Functions

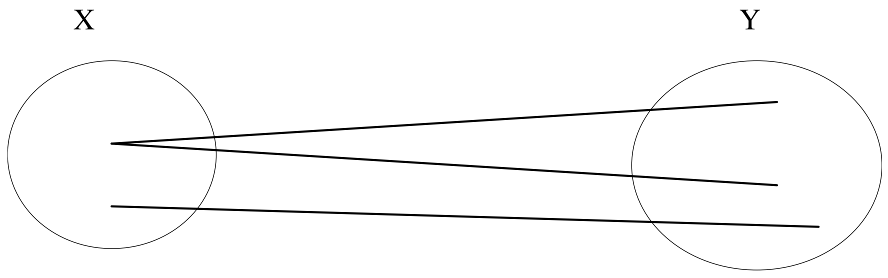
A function is a relation with extra constraints. The extra constraint is that for each element x in the domain of a function *f*, there must be only one mapping containing the element x.
Thus all we have seen about relations can be used with functions.

Functions and relations are both represented in Z as sets of ordered pairs. A function from set X to a set Y and a relation between X and Y are both of type ℙ(X x Y) or (X $\leftrightarrow Y$). A function differs in that we restrict the legal sets of pairs constituting a function so that there must be only one occurrence of each X element in the domain, i.e. members of the domain map to a unique element in the range. A function is a special case of a relation.

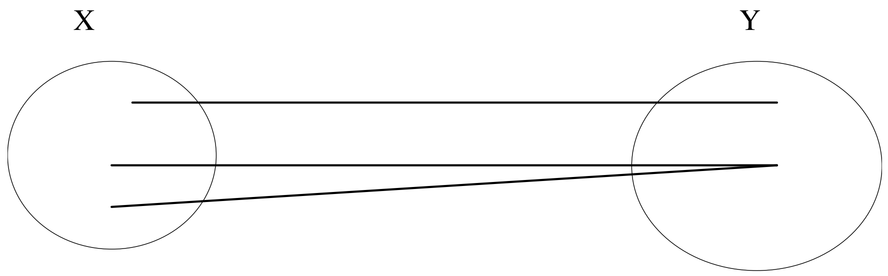
Definition of a function

Formally, we can describe the functionality requirement by the following property for relation R

$(x \mapsto y) \in R \wedge (x \mapsto z) \in R \Rightarrow y = z$



This is not a function



This is a function.

Example

ChildFather : PERSON $\leftrightarrow$PERSON

ChildFather = {Kate $\mapsto$ Michael, Deirdre $\mapsto$ Michael, Martin $\mapsto$ Brendan }

is a function (many-to-one) while

ChildParent : PERSON $\leftrightarrow$PERSON

ChildParent = {Kate $\mapsto$ Michael, Kate $\mapsto$ Breda, Martin $\mapsto$Brendan, Martin $\mapsto$ Anne}

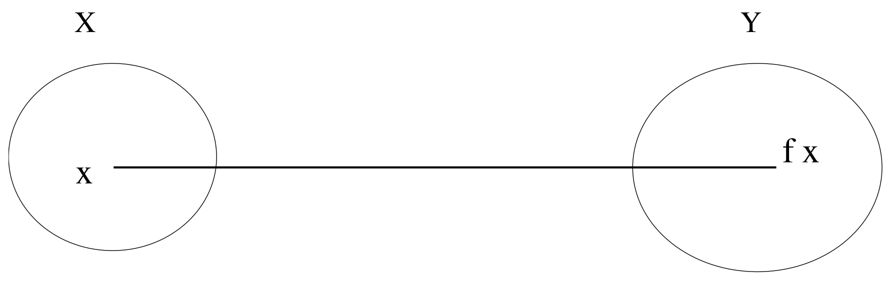
is just a relation (many-to-many)

It is allowed for two distinct values in the domain to be mapped onto the same element in the range of the function. Note that this means that the inverse of a function is not always itself a function (because the one value mapped to, from the two distinct values in the domain of the original function would map to the two original distinct domain values).

Function Application

Because there is exactly one mapping from each element in the domain, this gives us the capability to apply a function to a value. The value of *f* applied to x is the value in the range of the function *f* corresponding to the value x in its domain. The application is meaningless if the value of x is not in the domain of *f*. The application of the function *f* to the value x (called its argument) is written:

f x or *f*(x) (i.e. brackets not necessary)



For example, application of ChildFather to Kate yields a unique person.
ChildFather Kate = Michael.

ChildParent, on the other hand, maps Kate to Michael and Breda, so application is undefined. To follow the relation from Kate, we use the relational image of the set containing Kate. This yields a set containing persons.

ChildParent $(\!|$ { Kate } $|\!)$ = {Michael, Breda}

Properties of functions:

The following are all different kinds of functions:

- partial and total
- injective, surjective, bijective
- finite and infinite

The particular properties to be associated with a function for a given type are indicated in Z by the kind of arrow used in the function declaration. For example, f: X $\rightarrow$ Y declares a total function from X to Y while f:X $\nrightarrow$ Y declares a partial function from X to Y. The arrows are shorthand for the declaration of a type and a predicate restricting the allowed values.

Partial and total functions

Partial functions

A partial function is the most general kind of a function. All functions are partial functions. Partial functions from some set X map some or all of the elements of X.
A partial function fp from X to Y is declared in Z as :

fp : X $\nrightarrow$ Y

X $\nrightarrow$ Y denotes the set of partial functions from X to Y.

For example, if we look at an identity number scheme where not necessarily every person has an identity number. This is declared as

identityNo : PERSON $\nrightarrow \mathbb{N}$

(Note that there is probably a need for constraining the function so that no two people have the same identity number. That is not implied in the above declaration.)

Formally, we can generically define the partial function arrow.

$$X \nrightarrow Y ==$$
$$\{R : X \leftrightarrow Y \mid \ (\forall x : X;\ y, z : Y \bullet x \mapsto y \in R \land x \mapsto z \ \in R \Rightarrow y = z)\ \}$$

Note that this is the set of functions from X to Y.

Precluding elements outside domain

Most functions in Z-specifications are only partial. This means that function application won't always be defined. We need to preclude the application of a partial function to elements outside its domain. A useful phrase in specifications is :

$\forall$x: dom f • ...

This ensures that you're not applying a function to an element outside its domain. Use this whenever possible!

Total functions

A total function is one where there is a mapping for every possible value of x, so *f* x is always defined. Formally, total functions are functions whose domain is the whole of their source.

$X \rightarrow Y == \{ f: X \nrightarrow Y \mid \mathrm{dom}\ f = X \}$

$ft : X \rightarrow Y$ is an example of the declaration of a total function in Z.

For example, an age function (from PERSON to $\mathbb{N}$) would be total (every one has an age), so:

$age : PERSON \rightarrow \mathbb{N}$

The ChildFather we looked at earlier is total, because each child has one father.

$ChildFather : PERSON \rightarrow PERSON$

The greater use of partial functions comes from the need to treat mappings as dynamic entities. If a function is declared as total, then each time an mapping is changed, we need to check does this effect the totality of the function. For example, if we have the *owner* function to model ownership of objects in a database:

$owner : OBJECT \rightarrow PERSON$

Because of the totality of the function, objects cannot be deleted from the mapping. It is therefore more general to use a partial function in this case:

$owner : OBJECT \nrightarrow PERSON$

Therefore, unless the function is obviously total i.e. the totality of the function is very important in the meaning of the function, use partial functions.

Injective, surjective and bijective functions

Injective functions

(One-to-one)
An injection or an injective function is one which maps different values of the source onto different values of the target.. It is also known as one-to-one. Note that the inverse of an injective is itself a function and indeed an injective function. In fact, if a function is not an injective function then the inverse is in general a relation.
For instance, the function identityNo is injective if we propose that each identity number is unique.

We write it as follows:

identityNo : PERSON $\rightarrowtail\!\!\!\!\!+$ $\mathbb{N}$

This is a partial injective function (not everyone has an identity number) .If we added the constraint that every Person must have an associated identity number, then it becomes a total injective function.

identityNo$_1$: PERSON $\rightarrowtail$ $\mathbb{N}$

Formally, we can define the meaning of the injective arrows, partial first.

Partial injective function
$X \rightarrowtail\!\!\!\!\!+ Y == \{f: X \nrightarrow Y \mid (\forall x_1, x_2: dom\, f \bullet f\ x_1 = f\, x_2 \Rightarrow x_1 = x_2)\ \}$

Total injective function
$X \rightarrowtail Y == \{f: X \rightarrow Y \mid (\forall x_1, x_2: dom\, f \bullet f\, x_1 = f\, x_2 \Rightarrow x_1 = x_2)\ \}$

Example of a partial injective function
decrement is a function which subtracts one from the positive natural numbers.
decrement : $\mathbb{N} \rightarrowtail\!\!\!\!\!+ \mathbb{N}$
Note this is not total because there the function cannot be applied to 0.

Example of a total injective function
decrementtot : $\mathbb{N}_1 \rightarrowtail \mathbb{N}$
is a function from positive natural numbers to the natural numbers, so this it total (1 -1 = 0 , and $0 \in \mathbb{N}$).

Surjective functions

A surjection or a surjective function is a function for which there is a value in the range for every value in the target. Given a function *f* from X to Y,

ran f = Y

Surjective functions can be partial or total.
They are written

$\twoheadrightarrow$ partial surjective function

$\twoheadrightarrow$ total surjective function

For example, there are 20 staff PC's in the college, and 200 staff. Staff members are assigned the use of a particular machine, depending on needs, department etc. All staff PC's are used. Staff may be sharing PC's.

HasUseOf : STAFF $\twoheadrightarrow$ PC

This is a surjective function because all PC's are assigned to (possibly more than one) STAFF's.

If all staff members were assigned a PC and still all PC's were used, then this would yield

EveryoneHasUseOf: STAFF $\twoheadrightarrow$ PC

Formally :

$$X \twoheadrightarrow Y == \{ f: X \rightarrow Y \mid ran\, f = Y\}$$
$$X \twoheadrightarrow Y == \{ f: X \rightarrow Y \mid ran\, f = Y\}$$

Bijective functions

A bijection or a bijective function is one which maps every element of the source on to every element of the target in a one-to-one relationship. It is injective, total and surjective.

For example a function that maps uppercase letters to their lowercase equivalents:

lowercase : UPPERCASE $\rightarrowtail\!\!\!\!\twoheadrightarrow$ LOWERCASE

This function is bijective because

- for each uppercase letter there is exactly one corresponding lowercase letter.
- each lowercase letter has exactly one corresponding uppercase letter

A bijection between two sets is a very strong condition and is sometimes called an isomorphism; the sets UPPERCASE and LOWERCASE are said to be isomorphic and are essentially the same. The only difference between them is the names of their elements.

Finite Functions

It is useful to be able to ignore the issue of whether the sets being used to model the system are finite or infinite. The use of infinite sets avoids the need to address implementation issues at this stage. However, sometimes, it may be useful to stress that a function from an infinite set may have a finite domain.

The set of finite partial functions from X to Y is denoted

X ⇻ Y

The set of finite partial injections from X to Y is denoted

X ⤘ Y

Summary of notation:

- the long arrow indicates a function
- the double arrowhead indicates surjection
- the arrow-shaped tail indicates injection
- a central bar indicates that the function is partial
- a double central bar indicates that a partial function is finite

Constant functions and axiomatic definitions

Some functions are used as a means of providing a value, given a parameter or parameters. These are usually functions which maintain a constant mapping from their input parameters to their output values. If the value of the mapping is known, a value can be given to the function by an **axiomatic definition** e.g. to define the function square:

$$\begin{array}{|l}
\mathit{square} : \mathbb{N} \rightarrow \mathbb{N} \\
\hline
\forall n : \mathbb{N} \bullet \mathit{square}\ n = n * n
\end{array}$$

Several functions may be combined into one axiomatic definition

$$\begin{array}{|l}
\mathit{square} : \mathbb{N} \rightarrow \mathbb{N} \\
\mathit{cube} : \mathbb{N} \rightarrow \mathbb{N} \\
\hline
\forall n : \mathbb{N} \bullet \mathit{square}\ n = n * n \\
\forall n : \mathbb{N} \bullet \mathit{cube}\ n = n * n * n
\end{array}$$

Functional overriding

This is the same as overriding in relations. A function can be modified by adding mapping pairs to it or by removing pairs. It can also be changed so that for a particular set of values in the domain, the function maps to different values in the range. This is done by using the relational override operator.

It is denoted by :

$$f_1 \oplus f_2$$

Given $f_1, f_2 : X \nrightarrow Y$

$f_1 \oplus f_2 = f_2 \cup (\mathrm{dom}\ f_2 \lhd f_1)$

It is useful for modelling a small change to an existing function.

Example:

$bankBalance, bankBalance' : PERSON \nrightarrow \mathbb{N}$

$bankBalance' = bankBalance \oplus \{Eimear \mapsto 100\}$

Note that the function overriding

- updates (changes) the mapping if it exists
- adds the mapping if it's not already there

Further examples of function overriding:

The recorded age of a person p? is to be increased by one.

$age \oplus \{ p? \mapsto age\ p? + 1\}$

he function age is overridden by the function with only p? in its domain which maps to the former value plus one (using function application).

Functions of Several arguments

A function is just a set of pairs. So a function can only have one argument - the first member of the pair. If the function really has several arguments, these are combined into a single element.

wedding = { (Gerard, Catherine) ↦ 17July, (Pat, Mary) ↦ 31August, (Brendan, Anne) ↦ 30September) }

Note that we're mixing the (x, y) and $x \mapsto y$ notation for clarity.

Curried functions
Sometimes there is a more useful way of expressing functions of more than one argument. For instance if a person can have a bank account at several different banks, we might model the information as follows:
$\text{balance}_1 : \text{PERSON x BANK} \rightarrow\!\!\!\!\!\!+\; \mathbb{N}$

For example

$\text{balance}_1 = \{ (\text{Smurfit, AIB}) \mapsto \mathit{100000}, (\text{Smurfit, BOI}) \mapsto \mathit{500000}, (\text{LarryLecturer, AIB}) \mapsto \mathit{50} \}$

This doesn't naturally answer the question for Smurfit 'how much money do I have in all my accounts? '

Another way of modelling the above information :

$\text{balance}_2 : \text{PERSON} \rightarrow\!\!\!\!\!\!+\; \text{BANK} \rightarrow\!\!\!\!\!\!+\; \mathbb{N}$

$\text{balance}_2 = \{ \text{Smurfit} \mapsto \{ \text{AIB} \mapsto \mathit{100000}, \text{BOI} \mapsto \mathit{500000}\},$
$\quad (\text{LarryLecturer} \mapsto \{ \text{AIB} \mapsto \mathit{50} \} \}$

This function is a function from a person to all the accounts that person has.