

Part II

CALCULATING WITH RELATIONS

5

WHERE EVERYTHING BECOMES A RELATION

In the previous chapters, (recursive) functions were taken as a basis for expressing computations, exhibiting powerful laws for calculating programs in a functional programming style.

When writing such programs one of course follows some line of thought concerning *what* the programs should do. *What the program should do* is usually understood as the *specification* of the problem that motivates writing the program in the first place. Specifications can be quite complex in real life situations. In other situations, the complexity of the program that one writes is in strong contrast with the simplicity of the specification. Take the example of sorting, which can be specified as simply as:

Yield an ordered permutation of the input.

Where do you find, in this specification, the orientation (or inspiration) that will guide a programmer towards writing a bi-recursive program like *quicksort*?

The question is, then: are functions *enough* for one to calculate functional programs from given specifications? It is the experience in other fields of mathematics that sometimes it is easier to solve a problem of domain D if one generalizes from D to some wider domain D' . In the field of real numbers, for instance, most of trigonometric identities are easily derived (and memorized) from Euler's formula involving complex exponentials: $e^{ix} = \cos x + i(\sin x)$.

Similarly, it turns out that functional programs often become easier to calculate if one handles them in the wider mathematical domain of binary relations. At school one gets accustomed to the sentence *every function is a special case of a relation*. This chapter puts the usefulness of such a piece of common knowledge into practice.

5.1 FUNCTIONS ARE NOT ENOUGH

Consider the following fragment of a requirement posed by a (fictional) telecommunication company:

(...) For each list of calls stored in the mobile phone (eg. numbers dialed, SMS messages, lost calls), the store operation should work in a way such that (a) the more recently a call is made the more accessible

it is; (b) no number appears twice in a list; (c) only the last 10 entries in each list are stored.

A tentative, first implementation of the *store* operation could be

$$\begin{aligned} \text{store} &: \text{Call} \rightarrow \text{Call}^* \rightarrow \text{Call}^* \\ \text{store } c \ l &= c : l \end{aligned}$$

However, such a version of function *store* fails to preserve the *properties* required in the fragment above in case $\text{length } l = 10$, or $c \in \text{elems } l$, where *elems* yields the set of all elements of a finite list,

$$\text{elems} = (|[empty, \text{join}]|) \quad (5.1)$$

for $\text{empty } _ = \{ \}$ and $\text{join } (a, s) = \{a\} \cup s$.

Clearly, the designer would have to *restrict* the application of *store* to input values c, l such that the given properties are preserved. This could be achieved by adding a so-called “*pre-condition*”:

$$\begin{aligned} \text{store} &: \text{Call} \rightarrow \text{Call}^* \rightarrow \text{Call}^* \\ \text{store } c \ l &= c : l \\ \text{pre length } l &< 10 \wedge \neg (c \in \text{elems } l) \end{aligned}$$

Such a pre-condition is a predicate telling the range of *acceptable* input values, to be read as a *warning* provided by the designer that the function will not meet the requirements outside such a range of input values.

Thus *store* becomes a *partial function*, that is, a function defined only for some of its inputs. Although this partiality can be regarded as a symptom that the requirements have been partly misunderstood, it turns out that *partial functions* are the rule rather than the exception in mathematics and computing. For example, in the numeric field, we know what $1/2$ means; what about $1/0$? Ruling out this case means that *division* is a partial function. In list processing, given a sequence s , what does $s !! i$ mean in case $i > \text{length } s$? — list indexing is another *partial operation* (as are *head*, *tail* and so on).

Partial functions are not new to readers of this text: in section 4.1, the *Maybe monad* was used to “totalize” partial functions. In this chapter we shall adopt another strategy to cope with partiality, and one that has extra merits: it will also cope with computational nondeterminacy and vagueness of software requirements.

It can be shown that the following evolution of *store*,

$$\text{store } c = (\text{take } 10) \cdot (c:) \cdot \text{filter } (c \neq) \quad (5.2)$$

meets all requirements above with no need for preconditions, the extra components *take 10* and *filter (c ≠)* being added to comply with requirements (c) and (b), respectively.

Implementation (5.2) alone should be regarded as example of how functional programs can be built compositionally in a requirement-driven fashion. It does not, however, give any guarantees that the

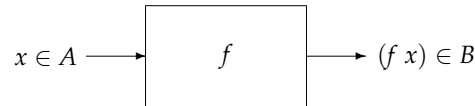
requirements are *indeed* met. How can we ensure this in the compositional way advocated in this book since its beginning? The main purpose of this chapter is to answer such a question.

5.2 FROM FUNCTIONS TO RELATIONS

The way functions are handled and expressed in standard maths books, e.g. in analysis and calculus,

$$y = f(x)$$

is indicative that, more important that the *reactive* behaviour of f ,



which was the starting point of section 2.1, mathematicians are more interested in expressing the input/output *relationship* of f , that is, the set of all pairs (y, x) such that $y = f x$. Such a set of pairs is often referred to as the “graph” of f , which can be plotted two-dimensionally in case types A and B are linearly ordered. (As is the standard case in which $A=B=\mathbb{R}$, the real numbers.)

It turns out that such a *graph* can be regarded as a special case of a *binary relation*. Take for instance the following functional declaration

$$\begin{cases} \text{succ} : \mathbb{N}_0 \rightarrow \mathbb{N}_0 \\ \text{succ } x = x + 1 \end{cases}$$

which expresses the computation rule of the *successor* function. Writing $y = \text{succ } n$ establishes the *binary relation* $y = x + 1$. This binary relation “coincides” with succ in the sense that writing

$$\begin{cases} \text{succ} : \mathbb{N}_0 \rightarrow \mathbb{N}_0 \\ y \text{ succ } x \Leftrightarrow y = x + 1 \end{cases}$$

means the same as the original definition, while making the i/o relationship explicit. Because there is *only one* y such that $y = x + 1$ we can safely drop both ys from $y \text{ succ } x \Leftrightarrow y = x + 1$, obtaining the original $\text{succ } x = x + 1$.

The new style is, however, more expressive, in the sense that it enables us to declare *genuine* binary relations, for instance

$$\begin{cases} R : \mathbb{N}_0 \rightarrow \mathbb{N}_0 \\ y R x \Leftrightarrow y \geq x + 1 \end{cases} \tag{5.3}$$

In this case, not only x and y such that $y = x + 1$ are admissible, but also $y = x + 2$, $y = x + 3$ and so on. It also enables us to express the *converse* of any function — an operation hitherto the privilege of isomorphisms only (2.16):

$$y f x \Leftrightarrow x f^\circ y \tag{5.4}$$

Converses of functions are very useful in problem solving, as we shall soon see. For instance, $\mathbb{N}_0 \xleftarrow{\text{succ}^\circ} \mathbb{N}_0$ denotes the *predecessor* relation in \mathbb{N}_0 . It is not a function because no y such that $y \text{ succ}^\circ 0$ exists — try and solve $0 = y + 1$ in \mathbb{N}_0 .

The intuitions above should suffice for us to start generalizing what we know about functions, from the preceding chapters, to binary relations. First of all, such relations are denoted by *arrows* exactly in the same way functions are. So,

we shall write $R : B \leftarrow A, R : A \rightarrow B, B \xleftarrow{R} A$ or $A \xrightarrow{R} B$
to indicate that relation R relates B -values to A -values.

That is, relations are typed in the same way as functions.

Given binary relation $R : B \leftarrow A$, writing $b R a$ (read: “ b is related to a by R ”) means the same as $a R^\circ b$, where R° is said to be the *converse* of R . In terms of grammar, R° corresponds to the *passive voice* — compare e.g.

$\underbrace{\text{John}}_b \underbrace{\text{loves}}_R \underbrace{\text{Mary}}_a$

with

$\text{Mary} \underbrace{\text{is loved by}}_{R^\circ} \text{John}$

That is, $(\text{loves})^\circ = (\text{is loved by})$. Another example:

Catherine eats the apple

— $R = (\text{eats})$, active voice — compared with

the apple is eaten by Catherine

— $R^\circ = (\text{is eaten by})$, passive voice.

Following a widespread convention, functions are denoted by lowercase characters (eg. f, g, ϕ) or identifiers starting with a lowercase character, while uppercase letters are reserved to arbitrary relations. In the case of functions ($R := f$), $b f a$ means exactly $b = f a$. This is because functions are *univocal*, that is, no two different b and b' are such that $b f a \wedge b' f a$. In fact, the following facts hold about *any* function f :

- *Univocality* (or “left” uniqueness) —

$$b f a \wedge b' f a \Rightarrow b = b' \tag{5.5}$$

- *Leibniz principle* —

$$a = a' \Rightarrow f a = f a' \tag{5.6}$$

Clearly, not every relation obeys (5.5), for instance

$$2 < 3 \wedge 1 < 3 \not\Rightarrow 2 = 1$$

Relations obeying (5.5) will be referred to as *simple*, according to a terminology to follow shortly.

Implication (5.6) expresses the (philosophically) interesting fact that no function (observation) can be found able to distinguish between two equal objects. This is another fact true about functions which does not generalize to binary relations, as we shall see when we come back to this later.

Recapitulating: we regard *function* $f : A \rightarrow B$ as the binary relation which relates b to a iff $b = f a$. So,

$$b f a \text{ literally means } b = f a \quad (5.7)$$

The purpose of this chapter is to generalize from

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

5.3 PRE/POST CONDITIONS

It should be noted that relations are used in virtually every body of science and it is hard to think of another way to express human knowledge in philosophy, epistemology and common life, as suggestively illustrated in figure 5.1. This figure is also illustrative of another popular ingredient when using relations — the *arrows* drawn to denote relationships.¹

In real life, “everything appears to be a relation”. This has lead software theorists to invent linguistic layouts for relational specification, leading to so-called *specification languages*. One such language, today historically relevant, is the language of the Vienna Development Method (VDM). In this notation, the relation described in (5.3) will be written:

$$\begin{array}{l} R (x : \mathbb{N}_0) y : \mathbb{N}_0 \\ \text{post } y \geq x + 1 \end{array}$$

where the clause prefixed by *post* is said to be a post-condition. The format also includes pre-conditions, if necessary. Such is the case of the following pre / post -styled specification of the operation that extracts an arbitrary element from a set:

$$\begin{array}{l} \text{Pick } (x : PA) (r : A, y : PA) \\ \text{pre } x \neq \{ \} \\ \text{post } r \in x \wedge y = x - \{r\} \end{array} \quad (5.8)$$

¹ Our extensive of arrows to denote relations in the sequel is therefore rooted on common, informal practice. Unfortunately, mathematicians do not follow such practice and insist on regarding relations just as sets of pairs.

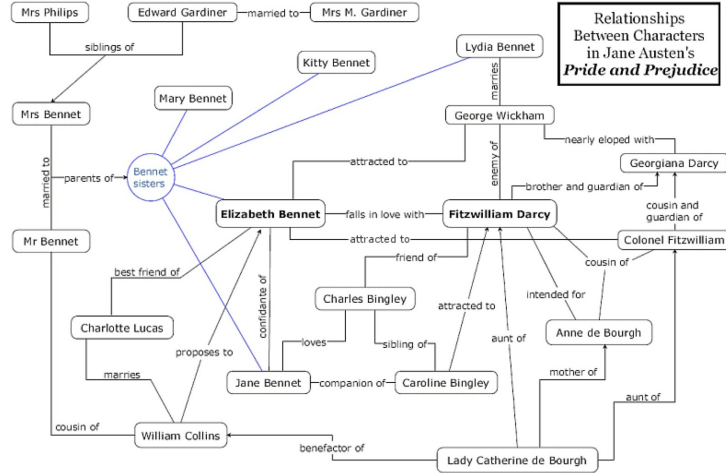


Figure 5.1.: Personal relationships in *Pride and Prejudice*, by Jane Austen, 1813. (Source: Wikipedia)

Here $PA = \{X \mid X \subseteq A\}$ is the set of all subsets of A . Mapping this back to the relational format of (5.3), *Pick* is the relation defined by:

$$\begin{cases} Pick : PA \rightarrow (A \times PA) \\ (r, y) Pick x \Leftrightarrow x \neq \{ \} \wedge r \in x \wedge y = b - \{ r \} \end{cases}$$

Note how $(r, y) Pick \{ \} \Leftrightarrow False$ for whatever r, y . Here follows the specification of *sorting* written in the pre / post -style,

$$\begin{aligned} &Sort (x : A^*) y : A^* \\ &post (ord y) \wedge bag y = bag x \end{aligned} \tag{5.9}$$

where *ord* is the predicate defined in section 3.16 and *bag* is the function that extracts the multiset of elements of a finite list.² Note how *Sort* defines sorting independently of giving an explicit algorithm. In fact, the pre / post -style provides a way of *hiding* the algorithmic details that any particular functional implementation is bound to include.

Wherever a post-condition is intended to specify a function f , one refers to such a condition as an *implicit specification* of f . Examples: *explicit* definition of the *abs* function

$$\begin{aligned} &abs : \mathbb{Z} \rightarrow \mathbb{Z} \\ &abs i = \mathbf{if} i < 0 \mathbf{then} -i \mathbf{else} i \end{aligned}$$

followed by an *implicit specification* of the same function:

$$\begin{aligned} &abs (i : \mathbb{Z}) r : \mathbb{Z} \\ &post r \geq 0 \wedge (r = i \vee r = -i) \end{aligned}$$

² Recall that *ord* assumes an ordering on type A . For further developments on this specification see exercise 5.17 later on.

Explicit definition of *max* function:

$$\begin{aligned} \text{max} &: \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z} \\ \text{max}(i, j) &= \text{if } i \leq j \text{ then } j \text{ else } i \end{aligned}$$

Its *implicit specification*:

$$\begin{aligned} \text{max}(i: \mathbb{Z}, j: \mathbb{Z}) r: \mathbb{Z} \\ \text{post } r \in \{i, j\} \wedge i \leq r \wedge j \leq r \end{aligned}$$

Of a different nature is the following *pre/post-pair*:

$$\begin{aligned} \text{Sqrt}: (i: \mathbb{R}) r: \mathbb{R} \\ \text{post } r^2 = i \end{aligned}$$

Here the *specifier* is telling the *implementer* that either solution $r = +\sqrt{i}$ or $r = -\sqrt{i}$ will do.³ Indeed, *square root* is not a function, it is the binary relation:

$$r \text{ Sqrt } i \Leftrightarrow r^2 = i \tag{5.10}$$

We proceed with a thorough study of the concept of a binary relation, by analogy with a similar study carried out about functions in chapter 2.

5.4 RELATIONAL COMPOSITION AND CONVERSE

Such as functions, relations can be combined via composition ($R \cdot S$), defined as follows:

$$\begin{array}{c} B \xleftarrow{R} A \xleftarrow{S} C \\ \text{R} \cdot \text{S} \end{array} \quad b(R \cdot S)c \equiv \langle \exists a : b R a : a S c \rangle \tag{5.11}$$

Example: $\text{Uncle} = \text{Brother} \cdot \text{Parent}$, expanding to

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

An explanation on the \exists -notation is on demand: \exists is an instance of a so-called *quantifier*, a main ingredient of formal logic. In this book we follow the so-called *Eindhoven quantifier* notation, whereby expressions of the form

$$\langle \forall x : P : Q \rangle$$

mean

“for **all** x in the range P , Q holds”

where P and Q are logical expressions involving x ; and expressions of the form

$$\langle \exists x : P : Q \rangle$$

³ This aspect of formal specification is called *vagueness*.

mean

“for *some* x in the range P , Q holds”.

Note how the symbols \exists and \forall “twist” letters E (exists) and A (all), respectively. P is known as the *range* of the quantification and Q as the quantified *term*.⁴ This logical notation enjoys a well-known set of properties, some of which are given in appendix A.2. As an example, by application of the \exists -trading rule (A.2), predicate $\langle \exists a :: b R a \wedge a S c \rangle$ in (5.11) can be written $\langle \exists a : b R a : a S c \rangle$.

Note how (5.11) *removes* \exists and bound variable a when applied from right to left. This is an example of conversion from pointwise to point-free notation, since “point” a also disappears. Indeed, we shall try and avoid lengthy, complex \forall, \exists -formulae by converting them to *pointfree* notation, as is the case in (5.11) once relational composition is used.

A simple calculation shows (5.11) to instantiate to (2.6) for the special case where R and S are functions, $R, S := f, g$:

$$\begin{aligned}
 b(f \cdot g)c &\equiv \langle \exists a :: b f a \wedge a g c \rangle \\
 &\equiv \{ \text{functions are univocal (simple) relations} \} \\
 &\quad \langle \exists a :: b = f a \wedge a = g c \rangle \\
 &\equiv \{ \exists\text{-trading rule (A.2)} \} \\
 &\quad \langle \exists a : a = g c : b = f a \rangle \\
 &\equiv \{ \exists\text{-“one-point” rule (A.6)} \} \\
 &\quad b = f (g c) \\
 &\quad \square
 \end{aligned}$$

Like its functional version (2.8), relation composition is associative:

$$R \cdot (S \cdot P) = (R \cdot S) \cdot P \tag{5.12}$$

Everywhere $T = R \cdot S$ holds, the replacement of T by $R \cdot S$ will be referred to as a “factorization” and that of $R \cdot S$ by T as “fusion”. Every relation $B \xleftarrow{R} A$ admits two trivial factorizations,

$$\begin{cases} R = R \cdot id_A \\ R = id_B \cdot R \end{cases} \tag{5.13}$$

where, for every X , id_X is the identity relation relating every element of X with itself (2.9). In other words: the identity (equality) *relation* coincides with the identity *function*.

In section 2.7 we introduced a very special case of function f — isomorphism — which has a converse f° such that (2.16) holds. A major advantage of generalizing functions to relations is that *every* relation $A \xrightarrow{R} B$ has a converse $A \xleftarrow{R^\circ} B$ defined by

$$b R a \iff a R^\circ b \tag{5.14}$$

⁴ In particular, Q or P can be universally False or True. Assertions of the form $\langle \forall x : \text{True} : Q \rangle$ or $\langle \exists x : \text{True} : Q \rangle$ are abbreviated to $\langle \forall x :: Q \rangle$ or $\langle \exists x :: Q \rangle$, respectively.

— the *passive voice* written relationally, as already mentioned. Two important properties of converse follow: it is an involution

$$(R^\circ)^\circ = R \tag{5.15}$$

and it commutes with composition in a contravariant way:

$$(R \cdot S)^\circ = S^\circ \cdot R^\circ \tag{5.16}$$

Converses of functions enjoy a number of properties from which the following is singled out as a way to introduce / remove them from logical expressions:

$$b(f^\circ \cdot R \cdot g)a \equiv (f b)R(g a) \tag{5.17}$$

For instance, the consequent of implication (5.6) could have been written $a(f^\circ \cdot id \cdot f)a'$, or even simpler as $a(f^\circ \cdot f)a'$, as it takes very little effort to show:

$$\begin{aligned} & a(f^\circ \cdot id \cdot f)a' \\ \equiv & \quad \{ (5.17) \} \\ & (f a)id(f a') \\ \equiv & \quad \{ b f a \equiv b = f a \} \\ & (f a) = id(f a') \\ \equiv & \quad \{ (2.9) \} \\ & f a = f a' \\ \square \end{aligned}$$

Exercise 5.1. Let $sq\ x = x^2$ be the function that computes the square of a real number. Use (5.17) to show that (5.10) reduces to

$$Sqrt = sq^\circ$$

in relational pointfree notation.

□

Exercise 5.2. Give an implicit definition of function $f\ x = x^2 - 1$ in the form of a post-condition not involving subtraction. Then re-write it without variables using (5.17).

□

5.5 RELATIONAL EQUALITY

Recall that function equality (2.5) is established by extensionality:

$$f = g \text{ iff } \langle \forall a : a \in A : f a = g a \rangle$$

Also recall that $f = g$ only makes sense iff both functions have the same type, say $A \rightarrow B$. Can we do the same for relations? The relational generalization of (2.5) will be

$$R = S \text{ iff } \langle \forall a, b : a \in A \wedge b \in B : b R a \Leftrightarrow b S a \rangle \quad (5.18)$$

Since \Leftrightarrow is bi-implication, we can replace the term of the quantification by

$$(b R a \Rightarrow b S a) \wedge (b S a \Rightarrow b R a)$$

Now, what does $b R a \Rightarrow b S a$ mean? It simply captures relational inclusion,

$$R \subseteq S \text{ iff } \langle \forall a, b :: b R a \Rightarrow b S a \rangle \quad (5.19)$$

whose righthand side can also be written

$$\langle \forall a, b : b R a : b S a \rangle$$

by \forall -trading (A.1). Note the same pointwise-pointfree move when one reads (5.19) from right to left: \forall, a and b disappear.

Altogether, (5.18) can be written in less symbols as follows:

$$R = S \quad \equiv \quad R \subseteq S \wedge S \subseteq R \quad (5.20)$$

This way of establishing relational equality is usually referred to as *circular inclusion*. Note that relational inclusion (5.19) is a partial order: it is *reflexive*, since

$$R \subseteq R \quad (5.21)$$

holds for every R ; it is *transitive*, since for all R, S, T

$$R \subseteq S \wedge S \subseteq T \Rightarrow R \subseteq T \quad (5.22)$$

holds; and it is *antisymmetric*, as established by circular-inclusion (5.20) itself. Circular-inclusion is also jocosely known as the “ping-pong” method for establishing $R = S$: first calculate $R \subseteq S$ (“ping”) and then $S \subseteq R$ (“pong”). This can be performed in one go by adopting the following calculation layout:

$$\begin{aligned} R &\subseteq \dots \\ &\subseteq S \\ &\subseteq \dots \\ &\subseteq R \\ &\square \end{aligned}$$

This has the advantage of making apparent that not only R and S are the same, but also that every two steps in the circular reasoning are so (just choose a different start and stop point in the “circle”).

Circular inclusion (5.20) is not the only way to establish relational equality. A less obvious, but very useful way of calculating the equality of two relations is the method of *indirect equality*:

$$R = S \equiv \langle \forall X :: (X \subseteq R \Leftrightarrow X \subseteq S) \rangle \quad (5.23)$$

$$\equiv \langle \forall X :: (R \subseteq X \Leftrightarrow S \subseteq X) \rangle \quad (5.24)$$

The reader unaware of this way of indirectly setting algebraic equalities will recognize that the same pattern of indirection is used when establishing set equality via the membership relation, cf.

$$A = B \equiv \langle \forall x :: x \in A \Leftrightarrow x \in B \rangle$$

The typical layout of using any of these rules is the following:

$$\left\{ \begin{array}{l} X \subseteq R \\ \equiv \quad \{ \dots \} \\ X \subseteq \dots \\ \equiv \quad \{ \dots \} \\ X \subseteq S \\ :: \quad \{ \text{indirect equality (5.23)} \} \\ R = S \\ \square \end{array} \right.$$

This proof method is very powerful and we shall make extensive use of it in the sequel. (The curious reader can have a quick look at section 5.9 for a simple illustration.)

RELATIONAL TYPES. From this point onwards we shall regard the type $B \leftarrow A$ as including not only all functions $f : A \rightarrow B$ but also all relations of the same type, $R : A \rightarrow B$. This is far more than we had before! In particular, type $A \rightarrow B$ includes:

- the *bottom* relation $B \leftarrow^{\perp} A$, which is such that, for all b, a ,

$$b \perp a \equiv \text{FALSE}$$

- the *topmost* relation $B \leftarrow^{\top} A$, which is such that, for all b, a ,

$$b \perp a \equiv \text{TRUE}$$

The former is referred to as the void, or *empty* relation. The latter is known as the universal, or *coexistence* relation. Clearly, for every R ,

$$\perp \subseteq R \subseteq \top \quad (5.25)$$

and

$$R \cdot \perp = \perp \cdot R = \perp \tag{5.26}$$

hold. By (5.25) and (5.20), writing $R = \perp$ (respectively, $R = \top$) is the same as writing $R \subseteq \perp$ (respectively, $\top \subseteq R$).

A relation $B \xleftarrow{V} A$ is said to be a *vector* if either A or B are the singleton type 1. Relation $1 \xleftarrow{X} A$ is said to be a *row-vector*; clearly, $X \subseteq !$. Relation $B \xleftarrow{Z} 1$ is said to be a *column-vector*; clearly, $Z \subseteq !^\circ$.⁵ A relation of type $1 \xleftarrow{S} 1$ is called a *scalar*.

Last but not least, note that in a relational setting types $B \leftarrow A$ and B^A do not coincide — B^A is the type of all *functions* from A to B , while $B \leftarrow A$ is the type of all *relations* from A to B . Clearly, $B^A \subseteq B \leftarrow A$.

5.6 DIAGRAMS

As happens with functions, the arrow notation adopted for functions makes it possible to express relational formulæ using diagrams. This is a major ingredient of the relational method because it provides a graphical way of picturing relation types and relational constraints.

Paths in diagrams are built by arrow chaining, which corresponds to relational composition $R \cdot S$ (5.11), meaning “... is R of some S of ...” in natural language.

Assertions of the form $X \subseteq Y$ where X and Y are relation compositions can be represented graphically by rectangle-shaped diagrams, as is the case in

$$\begin{array}{ccc}
 \text{Descriptor} & \xleftarrow{FT} & \text{Handle} \\
 \text{path} \downarrow & \subseteq & \downarrow \top \\
 \text{Path} & \xleftarrow{FS^\circ} & \text{File}
 \end{array} \tag{5.27}$$

in the context of modelling a file-system. Relation FS models a *file store* (a table mapping file system paths to the respective files), FT is the *open-file descriptor* table (holding the information about the files that are currently open⁶), function $path$ yields the path of a file descriptor and \top is the largest possible relation between file-handles and files, as seen above. The diagram depicts the constraint:

$$path \cdot FT \subseteq FS^\circ \cdot \top \tag{5.28}$$

What does (5.28) mean, then, in predicate logic?

⁵ The column and row qualifiers have to do with an analogy with vectors in linear algebra.
⁶ Open files are manipulated by the file system via open file descriptor data structures, which hold various relevant metadata (e.g. current position within the file). Such descriptors are identified by file handles which the file system provides to applications that manipulate files. This indirection layer avoids unnecessary coupling between applications and the details of the file system implementation.

FROM DIAGRAMS TO LOGIC. We reason:

$$\begin{aligned}
& path \cdot FT \subseteq FS^\circ \cdot \top \\
\equiv & \quad \{ \text{'at most' ordering (5.19)} \} \\
& \langle \forall p, h : p(path \cdot FT)h : p(FS^\circ \cdot \top)h \rangle \\
\equiv & \quad \{ \text{composition (5.11); } path \text{ is a function} \} \\
& \langle \forall p, h : \langle \exists d : p = path \, d : d \, FT \, h \rangle : p(FS^\circ \cdot \top)h \rangle \\
\equiv & \quad \{ \text{quantifier calculus — splitting rule (A.13)} \} \\
& \langle \forall d, h : d \, FT \, h : \langle \forall p : p = path \, d : p(FS^\circ \cdot \top)h \rangle \rangle \\
\equiv & \quad \{ \text{quantifier calculus — } \forall\text{-one-point rule (A.5)} \} \\
& \langle \forall d, h : d \, FT \, h : (path \, d)(FS^\circ \cdot \top)h \rangle
\end{aligned}$$

We still have to unfold term $(path \, d)(FS^\circ \cdot \top)h$:

$$\begin{aligned}
& (path \, d)(FS^\circ \cdot \top)h \\
\equiv & \quad \{ \text{composition (5.11)} \} \\
& \langle \exists x :: (path \, d)FS^\circ x \wedge x \top h \rangle \\
\equiv & \quad \{ \text{converse ; } x \top h \text{ always holds} \} \\
& \langle \exists x :: x \, FS \, (path \, d) \rangle
\end{aligned}$$

In summary, $path \cdot FT \subseteq FS^\circ \cdot \top$ unfolds into

$$\langle \forall d, h : d \, FT \, h : \langle \exists x :: x \, FS \, (path \, d) \rangle \rangle \quad (5.29)$$

Literally:

If h is the handle of some open-file descriptor d , then this holds the path of some existing file x .

In fewer words:

Non-existing files cannot be opened (referential integrity).

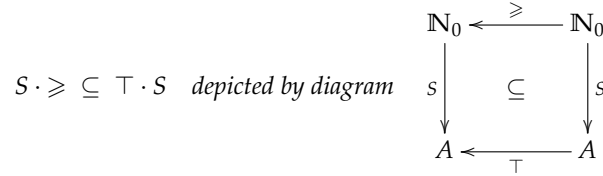
Thus we see how relation diagrams “hide” logically quantified formulæ capturing properties of the systems one wishes to describe.

Compared with the commutative diagrams of previous chapters, a diagram

$$\begin{array}{ccc}
A & \xleftarrow{S} & B \\
R \downarrow & \subseteq & \downarrow P \\
C & \xleftarrow{Q} & D
\end{array}$$

is said to be *semi-commutative* because $Q \cdot P \subseteq R \cdot S$ is not forced to hold, only $R \cdot S \subseteq Q \cdot P$ is. In case both hold, the \subseteq symbol is dropped, cf. (5.20).

Exercise 5.3. Let a $S n$ mean: “student a is assigned number n ”. Using (5.11) and (5.19), check that assertion



means that numbers are assigned to students in increasing order.

□

5.7 TAXONOMY OF BINARY RELATIONS

The Leibniz principle about functions (5.6) can now be simplified thanks to equivalence (5.19), as shown next:

$$\begin{aligned}
 & \langle \forall a, a' :: a = a' \Rightarrow f a = f a' \rangle \\
 \equiv & \quad \{ \text{introduction of } id; \text{ consequent as calculated already} \} \\
 & \langle \forall a, a' :: a = id a' \Rightarrow a(f^\circ \cdot f)a' \rangle \\
 \equiv & \quad \{ b f a \text{ means the same as } b = f a \} \\
 & \langle \forall a, a' :: a id a' \Rightarrow a(f^\circ \cdot f)a' \rangle \\
 \equiv & \quad \{ (5.19) \} \\
 & id \subseteq f^\circ \cdot f \tag{5.30}
 \end{aligned}$$

A similar calculation will reduce univocality (5.5) to

$$f \cdot f^\circ \subseteq id \tag{5.31}$$

Thus a function f is characterized by comparing $f^\circ \cdot f$ and $f \cdot f^\circ$ with the identity.⁷

The exact characterization of functions as special cases of relations is achieved in terms of converse, which is in fact of paramount importance in establishing the whole taxonomy of binary relations depicted in figure 5.2. First, we need to define two important notions: given a relation $B \xleftarrow{R} A$, the *kernel* of R is the relation $A \xleftarrow{\ker R} A$ defined by:

$$\ker R = R^\circ \cdot R \tag{5.32}$$

⁷ As we shall see in section 5.13, relations larger than the identity ($id \subseteq R$) are said to be *reflexive* and relations at most the identity ($R \subseteq id$) are said to be *coreflexive* or *partial identities*.

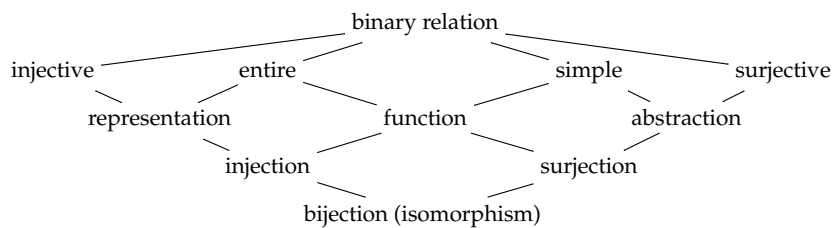


Figure 5.2.: Binary relation taxonomy

Clearly, $a' \ker R a$ holds between any two sources a and a' which have (at least) a common target c such that $c R a'$ and $c R a$. We can also define its dual, $B \xleftarrow{\text{img } R} B$, called the *image* of R , defined by:⁸

$$\text{img } R \stackrel{\text{def}}{=} R \cdot R^\circ \tag{5.33}$$

From (5.15, 5.16) one immediately draws:

$$\ker (R^\circ) = \text{img } R \tag{5.34}$$

$$\text{img } (R^\circ) = \ker R \tag{5.35}$$

Kernel and image lead to the four top criteria of the taxonomy of figure 5.2:

	<i>Reflexive</i>	<i>Coreflexive</i>	
$\ker R$	entire R	injective R	(5.36)
$\text{img } R$	surjective R	simple R	

In words: a relation R is said to be *entire* (or total) iff its kernel is reflexive and to be *simple* (or functional) iff its image is coreflexive. Dually, R is *surjective* iff R° is entire, and R is *injective* iff R° is simple.

Representing binary relations by Boolean matrices gives us a simple, graphical way of detecting properties such as simplicity, surjectiveness, and so on. Let the enumerated types $A = \{a_1, a_2, a_3, a_4, a_5\}$ and $B = \{b_1, b_2, b_3, b_4, b_5\}$ be given. Two examples of relations of type $A \rightarrow B$ are given in figure 5.3 — the leftmost and the rightmost, which we shall refer to as R and S , respectively.⁹ The matrix representing R is:

	a_1	a_2	a_3	a_4	a_5	
b_1	0	1	0	0	0	(5.37)
b_2	1	0	0	0	0	
b_3	0	0	1	1	0	
b_4	0	0	0	0	1	
b_5	0	0	0	0	0	

⁸ These operators are relational extensions of two concepts familiar from set theory: the image of a function f , which corresponds to the set of all y such that $\langle \exists x :: y = f x \rangle$, and the kernel of f , which is the equivalence relation $b \ker f a \Leftrightarrow (f b) = (f a)$. (See exercise 5.8 later on.)

⁹ Credits: <http://www.matematikaria.com/unit/injective-surjective-bijective.html>. Note that we enumerate a_1, a_2, \dots from the top to the bottom.

The 1 addressed by b_2 and a_1 means that $b_2 R a_1$ holds, that between b_1 and a_2 means $b_1 R a_2$, and so on and so forth. Then, R is:

- *simple* because there is *at most* one 1 in every column
- *entire* because there is *at least* one 1 in every column
- not *injective* because there is *more than* one 1 in some row
- not *surjective* because some row (the last) has no 1s.

So this relation is a *function* that is neither an injection nor a surjection.

Let us now have a look at the matrix that represents $S : A \rightarrow B$:

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

Now every row and every column has *exactly* one 1 — this tells us that S is not only a function but in fact a bijection. Looking at the matrix that represents $S^\circ : A \leftarrow B$,

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

we realize that it also is a function, in fact another bijection. This gives us a rule of thumb for (constructively) checking for bijections (isomorphisms):

$$\text{A relation } f \text{ is a bijection iff its converse } f^\circ \text{ is a function } g \quad (5.38)$$

Then g is also a bijection since $f^\circ = g \Leftrightarrow f = g^\circ$. Recall how some definitions of isomorphisms given before, e.g. (2.92), are nothing but applications of this rule $f^\circ = g$, once written pointwise with the help of (5.17):

$$f b = a \Leftrightarrow b = g a$$

Bijections (isomorphisms) are reversible functions — they don't lose any information. By contrast, $! : A \rightarrow 1$ (2.58) and indeed all constant functions $\underline{c} : A \rightarrow C$ (2.12) lose all the information contained in their inputs, recall (2.14). This property is actually more general,

$$\underline{c} \cdot R \subseteq \underline{c} \quad (5.39)$$

for all suitably typed R .

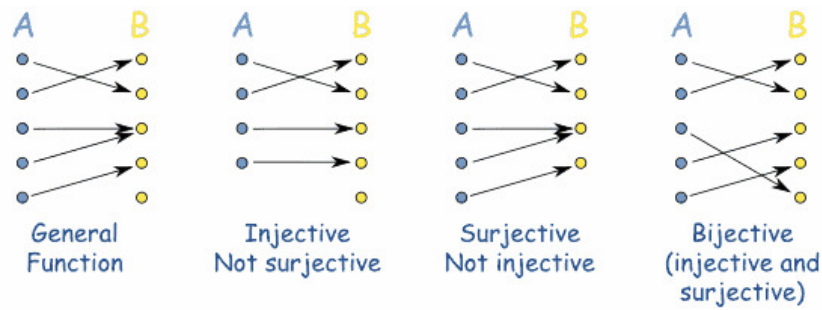


Figure 5.3.: Four binary relations.

In the same way $! : A \rightarrow 1$ is always a constant function — in fact the *unique* possible function of its type, $f : 1 \rightarrow A$ is bound to be a constant function too, for any choice of a target value in non-empty A . Because there are as many such functions as elements if A , functions $\underline{a} : 1 \rightarrow A$ are referred to as *points*. These two situations correspond to isomorphisms $1^A \cong 1$ (2.97) and $A^1 \cong A$ (2.98), respectively. Two short-hands are introduced for the constant functions

$$\text{true} = \underline{\text{True}} \tag{5.40}$$

$$\text{false} = \underline{\text{False}} \tag{5.41}$$

Exercise 5.4. Prove (5.38) by completing:

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

□

Exercise 5.5. Compute, for the relations in figure 5.3, the kernel and the image of each relation. Why are all these relations functions? (NB: note that the types are not all the same.)

□

Exercise 5.6. Recall the definition of a constant function (2.12),

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

where K is assumed to be non-empty. Show that $\ker \underline{k} = \top$ and compute which relations are defined by the expressions

$$\underline{b} \cdot \underline{c}^\circ, \quad \text{img } \underline{k} \tag{5.42}$$

Finally, show that (5.39) holds.

□

Exercise 5.7. Resort to (5.34,5.35) and (5.36) to prove the following rules of thumb:

- converse of injective is simple (and vice-versa) (5.43)

- converse of entire is surjective (and vice-versa) (5.44)

□

Exercise 5.8. Given a function $B \xleftarrow{f} A$, calculate the pointwise version

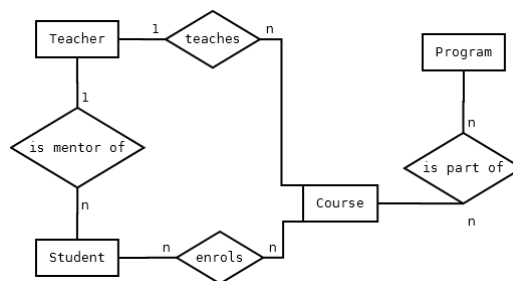
$$b(\ker f)a \equiv f b = f a \tag{5.45}$$

of $\ker f$. What is the outcome of the same exercise for $\text{img } f$?

□

ENTITY-RELATIONSHIP DIAGRAMS In the tradition of relational databases, so-called *entity-relationship* (ER) diagrams have become popular as an informal means for capturing the properties of the relationships involved in a particular database design.

Consider the following example of one such diagram:¹⁰



In the case of relation

$$Teacher \xleftarrow{\text{is mentor of}} Student$$

¹⁰ Credits: <https://dba.stackexchange.com/questions>.

the drawing tells not only that some teacher may mentor more than one student, but also that a given student has exactly one mentor. So *is mentor of* is a *simple* relation (figure 5.2).

The possibility $n = 0$ allows for students with no mentor. Should this possibility be ruled out ($n > 1$), the relation would become also *entire*, i.e. a function. Then

t is mentor of s

could be written

$t = \text{is mentor of } s$

— recall (5.7)— meaning:

t is *the* mentor of student s .

That is, *is mentor of* would become an *attribute* of *Student*. Note how definite article “the” captures the presence of functions in normal speech. “The” means not only determinism (one and only one output) but also definedness (there is always one such output). In the case of *is mentor of* being simple but not entire, we have to say:

t is *the* mentor of student s , if any.

Exercise 5.9. Complete the exercise of declaring in $A \xrightarrow{R} B$ notation the other relations of the ER-diagram above and telling which properties in Figure 5.2 are required for such relations.

□

5.8 FUNCTIONS, RELATIONALLY

Among all binary relations, functions play a central role in relation algebra — as can be seen in figure 5.2. Recapitulating, 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)

It turns out that *any* function f enjoys the following properties, known as *shunting rules*:

$$f \cdot R \subseteq S \equiv R \subseteq f^\circ \cdot S \tag{5.46}$$

$$R \cdot f^\circ \subseteq S \equiv R \subseteq S \cdot f \tag{5.47}$$

These will prove extremely useful in the sequel. Another very useful fact is the function *equality rule*:

$$f \subseteq g \equiv f = g \equiv f \supseteq g \quad (5.48)$$

Rule (5.48) follows immediately from (5.46,5.47) by “cyclic inclusion” (5.20):

$$\begin{aligned} & f \subseteq g \\ \equiv & \quad \{ \text{natural-id (2.10)} \} \\ & f \cdot id \subseteq g \\ \equiv & \quad \{ \text{shunting on } f \text{ (5.46)} \} \\ & id \subseteq f^\circ \cdot g \\ \equiv & \quad \{ \text{shunting on } g \text{ (5.47)} \} \\ & id \cdot g^\circ \subseteq f^\circ \\ \equiv & \quad \{ \text{converses; identity} \} \\ & g \subseteq f \end{aligned}$$

Then:

$$\begin{aligned} & f = g \\ \equiv & \quad \{ \text{cyclic inclusion (5.20)} \} \\ & f \subseteq g \wedge g \subseteq f \\ \equiv & \quad \{ \text{above} \} \\ & f \subseteq g \\ \equiv & \quad \{ \text{above} \} \\ & g \subseteq f \\ \square \end{aligned}$$

Exercise 5.10. Infer $id \subseteq \ker f$ (f is entire) and $\text{img } f \subseteq id$ (f is simple) from shunting rules (5.46) and (5.47).

□

Exercise 5.11. For $R := f$, the property (5.39) “immediately” coincides with (2.14). Why?

□

FUNCTION DIVISION. Given two functions $B \xrightarrow{g} C \xleftarrow{f} A$, we can compose f with the converse of g . This turns out to be a very frequent pattern in relation algebra, known as the *division* of f by g :

$$\frac{f}{g} = g^\circ \cdot f \quad \text{cf.} \quad \begin{array}{ccc} & B & \xleftarrow{\frac{f}{g}} & A \\ & \searrow g & & \swarrow f \\ & & C & \end{array} \quad (5.49)$$

That is,

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

Think of the sentence:

Mary lives where John was born.

This can be expressed by a division:

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

Thus $R = \frac{\text{birthplace}}{\text{residence}}$ is the relation "... resides in the birthplace of ...". In general,

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

This combinator enjoys a number of interesting properties, for instance:

$$\frac{f}{id} = f \quad (5.50)$$

$$\left(\frac{f}{g}\right)^\circ = \frac{g}{f} \quad (5.51)$$

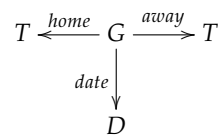
$$\frac{f \cdot h}{g \cdot k} = k^\circ \cdot \frac{f}{g} \cdot h \quad (5.52)$$

$$\frac{f}{f} = \ker f \quad (5.53)$$

$$a \neq b \Leftrightarrow \frac{a}{b} = \perp \quad (5.54)$$

Function division is a special case of the more general, and important, concept of relational division, a topic that shall be addressed in section 5.19.

Exercise 5.12. 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

$$id \subseteq \frac{away}{home} \cdot \frac{away}{home} \tag{5.55}$$

captures one of the requirements above — which?
□

Exercise 5.13. Check the properties of function division given above.

□

5.9 MEET AND JOIN

Like sets, two relations of the same type, say $B \xleftarrow{R,S} A$, can be intersected or joined in the obvious way:

$$b (R \cap S) a \equiv b R a \wedge b S a \tag{5.56}$$

$$b (R \cup S) a \equiv b R a \vee b S a \tag{5.57}$$

$R \cap S$ is usually called *meet* (intersection) and $R \cup S$ is called *join* (union). They lift pointwise conjunction and disjunction, respectively, to the pointfree level. Their meaning is nicely captured by the following *universal* properties:¹¹

$$X \subseteq R \cap S \equiv X \subseteq R \wedge X \subseteq S \tag{5.58}$$

$$R \cup S \subseteq X \equiv R \subseteq X \wedge S \subseteq X \tag{5.59}$$

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

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

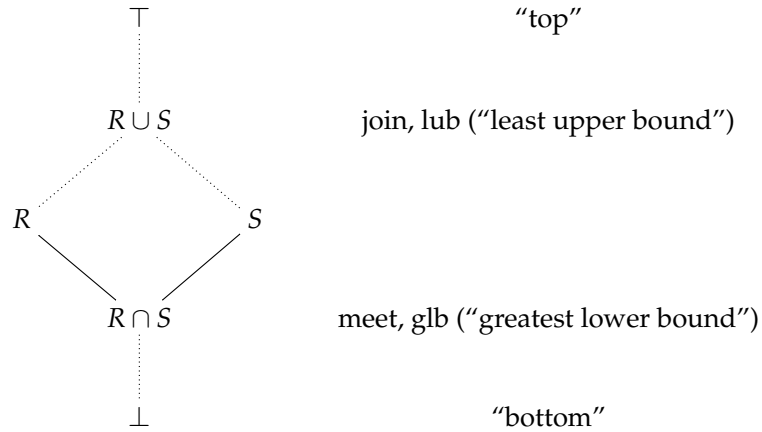
proved next by indirect equality (5.23):

$$\begin{aligned} & X \subseteq (R \cap S) \cap T \\ \equiv & \quad \{ \cap\text{-universal (5.58) twice } \} \\ & (X \subseteq R \wedge X \subseteq S) \wedge X \subseteq T \\ \equiv & \quad \{ \wedge \text{ is associative } \} \\ & X \subseteq R \wedge (X \subseteq S \wedge X \subseteq T) \\ \equiv & \quad \{ \cap\text{-universal (5.58) twice } \} \\ & X \subseteq R \cap (S \cap T) \\ \therefore & \quad \{ \text{indirection (5.23)} \} \\ & (R \cap S) \cap T = R \cap (S \cap T) \end{aligned}$$

□

¹¹ Recall the generic notions of *greatest lower bound* and *least upper bound*, respectively.

In summary, type $B \leftarrow A$ forms a lattice:



DISTRIBUTIVE PROPERTIES. As it will be proved later, *composition* distributes over *union*

$$R \cdot (S \cup T) = (R \cdot S) \cup (R \cdot T) \tag{5.60}$$

$$(S \cup T) \cdot R = (S \cdot R) \cup (T \cdot R) \tag{5.61}$$

while distributivity over *intersection* is side-conditioned:

$$(S \cap Q) \cdot R = (S \cdot R) \cap (Q \cdot R) \iff \begin{cases} Q \cdot \text{img } R \subseteq Q \\ \vee \\ S \cdot \text{img } R \subseteq S \end{cases} \tag{5.62}$$

$$R \cdot (Q \cap S) = (R \cdot Q) \cap (R \cdot S) \iff \begin{cases} (\text{ker } R) \cdot Q \subseteq Q \\ \vee \\ (\text{ker } R) \cdot S \subseteq S \end{cases} \tag{5.63}$$

Properties (5.60,5.61) express the *bilinearity* of relation composition with respect to relational join. These, and properties such as e.g.

$$(R \cap S)^\circ = R^\circ \cap S^\circ \tag{5.64}$$

$$(R \cup S)^\circ = R^\circ \cup S^\circ \tag{5.65}$$

will be shown to derive from a general construction that will be explained in section 5.18.

Exercise 5.14. Show that

$$R \cap \perp = \perp \tag{5.66}$$

$$R \cap \top = R \tag{5.67}$$

$$R \cup \top = \top \tag{5.68}$$

$$R \cup \perp = R \tag{5.69}$$

using neither (5.56) nor (5.57).

□

Exercise 5.15. Prove the union simplicity rule:

$$M \cup N \text{ is simple} \equiv M, N \text{ are simple and } M \cdot N^\circ \subseteq id \quad (5.70)$$

Using converses, derive from (5.70) the corresponding rule for injective relations.

□

Exercise 5.16. Prove the distributive property:

$$g^\circ \cdot (R \cap S) \cdot f = g^\circ \cdot R \cdot f \cap g^\circ \cdot S \cdot f \quad (5.71)$$

□

Exercise 5.17. Let $bag : A^* \rightarrow \mathbb{N}_0^A$ be the function that, given a finite sequence (list), indicates the number of occurrences of its elements, for instance,

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

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

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

Let $ord : A^* \rightarrow \mathbb{B}$ be the obvious predicate assuming a total order predefined in A . Finally, let $true = \underline{\text{True}}$ (5.40). Having defined

$$S = \frac{bag}{bag} \cap \frac{true}{ord} \quad (5.72)$$

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

□

Exercise 5.18. Derive the distributive properties:

$$\frac{f \cup g}{k} = \frac{f}{k} \cup \frac{g}{k}, \quad \frac{f \cap g}{k} = \frac{f}{k} \cap \frac{g}{k} \quad (5.73)$$

□

5.10 RELATIONAL THINKING

Binary relations provide a natural way of describing real life situations. Relation algebra can be used to reason about such formal descriptions. This can be achieved using suitable relational combinators (and their laws), in the *pointfree* style.

Let us see a simple example of such a *relational thinking* taking one of the PROPOSITIONES AD ACUENDOS IUUENES (“Problems to Sharpen the Young”) proposed by abbot Alcuin of York († 804) as case study. Alcuin states his puzzle in the following way, in Latin:

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?*

Our starting point will be the following (rather free) translation of the above to English:

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?*

We wish to describe the essence of this famous puzzle, which is the *guarantee* that

under no circumstances does the fox eat the goose or the goose eat the beans.

Clearly, we need two data types:

$Being = \{Farmer, Fox, Goose, Beans\}$
 $Bank = \{Left, Right\}$

Then we identify a number of relations involving such data:

$$\begin{array}{ccc} Being & \xrightarrow{Eats} & Being \\ & \text{where} \downarrow & \\ & & Bank \xrightarrow{cross} Bank \end{array} \quad (5.74)$$

Clearly, $cross\ Left = Right$ and $cross\ Right = Left$. So $cross$ is its own inverse and therefore a bijection (5.38). Relation $Eats$ can be described by the Boolean matrix:

$$Eats = \begin{array}{c|cccc} & Fox & Goose & Beans & Farmer \\ \hline Fox & 0 & 1 & 0 & 0 \\ Goose & 0 & 0 & 1 & 0 \\ Beans & 0 & 0 & 0 & 0 \\ Farmer & 0 & 0 & 0 & 0 \end{array} \quad (5.75)$$

Relation $where : Being \rightarrow Bank$ is necessarily a function because:

- everyone is somewhere in a bank (*where* is entire)
- no one can be in both banks at the same time (*where* is simple)

Note that there are only two constant functions of type $Being \rightarrow Bank$, \underline{Right} and \underline{Left} . The puzzle consists in changing from the state $where = \underline{Right}$ to the state $where = \underline{Left}$, for instance, without violating the property that *nobody eats anybody*. How does one record such a property? We need two auxiliary relations capturing, respectively:

- Being at the same bank:

$$SameBank = \ker where$$

- Risk of somebody eating somebody else:

$$CanEat = SameBank \cap Eats$$

Then “starvation” is ensured by the *Farmer’s* presence at the same bank:

$$CanEat \subseteq SameBank \cdot \underline{Farmer} \tag{5.76}$$

By (5.46), this “starvation” property (5.76) converts to:

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

In this version, (5.76) can be depicted as a diagram

$$\begin{array}{ccc}
 Being & \xleftarrow{CanEat} & Being \\
 where \downarrow & \subseteq & \downarrow \underline{Farmer} \\
 Bank & \xleftarrow{where} & Being
 \end{array} \tag{5.77}$$

which “reads” in a nice way:

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

Diagram (5.27) given earlier can now be identified as another example of assertion expressed relationally. Diagrams of this kind capture properties of data models that one wishes to hold at any time during the lifetime of the system being described. Such properties are commonly referred to as *invariants* and their preservation by calculation will be the main aim of chapter 7.

Exercise 5.19. Calculate the following pointwise version of the “starvation” property (5.77) by introducing quantifiers and simplifying:

$$\langle \forall b', b : b' Eat b : where b' = where b \Rightarrow where b' = where Farmer \rangle$$

□

Exercise 5.20. Recalling property (5.39), show that the “starvation” property (5.77) is satisfied by any of the two constant functions that model the start or end states of the Alcuin puzzle.

□

5.11 MONOTONICITY

As expected, relational composition is monotonic:

$$\frac{R \subseteq S \quad T \subseteq U}{(R \cdot T) \subseteq (S \cdot U)} \quad (5.78)$$

Indeed, all relational combinators studied so far are also monotonic, namely

$$R \subseteq S \Rightarrow R^\circ \subseteq S^\circ \quad (5.79)$$

$$R \subseteq S \wedge U \subseteq V \Rightarrow R \cap U \subseteq S \cap V \quad (5.80)$$

$$R \subseteq S \wedge U \subseteq V \Rightarrow R \cup U \subseteq S \cup V \quad (5.81)$$

hold.

Monotonicity and transitivity (5.22) are important properties for reasoning about a given relational inclusion $R \subseteq S$. In particular, the following rules are of help by relying on a “mid-point” relation M , $R \subseteq M \subseteq S$ (analogy with interval arithmetics).

- Rule A — *lowering the upper side*:

$$\begin{array}{c} R \subseteq S \\ \Leftarrow \{ M \subseteq S \text{ is known ; transitivity of } \subseteq \text{ (5.22) } \} \\ R \subseteq M \end{array}$$

Then proceed with $R \subseteq M$.

- Rule B — *raising the lower side*:

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

Then proceed with $M \subseteq S$.

The following proof of shunting property (5.46) combines these rules with monotonicity and circular implication:

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

Thus the equivalence in (5.46) is established by circular implication.

Rules A and B should be used only where other proof techniques (notably indirect equality) fail. They assume judicious choice of the mid-point relation M , at each step. The choice of an useless M can drive the proof nowhere.

Exercise 5.21. Unconditional distribution laws

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

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

□

Exercise 5.22. Prove that relational composition preserves all relational classes in the taxonomy of figure 5.2.

□

5.12 RULES OF THUMB

Quite often, involved reasoning in logic arguments can be replaced by simple and elegant calculations in relation algebra that arise thanks to smart “rules of thumb”. We have already seen two such rules, (5.43) and (5.44). Two others are:

$$- \text{smaller than injective (simple) is injective (simple)} \quad (5.82)$$

$$- \text{larger than entire (surjective) is entire (surjective)} \quad (5.83)$$

Let us see these rules in action in trying to infer what can be said of two functions f and r such that

$$f \cdot r = id$$

holds. On the one hand,

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

Since f is simple, f° is injective and so is r because “smaller than injective is injective”. On the other hand,

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

Since r is entire, r° is surjective and so is f because “larger than surjective is surjective”. We conclude that f is surjective and r is injective wherever $f \cdot r = id$ holds. Since both are functions, we furthermore conclude that

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

— cf. Figure 5.2.

The reason for this terminology can now be explained. Given $f : A \leftarrow C$ and $g : C \leftarrow A$ such that $f \cdot g = id$, that is, for all $a \in A$, $f (g a) = a$, think of C as a domain of *concrete* objects and of A as a domain of *abstract* data. For instance, let $A = \mathbb{B}$ and $C = \mathbb{N}_0$. Then define

$$\begin{cases} r : \mathbb{B} \rightarrow \mathbb{N}_0 \\ r b = \mathbf{if } b \mathbf{ then } k \mathbf{ else } 0 \end{cases}$$

(where k is any natural number different from 0) and

$$\begin{cases} f : \mathbb{B} \leftarrow \mathbb{N}_0 \\ f n = \mathbf{if } n = 0 \mathbf{ then } \mathbf{False} \mathbf{ else } \mathbf{True} \end{cases}$$

Clearly, by the definitions of f and r :

$$\begin{aligned} & f (r b) = \mathbf{if } (\mathbf{if } b \mathbf{ then } k \mathbf{ else } 0) = 0 \mathbf{ then } \mathbf{False} \mathbf{ else } \mathbf{True} \\ \equiv & \quad \{ \text{conditional-fusion rule (2.71)} \} \\ & f (r b) = \mathbf{if } (\mathbf{if } b \mathbf{ then } k = 0 \mathbf{ else } \mathbf{True}) \mathbf{ then } \mathbf{False} \mathbf{ else } \mathbf{True} \end{aligned}$$

$$\begin{aligned}
&\equiv \{ k = 0 \text{ is always false} \} \\
&f(r\ b) = \mathbf{if\ (if\ } b \mathbf{\ then\ False\ else\ True)\ then\ False\ else\ True} \\
&\equiv \{ \text{pointwise definition of } \neg b \} \\
&f(r\ b) = \mathbf{if\ } \neg b \mathbf{\ then\ False\ else\ True} \\
&\equiv \{ \text{trivial} \} \\
&b
\end{aligned}$$

That is, r represents the Booleans True and False by natural numbers while f abstracts from such real numbers back to Booleans. r being injective means $r\ \text{False} \neq r\ \text{True}$, that is, the Boolean information is not lost in the representation.¹² f being surjective means that any Boolean is representable. Note that $r \cdot f = id$ does not hold: $r\ (f\ 1) = r\ \text{True} = k$ and $k \neq 1$ in general.

The abstraction/representation pair (f, r) just above underlies the way Booleans are handled in programming languages such as C, for instance. Experienced programmers will surely agree that often what is going on in the code they write are processes of representing information using primitive data structures available from the adopted programming language. For instance, representing finite sets by finite lists corresponds to the *abstraction* given by `elems` (5.1).

Exercise 5.23. Recalling exercise 5.17, complete the definition of

$$\begin{aligned}
\mathit{bag}\ []\ a &= 0 \\
\mathit{bag}\ (h:t)\ a &= \mathbf{let\ } b = \mathit{bag}\ t \mathbf{\ in\ if\ } \dots
\end{aligned}$$

Is this function an abstraction or a representation? Justify your answer informally.

□

Exercise 5.24. Show that:

- $R \cap S$ 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.

□

5.13 ENDO-RELATIONS

Relations in general are of type $A \rightarrow B$, for some A and B . In the special case that $A = B$ holds, a relation $R : A \rightarrow A$ is said to be an *endo-relation*, or a *graph*. The $A = B$ coincidence gives room for some

¹² That is, r causes *no confusion* in the representation process.

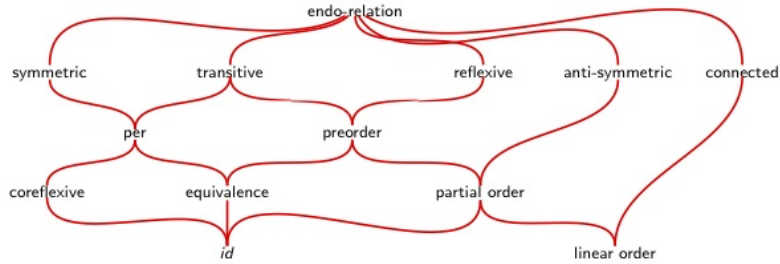


Figure 5.4.: Taxonomy of endorelations.

extra terminology, extending some already given. Besides an endo-relation $A \xleftarrow{R} A$ being

$$\text{reflexive:} \quad \text{iff } id \subseteq R \quad (5.84)$$

$$\text{coreflexive:} \quad \text{iff } R \subseteq id \quad (5.85)$$

it can also be:

$$\text{transitive:} \quad \text{iff } R \cdot R \subseteq R \quad (5.86)$$

$$\text{symmetric:} \quad \text{iff } R \subseteq R^\circ (\equiv R = R^\circ) \quad (5.87)$$

$$\text{anti-symmetric:} \quad \text{iff } R \cap R^\circ \subseteq id \quad (5.88)$$

$$\text{irreflexive:} \quad \text{iff } R \cap id = \perp \quad (5.89)$$

$$\text{connected:} \quad \text{iff } R \cup R^\circ = \top \quad (5.90)$$

By combining these criteria, endo-relations $A \xleftarrow{R} A$ can further be classified as in figure 5.4. In summary:

- *Preorders* are reflexive and transitive orders.
Example: $age\ y \leq age\ x$.
- *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}_0
- *Equivalences* are symmetric preorders
Example: $age\ y = age\ x$.¹³
- *Pers* are partial equivalences
Example: $y\ IsBrotherOf\ x$.

Preorders are normally denoted by asymmetric symbols such as e.g. $y \sqsubseteq x, y \leq x$. In case of a function f such that

$$f \cdot (\sqsubseteq) \subseteq (\leq) \cdot f \quad (5.91)$$

¹³ Kernels of functions are always equivalence relations, see exercise 5.25.

we say that f is monotonic. Indeed, this is equivalent to

$$a \sqsubseteq b \Rightarrow (f a) \leq (f b)$$

once shunting (5.46) takes place, and variables are added and handled via (5.17). Another frequent situation is that of two functions f and g such that

$$f \subseteq (\leq) \cdot g \quad (5.92)$$

This converts to the pointwise

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

that is, f is *always at most* g for all possible inputs. The following abbreviation is often used to capture this ordering on functions induced by a pre-order (\leq) on their outputs:

$$f \dot{\leq} g \text{ iff } f \subseteq (\leq) \cdot g \quad (5.93)$$

For instance, $f \dot{\leq} id$ means $f a \leq a$ for all inputs a .

CLOSURE OPERATORS Given a partial order (\leq), a function f is said to be a *closure operator* iff

$$(\leq) \cdot f = f^\circ \cdot (\leq) \cdot f \quad (5.94)$$

holds. The same with points — via (5.17) —, for all x, y :

$$y \leq f x \Leftrightarrow f x \leq f y \quad (5.95)$$

Clearly, for $(\geq) = (\leq)^\circ$, (5.94) can also be written

$$f^\circ \cdot (\geq) = f^\circ \cdot (\geq) \cdot f$$

Any of these alternatives is an elegant way of defining a closure operator f , in so far it can be shown to be equivalent to the conjunction of three facts about f : (a) f is monotonic; (b) $id \dot{\leq} f$ and (c) $f = f \cdot f$.

As an example, consider the function that *closes* a finite set of natural numbers by filling in the intermediate numbers, e.g. $f \{4, 2, 6\} = \{2, 3, 4, 5, 6\}$. Clearly, $x \subseteq f x$. If you apply f again, you get

$$f \{2, 3, 4, 5, 6\} = \{2, 3, 4, 5, 6\}$$

This happens because f is a closure operator.

Exercise 5.25. Knowing that property

$$f \cdot f^\circ \cdot f = f \quad (5.96)$$

holds for every function f , prove that $\ker f = \frac{f}{f}$ (5.53) is an equivalence relation.

□

Exercise 5.26. From $\ker \neq \top$ and (5.96) infer

$$\top \cdot R \subseteq \top \cdot S \iff R \subseteq \top \cdot S \quad (5.97)$$

Conclude that $(\top \cdot)$ is a closure operator.

□

Exercise 5.27. Generalizing the previous exercise, show that pre/post-composition with functional kernels are closure operations:

$$S \cdot \ker f \subseteq R \cdot \ker f \iff S \subseteq R \cdot \ker f \quad (5.98)$$

$$\ker f \cdot S \subseteq \ker f \cdot R \iff S \subseteq \ker f \cdot R \quad (5.99)$$

□

Exercise 5.28. Consider the relation

$$b R a \iff \text{team } b \text{ is playing against team } a$$

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

□

Exercise 5.29. Expand criteria (5.86) to (5.90) to pointwise notation.

□

Exercise 5.30. A relation R is said to be co-transitive or dense iff the following holds:

$$\langle \forall b, a : b R a : \langle \exists c : b R c : c R a \rangle \rangle \quad (5.100)$$

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

□

Exercise 5.31. Check which of the following properties,

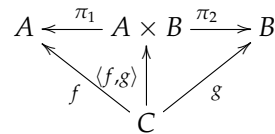
transitive, symmetric, anti-symmetric, connected

hold for the relation *Eats* (5.75) of the Alcuin puzzle.
 □

Exercise 5.32. Show that (5.55) of exercise 5.12 amounts to forcing relation home-away^o to be symmetric.
 □

5.14 RELATIONAL PAIRING

Recall from sections 2.8 and 2.9 that functions can be composed in parallel and in alternation, giving rise to so-called *products* and *coproducts*. Does a diagram like (2.23),



make sense when f e g are generalized to relations R and S ? We start from definition (2.20),

$$\langle f, g \rangle c \stackrel{\text{def}}{=} (f c, g c)$$

and try to see what such a generalization could mean. The relational expression of function $\langle f, g \rangle$ is $y = \langle f, g \rangle c$, which can be rephrased to $(a, b) = \langle f, g \rangle c$ knowing that $\langle f, g \rangle$ is of type $C \rightarrow A \times B$ in (2.23). We reason:

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

By in-lining the conjunction expressed by the braces just above, one gets

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

which proposes the generalization:

$$(a, b) \langle R, S \rangle c \Leftrightarrow a R c \wedge b S c \tag{5.101}$$

Recalling the projections $\pi_1 (a, b) = a$ and $\pi_2 (a, b) = b$, we shall try and remove variables a, b and c from the above, towards a closed definition of $\langle R, S \rangle$:

$$\begin{aligned}
& (a, b) \langle R, S \rangle c \Leftrightarrow a R c \wedge b S c \\
\equiv & \quad \{ \pi_1 (a, b) = a \text{ and } \pi_2 (a, b) = b \} \\
& (a, b) \langle R, S \rangle c \Leftrightarrow \pi_1 (a, b) R c \wedge \pi_2 (a, b) S c \\
\equiv & \quad \{ (5.17) \text{ twice} \} \\
& (a, b) \langle R, S \rangle c \Leftrightarrow (a, b) (\pi_1^\circ \cdot R) c \wedge (a, b) (\pi_2^\circ \cdot S) c \\
\equiv & \quad \{ (5.56) \} \\
& (a, b) \langle R, S \rangle c \Leftrightarrow (a, b) (\pi_1^\circ \cdot R \cap \pi_2^\circ \cdot S) c \\
\equiv & \quad \{ (5.19) \} \\
\langle R, S \rangle &= \pi_1^\circ \cdot R \cap \pi_2^\circ \cdot S \tag{5.102}
\end{aligned}$$

Next, we investigate which kind of universal property $\langle R, S \rangle$ defined by $\pi_1^\circ \cdot R \cap \pi_2^\circ \cdot S$ satisfies. The strategy is to use indirect equality:

$$\begin{aligned}
& X \subseteq \langle R, S \rangle \\
\equiv & \quad \{ (5.102) \} \\
& X \subseteq \pi_1^\circ \cdot R \cap \pi_2^\circ \cdot S \\
\equiv & \quad \{ (5.58) \} \\
& \begin{cases} X \subseteq \pi_1^\circ \cdot R \\ X \subseteq \pi_2^\circ \cdot S \end{cases} \\
\equiv & \quad \{ \text{shunting} \} \\
& \begin{cases} \pi_1 \cdot X \subseteq R \\ \pi_2 \cdot X \subseteq S \end{cases}
\end{aligned}$$

In summary, the universal property of $\langle R, S \rangle$ is:

$$X \subseteq \langle R, S \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot X \subseteq R \\ \pi_2 \cdot X \subseteq S \end{cases} \tag{5.103}$$

For functions, $X, R, S := k, f, g$ it can be observed that (5.103) coincides with (2.63). But otherwise, the corollaries derived from (5.103) are different from those that emerge from (2.63). For instance, cancellation becomes:

$$\begin{cases} \pi_1 \cdot \langle R, S \rangle \subseteq R \\ \pi_2 \cdot \langle R, S \rangle \subseteq S \end{cases}$$

This tells us that pairing R with S has the (side) effect of deleting from R all those inputs for which S is undefined (and vice-versa), since output pairs require that *both* relations respond to the input. Thus, for relations, laws such as the \times -fusion rule (2.26) call for a side-condition:

$$\begin{aligned}
\langle R, S \rangle \cdot T &= \langle R \cdot T, S \cdot T \rangle \\
&\Leftarrow R \cdot (\text{img } T) \subseteq R \vee S \cdot (\text{img } T) \subseteq S \tag{5.104}
\end{aligned}$$

Clearly,

$$\langle R, S \rangle \cdot f = \langle R \cdot f, S \cdot f \rangle \quad (5.105)$$

holds, since $\text{img } f \subseteq \text{id}$. Moreover, the *absorption* law (2.27) remains unchanged,

$$(R \times S) \cdot \langle P, Q \rangle = \langle R \cdot P, S \cdot Q \rangle \quad (5.106)$$

where $R \times S$ is defined in the same way as for functions:

$$R \times S = \langle R \cdot \pi_1, S \cdot \pi_2 \rangle \quad (5.107)$$

As generalization of (5.105) and also immediate by monotonicity, $\langle R, S \rangle \cdot T = \langle R \cdot T, S \cdot T \rangle$ holds for T simple.

Because (5.103) is not the universal property of a product, we tend to avoid talking about relational *products* and talk about relational *pairing* instead.¹⁴ In spite of the weaker properties, relational pairing has interesting laws, namely:

$$\langle R, S \rangle^\circ \cdot \langle X, Y \rangle = (R^\circ \cdot X) \cap (S^\circ \cdot Y) \quad (5.108)$$

Exercise 5.33. Derive from (5.108) the following properties:

$$\frac{f}{g} \cap \frac{h}{k} = \frac{f \vee h}{g \vee k} \quad (5.109)$$

$$(5.110)$$

$$\ker \langle R, S \rangle = \ker R \cap \ker S \quad (5.111)$$

$\langle R, \text{id} \rangle$ is always injective, for whatever R

□

Exercise 5.34. Recalling (5.38), prove that $\text{swap} = \langle \pi_2, \pi_1 \rangle$ (2.32) is its own converse and therefore a bijection.

□

Exercise 5.35. Derive from the laws studied thus far the following facts about relational pairing:

$$\text{id} \times \text{id} = \text{id} \quad (5.112)$$

$$(R \times S) \cdot (P \times Q) = (R \cdot P) \times (S \cdot Q) \quad (5.113)$$

□

¹⁴ Relational products do exist but are not obtained by $\langle R, S \rangle$. For more about this see section 5.23 later on.

5.15 RELATIONAL COPRODUCTS

Let us now show that, in contrast with products, coproducts extend perfectly from functions to relations, that is, universal property (2.65) extends to

$$X = [R, S] \Leftrightarrow \begin{cases} X \cdot i_1 = R \\ X \cdot i_2 = S \end{cases} \quad (5.114)$$

where $X : A + B \rightarrow C$, $R : A \rightarrow C$ and $S : B \rightarrow C$ are binary relations. First of all, we need to understand what $[R, S]$ means. Our starting point is $+$ -cancellation, recall (2.40):

$$\begin{aligned} & \begin{cases} [g, h] \cdot i_1 = g \\ [g, h] \cdot i_2 = h \end{cases} \\ \equiv & \quad \{ \text{equality of functions} \} \\ & \begin{cases} g \subseteq [g, h] \cdot i_1 \\ h \subseteq [g, h] \cdot i_2 \end{cases} \\ \equiv & \quad \{ \text{shunting followed by (5.57)} \} \\ & g \cdot i_1^\circ \cup h \cdot i_2^\circ \subseteq [g, h] \end{aligned}$$

On the other hand:

$$\begin{aligned} & \begin{cases} [g, h] \cdot i_1 = g \\ [g, h] \cdot i_2 = h \end{cases} \\ \equiv & \quad \{ \text{equality of functions} \} \\ & \begin{cases} [g, h] \cdot i_1 \subseteq g \\ [g, h] \cdot i_2 \subseteq h \end{cases} \\ \Rightarrow & \quad \{ \text{monotonicity} \} \\ & \begin{cases} [g, h] \cdot i_1 \cdot i_1^\circ \subseteq g \cdot i_1^\circ \\ [g, h] \cdot i_2 \cdot i_2^\circ \subseteq h \cdot i_2^\circ \end{cases} \\ \Rightarrow & \quad \{ \text{monotonicity (5.81) and distribution (5.60)} \} \\ & [g, h] \cdot (i_1 \cdot i_1^\circ \cup i_2 \cdot i_2^\circ) \subseteq g \cdot i_1^\circ \cup h \cdot i_2^\circ \\ \equiv & \quad \{ \text{img } i_1 \cup \text{img } i_2 = id, \text{ more about this below} \} \\ & [g, h] \subseteq g \cdot i_1^\circ \cup h \cdot i_2^\circ \end{aligned}$$

Altogether, we obtain:

$$[g, h] = g \cdot i_1^\circ \cup h \cdot i_2^\circ$$

Note how this matches with (2.37), once variables are introduced:

$$c [g, h] x \Leftrightarrow \langle \exists a : x = i_1 a : c = g a \rangle \vee \langle \exists b : x = i_2 b : c = h b \rangle$$

Fact

$$\text{img } i_1 \cup \text{img } i_2 = id \quad (5.115)$$

assumed above is a property stemming from the construction of coproducts,

$$A + B \stackrel{\text{def}}{=} \{i_1 a \mid a \in A\} \cup \{i_2 b \mid b \in B\}$$

since i_1 and i_2 are the *only* constructors of data of type $A + B$. Another property implicit in this construction is:

$$i_1^\circ \cdot i_2 = \perp \quad (5.116)$$

equivalent to its converse $i_2^\circ \cdot i_1 = \perp$. It spells out that, for any $a \in A$ and $b \in B$, $i_1 a = i_2 b$ is impossible.¹⁵ In other words, the union is a *disjoint* one.

Let us now generalize the above to relations instead of functions,

$$[R, S] = R \cdot i_1^\circ \cup S \cdot i_2^\circ \quad (5.117)$$

and show that (5.114) holds. First of all,

$$\begin{aligned} X &= R \cdot i_1^\circ \cup S \cdot i_2^\circ \\ \Rightarrow & \quad \{ \text{compose both sides with } i_1 \text{ and simplify; similarly for } i_2 \} \\ X \cdot i_1 &= R \wedge X \cdot i_2 = S \end{aligned}$$

The simplifications arise from i_1 and i_2 being injections, so their kernels are identities. On the other hand, $i_1^\circ \cdot i_2 = \perp$ and $i_2^\circ \cdot i_1 = \perp$, as seen above. The converse implication (\Leftarrow) holds:

$$\begin{aligned} X &= R \cdot i_1^\circ \cup S \cdot i_2^\circ \\ \equiv & \quad \{ (5.115) \} \\ X \cdot (\text{img } i_1 \cup \text{img } i_2) &= R \cdot i_1^\circ \cup S \cdot i_2^\circ \\ \equiv & \quad \{ \text{distribution} \} \\ X \cdot \text{img } i_1 \cup X \cdot \text{img } i_2 &= R \cdot i_1^\circ \cup S \cdot i_2^\circ \\ \Leftarrow & \quad \{ \text{Leibniz} \} \\ X \cdot i_1 \cdot i_1^\circ &= R \cdot i_1^\circ \wedge X \cdot i_2 \cdot i_2^\circ = S \cdot i_2^\circ \\ \Leftarrow & \quad \{ \text{monotonicity} \} \\ X \cdot i_1 &= R \wedge X \cdot i_2 = S \end{aligned}$$

□

Thus (5.114) holds in general, for relations:

$$(B + C) \rightarrow A \begin{array}{c} \xrightarrow{[-, -]^\circ} \\ \cong \\ \xrightarrow{[-, -]} \end{array} (B \rightarrow A) \times (C \rightarrow A) \quad (5.118)$$

¹⁵ Note that in (2.36) this is ensured by always choosing two different tags $t_1 \neq t_2$.

A most useful consequence of this is that all results known for coproducts of functions are valid for relational coproducts. In particular, relational direct sum

$$R + S = [i_1 \cdot R, i_2 \cdot S] \tag{5.119}$$

can be defined satisfying (2.43), (2.44) etc with relations replacing functions. Moreover, the McCarthy conditional (2.70) can be extended to relations in the expected way:

$$p \rightarrow R, S \stackrel{\text{def}}{=} [R, S] \cdot p? \tag{5.120}$$

The property for sums (coproducts) corresponding to (5.108) for products is:

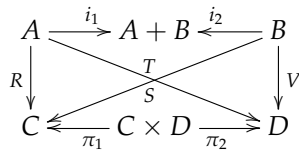
$$[R, S] \cdot [T, U]^\circ = (R \cdot T^\circ) \cup (S \cdot U^\circ) \tag{5.121}$$

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

Finally, the *exchange law* (2.49) extends to relations,

$$\langle [R, S], [T, V] \rangle = \langle [R, T], [S, V] \rangle \tag{5.122}$$

cf.



For the proof see the following exercise.

Exercise 5.36. Relying on both (5.114) and (5.105) prove (5.122). Moreover, prove

$$(R + S)^\circ = R^\circ + S^\circ \tag{5.123}$$

□

Exercise 5.37. From (5.117) prove (5.121). Then show that

$$\text{img } [R, S] = \text{img } R \cup \text{img } S \tag{5.124}$$

follows immediately from (5.121).

□

Exercise 5.38. Prove that the coproduct $[R, S]$ is injective iff both R, S are injective and $R^\circ \cdot S = \perp$.

□

Exercise 5.39. Prove:

$$\frac{f}{g} \times \frac{h}{k} = \frac{f \times h}{g \times k} \tag{5.125}$$

$$\frac{f}{g} + \frac{h}{k} = \frac{f + h}{g + k} \tag{5.126}$$

□

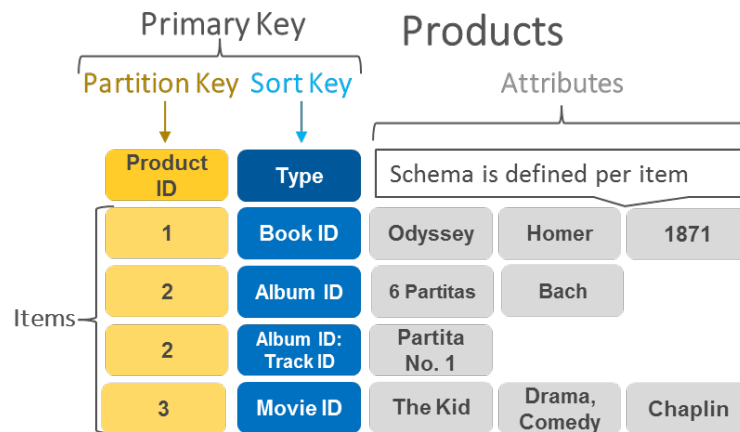
5.16 ON KEY-VALUE DATA MODELS

Simple relations abstract what is currently known as the *key-value-pair* data model in modern databases.¹⁶ In this setting, given a *simple* relation $K \xrightarrow{S} V$, K is regarded as a type of data *keys* and V as a type of data *values*.

By pairing (5.102) such key-value-pairs one obtains more elaborate stores. Conversely, one may use projections to select particular key-attribute relationships from key-value stores. Note that keys and values can be *anything* (that is, of any type) and, in particular, they can be compound, for instance

$$\underbrace{PartitionKey \times SortKey}_K \rightarrow \underbrace{Type \times \dots}_V$$

in the following example:¹⁷



¹⁶ For example, Hbase, Amazon DynamoDB, and so on, are examples of database systems that use the key-value pair data model.

¹⁷ Credits: <https://aws.amazon.com/nosql/key-value/>.

The example furthermore shows how keys and values can structure themselves even further. In particular, “*schema is defined per item*” indicates that the values may be of coproduct types, something like $Title \times (1 + Author \times (1 + Date \times \dots))$, for instance. Although the simplicity of the columnar model suggested by the key-value principle is somewhat sacrificed in the example, this shows how expressive *simple* relations involving *product* and *coproduct* types are.

One of the standard variations of the key-value model is to equip keys with time-stamps indicating *when* the pair was inserted or modified in the store, for instance

$$Student \times Course \times Time \rightarrow Result \tag{5.127}$$

telling the possibly different results of students in exams of a particular course. This combination of the key-value model with that of *temporal* (also called *historical*) databases is very powerful.

The relational combinators studied in this book apply naturally to key-value-pair storage processing and offer themselves as a powerful, pointfree high-level language for handling such data in a “noSQL” style.

5.17 WHAT ABOUT RELATIONAL “CURRYING”?

Recall isomorphism (2.93),

$$\begin{array}{ccc}
 & \xrightarrow{\text{uncurry}} & \\
 (C^B)^A & \cong & C^{A \times B} \\
 & \xleftarrow{\text{curry}} &
 \end{array}$$

that is at the core of the way binary functions are handled in functional programming. Does this isomorphism hold when functions are generalized to relations, something like...

$$A \times B \rightarrow C \cong A \rightarrow \dots?$$

Knowing that the type $A \times B \rightarrow C$ of relations is far larger than $C^{A \times B}$, it can be anticipated that the isomorphism will not extend to relations in the same way. In fact, a rather simpler one happens instead, among relations:

$$\begin{array}{ccc}
 & \xrightarrow{\text{trans}} & \\
 A \times B \rightarrow C & \cong & A \rightarrow C \times B \\
 & \xleftarrow{\text{untrans}} &
 \end{array} \tag{5.128}$$

This tells us that (obvious, but very useful fact) that relations involving product types can be reshaped in any way we like, leftwards or rightwards.

It is quite convenient to overload the notation used for functions and write \bar{R} to denote *trans* R and \hat{R} to denote *untrans* R . Then the isomorphism above is captured by universal property,¹⁸

$$\begin{array}{ccc}
 C \times B & & (C \times B) \times B \xrightarrow{\epsilon} C \\
 \bar{R} \uparrow & & \nearrow R \\
 A & & A \times B \xrightarrow{\bar{R} \times id} (C \times B) \times B
 \end{array}$$

where

$$\bar{R} = \langle R, \pi_2 \rangle \cdot \pi_1^\circ \quad \begin{array}{ccc} C \times B & & \\ \bar{R} \uparrow & \langle R, \pi_2 \rangle \swarrow & \\ A & \xrightarrow{\pi_1^\circ} & A \times B \end{array} \quad (5.129)$$

that is

$$(c, b) \bar{R} a \equiv c R (a, b)$$

Moral: every n -ary relation can be expressed as a binary relation; moreover, where each particular attribute is placed (input/output) is irrelevant.

By *converse duality*, $(\hat{S})^\circ = \overline{(S^\circ)}$, we obtain the definition of relational "uncurrying":

$$\hat{S} = \pi_1 \cdot \langle S^\circ, \pi_2 \rangle^\circ$$

Then

$$\epsilon = \hat{id} = \pi_1 \cdot \langle id, \pi_2 \rangle^\circ.$$

With points:

$$c_2 \in ((c_1, b_1), b_2) \equiv c_2 = c_1 \wedge b_1 = b_2$$

THE "PAIRING WHEEL" RULE The flexibility offered by (5.128) means that, in relation algebra, the information altogether captured by the three relations M, P and Q in

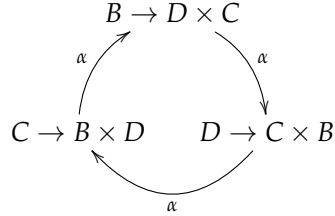
$$\begin{array}{ccc}
 & B & \\
 & \uparrow M & \\
 & A & \\
 Q \swarrow & & \searrow P \\
 C & & D
 \end{array} \quad (5.130)$$

can be aggregated in several ways, namely

¹⁸ Compare with (2.84).

$$\begin{aligned}
 B &\xrightarrow{\langle P, Q \rangle \cdot M^\circ} D \times C \\
 D &\xrightarrow{\langle Q, M \rangle \cdot P^\circ} C \times B \\
 C &\xrightarrow{\langle M, P \rangle \cdot Q^\circ} B \times C
 \end{aligned}$$

all isomorphic to each other:



The rotation among relations and types justifies the name “pairing wheel” given to (5.130). Isomorphism α holds in the sense that every entry of one of the aggregates is uniquely represented by another entry in any other aggregate, for instance:

$$\begin{aligned}
 &(d, c) (\langle P, Q \rangle \cdot M^\circ) b \\
 = &\quad \{ \text{composition ; pairing} \} \\
 &\langle \exists a : d P a \wedge c Q a : a M^\circ b \rangle \\
 = &\quad \{ \text{converse; } \wedge \text{ is associative and commutative} \} \\
 &\langle \exists a :: (c Q a \wedge b M a) \wedge a P^\circ d \rangle \\
 = &\quad \{ \text{composition ; pairing} \} \\
 &(c, b) (\langle Q, M \rangle \cdot P^\circ) d
 \end{aligned}$$

Thus: $\alpha (\langle P, Q \rangle \cdot M^\circ) = (\langle Q, M \rangle \cdot P^\circ)$.

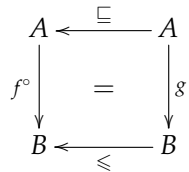
Exercise 5.40. Express α in terms of trans (5.128) and its converse (5.129).
□

5.18 GALOIS CONNECTIONS

Recall from section 5.13 that a preorder is a reflexive and transitive relation. Given two preorders \leq and \sqsubseteq , one may relate arguments and results of pairs of suitably typed functions f and g in a particular way,

$$f^\circ \cdot \sqsubseteq = \leq \cdot g \tag{5.131}$$

as in the diagram:



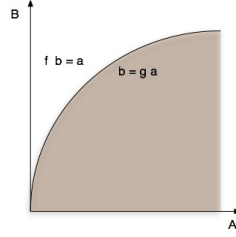
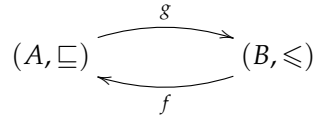


Figure 5.5.: Graphical interpretation of equation (5.131): (a) relation $B \xleftarrow{(\leq) \cdot g} A$ is the “area” below function g wrt. \leq ; (b) relation $B \xrightarrow{f \cdot (\sqsubseteq)} A$ is the “area” above function f wrt. \sqsubseteq , to the right (oriented 90°); (c) f and g are such that these areas are the same.

In this very special situation, f, g are said to be *Galois connected*. We write

$$f \vdash g \tag{5.132}$$

as abbreviation of (5.131) when the two preorders \sqsubseteq, \leq are implicit from the context. Another way to represent this is:



Function f (resp. g) is referred to as the *lower* (resp. *upper*) adjoint of the connection. By introducing variables in both sides of (5.131) via (5.17), we obtain, for all x and y

$$(f x) \sqsubseteq y \quad \equiv \quad x \leq (g y) \tag{5.133}$$

In particular, the two preorders in (5.131) can be the identity id , in which case (5.131) reduces to $f^\circ = g$, that is, f and g are each-other inverses — i.e., isomorphisms. Therefore, the Galois connection concept is a generalization of the concept of isomorphism.

Quite often, the two adjoints are *sections* of binary operators. Recall that, given a binary operator $a \theta b$, its two sections $(a\theta)$ and (θb) are unary functions f and g such that, respectively:

$$f = (a\theta) \quad \equiv \quad f b = a \theta b \tag{5.134}$$

$$g = (\theta b) \quad \equiv \quad g a = a \theta b \tag{5.135}$$

Galois connections in which the two preorders are relation inclusion ($\leq, \sqsubseteq := \subseteq, \subseteq$) and whose adjoints are sections of relational combinators are particularly interesting because they express universal properties about such combinators. Table 3 lists some connections that are relevant for this book.

$(f X) \subseteq Y \equiv X \subseteq (g Y)$			
Description	f	g	Obs.
converse	$(-)^{\circ}$	$(-)^{\circ}$	
<i>shunting rule</i>	$(h \cdot)$	$(h^{\circ} \cdot)$	h is a function
<i>“converse” shunting rule</i>	$(\cdot h^{\circ})$	$(\cdot h)$	h is a function
difference	$(- - R)$	$(R \cup)$	
implication	$(R \cap -)$	$(R \Rightarrow -)$	

Table 3.: Sample of Galois connections in the relational calculus. The general formula given on top is a logical equivalence universally quantified on S and R . It has a left part involving *lower adjoint* f and a right part involving *upper adjoint* g .

It is remarkably easy to recover known properties of the relation calculus from table 3. For instance, the first row yields

$$X^{\circ} \subseteq Y \equiv X \subseteq Y^{\circ} \quad (5.136)$$

since $f = g = (-)^{\circ}$ in this case. Thus converse is its own self adjoint. From this we derive

$$R \subseteq S \equiv R^{\circ} \subseteq S^{\circ} \quad (5.137)$$

by making $X, Y := R, S^{\circ}$ and simplifying by involution (5.15). Moreover, the entry marked *“shunting rule”* in the table leads to

$$h \cdot X \subseteq Y \equiv X \subseteq h^{\circ} \cdot Y$$

for all h, X and Y . By taking converses, one gets another entry in table 3, namely

$$X \cdot h^{\circ} \subseteq Y \equiv X \subseteq Y \cdot h$$

These are the equivalences (5.46) and (5.47) that we have already met, popularly known as *“shunting rules”*.

The fourth and fifth rows in the table are Galois connections that respectively introduce two new relational operators — relational *difference* $S - R$ and relational *implication* $R \Rightarrow S$ — as a *lower adjoint* and an *upper adjoint*, respectively:

$$X - R \subseteq Y \equiv X \subseteq Y \cup R \quad (5.138)$$

$$R \cap X \subseteq Y \equiv X \subseteq R \Rightarrow Y \quad (5.139)$$

There are *many* advantages in describing the meaning of relational operators by Galois connections. Further to the systematic tabulation of operators (of which table 3 is just a sample), the concept of a Galois connection is a *generic* one, which offers a rich algebra of *generic* properties, namely:

- both adjoints f and g in a Galois connection are monotonic;

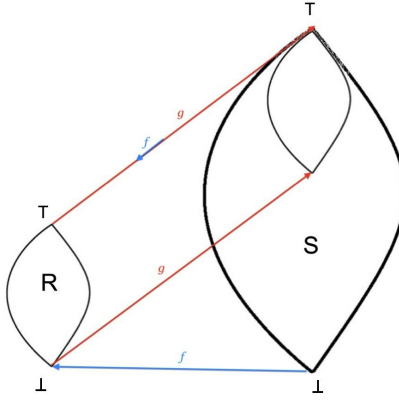


Figure 5.6.: Lower-perfect Galois connection $f \dashv g$ involving two lattices S and R .

- lower adjoint f distributes with join and upper-adjoint g distributes with meet, wherever these exist:

$$f(b \sqcup b') = (f b) \vee (f b') \tag{5.140}$$

$$g(a \wedge a') = (g a) \sqcap (g a') \tag{5.141}$$

- lower adjoint f preserves infima and upper-adjoint g preserves suprema, wherever these exist.¹⁹

$$f \perp = \perp \tag{5.142}$$

$$g \top = \top \tag{5.143}$$

- two cancellation laws hold,

$$(f \cdot g)a \leq a \quad \text{and} \quad b \sqsubseteq (g \cdot f)b \tag{5.144}$$

respectively known as *lower-cancellation* and *upper-cancellation*.

- Semi-inverse properties:

$$f = f \cdot g \cdot f \tag{5.145}$$

$$g = g \cdot f \cdot g \tag{5.146}$$

It may happen that a cancellation law holds up to equality, for instance $f(g a) = a$, in which case the connection is said to be *perfect* on the particular side. The picture of a lower-perfect Galois connection $f \dashv g$ is given in figure 5.6.²⁰

¹⁹ In these case both orders will form a so-called *lattice* structure.

²⁰ Adapted from [5].

Let us take for instance Galois connection (5.138) as example. Following the general rules above, we get *for free*: the monotonicity of $(- - R)$,

$$X \subseteq Z \Rightarrow X - R \subseteq Z - R$$

the monotonicity of $(- \cup R)$,

$$X \subseteq Z \Rightarrow X \cup R \subseteq Z \cup R$$

the distribution of $(- - R)$ over *join*,

$$(X \cup Y) - R = (X - R) \cup (Y - R)$$

the distribution of $(- \cup R)$ over *meet*,

$$(X \cap Y) \cup R = (X \cup R) \cap (Y \cup R)$$

the preservation of infima by $(- - R)$,

$$\perp - R = \perp$$

the preservation of suprema by $(- \cup R)$,

$$\top \cup R = \top$$

lower-cancellation ($Y := X - R$),

$$X \subseteq (X - R) \cup R$$

upper-cancellation ($X := Y \cup R$),

$$(Y \cup R - R) \subseteq Y$$

and finally the semi-inverse properties:

$$\begin{aligned} X - ((X - R) \cup R) &= X - R \\ ((X \cup R) - R) \cup R &= X \cup R \end{aligned}$$

The reader is invited to extract similar properties from the other connections listed in table 3. Altogether, we get 50 properties out of this table! Such is the power of *generic* concepts in mathematics.

Two such connections were deliberately left out from table 3, which play a central role in relation algebra and will deserve a section of their own — section 5.19.

Exercise 5.41. Show that $R - S \subseteq R$, $R - \perp = R$ and $R - R = \perp$ hold.

□

Exercise 5.42. Infer

$$b(R \Rightarrow S)a \equiv (b R a) \Rightarrow (b S a) \quad (5.147)$$

from the Galois connection

$$R \cap X \subseteq Y \quad \equiv \quad X \subseteq (R \Rightarrow Y) \quad (5.148)$$

Suggestion: note that $b (R \Rightarrow S) a$ can be written $id \subseteq \underline{b}^\circ \cdot (R \Rightarrow S) \cdot \underline{a}$ (check this!). Then proceed with (5.148) and simplify.

□

Exercise 5.43. (Lexicographic orders) The lexicographic chaining of two relations R and S is defined by:

$$R ; S = R \cap (R^\circ \Rightarrow S) \quad (5.149)$$

Show that (5.149) is the same as stating the universal property:

$$X \subseteq (R ; S) \equiv X \subseteq R \wedge X \cap R^\circ \subseteq S$$

□

Exercise 5.44. Let students in a course have two numeric marks,

$$\mathbb{N}_0 \xleftarrow{\text{mark1}} \text{Student} \xrightarrow{\text{mark2}} \mathbb{N}_0$$

and define the preorders:

$$\leq_{\text{mark1}} = \text{mark1}^\circ \cdot \leq \cdot \text{mark1}$$

$$\leq_{\text{mark2}} = \text{mark2}^\circ \cdot \leq \cdot \text{mark2}$$

Spell out in pointwise notation the meaning of lexicographic ordering

$$\leq_{\text{mark1}} ; \leq_{\text{mark2}}$$

□

NEGATION We define $\neg R = R \Rightarrow \perp$ since $b (\neg R) a \Leftrightarrow \neg (b R a)$. Clearly, $\neg \top = \perp$. It can also be shown that

$$R \cup \neg R = \top \quad (5.150)$$

holds and therefore:

$$\top - R \subseteq R \Rightarrow \perp \quad (5.151)$$

From the Galois connection of $R \Rightarrow S$ and through the usual rule of indirect equality, one immediately infers the so-called *de Morgan law*,

$$\neg(R \cup S) = (\neg R) \cap (\neg S) \quad (5.152)$$

and other expected properties analogous to logic negation. One of the most famous rules for handling negated relations is the so-called *Schröder's rule*:

$$\neg Q \cdot S^\circ \subseteq \neg R \Leftrightarrow R^\circ \cdot \neg Q \subseteq \neg S \quad (5.153)$$

Exercise 5.45. Assuming

$$f^\circ \cdot (R \Rightarrow S) \cdot g = (f^\circ \cdot R \cdot g) \Rightarrow (f^\circ \cdot S \cdot g) \tag{5.154}$$

and (5.151), prove:

$$\underline{c}^\circ \cdot (\top - \underline{c}) = \perp \tag{5.155}$$

□

5.19 RELATION DIVISION

However intimidating it may sound, structuring a calculus in terms of Galois connections turns out to be a great simplification, leading to *rules* that make the reasoning closer to school algebra. Think for instance the rule used at school to reason about whole division of two natural numbers x and y ,

$$z \times y \leq x \equiv z \leq x \div y \quad (y > 0) \tag{5.156}$$

assumed universally quantified in all its variables. Pragmatically, it expresses a “shunting” rule which enables one to trade between a whole division in the upper side of an inequality and a multiplication in the lower side. This rule is easily identified as the Galois connection

$$\underbrace{z(\times y)}_f \leq x \Leftrightarrow z \leq \underbrace{x(\div y)}_g.$$

where multiplication is the lower adjoint and division is the upper adjoint: $(\times y) \vdash (\div y)$, for $y \neq 0$.²¹

As seen in the previous section, many properties of (\times) and (\div) can be inferred from (5.156), for instance the cancellation $(x \div y) \times y \leq x$ — just replace z by $x \div y$ and simplify, and so on.

A parallel with relation algebra could be made by trying a rule similar to (5.156),

$$Z \cdot Y \subseteq X \equiv Z \subseteq X/Y \tag{5.157}$$

which suggests that, like integer multiplication, relational composition has an upper adjoint, denoted X/Y . The question is: does such a *relation division* operator actually exist? Proceeding with the parallel, note that, in the same way

$$z \times y \leq x \equiv z \leq x \div y$$

means that $x \div y$ is the largest *number* which multiplied by y approximates x , (5.157) means that X/Y is the largest *relation* Z which, pre-composed with Y , approximates X .

²¹ This connection is perfect on the lower side since $(z \times y) \div y = z$.

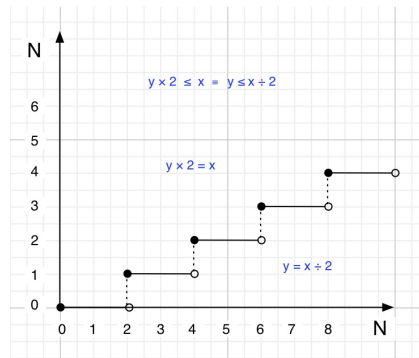
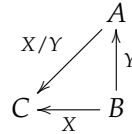


Figure 5.7.: Picturing Galois connection $(\times 2) \vdash (\div 2)$ as in figure 5.5. $f = (\times 2)$ is the lower adjoint of $g = (\div 2)$. The area below $g = (\div 2)$ is the same as the area above $f = (\times 2)$. $f = (\times 2)$ is not surjective. $g = (\div 2)$ is not injective.

What is the pointwise meaning of X/Y ? Let us first of all equip (5.157) with a type diagram:

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



Then we calculate:²²

$$\begin{aligned}
 & c (X/Y) a \\
 \equiv & \quad \{ \text{introduce points } C \xleftarrow{c} 1 \text{ and } A \xleftarrow{a} 1 ; (5.17) \} \\
 & x(\underline{c}^\circ \cdot (X/Y) \cdot \underline{a})x \\
 \equiv & \quad \{ \forall\text{-one-point (A.5)} \} \\
 & x' = x \Rightarrow x'(\underline{c}^\circ \cdot (X/Y) \cdot \underline{a})x \\
 \equiv & \quad \{ \text{go pointfree (5.19)} \} \\
 & id \subseteq \underline{c}^\circ \cdot (X/Y) \cdot \underline{a} \\
 \equiv & \quad \{ \text{shunting rules} \} \\
 & \underline{c} \cdot \underline{a}^\circ \subseteq X/Y \\
 \equiv & \quad \{ \text{universal property (5.157)} \} \\
 & \underline{c} \cdot \underline{a}^\circ \cdot Y \subseteq X \\
 \equiv & \quad \{ \text{now shunt } \underline{c} \text{ back to the right} \} \\
 & \underline{a}^\circ \cdot Y \subseteq \underline{c}^\circ \cdot X \\
 \equiv & \quad \{ \text{back to points via (5.17)} \} \\
 & \langle \forall b : a Y b : c X b \rangle
 \end{aligned}$$

²² Following the strategy suggested in exercise 5.42.

In summary:

$$c (X/Y) a \equiv \langle \forall b : a Y b : c X b \rangle \tag{5.158}$$

In words: in the same way relation *composition* hides an *existential* quantifier (5.11), *relation division* (5.158) hides a *universal* one. Let us feel what (5.158) means through an example: let

$a Y b$ = passenger a chooses flight b
 $c X b$ = company c operates flight b

Then (5.158) yields : whenever a chooses a flight b it turns out that b is operated by company c . So:

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

Therefore, (5.157) captures, in a rather eloquent way, the duality between universal and existential quantification. It is no wonder, then, that the relational equivalent to $(x \div y) \times y \leq x$ above is

$$(X/S) \cdot S \subseteq X$$

This *cancellation* rule, very often used in practice, unfolds to

$$\langle \forall b : a S b : c X b \rangle \wedge a S b' \Rightarrow c X b'$$

i.e. to the well-known device in logic known as *modus ponens*: $((S \rightarrow X) \wedge S) \rightarrow X$.

There is one important difference between (5.156) and (5.157): while multiplication in (5.156) is commutative, and thus writing $z \times y$ or $y \times z$ is the same, writing $Z \cdot Y$ or $Y \cdot Z$ makes a lot of difference because composition is not commutative in general. The dual division operator is obtained by taking converses over (5.157):

$$\begin{aligned} & Y \cdot Z \subseteq X \\ \equiv & \quad \{ \text{converses} \} \\ & Z^\circ \cdot Y^\circ \subseteq X^\circ \\ \equiv & \quad \{ \text{division (5.157)} \} \\ & Z^\circ \subseteq X^\circ / Y^\circ \\ \equiv & \quad \{ \text{converses} \} \\ & Z \subseteq \underbrace{(X^\circ / Y^\circ)^\circ}_{Y \setminus X} \end{aligned}$$

In summary:

$$X \cdot Z \subseteq Y \Leftrightarrow Z \subseteq X \setminus Y \tag{5.159}$$

Once variables are added to $Y \setminus X$ we get:

$$a(X \setminus Y)c \equiv \langle \forall b : b X a : b Y c \rangle \tag{5.160}$$

Thus we are ready to add two more rows to table 3:

$(f X) \subseteq Y \equiv X \subseteq (g Y)$			
Description	f	g	Obs.
Left-division	$(R \cdot)$	$(R \setminus)$	read "R under ..."
Right-division	$(\cdot R)$	$(/ R)$	read "...over R"

As example of left division consider the relation $a \in x$ between a set x and each of its elements a :

$$A \xleftarrow{\in} PA \tag{5.161}$$

Then inspect the meaning of relation $PA \xleftarrow{\in \setminus \in} PA$ using (5.160):

$$x_1 (\in \setminus \in) x_2 \Leftrightarrow \langle \forall a : a \in x_1 : a \in x_2 \rangle$$

We conclude that quotient $PA \xleftarrow{\in \setminus \in} PA$ expresses the inclusion relation among sets.

Relation division gives rise to a number of combinators in relation algebra that are very useful in problem specification. We review some of these below.

Exercise 5.46. Prove the equalities

$$X \cdot f = X / f^\circ \tag{5.162}$$

$$f \setminus X = f^\circ \cdot X \tag{5.163}$$

$$X / \perp = \top \tag{5.164}$$

$$X / id = X \tag{5.165}$$

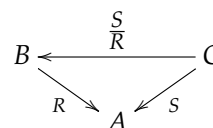
$$R \setminus (f^\circ \cdot S) = f \cdot R \setminus S \tag{5.166}$$

$$R \setminus \top \cdot S = ! \cdot R \setminus ! \cdot S \tag{5.167}$$

$$R / (S \cup P) = R / S \cap R / P \tag{5.168}$$

□

SYMMETRIC DIVISION Given two arbitrary relations R and S typed as in the diagram below, define the *symmetric division* $\frac{S}{R}$ of S by R by:

$$b \frac{S}{R} c \equiv \langle \forall a :: a R b \Leftrightarrow a S c \rangle$$


$$B \xleftrightarrow{\frac{S}{R}} C$$

$$B \xrightarrow{R} A$$

$$C \xrightarrow{S} A$$
(5.169)

That is, $b \stackrel{S}{R} c$ means that b and c are related to exactly the same outputs (in A) by R and by S . Another way of writing (5.169) is $b \stackrel{S}{R} c \equiv \{a \mid a R b\} = \{a \mid a S c\}$ which is the same as

$$b \stackrel{S}{R} c \equiv \Lambda R b = \Lambda S c \tag{5.170}$$

where Λ is the *power transpose operator*²³ which maps a relation $Q : Y \leftarrow X$ to the set valued function $\Lambda Q : X \rightarrow P Y$ such that $\Lambda Q x = \{y \mid y Q x\}$. Another way to define $\stackrel{S}{R}$ is

$$\frac{S}{R} = R \setminus S \cap R^\circ / S^\circ \tag{5.171}$$

which factors symmetric division into the two asymmetric divisions $R \setminus S$ (5.159) and R / S (5.157) already studied above. Moreover, for $R, S := f, g$, definition (5.171) instantiates to $\frac{f}{g}$ as defined by (5.49). By (5.159, 5.157), (5.171) is equivalent to the universal property:

$$X \subseteq \frac{S}{R} \equiv R \cdot X \subseteq S \wedge S \cdot X^\circ \subseteq R \tag{5.172}$$

From the definitions above a number of standard properties arise:

$$\left(\frac{S}{R}\right)^\circ = \frac{R}{S} \tag{5.173}$$

$$\frac{S}{R} \cdot \frac{Q}{S} \subseteq \frac{Q}{R} \tag{5.174}$$

$$f^\circ \cdot \frac{S}{R} \cdot g = \frac{S \cdot g}{R \cdot f} \tag{5.175}$$

$$id \subseteq \frac{R}{R} \tag{5.176}$$

Thus $\frac{R}{R}$ is always an *equivalence relation*, for any given R . Furthermore,

$$R = \frac{R}{R} \equiv R \text{ is an equivalence relation} \tag{5.177}$$

holds. Also note that, even in the case of functions, (5.174) remains an inclusion,

$$\frac{f}{g} \cdot \frac{h}{f} \subseteq \frac{h}{g} \tag{5.178}$$

since:

$$\begin{aligned} & \frac{f}{g} \cdot \frac{h}{f} \subseteq \frac{h}{g} \\ \Leftarrow & \quad \{ \text{factor } \frac{id}{g} \text{ out} \} \\ & f \cdot \frac{h}{f} \subseteq h \\ \Leftarrow & \quad \{ \text{factor } h \text{ out} \} \end{aligned}$$

²³ See section 5.24 for more details about this operator.

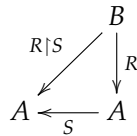
$$\begin{aligned}
 & f \cdot \frac{id}{f} \subseteq id \\
 \equiv & \quad \{ \text{shunting rule (5.47)} \} \\
 & f \subseteq f \\
 \equiv & \quad \{ \text{trivial} \} \\
 & \text{true} \\
 & \square
 \end{aligned}$$

From (5.178) it follows that $\frac{f}{f}$ is always transitive. By (5.173) it is symmetric and by (5.30) it is reflexive. Thus $\frac{f}{f}$ is an *equivalence relation*.

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

$$X \subseteq R \upharpoonright S \equiv X \subseteq R \wedge X \cdot R^\circ \subseteq S \tag{5.179}$$

cf. diagram:



This states that $R \upharpoonright S$ is the largest part of R such that, if it yields an output for an input x , it must be a maximum, with respect to S , among all possible outputs of x by R . By indirect equality, (5.179) is equivalent to the closed definition:

$$R \upharpoonright S = R \cap S / R^\circ \tag{5.180}$$

(5.179) can be regarded as a Galois connection between the set of all *subrelations* of R and the set of *optimization criteria* (S) on its outputs.

Combinator $R \upharpoonright S$ also makes sense when R and S are finite, relational data structures (eg. tables in a database). Consider, for instance, the following example of $R \upharpoonright S$ in a *data-processing* context: given

<i>Examiner</i>	<i>Mark</i>	<i>Student</i>
<i>Smith</i>	10	<i>John</i>
<i>Smith</i>	11	<i>Mary</i>
<i>Smith</i>	15	<i>Arthur</i>
<i>Wood</i>	12	<i>John</i>
<i>Wood</i>	11	<i>Mary</i>
<i>Wood</i>	15	<i>Arthur</i>

and wishing to “choose the best mark” for each student, project over *Mark*, *Student* and optimize over the \geq ordering on *Mark*:

$$\left(\begin{array}{c|c} \textit{Mark} & \textit{Student} \\ \hline 10 & \textit{John} \\ 11 & \textit{Mary} \\ 12 & \textit{John} \\ 15 & \textit{Arthur} \end{array} \right) \upharpoonright_{\geq} = \left(\begin{array}{c|c} \textit{Mark} & \textit{Student} \\ \hline 11 & \textit{Mary} \\ 12 & \textit{John} \\ 15 & \textit{Arthur} \end{array} \right)$$

Relational shrinking can be used in many other contexts. Consider, for instance, a sensor recording temperatures (T), $T \xleftarrow{S} \mathbb{N}_0$, where data in \mathbb{N}_0 are “time stamps”. Suppose one wishes to filter out repeated temperatures, keeping the first occurrences only. This can be specified by

$$T \xleftarrow{\text{sub } S} \mathbb{N}_0 = (S^\circ \upharpoonright \leq)^\circ$$

a function that removes all duplicates while keeping the first instances.

Among the properties of shrinking [47] we single out the two *fusion* rules:

$$(S \cdot f) \upharpoonright R = (S \upharpoonright R) \cdot f \quad (5.181)$$

$$(f \cdot S) \upharpoonright R = f \cdot (S \upharpoonright (f^\circ \cdot R \cdot f)) \quad (5.182)$$

Some more basic properties are: “chaotic optimization”,

$$R \upharpoonright \top = R \quad (5.183)$$

“impossible optimization”

$$R \upharpoonright \perp = \perp \quad (5.184)$$

and “brute force” determinization:

$$R \upharpoonright id = \text{largest deterministic fragment of } R \quad (5.185)$$

$R \upharpoonright id$ is the extreme case of the fact which follows:

$$R \upharpoonright S \text{ is simple} \Leftrightarrow S \text{ is anti-symmetric} \quad (5.186)$$

Thus anti-symmetric criteria always lead to determinism, possibly at the sacrifice of totality. Also, for R simple:

$$R \upharpoonright S = R \quad \equiv \quad \text{img } R \subseteq S \quad (5.187)$$

Thus (functions):

$$f \upharpoonright S = f \quad \Leftrightarrow \quad S \text{ is reflexive} \quad (5.188)$$

The distribution of shrinking by join,

$$(R \cup S) \upharpoonright Q = (R \upharpoonright Q) \cap Q/S^\circ \cup (S \upharpoonright Q) \cap Q/R^\circ \quad (5.189)$$

has a number of corollaries, namely a *conditional rule*,

$$(p \rightarrow R, T) \upharpoonright S = p \rightarrow (R \upharpoonright S), (p \upharpoonright S) \quad (5.190)$$

the *distribution* over alternatives (5.114),

$$[R, S] \upharpoonright U = [R \upharpoonright U, S \upharpoonright U] \quad (5.191)$$

and the “*function competition*” rule:

$$(f \cup g) \upharpoonright S = (f \cap S \cdot g) \cup (g \cap S \cdot f) \quad (5.192)$$

(Recall that $S/g^\circ = S \cdot g$.)

Putting universal properties (5.172,5.179) together we get, by indirect equality,

$$\frac{R}{g} = g^\circ \cdot (R \upharpoonright id) \quad (5.193)$$

$$\frac{f}{R} = (R \upharpoonright id)^\circ \cdot f \quad (5.194)$$

capturing a relationship between shrinking and symmetric division: knowing that $R \upharpoonright id$ is the deterministic fragment of R , we see how the *vagueness* of arbitrary R replacing either f or g in $\frac{f}{g}$ is forced to shrink.

Exercise 5.47. Use shrinking and other relational combinators to calculate, from a relation of type (5.127), the relation of type $Student \times Course \rightarrow Result$ that tells the final results of all exams. (NB: assume $Time = \mathbb{N}_0$ ordered by (\leq) .)

□

RELATION OVERRIDING Another operator enabled by relation division is the relational *overriding* combinator,

$$R \dagger S = S \cup R \cap \perp / S^\circ \quad (5.195)$$

which yields the relation which contains the whole of S and that part of R where S is undefined — read $R \dagger S$ as “ R overridden by S ”.

It is easy to show that $\perp \dagger S = S$, $R \dagger \perp = R$ and $R \dagger R = R$ hold. From (5.195) we derive, by indirect equality, the universal property:

$$X \subseteq R \dagger S \equiv X - S \subseteq R \wedge (X - S) \cdot S^\circ = \perp \quad (5.196)$$

The following property establishes a relationship between overriding and the McCarthy conditional.

$$p \rightarrow g, f = f \dagger (g \cdot \Phi_p) \quad (5.197)$$

Notation Φ_p is explained in the next section.

Exercise 5.48. Show that

$$R \dagger f = f$$

holds, arising from (5.196,5.138) — where f is a function, of course.

□

5.20 PREDICATES ALSO BECOME RELATIONS

Recall from (5.49) the notation $\frac{f}{g} = g^\circ \cdot f$ and define, given a predicate $p : A \rightarrow \mathbb{B}$, the relation $\Phi_p : A \rightarrow A$ such as:²⁴

$$\Phi_p = id \cap \frac{true}{p} \quad (5.198)$$

By (5.49), Φ_p is the *coreflexive* relation which represents predicate p as a binary relation,

$$y \Phi_p x \Leftrightarrow y = x \wedge p x \quad (5.199)$$

as can be easily checked. From (5.198) one gets the limit situations:²⁵

$$\Phi_{true} = id \quad (5.200)$$

$$\Phi_{false} = \perp \quad (5.201)$$

Moreover,

$$\Phi_{p \wedge q} = \Phi_p \cap \Phi_q \quad (5.202)$$

$$\Phi_{p \vee q} = \Phi_p \cup \Phi_q \quad (5.203)$$

$$\Phi_{\neg p} = id - \Phi_p \quad (5.204)$$

follow immediately from (5.199) and from (5.39) one infers $\frac{true}{p} \cdot R \subseteq \frac{true}{p}$ for any R . In particular, $\frac{true}{p} \cdot \top = \frac{true}{p}$ since $\frac{true}{p} \subseteq \frac{true}{p} \cdot \top$ always holds. Then, by distributive property (5.62):

$$\Phi_p \cdot \top = \frac{true}{p} \quad (5.205)$$

Moreover, the following two properties hold:

$$R \cdot \Phi_p = R \cap \top \cdot \Phi_p \quad (5.206)$$

$$\Phi_q \cdot R = R \cap \Phi_q \cdot \top \quad (5.207)$$

We check (5.207):²⁶

$$\begin{aligned} & \Phi_q \cdot R \\ = & \{ (5.109); (5.198) \} \\ & \frac{id \vee true}{id \vee p} \cdot R \\ = & \{ (5.104) \} \\ & (id \vee p)^\circ \cdot (R \vee true) \\ = & \{ (5.108) \} \end{aligned}$$

²⁴ Recall that *true* is the *constant* function yielding True for every argument (5.40).

²⁵ $\Phi_{false} = \perp$ arises from (5.54) since True \neq False.

²⁶ The other is obtained from (5.207) by taking converses.

$$\begin{aligned}
 & R \cap \frac{true}{p} \\
 = & \quad \{ (5.205) \} \\
 & R \cap \Phi_p \cdot \top \\
 & \square
 \end{aligned}$$

Note the meaning of (5.206) and (5.207):

$$\begin{aligned}
 b (R \cdot \Phi_p) a & \Leftrightarrow b R a \wedge (p a) \\
 b (\Phi_q \cdot R) a & \Leftrightarrow b R a \wedge (q b)
 \end{aligned}$$

So (5.206) — resp. (5.207) — restricts R to inputs satisfying p — resp. outputs satisfying q .

Below we show how to use relation restriction and overriding in specifying the operation that, in the Alcuin puzzle — recall (5.74)

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

— specifies the move of *Beings* to the other bank:

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

By (5.197) this simplifies to a McCarthy conditional:

$$\textit{carry who where} = (\in \textit{who}) \rightarrow \textit{cross} \cdot \textit{where} , \textit{where} \quad (5.208)$$

In pointwise notation, *carry* is the function:

$$\begin{aligned}
 \textit{carry who where } b = & \\
 & \mathbf{if } b \in \textit{who} \\
 & \mathbf{then } \textit{cross } m \mathbf{ else } m \\
 & \mathbf{where } m = \textit{where } b
 \end{aligned}$$

Note the type $\textit{carry} : P\textit{Being} \rightarrow \textit{Bank}^{\textit{Being}} \rightarrow \textit{Bank}^{\textit{Being}}$.

A notable property of coreflexive relations is that their composition coincides with their meet:

$$\Phi_q \cdot \Phi_p = \Phi_q \cap \Phi_p \quad (5.209)$$

In consequence, composing a coreflexive with itself yields that very same coreflexive: $\Phi_p \cdot \Phi_p = \Phi_p$. (5.209) follows from (5.206,5.207):

$$\begin{aligned}
 & \Phi_q \cdot \Phi_p \\
 = & \quad \{ R = R \cap R \} \\
 & \Phi_q \cdot \Phi_p \cap \Phi_q \cdot \Phi_p \\
 = & \quad \{ (5.206,5.207) \} \\
 & \Phi_q \cap \top \cdot \Phi_p \cap \Phi_p \cap \Phi_p \cdot \top \\
 = & \quad \{ \textit{since } \Phi_p \subseteq \top \cdot \Phi_p \textit{ and } \Phi_q \subseteq \Phi_q \cdot \top \} \\
 & \Phi_q \cap \Phi_p \\
 & \square
 \end{aligned}$$

EQUALIZERS The definition of Φ_p (5.187) can be regarded as a particular case of an *equalizer*: given two functions $B \xleftarrow{f,g} A$, the equalizer of f and g is the relation $eq(f, g) = id \cap \frac{f}{g}$. By indirect equality,

$$X \subseteq eq(f, g) \Leftrightarrow X \subseteq id \wedge g \cdot X \subseteq f$$

That is, $eq(f, g)$ is the largest coreflexive X that restricts g so that f and g yield the same outputs.

Clearly, $eq(f, f) = id$. Note that an equalizer can be empty, cf. e.g. $eq(true, false) = \perp$.

Exercise 5.49. Based on (5.71) show that

$$g^\circ \cdot \Phi_p \cdot f = \frac{f}{g} \cap \frac{true}{p \cdot g} \quad (5.210)$$

holds.²⁷

□

5.21 GUARDS, COREFLEXIVES AND THE MCCARTHY CONDITIONAL

From the definition of a McCarthy conditional (2.70) we obtain $p? = p \rightarrow i_1$, i_2 and then $p? = i_2 \dagger i_1 \cdot \Phi_p$ by (5.197). A third way to express the guard $p?$ is

$$p? = i_1 \cdot \Phi_p \cup i_2 \cap (\perp / (i_1 \cdot \Phi_p)^\circ) \quad (5.211)$$

by (5.195), which simplifies to:

$$p? = [\Phi_p, \Phi_{\neg p}]^\circ \quad (5.212)$$

To prove (5.212) note that $\perp / (i_1 \cdot \Phi_p)^\circ = \perp / \Phi_p$, immediate by the laws of S/R and shunting. Then, $\perp / \Phi_p = \top \cdot \Phi_{\neg p}$. Here one only needs to check:

$$\begin{aligned} & \perp / \Phi_p \subseteq \top \cdot \Phi_{\neg p} \\ \equiv & \quad \left\{ \frac{\neg p}{true} = \frac{p}{false} \right\} \\ & \perp / \Phi_p \subseteq \frac{p}{false} \\ \equiv & \quad \{ \text{going pointwise} \} \\ & \langle \forall y, x : y (\perp / \Phi_p) x : p x = False \rangle \\ \equiv & \quad \{ (5.159); (5.199) \} \\ & \langle \forall y, x : p x \Rightarrow False : p x = False \rangle \\ \equiv & \quad \{ \text{trivial} \} \\ & true \end{aligned}$$

□

²⁷ Both sides of the equality mean $g b = f a \wedge p (g b)$.

Finally, back to (5.211):

$$\begin{aligned}
 p? &= i_1 \cdot \Phi_p \cup i_2 \cap \top \cdot \Phi_{\neg p} \\
 &\equiv \{ (5.206); \text{converses} \} \\
 (p?)^\circ &= \Phi_p \cdot i_1^\circ \cup \Phi_{\neg p} \cdot i_2^\circ \\
 &\equiv \{ (5.121) \} \\
 p? &= [\Phi_p, \Phi_{\neg p}]^\circ \\
 &\square
 \end{aligned}$$

Exercise 5.50. From (5.211) infer

$$p \rightarrow R, S = R \cap \frac{p}{\text{true}} \cup S \cap \frac{p}{\text{false}} \quad (5.213)$$

and therefore $p \rightarrow R, S \subseteq R \cup S$. Furthermore, derive (2.78) from (5.213) knowing that $\text{true} \cup \text{false} = \top$.

□

DOMAIN AND RANGE Suppose one computes $\ker \langle R, id \rangle$ instead of $\ker R$. Since $\ker \langle R, id \rangle = \ker R \cap id$ (5.111), coreflexive relation is obtained. This is called the *domain* of R , written:

$$\delta R = \ker \langle R, id \rangle \quad (5.214)$$

Since²⁸

$$\top \cdot R \cap id = R^\circ \cdot R \cap id \quad (5.215)$$

domain can be also defined by

$$\delta R = \top \cdot R \cap id \quad (5.216)$$

Dually, one can define the *range* of R as the domain of its converse:

$$\rho R = \delta R^\circ = \text{img } R \cap id \quad (5.217)$$

For functions, range and image coincide, since $\text{img } f \subseteq id$ for any f . For injective relations, domain and kernel coincide, since $\ker R \subseteq id$ in such situations. These two operators can be shown to be characterized by two Galois connections, as follows:

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

²⁸ (5.215) follows from $id \cap \top \cdot R \subseteq R^\circ \cdot R$ which can be easily checked pointwise.

Let us show that indeed

$$\delta X \subseteq Y \equiv X \subseteq T \cdot Y \quad (5.218)$$

$$\rho R \subseteq Y \equiv R \subseteq Y \cdot T \quad (5.219)$$

hold, where variable Y ranges over coreflexive relations only. We only derive (5.218), from which (5.219) is obtained taking converses. We rely on the definition just given and on previously defined connections:

$$\begin{aligned} & \delta X \subseteq Y \\ \equiv & \quad \{ (5.216) \} \\ & T \cdot X \cap id \subseteq Y \\ \equiv & \quad \{ \text{two Galois connections} \} \\ & X \subseteq T \setminus (id \Rightarrow Y) \\ \equiv & \quad \{ T \setminus (id \Rightarrow Y) = T \cdot Y, \text{ see below} \} \\ & X \subseteq T \cdot Y \\ \square \end{aligned}$$

To justify the hint above, first note that $T \cdot Y \subseteq id \Rightarrow Y$, for Y coreflexive — recall (5.198) and (5.205). Then:

$$\begin{aligned} & T \setminus (id \Rightarrow Y) \subseteq T \cdot Y \\ \Leftarrow & \quad \{ \text{monotonicity ; rule "raise-the-lower-side"} \} \\ & T \setminus (T \cdot Y) \subseteq T \cdot Y \\ \equiv & \quad \{ (5.167) ; f \cdot f^\circ \cdot f = f \text{ for } f := ! \text{ (twice)} \} \\ & ! \setminus ! \cdot Y \subseteq T \cdot Y \\ \equiv & \quad \{ f \setminus R = f^\circ \cdot R ; T = \ker ! \} \\ & T \cdot Y \subseteq T \cdot Y \\ \square \end{aligned}$$

Note the left-cancellation rule of the δ connection:

$$R \subseteq T \cdot \delta R \quad (5.220)$$

From this the following domain/range elimination rules follow:

$$T \cdot \delta R = T \cdot R \quad (5.221)$$

$$\rho R \cdot T = R \cdot T \quad (5.222)$$

$$\delta R \subseteq \delta S \equiv R \subseteq T \cdot S \quad (5.223)$$

Proof of (5.221):

$$\begin{aligned} & T \cdot \delta R = T \cdot R \\ \equiv & \quad \{ \text{circular inclusion} \} \end{aligned}$$

$$\begin{aligned}
& \top \cdot \delta R \subseteq \top \cdot R \wedge \top \cdot R \subseteq \top \cdot \delta R \\
\equiv & \quad \{ (5.97) \text{ twice} \} \\
& \delta R \subseteq \top \cdot R \wedge R \subseteq \top \cdot \delta R \\
\equiv & \quad \{ \text{cancellation (5.220)} \} \\
& \delta R \subseteq \top \cdot R \\
\equiv & \quad \{ \delta R = \top \cdot R \cap id \text{ (5.216)} \} \\
& \text{true} \\
& \square
\end{aligned}$$

Rule (5.222) follows by dualization (converses) and (5.223) follows from (5.218) and (5.221). More facts about domain and range:

$$\delta (R \cdot S) = \delta (\delta R \cdot S) \quad (5.224)$$

$$\rho (R \cdot S) = \rho (R \cdot \rho S) \quad (5.225)$$

$$R = R \cdot \delta R \quad (5.226)$$

$$R = \rho R \cdot R \quad (5.227)$$

Last but not least: given predicate q and function f ,

$$\Phi_{(q \cdot f)} = \delta (\Phi_q \cdot f) \quad (5.228)$$

holds. Proof:

$$\begin{aligned}
& \Phi_{(q \cdot f)} \\
= & \quad \{ (5.198) \} \\
& id \cap \frac{true}{q \cdot f} \\
= & \quad \{ \text{since } \frac{f}{f} \text{ is reflexive (5.30)} \} \\
& id \cap \frac{f}{f} \cap \frac{true \cdot f}{q \cdot f} \\
= & \quad \{ (5.109); \text{products} \} \\
& id \cap \frac{(id \vee true) \cdot f}{(id \vee q) \cdot f} \\
= & \quad \{ (5.49); (5.109) \} \\
& id \cap f^\circ \cdot (id \cap \frac{true}{q}) \cdot f \\
= & \quad \{ (5.198) \} \\
& id \cap f^\circ \cdot \Phi_q \cdot f \\
= & \quad \{ \delta R = id \cap R^\circ \cdot R \} \\
& \delta (\Phi_q \cdot f) \\
& \square
\end{aligned}$$

Exercise 5.51. Recalling (5.206), (5.207) and other properties of relation algebra, show that: (a) (5.218) and (5.219) can be re-written with R replacing \top ; (b) $\Phi \subseteq \Psi \equiv ! \cdot \Phi \subseteq ! \cdot \Psi$ holds.²⁹

□

5.22 DIFUNCTIONALITY

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

The class of difunctional relations is vast. \top and \perp are difunctional, and so are all coreflexive relations, as is easy to check. It also includes all simple relations, since $R \cdot R^\circ = \text{img } R \subseteq \text{id}$ wherever R is simple. Moreover, divisions of functions are difunctional because every symmetric division is so, as is easy to check by application of laws (5.174) and (5.173):

$$\begin{aligned} & \frac{f}{g} \cdot \left(\frac{f}{g} \right)^\circ \cdot \frac{f}{g} \subseteq \frac{f}{g} \\ \Leftrightarrow & \quad \{ (5.51); (5.178) \} \\ & \frac{f}{g} \cdot \frac{f}{f} \subseteq \frac{f}{g} \\ \Leftrightarrow & \quad \{ (5.178) \} \\ & \frac{f}{g} \subseteq \frac{f}{g} \\ \square & \end{aligned}$$

For $g = \text{id}$ above we get that any function f being difunctional can be expressed by $f \cdot \frac{f}{f} = f$.

Recall that an equivalence relation can always be represented by the kernel of some function, typically by $R = \frac{\Delta R}{\Lambda R}$. So equivalence relations are difunctional. The following rule is of practical relevance:

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

Proof (of transitivity):

$$\begin{aligned} & R \cdot R \subseteq R \\ \equiv & \quad \{ R \text{ is difunctional} \} \\ & R \cdot R \subseteq R \cdot R^\circ \cdot R \\ \equiv & \quad \{ R \text{ is symmetric} \} \\ & R \cdot R \subseteq R \cdot R \cdot R \end{aligned}$$

²⁹ Thus coreflexives can be represented by *vectors* and vice-versa.

$$\begin{aligned} &\Leftarrow \{ \text{monotonicity} \} \\ &\quad id \subseteq R \\ &\square \end{aligned}$$

Difunctional relations are also called *regular*, *rational* or *uniform*. First, some intuition about what “regularity” means: a regular (difunctional) relation is such that, wherever two inputs have a common image, then they have *exactly the same* set of images. In other words, the image sets of two different inputs are either disjoint or the same. As a counterexample, take the following relation, represented as a matrix with inputs taken from set $\{a_1, \dots, a_5\}$ and outputs delivered into set $\{b_1, \dots, b_5\}$:

R	a_1	a_2	a_3	a_4	a_5	
b_1	0	0	1	0	1	(5.229)
b_2	0	0	0	0	0	
b_3	0	1	0	0	0	
b_4	0	1	0	1	0	
b_5	0	0	0	1	0	

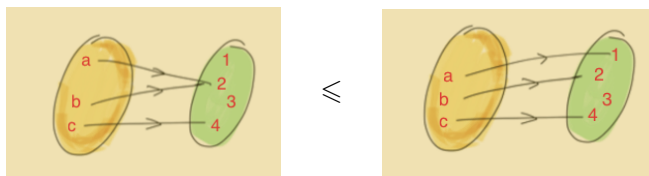
Concerning inputs a_3 and a_5 , regularity holds; but sets $\{b_3, b_4\}$ and $\{b_4, b_5\}$ — the images of a_2 and a_4 , respectively — are neither disjoint nor the same: so R isn’t regular. It would become so if e.g. b_4 were dropped from both image sets or one of b_3 or b_5 were replaced for the other in the corresponding image set.

5.23 OTHER ORDERINGS ON RELATIONS

THE INJECTIVITY PREORDER The kernel relation $\ker R = R^\circ \cdot R$ measures the level of *injectivity* of R according to the preorder

$$R \leq S \equiv \ker S \subseteq \ker R \tag{5.230}$$

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



This ordering is surprisingly useful in formal specification because of its properties. For instance, it is upper-bounded by relation *pairing*, recall (5.103):

$$\langle R, S \rangle \leq X \equiv R \leq X \wedge S \leq X \tag{5.231}$$

Cancellation of (5.231) means that *pairing* always *increases injectivity*:

$$R \leq \langle R, S \rangle \text{ and } S \leq \langle R, S \rangle. \tag{5.232}$$

(5.232) unfolds to $\ker \langle R, S \rangle \subseteq (\ker R) \cap (\ker S)$, confirming (5.111). The following injectivity *shunting law* arises as a Galois connection:

$$R \cdot g \leq S \equiv R \leq S \cdot g^\circ \tag{5.233}$$

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

$$! \leq f \leq id$$

where (recall) $1 \xleftarrow{!} A$ is the unique function of its type, where 1 is the singleton type. Moreover,

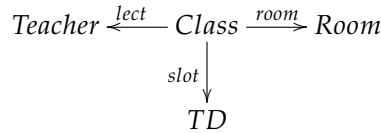
- A function is *injective* iff $id \leq f$. Thus $\langle f, id \rangle$ is always *injective* (5.232).
- Two functions f e g are said to be *complementary* wherever $id \leq \langle f, g \rangle$.
- Any relation R can be factored into the composition $f \cdot g^\circ$ of two complementary functions f and g .³⁰

For instance, the *projections* $\pi_1 (a, b = a)$, $\pi_2 (a, b = b)$ are complementary since $\langle \pi_1, \pi_2 \rangle = id$ (2.32).

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

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

in the context of the data model



where *TD* abbreviates time and date. What are we telling about this model by writing $room \leq \langle lect, slot \rangle$? We unfold this constraint in the expected way:

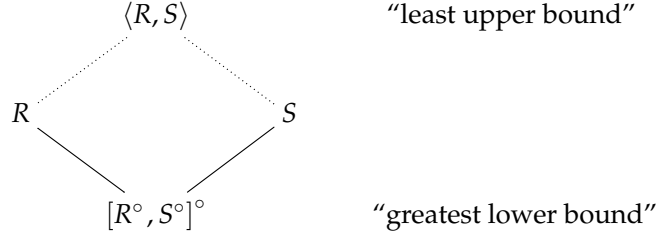
$$\begin{aligned} & room \leq \langle lect, slot \rangle \\ \equiv & \{ (5.230) \} \\ & \ker \langle lect, slot \rangle \subseteq \ker room \\ \equiv & \{ (5.111); (5.53) \} \\ & \frac{lect}{lect} \cap \frac{slot}{slot} \subseteq \frac{room}{room} \\ \equiv & \{ \text{going pointwise, for all } c_1, c_2 \in Class \} \\ & (lect\ c_1 = lect\ c_2 \wedge slot\ c_1 = slot\ c_2) \Rightarrow (room\ c_1 = room\ c_2) \end{aligned}$$

This $room \leq \langle lect, slot \rangle$ constrains the model in the sense of imposing that a given lecturer cannot be in two different rooms at the same time.

³⁰ This remarkable factorization is known as a *tabulation* of R [12].

c_1 and c_2 are classes shared by different courses, possibly of different degrees. In the standard terminology of database theory this is called a *functional dependency*, see exercises 5.54 and 5.55 in the sequel.

Interestingly, the injectivity preorder not only has least upper bounds but also greatest lower bounds,



that is,

$$X \leq [R^\circ, S^\circ]^\circ \iff X \leq R \wedge X \leq S \tag{5.234}$$

as the calculation shows:

$$\begin{aligned} X &\leq [R^\circ, S^\circ]^\circ \\ &\equiv \{ \text{injectivity preorder ; } \ker R^\circ = \text{img } R \} \\ &\text{img } [R^\circ, S^\circ] \subseteq \ker X \\ &\equiv \{ (5.124) \} \\ &R^\circ \cdot R \cup S^\circ \cdot S \subseteq \ker X \\ &\equiv \{ \text{kernel; } \cdot \cup \text{-universal} \} \\ &\ker R \subseteq \ker X \wedge \ker S \subseteq \ker X \\ &\equiv \{ \text{injectivity preorder (twice)} \} \\ &X \leq R \wedge X \leq S \\ &\square \end{aligned}$$

Note the meaning of the glb of R and S ,

$$x [R^\circ, S^\circ]^\circ a \iff \langle \exists b : x = i_1 b : b R a \rangle \vee \langle \exists c : x = i_2 c : c R a \rangle$$

since $[R^\circ, S^\circ]^\circ = i_1 \cdot R \cup i_2 \cdot S$. This is the most injective relation that is less injective than R and S because it just “collates” the outputs of both relations without confusing them.³¹

Exercise 5.52. Show that $R^\circ \cdot S = \perp$ is necessary for the coproduct of two injective relations R and S to be injective:

$$id \leq [R, S] \iff id \leq R \wedge id \leq S \wedge R^\circ \cdot S = \perp \tag{5.235}$$

³¹ It turns out that universal property $X = [R^\circ, S^\circ]^\circ \iff i_1^\circ \cdot X = R \wedge i_2^\circ \cdot X = S$ holds, as is easy to derive from (5.114). So $[R^\circ, S^\circ]^\circ$ is the *categorical product* for relations:

$$A \rightarrow (B + C) \overset{\cong}{\rightleftarrows} (A \rightarrow B) \times (A \rightarrow C)$$

That is, among relations, the product is obtained as the converse dual of the coproduct. This is called a *biproduct* [38].

□

Exercise 5.53. The Peano algebra $\mathbb{N}_0 \xleftarrow{\text{in}} 1 + \mathbb{N}_0 = [0, \text{succ}]$ is an isomorphism³², and therefore injective. Check what (5.235) means in this case.

□

Exercise 5.54. An SQL-like relational operator is projection,

$$\pi_{g,f}R \stackrel{\text{def}}{=} g \cdot R \cdot f^\circ \quad \begin{array}{ccc} B & \xleftarrow{R} & A \\ g \downarrow & & \downarrow f \\ C & \xleftarrow{\pi_{g,f}R} & D \end{array} \quad (5.236)$$

whose set-theoretic meaning is³³

$$\pi_{g,f}R = \{(g \ b, f \ a) \mid b \in B \wedge a \in A \wedge b \ R \ a\} \quad (5.237)$$

Functions f and g are often referred to as attributes of R . Derive (5.237) from (5.236).

□

Exercise 5.55. A relation R is said to satisfy functional dependency (FD) $g \rightarrow f$, written $g \xrightarrow{R} f$ wherever projection $\pi_{f,g}R$ (5.236) is simple.

1. Recalling (5.230), prove the equivalence:

$$g \xrightarrow{R} f \equiv f \leq g \cdot R^\circ \quad (5.238)$$

2. Show that $g \xrightarrow{R} f$ trivially holds wherever g is injective and R is simple, for all (suitably typed) f .
3. Prove the composition rule of FDs:

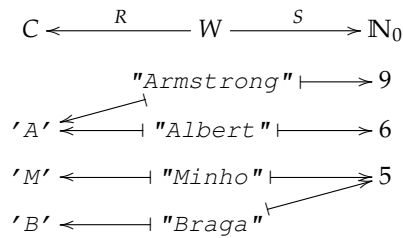
$$h \xleftarrow{S \cdot R} g \Leftarrow h \xleftarrow{S} f \wedge f \xleftarrow{R} g \quad (5.239)$$

□

³² Recall section 3.1.

³³ Note that any relation $R : B \leftarrow A$ defines the set of pairs $\{(b, a) \mid b \ R \ a\}$. Predicate $b \ R \ a$ describes R intensionally. The set $\{(b, a) \mid b \ R \ a\}$ is the extension of R .

Exercise 5.56. Let R and S be the two relations depicted as follows:



Check the assertions:

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

□

Exercise 5.57. As follow up to exercise 5.9,

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

□

THE DEFINITION PREORDER The injectivity preorder works perfectly for functions, which are entire relations. For non-entire R it behaves in a mixed way, measuring not only injectivity but also definition (entireness). It is useful to order relations with respect to how defined they are:

$$R \preceq S \equiv \delta R \subseteq \delta S \tag{5.240}$$

From $\top = \ker !$ one draws another version of (5.240), $R \preceq S \equiv ! \cdot R \subseteq ! \cdot S$. The following Galois connections

$$R \cup S \preceq T \equiv R \preceq T \wedge S \preceq T \tag{5.241}$$

$$R \cdot f^\circ \preceq S \equiv R \preceq S \cdot f \tag{5.242}$$

are easy to prove. Recalling (5.223), (5.240) can also be written

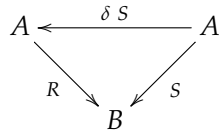
$$\delta R \subseteq \delta S \equiv R \subseteq \top \cdot S \tag{5.243}$$

THE REFINEMENT ORDER Standard programming theory relies on a notion of program *refinement*. As a rule, the starting point in any software design is a so-called *specification*, which indicates the expected behaviour of the program to be developed with no indication of *how* outputs are computed from the inputs. So, “vagueness” is a chief ingredient of a good specification, giving freedom to the programmer to choose a particular algorithmic solution.

Relation algebra captures this by ordering relations with respect to the degree in which they are closer to implementations:

$$S \vdash R \equiv \delta S \subseteq \delta R \wedge R \cdot \delta S \subseteq S \tag{5.244}$$

cf.



$S \vdash R$ is read: “ S is refined by R ”. In the limit situation, R is a function f , and then

$$S \vdash f \Leftrightarrow \delta S \subseteq f^\circ \cdot S \tag{5.245}$$

by shunting (5.46). This is a limit in the sense that f can be neither more defined nor more deterministic.

As maxima of the refinement ordering, functions are regarded as implementations “*par excellence*”. Note how (5.245) captures *implicit specification* S being refined by some function f — recall section 5.3. Back to points and thanks to (5.17) we obtain, in classical “VDM-speak”:

$$\forall a. \text{pre-}S(a) \Rightarrow \text{post-}S(f\ a, a)$$

In case S is entire, (5.245) simplifies to $S \vdash f \Leftrightarrow f \subseteq S$. As example of this particular case we go back to section 5.3 and prove that *abs*, explicitly defined by *abs* $i = \text{if } i < 0 \text{ then } -i \text{ else } i$, meets the implicit specification given there, here encoded by $S = \frac{\text{true}}{\text{geq0}} \cap (id \cup \text{sym})$ where $\text{geq0 } x = x \geq 0$ and $\text{sym } x = -x$. The explicit version below uses a McCarthy conditional, for $\text{lt0 } x = x < 0$. By exercise 5.50 term $id \cup \text{sym}$ in S can be ignored:

$$\begin{aligned} & \text{lt0} \rightarrow \text{sym}, id \subseteq \frac{\text{true}}{\text{geq0}} \\ \equiv & \quad \{ \text{shunting (5.46)} \} \\ & \text{geq0} \cdot (\text{lt0} \rightarrow \text{sym}, id) \subseteq \text{true} \\ \equiv & \quad \{ \text{fusion (2.71)} \} \\ & \text{lt0} \rightarrow \text{geq0} \cdot \text{sym}, \text{geq0} \subseteq \text{true} \\ \equiv & \quad \{ -x \geq 0 \Leftrightarrow x \leq 0 = \text{leq0 } x \} \\ & \text{lt0} \rightarrow \text{leq0}, \text{geq0} \subseteq \text{true} \end{aligned}$$

$$\begin{aligned} &\equiv \{ x < 0 \Rightarrow x \leq 0 \text{ and } \neg(x < 0) \Leftrightarrow x \geq 0 \} \\ &\quad \text{lt}0 \rightarrow \text{true}, \text{true} \subseteq \text{true} \\ &\equiv \{ p \rightarrow f, f = f \text{ (exercise 5.50)} \} \\ &\quad \text{true} \\ &\square \end{aligned}$$

Finally note that an equivalent way of stating (5.244) without using the domain operator is:

$$S \vdash R \equiv T \cdot S \cap T \cdot R \cap (R \cup S) = R \tag{5.246}$$

Exercise 5.58. Prove (5.246).
 □

5.24 BACK TO FUNCTIONS

In this chapter we have argued that one needs *relations* in order to reason about *functions*. The inverse perspective — that relations can be represented as functions — also makes sense and it is, in many places, the approach that is followed.

Indeed, relations can be *transposed* back to functions without losing information. There are two transposes of interest. One is complete in the sense that it allows us to see *any* relation as a function. The other is specific, in the sense that it only applies to (the very important class of) simple relations (vulg. *partial* functions).

POWER TRANSPOSE Let $A \xrightarrow{R} B$ be a relation and define the function

$$\begin{aligned} \Lambda R &: A \rightarrow \mathbb{P} B \\ \Lambda R a &= \{ b \mid b R a \} \end{aligned}$$

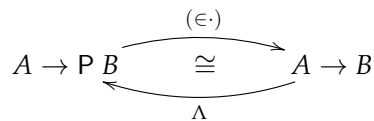
which is such that:

$$\Lambda R = f \equiv \langle \forall b, a :: b R a \Leftrightarrow b \in f a \rangle \tag{5.247}$$

That is:

$$f = \Lambda R \Leftrightarrow \in \cdot f = R \tag{5.248}$$

cf.



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

Moving the variables of (5.170) outwards by use of (5.17), we obtain the following *power transpose cancellation rule*:³⁴

$$\frac{\Lambda S}{\Lambda R} = \frac{S}{R} \tag{5.249}$$

Read from right to left, this shows a way of converting arbitrary symmetric divisions into function divisions.

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

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

such that:

$$\Gamma S = f \iff (\forall b, a :: b S a \iff (i_1 b) = f a)$$

That is:

$$f = \Gamma S \iff S = i_1^\circ \cdot f \tag{5.250}$$

cf.

$$A \rightarrow B + 1 \begin{array}{c} \xrightarrow{(i_1^\circ \cdot)} \\ \cong \\ \xleftarrow{\Gamma} \end{array} A \rightarrow B$$

In words: simple *relations* can always be represented by “Maybe”, or “pointer”-valued *functions*. Recall section 4.1, where the *Maybe* monad was used to “totalize” partial functions. Isomorphism (5.250) explains why such a totalization makes sense. For finite relations, and assuming these represented extensionally as lists of pairs, the function $lookup :: Eq a \Rightarrow a \rightarrow [(a, b)] \rightarrow Maybe b$ in Haskell implements the “Maybe”-transpose.

5.25 BIBLIOGRAPHY NOTES

Chronologically, relational notation emerged — earlier than predicate logic itself — in the work of Augustus De Morgan (1806-71) on binary relations [40]. Later, Peirce (1839-1914) invented quantifier notation to explain De Morgan’s algebra of relations (see eg. [40] for details). De Morgan’s pioneering work was ill fated: the language³⁵ invented to explain his calculus of relations became eventually more popular than the calculus itself. Alfred Tarski (1901-83), who had a life-long struggle with quantified notation [16, 23], revived relation algebra. Together

³⁴ This rule is nothing but another way of stating exercise 4.48 proposed in [12]. Note that ΛR is always a function.

³⁵ Meanwhile named FOL, first order logic.

with Steve Givant he wrote a book (published posthumously) on *set theory without variables* [67].

Meanwhile, category theory was born, stressing the role of *arrows* and *diagrams* and on the arrow language of diagrams, which is inherently *pointfree*. The category of sets and functions immediately provided a basis for pointfree functional reasoning, but this was by and large ignored by John Backus (1924-2007) in his FP algebra of programs [7] which is APL-flavoured. (But there is far more in it than such a flavour, of course!) Anyway, Backus' landmark FP paper was among the first to show how relevant such reasoning style is to computing.

A bridge between the two pointfree schools, the relational and the categorial, was eventually established by Freyd and Ščedrov [19] in their proposal of the concept of an *allegory*. This gave birth to *typed* relation algebra and relation diagrams like those adopted in the current book for *relational thinking*. The pointfree algebra of programming (AoP) as it is understood today, which has reached higher education thanks to textbook [12] written by Bird and Moor, stems directly from [19].

In his book on *relational mathematics* [66], Gunther Schmidt makes extensive use of matrix displays, notation, concepts and operations in relation algebra. Winter [71] generalizes relation algebra to so-called Goguen categories.

In the early 1990s, the Groningen-Eindhoven MPC group led by Backhouse [1, 5] contributed decisively to the AoP by structuring relation algebra in terms of Galois connections. This elegant approach has been very influential in the way (typed) relation algebra was faced afterwards, for instance in the way relation shrinking was introduced in the algebra [47, 59]. Most of the current chapter was inspired by [5].

A

BACKGROUND — EINDHOVEN QUANTIFIER CALCULUS

This appendix is a summary of section 4.3 of reference [5].

A.1 NOTATION CONVENTIONS

The Eindhoven quantifier calculus adopts the following notation:

- $\langle \forall x : R : T \rangle$ means: “for all x in the range R , term T holds”, where R and T are logical expressions involving x .
- $\langle \exists x : R : T \rangle$ means: “for some x in the range R , term T holds”

A.2 RULES

The main rules of the Eindhoven quantifier calculus are listed below:

Trading:

$$\langle \forall k : R \wedge S : T \rangle = \langle \forall k : R : S \Rightarrow T \rangle \quad (\text{A.1})$$

$$\langle \exists k : R \wedge S : T \rangle = \langle \exists k : R : S \wedge T \rangle \quad (\text{A.2})$$

de Morgan:

$$\neg \langle \forall k : R : T \rangle = \langle \exists k : R : \neg T \rangle \quad (\text{A.3})$$

$$\neg \langle \exists k : R : T \rangle = \langle \forall k : R : \neg T \rangle \quad (\text{A.4})$$

One-point:

$$\langle \forall k : k = e : T \rangle = T[k := e] \quad (\text{A.5})$$

$$\langle \exists k : k = e : T \rangle = T[k := e] \quad (\text{A.6})$$

Nesting:

$$\langle \forall a, b : R \wedge S : T \rangle = \langle \forall a : R : \langle \forall b : S : T \rangle \rangle \quad (\text{A.7})$$

$$\langle \exists a, b : R \wedge S : T \rangle = \langle \exists a : R : \langle \exists b : S : T \rangle \rangle \quad (\text{A.8})$$

Rearranging- \forall :

$$\langle \forall k : R \vee S : T \rangle = \langle \forall k : R : T \rangle \wedge \langle \forall k : S : T \rangle \quad (\text{A.9})$$

$$\langle \forall k : R : T \wedge S \rangle = \langle \forall k : R : T \rangle \wedge \langle \forall k : R : S \rangle \quad (\text{A.10})$$

Rearranging- \exists :

$$\langle \exists k : R : T \vee S \rangle = \langle \exists k : R : T \rangle \vee \langle \exists k : R : S \rangle \quad (\text{A.11})$$

$$\langle \exists k : R \vee S : T \rangle = \langle \exists k : R : T \rangle \vee \langle \exists k : S : T \rangle \quad (\text{A.12})$$

Splitting:

$$\langle \forall j : R : \langle \forall k : S : T \rangle \rangle = \langle \forall k : \langle \exists j : R : S \rangle : T \rangle \quad (\text{A.13})$$

$$\langle \exists j : R : \langle \exists k : S : T \rangle \rangle = \langle \exists k : \langle \exists j : R : S \rangle : T \rangle \quad (\text{A.14})$$