Specification and modelling: where everything becomes a relation

J.N. Oliveira

Dept. Informática,
Universidade do Minho
Braga, Portugal

DI/UM (original slides: 2007 ; last update: Nov-2016)

Motivation

In the previous lectures you have used **predicate logic** and **finite automata** to capture the subtleties of real-life problems.

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

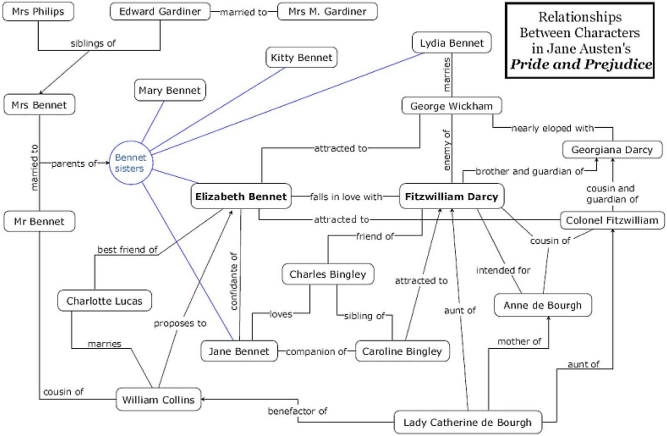
Historically, predicate logic was **not** the first to be proposed:

- Augustus de Morgan (1806-71) — recall *de Morgan* laws (121,122) — 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.)

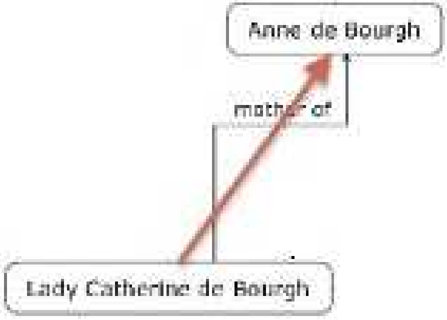
Arrow notation for relations

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

$$\textit{mother\_of} : \textit{People} \rightarrow \textit{People}$$

as the **type** of **relation** $\textit{mother\_of}$.



Pairs

Consider assertions

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

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

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

$$(0, \pi)$$
$$(\texttt{Catherine}, \texttt{Anne})$$
$$(3, 2)$$

respectively.

Sets of pairs

So, we might have written instead:

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

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

- they can be regarded as **sets of pairs**
- better, they should be regarded as **binary relations**.

Therefore,

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

Binary Relations

Binary relations are typed:

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

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

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

$$b \; R \; a$$

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

Binary Relations

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*

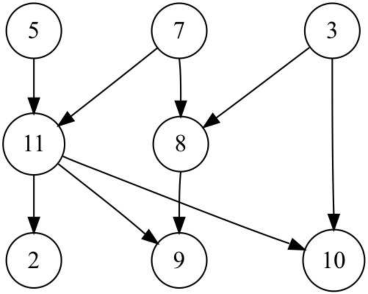
$$b \ R \ a$$

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

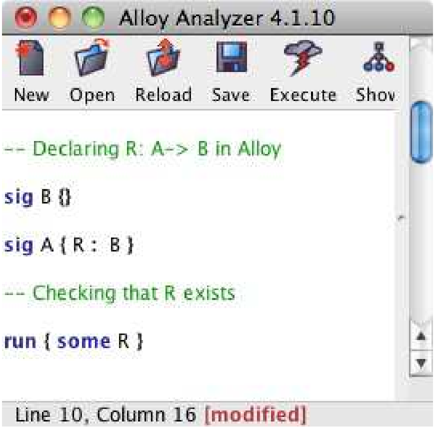
In this case $A = B = \{1..11\}$. Relations $A \xleftarrow{R} A$ over a single type are also referred to as (directed) **graphs**.

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** 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:

$$\mathit{Being} = \{\mathit{Farmer}, \mathit{Fox}, \mathit{Goose}, \mathit{Beans}\} \tag{2}$$

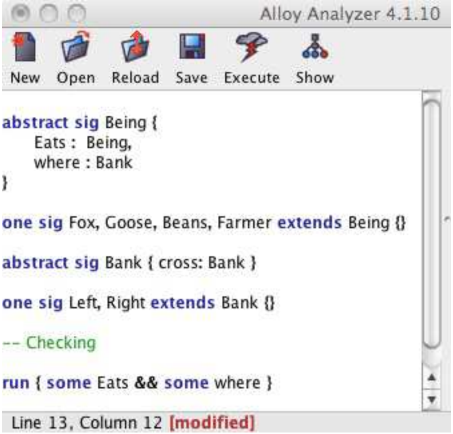
$$\mathit{Bank} = \{\mathit{Left}, \mathit{Right}\} \tag{3}$$

Relations:

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

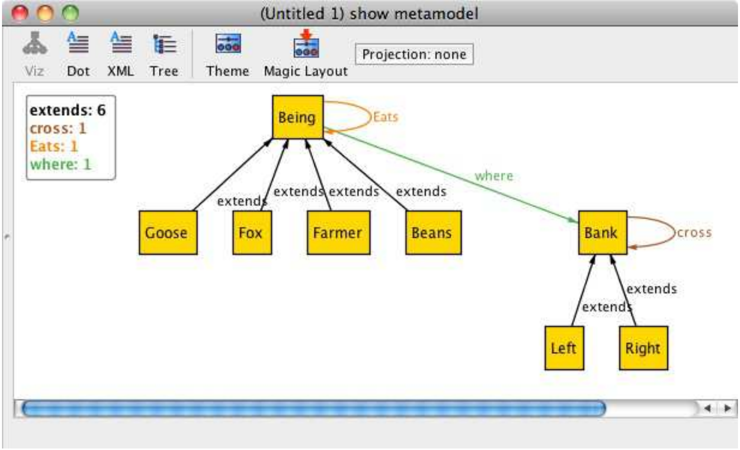
PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Specification source written in Alloy:



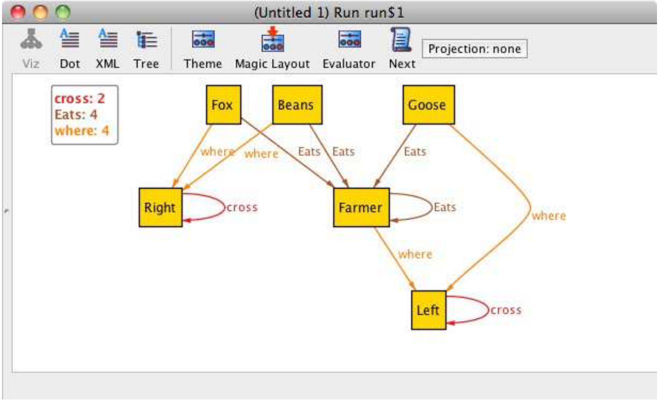
Propositio de homine et capra et lvpo

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).

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

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

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

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

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

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

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

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

Check generalization

Back to functions, (6) becomes[1]

$$
\begin{array}{rcl}
b(f \cdot g)c & \equiv & \langle \exists\, a \;::\; b\; f\; a \wedge a\; g\; c \rangle \\
& \equiv & \qquad \{ \;\; a\; g\; c \text{ means } a = g\; c \;(1) \;\; \} \\
& & \langle \exists\, a \;::\; b\; f\; a \wedge a = g\; c \rangle \\
& \equiv & \qquad \{ \;\; \exists\text{-trading } (120)\; ;\; b\; f\; a \text{ means } b = f\; a \;(1) \;\; \} \\
& & \langle \exists\, a \;:\; a = g\; c \;:\; b = f\; a \rangle \\
& \equiv & \qquad \{ \;\; \exists\text{-one point rule } (124) \;\; \} \\
& & b = f(g\; c)
\end{array}
$$

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, i.e. it is **reflexive**,

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

transitive

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

and **antisymmetric**:

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

Therefore:

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

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

$$
\begin{array}{cll}
 & R & \\
\subseteq & & \{\ \ldots.\ \} \\
 & S & \\
\subseteq & & \{\ \ldots.\ \} \\
 & R & \\
:: & & \{\ \text{"ping-pong"}\ \} \\
 & R = S &
\end{array}
$$

Relation equality

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

Indirect equality *rules:*

$$
\begin{aligned}
R = S \quad &\equiv \quad \langle \forall\, X \;::\; (X \subseteq R \equiv X \subseteq S) \rangle && (13)\\
&\equiv \quad \langle \forall\, X \;::\; (R \subseteq X \equiv S \subseteq X) \rangle && (14)
\end{aligned}
$$

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

Every type $A \longleftarrow A$ has the

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

$$id\ a \;=\; a \qquad (15)$$

Clearly, for every R,

$$\perp \;\subseteq\; R \;\subseteq\; \top \qquad (16)$$

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:

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

$\square$

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). $\square$

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

Converses

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

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

Note that converse commutes with composition

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

and with itself:

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

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

Function converses

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

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

cf. diagram:

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

Therefore (tell why):

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

Let us see an example of using these rules.

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:

$$
\begin{array}{ll}
 & id \subseteq f \cdot f^\circ \\
\equiv & \qquad \{ \text{ relational inclusion (8) } \} \\
 & \langle \forall\ y, x\ :\ y(id)x :\ y(f \cdot f^\circ)x \rangle \\
\equiv & \qquad \{ \text{ identity relation ; composition (6) } \} \\
 & \langle \forall\ y, x\ :\ y = x :\ \langle \exists\ z\ ::\ y\ f\ z \wedge z\ f^\circ x \rangle \rangle \\
\equiv & \qquad \{ \text{ } \forall\text{-one point (123) ; converse (20) } \} \\
 & \langle \forall\ x\ ::\ \langle \exists\ z\ ::\ x\ f\ z \wedge x\ f\ z \rangle \rangle \\
\equiv & \qquad \{ \text{ trivia ; function } f \text{ } \} \\
 & \langle \forall\ x\ ::\ \langle \exists\ z\ ::\ x = f\ z \rangle \rangle \\
\equiv & \qquad \{ \text{ recalling definition from maths } \} \\
 & f \text{ is } \textbf{surjective}
\end{array}
$$

Why *id* (really) matters

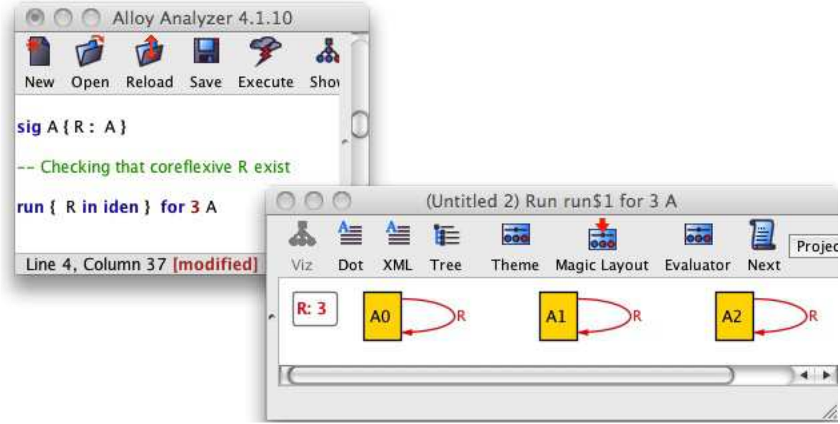
Terminology:

- 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 $\ker f$:

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

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:

$$
\begin{array}{rcl}
\underline{k} & : & A \longrightarrow K \\
\underline{k}\ a & = & k
\end{array}
\tag{25}
$$

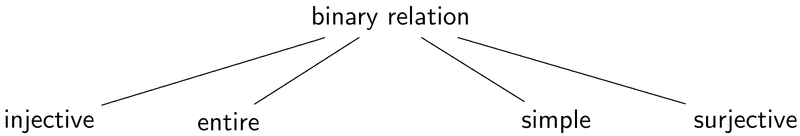
Compute which relations are defined by the following expressions:

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

□

Binary relation taxonomy

Topmost criteria:



Definitions:

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

(27)

Facts:

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

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

Binary relation taxonomy

The whole picture:

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

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)

□

Exercise

Exercise 6: Prove the following fact

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

by completing:

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

$\Box$

Propositio de homine et capra et lvpo

Exercise 7: 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" $Fox > Goose > Beans$).
□

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

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

It therefore is a bijection. Why? □

Propositio de homine et capra et lvpo

Exercise 9: 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**. $\square$

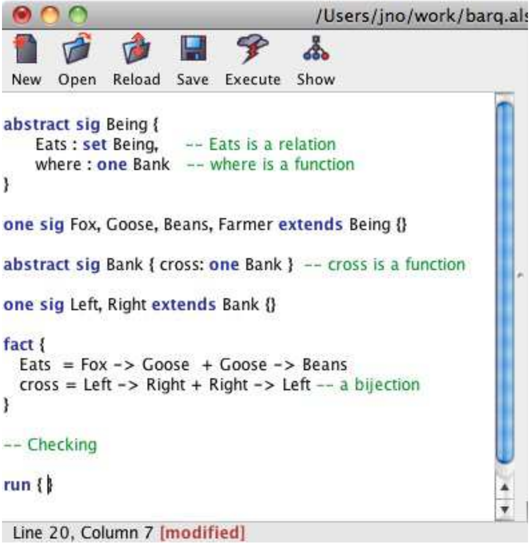
Exercise 10: There are only two **constant** functions in the type $Being \longrightarrow Bank$ of $where$. Identify them and explain their role in the puzzle. $\square$

Exercise 11: 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. $\square$

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:

$$f \cdot R \subseteq S \quad \equiv \quad R \subseteq f^\circ \cdot S \qquad (32)$$

$$R \cdot f^\circ \subseteq S \quad \equiv \quad R \subseteq S \cdot f \qquad (33)$$

Equality rule:

$$f \subseteq g \;\; \equiv \;\; f = g \;\; \equiv \;\; f \supseteq g \qquad (34)$$

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

Proof of functional equality rule (34)

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

Then:

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

$\square$

Exercises

Exercise 12: Infer $id \subseteq \ker f$ (f is total) and $\mathrm{img}\ f \subseteq id$ (f is simple) from any of the shunting rules (32) or (33). $\Box$

Exercise 13: Given two functions $B \xrightarrow{g} C \xleftarrow{f} A$ define their *division* by

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

Check the properties:

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

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

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

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

$\Box$

Taxonomy of endo-relations

Besides

$$\textbf{reflexive:} \quad \text{iff } id \subseteq R \tag{40}$$

$$\textbf{coreflexive:} \quad \text{iff } R \subseteq id \tag{41}$$

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

$$\textbf{transitive:} \quad \text{iff } R \cdot R \subseteq R \tag{42}$$

$$\textbf{anti-symmetric:} \quad \text{iff } R \cap R^\circ \subseteq id \tag{43}$$

$$\textbf{symmetric:} \quad \text{iff } R \subseteq R^\circ (\equiv R = R^\circ) \tag{44}$$

$$\textbf{connected:} \quad \text{iff } R \cup R^\circ = \top \tag{45}$$

where, in general,

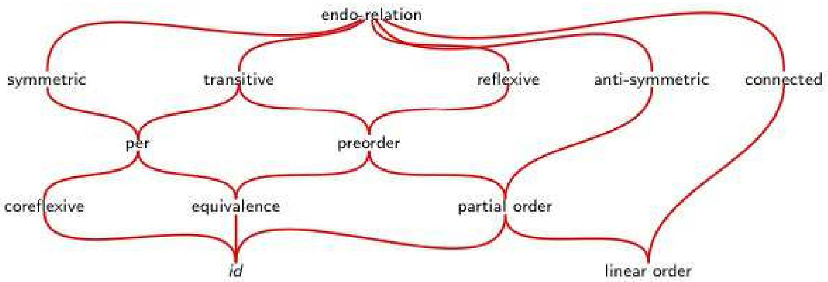
$$b\ (R \cap S)\ a \equiv b\ R\ a \wedge b\ S\ a \tag{46}$$

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

for R, S of the same type.

Taxonomy of endo-relations

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



Taxonomy of endo-relations

Exercise 14: Consider the relation

$$b\ R\ a \quad \equiv \quad \textit{team } b \textit{ is playingagainst team } a$$

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

Exercise 15: Expand criteria (42) to (45) to pointwise notation. □

Exercise 16: A relation R is said to be *co-transitive* iff the following holds:

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

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: $age\ x = age\ y$. [2]
- **Pers** are partial equivalences
  Example: $y\ IsBrotherOf\ x$.

[2] Kernels of functions are always equivalence relations, see exercise 19.

Exercises

Exercise 17: Check which of the following properties,

transitive, symmetric, anti-symmetric, connected

hold for the relation *Eats* of exercise 7. □

Exercise 18: Suppose that finite lists are represented by **simple** relations of type $A \xleftarrow{L} \mathbb{N}$, that is, as mappings from **indices** ($\mathbb{N}$) to list **elements** (A). Assuming that A is equipped with a **total order** $<_A$, show that assertion

$$L \cdot < \cdot L^{\circ} \subseteq <_A \tag{49}$$

specifies that L is a strictly **ordered** list. □

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 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.

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

In summary

Type $B \longleftarrow A$ forms a lattice:

$$
\begin{tikzcd}[arrows=-]
 & \top \arrow[d, dotted] & \\
 & R \cup S \arrow[dl, dotted] \arrow[dr, dotted] & \\
R \arrow[dr] & & S \arrow[dl] \\
 & R \cap S \arrow[d, dotted] & \\
 & \bot &
\end{tikzcd}
$$

"top"

join, lub ("least upper bound")

meet, glb ("greatest lower bound")

"bottom"

Propositio de homine et capra et lvpo

Back to our running example, we specify:

Being at the same bank:

$$SameBank \quad = \quad \ker where$$

Risk of somebody eating somebody else:

$$CanEat \quad = \quad SameBank \cap Eats$$

"Starving" ensured by Farmer's presence at the same bank:

$$CanEat \quad \subseteq \quad SameBank \cdot \underline{Farmer} \tag{52}$$

Propositio de homine et capra et lvpo

By (32), "starving" property (52) converts to:

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

In this version, (52) can be depicted as a diagram:

$$\begin{tikzcd} \mathit{Being} \arrow[d, "\mathit{where}"'] \arrow[dr, phantom, "\subseteq"] & \mathit{Being} \arrow[l, "\mathit{CanEat}"'] \arrow[d, "\underline{\mathit{Farmer}}"] \\ \mathit{Bank} & \mathit{Being} \arrow[l, "\mathit{where}"] \end{tikzcd} \tag{53}$$

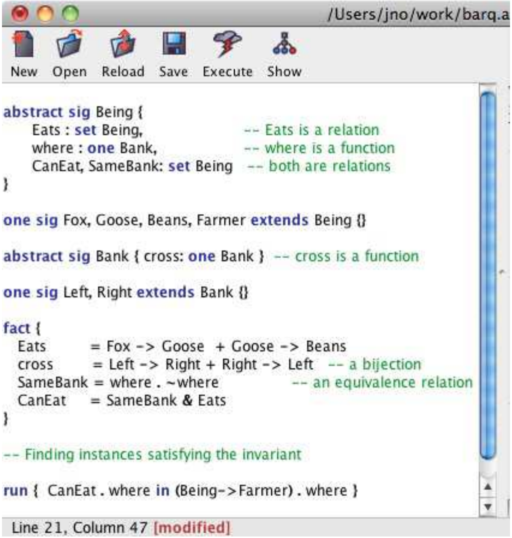
which "reads" in a nice way:

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

PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Properties which — such as (53) — are desirable and must **always hold** are called **invariants**.

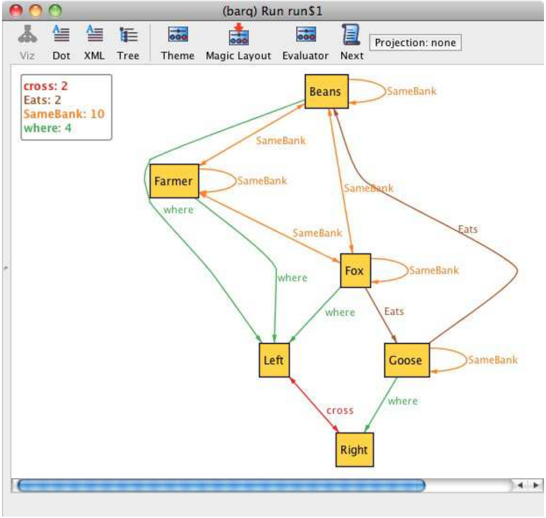
See aside the 'starving' invariant (53) written in **Alloy**.



PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Carefully observe instance of 'starving' invariant aside:

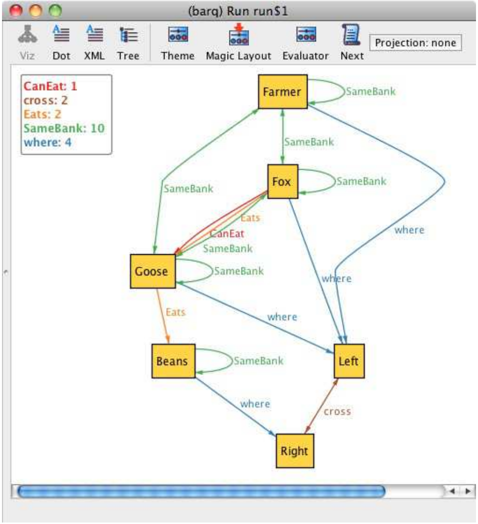
- *SameBank* is an **equivalence** — exactly the **kernel** of *where*
- *Eats* is simple but not transitive
- *cross* is a **bijection**
- *CanEat* is empty
- etc



Propositio de homine et capra et lvpo

Another instance of 'starving' invariant where:

- *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 19: Knowing that property

$$f \cdot f^{\circ} \cdot f = f \tag{54}$$

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

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

which is very common in requirements.

The football-agenda design pattern

Exercise 20: Two relations $B \xleftarrow{R} A \xrightarrow{S} C$ relate football **teams** (in A) with their scheduled **national** matches (in B) and **international** matches (in C).

Attributes $B \xrightarrow{f} D \xleftarrow{g} C$ indicate the **dates** (in D) of such matches.

Use the relational combinators you've studied so far to complete the following definition of a **property** that should ensure that no international match collides with the national matches of a particular team:

$$\Phi\,(R, S, f, g) = \;\ldots\ldots\ldots\ldots \;\subseteq\; \ldots\ldots\ldots\ldots$$

NB: Recall that properties of this kind, which should **always hold** whatever changes take place in football team agendas, are known as **invariant** properties.

□

Distributivity

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

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

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

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

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

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

Exercises

Exercise 21: As generalization of exercise 1, draw the most general type diagram that accommodates relational assertion:

$$M \cdot R^\circ \quad \subseteq \quad \top \cdot M \tag{59}$$

□

Exercise 22: Type the following relational assertions

$$M \cdot N^\circ \quad \subseteq \quad \bot \tag{60}$$
$$M \cdot N^\circ \quad \subseteq \quad id \tag{61}$$
$$M^\circ \cdot \top \cdot N \quad \subseteq \quad > \tag{62}$$

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

Exercises

Exercise 23: An SQL-like relational operator is **projection**,

$$\pi_{g,f} R \stackrel{\mathrm{def}}{=} g \cdot R \cdot f^{\circ} \qquad \begin{tikzcd} B \arrow[d, "g"'] & A \arrow[l, "R"'] \arrow[d, "f"] \\ C & D \arrow[l, "\pi_{g,f} R"] \end{tikzcd} \tag{63}$$

whose set-theoretic meaning is

$$\pi_{g,f} R = \{(g\ b, f\ a) : b\ R\ a\} \tag{64}$$

Derive (64) from (63). $\square$

Exercises

Exercise 24: A relation R is said to satisfy **functional dependency** (FD) $g \rightarrow f$, written $g \overset{R}{\twoheadrightarrow} f$ wherever projection $\pi_{f,g} R$ (63) is **simple**.

1. Prove the equivalence:

$$g \overset{R}{\twoheadrightarrow} f \quad \equiv \quad \ker\,(g \cdot R^{\circ}) \subseteq \ker f \tag{65}$$

2. Show that (65) trivially holds wherever g is injective and R is simple, for all (suitably typed) f.

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

$$h \overset{S \cdot R}{\twoheadleftarrow} g \quad \Leftarrow \quad h \overset{S}{\twoheadleftarrow} f \quad \wedge \quad f \overset{R}{\twoheadleftarrow} g \tag{66}$$

$$h \overset{S \cdot R}{\twoheadleftarrow} g \quad \Leftarrow \quad h \overset{S}{\twoheadleftarrow} f \quad \wedge \quad f \overset{R}{\twoheadleftarrow} g \tag{67}$$

$\square$

Monotonicity

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

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

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

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

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

etc hold.

Exercise 25: Prove the **union simplicity** rule:

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

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

Proofs by $\subseteq$-transitivity

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**

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

  and then proceed with $R \subseteq M$.

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

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

  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 26: 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.

$\square$

Exercise 27: Prove that relational **composition** preserves **all** relational classes in the taxonomy of (30). $\square$

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:

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

Facts:

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

Clauses:

$$mother\_child \cup father\_child \subseteq parent\_child \tag{73}$$

$$parent\_child^{\circ} \cdot parent\_child \subseteq sibling \tag{74}$$

$$parent\_child \cdot parent\_child \subseteq grand\_parent \tag{75}$$

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:

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

$$\equiv \qquad \{ \; R \subseteq \top, \forall \, R \; ; \; 1 \xleftarrow{\top} 1 \; = id \; \}$$

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

$$\Leftarrow \qquad \{ \text{ shunting (32) ; } \ker \, parent\_child \subseteq sibling \; \}$$

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

$$\Leftarrow \qquad \{ \; \underline{tom} \cdot \underline{erica}^\circ \subseteq parent\_child \text{ etc } \}$$

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

$$\equiv \qquad \{ \text{ kernel of constant functions in type } 1 \; \}$$

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

$$\equiv \qquad \{ \text{ trivial } \}$$

$$true$$

$\square$

Predicates become relations

Recall from (35) the notation

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

and define, given a predicate p,

$$p? = id \cap \frac{true}{p} \tag{76}$$

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 28: Show that $y \; p? \; x \;\; \equiv \;\; y = x \wedge p \; x$ □

Predicates become relations

Thanks to distributive property (57) and the so-called *free theorem* of any constant function $\underline{k}$,

$$\underline{k} \cdot R \subseteq \underline{k} \tag{77}$$

we get

$$p? \cdot \top = \frac{true}{p} \tag{78}$$

and then:

$$q? \cdot R = R \cap q? \cdot \top \tag{79}$$
$$R \cdot p? = R \cap \top \cdot p? \tag{80}$$

(The second is obtained from (79) by taking converses.)

Exercises

Exercise 29: Prove the distributive property:

$$g^{\circ} \cdot (R \cap S) \cdot f \quad = \quad g^{\circ} \cdot R \cdot f \cap g^{\circ} \cdot S \cdot f \tag{81}$$

Then show that

$$g^{\circ} \cdot p? \cdot f \quad = \quad \frac{f}{g} \cap \frac{true}{p \cdot g} \tag{82}$$

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

Exercise 30: Infer

$$q? \cdot p? = q? \cap p? \tag{83}$$

from properties (80) and (79). $\square$

Contracts

Now assume that, given function f, p and q are predicates such that

$$f \cdot p? \ \subseteq \ q? \cdot f \tag{84}$$

holds. That is, $\langle \forall\ a\ :\ p\ a :\ q\ (f\ a) \rangle$ by exercise 28. 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

$$p \xrightarrow{f} q \tag{85}$$

— a notation that generalizes the type of f. **Important**: thanks to (79), (84) 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.

$$
\left\{
\begin{array}{ll}
 & f \cdot p? \subseteq q? \cdot f \\
\equiv & \qquad \{ \text{ see above (79) } \} \\
 & f \cdot p? \subseteq q? \cdot \top \\
\equiv & \qquad \{ \text{ shunting (32); (78) } \} \\
 & p? \subseteq f^{\circ} \cdot \frac{true}{q} \\
\equiv & \qquad \{ \text{ (37) } \} \\
 & p? \subseteq \frac{true}{q \cdot h} \\
\equiv & \qquad \{ \; p? \subseteq id \; ; \text{(50)} \; \} \\
 & p? \subseteq id \cap \frac{true}{q \cdot f} \\
\equiv & \qquad \{ \text{ (76) } \} \\
 & p? \subseteq (q \cdot f)?
\end{array}
\right.
$$

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 31: 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 32: 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 (85), 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 **starving** 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: $\mathrm{WP}(trip\ b, starving) = starving$.

Things are not that easy, however: there is a need for a **pre-condition** ensuring that b is on the farmer's bank and is the right being to carry! Let us see a simple example first.

Library loan example

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

$u \; R \; b$ 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:

$$\mathrm{img}\ R \subseteq id \tag{86}$$

- no book without an author:

$$id \subseteq \ker Auth \tag{87}$$

- no two users with the same card Id:

$$\ker card \subseteq id \tag{88}$$

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:

$$return\ S\ R = R - S \tag{89}$$

$$borrow\ S\ R = S \cup R \tag{90}$$

NB: the first uses the operator $R - S$ of **relational difference** which is defined by the following universal property:

$$R - S \subseteq X \quad \equiv \quad R \subseteq S \cup X \tag{91}$$

Exercise 33: Show that $R - S \subseteq R$ and that $R - \bot = R$ hold.

□

Library loan example

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

$$inv\ R = \mathrm{img}\ R \subseteq id \tag{92}$$

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

$$inv \xleftarrow{return\ S} inv \tag{93}$$

$$inv \xleftarrow{borrow\ S} inv \tag{94}$$

ok?

We check (93,94) below.

Library loan example

Checking (93):

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

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

Library loan example

At this point note that (93) 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 33) and "smaller than injective is injective" (exercise 26), it is immediate that inv (92) is preserved.

To see this better, unfold and draw definition (92):

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

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:

$$
\begin{array}{ll}
 & inv\ (borrow\ S\ R) \\
\equiv & \qquad \{ \text{ inline definitions } \} \\
 & \mathrm{img}\ (S \cup R) \subseteq id \\
\equiv & \qquad \{ \text{ exercise 25 } \} \\
 & \mathrm{img}\ R \subseteq id \wedge \mathrm{img}\ S \subseteq id \wedge S \cdot R^{\circ} \subseteq id \\
\equiv & \qquad \{ \text{ definition of } inv \} \\
 & inv\ R \wedge \underbrace{\mathrm{img}\ S \subseteq id \wedge S \cdot R^{\circ} \subseteq id}_{\mathrm{WP}(borrow\ S, inv)}
\end{array}
$$

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:

$$
\begin{array}{l}
\text{sig } Book\ \{ \\
\quad title : \text{one } Title, \\
\quad isbn : \text{one } ISBN, \\
\quad Auth : some\ Author, \\
\quad R : \text{lone } User \\
\} \\
\text{sig } User\ \{ \\
\quad name : \text{one } Name, \\
\quad \text{add} : some\ Address, \\
\quad card : \text{one } Id \\
\} \\
\text{sig } Title, ISBN, Author, \\
\quad Name, Address, Id\ \{\,\}
\end{array}
$$

$$
\begin{array}{l}
\text{fact } \{ \\
\quad card \mathrel{.\tilde{}}\ card\ \textbf{in}\ \text{iden} \\
\qquad\quad \text{-- card is injective} \\
\} \\
\text{fun } borrow \\
\qquad [S, R : Book \rightarrow \text{lone } User] : \\
\qquad\quad Book \rightarrow \text{lone } User\ \{ \\
\qquad R + S \\
\} \\
\text{fun } return \\
\qquad [S, R : Book \rightarrow \text{lone } User] : \\
\qquad\quad Book \rightarrow \text{lone } User\ \{ \\
\qquad R - S \\
\}
\end{array}
$$

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:

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

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

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

Library loan example (Alloy)

In the step that follows, we make the model **dynamic**, in the sense that we need at least two instances of relation R — one before *borrow* is applied and the other after.

We introduce *Time* as a way of recording such two moments, pulling R out of *Book*

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

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

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

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:

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

Once we check this, for instance running

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

we shall obtain counter-examples. (These were expected...)

Library loan example (Alloy)

The counter-examples will quickly tell us what the problems are, guiding us to add the following pre-condition to the contract:

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

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.

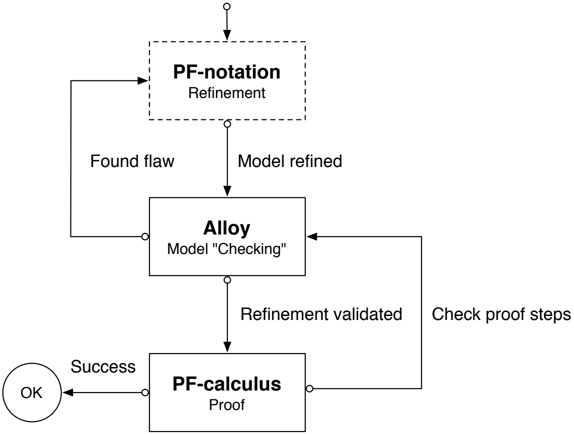
Library loan example (Alloy)

Note that pre-conditioned $\mathit{borrow}\ S \cdot \mathit{pre?}$ 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:

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

Alloy + Relation Algebra round-trip



Source: [2].

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.[3]

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

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

(95)

From pairing we derived the (Kronecker) **product**:

$$R \times S = \langle R \cdot \pi_1, S \cdot \pi_2 \rangle \tag{96}$$

Relational pairing example (in matrix layout)

Example — given

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

pairing them up evaluates to:

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

Exercises

Exercise 34: Show that

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

PF-transforms to

$$\langle R, S\rangle \ = \ \pi_1^\circ \cdot R \cap \pi_2^\circ \cdot S \tag{97}$$

Then infer universal property

$$\pi_1 \cdot X \subseteq R \ \wedge \ \pi_2 \cdot X \subseteq S \quad \equiv \quad X \subseteq \langle R, S\rangle \tag{98}$$

from (97) via indirect equality (13). □

Exercise 35: What can you say about (98) in case X, R and S are functions? □

Library loan example revisited

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

$$\begin{tikzcd}
ISBN \arrow[d, "M"'] \arrow[dr, phantom, "\supseteq"] & ISBN \times UID \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, "R"] \arrow[dr, phantom, "\subseteq"] & UID \arrow[d, "N"] \\
Title \times Publisher \arrow[r, "\top"'] & Date & Name \times Address \times Phone \arrow[l, "\top"]
\end{tikzcd} \tag{99}$$

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 36: Add variables to both squares in (99) so that the same conditions are expressed pointwise. Then show that the conjunction of the two squares means the same as assertion

$$R^\circ \quad \subseteq \quad \langle M^\circ \cdot \top, N^\circ \cdot \top \rangle \tag{100}$$

and draw this in a diagram. □

Library loan example revisited

Exercise 37: 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):

$$borrow(K, (M, R, N)) \quad \triangleq \quad (M, R \cup K, N) \tag{101}$$

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

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

is necessary for maintaining (99) (why?) but it is not enough. Calculate — for a rectangle in (99) of your choice — the corresponding clause to be added to pre-$borrow$. □

Library loan example revisited

Exercise 38: The operations that **buy** new books

$$buy(X,(M,R,N)) \quad \triangleq \quad (M \cup X, R, N) \tag{102}$$

and **register** new users

$$register(Y,(M,R,N)) \quad \triangleq \quad (M, R, N \cup Y) \tag{103}$$

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

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

Exercises

Exercise 39: Unconditional distribution laws

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

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

Exercise 40: Derive from

$$\langle R, S\rangle^\circ \cdot \langle X, Y\rangle = (R^\circ \cdot X) \cap (S^\circ \cdot Y) \tag{104}$$

the following properties:

$$\ker \langle R, S\rangle = \ker R \cap \ker S \tag{105}$$

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

□

Exercises

Exercise 41: Show that the following conditional fusion law holds:

$$\langle R, S\rangle \cdot T = \langle R \cdot T, S \cdot T\rangle \quad \Leftarrow \quad R \cdot (\mathrm{img}\ T) \subseteq R \vee S \cdot (\mathrm{img}\ T) \subseteq S$$

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

Exercise 42:

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

Exercises

Exercise 43: Recalling (31), prove that

$$swap \triangleq \langle \pi_2, \pi_1 \rangle \tag{106}$$

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

Exercise 44: 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 $\sqsubseteq \ \triangleq \ f^\circ \cdot \leqslant \cdot f$
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.

$\square$

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** (84) which follow.

Relational types vs abstract simulation

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

$$h \cdot R \subseteq S \cdot h \qquad \begin{tikzcd} B \arrow[d, "h"'] & B \arrow[l, "R"'] \arrow[d, "h"] \\ A & A \arrow[l, "S"] \end{tikzcd} \tag{107}$$

holds.

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

Exercise 45: What does (107) 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*

$$\langle \forall\ b, a\ :\ b\ R\ a :\ h\ b = h\ a \rangle \tag{108}$$

holds.

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

Exercise 46: 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 $Q \xleftarrow{k \cdot h} R$. $\square$

Relational contracts

Finally, let the following definition

$$p \xrightarrow{R} q \quad \equiv \quad R \cdot p? \subseteq q? \cdot R \tag{109}$$

generalize functional contracts (84) to arbitrary relations, meaning:

$$\langle \forall\ b, a\ :\ b\ R\ a :\ p\ a \Rightarrow q\ b \rangle \tag{110}$$

Exercise 47: Sow that an alternative way of stating (109) is

$$p \xrightarrow{R} q \quad \equiv \quad R \cdot p? \subseteq q? \cdot \top \tag{111}$$

□

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

$$
\begin{array}{ll}
 & R \cdot (p \cdot h)? \subseteq (p \cdot h)? \cdot \top \\
\equiv & \quad \{ \text{ (78) etc } \} \\
 & R \cdot (p \cdot h)? \subseteq h^{\circ} \cdot p? \cdot \top \\
\equiv & \quad \{ \text{ shunting } \} \\
 & h \cdot R \cdot (p \cdot h)? \subseteq p? \cdot \top \\
\Leftarrow & \quad \{ R \xrightarrow{h} S \} \\
 & S \cdot h \cdot (p \cdot h)? \subseteq p? \cdot \top \\
\Leftarrow & \quad \{ (p \cdot h)? \subseteq h^{\circ} \cdot p? \cdot h \text{ (82) } \} \\
 & S \cdot h \cdot h^{\circ} \cdot p? \cdot h \subseteq p? \cdot \top \\
\Leftarrow & \quad \{ \top = \top \cdot h \text{ (cancel } h\text{); } \mathrm{img}\ h \subseteq id \} \\
 & S \cdot p? \subseteq p? \cdot \top \\
\Box &
\end{array}
$$

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.
- $R^* = \bigcup_{i \geq 0} R^i$ — 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,

$$\langle \forall\ n\ :\ n \geqslant 0 :\ R^n \cdot I \subseteq S \rangle \tag{112}$$

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

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

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

$$\langle \forall\ n\ :\ n \geqslant 0 :\ R^n \cdot I \subseteq \Phi \cdot \top \rangle \tag{113}$$

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

$$\langle \exists\, n \;:\; n \geqslant 0 : \; Q \;\subseteq\; R^n \cdot I \rangle \tag{114}$$

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,

$$\langle \exists\, n \;:\; n \geqslant 0 : \; \Phi \cdot \top \;\subseteq\; R^n \cdot I \rangle \tag{115}$$

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

Ensuring safety / liveness properties

The first difficulty in ensuring properties such as (113) e (115) is the quantification on the number of path steps.

In the case of (115) 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**.

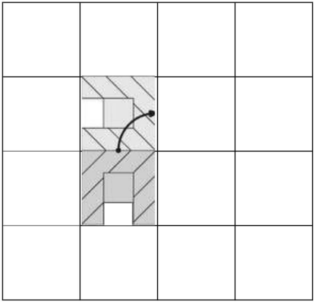
Example — Heavy armchair problem

In this problem taken from [1] 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:

$$\textit{For some } n,\ ((y, x+1), d)\ R^n\ ((y, x), d) \textit{ holds.} \qquad (116)$$

The same, in pointfree notation:

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

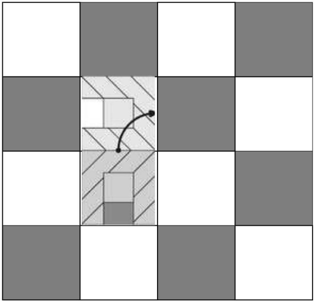
In words: *there is a path with* n *steps whose meaning is* **function** $(id \times (1+)) \times id$.

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

$$f\ (c, d) = c\ \oplus\ d \tag{117}$$

is s-**invariant** (108), that is

$$f \cdot s = f \tag{118}$$

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, (116) 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:

$$
\begin{array}{ll}
& s^n\ (c,d) = (\neg\ c, d) \\
\Rightarrow & \quad \{\ \text{Leibniz}\ \} \\
& f\ (s^n\ (c,d)) = f\ (\neg\ c, d) \\
\equiv & \quad \{\ f \text{ is } s\textbf{-invariant}\ \} \\
& f\ (c,d) = f\ (\neg\ c, d) \\
\equiv & \quad \{\ (117)\ \} \\
& c \oplus d = \neg\ c \oplus d \\
\equiv & \quad \{\ 1 \oplus d = \neg\ d \text{ and } 0 \oplus d = d\ \} \\
& d = \neg\ d \\
\equiv & \quad \{\ \text{trivia}\ \} \\
& \mathit{false}
\end{array}
$$

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

Alcuin puzzle example

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

Symmetry of the problem invites us to unify Fox with Beans [1]:

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

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

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

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

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.

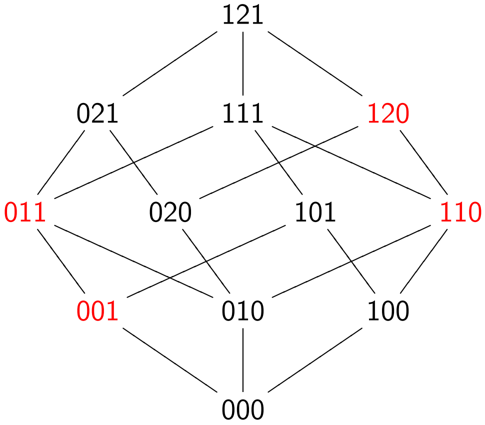
Moreover, to obtain the other bank, we use the a complement operator:

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

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

Alcuin puzzle abstraction

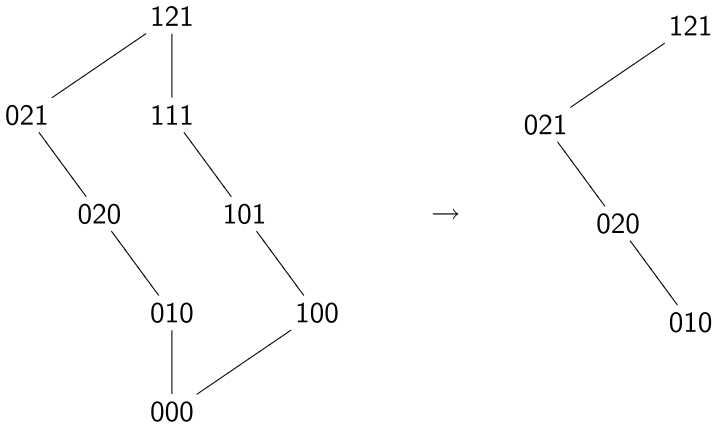
8 valid state vectors ordered by $(\leqslant)$:



The four invalid states are marked in red.

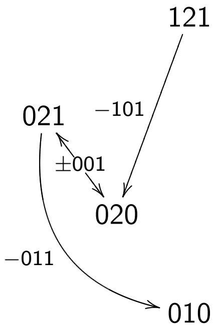
Only 4 state vectors required

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



Alcuin puzzle: abstract determinism

Abstract automaton:



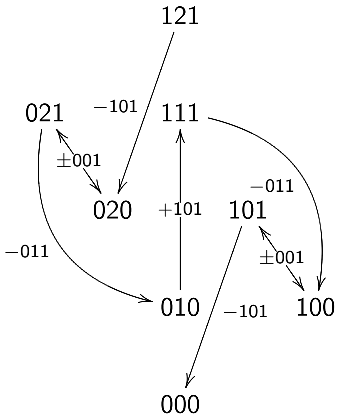
Termination is ensured by disabling toggling between states 021 and 020:

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

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

Alcuin puzzle: abstract solution

Altogether:



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

References

Background — Eindhoven quantifier calculus

Trading:

$$\langle \forall\ k\ :\ R \wedge S :\ T \rangle \ = \ \langle \forall\ k\ :\ R :\ S \Rightarrow T \rangle \tag{119}$$
$$\langle \exists\ k\ :\ R \wedge S :\ T \rangle \ = \ \langle \exists\ k\ :\ R :\ S \wedge T \rangle \tag{120}$$

de Morgan:

$$\neg\langle \forall\ k\ :\ R :\ T \rangle \ = \ \langle \exists\ k\ :\ R :\ \neg T \rangle \tag{121}$$
$$\neg\langle \exists\ k\ :\ R :\ T \rangle \ = \ \langle \forall\ k\ :\ R :\ \neg T \rangle \tag{122}$$

One-point:

$$\langle \forall\ k\ :\ k = e :\ T \rangle \ = \ T[k := e] \tag{123}$$
$$\langle \exists\ k\ :\ k = e :\ T \rangle \ = \ T[k := e] \tag{124}$$

Background — Eindhoven quantifier calculus

Nesting:

$$\langle \forall\, a, b \;:\; R \wedge S :\; T \rangle \;=\; \langle \forall\, a \;:\; R :\; \langle \forall\, b \;:\; S :\; T \rangle\rangle \tag{125}$$
$$\langle \exists\, a, b \;:\; R \wedge S :\; T \rangle \;=\; \langle \exists\, a \;:\; R :\; \langle \exists\, b \;:\; S :\; T \rangle\rangle \tag{126}$$

Rearranging-$\forall$:

$$\langle \forall\, k \;:\; R \vee S :\; T \rangle \;=\; \langle \forall\, k \;:\; R :\; T \rangle \wedge \langle \forall\, k \;:\; S :\; T \rangle \tag{127}$$
$$\langle \forall\, k \;:\; R :\; T \wedge S \rangle \;=\; \langle \forall\, k \;:\; R :\; T \rangle \wedge \langle \forall\, k \;:\; R :\; S \rangle \tag{128}$$

Rearranging-$\exists$:

$$\langle \exists\, k \;:\; R :\; T \vee S \rangle \;=\; \langle \exists\, k \;:\; R :\; T \rangle \vee \langle \exists\, k \;:\; R :\; S \rangle \tag{129}$$
$$\langle \exists\, k \;:\; R \vee S :\; T \rangle \;=\; \langle \exists\, k \;:\; R :\; T \rangle \vee \langle \exists\, k \;:\; S :\; T \rangle \tag{130}$$

Splitting:

$$\langle \forall\, j \;:\; R :\; \langle \forall\, k \;:\; S :\; T \rangle\rangle \;=\; \langle \forall\, k \;:\; \langle \exists\, j \;:\; R :\; S \rangle :\; T \rangle \tag{131}$$
$$\langle \exists\, j \;:\; R :\; \langle \exists\, k \;:\; S :\; T \rangle\rangle \;=\; \langle \exists\, k \;:\; \langle \exists\, j \;:\; R :\; S \rangle :\; T \rangle \tag{132}$$

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

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.

.