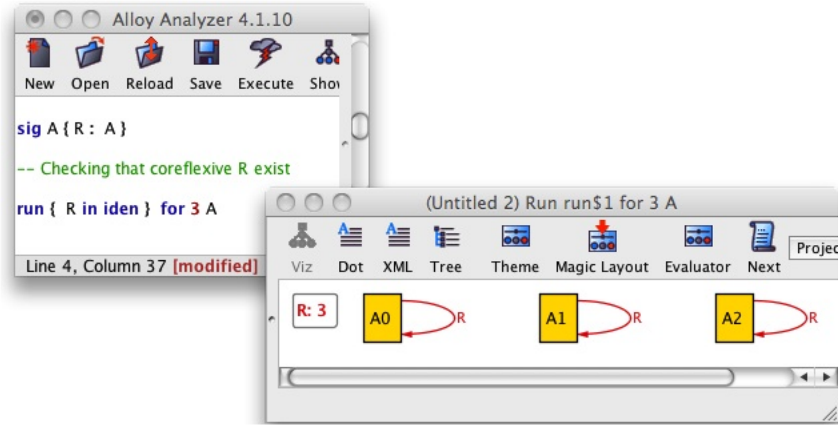
Terminology:

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

Alloy: checking for coreflexive relations



Kernels of functions

Meaning of $\ker f$:

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

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

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

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

Compute which relations are defined by the following expressions:

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

□

Binary relation taxonomy

Topmost criteria:

- binary relation
  - injective
  - entire
  - simple
  - surjective

Definitions:

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

(27)

Facts:

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

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

Binary relation taxonomy

The whole picture:

$$
\begin{tikzcd}[column sep=tiny]
 & & & \text{binary relation} \arrow[dlll, dash] \arrow[dl, dash] \arrow[dr, dash] \arrow[drrr, dash] & & & \\
\text{injective} \arrow[dr, dash] & & \text{entire} \arrow[dl, dash] \arrow[dr, dash] & & \text{simple} \arrow[dl, dash] \arrow[dr, dash] & & \text{surjective} \arrow[dl, dash] \\
 & \text{representation} \arrow[dr, dash] & & \text{function} \arrow[dl, dash] \arrow[dr, dash] & & \text{abstraction} \arrow[dl, dash] & \\
 & & \text{injection} \arrow[dr, dash] & & \text{surjection} \arrow[dl, dash] & & \\
 & & & \text{bijection} & & &
\end{tikzcd}
\tag{30}
$$

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

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

□

The same in Alloy

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

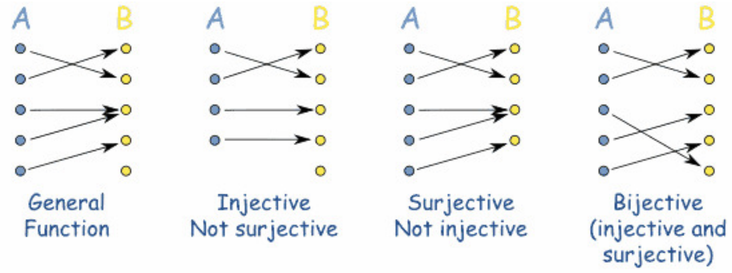
A lone -> one B	A some -> one B
injection	surjection

A one -> one B
bijection

(Courtesy of Alcino Cunha.)

Exercises

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

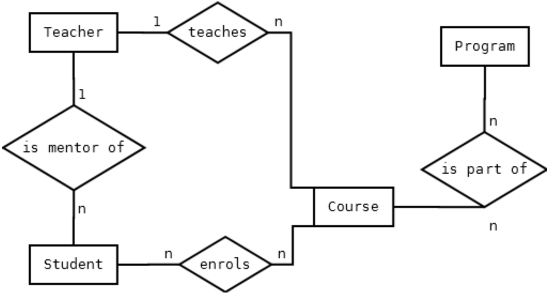


and compute, in each case, the **kernel** and the **image** of each relation. Why are all these relations **functions**? □

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

Exercises

Exercise 7: So-called "**Entity-Relationship**" (ER) diagrams are commonly used to capture relational information, e.g.[3]



Draw the same using $A \xrightarrow{R} B$ notation and tell which properties in (30) are required for each relation in the diagram. □

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

Exercises

Exercise 8: Prove the following fact

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

by completing:

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

□

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

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

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

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

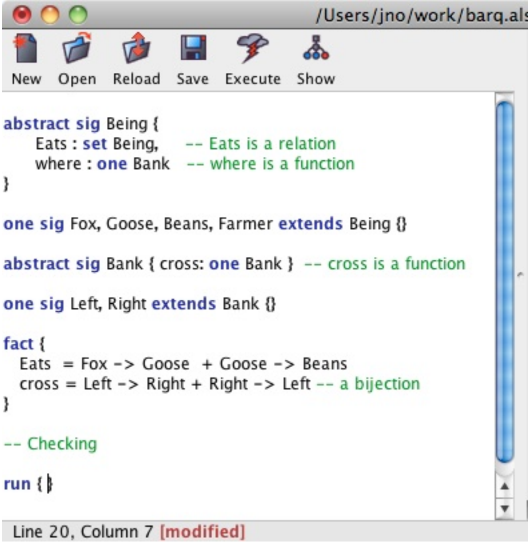
Exercise 12: There are only two **constant** functions (25) in the type $Being \longrightarrow Bank$ of *where*. Identify them and explain their role in the puzzle. □

Exercise 13: Two functions f and g are bijections iff $f^\circ = g$, recall (31). Convert $f^\circ = g$ to point-wise notation and check its meaning. □

Propositio de homine et capra et lvpo

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

(More about Alloy syntax and semantics later.)



Functions in one slide

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

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

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

Functions, relationally

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

Shunting rules:

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

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

Equality rule:

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

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

Proof of functional equality rule (34)

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

Then:

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

Dividing functions

Given functions $B \xrightarrow{g} C \xleftarrow{f} A$, we define their **division** by

$$\frac{f}{g} = g^\circ \cdot f \tag{35}$$

Exercise 14: Check the properties:

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

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

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

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

□

Exercise 15: Infer $id \subseteq \ker f$ (f is total) and $\mathrm{img}\ f \subseteq id$ (f is simple) from the shunting rules (32) or (33). □

Taxonomy of endo-relations

Besides

reflexive: iff $id \subseteq R$ (40)

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

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

transitive: iff $R \cdot R \subseteq R$ (42)

symmetric: iff $R \subseteq R^\circ (\equiv R = R^\circ)$ (43)

anti-symmetric: iff $R \cap R^\circ \subseteq id$ (44)

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

connected: iff $R \cup R^\circ = \top$ (45)

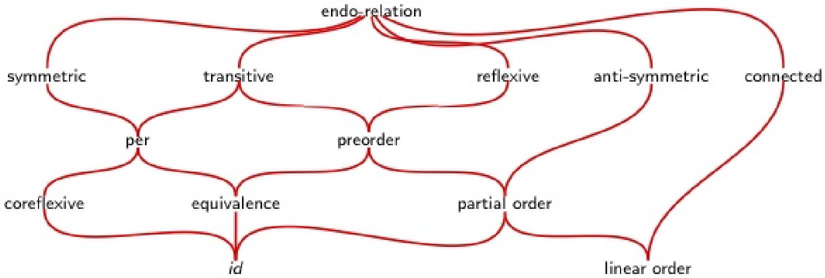
where, in general, for R, S of the same type:

$$b\ (R \cap S)\ a \equiv b\ R\ a \wedge b\ S\ a \tag{46}$$

$$b\ (R \cup S)\ a \equiv b\ R\ a \vee b\ S\ a \tag{47}$$

Taxonomy of endo-relations

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



Taxonomy of endo-relations

Exercise 16: Consider the relation

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

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

Exercise 17: Expand criteria (42) to (45) to pointwise notation. □

Exercise 18: A relation R is said to be **co-transitive** or **dense** iff the following holds:

$$\langle \forall\ b, a\ :\ b\ R\ a :\ \langle \exists\ c\ :\ b\ R\ c :\ c\ R\ a \rangle\rangle \tag{48}$$

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

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: $age\ y = age\ x$. [4]
- **Pers** are partial equivalences
  Example: $y\ IsBrotherOf\ x$.

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

Injectivity preorder

$\ker R = R^\circ \cdot R$ *measures* the level of **injectivity** of R according to the preorder

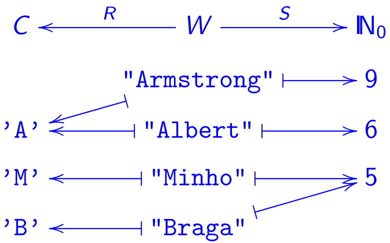
$$R \leqslant S \quad \equiv \quad \ker S \subseteq \ker R \tag{49}$$

telling that R is *less injective* or *more defined* (entire) than S.

Exercise 19: Let R and S be the two relations depicted on the right.

Check the assertions:

1. $R \leqslant S$
2. $S \leqslant R$
3. Both hold
4. None holds.



□

Exercises

Exercise 20: Check which of the following properties,

transitive, symmetric, anti-symmetric, connected

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

Exercise 21: As follow up to exercise 7,

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

□

Meet and join

Recall **meet** (intersection) and **join** (union), introduced by (46) and (47), respectively.

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

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

$$X \subseteq R \cap S \equiv X \subseteq R \wedge X \subseteq S \tag{50}$$

$$R \cup S \subseteq X \equiv R \subseteq X \wedge S \subseteq X \tag{51}$$

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

Properties

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

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

proved aside by indirect equality.

$$\begin{array}{cl}
 & X \subseteq (R \cap S) \cap T \\
\equiv & \qquad \{\ \cap\text{-universal (50) twice}\ \} \\
 & (X \subseteq R \wedge X \subseteq S) \wedge X \subseteq T \\
\equiv & \qquad \{\ \wedge \text{ is associative}\ \} \\
 & X \subseteq R \wedge (X \subseteq S \wedge X \subseteq T) \\
\equiv & \qquad \{\ \cap\text{-universal (50) twice}\ \} \\
 & X \subseteq R \cap (S \cap T) \\
:: & \qquad \{\ \text{indirection (13)}\ \} \\
 & (R \cap S) \cap T = R \cap (S \cap T) \\
\Box &
\end{array}$$

In summary

Type $B \longleftarrow A$ forms a lattice:

$$
\begin{tikzcd}[row sep=large]
 & \top \arrow[d, dotted, no head] & \\
 & R \cup S \arrow[dl, dotted, no head] \arrow[dr, dotted, no head] & \\
R \arrow[dr, no head] & & S \arrow[dl, no head] \\
 & R \cap S \arrow[d, dotted, no head] & \\
 & \bot &
\end{tikzcd}
$$

$\top$	"top"
$R \cup S$	join, lub ("least upper bound")
$R \cap S$	meet, glb ("greatest lower bound")
$\bot$	"bottom"

Exercise

Exercise 22: Let $bag : A^* \to \mathbb{N}^A$ be the function that, given a finite sequence (list) indicates the number of occurrences of its elements, for instance,

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

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

$$S = \frac{bag}{bag} \cap \frac{true}{ordered} \tag{52}$$

identify the type of S and, going pointwise and simplifying, tell which operation is specifyied by S. $\Box$

PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Back to our running example, we specify:

Being at the same bank:

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

Risk of somebody eating somebody else:

$$\mathit{CanEat} \;=\; \mathit{SameBank} \cap \mathit{Eats}$$

Then

"Starvation" is ensured by Farmer*'s presence at the same bank:*

$$\mathit{CanEat} \;\subseteq\; \mathit{SameBank} \cdot \underline{\mathit{Farmer}} \tag{53}$$

Propositio de homine et capra et lvpo

By (32), “starvation” property (53) converts to:

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

In this version, (53) can be depicted as a diagram:

$$\begin{tikzcd} \mathit{Being} \arrow[d, "\mathit{where}"'] \arrow[dr, phantom, "\subseteq"] & \mathit{Being} \arrow[l, "\mathit{CanEat}"'] \arrow[d, "\underline{\mathit{Farmer}}"] \\ \mathit{Bank} & \mathit{Being} \arrow[l, "\mathit{where}"] \end{tikzcd} \tag{54}$$

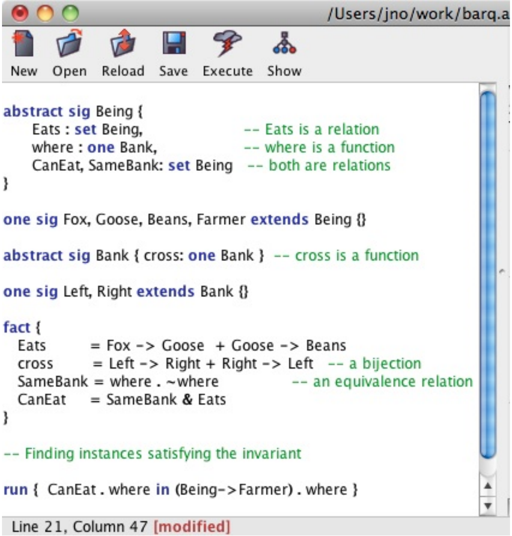
which “reads” in a nice way:

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

PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Properties which — such as (54) — are desirable and must **always hold** are called **invariants**.

See aside the 'starvation' invariant (54) written in **Alloy**.

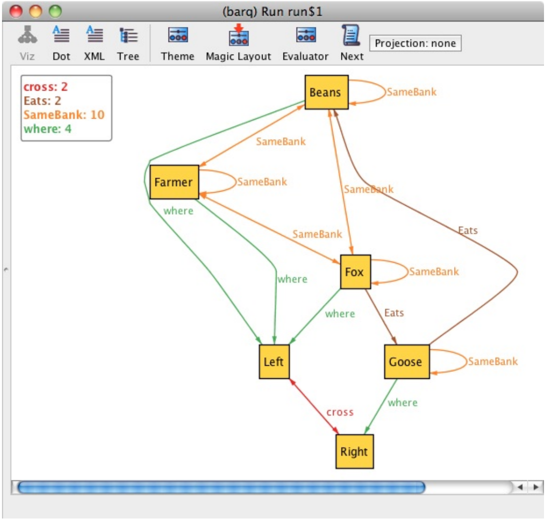



```
abstract sig Being {
    Eats : set Being,                  -- Eats is a relation
    where : one Bank,                  -- where is a function
    CanEat, SameBank: set Being        -- both are relations
}

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

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

one sig Left, Right extends Bank {}

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

-- Finding instances satisfying the invariant

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

PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Carefully observe instance of such an invariant (aside):

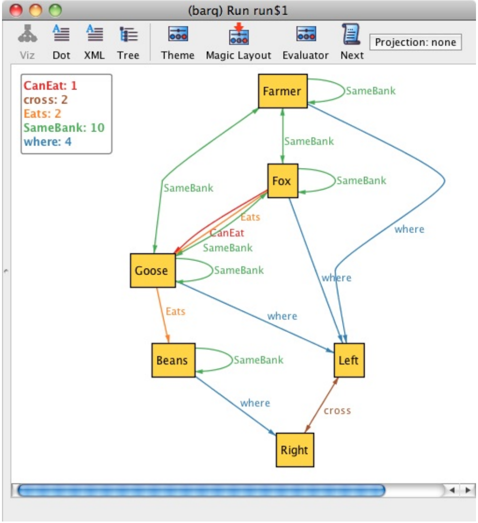
- *SameBank* is an **equivalence** — exactly the **kernel** of *where*
- *Eats* is simple but not transitive
- *cross* is a **bijection**
- *CanEat* is empty
- etc



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

$$f \cdot f^\circ \cdot f = f \tag{55}$$

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

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

---

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

---

which is very common in requirements.

Distributivity

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

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

$$(S \cup T) \cdot R \quad = \quad (S \cdot R) \cup (T \cdot R) \tag{57}$$

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

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

$$R \cdot (Q \cap S) = (R \cdot Q) \cap (R \cdot S) \quad \Leftarrow \quad \left\{ \begin{array}{c} (\mathrm{ker}\ R) \cdot Q \subseteq Q \\ \vee \\ (\mathrm{ker}\ R) \cdot S \subseteq S \end{array} \right. \tag{59}$$

Exercises

Exercise 24: The teams (T) of a football league play games (G) at home or away, and every game takes place in some date:

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

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

$$id \subseteq \frac{away}{home} \cdot \frac{away}{home} \tag{60}$$

captures one of the requirements above (which?) and that (60) amounts to forcing $home \cdot away^{\circ}$ to be symmetric. □

Exercises

Exercise 25: Show that $1 \xleftarrow{\top} 1 = id$. $\square$

Exercise 26: As generalization of exercise 1, draw the most general type diagram that accommodates relational assertion:

$$M \cdot R^\circ \subseteq \top \cdot M \tag{61}$$

$\square$

Exercise 27: Type the following relational assertions

$$M \cdot N^\circ \subseteq \bot \tag{62}$$
$$M \cdot N^\circ \subseteq id \tag{63}$$
$$M^\circ \cdot \top \cdot N \subseteq > \tag{64}$$

and check their pointwise meaning. Confirm your intuitions by repeating this exercise in Alloy. $\square$

Exercises

Exercise 28: An SQL-like relational operator is **projection**,

$$\pi_{g,f} R \stackrel{\mathrm{def}}{=} g \cdot R \cdot f^\circ \qquad \begin{tikzcd} B \arrow[d, "g"'] & A \arrow[l, "R"'] \arrow[d, "f"] \\ C & D \arrow[l, "\pi_{g,f} R"] \end{tikzcd} \tag{65}$$

whose set-theoretic meaning is

$$\pi_{g,f} R = \{(g\ b, f\ a) : b \in B \wedge a \in A \wedge b\ R\ a\} \tag{66}$$

Derive (66) from (65). $\Box$

Exercises

Exercise 29: A relation R is said to satisfy **functional dependency** (FD) $g \rightarrow f$, written $g \xrightarrow{R}\!\!\!\!\twoheadrightarrow f$ wherever projection $\pi_{f,g} R$ (65) is **simple**.

1. Recalling (49), prove the equivalence:

$$g \xrightarrow{R}\!\!\!\!\twoheadrightarrow f \quad \equiv \quad f \leqslant g \cdot R^{\circ} \tag{67}$$

2. Show that (67) trivially holds wherever g is injective and R is simple, for all (suitably typed) f.

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

$$h \twoheadleftarrow\!\!\!\!\xleftarrow{S \cdot R} g \quad \Leftarrow \quad h \twoheadleftarrow\!\!\!\!\xleftarrow{S} f \quad \wedge \quad f \twoheadleftarrow\!\!\!\!\xleftarrow{R} g \tag{68}$$

□

Monotonicity

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

$$R \subseteq S \Rightarrow R^\circ \subseteq S^\circ \tag{69}$$

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

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

$$R \subseteq S \wedge U \subseteq V \Rightarrow R \cup U \subseteq S \cup V \tag{72}$$

etc hold.

Exercise 30: Prove the **union simplicity** rule:

$$M \cup N \text{ is simple} \quad \equiv \quad M,\ N \text{ are simple and } M \cdot N^\circ \subseteq id \tag{73}$$

Derive from (73) the corresponding rule for **injective** relations. □

Proofs by $\subseteq$-transitivity

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

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

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

  and then proceed with $R \subseteq M$.

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

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

  and then proceed with $M \subseteq S$.

Example

Proof of shunting rule (32):

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

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

Exercises (monotonicity and transitivity)

Exercise 31: 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 32: Prove that relational **composition** preserves **all** relational classes in the taxonomy of (30). □

By the way: relational programming

A simple PROLOG program:

```
mother_child(trude, sally).

father_child(tom, sally).
father_child(tom, erica).
father_child(mike, tom).

parent_child(X, Y) :- father_child(X, Y).
parent_child(X, Y) :- mother_child(X, Y).

sibling(X, Y)      :- parent_child(Z, X), parent_child(Z, Y).
grand_parent(X, Y) :- parent_child(X, Z), parent_child(Z, Y).
```

Relational programming

Relational meaning:

Types:

$$P \xleftarrow{\begin{array}{c} sibling \\ grand\_parent \end{array}}_{\begin{array}{c} father\_child \\ mother\_child \\ parent\_child \end{array}} P \xleftarrow{\underline{trude}, \underline{sally}, \ldots} 1$$

Facts:

$$\begin{array}{l} mother\_child = \\ \quad \underline{trude} \cdot \underline{sally}^\circ \\ father\_child = \\ \quad \underline{tom} \cdot \underline{sally}^\circ \cup \\ \quad \underline{tom} \cdot \underline{erica}^\circ \cup \\ \quad \underline{mike} \cdot \underline{tom}^\circ \end{array}$$

Clauses:

$$mother\_child \cup father\_child \subseteq parent\_child \tag{74}$$

$$parent\_child^\circ \cdot parent\_child \subseteq sibling \tag{75}$$

$$parent\_child \cdot parent\_child \subseteq grand\_parent \tag{76}$$

Note how **type** P (for "people") is made explicit.

Relational programming

Running query

```
?- sibling(erica,sally)
```

cf. diagram

$$\begin{tikzcd} & 1 \arrow[dl, "\underline{erica}"'] \arrow[d, "\underline{sally}"] \\ P & P \arrow[l, "sibling"] \end{tikzcd}$$

corresponds to checking whether arrow $1 \xleftarrow{\underline{erica}^\circ \cdot sibling \cdot \underline{sally}} 1$ (a "scalar") is empty or not.

NB: *erica* and *sally* are **atoms** captured by constant functions $\underline{erica}$ and $\underline{sally}$, respectively.

Relational programming

Checking:

$$\top = \underline{erica}^\circ \cdot sibling \cdot \underline{sally}$$

$\equiv \quad \{ R \subseteq \top, \forall R \ ; \ 1 \xleftarrow{\top} 1 = id$, cf exercise 25 $\}$

$$id \subseteq \underline{erica}^\circ \cdot sibling \cdot \underline{sally}$$

$\Leftarrow \quad \{$ shunting (32) ; $\ker parent\_child \subseteq sibling \}$

$$\underline{erica} \subseteq \ker parent\_child \cdot \underline{sally}$$

$\Leftarrow \quad \{ \underline{tom} \cdot \underline{erica}^\circ \subseteq parent\_child$ etc $\}$

$$\underline{erica} \subseteq (\underline{tom} \cdot \underline{erica}^\circ)^\circ \cdot (\underline{tom} \cdot \underline{sally}^\circ) \cdot \underline{sally}$$

$\equiv \quad \{$ kernel of constant functions in type 1 $\}$

$$\underline{erica} \subseteq \underline{erica} \cdot id \cdot id$$

$\equiv \quad \{$ trivial $\}$

true

$\square$

Relational pairing

Pairing is among the most important operations in relation algebra:

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

We assume projections $\pi_1(a, b) = a$ and $\pi_2(a, b) = b$. Then:

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

(77)

From pairing one derives the (Kronecker) **product**:

$$R \times S = \langle R \cdot \pi_1, S \cdot \pi_2 \rangle \tag{78}$$

Relational pairing example (in matrix layout)

Example — given relations

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

pairing them up evaluates to:

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

Exercises

Exercise 33: Show that

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

PF-transforms to

$$\langle R, S\rangle \ = \ \pi_1^\circ \cdot R \cap \pi_2^\circ \cdot S \tag{79}$$

Then infer universal property

$$\pi_1 \cdot X \subseteq R \ \wedge \ \pi_2 \cdot X \subseteq S \quad \equiv \quad X \subseteq \langle R, S\rangle \tag{80}$$

from (79) via indirect equality (13). □

Exercise 34: What can you say about (80) in case X, R and S are functions? □

Exercises

Exercise 35: Unconditional distribution laws

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

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

Exercise 36: Derive from

$$
\langle R, S \rangle^{\circ} \cdot \langle X, Y \rangle = (R^{\circ} \cdot X) \cap (S^{\circ} \cdot Y) \tag{81}
$$

the following properties:

$$
\ker \langle R, S \rangle = \ker R \cap \ker S \tag{82}
$$

$\langle R, id \rangle$ is always **injective**, for whatever R

□

Relation pairing continued

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

$$\begin{aligned} \langle R, S\rangle \cdot T = {} & \langle R \cdot T, S \cdot T\rangle \\ & \Leftarrow \; R \cdot (\mathrm{img}\ T) \subseteq R \vee S \cdot (\mathrm{img}\ T) \subseteq S \end{aligned} \tag{83}$$

The **absorption** law

$$(R \times S) \cdot \langle P, Q\rangle = \langle R \cdot P, S \cdot Q\rangle \tag{84}$$

holds unconditionally.

Exercise 37: Derive from the laws of pairing studied thus far the following facts about relational product:

$$id \times id = id \tag{85}$$

$$(R \times S) \cdot (P \times Q) = (R \cdot P) \times (S \cdot Q) \tag{86}$$

□

Exercises

Exercise 38: Show that (83) holds. Suggestion: recall (58). From this infer that no side-condition is required for T simple. $\square$

Exercise 39:

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

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

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

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

Exercises

Exercise 40: Recalling (31), prove that

$$swap = \langle \pi_2, \pi_1 \rangle \tag{87}$$

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

Exercise 41: Let $\leqslant$ be a **preorder** and f be a function taking values on the carrier set of $\leqslant$.

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

□

Relational sums

Example (Haskell):

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

PF-transforms to

$$\begin{tikzcd} Bool \arrow[r, "i_1"] \arrow[dr, "Boo"'] & Bool + String \arrow[d, "{[Boo\ ,Err]}"] & String \arrow[l, "i_2"'] \arrow[dl, "Err"] \\ & X & \end{tikzcd} \qquad (88)$$

where

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

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

Relational sums

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

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

(check this).

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

$$[R\ ,S] = X \quad\equiv\quad R = X \cdot i_1 \;\wedge\; S = X \cdot i_2 \tag{89}$$

Thus $[i_1\ ,i_2] = id$ — solve (89) for R and S when $X = id$, etc etc.

Divide and conquer

The property for sums (coproducts) corresponding to (81) for products is:

$$[R\ ,S] \cdot [T\ ,U]^\circ \quad = \quad (R \cdot T^\circ) \cup (S \cdot U^\circ) \tag{90}$$

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

Exercise 42: Show that:

$$\begin{aligned} \mathrm{img}\ [R\ ,S] \quad &= \quad \mathrm{img}\ R \cup \mathrm{img}\ S && (91) \\ \mathrm{img}\ i_1 \cup \mathrm{img}\ i_2 \quad &= \quad id && (92) \end{aligned}$$

□

$+$ meets $\times$

The **exchange law**

$$[\langle R, S\rangle\ ,\langle T, V\rangle] \;=\; \langle [R\ ,\ T], [S\ ,\ V]\rangle \tag{93}$$

holds for all relations as in diagram

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

and the **fusion** law

$$\langle R, S\rangle \cdot f \;=\; \langle R \cdot f, S \cdot f\rangle \tag{94}$$

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

Exercise 43: Relying on both (89) and (94) prove (93). $\square$

Lexicographic orderings

Let $R \Rightarrow S$ be the relational operator

$$b(R \Rightarrow S)a \quad \equiv \quad (b\ R\ a) \Rightarrow (b\ S\ a) \tag{95}$$

It can be that $\Rightarrow$ has the universal property:

$$R \cap X \subseteq Y \quad \equiv \quad X \subseteq (R \Rightarrow Y) \tag{96}$$

We define the **lexicographic chaining** of two relations R and S as follows:

$$R\ ;\ S \quad = \quad R \cap (R^{\circ} \Rightarrow S) \tag{97}$$

Exercise 44: Show that (97) is the same as the universal property

$$X \ \subseteq\ (R; S) \ \equiv\ X \ \subseteq\ R \wedge X \cap R^{\circ} \ \subseteq\ S \tag{98}$$

□

Exercise

Exercise 45: Let students in a course have two numeric marks,

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

and define the **preorders**:

$$\begin{array}{rcl} \leqslant_{mark1} & = & mark1^\circ \cdot \leqslant \cdot\, mark1 \\ \leqslant_{mark2} & = & mark2^\circ \cdot \leqslant \cdot\, mark2 \end{array}$$

Spell out in pointwise notation the meaning of lexicographic ordering

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

□

Exercises

Exercise 46: (a) From (96) infer:

$$\bot \Rightarrow R = \top \tag{99}$$
$$R \Rightarrow \top = \top \tag{100}$$

(b) via indirect equality over (97) show that

$$\top ; S = S \tag{101}$$

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

$$R ; R = R \tag{102}$$

$\square$

Relational division

In the same way

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

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

$$Z \cdot Y \subseteq X \;\equiv\; Z \subseteq X / Y \tag{103}$$

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

What is the pointwise meaning of X / Y?

We reason:

First, the types of

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

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

Next, the calculation:

$$\begin{array}{ll}
 & c\ (X/Y)\ a \\
\equiv & \qquad \{ \text{ introduce points } C \xleftarrow{\underline{c}} 1 \text{ and } A \xleftarrow{\underline{a}} 1 \ \} \\
 & x(\underline{c}^\circ \cdot (X/Y) \cdot \underline{a})x \\
\equiv & \qquad \{ \text{ one-point (173) } \} \\
 & x' = x \;\Rightarrow\; x'(\underline{c}^\circ \cdot (X/Y) \cdot \underline{a})x
\end{array}$$

Proceed by going pointfree:

We reason

$$
\begin{array}{ll}
 & id \subseteq \underline{c}^\circ \cdot (X/Y) \cdot \underline{a} \\
\equiv & \qquad \{ \text{ shunting rules } \} \\
 & \underline{c} \cdot \underline{a}^\circ \subseteq X/Y \\
\equiv & \qquad \{ \text{ universal property (103) } \} \\
 & \underline{c} \cdot \underline{a}^\circ \cdot Y \subseteq X \\
\equiv & \qquad \{ \text{ now shunt } \underline{c} \text{ back to the right } \} \\
 & \underline{a}^\circ \cdot Y \subseteq \underline{c}^\circ \cdot X \\
\equiv & \qquad \{ \text{ back to points via (23) } \} \\
 & \langle \forall\, b \;:\; a \; Y \; b : \; c \; X \; b \rangle
\end{array}
$$

Outcome

In summary:

$$c \; (X/Y) \; a \;\equiv\; \langle \forall \; b \; : \; a \; Y \; b : \; c \; X \; b \rangle \qquad \begin{tikzcd} & a \arrow[dl, "X/Y"'] \\ c & b \arrow[l, "X"] \arrow[u, "Y"'] \end{tikzcd} \tag{104}$$

Example:

$a \; Y \; b$ = passenger a choses 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.

Pointwise meaning in full

The full pointwise encoding of

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

is:

$$\langle \forall\ c, b \ :\ \langle \exists\ a \ :\ cZa :\ aYb \rangle :\ cXb \rangle \;\equiv\; \langle \forall\ c, a \ :\ cZa :\ \langle \forall\ b \ :\ aYb :\ c$$

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

$$\langle \forall\ b \ :\ \langle \exists\ a \ :\ Z :\ Y \rangle :\ X \rangle \;\equiv\; \langle \forall\ a \ :\ Z :\ \langle \forall\ b \ :\ Y :\ X \rangle \rangle$$

recognizable as the **splitting** rule (181) of the Eindhoven calculus.

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

Exercises

Exercise 47: Prove the equalities

$$X \cdot f = X / f^\circ \tag{105}$$
$$X / \bot = \top \tag{106}$$
$$X / id = X \tag{107}$$

and check their pointwise meaning. □

Exercise 48: Define

$$X \setminus Y = (Y^\circ / X^\circ)^\circ \tag{108}$$

and infer:

$$a(R \setminus S)c \equiv \langle \forall\, b \,:\, b\, R\, a : \, b\, S\, c \rangle \tag{109}$$
$$R \cdot X \subseteq Y \equiv X \subseteq R \setminus Y \tag{110}$$

□

Relation difference and overriding

Relational difference $R - S$ is defined by the following universal property:

$$R - S \subseteq X \quad \equiv \quad R \subseteq S \cup X \tag{111}$$

The relational **overriding** combinator

$$R \dagger S \;=\; S \cup R \cap \bot / S^{\circ} \tag{112}$$

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

Exercise 50: (a) Show that $\bot \dagger S = S$, $R \dagger \bot = R$ and $R \dagger R = R$ hold. (b) Infer the universal property:

$$X \subseteq R \dagger S \quad \equiv \quad X - S \subseteq R \wedge (X - S) \cdot S^{\circ} = \bot \tag{113}$$

□

Predicates become relations

Recall from (35) the notation

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

and define, given a predicate p,

$$p? = id \cap \frac{true}{p} \tag{114}$$

where $true$ denotes the **constant** function yielding true for every argument.

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

Exercise 51: Show that $y\ p?\ x \;\equiv\; y = x \wedge p\ x$ $\square$

Predicates become relations

Thanks to distributive property (58) and the so-called *free theorem* of any constant function $\underline{k}$,

$$\underline{k} \cdot R \;\subseteq\; \underline{k} \tag{115}$$

— see exercise 68 later on — we get

$$p? \cdot \top = \frac{true}{p} \tag{116}$$

and then:

$$q? \cdot R \;=\; R \cap q? \cdot \top \tag{117}$$

$$R \cdot p? \;=\; R \cap \top \cdot p? \tag{118}$$

(The second is obtained from (117) by taking converses.)

Propositio de homine et capra et lvpo

Recalling the data model (4)

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

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

$$move(where, who) = where \dagger (cross \cdot where \cdot who?) \qquad (119)$$

In Alloy syntax:

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

Exercises

Exercise 52: Show that

$$R \dagger f = f$$

holds, arising from (113,111) — where f is a function, of course. □

Exercise 53: Function *move* (119) could have been defined by

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

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

$$R^f_p \quad = \quad R \dagger (f \cdot R \cdot p?) \tag{120}$$

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

□

Exercises

Exercise 54: Prove the distributive property:

$$g^\circ \cdot (R \cap S) \cdot f \quad = \quad g^\circ \cdot R \cdot f \cap g^\circ \cdot S \cdot f \tag{121}$$

Then show that

$$g^\circ \cdot p? \cdot f \quad = \quad \frac{f}{g} \cap \frac{true}{p \cdot g} \tag{122}$$

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

Exercise 55: Infer

$$q? \cdot p? = q? \cap p? \tag{123}$$

from properties (118) and (117). □

Power transpose

Implicit in how Alloy handles relations and sets is the fact that relations can be represented by functions. Let $A \xrightarrow{R} B$ be a relation in

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

such that:

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

That is:

$$\begin{tikzcd} A \to \mathcal{P}\ B \arrow[rr, bend left, "(\in\cdot)"] \arrow[rr, phantom, "\cong"] & & A \to B \arrow[ll, bend left, "\Lambda"] \end{tikzcd} \qquad f = \Lambda R \quad \equiv \quad \in \cdot f = R \qquad (124)$$

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

"Maybe" transpose

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

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

such that:

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

That is:

$$\begin{tikzcd} A \to B + 1 \arrow[rr, bend left, "(i_1^\circ \cdot)"] \arrow[rr, phantom, "\cong"] & & A \to B \arrow[ll, bend left, "\Gamma"] \end{tikzcd} \qquad f = \Gamma S \quad \equiv \quad S = i_1^\circ \cdot f \quad (125)$$

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

Contracts

Contracts

Given a function f, assume that p and q are predicates such that

$$f \cdot p? \subseteq q? \cdot f \tag{126}$$

holds. That is, $\langle \forall\ a\ :\ p\ a :\ q\ (f\ a) \rangle$ by exercise 51. In words:

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

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

$$p \xrightarrow{f} q \tag{127}$$

— a notation that generalizes the type of f. **Important**: thanks to (117), (126) can also be written: $f \cdot p? \subseteq q? \cdot \top$.

Weakest pre-conditions

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

Indeed, contract

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

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

See the calculation aside.

$$
\left\{
\begin{array}{ll}
 & f \cdot p? \subseteq q? \cdot f \\
\equiv & \qquad \{ \text{ see above (117) } \} \\
 & f \cdot p? \subseteq q? \cdot \top \\
\equiv & \qquad \{ \text{ shunting (32); (116) } \} \\
 & p? \subseteq f^{\circ} \cdot \frac{true}{q} \\
\equiv & \qquad \{ \text{ (37) } \} \\
 & p? \subseteq \frac{true}{q \cdot f} \\
\equiv & \qquad \{ \; p? \subseteq id \text{ ; (50) } \} \\
 & p? \subseteq id \cap \frac{true}{q \cdot f} \\
\equiv & \qquad \{ \text{ (114) } \} \\
 & p? \subseteq (q \cdot f)?
\end{array}
\right.
$$

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

Weakest pre-conditions

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

Exercise 56: Calculate the weakest pre-condition $\mathrm{WP}(f, q)$ for the following function / post-condition pairs:

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

□

Exercise 57: 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 (127), 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, $p? \subseteq q?$.

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 $(p \wedge q)? = p? \cdot q?$.*

Invariants versus contracts

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

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

Then the **contract**

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

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

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! Let us see a simple example first.

Library loan example

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

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

Desired properties:

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

NB: lowercase arrow labels denote functions, as usual.

Library loan example

Encoding of desired properties:

- no book on loan to more than one user:

$$Book \xrightarrow{R} User \quad \textit{is } \textbf{simple}$$

- no book without an author:

$$Book \xrightarrow{Auth} Author \quad \textit{is } \textbf{entire}$$

- no two users with the same card Id:

$$User \xrightarrow{card} Id \quad \textit{is } \textbf{injective}$$

NB: as all other arrows are functions, they are simple+entire.

Library loan example

Encoding of desired properties as relational **invariants**:

- no book on loan to more than one user:

$$\mathrm{img}\ R \subseteq id \tag{128}$$

- no book without an author:

$$id \subseteq \ker Auth \tag{129}$$

- no two users with the same card Id:

$$\ker card \subseteq id \tag{130}$$

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:

$$return\ S\ R = R - S \tag{131}$$

$$borrow\ S\ R = S \cup R \tag{132}$$

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

$$inv\ R = \mathrm{img}\ R \ \subseteq\ id \tag{133}$$

The question is, then: are the following "types"

$$inv \xleftarrow{return\ S} inv \tag{134}$$

$$inv \xleftarrow{borrow\ S} inv \tag{135}$$

ok? We check (134,135) below.

Library loan example

Checking (134):

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

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

Library loan example

At this point note that (134) 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 49) and "smaller than injective is injective" (exercise 31), it is immediate that inv (133) is preserved.

To see this better, unfold and draw definition (133):

$$inv\ R \quad = \quad \begin{tikzcd} Book \arrow[d, "R"'] \arrow[dr, phantom, "\subseteq"] & User \arrow[l, "R^\circ"'] \arrow[d, "id"] \\ User & User \arrow[l, "id"] \end{tikzcd}$$

As R is on the lower-path of the diagram, 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:

$$
\begin{array}{ll}
 & inv\ (borrow\ S\ R) \\
\equiv & \quad \{\ \text{inline definitions}\ \} \\
 & \mathrm{img}\ (S \cup R) \subseteq id \\
\equiv & \quad \{\ \text{exercise 30}\ \} \\
 & \mathrm{img}\ R \subseteq id \wedge \mathrm{img}\ S \subseteq id \wedge S \cdot R^{\circ} \subseteq id \\
\equiv & \quad \{\ \text{definition of } inv\ \} \\
 & inv\ R \wedge \underbrace{\mathrm{img}\ S \subseteq id \wedge S \cdot R^{\circ} \subseteq id}_{\mathrm{WP}(borrow\ S, inv)}
\end{array}
$$

Library loan example (Alloy)

Note, however, that in general our **workflow** does not go immediately to the **calculation** of the **weakest precondition** of a **contract**.

We **model-check** first 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.

Library loan example (Alloy)

First we write the Alloy model of what we have thus far:

```
sig Book {
   title : one Title,
   isbn : one ISBN,
   Auth : some Author,
   R : lone User
}
sig User {
   name : one Name,
   add : some Address,
   card : one Id
}
sig Title, ISBN, Author,
   Name, Address, Id { }
```

```
fact {
   card .~ card in iden
        -- card is injective
}
fun borrow
     [S, R : Book → lone User] :
        Book → lone User {
     R + S
}
fun return
     [S, R : Book → lone User] :
        Book → lone User {
     R − S
}
```

Library loan example (Alloy)

As we have seen, *return* is no problem, so we focus on *borrow*.

Realizing that most attributes of *Book* and *User* don't matter wrt. checking *borrow*, we comment them all, obtaining a much smaller model:

$$
\begin{array}{l}
\text{sig } Book\ \{ R : \text{lone } User \} \\
\text{sig } User\ \{ \} \\
\text{fun } borrow \\
\quad [S, R : Book \to \text{lone } User] : \\
\qquad Book \to \text{lone } User\ \{ \\
\quad R + S \\
\}
\end{array}
$$

$$
\begin{array}{l}
\text{sig } Book\ \{ R : User \} \\
\text{sig } User\ \{ \} \\
\text{pred } inv\ \{ \\
\quad R\ \mathbf{in}\ Book \to \text{lone } User \\
\} \\
\text{fun } borrow \\
\quad [S, R : Book \to User] : \\
\qquad Book \to User\ \{ \\
\quad R + S \\
\}
\end{array}
$$

Next, we single out the **invariant**, making it explicit as a predicate (aside).

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*

$$\text{sig } Time \,\{\, r : Book \to User \,\}$$
$$\text{sig } Book \;\{\,\}$$
$$\text{sig } User \;\{\,\}$$

and re-writing *inv* accordingly (aside).

$$\begin{array}{l} \text{pred } inv\ [t : Time]\ \{ \\ \quad t \cdot r \ \mathbf{in}\ Book \to \text{lone } User \\ \} \end{array}$$

Note how $r : Time \to (Book \to User)$ 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:

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

Once we check this, for instance running

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

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:

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

The fact that this does not yield counter-examples anymore does not tell us that

- *pre* is enough in general
- *pre* is weakest.

This we have to prove by calculation — as we have seen before.

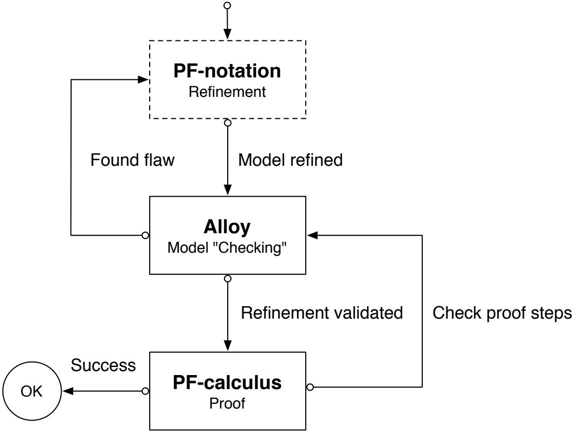
Library loan example (Alloy)

Note that pre-conditioned $borrow\ S \cdot pre?$ is not longer a **function**, because it is not **entire** anymore.

We can encode such a relation in Alloy in an easy-to-read way, as a predicate structured in two parts — pre-condition and post-condition:

pred $borrow\ [t, t' : Time, S : Book \rightarrow User]$ {
-- pre-condition
S **in** $Book \rightarrow$ lone $User$
$\sim S \cdot (t \cdot r)$ **in** iden
-- post-condition
$t' \cdot r = t \cdot r + S$
}

Alloy + Relation Algebra round-trip



Source: [6].

Summary

- The Alloy + Relation Algebra round-trip enables us to take advantage of the best of the two verification strategies.
- Diagrams of **invariants** help in detecting which **contracts** don't need to be checked.
- Functional specifications are good as starting point but soon evolve towards becoming relations, comparable to the **methods** of an OO programming language.
- Time was added to the model just to obtain more than one "state". In general, *Time* will be **linearly ordered** so that the **traces** of the model can be reasoned about.[5]

[5] In Alloy, just declare: open util/ordering[Time].

Library loan example revisited

More detailed data model of our **library** with **invariants** captured by diagram

$$
\begin{tikzcd}
\mathit{ISBN} \arrow[d, "M"'] & & \mathit{ISBN} \times \mathit{UID} \arrow[ll, "\pi_1"'] \arrow[rr, "\pi_2"] \arrow[d, "R"] \arrow[dll, phantom, "\supseteq"] \arrow[drr, phantom, "\subseteq"] & & \mathit{UID} \arrow[d, "N"] \\
\mathit{Title} \times \mathit{Publisher} \arrow[rr, "\top"'] & & \mathit{Date} & & \mathit{Name} \times \mathit{Address} \times \mathit{Phone} \arrow[ll, "\top"]
\end{tikzcd}
\tag{136}
$$

where

- M — records **books** on loan, identified by ISBN;
- N — records library **users** (identified by user id's in UID);

(both simple) and

- R — records **loan** dates.

Library loan example revisited

The two squares in the diagram impose bounds on R:

- Non-existing **books** cannot be on loan (left square);
- Only known **users** can take books home (right square).

(**NB:** in the database terminology these are known as **integrity constraints**.)

Exercise 58: Add variables to both squares in (136) so that the same conditions are expressed pointwise. Then show that the conjunction of the two squares means the same as assertion

$$R^\circ \quad \subseteq \quad \langle M^\circ \cdot \top, N^\circ \cdot \top \rangle \tag{137}$$

and draw this in a diagram. □

Library loan example revisited

Exercise 59: Consider implementing M, R and N as **files** in a relational **database**. For this, think of **operations** on the database such as, for example, that which records new loans (K):

$$borrow(K,(M,R,N)) \quad = \quad (M, R \cup K, N) \tag{138}$$

It can be checked that the **pre-condition**

$$\text{pre-}borrow(K,(M,R,N)) \quad = \quad R \cdot K^{\circ} \subseteq id$$

is necessary for maintaining (136) (why?) but it is not enough. Calculate — for a rectangle in (136) of your choice — the corresponding clause to be added to pre-$borrow$. □

Library loan example revisited

Exercise 60: The operations that **buy** new books

$$buy(X, (M, R, N)) \quad = \quad (M \cup X, R, N) \tag{139}$$

and **register** new users

$$register(Y, (M, R, N)) \quad = \quad (M, R, N \cup Y) \tag{140}$$

don't need any **pre-conditions**. Why? (Hint: compute their WP.) $\square$

NB: see annex on proofs by $\subseteq$-monotonicity for a strategy generalizing the exercise above.

Abstract interpretation

Abstraction

Model checking / proofs of particular properties may be hard to perform due to the **complexity** of **real-life** problems.

“On demand” **abstraction** can help.

By “on demand” we mean making a model more **abstract** with respect to the **property** we want to check.

In general, techniques of this kind are known as **abstract interpretation** and play a major role in **program analysis**, for instance.

We need the two extensions to functional **contracts** (126) which follow.

Relational types vs abstract simulation

A function h *is said to have* **relation type** $R \to S$, *written* $R \xrightarrow{h} S$ *if*

$$h \cdot R \subseteq S \cdot h \qquad \begin{tikzcd} B \arrow[d, "h"'] & B \arrow[l, "R"'] \arrow[d, "h"] \\ A & A \arrow[l, "S"] \end{tikzcd} \tag{141}$$

holds.

Regarding $h : B \to A$ as an **abstraction function**, we also say that $A \xleftarrow{S} A$ is an **abstract simulation** of $B \xleftarrow{R} B$.

Exercise 61: What does (141) mean in case R and S are partial orders?
□

Invariant functions

A special case of relational type defines **invariant functions**:

A function of relation type $R \xrightarrow{h} id$ *is said to be* R-**invariant**, *in the sense that*

$$\langle \forall\ b, a\ :\ b\ R\ a :\ h\ b = h\ a \rangle \qquad (142)$$

holds.

When h is R-invariant, observations by h are not affected by R-transitions.

Exercise 62: Show that an R-invariant function h is always such that $R \subseteq \frac{h}{h}$ holds.

Moreover, show that relational types compose, that is $Q \xleftarrow{k} S$ and $S \xleftarrow{h} R$ entail $Q \xleftarrow{k \cdot h} R$. □

Relational contracts

Finally, let the following definition

$$p \xrightarrow{R} q \quad \equiv \quad R \cdot p? \subseteq q? \cdot R \tag{143}$$

generalize functional contracts (126) to arbitrary relations, meaning:

$$\langle \forall\ b, a \ :\ b\ R\ a :\ p\ a \Rightarrow q\ b \rangle \tag{144}$$

Exercise 63: Sow that an alternative way of stating (143) is

$$p \xrightarrow{R} q \quad \equiv \quad R \cdot p? \subseteq q? \cdot \top \tag{145}$$

□

Exercise 24 (continued)

Exercise 64: Recalling exercise 24, let the following relation specify that two dates are at least one week apart in time:

$$d \; Ok \; d' \;\equiv\; \mid d - d' \mid > 1 \; week$$

Looking at the type diagram below right, say in your own words the meaning of the invariant specified by the relational type (141) statement below, on the left:

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

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

□

Abstract interpretation

Suppose that you want to show that $q : B \to \mathbb{B}$ is an invariant of $B \xrightarrow{R} B$, i.e. that $q \xrightarrow{R} q$ holds and you know that $q = p \cdot h$, for some $h : B \to A$.

Then you can factor your proof in two steps:

- show that there is an abstract **simulation** S such that $R \xrightarrow{h} S$
- Prove $p \xrightarrow{S} p$, that is, that p is an (abstract) **invariant** of (abstract) S.

See the calculation in the next slide.

Abstract interpretation

$$
\begin{array}{ll}
& R \cdot (p \cdot h)? \subseteq (p \cdot h)? \cdot \top \\
\equiv & \quad \{ \ (116) \text{ etc } \} \\
& R \cdot (p \cdot h)? \subseteq h^\circ \cdot p? \cdot \top \\
\equiv & \quad \{ \ \text{shunting } \} \\
& h \cdot R \cdot (p \cdot h)? \subseteq p? \cdot \top \\
\Leftarrow & \quad \{ \ R \xrightarrow{h} S \ \} \\
& S \cdot h \cdot (p \cdot h)? \subseteq p? \cdot \top \\
\Leftarrow & \quad \{ \ (p \cdot h)? \subseteq h^\circ \cdot p? \cdot h \ (122) \ \} \\
& S \cdot h \cdot h^\circ \cdot p? \cdot h \subseteq p? \cdot \top \\
\Leftarrow & \quad \{ \ \top = \top \cdot h \text{ (cancel } h\text{); } \mathrm{img}\ h \subseteq id \ \} \\
& S \cdot p? \subseteq p? \cdot \top \\
\Box &
\end{array}
$$

State-based models

Functional models generalize to so called **state-based** models in which there is

- a set Σ of **states**
- a subset $I \subseteq \Sigma$ of **initial** states
- a **step** relation $\Sigma \xrightarrow{R} \Sigma$ which expresses transition of states

We define:

- $R^0 = id$ — no action or transition takes place
- $R^{i+1} = R \cdot R^i$ — a "path" of $i + 1$ transitions.
- $R^* = \bigcup_{i>0} R^i$ — the set of all possible paths

We represent the set I by the coreflexive $\Sigma \xrightarrow{(\in I)?} \Sigma$, simplified to $\Sigma \xrightarrow{I} \Sigma$ to avoid symbol cluttering.

Safety properties

Safety properties are of the form $R^* \cdot I \;\subseteq\; S$, that is,

$$\langle \forall\; n \;:\; n \geqslant 0 :\; R^n \cdot I \;\subseteq\; S \rangle \tag{146}$$

for some safety relation $S : \Sigma \to \Sigma$, meaning:

All paths in the model originating from its initial states are **bounded** by S.

In particular, $S = \Phi \cdot \top$ — in this case,

$$\langle \forall\; n \;:\; n \geqslant 0 :\; R^n \cdot I \;\subseteq\; \Phi \cdot \top \rangle \tag{147}$$

means that formula Φ (encoded as a coreflexive) holds for every state reachable by R from an initial state.

Liveness properties

Liveness properties are of the form

$$\langle \exists\ n\ :\ n \geqslant 0 :\ Q\ \subseteq\ R^n \cdot I \rangle \tag{148}$$

for some **target** relation $Q : \Sigma \to \Sigma$, meaning:

A target relation Q is eventually **realizable**, *after n steps starting from an initial state.*

In particular, $Q = \Phi \cdot \top$ — in this case,

$$\langle \exists\ n\ :\ n \geqslant 0 :\ \Phi \cdot \top\ \subseteq\ R^n \cdot I \rangle \tag{149}$$

means that, for a sufficiently large n, formula Φ will eventually hold.

Ensuring safety / liveness properties

The first difficulty in ensuring properties such as (147) e (149) is the quantification on the number of path steps.

In the case of (149) one can try and find a particular path using a **model checker**.

In both cases, the complexity /size of the **state space** may offer some impedance to proving / model checking.

Below we show how to circumvent such difficulties by use of **abstract interpretation**.

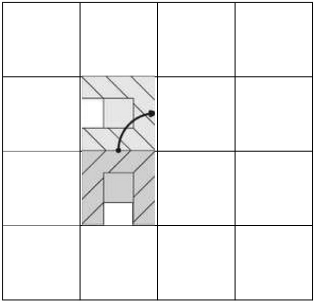
Example — Heavy armchair problem

In this problem taken from [3] the step relation is

$$R = P \times Q$$

where P captures the **adjacency** of two squares and Q captures $90°$ rotations.

A **rotation** multiplies by $\pm\, i$ a complex number in $\{1, i, -1, -i\}$ indicating the orientation of the armchair.



Altogether:

$$((y', x'), d')\ R\ ((y, x), d) \ \equiv \left\{ \begin{array}{l} y' = y \ \pm\ 1 \wedge x' = x \vee y' = y \wedge x' = x \ \pm\ 1 \\ d' = (\pm\ i)\ d \end{array} \right.$$

Heavy armchair problem

We want to check the **liveness** property:

$$\textit{For some } n,\ ((y, x+1), d)\ R^n\ ((y, x), d) \textit{ holds.} \tag{150}$$

The same, in pointfree notation:

$$\langle \exists\ n\ ::\ (id \times (1+)) \times id\ \subseteq\ R^n \rangle$$

In words: *there is a path with n steps whose meaning is* **function** $(id \times (1+)) \times id$.

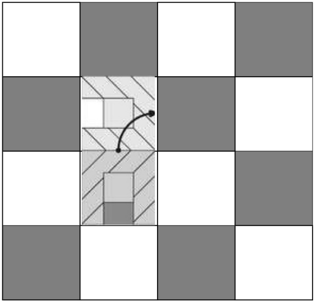
Note how the state of this problem is arbitrarily big (the squared area is unbounded).

We resort to **abstract interpretation** to obtain a bounded, **functional** model.

Heavy armchair — abstract interpretation

We color the floor as a chess board and abstract the armchair by function $h = col \times dir$ which tells the colour of the square where the armchair is and its orientation.

Since there are two colours (black, white) and two orientations (horizontal, vertical), we can model both by Booleans.



The action of moving to any adjacent square abstracts to **color** negation and any 90° rotation abstracts to **direction** negation:

$$P \xrightarrow{col} (\neg)$$

$$Q \xrightarrow{dir} (\neg)$$

Heavy armchair — abstract interpretation

Thus

$$R \xrightarrow{col \times dir} (\neg \times \neg)$$

that is, the step relation R is simulated by the function $s = col \times dir$, i.e.

$$s\ (c, d) = (\neg\ c, \neg\ d)$$

over a state space with 4 possibilities only.

At this level, we note that **observation** function

$$f\ (c, d) = c\ \oplus\ d \tag{151}$$

is s-**invariant** (142), that is

$$f \cdot s = f \tag{152}$$

since $\neg\ c\ \oplus\ \neg\ d = c\ \oplus\ d$ holds. By induction on n, $f \cdot s^n = f$.

Heavy armchair abstraction

Expressed under this abstraction, (150) is rephrased into: *there is a number of steps* n *such that* $s^n\ (c,d) = (\neg\ c, d)$ *holds*.

Aside we check this, assuming variable n existentially quantified:

$$
\begin{array}{ll}
 & s^n\ (c,d) = (\neg\ c, d) \\
\Rightarrow & \qquad \{\ \text{Leibniz}\ \} \\
 & f\ (s^n\ (c,d)) = f\ (\neg\ c, d) \\
\equiv & \qquad \{\ f \text{ is } s\text{-}\textbf{invariant}\ \} \\
 & f\ (c,d) = f\ (\neg\ c, d) \\
\equiv & \qquad \{\ (151)\ \} \\
 & c \oplus d = \neg\ c \oplus d \\
\equiv & \qquad \{\ 1 \oplus d = \neg\ d \text{ and } 0 \oplus d = d\ \} \\
 & d = \neg\ d \\
\equiv & \qquad \{\ \text{trivia}\ \} \\
 & \mathit{false}
\end{array}
$$

Thus, for all paths of arbitrary length n, $s^n\ (c,d) \neq (\neg\ c, d)$.

Alcuin puzzle example

16 possible states of type $\mathit{Being} \to \mathit{Bank}$, $2^4 = 16$.

Symmetry of the problem invites us to unify *Fox* with *Beans* [3]:

$$f : \mathit{Being} \to \{\alpha, \beta, \gamma\}$$

$$f = \left(\begin{tikzcd} \mathit{Goose} \arrow[r] & \alpha \\ \mathit{Fox} \arrow[r] & \beta \\ \mathit{Beans} \arrow[ur] & \\ \mathit{Farmer} \arrow[r] & \gamma \end{tikzcd}\right)$$

So we define a **state-abstraction** function based on f

$$h : (\mathit{Being} \to \mathit{Bank}) \to (\{\alpha, \beta, \gamma\} \to \{0, 1, 2\})$$
$$h\ w\ x = \langle \textstyle\sum\ b\ :\ x = f\ b \wedge w\ b = \mathit{Left}\ :\ 1 \rangle$$

Alcuin puzzle example

For instance,

$$\begin{array}{rcl} h\ \underline{Left} & = & 121 \\ h\ \underline{Right} & = & 000 \end{array}$$

abbreviating the mapping $\{\alpha \mapsto x, \beta \mapsto y, \gamma \mapsto z\}$ by the vector xyz.

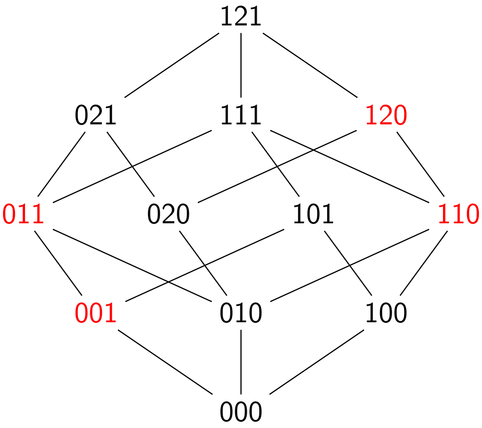
Moreover, to obtain the other bank, we use the a complement operator:

$$\overline{x} = 121 - x$$

Note that there are $2 \times 3 \times 2 = 12$ possible state vectors.

Alcuin puzzle abstraction

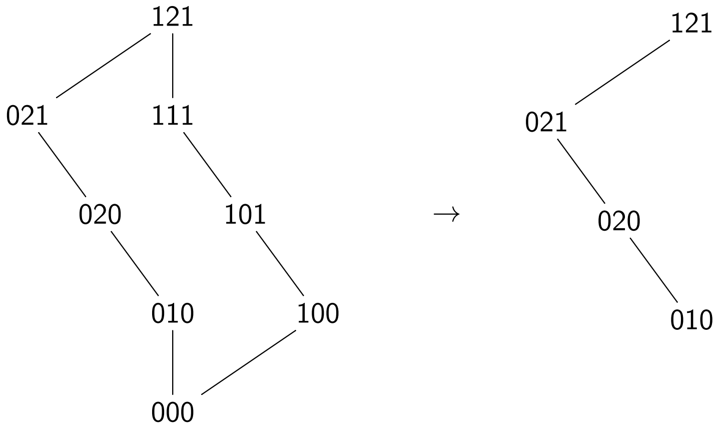
8 valid state vectors ordered by $(\leqslant)$:



The four invalid states are marked in red.

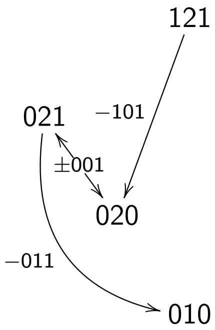
Only 4 state vectors required

Due to complementation, we only need to reach state 010, and then reverse the path through the complements:



Alcuin puzzle: abstract determinism

Abstract automaton:



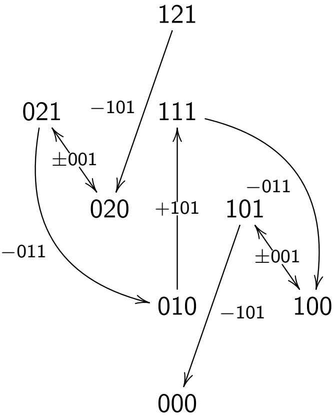
Termination is ensured by disabling toggling between states 021 and 020:

$$
\begin{array}{r}
121 \\
-101 \\
\hline
020 \\
+001 \\
\hline
021 \\
-011 \\
\hline
010
\end{array}
$$

We then take the complemented path $111 \to 100 \to 101 \to 000$.

Alcuin puzzle: abstract solution

Altogether:



$$
\begin{array}{r}
121 \\
-101 \\
\hline
020 \\
+001 \\
\hline
021 \\
-011 \\
\hline
010 \\
+101 \\
\hline
111 \\
-011 \\
\hline
100 \\
+001 \\
\hline
101 \\
-101 \\
\hline
000
\end{array}
$$

Theorems for free

Parametric polymorphism by example

Function

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

and

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

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

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

Parametric polymorphism: why?

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

Polymorphic type signatures

Polymorphic function signature:

$$f \quad : \quad t$$

where t is a functional type, according to the following "grammar" of types:

$$t \quad ::= \quad t' \leftarrow t''$$
$$t \quad ::= \quad \mathcal{F}(t_1, \ldots, t_n) \qquad \text{type constructor } \mathcal{F}$$
$$t \quad ::= \quad v \qquad \text{type } \textit{variables } v, \text{ cf. } \textit{polymorphism}$$

What does it mean for f to be **parametrically** polymorphic?

Free theorem of type t

Let

- V be the set of type variables involved in type t
- $\{R_v\}_{v \in V}$ be a V-indexed family of relations (f_v in case all such R_v are functions).
- R_t be a relation defined inductively as follows:

$$R_{t:=v} = R_v \tag{153}$$

$$R_{t:=\mathcal{F}(t_1,\ldots,t_n)} = \mathcal{F}(R_{t_1},\ldots,R_{t_n}) \tag{154}$$

$$R_{t:=t'\leftarrow t''} = R_{t'} \leftarrow R_{t''} \tag{155}$$

Questions: What does $\mathcal{F}$ in the RHS of (154) mean? What kind of relation is $R_{t'} \leftarrow R_{t''}$? See next slides.

Background: relators

Parametric datatype $\mathcal{G}$ is said to be a **relator** [2] wherever, given a relation from A to B, $\mathcal{G}R$ extends R to $\mathcal{G}$-structures: it is a relation

$$\begin{tikzcd} A \arrow[r, dotted, no head] \arrow[d, "R"'] & \mathcal{G}A \arrow[d, "\mathcal{G}R"] \\ B \arrow[r, dotted, no head] & \mathcal{G}B \end{tikzcd} \tag{156}$$

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

$$\mathcal{G}id = id \tag{157}$$

$$\mathcal{G}\,(R \cdot S) = (\mathcal{G}\ R) \cdot (\mathcal{G}\ S) \tag{158}$$

$$\mathcal{G}(R^\circ) = (\mathcal{G}\ R)^\circ \tag{159}$$

and is monotonic:

$$R \subseteq S \quad \Rightarrow \quad \mathcal{G}R \subseteq \mathcal{G}S \tag{160}$$

Relators: *"Maybe"* example

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

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

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

$$y(id + R)x$$

$$\equiv \quad \{ \text{ unfolding the sum, cf. } id + R = [i_1 \cdot id \ , i_2 \cdot R] \ \}$$

$$y(i_1 \cdot i_1^\circ \cup i_2 \cdot R \cdot i_2^\circ)x$$

$$\equiv \quad \{ \text{ relational union (47); image } \}$$

$$y(\mathrm{img}\ i_1)x \vee y(i_2 \cdot R \cdot i_2^\circ)x$$

$$\equiv \quad \{ \text{ let } NIL \text{ be \underline{the} inhabitant of the singleton type } \}$$

$$y = x = i_1 NIL \vee \langle \exists\ b, a\ :\ y = i_2\ b \wedge x = i_2\ a :\ b\ R\ a \rangle$$

Relators: example

Take $\mathcal{F}X = X^\star$.

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

$$
\begin{aligned}
s'(R^\star)s \quad \equiv \quad & \mathit{inds}\ s' = \mathit{inds}\ s \wedge \\
& \langle \forall\ i\ :\ i \in \mathit{inds}\ s :\ (s'\ i)R(s\ i)\rangle
\end{aligned}
\tag{161}
$$

Exercise 65: Check properties (157) and (159) for the list relator defined above. □

Exercises

Exercise 66: Show that the *identity* relator $\mathcal{I}$, which is such that $\mathcal{I}\ R \;=\; R$ and the *constant* relator $\mathcal{K}$ (for a given data type K) which is such that $\mathcal{K}\ R \;=\; id_K$ are indeed relators. □

Exercise 67: Show that (Kronecker) product

$$
\begin{array}{ccc}
A & C & \mathcal{G}(A,C) = A \times C \\
R \downarrow & S \downarrow & \downarrow \mathcal{G}(R,S) = R \times S \\
B & D & \mathcal{G}(B,D) = B \times D
\end{array}
$$

is a (binary) relator. □

Background: "Reynolds arrow" operator

The following relation on functions

$$f(R \leftarrow S)g \quad \equiv \quad f \cdot S \subseteq R \cdot g \qquad \begin{tikzcd} A \arrow[d, "f"'] & B \arrow[l, "S"'] \arrow[d, "g"] \\ C & D \arrow[l, "R"] \end{tikzcd} \tag{162}$$

generalizes (141).

That is to say,

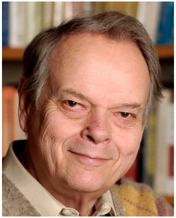
$$\frac{\begin{array}{c} A \xleftarrow{S} B \\ C \xleftarrow{R} D \end{array}}{C^A \xleftarrow{R \leftarrow S} D^B}$$

For instance, $f(id \leftarrow id)g \quad \equiv \quad f = g$ that is, $id \leftarrow id \quad = \quad id$

Free theorem (FT) of type t

The *free theorem* (FT) of type t is the following (remarkable) result due to J. Reynolds [7], advertised by P. Wadler [8] and re-written by Backhouse [1] in the pointfree style:

> *Given any function $\theta : t$, and V as above, then $\theta\ R_t\ \theta$ holds, for any relational instantiation of type variables in V.*



J.C. Reynolds
(1935–2013)

Note that this theorem

- is a result about t
- holds **independently** of the actual definition of θ.
- holds about any polymorphic function of type t

First example (id)

The target function:

$$\theta = id : a \leftarrow a$$

Calculation of $R_{t=a\leftarrow a}$:

$$
\begin{array}{cl}
 & R_{a\leftarrow a} \\
\equiv & \{ \text{ rule } \quad R_{t=t'\leftarrow t''} \quad = \quad R_{t'} \leftarrow R_{t''} \quad \} \\
 & R_a \leftarrow R_a
\end{array}
$$

Calculation of FT (R_a abbreviated to R):

$$
\begin{array}{cl}
 & id(R \leftarrow R)id \\
\equiv & \{ \quad (162) \quad \} \\
 & id \cdot R \subseteq R \cdot id
\end{array}
$$

First example (id)

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

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

cf.

$$\begin{tikzcd} a \arrow[d, "f"'] & a \arrow[l, "id"'] \arrow[d, "f"] \\ b & b \arrow[l, "id"] \end{tikzcd}$$

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

Second example (*reverse*)

The target function: $\theta = reverse : a^\star \leftarrow a^\star$.

Calculation of $R_{t=a^\star\leftarrow a^\star}$:

$$
\begin{array}{ll}
 & R_{a^\star\leftarrow a^\star} \\
\equiv & \qquad \{ \text{ rule } \; R_{t=t'\leftarrow t''} \; = \; R_{t'} \leftarrow R_{t''} \; \} \\
 & R_{a^\star} \leftarrow R_{a^\star} \\
\equiv & \qquad \{ \text{ rule } \; R_{t=\mathcal{F}(t_1,\ldots,t_n)} \; = \; \mathcal{F}(R_{t_1},\ldots,R_{t_n}) \; \} \\
 & {R_a}^\star \leftarrow {R_a}^\star
\end{array}
$$

where $s\ R^\star s'$ is given by (161). The calculation of FT follows.

Second example (*reverse*)

The FT itself will predict (R_a abbreviated to R):

$$
\begin{array}{ll}
 & reverse(R^\star \leftarrow R^\star)reverse \\
\equiv & \{ \text{ definition } \ f(R \leftarrow S)g \ \equiv \ f \cdot S \subseteq R \cdot g \ \} \\
 & reverse \cdot R^\star \subseteq R^\star \cdot reverse
\end{array}
$$

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

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

that is,

$$reverse \ [\ r \ a \mid a \leftarrow l\ ] \ = \ [\ r \ b \mid b \leftarrow reverse \ l\ ]$$

Second example (*reverse*)

Further calculation (back to R):

$$
\begin{array}{ll}
 & reverse \cdot R^{\star} \subseteq R^{\star} \cdot reverse \\
\equiv & \qquad \{ \text{ shunting rule (32) } \} \\
 & R^{\star} \subseteq reverse^{\circ} \cdot R^{\star} \cdot reverse \\
\equiv & \qquad \{ \text{ going pointwise (8, 23) } \} \\
 & \langle \forall\ s, r\ ::\ s\ R^{\star} r \Rightarrow (reverse\ s) R^{\star} (reverse\ r) \rangle
\end{array}
$$

An instance of this pointwise version of *reverse*-FT will state that, for example, *reverse* will respect element-wise orderings ($R := <$):

Second example (*reverse*)

$$length\ s = length\ r \wedge \langle \forall\ i\ :\ i \in inds\ s :\ (s\ i) < (r\ i) \rangle$$
$$\Downarrow$$
$$length(reverse\ s) = length(inv\ r)$$
$$\wedge$$
$$\langle \forall\ j\ :\ j \in inds\ s :\ (reverse\ s)j < (reverse\ r)j \rangle$$

(Guess other instances.)

Third example: FT of *sort*

Our next example calculates the FT of

$$sort : a^\star \leftarrow a^\star \leftarrow (Bool \leftarrow (a \times a))$$

where the first parameter stands for the chosen ordering relation, expressed by a binary predicate:

$$
\begin{array}{ll}
 & sort(R_{(a^\star \leftarrow a^\star)\leftarrow(Bool \leftarrow (a \times a))})sort \\
\equiv & \quad \{ \text{ (154, 153, 155); abbreviate } R_a := R \ \} \\
 & sort((R^\star \leftarrow R^\star) \leftarrow (R_{Bool} \leftarrow (R \times R)))sort \\
\equiv & \quad \{ \ R_{t:=Bool} = id \text{ (constant relator) — cf. exercise 66 } \} \\
 & sort((R^\star \leftarrow R^\star) \leftarrow (id \leftarrow (R \times R)))sort
\end{array}
$$

Third example: FT of *sort*

$$
\begin{array}{ll}
 & sort((R^\star \leftarrow R^\star) \leftarrow (id \leftarrow (R \times R)))sort \\
\equiv & \quad \{\ (162)\ \} \\
 & sort \cdot (id \leftarrow (R \times R)) \ \subseteq \ (R^\star \leftarrow R^\star) \cdot sort \\
\equiv & \quad \{\ \text{shunting } (32)\ \} \\
 & (id \leftarrow (R \times R)) \ \subseteq \ sort^\circ \cdot (R^\star \leftarrow R^\star) \cdot sort \\
\equiv & \quad \{\ \text{introduce variables } f \text{ and } g \ (8, 23)\ \} \\
 & f(id \leftarrow (R \times R))g \ \Rightarrow \ (sort\ f)(R^\star \leftarrow R^\star)(sort\ g) \\
\equiv & \quad \{\ (162) \text{ twice}\ \} \\
 & f \cdot (R \times R) \subseteq g \ \Rightarrow \ (sort\ f) \cdot R^\star \subseteq R^\star \cdot (sort\ g)
\end{array}
$$

Third example: FT of *sort*

Case $R := r$:

$$f \cdot (r \times r) = g \quad \Rightarrow \quad (sort\ f) \cdot r^\star = r^\star \cdot (sort\ g)$$

$$\equiv \qquad \{ \text{ introduce variables } \}$$

$$\left\langle \begin{array}{c} \forall\ a, b\ :: \\ f(r\ a, r\ b) = g(a, b) \end{array} \right\rangle \Rightarrow \left\langle \begin{array}{c} \forall\ l\ :: \\ (sort\ f)(r^\star\ l) = r^\star(sort\ g\ l) \end{array} \right\rangle$$

Denoting predicates f, g by infix orderings $\leqslant, \preceq$:

$$\left\langle \begin{array}{c} \forall\ a, b\ :: \\ r\ a \leqslant r\ b \equiv a \preceq b \end{array} \right\rangle \Rightarrow \left\langle \begin{array}{c} \forall\ l\ :: \\ sort\ (\leqslant)(r^\star\ l) = r^\star(sort\ (\preceq)\ l) \end{array} \right\rangle$$

That is, for r monotonic and injective,

$$sort\ (\leqslant)\ [\ r\ a \mid a \leftarrow l\ ]$$

is always the same list a

$$[\ r\ a \mid a \leftarrow sort\ (\preceq)\ l\ ]$$

Exercises

Exercise 68: Let C be a nonempty data domain and let and $c \in C$. Let $\underline{c}$ be the *"everywhere c"* function, recall (25). Show that the free theorem of $\underline{c}$ reduces to

$$\langle \forall\ R\ ::\ R \subseteq \top \rangle \tag{163}$$

□

Exercise 69: Calculate the free theorem associated with the projections $A \xleftarrow{\pi_1} A \times B \xrightarrow{\pi_2} B$ and instantiate it to (a) functions; (b) coreflexives. Introduce variables and derive the corresponding pointwise expressions. □

Exercises

Exercise 70: Consider higher order function `const: a -> b -> a` such that, given any x of type a, produces the constant function $const\ x$. Show that the equalities

$$\begin{aligned} const(f\ x) &= f \cdot (const\ x) && (164)\\ (const\ x) \cdot f &= const\ x && (165)\\ (const\ x)^\circ \cdot (const\ x) &= \top && (166) \end{aligned}$$

arise as corollaries of the *free theorem* of $const$. □

Exercises

Exercise 71: The following is a well-known Haskell function

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

Calculate the free theorem associated with its type

$$\mathit{filter} : a^\star \leftarrow a^\star \leftarrow (\mathit{Bool} \leftarrow a)$$

and instantiate it to the case where all relations are functions. $\Box$

Exercise 72: In many sorting problems, data are sorted according to a given *ranking* function which computes each datum's numeric rank (eg. students marks, credits, etc). In this context one may parameterize sorting with an extra parameter f ranking data into a fixed numeric datatype, eg. the integers: $\mathit{serial} : (a \to \mathbb{N}) \to a^\star \to a^\star$.
Calculate the FT of serial. $\Box$

Exercises

Exercise 73: Consider the following function from Haskell's Prelude:

$$\begin{aligned}&\mathit{findIndices} :: (a \to \mathbb{B}) \to [a] \to [\mathbb{Z}]\\&\mathit{findIndices}\ p\ xs = [i \mid (x,i) \leftarrow \mathrm{zip}\ xs\ [0\,..],\ p\ x]\end{aligned}$$

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

Exercises

Exercise 75: 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 $f \mathrel{\dot{\leqslant}} g$. In symbols:

$$f \mathrel{\dot{\leqslant}} g \quad = \quad f \subseteq (\leqslant) \cdot g \qquad \text{cf. diagram} \quad \begin{tikzcd} & A \arrow[dl, "f"'] \arrow[dr, "g"] \arrow[d, phantom, "\subseteq"] & \\ B & & B \arrow[ll, "\leqslant"] \end{tikzcd} \tag{167}$$

Show that implication

$$f \mathrel{\dot{\leqslant}} g \quad \Rightarrow \quad (map\ f) \mathrel{\dot{\leqslant}^{\star}} (map\ g) \tag{168}$$

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

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

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

which implements a generic `while`-loop. Derive its corollary for functions and compare your result with that produced by the tool above. $\square$

Background — Eindhoven quantifier calculus

Trading:

$$\langle\forall\ k\ :\ R \wedge S :\ T\rangle \ = \ \langle\forall\ k\ :\ R :\ S \Rightarrow T\rangle \tag{169}$$

$$\langle\exists\ k\ :\ R \wedge S :\ T\rangle \ = \ \langle\exists\ k\ :\ R :\ S \wedge T\rangle \tag{170}$$

de Morgan:

$$\neg\langle\forall\ k\ :\ R :\ T\rangle \ = \ \langle\exists\ k\ :\ R :\ \neg T\rangle \tag{171}$$

$$\neg\langle\exists\ k\ :\ R :\ T\rangle \ = \ \langle\forall\ k\ :\ R :\ \neg T\rangle \tag{172}$$

One-point:

$$\langle\forall\ k\ :\ k = e :\ T\rangle \ = \ T[k := e] \tag{173}$$

$$\langle\exists\ k\ :\ k = e :\ T\rangle \ = \ T[k := e] \tag{174}$$

Background — Eindhoven quantifier calculus

Nesting:

$$\langle \forall\ a, b\ :\ R \wedge S :\ T \rangle \ = \ \langle \forall\ a\ :\ R :\ \langle \forall\ b\ :\ S :\ T \rangle \rangle \tag{175}$$

$$\langle \exists\ a, b\ :\ R \wedge S :\ T \rangle \ = \ \langle \exists\ a\ :\ R :\ \langle \exists\ b\ :\ S :\ T \rangle \rangle \tag{176}$$

Rearranging-$\forall$:

$$\langle \forall\ k\ :\ R \vee S :\ T \rangle \ = \ \langle \forall\ k\ :\ R :\ T \rangle \wedge \langle \forall\ k\ :\ S :\ T \rangle \tag{177}$$

$$\langle \forall\ k\ :\ R :\ T \wedge S \rangle \ = \ \langle \forall\ k\ :\ R :\ T \rangle \wedge \langle \forall\ k\ :\ R :\ S \rangle \tag{178}$$

Rearranging-$\exists$:

$$\langle \exists\ k\ :\ R :\ T \vee S \rangle \ = \ \langle \exists\ k\ :\ R :\ T \rangle \vee \langle \exists\ k\ :\ R :\ S \rangle \tag{179}$$

$$\langle \exists\ k\ :\ R \vee S :\ T \rangle \ = \ \langle \exists\ k\ :\ R :\ T \rangle \vee \langle \exists\ k\ :\ S :\ T \rangle \tag{180}$$

Splitting:

$$\langle \forall\ j\ :\ R :\ \langle \forall\ k\ :\ S :\ T \rangle \rangle \ = \ \langle \forall\ k\ :\ \langle \exists\ j\ :\ R :\ S \rangle :\ T \rangle \tag{181}$$

$$\langle \exists\ j\ :\ R :\ \langle \exists\ k\ :\ S :\ T \rangle \rangle \ = \ \langle \exists\ k\ :\ \langle \exists\ j\ :\ R :\ S \rangle :\ T \rangle \tag{182}$$

References

K. Backhouse and R.C. Backhouse.
Safety of abstract interpretations for free, via logical relations and Galois connections.
SCP, 15(1–2):153–196, 2004.

R.C. Backhouse, P. de Bruin, P. Hoogendijk, G. Malcolm, T.S. Voermans, and J. van der Woude.
Polynomial relators.
In *AMAST'91*, pages 303–362. Springer-Verlag, 1992.

Roland Backhouse.
Algorithmic Problem Solving.
Wiley Publishing, 1st edition, 2011.

D. Jackson.
Software Abstractions: Logic, Language, and Analysis.
The MIT Press, Cambridge Mass., 2012.
Revised edition, ISBN 0-262-01715-2.

C.B. Jones.
Software Development — A Rigorous Approach.

Series in Computer Science. Prentice-Hall International, Upper Saddle River, NJ, USA, 1980.
C.A.R. Hoare (series editor).

J.N. Oliveira and M.A. Ferreira.
Alloy meets the algebra of programming: A case study.
IEEE Trans. Soft. Eng., 39(3):305–326, 2013.
.

J.C. Reynolds.
Types, abstraction and parametric polymorphism.
Information Processing 83, pages 513–523, 1983.

P.L. Wadler.
Theorems for free!
In *4th International Symposium on Functional Programming Languages and Computer Architecture*, pages 347–359, London, Sep. 1989. ACM.