Functions

Mairead Meagher

Waterford Institute of Technology,

2017

Waterford Institute *of* Technology

INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Functions

- A function is a relation with extra constraints.
- 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

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

- There must be only one occurrence of each X element in the domain, i.e. members of the domain map
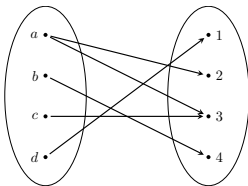
What is a Function
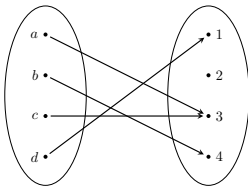
Figure 1: This is not a function

Figure 2: This is a function

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

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

Example

Given:

$[PERSON, MODULE]$

$studies : PERSON \leftrightarrow MODULE$

    maps a student to the modules they are currently studying

             e.g.

$studies = \{nathan \mapsto prog, nathan \mapsto database,$
$\qquad\qquad viola \mapsto maths, viola \mapsto prog\}$

is a relation(many-to-many)

    $inSemester : PERSON \leftrightarrow \mathbb{N}$

             maps a student to their semester number

    $inSemester = \{oliver \mapsto 6, nathan \mapsto 6, gabriel \mapsto 8\}$

is a function (many-to-one)

- Two distinct values in the domain to be mapped onto the same element in the range of the function (many-to-one).
- this means that the inverse of a function is not always itself a function

Function Application

- We have 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 - thus not defined.
- The application of the function f to the value x (called its argument) is written:

$f\ x$ or

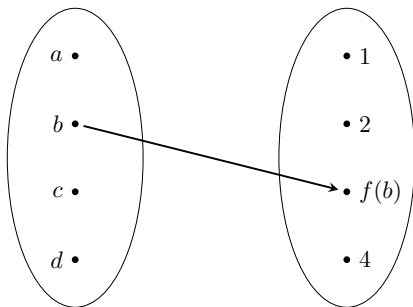$f(x)$ (i.e. brackets not necessary)

Function Application

Figure 3: Function Application

For example, application of **inSemester** to oliver yields a unique semester number.

$$inSemester(oliver) = 6$$

studies, on the other hand, maps nathan to prog and database, so application is undefined. To follow the relation from nathan, we use the **relational image** of the set containing nathan. This yields a set containing MODULEs.

$$studies \mathbin{(\!|} \{nathan\} \mathbin{|\!)} = \{prog, database\}$$

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,

$$f : X \rightarrow Y$$
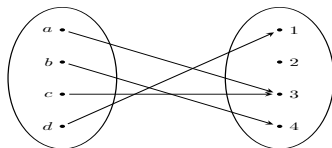
Partial Functions

Figure 4: Partial Function

- Most general kind of a function.
- All functions are partial
- Partial functions from some set X map some or all of the elements of X.

Partial Functions

- A partial function fp from X to Y is declared in Z as :

  $$fp : X \nrightarrow Y$$

- $X \nrightarrow Y$ denotes the set of all possible 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

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

- Formally, we can generically define the partial function arrow.

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

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

Precluding elements outside the domain

- Most functions in Z-specifications are partial.
- This means that function application wont always be defined.
- We need to preclude the application of a partial function to elements outside its domain.
- A useful phrase in specifications is :

$$\forall x : dom\ f \ldots$$

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

(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.)
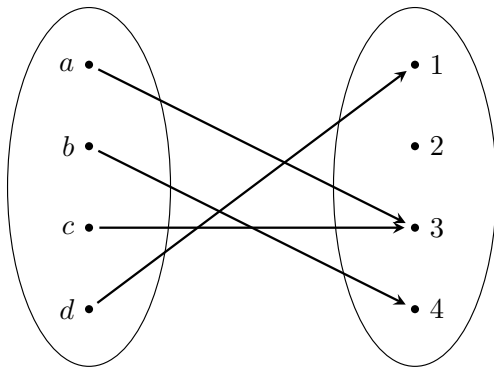
Figure 5: Total Function

- 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.

$$X \to Y == \{f : X \nrightarrow Y \mid dom\ f = X \bullet f\}$$

-

$$ft : X \to Y$$

is an example of the declaration of a total function in Z.

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

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

- A function to map a number to its double (we call these constant or mathematical functions):

    double : $\mathbb{Z} \rightarrow \mathbb{Z}$

Total Functions - Use with care

- Use partial functions when we need to treat mappings as dynamic entities.
- 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. Better to use a partial function in this case:

    $owner : OBJECT \nrightarrow PERSON$

- Unless the function is obviously total, use partial functions.
- We will see that constant functions (e.g. mathematical functions) are the examples of total functions that we will use.
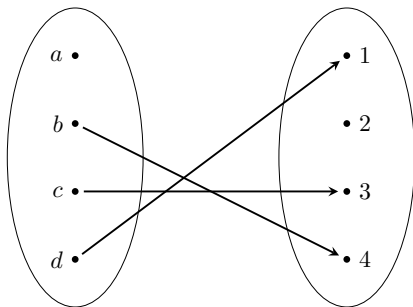
Figure 6: Injective Function

- An injection, an injective function, or a one-to-one function is one which maps different values of the source onto different values of the target.

- 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.

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

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

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

- Total injective function

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

Examples of partial, total injective functions

partial **decrement** is a function which subtracts one from the positive natural numbers.

$$decrement : \mathbb{N} \rightarrowtail\!\!\!\!\rightarrow \mathbb{N}$$

- Note this is not total because there the function cannot be applied to 0.

total **decrementtot** is a function which subtracts one from all the positive natural numbers to all the natural numbers.

$$decrementtot : \mathbb{N}_1 \rightarrowtail \mathbb{N}$$

- This is a function from positive natural numbers to the natural numbers, so it is total (1 -1 = 0 , and $0 \notin \mathbb{N}_1$).
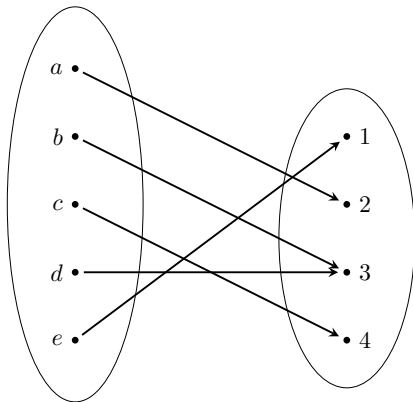
Figure 7: Surjective Function

- 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, surjective functions have the property that

    ran f = Y

- Surjective functions can be partial or total.
- They are written

      ⟶↠partial surjective function
      ⟶↠total surjective function

Surjective functions - An Example

- There are 100 student lockers in the college, and 200 students.
- Students are assigned the use of a particular locker, depending on needs.
- All lockers are used. Students may share lockers.
- Not all students are assigned a locker.

[*STUDENT*, *LOCKER*]

hasUseOf : *STUDENT* $\twoheadrightarrow$ *LOCKER*

Surjective functions - An Example

- This is a partial surjective function because all lockers are assigned to (possibly more than one) student.
- If all students were assigned a locker and still all lockers were used, then this would yield

  everyoneHasUseOf : *STUDENT* $\twoheadrightarrow$ *LOCKER*

- However, this means that

  'Every STUDENT who ever lived, lives or ever will live'

  is assigned a locker.
- This is not usually what we want.

Formally,

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

Note: There should be a vertical line through the arrow for the partial surjection.

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* ↣ *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

$$X \nrightarrow Y$$

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

$$X \rightarrowtail\mkern-14mu\rightarrow Y$$

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 - 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**:

$$square : \mathbb{N} \to \mathbb{N}$$
$$\forall\, n : \mathbb{N} \bullet square\ n = n^2$$

- Several functions may be combined into one axiomatic definition

$$square : \mathbb{N} \to \mathbb{N}$$
$$cube : \mathbb{N} \to \mathbb{N}$$

$$\forall\, n : \mathbb{N} \bullet square\ n = n^2$$
$$\forall\, n : \mathbb{N} \bullet cube\ n = n^3$$

Constant Functions - Set Comprehension

- Constant functions can also be written/specified using set comprehension

- Looking at the square function as seen in the previous slide,

  $$square \; == \; \{n : \mathbb{N} \bullet (n, n^2)\}$$

- **Note** that the totality of the function or otherwise can be implied from the need or otherwise of any constraint.

- Both these specifications specify the same function.

- 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 :

    $f1 \oplus f2$

- Functional Overriding is used for modelling a small change to an existing function.

Example Given the following and using the before $\rightarrow$ after state notation :

> $bankBalance, bankBalance' : PERSON \nrightarrow \mathbb{Z}$
> . . .and. . .
> $bankBalance = \{\ldots, markus \mapsto 1000, mairead \mapsto 100, \ldots\}$
> Then the following two statements specify the effect
> of functional overriding
>
> $bankBalance' = bankBalance \oplus \{markus \mapsto 500\}$
> $bankBalance' = \{\ldots, markus \mapsto 500, mairead \mapsto 100, \ldots\}$

- Given

$$f, g : X \nrightarrow Y$$

- Write down the specification of

$$f \oplus g$$

Given

$$f, g : X \nrightarrow Y$$

$$f \oplus g = g \ \cup \ (dom \ g \lhd f)$$

Note that the function overriding

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

Given

$$age : PERSON \rightarrow \mathbb{N}$$
$$age = \{\ldots, barack \mapsto 55, \ldots\}$$

$$age' = age \oplus \{barack \mapsto 56\}$$
then
$$age' = \{\ldots, barack \mapsto 56, \ldots\}$$

But: Is there anyway of 'formulating'

It's Barack's birthday - increase his age by one.

?

Try this as an exercise. (Hint : use function application.)

Solution:

$$age' = age \oplus \{barack \mapsto age(barack) + 1\}$$

More generally: The recorded age of a person **p?** is to be increased by one. (Note we would need to be sure that $p \in \mathrm{dom}\, age$)

$$age' = age \oplus \{p? \mapsto age(p?) + 1\}$$

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

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.

$wedding : (PERSON \times PERSON) \nrightarrow \mathbb{N}$
then
$wedding = \{(brad, angelina) \mapsto 2014, (angela, joachim) \mapsto 1968,$
$\qquad (homer, marge) \mapsto 1985)\}$

- Note that were mixing the (x, y) and $x \mapsto y$ notation for clarity.
- Note that this model does not allow the same couple to marry twice!

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:

$$balance : (PERSON \times BANK) \nrightarrow \mathbb{Z}$$

For example:

$$balance = \{(georgShaeffler, NRW) \mapsto 100000,$$
$$(suzanneKlatten, Commerzbank) \mapsto 500000,$$
$$(suzanneKlatten, DeutscheBank) \mapsto 1000000,$$
$$(LarryLecturer, Volksbank) \mapsto 50\}$$

This doesn't naturally answer the question for suzanneKlatten

'how much money do I have in all my accounts?'

Another way of modelling the above information :

$$balance_2 : PERSON \nrightarrow (BANK \nrightarrow \mathbb{Z})$$
$$balance_2 = \{\dots$$
$$georgShaeffler \mapsto \{(NRW, 100000)\},$$
$$suzanneKlatten \mapsto \{(Commerzbank, 500000),$$
$$(DeutscheBank, 1000000)\},$$
$$LarryLecturer \mapsto \{(Volksbank, 50), \dots\}$$
$$\dots\}$$

- This function is a function from a person to a function mapping all the accounts that person has to the amount in that account.
- This use of a function on the left hand side is called "Currying".

Which data model to use?

- There will usually be many ways to model the data we are analysing.

- We choose the option that best helps to simplify the operations that need to be specified.

- This is not an exact Science.

- We look for elegance and clarity of specification.

Any questions?