CHAPTER 6

Relations

Aims

To introduce relations, to describe the operators associated with relations, and to illustrate the use of relations in constructing specifications.

Learning objectives

When you have completed this chapter, you should be able to:

- appreciate the importance of relations as a central feature of most Z specifications;
- recognise those parts of an informally described system which are appropriate for modelling with relations;
- select appropriate Z operators for specifying state invariant properties and operation pre- and postconditions involving relations;
- read and understand Z specifications involving relations, and provide informed criticism of them in respect of clarity and fitness for purpose.

6.1 Introduction

We have seen that the state of many simple systems can be represented by one or more sets of atomic items, with appropriate constraints. However, the specification of most systems of interest requires more sophisticated mathematical structures. In particular, we may wish to represent the information that there is a connection, or relationship, between some of the objects in our specification; for example, that two people are siblings or that one or more students are studying one or more modules. We can represent this sort of information using a new type of set called a *relation*. Let's look at some of the fundamental mathematical ideas that we need.

6.2 Ordered pairs, Cartesian product and relations

An *ordered pair* (a, b) consists of two elements: a is the first element and b is the second element. In Z, ordered pairs are represented by the *maplet* notation $a \mapsto b$. The *Cartesian product* $A \times B$ of sets A and B is the set of all the ordered pairs which can be formed such that the first element in each pair is a member of A and the second element in each pair is a member of B.

$$A \times B = \{a : A, b : B \bullet a \mapsto b\}$$

For example, given

$$A = \{1, 2\}$$
$$B = \{a, b, c\}$$

We have that

$$A \times B = \{1 \mapsto a, 1 \mapsto b, 1 \mapsto c, 2 \mapsto a, 2 \mapsto b, 2 \mapsto c\}$$

Note that $\#(A \times B) = \#A * \#B$.

The operator $\times$ is another means of constructing new types from existing ones; that is, if A and B are types, then $A \times B$ is a type, the elements of which are maplets.

A *binary relation* between two sets A and B is any subset of $A \times B$. Thus the declaration of a variable R which is a binary relation between sets A and B is given by

$$R : \mathbb{P}(A \times B)$$

In other words, R is a member of the powerset of the Cartesian product of A and B. The Z shorthand for this declaration is

$$R : A \leftrightarrow B$$

In the special case in which A and B are the same set, the relation is said to be *homogeneous*.

Note that if A and B are not types, then the type of R must be inferred from the types of A and B, as before.

Given the types

$[PERSON, CAT]$ the set of all people and the set of all cats respectively

the relation

$$owns = \{cathy \mapsto tiddles, susan \mapsto puss, harry \mapsto tiger\}$$

is a relation of type

$$PERSON \leftrightarrow CAT$$

The relation

$$R = \{1 \mapsto 5, 4 \mapsto 4, 1 \mapsto 8, 7 \mapsto 2\}$$

is a homogeneous relation of type $\mathbb{Z} \leftrightarrow \mathbb{Z}$.

To indicate that a maplet $a \mapsto b$ is a member of a binary relation R we can either use the conventional set membership operator, that is

$$a \mapsto b \in R$$

or use the name of the relation as an infix operator, that is

$$a\ R\ b$$

In either case, the meaning is that a is related to b by R. For example, for the above relation R the following are both true:

$$1 \mapsto 5 \in R$$
$$1\ R\ 8$$

Note that the concept of Cartesian product may be generalised to more than two sets. Thus

$$A \times B \times C$$

defines the set of all triples (a, b, c) such that $a \in A$, $b \in B$, $c \in C$. In general, if there are n sets in the product, then the product set is a set of n-tuples (a 2-tuple is a pair, a 3-tuple is a triple, a 4-tuple is a quadruple).

Exercise 6.1

Given $A = \{1\}$, $B = \{2, 3\}$, write down the sets:

(i) $A \times B$
(ii) $\mathbb{P}(A) \times B$
(iii) $\mathbb{P}(A \times B)$
(iv) $(A \times B) \times (A \times B)$

6.3 The university modular degree scheme

A university operates a modular degree scheme, wherein students choose a selection of modules from a large menu. (There are certain restrictions on their choices, discussion of which we will defer until later.) One function of the administration system for the scheme is to keep track of which students are doing which modules. We define the basic types

$[PERSON, MODULE]$ the set of all people and the set of all modules respectively

Then the type

$$PERSON \times MODULE = \{p : PERSON, m : MODULE \bullet p \mapsto m\}$$

is the set of all ordered pairs (maplets) such that the first thing in each pair is a person and the second thing in each pair is a module. We will consider the maplet $p \mapsto m$ to represent the information that person p is taking module m. The information for the entire modular degree scheme would therefore be represented by a *set* of maplets, that is by a relation which is a subset of $PERSON \times MODULE$. We will call this relation

$$taking : PERSON \leftrightarrow MODULE$$

Exercises 6.2

1. What relation would represent a degree scheme where none of the people are taking any of the modules?
2. What relation would represent a degree scheme where every person in the world is taking every possible module?
3. Given the set

   $$firstYear : \mathbb{P}\ PERSON$$

   the set of all first-year students, define the set of all first-year students who are taking the module *programming*.

6.4 Source, target, domain and range of a relation

Suppose the relation *taking* has the value

$$\{Alice \mapsto C{+}{+}, Chris \mapsto C{+}{+}, Chris \mapsto Z, Sandra \mapsto Z, Sandra \mapsto Database\}$$

We can represent this relation as a picture as shown in Figure 6.1.

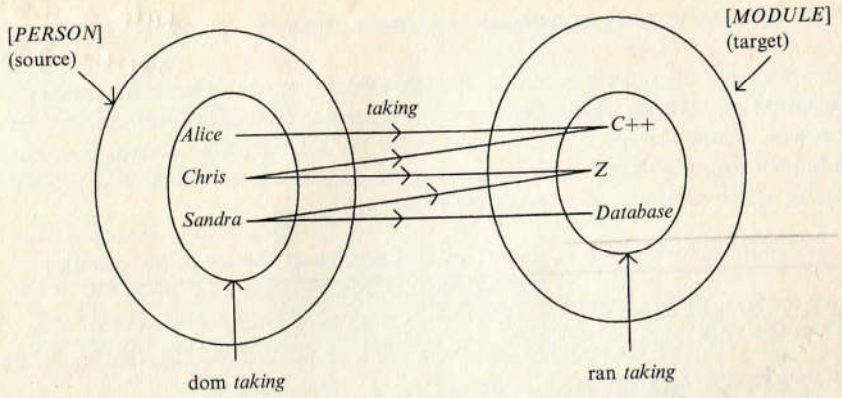

Figure 6.1 The relation *taking*

The types (maximal sets) *PERSON* and *MODULE* named in the declaration of the relation *taking* are sometimes called the *source* and *target* sets respectively. The source is the set from which the first element of each maplet in a given relation must be drawn, and the target is the set from which the second element in each maplet in a given relation must be drawn. Each maplet in the relation *taking* is represented by an arrow in Figure 6.1.

The *domain* of the relation *taking*, referred to in Z as dom *taking*, is that subset of the source set whose members have at least one arrow coming out of them. In other words, the set of all people who occur as the first element of at least one of the maplets in *taking*.

$$\text{dom } taking = \{p : PERSON, m : MODULE \mid p \mapsto m \in taking \bullet p\}$$

For the above example

$$\text{dom } taking = \{Alice, Chris, Sandra\}$$

The *range* of *taking*, referred to as ran *taking*, is that subset of the target set whose members have at least one arrow entering them; in other words the set of all modules which occur as the second element of at least one of the maplets in *taking*.

$$\text{ran } taking = \{p : PERSON, m : MODULE \mid p \mapsto m \in taking \bullet m\}$$

For the above example

$$\text{ran } taking = \{C++, Z, Database\}$$

6.5 Module registration

Clearly, *taking* will be an important part of the representation of the state of our administration system. However, we are only interested in the set of people who are registered as students at the university, and the set of modules which are part of the modular degree scheme.

$$students : \mathbb{P}\ PERSON$$
$$degModules : \mathbb{P}\ MODULE$$

This places a restriction on the maplets allowed in the relation *taking*. The people doing modules must be registered as students, and the modules they are doing must be bona fide degree modules at our university.

$$\text{dom } taking \subseteq students$$
$$\text{ran } taking \subseteq degModules$$

One of the good things about Z is that using it makes us focus on the problem – to concentrate on precisely what we are trying to specify. A good notation cannot make us write correct specifications, but it does make it more *likely* that we will. We write something down in the Z notation; it only has one meaning, and in considering what we have written down we must confront the question of what it was that we were trying to say, or should have been trying to say, in the first place. At this stage, we might consider issues such as:

- is every student taking one or more modules?
- does every module have students taking it?
- can a student be registered for more than one course (different named degrees within the modular scheme, or HND and HNC courses)?
- are the sets of modules for each course disjoint?

For simplicity, we choose not to consider other courses within the university, but we do allow the possibility that a student registered on the degree scheme may not be doing any modules (the student may be intermitting, that is taking a break from the course, or may be on a sandwich placement), and that a module within the scheme may not have any students. The schema describing the state of the module registration part of the administration system is thus

ModuleReg

$$students : \mathbb{P}\ PERSON$$
$$degModules : \mathbb{P}\ MODULE$$
$$taking : PERSON \leftrightarrow MODULE$$

$$\text{dom } taking \subseteq students$$
$$\text{ran } taking \subseteq degModules$$

Note that because relations are sets, all the set operations that we have met so far (union, intersection, membership, etc.) may be applied to them. In fact, in this and the following chapters, we will be considering a hierarchy of different types of sets, each of which is a more restricted form of its predecessor, and each of which is associated with successively richer sets of operations, comprising those operations 'inherited' from the predecessors together with new operations specific to that structure. Thus relations are a special type of set, functions (Chapter 7) are a special type of relation, and sequences (Chapter 9) are a special type of function.

Exercises 6.3

1. Give a Z expression for the set of all students who are not taking any modules.
2. Give a Z expression for the set of all degree modules which have no students.
3. Suppose that there is a maximum of n people allowed to study a module. Give a Z expression for the set of all modules which are full; that is, which have precisely n people taking them.
4. Write down the additional predicate required in the schema *ModuleReg*, to incorporate the maximum limit of n people described at 3 above.
5. Give an appropriate type for a variable *matches*, which represents the draw for a round in a tennis tournament. What invariant predicates would have to be associated with *matches* to ensure that each person in the draw is only in one match, and is not playing against themselves?

6.6 Relational image

Given a relation $R : A \leftrightarrow B$ and a set $S \subseteq A$, the *relational image* of S in R is defined as follows:

$$R(S) = \{b : B \mid \exists a : A \bullet a \in S \land a \mapsto b \in R\}$$

In other words, the set of all those members of ran R that are related by R to members of S.

For example, the set of all modules being studied by student p is

$$taking(\{p\})$$

For the value of *taking* shown in Figure 6.1

$$taking(\{Alice, Chris\}) = \{C++, Z\}$$

6.7 Inverse of a relation

Given a relation $R : A \leftrightarrow B$, the *inverse* of R is defined as

$$R^{-1} = \{a : A, b : B \mid a \mapsto b \in R \bullet b \mapsto a\}$$

In other words, R^{-1} is the relation obtained by reversing the order of each of the maplets in R, or to put it another way, by reversing the direction of all the arrows in the picture of R. The *type* of the inverse relation is therefore

$$R^{-1} : B \leftrightarrow A$$

For example, for the value of *taking* shown in Figure 6.1 we have

$$taking^{-1} = \{C++ \mapsto Alice,\ C++ \mapsto Chris,\ Z \mapsto Chris,\ Z \mapsto Sandra,\ Database \mapsto Sandra\}$$

Exercises 6.4

1. Write a Z expression for the set of all students taking module m.
2. Write a Z expression for the set of all students who are taking at least one module which student s is taking.
3. Write an alternative answer to question 4 of Exercises 6.3, using relational image and inverse.

6.8 Operations

We will now consider the successful cases of some of the operations which the university's administration staff will wish to perform on this part of the system state. As before, some of these will cause the state to change and some will simply interrogate the state for information.

Adding a new student to the university

For a person $p?$ to become a student at the university, the precondition is that s/he is not one already!

AddStudent

Δ *ModuleReg*
$p? : PERSON$

$p? \notin students$
$students' = students \cup \{p?\}$
$degModules' = degModules$
$taking' = taking$

Registering a student for a module

In order that a person $p?$ may become registered for a module $m?$ we require that:

1. $p?$ is a student at the university.
2. $m?$ is a valid module in the degree scheme.
3. $p?$ is not already registered for $m?$.

We can add the maplet $p? \mapsto m?$ to the relation *taking* by placing it in a singleton set and taking the union of this set with *taking*. This technique is very commonly used when writing operation specifications using relations. The schema is as follows:

RegForModule

Δ *ModuleReg*
$p? : PERSON$
$m? : MODULE$

$p? \in students$
$m? \in degModules$
$p? \mapsto m? \notin taking$
$taking' = taking \cup \{p? \mapsto m?\}$
$students' = students$
$degModules' = degModules$

Exercises 6.5

1. Write Z schemas for operations to:

(i) remove a student from the university;
(ii) withdraw a student from a module (Hint: Use the set difference operator \.);
(iii) add a new module to the degree scheme;
(iv) remove a module from the degree scheme.

2. Let there be a limit of n students who may be registered for any single module. Modify the *RegForModule* schema to take account of this extra constraint, and use the schema calculus to define a total version of this operation, that is one which deals appropriately with all possible exceptions to the schema preconditions.

6.9 Domain and range restriction and anti-restriction

Given a relation $R : A \leftrightarrow B$ and a set $S \subseteq A$, R *domain restricted* to S may be defined as follows:

$$S \lhd R = \{a \mapsto b : A \times B \mid a \mapsto b \in R \wedge a \in S\}$$

In other words, $S \lhd R$ defines a relation which is the result of removing from R all maplets with first elements that are not members of S. For example, suppose we have the set $firstYear \subseteq students$, the set of all students in the first year of their degree course. The subset of *taking* which relates just first-year students to the modules they are taking is given by

$$firstYear \lhd taking$$

The operation of domain restriction is complemented by that of domain anti-restriction. R *domain anti-restricted* to S may be defined as follows:

$$S \lhd\!\!\!\!- R = \{a \mapsto b : A \times B \mid a \mapsto b \in R \wedge a \notin S\}$$

In other words, $S \lhd\!\!\!\!- R$ defines a relation which is the result of removing from R all maplets with first elements that *are* members of S. For example, the subset of *taking* which relates all degree students that are not in their first year to the modules they are taking is

$$firstYear \lhd\!\!\!\!- taking$$

The above operators restrict a relation to those maplets whose *first element* is or is not a member of a given set. We can also define similar operators which restrict a relation to those maplets whose *second element* is or is not a member of a given set, namely *range* restriction and anti-restriction respectively. Given the relation R as above and the set $T \subseteq B$, then R *range restricted* to T may be defined as follows:

$$R \rhd T = \{a \mapsto b : A \times B \mid a \mapsto b \in R \wedge b \in T\}$$

In other words, $R \rhd T$ defines a relation which is the result of removing from R all maplets with second elements which are not members of T. Finally, R *range anti-restricted* to T may be defined as follows:

$$R \rhd\!\!\!\!- T = \{a \mapsto b : A \times B \mid a \mapsto b \in R \wedge b \notin T\}$$

In other words, $R \rhd\!\!\!\!- T$ defines a relation which is the result of removing from R all maplets with second elements which *are* members of T. For example, suppose we have the set

$$progMods \subseteq degModules$$

the set of all modules which involve programming. The subset of *taking* which relates students to just their modules which involve programming is

$$taking \rhd progMods$$

and the subset of *taking* which relates students to just their modules which don't involve programming is

$$taking \rhd progMods$$

As a further example, suppose the relation *taking* has the value given in Section 6.4 above.

$$taking = \{Alice \mapsto C{+}{+}, Chris \mapsto C{+}{+}, Chris \mapsto Z, Sandra \mapsto Z, Sandra \mapsto Database\}$$

Then the following are all true:

$$\{Alice, Chris\} \lhd taking = \{Alice \mapsto C{+}{+}, Chris \mapsto C{+}{+}, Chris \mapsto Z\}$$
$$\{Alice, Chris\} \lhd taking = \{Sandra \mapsto Z, Sandra \mapsto Database\}$$
$$taking \rhd \{Z\} = \{Chris \mapsto Z, Sandra \mapsto Z\}$$
$$taking \rhd \{Z, C{+}{+}\} = \{Sandra \mapsto Database\}$$

Exercises 6.6

1. Given the relation

$$R = \{1 \mapsto 1, 2 \mapsto 4, 3 \mapsto 9, 4 \mapsto 16, 5 \mapsto 25\}$$

and the set $S = \{1, 4, 5\}$, simplify the value of each of the following expressions:
(i) $S \lhd R$
(ii) $R \rhd S$
(iii) $S \lhd R \rhd S$
(iv) $(R \rhd \mathrm{dom}\ R)^{-1} \rhd S$

2. Given the set *firstYear* as above, write down two ways to describe in Z the set of all modules which first-year students are taking. (Note that for many problems there is more than one reasonable way to specify a solution. Trying to think of alternative solutions to any given problem will improve your familiarity with the notation and its application.)
3. Given the set *progMods* as above, write down a Z expression for the set of all students who aren't studying any programming.

4. Look back through the other exercises in this chapter and see whether they could have been answered using the restriction and anti-restriction operators.

6.10 Composition and transitive closure

The *forward composition* of relations $R: A \leftrightarrow B$ and $S: B \leftrightarrow C$ is denoted by

$$R ; S$$

and is the relation of type $A \leftrightarrow C$ defined as follows:

set construction

$$R ; S = \{a : A; c : C \mid (\exists b : B \bullet a \mapsto b \in R \land b \mapsto c \in S) \bullet a \mapsto c\}$$

The important point is that the target set of R is a subset of the source set of S. (?)

For example, let

$$A = \{1, 2, 3, 4\}$$
$$B = \{a, b, c, d\}$$
$$C = \{p, q, r, s\}$$

and

$$R = \{1 \mapsto a, 1 \mapsto b, 2 \mapsto b, 4 \mapsto d\}$$
$$S = \{a \mapsto p, b \mapsto q, c \mapsto q, c \mapsto r\}$$

Then

$$R ; S = \{1 \mapsto p, 1 \mapsto q, 2 \mapsto q\}$$

This is illustrated in Figure 6.2

Informally, a maplet $x \mapsto y$ is a member of $R ; S$ iff you can get from x to y in the picture by following two consecutive arrows.

If a relation is *homogeneous*, it can be composed with itself. For example, when someone writes an academic paper, they include a list of references, citing other papers, the content of which they have referred to in their own paper. Let the set of all academic papers be

$$[PAPER]$$

and let *cites*: $PAPER \leftrightarrow PAPER$ be the relation such that x *cites* y (where x and y are papers) has the obvious meaning. Then

$$x\ cites ; cites\ y$$

composition of cites with cites

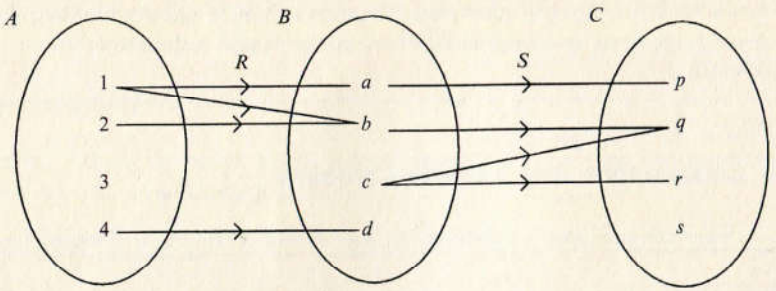

Figure 6.2 Composition of relations R and S

would mean that paper x cites a paper which cites paper y and

$$x \; cites ; cites ; cites \; y$$

would mean that x cites a paper which cites a paper which cites y, and so on. There is a shorthand notation to express such multiple composition, as follows.

The *identity relation* on a set X is

$$\mathrm{id}\, X = \{x : X \bullet x \mapsto x\}$$

that is, the relation which maps every element of X to itself. For a relation $R : X \leftrightarrow X$

$$\begin{aligned} R^0 &= \mathrm{id}\, X \\ R^1 &= R \\ R^2 &= R ; R \\ R^3 &= R ; R ; R \\ &\vdots \end{aligned}$$

The *transitive closure* of R, denoted by R^+, is the relation obtained by taking the union of all of these relations except R^0, that is

$$R^+ = \bigcup \{n : \mathbb{N} \mid n > 0 \bullet R^n\} \qquad = R^1 \cup R^2 \cup R^3 \cdots$$

The *reflexive transitive closure* of R, denoted by R^*, is obtained by including R^0 in the union:

$$R^* = \bigcup \{n : \mathbb{N} \mid n \geqslant 0 \bullet R^n\} = R^+ \cup R^0$$

So $x \; cites^+ \; y$ means that x cites y either directly or indirectly via one or more other papers. In other words, there exists at least one sequence of papers, beginning with x and ending with y, in which each successive paper (except the last one) cites the next in the sequence. Here we are using the term 'sequence' in its informal sense. In Chapter 9 we will give a formal description of sequences, which are a powerful tool in writing Z specifications. Note that the relation $cites^*$ would also relate every paper to itself, although such a citation is not normal practice in the academic community!

Exercises 6.7

1. Write a Z expression for the set of all papers cited directly or indirectly by paper x.
2. Write a Z expression for the set of all papers which cite other papers (directly or indirectly) but are not themselves cited (directly or indirectly).
3. Write a Z expression which states that if any paper cites another (directly or indirectly), then the second one may not cite the first (directly or indirectly).
4. The university also wishes to keep track of which modules each student has completed. Extend the state schema of the module registration system specification to incorporate this feature. You might also like to think about any new operations which would be necessary.
5. A student may not retake a module which s/he has already completed. Modify the schema which registers a student for a module to allow for this restriction.
6. In a modular degree scheme such as the one described above, some modules may only be taken if one or more prerequisite modules have been passed. Extend the state schema to incorporate a prerequisite structure. You may also wish to specify some operations for the new state, for example an operation to enquire which modules are prerequisite for a given module.