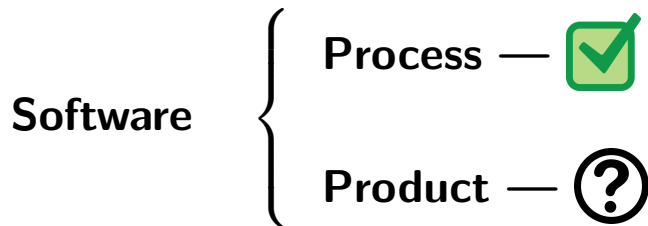


CSI — A Relational Approach to Software Design

Class 1 — Introduction

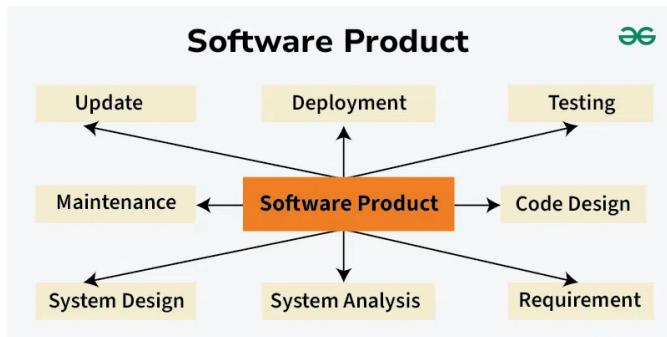
What brings us to these classes?

Software "as a problem":



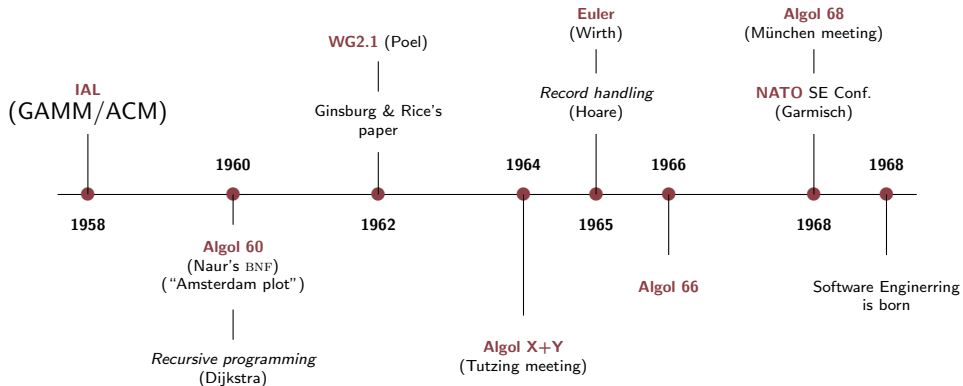
Software product?

Q.: What is the **essence** of a *software product*?

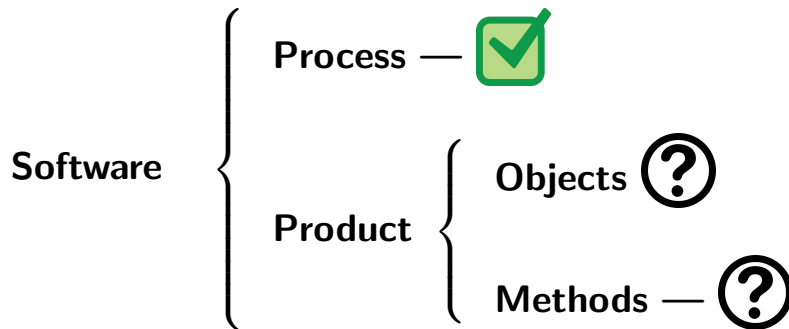


Credits: Software Engineering Tutorial (geeksforgeeks.org)

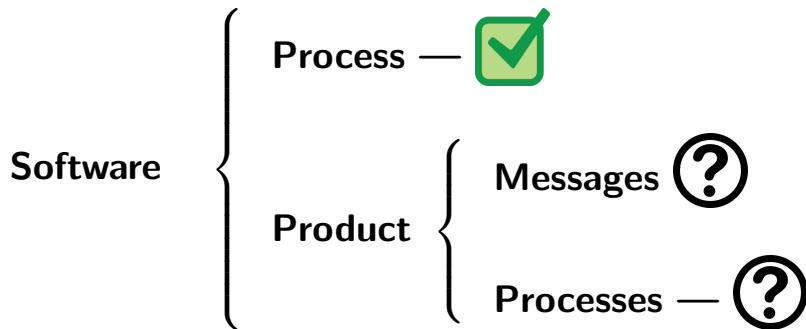
60+ years ago (1958-)



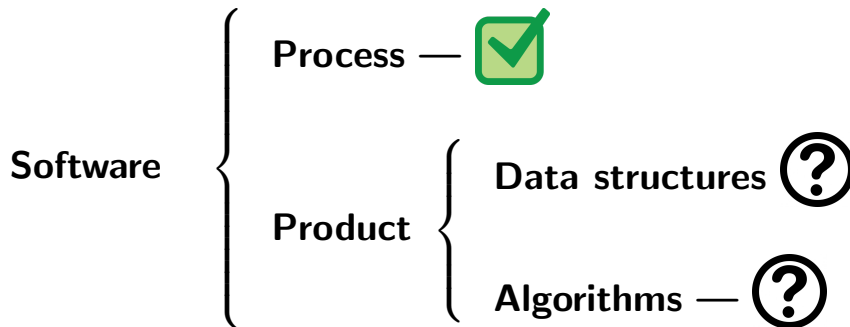
A 60 year old problem



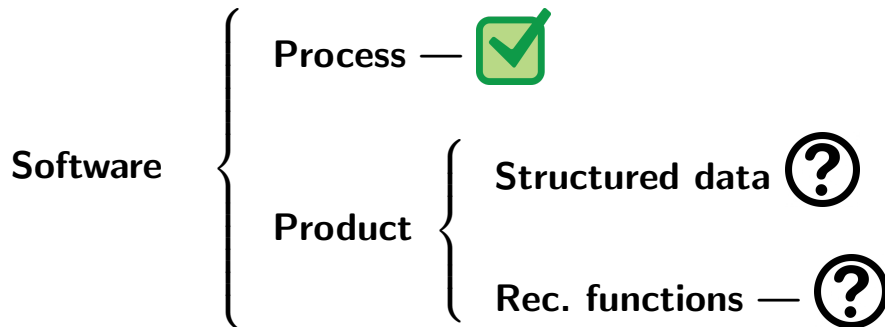
A 60 year old problem



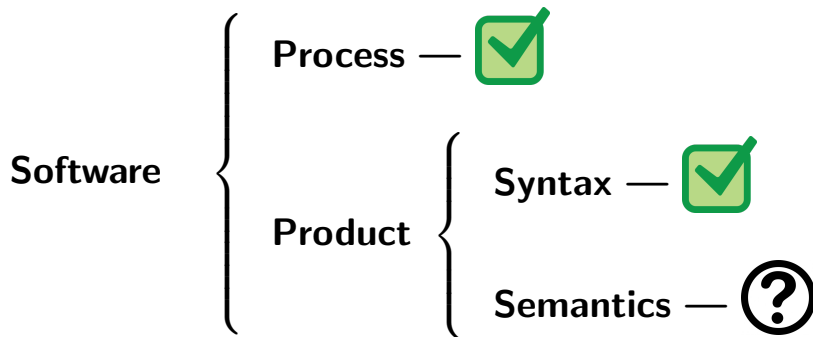
A 60 year old problem



A 60 year old problem

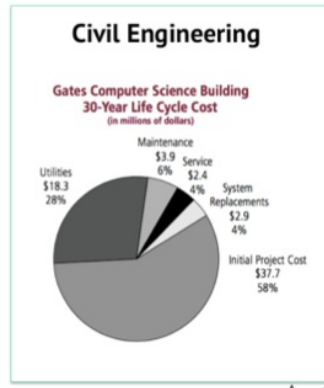


Linguistic approach



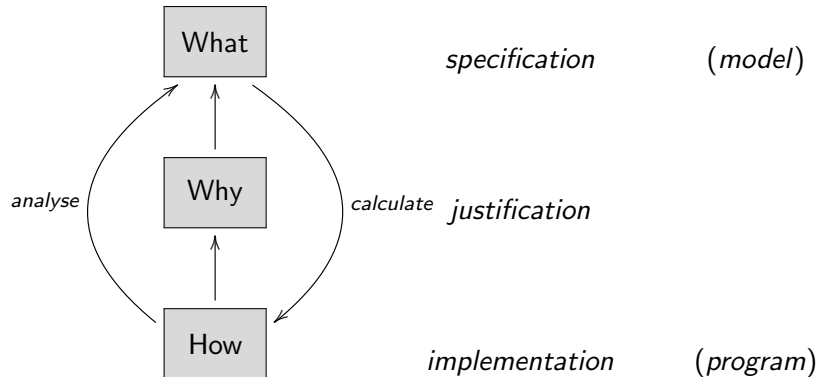
What about Software Engineering?

Global picture:

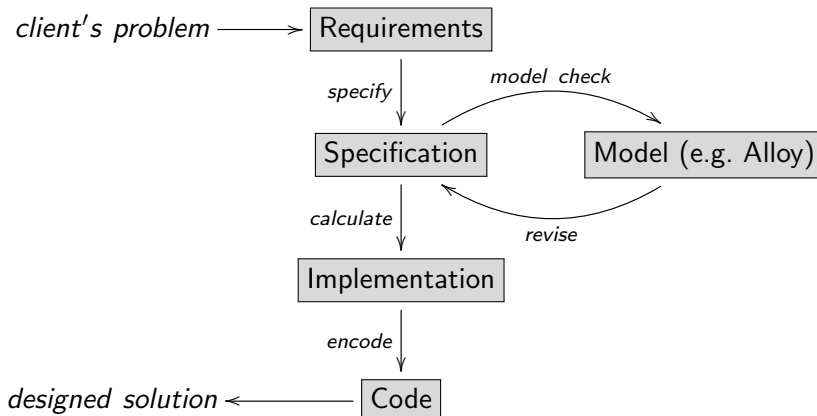


Credits: Zhenjiang Hu, NII, Tokyo JP

FM — scientific software design



FM — simplified life-cycle



Have you ever used a FM?

Of course you have! Check this:

A problem

*My three children were born at
a 3 year interval rate.*

*Altogether, they are as old as
me. I am 48. How old are they?*

A model

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

— maths description of the problem.

Some calculations

$$3x + 9 = 48$$

$$\equiv \quad \{ \text{"al-djabr" rule} \}$$

$$3x = 48 - 9$$

$$\equiv \quad \{ \text{"al-hatt" rule} \}$$

$$x = 16 - 3$$

The solution

$$x = 13$$

$$x + 3 = 16$$

$$x + 6 = 19$$

Have you ever used a FM?

Of course you have! Check this:

A problem

*My three children were born at
a 3 year interval rate.*

*Altogether, they are as old as
me. I am 48. How old are they?*

A model

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

— maths description of the problem.

Some calculations

$$3x + 9 = 48$$

$$\equiv \quad \{ \text{"al-djabr" rule} \}$$

$$3x = 48 - 9$$

$$\equiv \quad \{ \text{"al-hatt" rule} \}$$

$$x = 16 - 3$$

The solution

$$x = 13$$

$$x + 3 = 16$$

$$x + 6 = 19$$

Have you ever used a FM?

Of course you have! Check this:

A problem

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

A model

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

— maths description of the problem.

Some calculations

$$3x + 9 = 48$$

$$\equiv \quad \{ \text{"al-djabr" rule} \}$$

$$3x = 48 - 9$$

$$\equiv \quad \{ \text{"al-hatt" rule} \}$$

$$x = 16 - 3$$

The solution

$$x = 13$$

$$x + 3 = 16$$

$$x + 6 = 19$$

Have you ever used a FM?

Of course you have! Check this:

A problem

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

A model

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

— maths description of the problem.

Some calculations

$$3x + 9 = 48$$

$$\equiv \quad \{ \text{"al-djabr" rule} \}$$

$$3x = 48 - 9$$

$$\equiv \quad \{ \text{"al-hatt" rule} \}$$

$$x = 16 - 3$$

The solution

$$x = 13$$

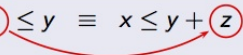
$$x + 3 = 16$$

$$x + 6 = 19$$

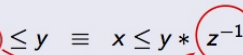
Have you ever used a FM?

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

al-djabr

$$x - \textcircled{z} \leq y \equiv x \leq y + \textcircled{z}$$


al-hatt

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


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

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

Software problems

Now, suppose the **problem** was

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

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

Yes, of course there is — see aside:

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

where

sorted = ... marks ...

bag = ...

But,

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

Yes — Wait and see :-)

Software problems

Now, suppose the **problem** was

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

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

Yes, of course there is — see aside:

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

where

sorted = ... marks ...

bag = ...

But,

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

Yes — Wait and see :-)

Software problems

Now, suppose the **problem** was

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

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

Yes, of course there is — see aside:

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

where

sorted = ... marks ...

bag = ...

But,

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

Yes — Wait and see :-)

Software problems

Now, suppose the **problem** was

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

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

Yes, of course there is — see aside:

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

where

sorted = ... marks ...

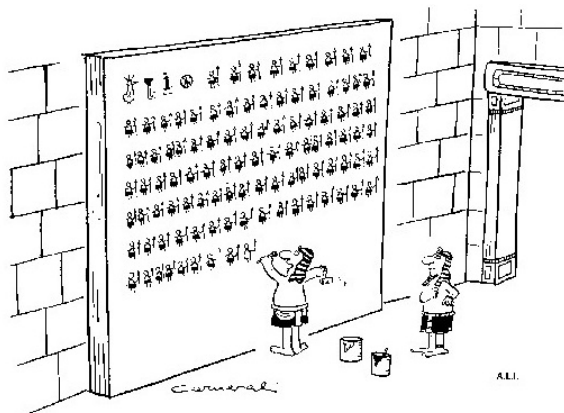
bag = ...

But,

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

Yes — Wait and see :-)

Notation matters!



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

Well-known FM notations / tools / resources

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

Notations:

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

Tools:

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

Resources:

- Formal Methods Europe
- Formal Methods wiki (Oxford)

The age of AI

Dramatic changes in SE.

Writing code no longer a valuable skill.

Verifying code is the "new" valuable skill!



The age of AI

"We should not rule out any technology enabling us to produce good software, be it through AI, mathematics or both."

"One thing can be taken for granted: who in the future will fly in a plane whose LLM-generated software has not been verified 100% correct?" [3]



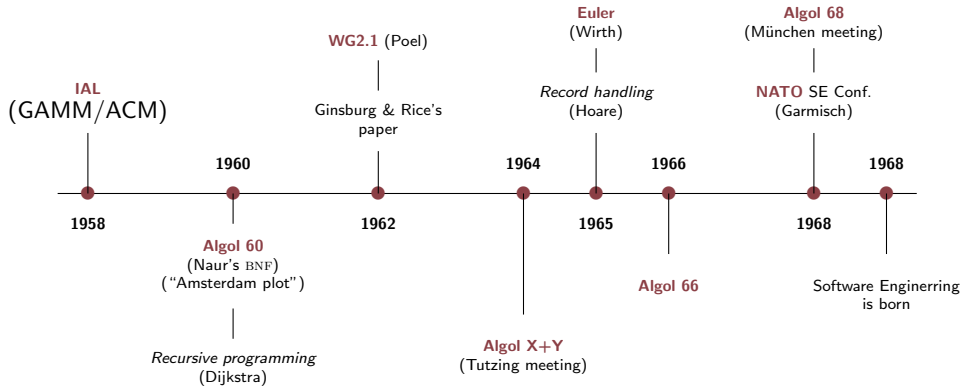
The age of AI

"On the day direct generation of programs from requirements proves to be definitively effective, the need for (formal) verification will remain."

"And this is where jobs in the future of computing are likely to be found."
[3]



60+ years ago (1958-)



Hoare Logic — “turning point” (1968)

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

Summary.

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

(ADB/IFIP/1164;1456)

Inv/pre/post

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

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

Inv/pre/post

Error handling...

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

Inv/pre/post

Pre-conditions?

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

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

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

Inv/pre/post

pre...? choice...?

- Non-determinism
- Parallelism
- Abstraction

Inv/pre/post

pre...? choice...?

- Non-determinism
- Parallelism
- Abstraction

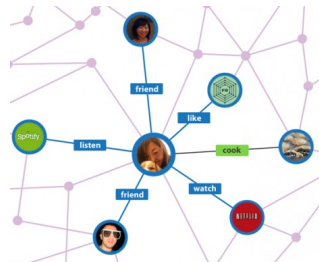
Functions not enough!

Solution?

Relations (*which extend functions*)



Is “everything” a relation?



How to “dematerialize” them?

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

Software is too **diverse** — many approaches, lack of unity

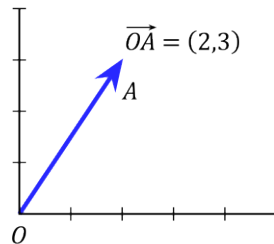
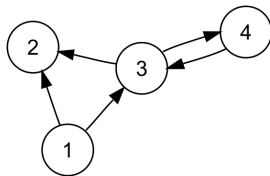
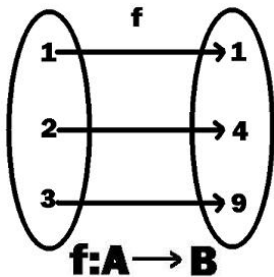
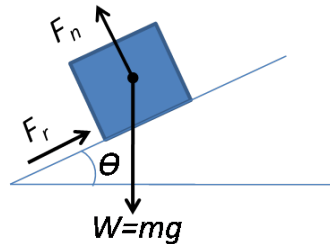
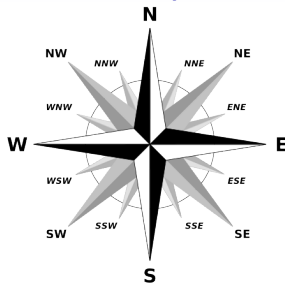
Software is too **wide** — from assembly to quantum programming

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

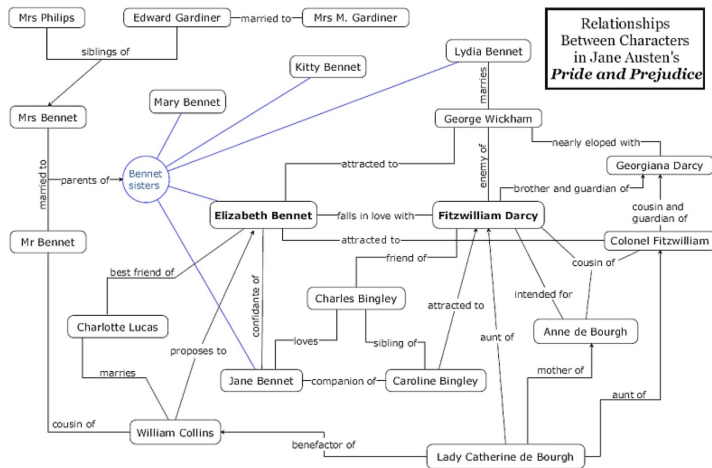
Put in another way:

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

Check the pictures...

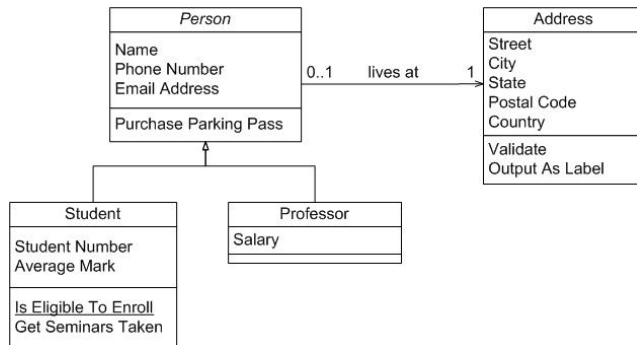
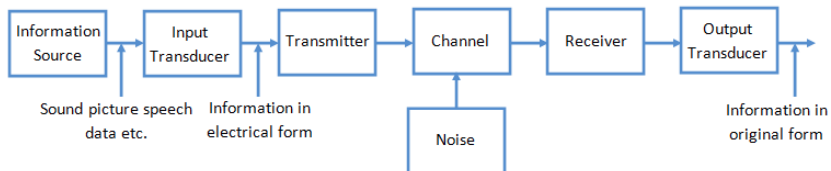


Check the pictures



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

Check the pictures



Check the pictures

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



Arrows everywhere

Arrows! A (graphical) device **common** to describing (many) **different** fields of human activity.

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

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

Relation algebra

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

Question:

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

Relation algebra

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

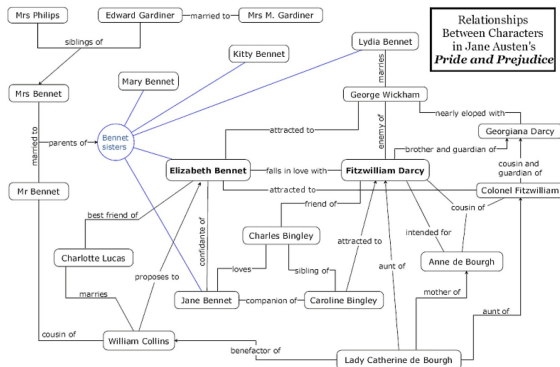
- Augustus de Morgan (1806-71) — recall *de Morgan* laws — proposed a **Logic of Relations** as early as 1860.
- Predicate logic appeared later.



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 Austen, 1813.)

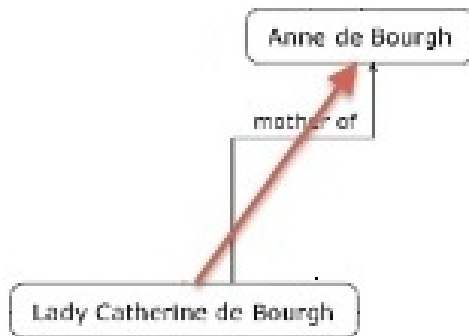
Arrow notation for relations

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

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

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

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

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

respectively.

Sets of pairs

So, one might have written instead:

$$(0, \pi) \in (\leq)$$

$$(\text{Catherine}, \text{Anne}) \in \text{isMotherOf}$$

$$(3, 2) \in (1+)$$

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

- They can be regarded as **sets of pairs**
- Better: they should be regarded as **binary relations**.

Therefore,

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

Binary Relations

Binary relations are typed:

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

A, B are types.

Writing

$$B \xleftarrow{R} A$$

means the same as

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

Notation

Infix notation

The usual infix notation used in natural language — eg.

Catherine isMotherOf Anne — and in maths — eg. $0 \leq \pi$ — extends to

arbitrary $B \xleftarrow{R} A$: we write

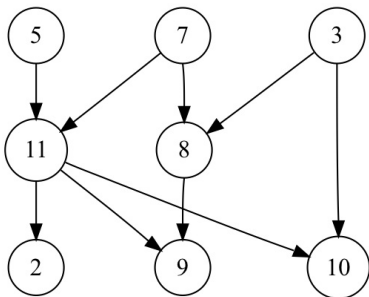
$b R a$

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

Binary relations are matrices

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

Relation R :



Matrix M :

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

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

Class 2 - Going Relational

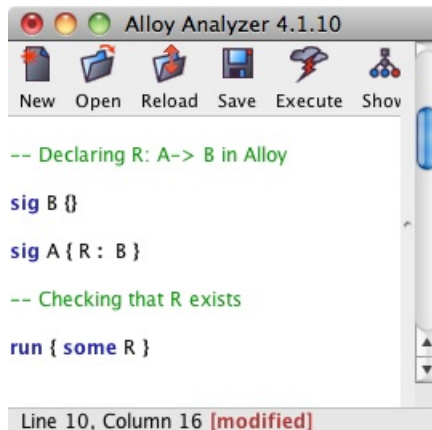
Alloy: where “everything is a relation”

Declaring binary relation

$A \xrightarrow{R} B$ in **Alloy**
(aside).

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

We shall be using **Alloy**
[1] 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,

$$b\ f\ a \text{ literally means } b = f\ a \quad (1)$$

Therefore, we generalize

$B \xleftarrow{f} A$ $b = f\ a$	to	$B \xleftarrow{R} A$ $b\ R\ a$
---------------------------------	----	--------------------------------

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:

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

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

Relations:

$$\begin{array}{ccc} \textit{Being} & \xrightarrow{\textit{Eats}} & \textit{Being} \\ & \downarrow \textit{where} & \\ & \textit{Bank} & \xrightarrow{\textit{cross}} \textit{Bank} \end{array} \quad (4)$$

Exercise

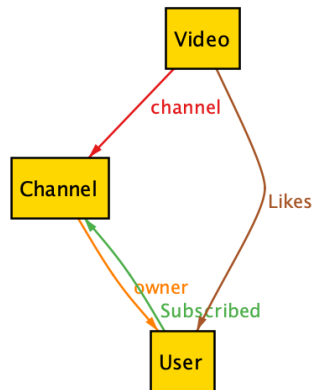
Identify the **types** and **relations** needed to model a (much simplified!) YouTube-like social network, knowing that:

- a) YouTube channels exhibit videos that YouTube users may choose to like;
- b) a YouTube user may own one channel only but can subscribe to multiple channels.



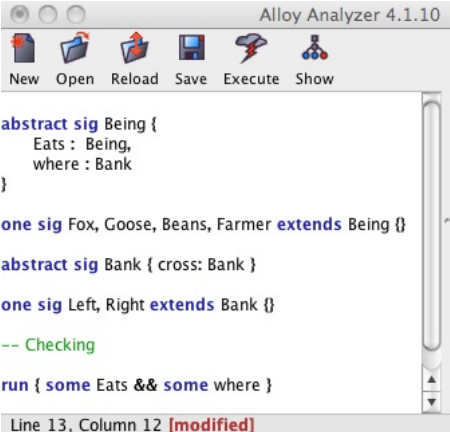
Exercise

- a) YouTube channels exhibit videos that YouTube users may choose to like;
- b) a YouTube user may own one channel only but can subscribe to multiple channels.



PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Specification source written in Alloy:



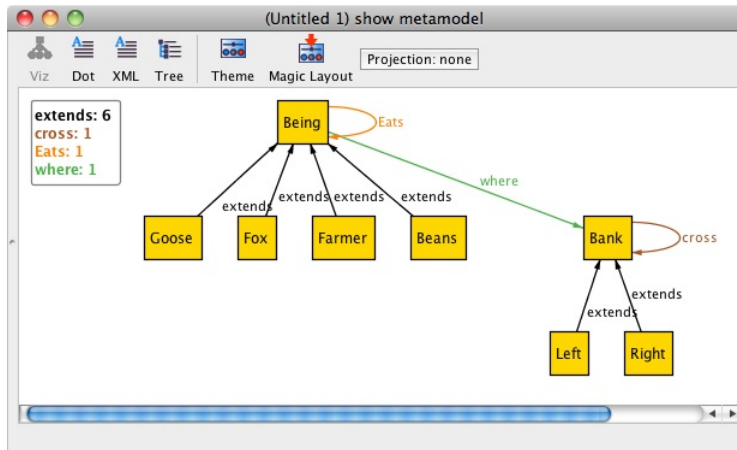
The screenshot shows the Alloy Analyzer 4.1.10 window. The title bar reads "Alloy Analyzer 4.1.10". Below the title bar is a toolbar with icons for New, Open, Reload, Save, Execute, and Show. The main text area contains the following Alloy specification:

```
abstract sig Being {  
  Eats : Being,  
  where : Bank  
}  
  
one sig Fox, Goose, Beans, Farmer extends Being {}  
  
abstract sig Bank { cross: Bank }  
  
one sig Left, Right extends Bank {}  
  
-- Checking  
  
run { some Eats && some where }
```

At the bottom of the window, a status bar indicates "Line 13, Column 12 [modified]".

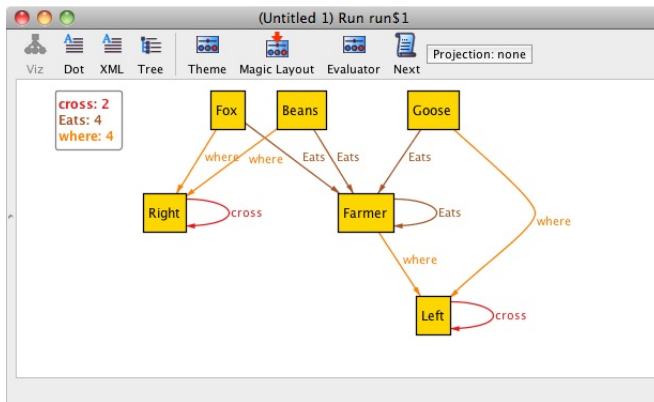
PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Diagram of specification (model) given by Alloy:



PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Diagram of instance of the model given by Alloy:



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

PROPOSITIO DE HOMINE ET CAPRA ET LVPO

- How do we “fine tune” it?
- We need to be able to think in terms of **relations**

“Relational thinking”

- Is there a “**language**” for relations ?

Yes! And a formal one



PROPOSITIO DE HOMINE ET CAPRA ET LVPO

- How do we “fine tune” it?
- We need to be able to think in terms of **relations**

“Relational thinking”

- Is there a “**language**” for relations ?

Yes! And a formal one



PROPOSITIO DE HOMINE ET CAPRA ET LVPO

- How do we “fine tune” it?
- We need to be able to think in terms of **relations**

“Relational thinking”

- Is there a “**language**” for relations ?

Yes! And a formal one



PROPOSITIO DE HOMINE ET CAPRA ET LVPO

- How do we “fine tune” it?
- We need to be able to think in terms of **relations**

“Relational thinking”

- Is there a “**language**” for relations ?

Yes! And a formal one 😊

Composition

Recall **function composition**
(aside).

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

$$\begin{array}{ccccc} B & \xleftarrow{f} & A & \xleftarrow{g} & C \\ & \searrow & & \swarrow & \\ & f \cdot g & & & \end{array} \quad (5)$$
$$b = f(g \ c)$$

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

Composition

That is:



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

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

$$u \text{ Uncle } n \equiv \langle \exists p :: u \text{ Brother } p \wedge p \text{ Parent } n \rangle$$

Note how rule (6) *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

$$\begin{aligned}
 b(f \cdot g)c &\equiv \langle \exists a :: b f a \wedge a g c \rangle \\
 &\equiv \{ a g c \text{ means } a = g c (1) \} \\
 &\quad \langle \exists a :: a = g c \wedge b f a \rangle \\
 &\equiv \{ \exists\text{-trading (78)} ; b f a \text{ means } b = f a (1) \} \\
 &\quad \langle \exists a : a = g c : b = f a \rangle \\
 &\equiv \{ \exists\text{-one point rule (82)} \} \\
 &\quad b = f(g c)
 \end{aligned}$$

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

¹Check the appendix on predicate calculus.

Relation inclusion

Relation inclusion generalizes function equality:

Equality *on functions*

$$f = g \equiv \langle \forall a :: f\ a = g\ a \rangle \quad (7)$$

generalizes to **inclusion** on relations:

$$R \subseteq S \equiv \langle \forall b, a : b\ R\ a : b\ S\ a \rangle \quad (8)$$

(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, meaning it is

reflexive,

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

transitive

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

and **antisymmetric:**

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

Therefore:

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

Special relations

Every type $B \longleftarrow A$ has its

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

Every type $A \longleftarrow A$ has the

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

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

Clearly, for every R ,

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

Relational equality

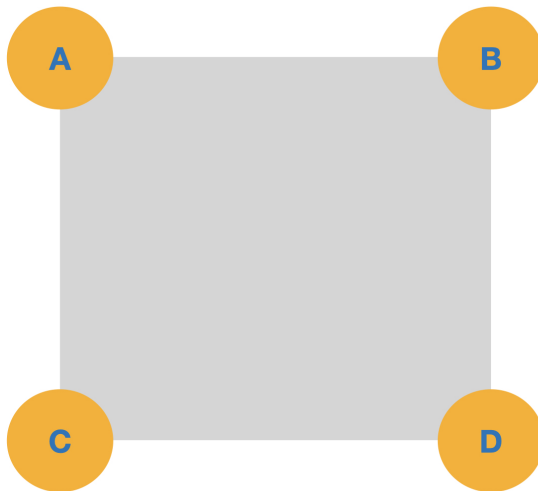
Both (12) and (11) establish **relation equality**, resp. in PW/PF fashion.

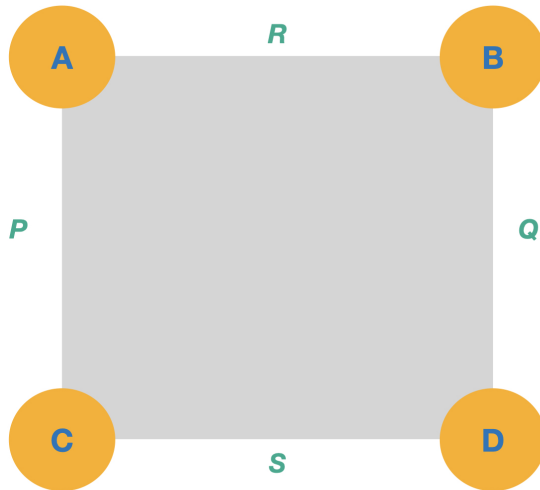
Rule (11) is also called “ping-pong” or **cyclic inclusion**, often taking the format:

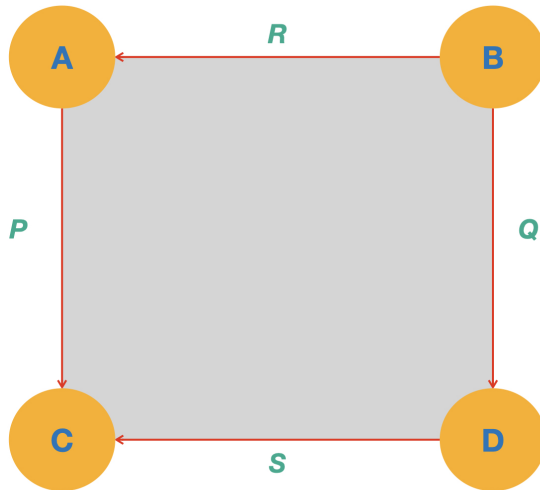
$$\begin{array}{l}
 R \\
 \subseteq \quad \{ \text{....} \} \\
 S \\
 \subseteq \quad \{ \text{....} \} \\
 R \\
 \vdots \quad \{ \text{“ping-pong” (11)} \} \\
 R = S
 \end{array}$$

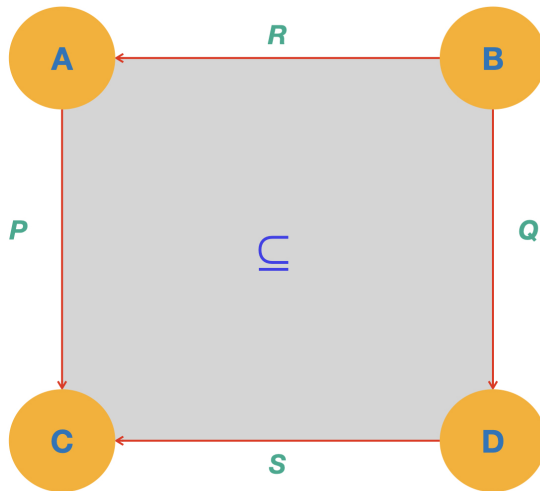
Squares



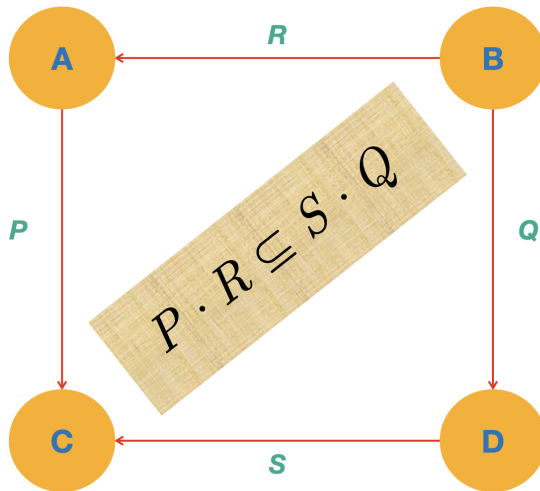






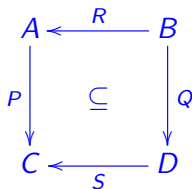


"Magic" square



“Magic” square

Four binary relations:



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

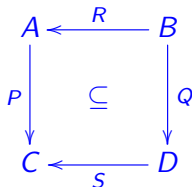
(15)

Terminology:

- R — *pre-producer*
- P — *pre-consumer*
- Q — *post-producer*
- S — *post-consumer*

“Magic” square

Pointfree:



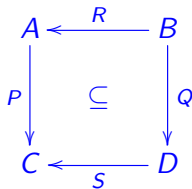
$$P \cdot R \subseteq S \cdot Q$$

Pointwise:

$$\langle \forall c, b :: \langle \exists a :: c P a \wedge a R b \rangle \Rightarrow \langle \exists d :: c S d \wedge d Q b \rangle \rangle \quad (16)$$

"Magic" square

Pointfree:



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

Pointwise:

$$\begin{array}{c} \exists \\ \quad a \quad d \\ \quad \vdots \quad \vdots \\ P \cdot R \Rightarrow S \cdot Q \\ \quad \vdots \quad \vdots \\ \forall \quad c \quad b \quad c \quad b \end{array}$$

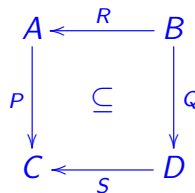
"Magic" square



1	14	14	4
11	7	6	9
8	10	10	5
13	2	3	15

Perhaps not what you were expecting...

... but we can do **a lot** with

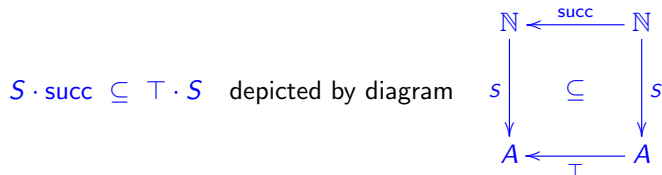


Exercises

Thus, “magic squares” are **diagrams** capturing **assertions** of the form $X \subseteq Y$ where X and Y are relation compositions.

Let us have a feeling of what can be represented graphically by such square-shaped diagrams.

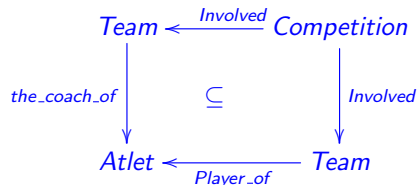
Exercise 1: Let $a S n$ mean: “student a is assigned number n ”. Using (6) and (8), check that assertion



(where $\text{succ } n = n + 1$) means that numbers are assigned to students sequentially. $\square$

Exercises

Exercise 2: Consider sports **competitions** involving **teams** which have **atlets** (players) and **coaches**. Follow rule (16) to spell out the logical meaning of the following *magic square*:



Then express this meaning in natural language, avoiding reading completely through the logic assertion obtained via (16). $\square$

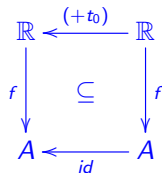
Exercises

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

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

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

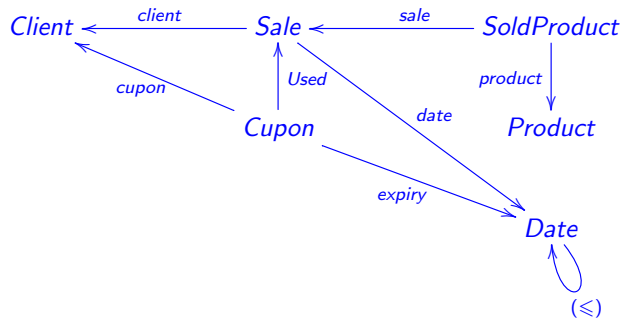
Exercise 4: Let t_0 be a real number. Show that the “magic square”



tells that f is a periodic function (on $\mathbb{R}$) with period t_0 . $\square$

Exercises

Exercise 5: ("D. Acácia grocery")

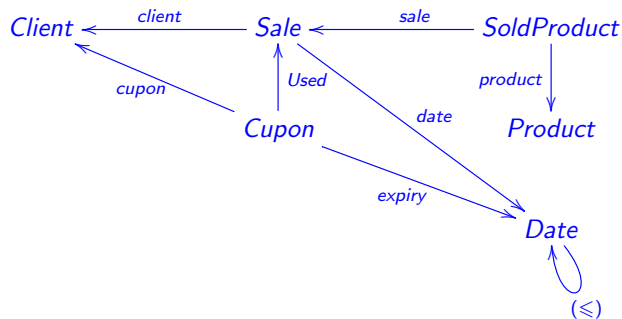


Find "magic square" for property:

Coupons cannot be used beyond their expiry date.

Exercises

Exercise 6: ("D. Acácia grocery")



Find "magic square" for property:

Coupons can only be used by clients who own them.

Exercises

Exercise 7: Use (6) and (8) and predicate calculus to show that

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

$$R \cdot \perp = \perp = \perp \cdot R \tag{18}$$

hold and that composition is associative:

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

□

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

Class 3 - Basic relation algebra

Converses

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

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

Note that converse commutes with composition

$$(R \cdot S)^\circ = S^\circ \cdot R^\circ \quad (21)$$

and with itself:

$$(R^\circ)^\circ = R \quad (22)$$

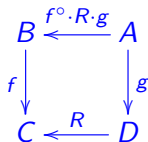
Converse captures the **passive voice**: *Catherine eats the apple* — $R = (\text{eats})$ — is the same as *the apple is eaten by Catherine* — $R^\circ = (\text{is eaten by})$.

Function converses

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

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

cf. diagram:



Therefore (tell why):

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

Let us see an example of using these rules.

PF-transform at work

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

f is **injective**

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

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

$\equiv \{ (24) \text{ for } f = g \}$

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

$\equiv \{ (23) \text{ for } R = f = g = id \}$

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

$\equiv \{ \text{go pointfree (8) i.e. drop } y, x \}$

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

The other way round

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

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

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

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

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

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

$$\equiv \{ \text{\(\forall\)-one point (81) ; converse (20)} \}$$

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

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

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

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

f is **surjective**

Why *id* (really) matters

Terminology:

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

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

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

Alloy: checking for coreflexive relations

The image shows two windows from the Alloy Analyzer 4.1.10. The background window displays the Alloy code:

```
sig A { R : A }  
-- Checking that coreflexive R exist  
run { R in iden } for 3 A
```

The status bar indicates "Line 4, Column 37 [modified]". The foreground window, titled "(Untitled 2) Run run\$1 for 3 A", shows the visualization of the model. It features a toolbar with options: Viz, Dot, XML, Tree, Theme, Magic Layout, Evaluator, and Next. The visualization displays a set R with a cardinality of 3, represented by a box labeled "R: 3". To the right, three yellow boxes labeled $A0$, $A1$, and $A2$ are shown. Each box has a red curved arrow labeled R pointing from the box to itself, indicating that each element in A is related to itself, which is the coreflexive property.

Kernels of functions

Meaning of $\ker f$:

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

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

Exercise 8: Let K be a nonempty data domain, $k \in K$ and $\underline{k}$ be the "everywhere k " function:

$$\begin{aligned}
 \underline{k} & : A \rightarrow K \\
 \underline{k} a & = k
 \end{aligned} \tag{25}$$

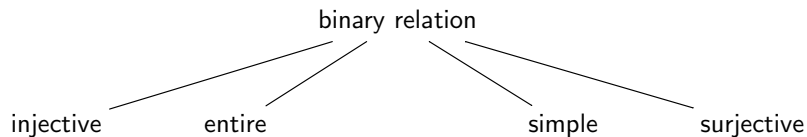
Compute which relations are defined by the following expressions:

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

□

Binary relation taxonomy

Topmost criteria:



Definitions:

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

(27)

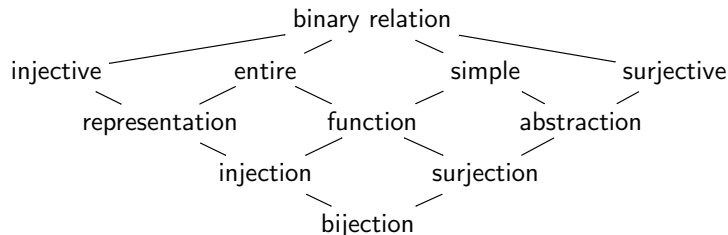
Facts:

$$\ker (R^\circ) = \text{img } R \quad (28)$$

$$\text{img } (R^\circ) = \ker R \quad (29)$$

Binary relation taxonomy

The whole picture:



(30)

Exercise 9: 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

$A \text{ lone} \rightarrow B$	$A \rightarrow \text{some } B$	$A \rightarrow \text{lone } B$	$A \text{ some} \rightarrow B$
injective	entire	simple	surjective

$A \text{ lone} \rightarrow \text{some } B$	$A \rightarrow \text{one } B$	$A \text{ some} \rightarrow \text{lone } B$
representation	function	abstraction

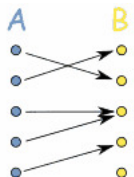
$A \text{ lone} \rightarrow \text{one } B$	$A \text{ some} \rightarrow \text{one } B$
injection	surjection

$A \text{ one} \rightarrow \text{one } B$
bijection

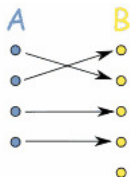
(Courtesy of Alcino Cunha.)

Exercises

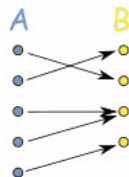
Exercise 10: Label the items (uniquely) in these drawings²



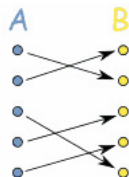
General
Function



Injective
Not surjective



Surjective
Not injective



Bijjective
(injective and
surjective)

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

²Credits: <http://www.matematikaria.com/unit/injective-surjective-bijective.html>.

Exercises

Exercise 11: Prove the following fact

A function f is a bijection iff its converse f° is a function (31)

by completing:

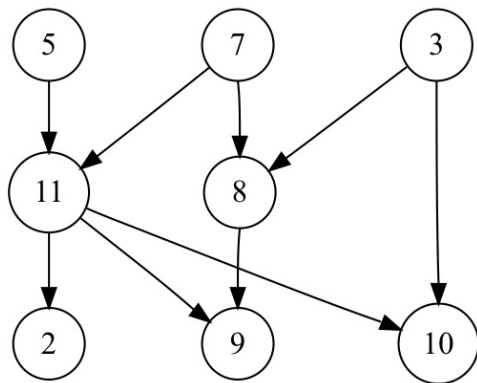
$$\begin{aligned}
 & f \text{ and } f^\circ \text{ are functions} \\
 \equiv & \quad \{ \dots \} \\
 & (id \subseteq \ker f \wedge \text{img } f \subseteq id) \wedge (id \subseteq \ker (f^\circ) \wedge \text{img } (f^\circ) \subseteq id) \\
 \equiv & \quad \{ \dots \} \\
 & \vdots \\
 \equiv & \quad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

□

Taxonomy using matrices

Recall that binary relations can be regarded as Boolean **matrices**, eg.

Relation R :



Matrix M :

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

Taxonomy using matrices

- **entire** — at least one 1 in every column (32)
- **surjective** — at least one 1 in every row (33)
- **simple** — at most one 1 in every column (34)
- **injective** — at most one 1 in every row (35)
- **bijective** — exactly one 1 in every column and every row. (36)

PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Exercise 12: Let relation $Bank \xrightarrow{cross} Bank$ (4) be defined by:

$Left \ cross \ Right$
 $Right \ cross \ Left$

It therefore is a bijection. Why? $\square$

Exercise 13: Check which of the following properties,

$simple$, $entire$,
 $injective$, $surjective$,
 $reflexive$, $coreflexive$

<i>Eats</i>	<i>Fox</i>	<i>Goose</i>	<i>Beans</i>	<i>Farmer</i>
<i>Fox</i>	0	1	0	0
<i>Goose</i>	0	0	1	0
<i>Beans</i>	0	0	0	0
<i>Farmer</i>	0	0	0	0

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

PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Exercise 14: Relation $\textit{where} : \textit{Being} \rightarrow \textit{Bank}$ should obey the following constraints:

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

Express such constraints in relational terms. Conclude that $\textit{where}$ should be a **function**. $\square$

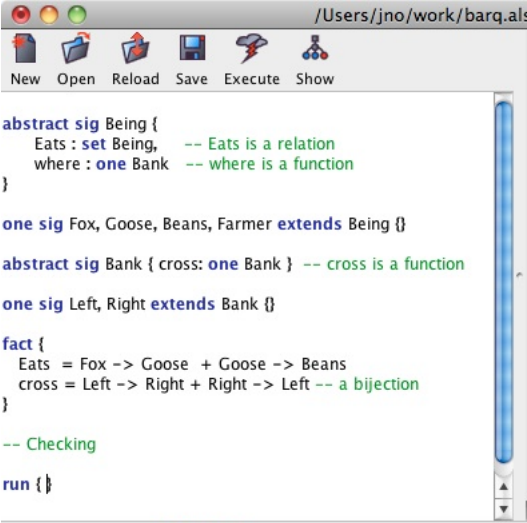
Exercise 15: There are only two **constant** functions (25) in the type $\textit{Being} \longrightarrow \textit{Bank}$ of $\textit{where}$. Identify them and explain their role in the puzzle. $\square$

Exercise 16: 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.



The screenshot shows a text editor window titled "/Users/jno/work/barq.als". The window has a toolbar with icons for New, Open, Reload, Save, Execute, and Show. The text inside the editor is as follows:

```

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

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

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

one sig Left, Right extends Bank {}

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

-- Checking

run {}
  
```

At the bottom of the window, a status bar indicates "Line 20, Column 7 [modified]".

Functions in one slide

As seen before, a **function** f is a binary relation such that

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

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

Functions, relationally

The following properties of any function f are **extremely** useful:

Shunting rules:

$$f \cdot R \subseteq S \equiv R \subseteq f^\circ \cdot S \quad (37)$$

$$R \cdot f^\circ \subseteq S \equiv R \subseteq S \cdot f \quad (38)$$

Equality rule:

$$f \subseteq g \equiv f = g \equiv f \supseteq g \quad (39)$$

Rule (39) follows from (37,38) by “cyclic inclusion” (next slide).

Proof of functional equality rule (39)

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

Then:

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

Exercises

Exercise 17: Identify which properties of a function f are captured by the following “magic squares”:

$$\begin{array}{ccccc}
 A & \xleftarrow{id} & A & \xleftarrow{f^\circ} & B \\
 id \downarrow & \subseteq & \downarrow f & \subseteq & \downarrow id \\
 A & \xleftarrow{f^\circ} & B & \xleftarrow{id} & B
 \end{array} \tag{40}$$

$$\begin{array}{ccc}
 \begin{array}{ccc}
 A & \xleftarrow{R} & C \\
 f \downarrow & \subseteq & \downarrow Q \\
 B & \xleftarrow{id} & B
 \end{array} & \equiv & \begin{array}{ccc}
 A & \xleftarrow{R} & C \\
 id \downarrow & \subseteq & \downarrow Q \\
 A & \xleftarrow{f^\circ} & B
 \end{array}
 \end{array} \tag{41}$$

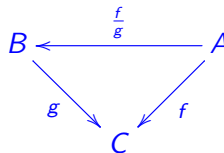


Dividing functions

Definition:

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

cf.



(42)

By (23) we have:

$$b \frac{f}{g} a \equiv g b = f a$$

(43)

How useful is this?

Dividing functions

Think of the following sentence:

Mary lives where John was born.

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

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

In general,

$b \frac{f}{g} a$ means “the g of b is the f of a ”.

Dividing functions

Exercise 18: Check the properties:

$$\frac{f}{id} = f \quad (44)$$

$$\frac{f \cdot h}{g \cdot k} = k^\circ \cdot \frac{f}{g} \cdot h \quad (45)$$

$$\frac{f}{f} = \ker f \quad (46)$$

$$\left(\frac{f}{g}\right)^\circ = \frac{g}{f} \quad (47)$$

□

Exercise 19: (a) Show that

$$\frac{f}{g} \cdot \frac{h}{f} \subseteq \frac{h}{g} \quad (48)$$

holds.

(b) Which property of f strengthens (48) to an equality? □

Class 4 – Meet and Join

'Meet' and 'Join'

For $R, S : A \rightarrow B$ (i.e. they have the same type), relational **meet** (cf. intersection) and **join** (cf. union) are defined as follows:

$$b (R \cap S) a \equiv b R a \wedge b S a \quad (49)$$

$$b (R \cup S) a \equiv b R a \vee b S a \quad (50)$$

Clearly, they lift pointwise **conjunction** and **disjunction**, respectively, to the pointfree level.

Endo-relations

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

endo-relation.

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

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

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

Taxonomy of endo-relations

Besides

reflexive: iff $id \subseteq R$ (51)

coreflexive: iff $R \subseteq id$ (52)

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

transitive: iff $R \cdot R \subseteq R$ (53)

symmetric: iff $R \subseteq R^\circ (\equiv R = R^\circ)$ (54)

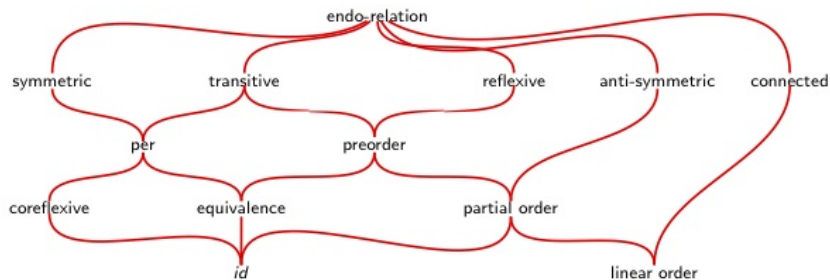
anti-symmetric: iff $R \cap R^\circ \subseteq id$ (55)

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

connected: iff $R \cup R^\circ = \top$ (56)

Taxonomy of endo-relations

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



Taxonomy of endo-relations

In summary:

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

³Kernels of functions are always equivalence relations, see exercise 28.

Exercises

Exercise 20: Infer $id \subseteq \ker f$ (f is total) and $\text{img } f \subseteq id$ (f is simple) from the shunting rules (37) or (38). $\square$

Exercise 21: Consider the relation

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

Is this relation: reflexive? irreflexive? transitive? anti-symmetric? symmetric? connected? $\square$

Exercise 22: Check which of the following properties,

transitive, symmetric, anti-symmetric, connected

hold for the relation $Eats$ of exercise 13. $\square$

Exercises

Exercise 23: Let $\mathbb{R} \xleftarrow{R} \mathbb{R}$ be the binary relation that defines the unit circumference,

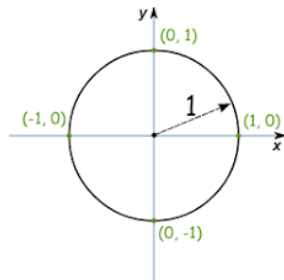
$$y R x \stackrel{\text{def}}{=} y^2 + x^2 = 1 \quad (57)$$

that is,

$$R = \frac{(1-)\cdot sq}{sq} \quad (58)$$

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

Without using (57), show that R is **symmetric**. $\square$



Exercises

Exercise 24: A relation R is said to be **co-transitive** or **dense** iff the following holds:

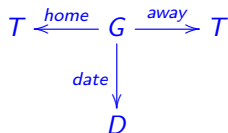
$$\langle \forall b, a : b R a : \langle \exists c : b R c : c R a \rangle \rangle \quad (59)$$

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

Exercise 25: Expand criteria (53) to (56) to pointwise notation. $\square$

Exercises

Exercise 26: The teams (T) of a football league play games (G) at home or away, and every game takes place in some date:



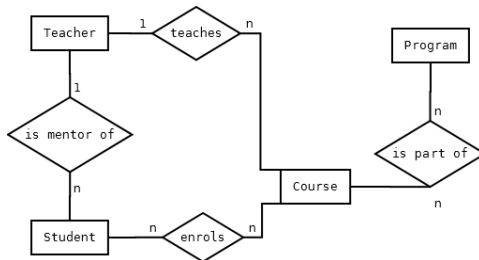
Moreover, (a) No team can play two games on the same date; (b) All teams play against each other but not against themselves; (c) For each home game there is another game away involving the same two teams. Show that

$$id \subseteq \frac{\text{away}}{\text{home}} \cdot \frac{\text{away}}{\text{home}} \quad (60)$$

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

Formalizing ER diagrams

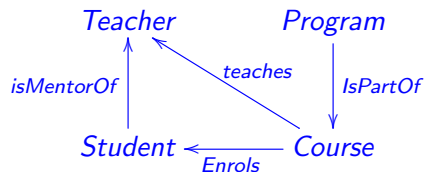
So-called “**Entity-Relationship**” (ER) diagrams are commonly used to capture relational information, e.g.⁴



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

⁴Credits: <https://dba.stackexchange.com/questions>.

Exercise



(61)

Exercise 27: Looking at diagram (61),

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

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

inadequate as answer to the previous question?



Meet and join (pointfree)

Recall **meet** (intersection) and **join** (union), introduced by (49) and (50), respectively.

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

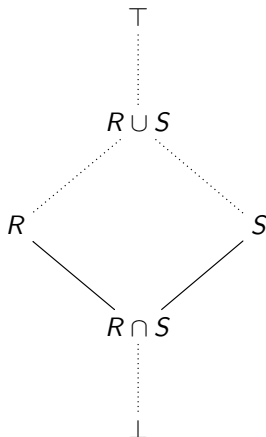
$$X \subseteq R \cap S \equiv X \subseteq R \wedge X \subseteq S \quad (62)$$

$$R \cup S \subseteq X \equiv R \subseteq X \wedge S \subseteq X \quad (63)$$

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

In summary

Type $B \leftarrow A$ forms a lattice:



"top"

join, lub ("least upper bound")

meet, glb ("greatest lower bound")

"bottom"

How universal properties help

Taking (62) as example:

$$X \subseteq R \cap S \equiv \begin{cases} X \subseteq R \\ X \subseteq S \end{cases}$$

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

$$\begin{cases} R \cap S \subseteq R \\ R \cap S \subseteq S \end{cases} \quad (64)$$

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

$$R = R \cap S \equiv R \subseteq S \quad (65)$$

PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Back to our running example, we specify:

Being at the same bank:

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

Risk of somebody eating somebody else:

$$\text{CanEat} = \text{SameBank} \cap \text{Eats}$$

Then

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

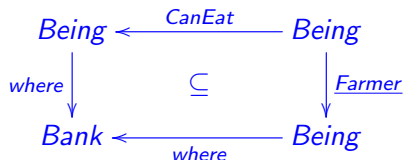
$$\text{CanEat} \subseteq \text{SameBank} \cdot \underline{\text{Farmer}} \quad (66)$$

PROPOSITIO DE HOMINE ET CAPRA ET LVPO

By (37), “starvation” property (66) converts to:

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

In this version, (66) can be depicted as a ‘magic square’:



(67)

This “reads” in a nice way:

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

PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Properties which — such as (67) — are desirable and must **always hold** are called **invariants**.

See aside the ‘starvation’ invariant (67) written in **Alloy**.

```

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

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

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

one sig Left, Right extends Bank {}

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

-- Finding instances satisfying the invariant

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

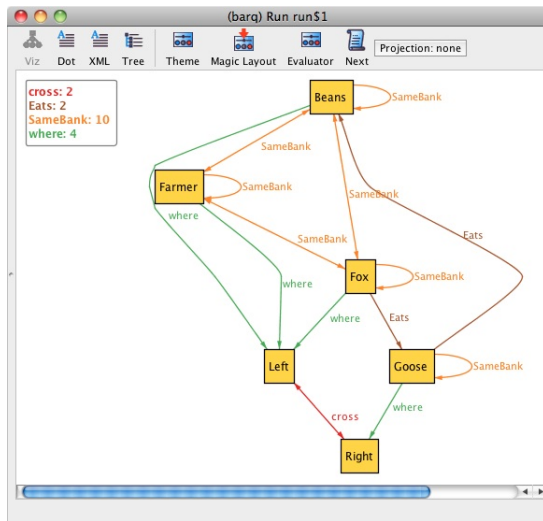
```

Line 21, Column 47 [modified]

PROPOSITIO DE HOMINE ET CAPRA ET LVPO

Carefully observe instance of such an invariant (aside):

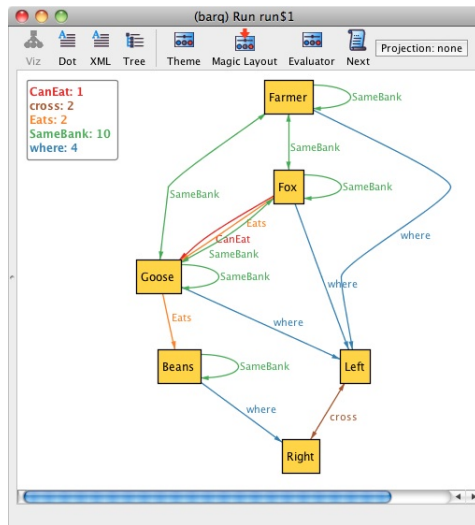
- *SameBank* is an **equivalence** — exactly the **kernel** of *where*
- *Eats* is simple but not transitive
- *cross* is a **bijection**
- *CanEat* is empty
- etc



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

Exercise 28: Knowing that property

$$f \cdot f^\circ \cdot f = f \tag{68}$$

holds for every function f (to be shown later), prove that $\ker f = \frac{f}{f}$ (46) is an **equivalence** relation. $\square$

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.

Indirect equality

Universal properties such as e.g. (62,63) blend nicely with the so-called **indirect equality** way of proving relation equality:

Indirect equality *rules*:

$$R = S \equiv \langle \forall X :: (X \subseteq R \equiv X \subseteq S) \rangle \quad (69)$$

$$\equiv \langle \forall X :: (R \subseteq X \equiv S \subseteq X) \rangle \quad (70)$$

Compare with eg. equality of sets in discrete maths:

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

Indirect relation equality

The typical layout is e.g.

$$\left\{ \begin{array}{l} X \subseteq R \\ \equiv \quad \{ \dots \} \\ X \subseteq \dots \\ \equiv \quad \{ \dots \} \\ X \subseteq S \\ \vdots \quad \{ \text{indirect equality (69)} \} \\ R = S \\ \square \end{array} \right.$$

How universal properties help

$$R \cap \top = R.$$

Why?

Again recall (62)

$$X \subseteq R \cap S \equiv \begin{cases} X \subseteq R \\ X \subseteq S \end{cases}$$

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

$$\begin{aligned} & X \subseteq R \cap \top \\ \equiv & \quad \{ \text{universal property (62)} \} \\ & \begin{cases} X \subseteq R \\ X \subseteq \top \end{cases} \\ \equiv & \quad \{ \top \text{ is above anything} \} \\ & X \subseteq R \\ :: & \quad \{ \text{indirect equality (69)} \} \\ & R \cap \top = R \end{aligned}$$

How universal properties help

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

Take **associativity**

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

as example and follow the reasoning aside.

$$X \subseteq (R \cap S) \cap Q$$

$$\equiv \{ \cap\text{-universal (62) twice} \}$$

$$(X \subseteq R \wedge X \subseteq S) \wedge X \subseteq Q$$

$$\equiv \{ \wedge \text{ is associative} \}$$

$$X \subseteq R \wedge (X \subseteq S \wedge X \subseteq Q)$$

$$\equiv \{ \cap\text{-universal (62) twice} \}$$

$$X \subseteq R \cap (S \cap Q)$$

$$\therefore \{ \text{indirection (69)} \}$$

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

□

Distributivity

As will be shown later, **composition** distributes over **union**

$$R \cdot (S \cup Q) = (R \cdot S) \cup (R \cdot Q) \quad (71)$$

$$(S \cup Q) \cdot R = (S \cdot R) \cup (Q \cdot R) \quad (72)$$

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

$$(S \cap Q) \cdot R = (S \cdot R) \cap (Q \cdot R) \Leftrightarrow \begin{cases} S \cdot \text{img } R \subseteq S \\ \vee \\ Q \cdot \text{img } R \subseteq Q \end{cases} \quad (73)$$

$$R \cdot (Q \cap S) = (R \cdot Q) \cap (R \cdot S) \Leftrightarrow \begin{cases} (\ker R) \cdot Q \subseteq Q \\ \vee \\ (\ker R) \cdot S \subseteq S \end{cases} \quad (74)$$

Exercises

Exercise 29: Prove the **union simplicity** rule:

$$M \cup N \text{ is simple} \quad \equiv \quad M, N \text{ are simple and } M \cdot N^\circ \subseteq id \quad (75)$$

□

Exercise 30: Derive from (75) the corresponding rule for **injective** relations. □

Exercise 31: Explain in your own words the following equalities:

$$1 \xleftarrow{\top} 1 = 1 \xleftarrow{!} 1 = id \quad (76)$$

□

Class 5 — Monotone reasoning

Annex

Background — Eindhoven quantifier calculus

Trading:

$$\langle \forall k : \phi \wedge \varphi : \gamma \rangle = \langle \forall k : \phi : \varphi \Rightarrow \gamma \rangle \quad (77)$$

$$\langle \exists k : \phi \wedge \varphi : \gamma \rangle = \langle \exists k : \phi : \varphi \wedge \gamma \rangle \quad (78)$$

de Morgan:

$$\neg \langle \forall k : \phi : \gamma \rangle = \langle \exists k : \phi : \neg \gamma \rangle \quad (79)$$

$$\neg \langle \exists k : \phi : \gamma \rangle = \langle \forall k : \phi : \neg \gamma \rangle \quad (80)$$

One-point:

$$\langle \forall k : k = e : \gamma \rangle = \gamma[k := e] \quad (81)$$

$$\langle \exists k : k = e : \gamma \rangle = \gamma[k := e] \quad (82)$$

Background — Eindhoven quantifier calculus

Nesting:

$$\langle \forall a, b : \phi \wedge \varphi : \gamma \rangle = \langle \forall a : \phi : \langle \forall b : \varphi : \gamma \rangle \rangle \quad (83)$$

$$\langle \exists a, b : \phi \wedge \varphi : \gamma \rangle = \langle \exists a : \phi : \langle \exists b : \varphi : \gamma \rangle \rangle \quad (84)$$

Rearranging- $\forall$:

$$\langle \forall k : \phi \vee \varphi : \gamma \rangle = \langle \forall k : \phi : \gamma \rangle \wedge \langle \forall k : \varphi : \gamma \rangle \quad (85)$$

$$\langle \forall k : \phi : \gamma \wedge \varphi \rangle = \langle \forall k : \phi : \gamma \rangle \wedge \langle \forall k : \phi : \varphi \rangle \quad (86)$$

Rearranging- $\exists$:

$$\langle \exists k : \phi : \gamma \vee \varphi \rangle = \langle \exists k : \phi : \gamma \rangle \vee \langle \exists k : \phi : \varphi \rangle \quad (87)$$

$$\langle \exists k : \phi \vee \varphi : \gamma \rangle = \langle \exists k : \phi : \gamma \rangle \vee \langle \exists k : \varphi : \gamma \rangle \quad (88)$$

Splitting:

$$\langle \forall j : \phi : \langle \forall k : \varphi : \gamma \rangle \rangle = \langle \forall k : \langle \exists j : \phi : \varphi \rangle : \gamma \rangle \quad (89)$$

$$\langle \exists j : \phi : \langle \exists k : \varphi : \gamma \rangle \rangle = \langle \exists k : \langle \exists j : \phi : \varphi \rangle : \gamma \rangle \quad (90)$$

References



D. Jackson.

Software Abstractions: Logic, Language, and Analysis.

The MIT Press, Cambridge Mass., 2012.

Revised edition, ISBN 0-262-01715-2.



C.B. Jones.

Software Development — A Rigorous Approach.

Prentice-Hall, 1980.

IBSN 0138218846.



J.N. Oliveira.

Using ChatGPT in education - a personal experience, 2025.

INESC TEC Science&Society, 1(7). PDF.