6SENG001W Reasoning about Programs

Lecture 5

Introduction to Relations
&
Using Relations in B

Introduction: Relations in B

The aim of this lecture is to introduce the following aspects of *relations* in B:

- Concept of a *relation*
- Definitions of *Cartesian product*, *binary relation*, *ordered pair* & *maplet*
- Definitions of *projections*, *membership*, *domain*, *range*, *source* & *target* of a relation
- *Operators* on relations:
  - *domain* – $\mathrm{dom}(R)$
  - *range* – $\mathrm{ran}(R)$
  - *relational image* – $R[\ S\ ]$
  - *domain restriction* – $S \lhd R$
  - *range restriction* – $R \rhd S$
  - *domain anti-restriction* – $S \ntriangleleft R$
  - *range anti-restriction* – $R \ntriangleright S$
  - *relational override* – $R_1 \Leftarrow\!\!\!+ R_2$

(Continued)

- Special relations:
  - *identity relation* – $\mathrm{id}(X)$
  - *inverse* of a relation – R^{-1}
- Introduce *composition* of relations & *composition closures*:
  - *forward composition* – $R\,;Q$
  - *repeated composition* – R^n
  - *Transitive Closure* – R^+
  - *Reflexive-Transitive Closure* – R^*
- Using *relations* in B:
  - defining relations using *set comprehension*
  - AMN for relations
  - Example B machine using relations.

PART I

Introduction to Relations

Relations in B

A *relation* is used to *relate* the elements of one set to the elements of another set.

The two sets related in this way, can both have the same *type* or have different types.

For example, we can relate *people* to *colours* & use this to represent a person's favourite colours, as follows:

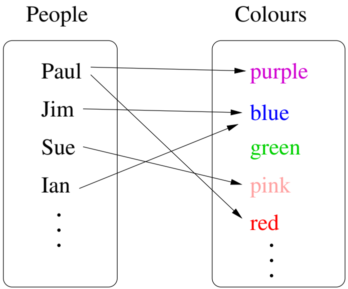

Figure : 5.1 A Relation between People & Colours

Definition of a Relation

The people & colours relationship is represented formally by a *relation*.

We shall call it the $favourite$ relation & it would be represented by the following *set of pairs of values*:

$$\begin{aligned} favourite \; = \; \{ \; & (Paul, purple), \; (Paul, red), \; (Jim, blue), \\ & (Sue, pink), \; (Ian, blue), \; \ldots \; \} \end{aligned}$$

An *element* of this relation (set) is a *pair of values*:

$$(Paul, \; purple)$$

Where $Paul$ is of type $PEOPLE$ & $purple$ is of type $COLOUR$.

Cartesian Products & Relations

Formally, a *relation* is based on the idea of a *Cartesian product*.

A *Cartesian product*: $X \times Y$ is the *product* of two or more sets & forms a new set.

For example, given the two sets A & X defined as follows:

$$A = \{ a, b \} \qquad X = \{ 1, 2, 3 \}$$

The *Cartesian product* $A \times X$ is the new set:

$$A \times X = \{ (a,1), (a,2), (a,3), (b,1), (b,2), (b,3) \}$$

This set contains all of the *combinations/pairings* of an element of the first set A with an element of the second set X.

There are *no restrictions* on how the pairs of values from each set are combined, *relations* are said to be *"many-to-many"*.

That is, one of the first values, e.g. a can be *"related"* to *many* different second values, e.g. $1, 2, 3$; & one of the second values, e.g. 2, can be *"related"* to *many* different first values, e.g. a, b.

Non-Binary Cartesian Products

A *Cartesian product* is not restricted to combining only two sets in the way illustrated above, i.e. a binary Cartesian product, but can be used to combine any number of sets.

For example, given three sets X, Y & Z, the *Cartesian product* of three sets X, Y & Z is:

$$X \times Y \times Z$$

A value of this type is called a *tuple*:

$$(x, y, z)$$

where x is of type X ($x \in X$), y is of type Y ($y \in Y$) & z is of type Z ($z \in Z$).

A *tuple* is *ordered*, since the order of the components is important.

For example, in general:

$$(x, y, z) \;\neq\; (y, x, z) \;\neq\; (z, y, x)$$

as they are all of *different types*.

Binary Relations

A special case of a *Cartesian product* is an *ordered pair*.

A *binary* relation is a set of *ordered pairs*, of related values.

Example a $speaks$ relation between countries & languages:

$$\begin{aligned} COUNTRY &= \{ France, Canada, England, Wales, \\ &\quad\ Scotland, NIreland,\ Italy,\ USA,\ \ldots \} \\ LANGUAGE &= \{ French,\ English,\ Welsh,\ Italian,\ \ldots \} \end{aligned}$$

An example of a possible value for this *relation* (set of ordered pairs) is:

$$\begin{aligned} speaks = \{ &(France,\ French), && (Canada,\ French), \\ &(Canada,\ English), && (England,\ English), \\ &(Wales,\ Welsh), && (Scotland,\ English), \\ &(NIreland,\ English), && \ldots \\ \} \end{aligned}$$

Definition of $speaks$ Binary Relations

The $speaks$ relation (set) can be declared as follows:

$$speaks \in \mathbb{P}(COUNTRY \times LANGUAGE)$$

Note that the ***type*** of the ***elements*** of the $speaks$ relation – the ***ordered pairs***, e.g. $(Wales, Welsh)$, are all of the following ***type***:

$$(Wales, Welsh) \in COUNTRY \times LANGUAGE$$

The types could be declared in the ***opposite order*** to define a $spoken\_in$ relation, as follows:

$$spoken\_in \in \mathbb{P}(LANGUAGE \times COUNTRY)$$

Alternatively, we could define both as ***subsets of the Cartesian products***:

$$\begin{aligned} speaks &\subseteq COUNTRY \times LANGUAGE \\ spoken\_in &\subseteq LANGUAGE \times COUNTRY \end{aligned}$$

The $speaks$ relation

We can represent the $speaks$ relation diagrammatically as follows:

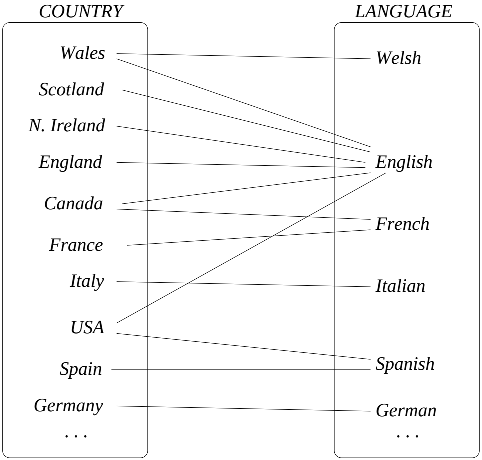

Figure : 5.2 Speaks relation

Declaring Relations

Given any two types (sets):

$$X, \ Y$$

The usual style used to *declare a relation* R is as follows:

$$R \ \in \ X \leftrightarrow Y$$

For example, $speaks$ & $favourite$ would usually be define as follows:

$$\begin{aligned} speaks \ &\in \ COUNTRY \leftrightarrow LANGUAGE \\ favourite \ &\in \ PEOPLE \leftrightarrow COLOUR \end{aligned}$$

Note the following equivalence between the notation:

$$X \leftrightarrow Y \ = \ \mathbb{P}(X \times Y)$$

So the $speaks$ & $favourite$ relations could equally be define as follows:

$$\begin{aligned} speaks \ &\in \ \mathbb{P}(COUNTRY \times LANGUAGE) \\ favourite \ &\in \ \mathbb{P}(PEOPLE \times COLOUR) \end{aligned}$$

Maplets & Ordered Pairs

The idea of a related pair, i.e. *ordered pair*, is also captured by the idea of a *maplet*:

$$x \mapsto y \quad = \quad (x, y)$$

meaning that *"x maps to y"*.

For example, from the $speaks$ & $favourite$ relations:

$$\begin{aligned} Wales \mapsto Welsh \quad &= \quad (Wales, Welsh) \\ Paul \mapsto purple \quad &= \quad (Paul,\ purple) \end{aligned}$$

Note: that **ProB** excepts both ordered pair formats, i.e. "(x, y)" & "$x \mapsto y$".

Unfortunately, **Atelier B** does not type check the "(x, y)" format correctly.

So when entering ordered pairs into either tool *only use the maplet format:* "`x |-> y`".

Projections of a Relation

For any ordered pair $(x,\ y) \in X \times Y$, the two values $x \in X$ & $y \in Y$ can be individually *"selected"*, i.e. "extracted" from the ordered pair "$(x,\ y)$" & "$(x \mapsto y)$".

This is achieved by two operators prj_1 and prj_2:

$$\begin{aligned} \mathrm{prj}_1(X,\ Y)(x,\ y) &= \mathrm{prj}_1(X,\ Y)(x \mapsto y) = x \\ \mathrm{prj}_2(X,\ Y)(x,\ y) &= \mathrm{prj}_2(X,\ Y)(x \mapsto y) = y \end{aligned}$$

Note: prj_1 and prj_2 are know as *"projection" functions* or just *projections*.

Note that prj_1 & prj_2 can be applied to *any* ordered pair, no matter what types the ordered pairs are, since the types, e.g X & Y have to be supplied as the first parameter, e.g. "(X, Y)".

For example, we can apply them to ordered pairs in the $speaks$ relation:

$$\begin{aligned} \mathrm{prj}_1(COUNTRY,\ LANGUAGE)(Wales, Welsh) &= Wales \\ \mathrm{prj}_2(COUNTRY,\ LANGUAGE)(Wales, Welsh) &= Welsh \end{aligned}$$

Membership of a Relation

To see if *a pair of values are related by a relation*, just test if the pair is a member of the relation, i.e. a member of the set representing the relation.

For example:

“Is French spoken in Wales?”

from the above definition the answer is no, as the following is true:

$$Wales \mapsto French \notin speaks \qquad \text{or} \qquad (Wales, French) \notin speaks$$

What about

“Is French spoken in France?”

from the above definition of $speaks$ the answer is yes, e.g.

$$France \mapsto French \in speaks$$

We have that for any relation R:

$$x \mapsto y \in R \Leftrightarrow (x, y) \in R$$

Example: Noughts & Crosses

Given the following *noughts & crosses* grid:

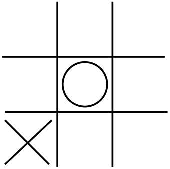

Figure : 5.3 Noughts & Crosses Grid

Question: how could we represent this using a relation?

Hints: consider the following:

- It is a 3 by 3 grid.
- How would it help us if we numbered the columns & rows, i.e. 1 – 3.
- Does this correspond to any of the things we have just been looking at?

PART II

Extracting Information & Creating Relations

Extracting Information & Creating Relations

The following relational operations allow us to either:

Extract information about a relation:

- *domain* of a relation – $\mathrm{dom}(R)$
- *range* of a relation – $\mathrm{ran}(R)$
- *relational image* – $R[\ A\ ]$

Create a new relation from an existing one:

- *domain restriction* – $A \lhd R$
- *range restriction* – $R \rhd B$
- *domain anti-restriction* – $A \ntriangleleft R$
- *range anti-restriction* – $R \ntriangleright B$
- *relational override* – $R_1 \oplus R_2$

We will use the following *relation* and *sets*:

$$
\begin{array}{lllllll}
R \in X \leftrightarrow Y, & & R_1 \in X \leftrightarrow Y, & & R_2 \in X \leftrightarrow Y \\
A \in \mathbb{P}(X) & \textit{or} & A \subseteq X \\
B \in \mathbb{P}(Y) & \textit{or} & B \subseteq Y
\end{array}
$$

Domain & Range of a Relation

A *relation* R:

$$R \in X \leftrightarrow Y$$

relates values of a set called the *source*, i.e. X, to values of a set called the *target*, i.e. Y.

For $speaks$: $COUNTRY$ is the *source* & $LANGUAGE$ is the *target*.

Usually, only a subset of the *source* & *target* sets are used in the definition of a relation.

These are called respectively, the *domain* & *range* of the relation.

For example given the following relation:

$$R \in X \leftrightarrow Y$$

The *domain* & *range* of R are:

$\mathrm{dom}(R)$ – which is of type $\mathbb{P}(X)$

$\mathrm{ran}(R)$ – which is of type $\mathbb{P}(Y)$

Exercise: what are the values of: $\mathrm{dom}(speaks)$ & $\mathrm{ran}(speaks)$?

Domain & Range of the $speaks$ relation

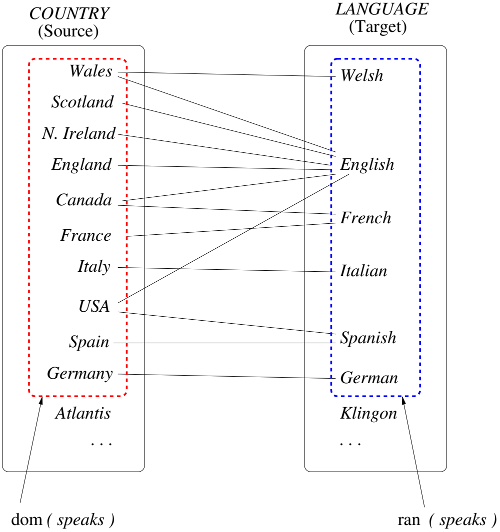

Figure : 5.4 $speaks$ Relation's Domain & Range

Relational Image

This operator is used to discover the set of values from the range of a relation R related to a set of values from the domain A.

$$R[\ A\ ]$$

Where the following are true:

$$\begin{aligned} R[\ A\ ] &\in \mathbb{P}(Y) \\ A &\subseteq \mathrm{dom}(R) \\ R[\ A\ ] &\subseteq \mathrm{ran}(R) \end{aligned}$$

For example:

"What languages are spoken in Canada & Wales?"

from the definition of $speaks$:

$$speaks\ [\ \{\ Canada, Wales\ \}\ ]\ =\ \{\ French,\ English,\ Welsh\ \}$$

Domain Restriction

Used to create a new relation by *restricting* a relation to that part where the *domain* is contained in a set.

$$A \lhd R$$

Where the following are true:

$$A \lhd R \in X \leftrightarrow Y$$
$$A \subseteq \mathrm{dom}(R)$$
$$A \lhd R \subseteq R$$

A *restriction operator* forms a *smaller relation* than the original one, since it usually contains fewer ordered pairs.

For example:

"Restrict the speaks relation to just Britain."

$$\begin{aligned} &\{\, Wales,\ Scotland,\ England,\ NIreland \,\} \lhd speaks \\ &\quad = \{\, (Wales, Welsh), (England, English), \\ &\qquad (Scotland, English), (NIreland, English),\ \ldots \,\} \end{aligned}$$

Range Restriction

This is used to restrict a relation R to that part where the *range* is contained in a set B.

$$R \rhd B$$

Where the following are true:

$$R \rhd B \ \in \ X \leftrightarrow Y$$
$$B \subseteq \mathrm{ran}(R)$$
$$R \rhd B \subseteq R$$

For example:

"Restrict the speaks relation to those countries which speak French."

$$speaks \rhd \{French\} \ = \ \{\ (France, French),\ (Canada, French)\ \}$$

Domain Anti-Restriction

Domain anti-restriction is also known as *domain subtraction*.

This is used to restrict a relation to that part where the *domain* is *not* contained in a set.

$$A \lhd R$$

In other words, the part of relation R that *does not include* any of the values that are in A in its domain.

Where the following are true:

$$\begin{aligned} A \lhd R &\in X \leftrightarrow Y \\ A &\subseteq \operatorname{dom}(R) \\ A \lhd R &\subseteq R \\ \operatorname{dom}(A \lhd R) \cap A &= \varnothing \end{aligned}$$

For example: *"Restrict the speaks relation to non British countries."*

$$\{ \, Wales, \; Scotland, \; England, \; NIreland \, \} \lhd speaks$$
$$= \{ \, (France, French), (Canada, French), (Canada, English) \, \}$$

Range Anti-Restriction

Used to restrict a relation to that part where the *range* is *not* contained in a set.

$$R \rhd B$$

In other words, the part of relation R that *does not include* any of the values that are in B in its range.

Where the following are true:

$$\begin{aligned} R \rhd B &\in X \leftrightarrow Y \\ B &\subseteq \mathrm{ran}(R) \\ R \rhd B &\subseteq R \\ \mathrm{ran}(R \rhd B) \cap B &= \varnothing \end{aligned}$$

For example:

> *"Restrict the speaks relation to those countries where a language other than English is spoken."*

$$\begin{aligned} &speaks \rhd \{ English \} \\ &\quad = \{ (France, French), (Canada, French), (Wales, Welsh) \} \end{aligned}$$

Example of Relational Restriction Operators

Given the following set & relations: R, $R1$ & $R2$:

$$LETTER = \{ a,\ b,\ c,\ \ldots,\ z \}$$

$$\begin{aligned} R &\in LETTER \leftrightarrow \mathbb{N} \\ R1 &\in LETTER \leftrightarrow \mathbb{N} \\ R2 &\in LETTER \leftrightarrow \mathbb{N} \end{aligned}$$

$$\begin{aligned} R &= \{ (a,\ 1),\ (b,\ 1),\ (b,\ 2),\ (c,\ 3),\ (d,\ 2), \\ &\qquad (e,\ 4),\ (f,\ 4),\ (g,\ 5),\ (h,\ 6) \} \\ R1 &= \{ (a,\ 1),\ (b,\ 1),\ (b,\ 2),\ (c,\ 3),\ (d,\ 2) \} \\ R2 &= \{ (e,\ 4),\ (f,\ 4),\ (g,\ 5),\ (h,\ 6) \} \end{aligned}$$

R, $R1$ & $R2$ Represented Diagrammatically

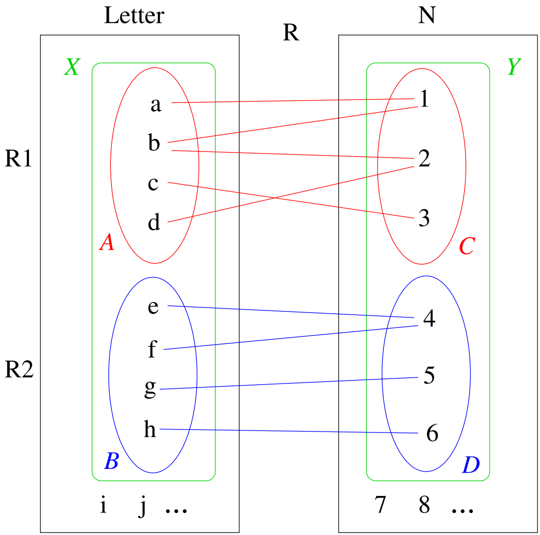

Figure : 5.5 R, $R1$ & $R2$ Relation Restriction

Example: Applying the Relational Restriction Operators

Given the definition of the sets:

$$X = A \cup B \qquad A = \{\, a,\ b,\ c,\ d \,\} \qquad B = \{\, e,\ f,\ g,\ h \,\}$$
$$Y = C \cup D \qquad C = \{\, 1,\ 2,\ 3 \,\} \qquad D = \{\, 4,\ 5,\ 6 \,\}$$

For the three relations R, $R1$ & $R2$ the ***source*** is $LETTER$ & the ***target*** is $\mathbb{N}$.

Their ***domain*** & ***range*** are as follows:

$$\mathrm{dom}(R) = X \qquad \mathrm{ran}(R) = Y$$
$$\mathrm{dom}(R1) = A \qquad \mathrm{ran}(R1) = C$$
$$\mathrm{dom}(R2) = B \qquad \mathrm{ran}(R2) = D$$

Then using the sets with the relation restriction operators we have:

$$X \lhd R = R \qquad R \rhd Y = R$$
$$A \lhd R = R1 \qquad B \lhd R = R2$$
$$R \rhd C = R1 \qquad R \rhd D = R2$$
$$A \unlhd R = R2 \qquad B \unlhd R = R1$$
$$R \unrhd C = R2 \qquad R \unrhd D = R1$$

Relational Override operator $\oplus$

A relation can be *modified* by:

- *adding pairs* to it, or
- by *removing pairs* from it, or
- by *altering* the *“mappings”* of a set of values in the domain, by changing the values they map to in the range.

To do this we use the *relational overriding* operator $\oplus$.

Given two relation:

$$R \in X \leftrightarrow Y \qquad\qquad Q \in X \leftrightarrow Y$$

then R *overridden by the relation* Q is written as follows:

$$R \oplus Q$$

This new relation *is the same as*:

- R for all values that are **not** in the domain of Q &
- Q for all values that are **in** the domain of Q.

Definition of Relational Override operator $\oplus$

If R & Q have *disjoint domains*:

$$\mathrm{dom}(R) \cap \mathrm{dom}(Q) = \varnothing$$

then

$$R \oplus Q = R \cup Q$$

It can be defined using the other relational operators as follows:

$$R \oplus Q \quad = \quad (\,(\;\mathrm{dom}(Q)\,) \lhd R\,) \cup Q$$

Example: $R \lhd\!\!\!+ \; Q$

Given the two relations R and Q:

$$R \in \mathbb{N} \leftrightarrow \mathbb{N}$$
$$Q \in \mathbb{N} \leftrightarrow \mathbb{N}$$

$$R = \{ (0,0),\ (1,2),\ (2,3),\ (3,3),\ (3,4),\ (3,5),\ (4,5) \}$$
$$Q = \{ (0,1),\ (3,3),\ (4,5),\ (4,6),\ (5,5),\ (6,7) \}$$

then their *domains* are:

$$\begin{aligned} \mathrm{dom}(R) &= \{ 0,\ 1,\ 2,\ 3,\ 4 \} \\ \mathrm{dom}(Q) &= \{ 0,\ 3,\ 4,\ 5,\ 6 \} \\ \mathrm{dom}(R) \cap \mathrm{dom}(Q) &= \{ 0,\ 3,\ 4 \} \end{aligned}$$

Since the domains of R & Q are *not disjoint*:

$$R \lhd\!\!\!+ \, Q \neq R \cup Q$$

Diagram of R & Q Relations

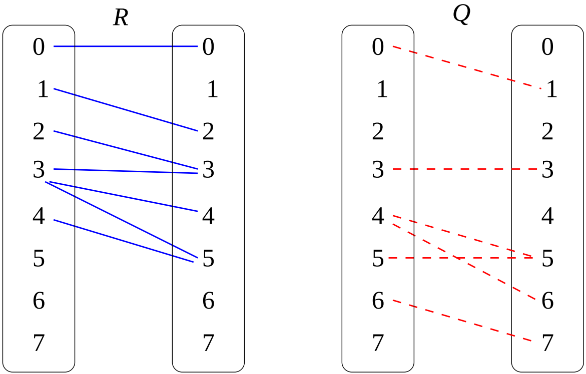

Figure : 5.6 R & Q Relations

R is the solid (blue) line and Q is the dashed (red) line.

Calculating $R \oplus Q$

We can calculate $R \oplus Q$ using its definition as follows:

$$
\begin{array}{lll}
R \oplus Q & & \\
= ((\ \mathrm{dom}(Q) \ ) \lhd R \ ) \cup Q & & \text{[Def. } \oplus\text{]} \\
= ((\{ \ 0, \ 3, \ 4, \ 5, \ 6 \ \} \ ) \lhd R \ ) \cup Q & & \text{[Def. } \mathrm{dom}(Q)\text{]} \\
= (\{ \ 0, \ 3, \ 4, \ 5, \ 6 \ \} & & \\
\quad \lhd \{ \ (0,0), \ (1,2), \ (2,3), \ (3,3), \ (3,4), \ (3,5), \ (4,5) \ \} \ ) & & \\
\quad \cup Q & & \text{[Def. } R\text{]} \\
= (\{ \ (1,2), \ (2,3) \ \} \ ) \cup Q & & \text{[Def. } \lhd\text{]} \\
= (\{ \ (1,2), \ (2,3) \ \} \ ) & & \\
\quad \cup \{ \ (0,1), \ (3,3), \ (4,5), \ (4,6), \ (5,5), \ (6,7) \ \} & & \text{[Def. } Q\text{]} \\
= \{ \ (0,1), \ (1,2), \ (2,3), \ (3,3), \ (4,5), \ (4,6), \ (5,5), \ (6,7) \ \} & & \text{[Def. } \cup\text{]}
\end{array}
$$

Therefore, the new relation of R *overridden by* Q is:

$$R \oplus Q \ = \ \{ \ (0,1), \ (1,2), \ (2,3), \ (3,3), \ (4,5), \ (4,6), \ (5,5), \ (6,7) \ \}$$

Diagram of $R \lhd\!\!\!+ Q$

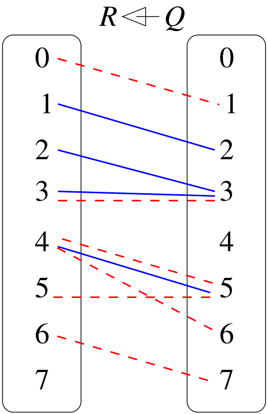

Figure : 5.7 Relational Overriding: $R \lhd\!\!\!+ Q$

R is the solid (blue) line and Q is the dashed (red) line.

What is the overall effect of $R \lhd\!\!\!+ Q$?

$R \lhd\!\!\!+ Q$ Elements	Comment
(0, 1)	Q's (0, 1) overrides R's (0, 0)
(1, 2)	R's (1, 2) as $1 \notin \mathrm{dom}(Q)$
(2, 3)	R's (2, 3) as $2 \notin \mathrm{dom}(Q)$
(3, 3)	Q's (3, 3) overrides R's (3, 3), (3, 4), (3, 5)
(4, 5), (4, 6)	Q's (4, 5), (4, 6) overrides R's (4, 5)
(5, 5)	Q's (5, 5)
(6, 7)	Q's (6, 7)

So the overall effect is the following:

- 0 now relates to 1 rather than 0.
- 3 is now only related to 3 & not 3, 4 & 5.
- 4 is now related to 5 & 6, not just 5.

PART III

Special Relations:
Identity, Inverse, Composition
&
Closures Relations

Special Relation: Identity Relation

The *identity relation* for a set of values X is defined as follows:

$$
\begin{aligned}
X &= \{\, x1,\ x2,\ x3,\ x4,\ \ldots,\ xn \,\} \\
\mathrm{id}(X) &\in X \leftrightarrow X \\
\mathrm{id}(X) &= \{\, (x1,\ x1),\ (x2,\ x2),\ \ldots,\ (xn,\ xn) \,\}
\end{aligned}
$$

is a relation which *maps all the elements of X to themselves*.

NOTE: to get the *identity relation* for a set X all we have to do is use "$\mathrm{id}(X)$", we do not have to write it out explicitly.

We can give a set comprehension equivalence for $\mathrm{id}(X)$ as follows:

$$
\mathrm{id}(X) = \{\, x,\ y \mid x \in X \ \wedge\ y \in X \ \wedge\ x = y \,\}
$$

It is **necessary to use two variables** x & y in the set comprehension & we achieve the *elements mapped to themselves* be using "$x = y$".

For example:

$$
\begin{aligned}
X &= \{\, 1, 2, 3, 4, 5 \,\} \\
\mathrm{id}(X) &\in \mathbb{N} \leftrightarrow \mathbb{N} \\
\mathrm{id}(X) &= \{\, (1,1),\ (2,2),\ (3,3),\ (4,4),\ (5,5) \,\}
\end{aligned}
$$

Special Relation: Inverse Relation

The ***inverse*** of a relation R:

$$R \in X \leftrightarrow Y$$

is written as follows & has the ***"opposite"*** type:

$$R^{-1} \in Y \leftrightarrow X$$

Hence both the following are true:

$$x \mapsto y \in R \qquad \text{and} \qquad y \mapsto x \in R^{-1}$$

For example, the ***inverse*** of the $speaks$ relation is:

$$speaks^{-1} \in LANGUAGE \leftrightarrow COUNTRY$$

which could be used to define the $spoken\_in$ relation:

$$spoken\_in = speaks^{-1}$$

so its value is then:

$$\begin{aligned} spoken\_in = \{ & (French, France), (French, Canada), \\ & (English, Canada), (English, England), \\ & (Welsh, Wales), (English, Scotland), \\ & (English, NIreland), \ldots \} \end{aligned}$$

Composition of Relations

Relations can be joined together by an operation called *composition*, to form new relations.

Given two relations R & Q:

$$R \in X \leftrightarrow Y$$
$$Q \in Y \leftrightarrow Z$$

then *"forward" relational composition* of R & Q is defined as follows:

$$R\,;Q \in X \leftrightarrow Z$$

Then for any pair:

$$(x, z) \in R\,;Q$$

there is a *"connecting"* y:

$$(x, z) \in R\,;Q \Leftrightarrow \exists\, y \cdot (\, y \in Y \land x \mapsto y \in R \land y \mapsto z \in Q\,)$$

Example: Composition of Relations

Given the two relations:

$$R \in LETTER \leftrightarrow \mathbb{N}$$
$$Q \in \mathbb{N} \leftrightarrow COLOUR$$

$$R = \{ (a,\ 1),\ (a,\ 2),\ (b,\ 2),\ (c,\ 3),\ (d,\ 4),\ (e,\ 5) \}$$
$$Q = \{ (1,\ red),\ (2,\ red),\ (2,\ blue),\ (3,\ green),\ (5,\ purple) \}$$

Then since the *target* of R is the *same type* as the *source* of Q, i.e. $\mathbb{N}$, we can *compose* these two relations using the relational composition operator "**;**".

Example: Composition of $R; Q$

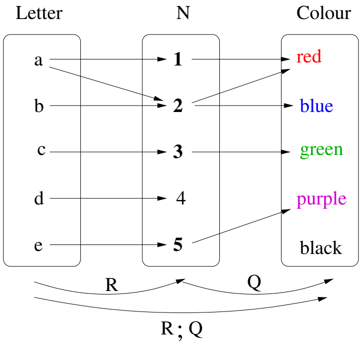

Figure : 5.8 Relational Composition

Then the composition of R & Q is:

$$R \,;\, Q \;:\; LETTER \leftrightarrow COLOUR$$
$$R \,;\, Q \;=\; \{\, (a,\ red),\ (a,\ blue),\ (b,\ red),\ (b,\ blue),\ (c,\ green),\ (e,\ purple) \,\}$$

Repeated Composition of a Relation

It is possible to repeatedly compose a relation R with itself, provided it is a *homogeneous* relation.

A *homogeneous* relation is one that relates values of the same type, i.e. *the source & target are the same type (set)*.

$$R \in X \leftrightarrow X$$

The repeated composition of R is:

$$R \,;\, R \in X \leftrightarrow X$$

We use the notation R^n for this & it is defined as follows:

$$\begin{array}{rcl} R^0 & == & \mathrm{id}(X) \\ R^1 & == & R \\ R^2 & == & R \,;\, R \\ R^3 & == & R \,;\, R \,;\, R \\ R^{n+1} & == & R \,;\, R^n \\ & == & \underbrace{R \,;\, \ldots \,;\, R}_{n+1} \end{array}$$

Example: Repeated Composition of a Relation

If we define a simple relation as follows:

$$alphabet \in LETTER \leftrightarrow LETTER$$
$$alphabet = \{ (a, b), (b, c), (c, d), (d, e), (e, f), (f, g), (g, h), (h, i), (a, x), (b, y), (c, z) \}$$

Then we can define repeated compositions of $alphabet$:

$$alphabet^2 = \{ (a, c), (b, d), (c, e), (d, f), (e, g), (f, h), (g, i), (a, y), (b, z) \}$$
$$alphabet^3 = \{ (a, d), (a, z), (b, e), (c, f), (d, g), (e, h), (f, i) \}$$
$$alphabet^4 = \{ (a, e), (b, f), (c, g), (d, h), (e, i) \}$$
$$alphabet^5 = \{ (a, f), (b, g), (c, h), (d, i) \}$$
$$alphabet^6 = \{ (a, g), (b, h), (c, i) \}$$
$$alphabet^7 = \{ (a, h), (b, i) \}$$
$$alphabet^8 = \{ (a, i) \}$$
$$alphabet^9 = \{ \}$$
$$alphabet^n = \{ \} \qquad \text{[For any } n \geq 9\text{]}$$

Closures of Relations: Transitive, Reflexive-Transitive

There are two important *"closure"* operations that can be performed on a *homogeneous* relation ($R : X \leftrightarrow X$) using the notion of repeated composition, these are:

Transitive Closure (R^+)

In general $x, y : X$

$$x \mapsto y \ \in \ R^+$$

means that there is a *repeated composition of R that relates* x & y.

A definition of R^+ is the following:

$$R^+ \ = \ \bigcup \{ \, R^k \mid k \in \mathbb{N}_1 \, \} \ = \ R^1 \ \cup \ R^2 \ \cup \ R^3 \ \ldots$$

Reflexive-Transitive Closure (R^*)

R^* is the *repeated composition of a relation R with the identity relation* id:

$$R^* \ = \ \mathrm{id}(X) \ \cup \ R^+ \qquad\qquad [\ \mathrm{id}(X) = R^0 \ ]$$

An alternative definition is the following:

$$R^* \ = \ \bigcup \{ \, R^k \mid k \in \mathbb{N} \, \} \ = \ \mathrm{id}(X) \ \cup \ R^1 \ \cup \ R^2 \ \cup \ R^3 \ \ldots$$

PART IV

Using Relations in B

AMN: Ordered Pairs & Relations

The following tables give the B AMN for relations.

B	*ASCII*	*Description*
$X \times Y$	`X * Y`	Cartesian product of X and Y
$x \mapsto y, \quad (x, y)$	`x |-> y`	Ordered pair, Maplet(*)
$\mathrm{prj}_1(S, T)(x \mapsto y)$ $\mathrm{prj}_1(S, T)(x, y)$	`prj1(S,T)(x|->y)`	Ordered pair projection function
$\mathrm{prj}_2(S, T)(x \mapsto y)$ $\mathrm{prj}_2(S, T)(x, y)$	`prj2(S,T)(x|->y)`	Ordered pair projection function
$\mathbb{P}(X \times Y)$	`POW(X * Y)`	Set of relations between X & Y
$X \leftrightarrow Y$	`X <-> Y`	Set of relations between X & Y
$\mathrm{dom}(R)$	`dom(R)`	Domain of relation R
$\mathrm{ran}(R)$	`ran(R)`	Range of relation R

NOTE: (*) that **ProB** excepts both ordered pair formats, i.e. "(x, y)" & "$x \mapsto y$". Unfortunately, **Atelier B** does not type check the "(x, y)" format correctly. So when entering ordered pairs into either tool *only use the maplet format:* "`x |-> y`".

AMN: Relations Operators

B	*ASCII*	*Description*
$A \lhd R$	`A <| R`	Domain restriction of R to the set A
$A ⩤ R$	`A <<| R`	Domain subtraction of R by the set A
$R \rhd B$	`R |> B`	Range restriction of R to the set B
$R ⩥ B$	`R |>> B`	Range anti-restriction of R by the set B
$R[B]$	`R[B]`	Relational Image of the set B of relation R
$R_1 \Leftarrow\!\!\!\!+ R_2$	`R1 <+ R2`	R_1 overridden by relation R_2
$R\,;Q$	`( R ; Q )`	Forward Relational composition (*)
$\mathrm{id}(X)$	`id(X)`	Identity relation
R^{-1}	`R~`	Inverse relation
R^n	`iterate(R,n)`	Iterated Composition of R
R^+	`closure1(R)`	Transitive closure of R
R^*	`closure(R)`	Reflexive-transitive closure of R

NOTE: (*) relational composition **must be enclosed in brackets**: “(“ & “)”.

Defining Relations using Set Comprehension

It is possible to specify a relation using *set comprehension*.

The type of the elements of the set are simply *maplets* (or *ordered pairs*), & the type of the set is the *Cartesian product* of the two sets being related.

For example, to define a relation that maps the numbers 0 to 3 to their squares, we define it as follows using AMN:

```
{ xx, yy | xx : NAT & yy : NAT & xx < 4 & yy = xx * xx }
=  { (0 |-> 0), (1|->1), (2|->4), (3|->9) }
```

A relation which contains all pairs of numbers between 1 & 6. (What could this be used to represent?)

```
{ xx, yy | xx : NAT1 & yy : NAT1 & xx <= 6 & yy <= 6 }
=  { (1|->1), (1|->2), (1|->3), (1|->4), (1|->5), (1|->6),
     (2|->1), (2|->2), (2|->3), ...,
     (3|->1), ...,
     ...,
     (6|->1), (6|->2), (6|->3), (6|->4), (6|->5), (6|->6) }
```

Example: `HotelRooms` a B Machine using Relations

The `HotelRooms` machine models a very simple hotel that has a number of *rooms* that can be either *empty* or *occupied by several guests*.

- There are *only 5 rooms*: `rm1` - `rm5`.
- There are *only 5 guests*: `Ian, Sue, Tom, Jim, Bill`.
- A room can either be *occupied by one or more guests* or *is empty*.
- *Initially all rooms are empty*.
- The *room status* regarding guest occupancy is represented by the `guests` relation between a room & the guests that occupy it.
- It has the following *operations*:
  - `guestsCheckIn` – check guest(s) into a room.
  - `guestsCheckOut` – check all guests out of a room.
  - `roomOccupants` – output guests occupying a room.
  - `hasGuestCheckedIn` – check if a guest is staying in the hotel, i.e. in any room.
  - `guestsSwapRoom` – swap the guests staying in two rooms, e.g. all guests in room i moved to room j & all guests in room j move to room i.

HotelRooms B Machine

HotelRooms State

```
MACHINE  HotelRooms

  SETS
     ROOM   = { rm1, rm2, rm3, rm4, rm5 } ;
     NAME   = { Ian, Sue, Tom, Jim, Bill, empty } ;
     ANSWER = { Yes, No }

  VARIABLES
     guests

  INVARIANT
     guests : ROOM <-> NAME

  INITIALISATION
     guests := ROOM * { empty }  /* All rooms are empty */
```

Note: The initialisation of `guests` uses *Cartesian Product* (*) in `ROOM * { empty }`, this expands to the following:

```
guests := { (rm1|->empty), (rm2|->empty), (rm3|->empty),
            (rm4|->empty), (rm5|->empty)                 }
```

Example: HotelRooms OPERATIONS (I)

HotelRoom Operations (I)

```
OPERATIONS

  guestsCheckIn( room, gnames ) =
    PRE
        (room : ROOM) & (gnames <: NAME) &
        (gnames /= {}) & (empty /: gnames)
    THEN
          guests := guests <+ ( { room } * gnames )
    END ;


  guestsCheckOut( room ) =
    PRE
        room : ROOM
    THEN
          guests := guests <+ { room |-> empty }
    END ;
```

Example: `HotelRooms` `OPERATIONS` (II)

```
HotelRoom Operations (II)

rmOcc <-- roomOccupants( room ) =
        PRE
            room : ROOM
        THEN
              rmOcc := ran( { room } <| guests )
        END ;

ans <-- hasGuestCheckedIn( gname ) =
      PRE
           (gname : NAME) & (gname /= empty)
      THEN
          IF ( gname : ran(guests) )
          THEN
                 ans := Yes
          ELSE
                 ans := No
          END
      END ;
```

Note: In `roomOccupants` an alternative is to use the *relational image*:

```
rmOcc :=  guests[ { room } ]
```

Example: HotelRooms OPERATIONS (III)

```
                    HotelRoom operations (III)

   guestsSwapRoom( roomi, roomj ) =
     PRE
          (roomi : ROOM) & (roomj : ROOM)
     THEN
         guests := guests
                   <+ (    ( { roomi } * guests[ { roomj } ] )
                       \/
                           ( { roomj } * guests[ { roomi } ] )
                      )
     END

 END /* HotelRoom */
```

Using **ProB**, an example of the value of the `guests` relation variable after several guests have checked in is:

```
guests =  { (rm1|->Ian),
            (rm2|->Ian), (rm2|->Sue),
            (rm3|->empty),
            (rm4|->Ian), (rm4|->Sue), (rm4|->Tom),
            (rm5|->empty) }
```