WEEK:2-3
SETS – RELATIONS-FUNCTIONS

Outlines

- ❑ Sets and Relations
- ❑ Functions

SETS

Mathematical objects are often seen as collections of other objects: (x,f(x)),y = 3x

- a square is a collection of points in a plane; {…} (…)
- a function is a collection of pairs linking arguments with values.

These collections are called sets. Set theory is fundamental part of Mathematics and Mathematics forms the basis of modern software engineering.

The elements in a set are not <u>ordered</u>

e.g {a,b,c } is the same as { c,b,a}

The elements are not <u>repeated</u>

e.g. {a,a,b} is the same as {a,b}

Examples:

- the passwords that may be generated using eight lower-case letters
- the collection of programs written in *C*++ that halt if run for a sufficient time on a computer with unlimited storage

Examples: $\mathbb{N} == \{0,1,2,3 \ldots .\} set\ of\ natural\ numbers$

$\mathbb{N}_1 == \{1,2,3 \ldots\}$

Examples:

i)The set of even integers is given as

$$\{x : \mathbb{Z} \mid \exists\, k : \mathbb{Z} \cdot x = 2k\}$$

1.**Questions:** Write the set of natural numbers which when divided by 7 leave a remainder of 4

MEMBERSHIP

Suppose $x \epsilon\, X$ *is a predicate, then the predicate is* ε

- True if x is in the set X
- False if x in not in the set X

Note: $\forall\, x\colon \mathbb{Z} \cdot x > 5 \Rightarrow x \in \mathbb{N}$ means x : $\mathbb{Z}$ *declares a new variable of type* $\mathbb{Z}$
$x\epsilon\mathbb{N}$ *is a predicate which is true or false depending on the value prervious* declared x

CARDINALITY

- The *cardinality* (#) of a finite set is the ***number of its elements***
- Examples:
  - – # {red, yellow, blue} = 3
  - – # {1, 23} = 2
  - – # $\mathbb{Z}$ = ?

CARTESIAN PRODUCT

- Given two sets A and B, the cartesian product of A and B, usually denoted A x B, is the set of all possible pairs (a, b) where $a \in A$ and $b \in B$

$$A \times B \equiv \{ (a, b) \mid a \in A, b \in B \}$$

- Example: PrimaryColor x Boolean:

  (red,true), (red, false),
  (blue,true), (blue, false),
  (yellow, true), (yellow, false)

In a formal description of a software system, we may wish to associate objects of different kinds: names; numbers; various forms of composite data. We may also wish to associate two or more objects of the same kind, respecting order and multiplicity, in that case Z notation is used

POWER SETS

- The power set of set S (denoted *Pow* (S) or ℙ S) is the set of all subsets of S, i.e.,

  Pow(S) ≡ {e | e ⊆ S} or ℙ S = {s | s ⊆ S}

  Note: If S has k elements, ℙ S has 2^k *elements*

- Example:
  - *Pow* ({a,b,c}) = {Ø, {a}, {b}, {c},
    {a,b}, {a,c}, {b,c},
    {a,b,c}}

Note: for any S, Ø ⊆ S and thus Ø ∈ Pow (S)

COMPUTER REPRESENTATION OF SETS

The set M is represented in a computer as a string of binary digits b_1, b_2, b_3-----b_n where n is the cardinality of the universal set **U** The bits b_i (where i ranges over the values 1, 2, … n) are determined according to the rule:

b_i = 1 if ith element of **U** is in M.
b_i = 0 if ith element of **U** is not in M.

For example, if **U** = {1, 2, … 10}, then the representation of M = {1, 2, 5, 8} is given by the bit string 1100100100 where this is given by looking at each element of **U** in turn and writing down 1 if it is in M and 0 otherwise.

Similarly, the bit string 0100101100 represents the set M = {2, 5, 7, 8}, and this is determined by writing down the corresponding element in U that corresponds to a 1 in the bit string

DATA STRUCTURES

- Objects from discrete mathematics can model *data structures*.
  - Tuples (records)
  - Relations (tables, linked data structures)
  - Functions (lookup tables, trees and lists)
  - Sequences (lists, arrays)

In a formal specification, it is often necessary to describe relationships between objects:

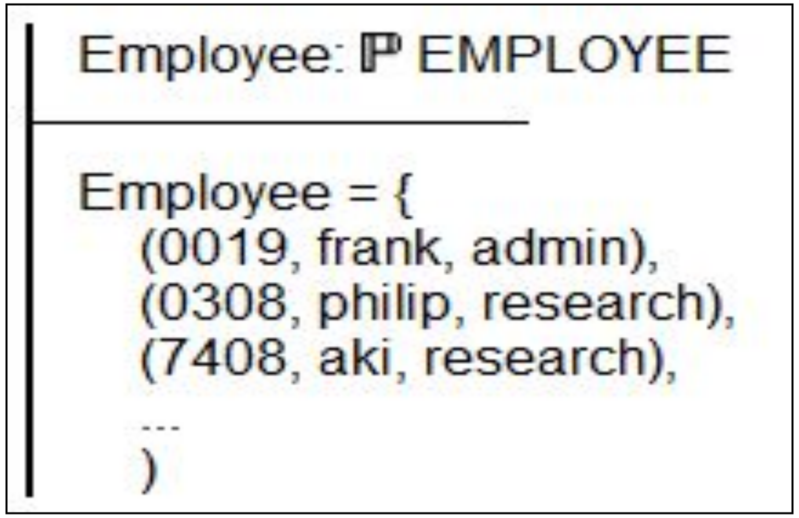
- this record is stored under that key;
- this input channel is connected to that output channel;
- this action takes priority over that one.

These relationships, and others like them, can be described using simple mathematical objects called relations

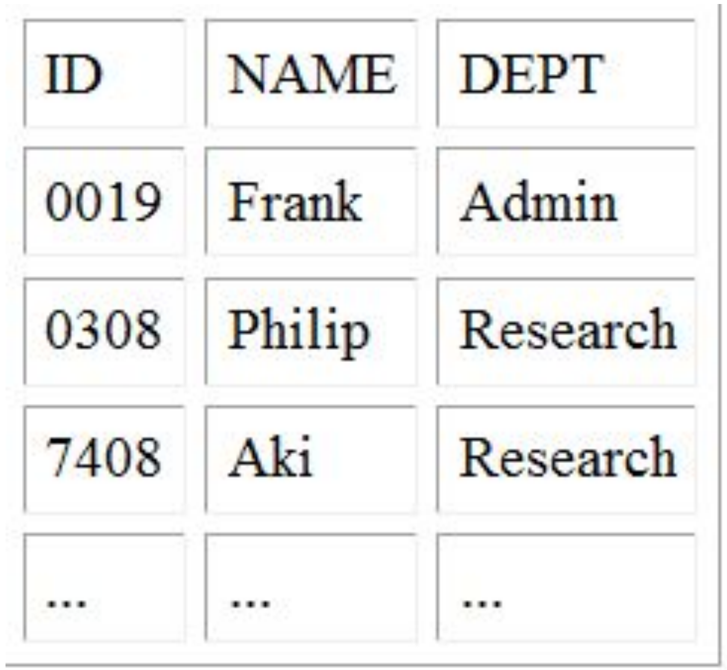
RELATIONS

- *Relations* can be sets of tuples. They can resemble *tables* or *databases*.

- In Z this can be expressed



```
Employee: ℙ EMPLOYEE
────────────────────
Employee = {
    (0019, frank, admin),
    (0308, philip, research),
    (7408, aki, research),
    ...
    )
```



ID	NAME	DEPT
0019	Frank	Admin
0308	Philip	Research
7408	Aki	Research
...	...	...

DOMAIN AND RANGE

A relation may contain a great deal of information; often, we require only a small part. To enable us to extract the information that we need the concept of **domain** and **range** functions is used

R is a relation of type $X \leftrightarrow Y$, then the domain of R is the set of elements in X related to something in Y :

$$\text{dom } R \;=\{x : X; y : Y \mid x \rightarrow y \;\square\; R \;\; o \;\; x\}$$

The range of R is the set of elements of Y to which some element of X is related:

$$\text{ran}R \;=\{x : X; y : Y \mid x \rightarrow y \;\square\; R \; o \; y\}$$

Example: The of people that drive is the domain of *drives*:

$$\text{dom}\mathit{drives} \;.\; \{\mathit{helen},\;\; \mathit{Kate}, \mathit{jim}, \mathit{John}\}$$

The set of cars that are driven is the range:

$$\text{ran } \mathit{drives} \;.\; \{\mathit{Volvo}, \mathit{Benz}, \mathit{Toyota}\}$$

BINARY RELATIONS

- *Binary relations* are sets of pairs.

- ℙ (NAME x PHONE) OR
- NAME ↔ PHONE

- Binary relations can model lookup tables
- Binary relations are many-to-many relations

NAME	PHONE
Aki	4019
Philip	4107
Doug	4107
Doug	4136
Philip	0113
Frank	0110
Frank	6190
…	…

BINARY RELATION

- A binary relation R between A and B is an element of *Pow* (A x B), i.e., R ⊆ A x B
- Examples:
  - Parent : Person x Person
    - Parent == {(John, Jerry), (John, Sam)}
  - Square : Z x N
    - Square == {(1,1), (-1,1), (-2,4)}
  - ClassGrades : Person x {A, B, C, D, F}
    - ClassGrades == {(Todd,A), (Jane,B)}

TERNARY RELATION

- A ternary relation R between A, B and C is an element of *Pow* (A x B x C)
- Example:
  - FavoriteDrink : Person x Drink x Price
    - FavoriteDrink == {(John, Miller, \$2), (Ted, Heineken, \$4), (Steve, Miller, \$2)}
- N-ary relations with n>3 are defined analogously (n is the arity of the relation)

PAIRS

- *Pairs* are tuples with just two components.

  (aki, 4117)

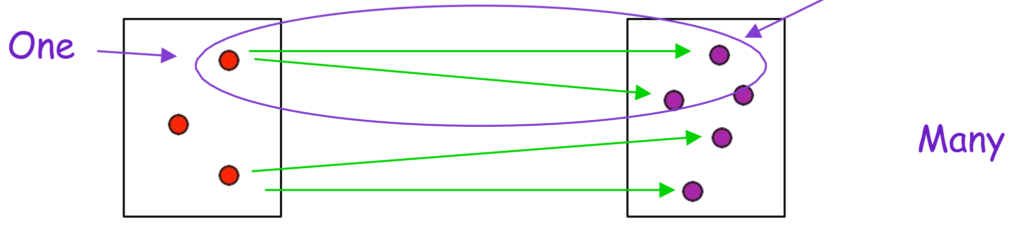
- The *maplet* arrow provides alternate syntax without parentheses.

$$\text{aki} \mapsto 4117$$

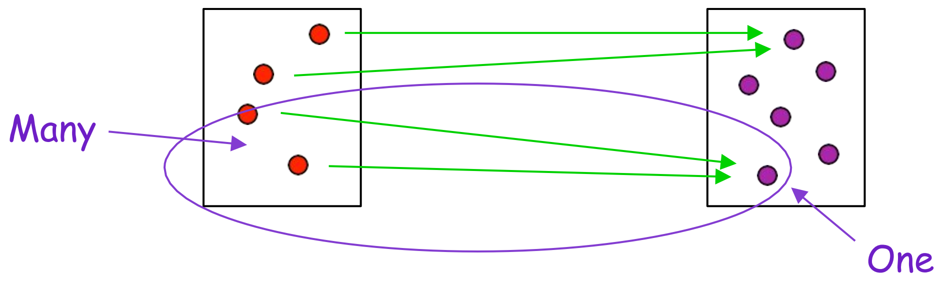
- The *projection* operators ***first*** and ***second*** extract the components of a pair.
- first(aki,4117) = aki
- second(aki, 4117) = 4117

COMMON RELATION STRUCTURES

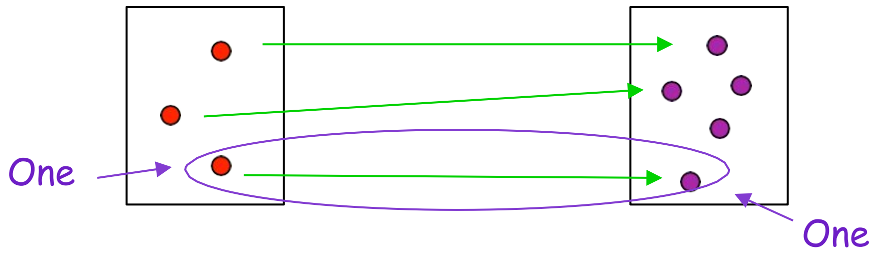
One-to-Many



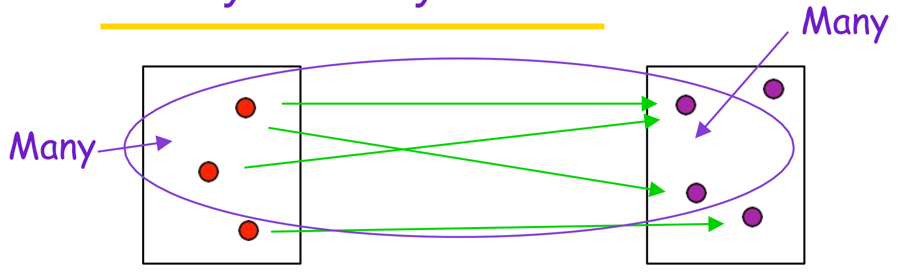
Many-to-One



One-to-One



Many-to-Many



RELATIONAL CALCULUS

•***Restriction operators*** can model database queries.
•***Domain restriction*** selects pairs based on their first component.

$$\{ doug, philip \} \lhd phone =$$

$$\{ philip \mapsto 4107, \\ doug \mapsto 4107, \\ doug \mapsto 4136, \\ philip \mapsto 0113 \}$$

Range restriction selects according to the second element.

$$phone \rhd (4000 \mathinner{..} 4999) = \{ \\ \ldots \\ aki \mapsto 4019, \\ philip \mapsto 4107, \\ doug \mapsto 4107, \\ doug \mapsto 4136, \\ \ldots \\ \}$$

OPERATOR SYMBOLS

Operator symbols are defined systematically.

$X \lhd R$	Domain restriction
$X ⩤ R$	Domain anti-restriction
$R \rhd Y$	Range restriction
$R ⩥ Y$	Range anti-restriction

Pictorial Domain & Range restriction operators can also be combined

$\{ doug, philip \} \lhd phone \rhd (4000 .. 4999) =$

$\{ philip \mapsto 4107,$
$doug \mapsto 4107,$
$doug \mapsto 4136)$

SEQUENCES

It is sometimes necessary to record the order in which objects are arranged:
For example, *data may be indexed by an ordered collection of keys*; *messages may be stored in order of arrival; tasks may be performed in order of importance*

Definition and Notation

A **sequence** is an ordered collection of objects. If there are no objects in the collection, the sequence is the ***empty sequence***, and is denoted as ‘< >’.
For example, the expression <*a, b, c*> denotes the sequence containing objects *a*, *b*, and *c*, in that order

Concatenation: This is a means of composing sequences in which two sequences are combined in such a way that the elements of one follow the elements of the other, and order is maintained. For example: <a,b,c> and <d,e> becomes <a,b,c>^<d,e> = <a,b,c,d,e>

Application: The ticket office in a railway station has a choice of two counters at which tickets may be purchased. There are two queues of people, one at each counter; these may be modelled as sequences:

queue_a = <john, joy, ashley>   and **queue_b** = <kim, max,thomas>

John and Kim are at the head of their respective queues, but.just as Kim is about to be served the ticket machine at Counter *b* breaks down, and the people waiting there join the end of other queue. Order is maintained, so the result is given by *queue_a ^queue_b*, the sequence is given by

<john,joy,Ashley,kim,max,thomas>

A queue of six people forms at Counter *a*

A sequence contains information about a **collection of elements** and the *order* in which they occur. It may be that not all of this information concerns us: we may restrict our attention to elements from a given set using the filter operator: if *s* is a sequence, then *s* | *A* is the largest subsequence of *s* containing ***only those objects that are elements of A***

<a, b, c,d, e, d, c, b, a>| {a,d} = <a,d,d,a>

The order and multiplicity of elements is preserved

APPLICATION EXAMPLE

Example:
In a train station, there is a destination board displaying a list of trains, arranged in order of departure:
This may be modelled as a sequence of pairs, each recording a time and a destination.

*trains==<(*10:15 , *London),(*10:38, *Edinburgh),(*10:40,*London),(*11:15, *Birmingham), (*11:20, *reading),(*11:40, *London>*

Suppose John is interested only in those trains that are going to London; he would be satisfied with the filtered sequence

trains |{ *t* : *Time* ***o*** *(t, london) }*

that is, *<(*10:15, *london),(*10:40, *london),(*11:40, *london)>*

It may be that we need to refer to the first element of a sequence, or to the part of the sequence that follows the first element; these are called the ***head*** and ***tail***, respectively.

head <a, b, c, d,e>=a
tail < a, b, c, d, e> = <b,c,d,e>

Note: the head of a sequence is an element, while the tail is another sequence. If *s* is any non-empty sequence, then

S= <head s>^ tail s

FUNCTIONS

- *Functions* are binary relations where each element in the domain appears just once. Each domain element is a *unique key*.
- A function cannot be a many-to-many or even one-to-many relation

$phonef : NAME \nrightarrow PHONE$

$phonef = \{$

...

$aki \mapsto 4019,$
$philip \mapsto 4107,$
$doug \mapsto 4107,$
$frank \mapsto 6190,$

...

$)$

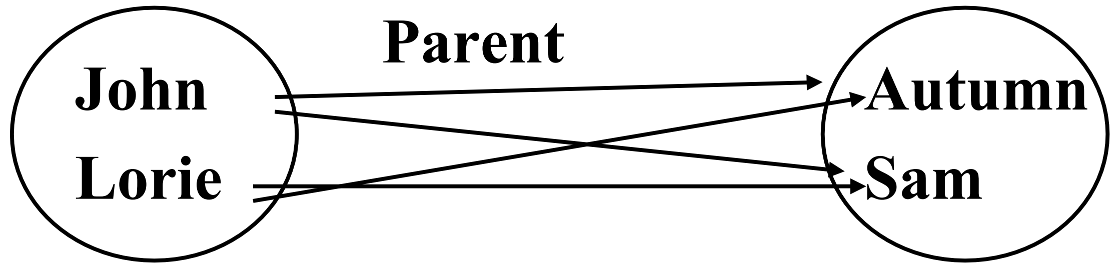
Function application is a special case of relational image. It associates a domain element with its unique range element.

$$phonef \llparenthesis \{ doug \} \rrparenthesis = 4107$$

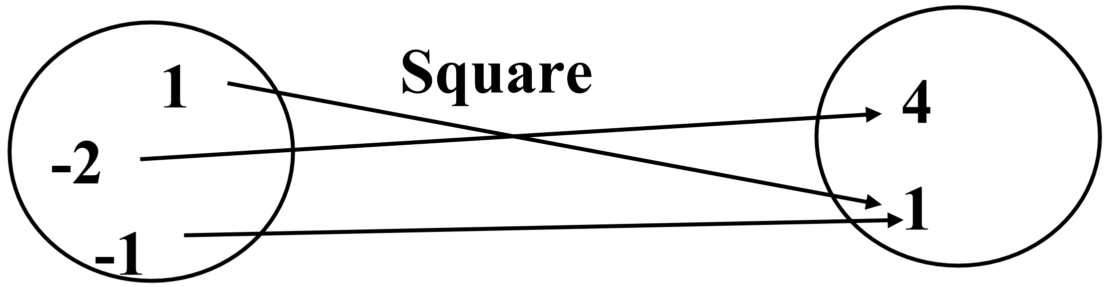
$$phonef\,(doug) = 4107$$

$$phonef\ doug = 4107$$

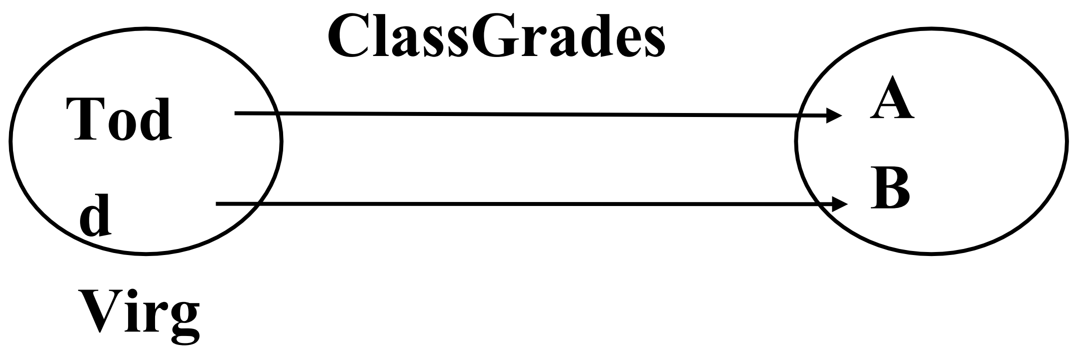
RELATIONS VS. FUNCTIONS



Many-to-man
y



Many-to-on
e



One-to-on
e

SPECIAL KINDS OF FUNCTIONS

- Consider a function f from S to T

- f is *total* if defined for all values of S
- f is *partial* if defined for some values of S

- Examples
  - Squares : Z -> N, Squares = {(-1,1), (2,4)}
  - Abs = {(x,y) : Z x N |
    (x < 0 and y = -x) or (x ≥ 0 and y = x)}

SPECIAL FUNCTIONS ON SEQUENCES

Concatenation

<a,b>^<b,a,c> = <a,b,b,a,c>

Head

$$\begin{array}{|l}
\mathit{Head} : \mathit{seq}_1 A \longrightarrow A \\
\hline
\forall\ s : \mathit{seq}_1\ A \bullet\ \mathit{head}(s) = s(1)
\end{array}$$

head<c, b, b> = c

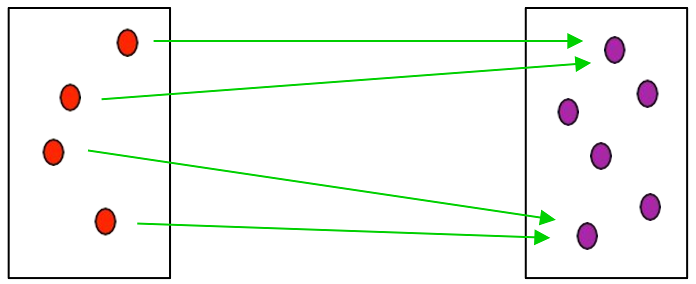
Tail

$$\begin{array}{|l}
\mathit{Tail} : \mathit{seq}_1\ A \longrightarrow A \\
\hline
\forall\ s : \mathit{seq}_1\ A \bullet <\mathit{head}(s)\hat{}\ \mathit{tail}(s) =
\end{array}$$

tail<c,b,b> = <b,b>

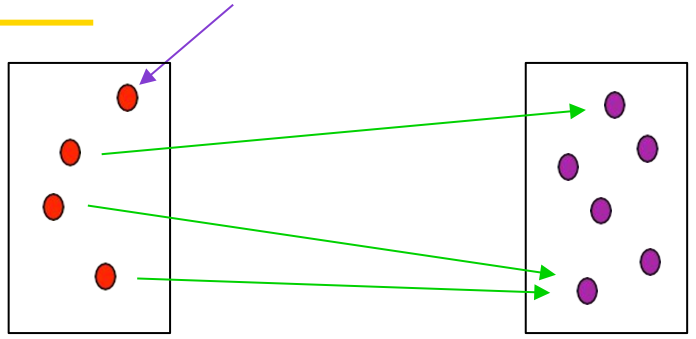
FUNCTION STRUCTURES

Total Function



Partial Function

Undefined for this input



Note: the empty relation is a partial function

TYPES OF FUNCTIONS

A function f: S ⟶ T is

Surjective Function

- **Surjective** (onto) if every element of the domain is mapped to some element of the range. some domain elements may be mapped to more than one range elements. (Total Injections)

Injective Function

- *one-to-one* (*injective*) if no image element is associated with multiple domain elements (Partial injections)

Bijective Function

- **Bijective** (one-to-one and onto) iff it is both injective and surjective. (Equivalently, every element of the domain is mapped to exactly one element of the range.) A bijective function is a bijection (one-to-one correspondence), and is reversible.

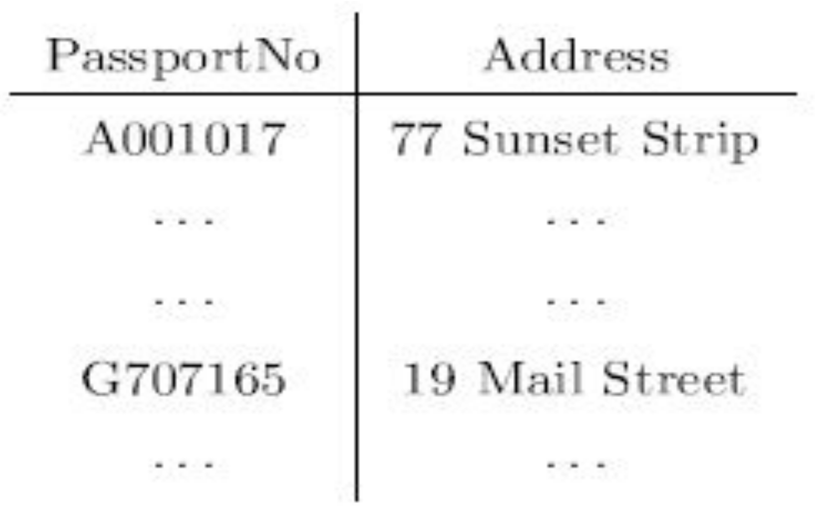
SPECIFYING FUNCTIONS

1.Using a Look-up- Table

If a function f : $A \nrightarrow B$ is finite (and not too large) we can specify the function explicitly by listing all the pairs (a b) in the subset A x B where f(a) = b

e.g

address : PassportNo $\longrightarrow$ Address



PassportNo	Address
A001017	77 Sunset Strip
...	...
...	...
G707165	19 Mail Street
...	...

2.Giving an Algorithm

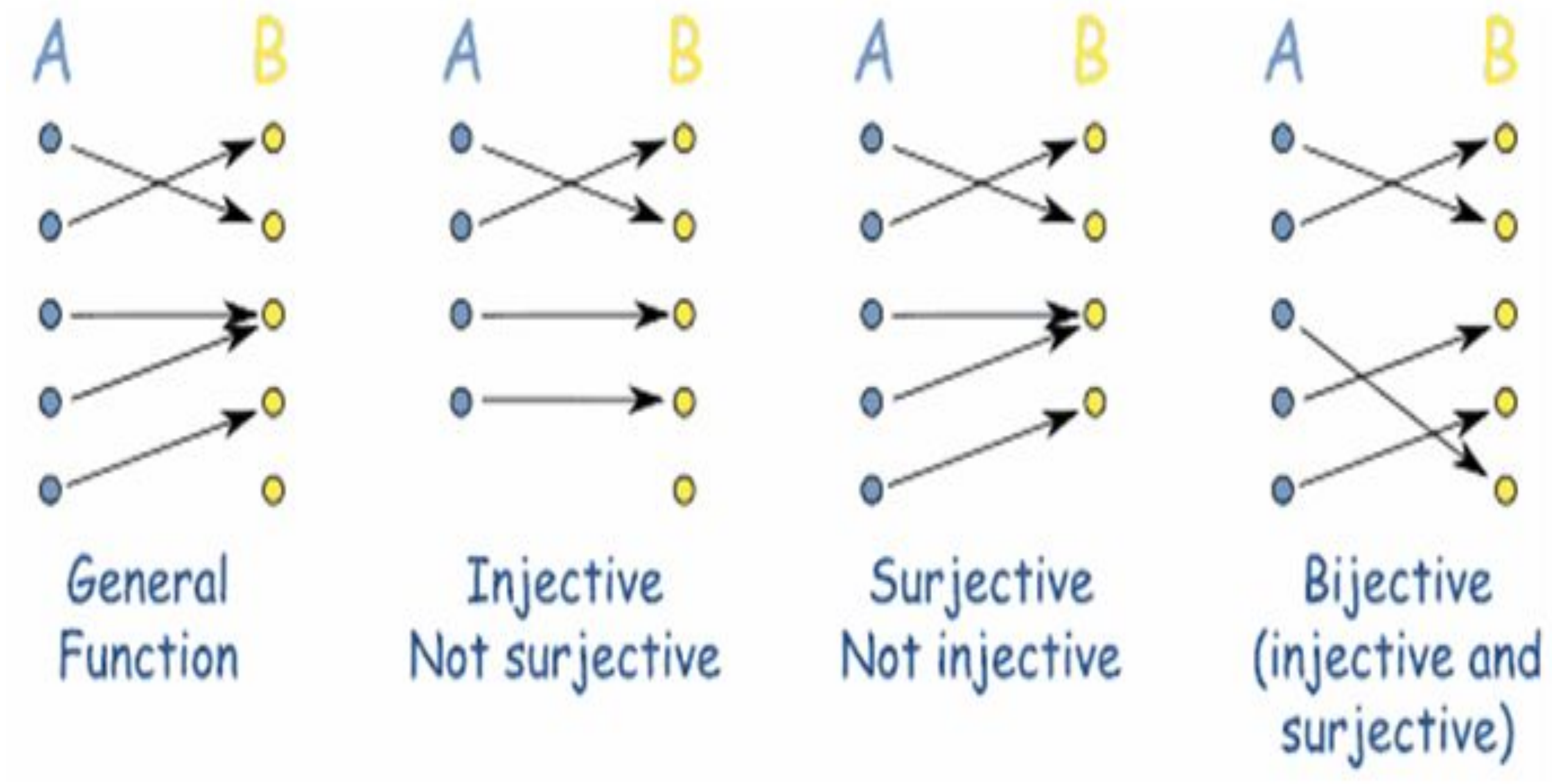
A function f : A $\nrightarrow B$ is specified by an algorithm(i.e a program) such that given any element a in the domain of f, the element f(a) can be computed using the algorithm

```
input $n : \mathbb{N}$
var $x, y$: integer;
begin
$x := n;$ $y := 0;$
while $x \neq 0$ do
  begin
  $x := x - 1;$ $y := y + 2$
  end;
write($y$)
end.
```

This algorithm computes the function. But how can we prove this?

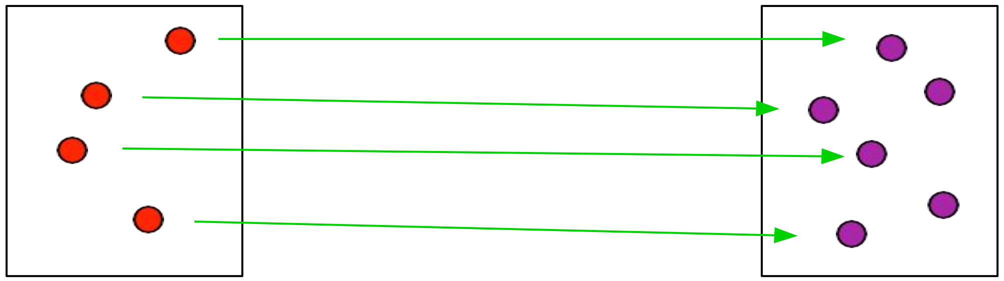
$$\begin{array}{|l} double : \mathbb{N} \rightarrow \mathbb{N} \\ \hline \forall\, n : \mathbb{N} \bullet double(n) = 2n \end{array}$$

ILLUSTRATIONS

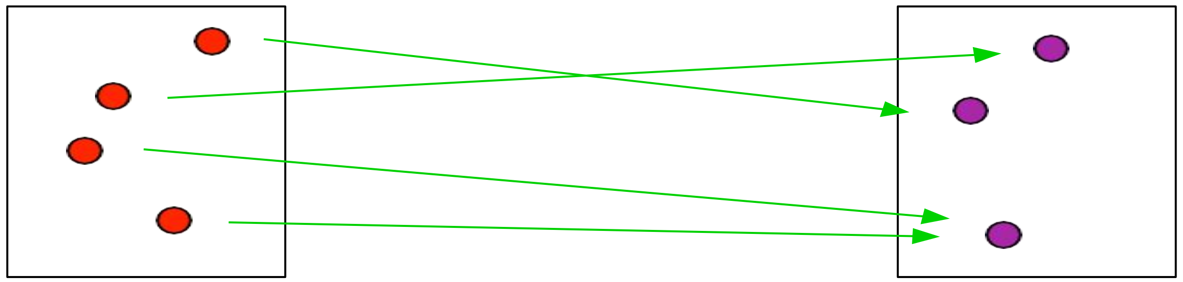


TYPES OF FUNCTION

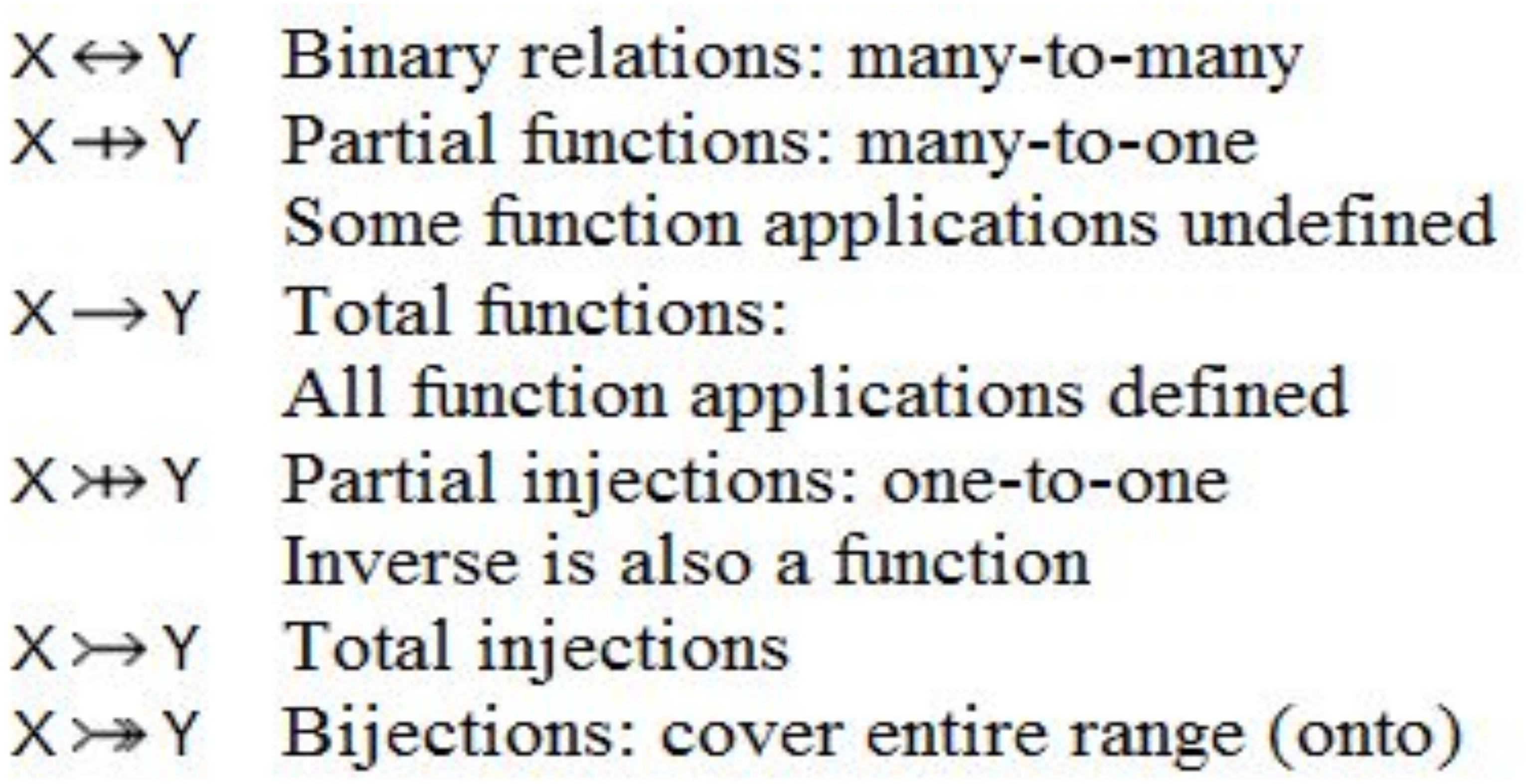
Injective(one-to-one Function



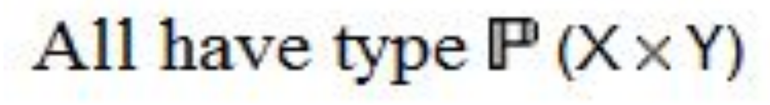
Surjective(onto)Function



BINARY RELATIONS AND FUNCTIONS



$X \leftrightarrow Y$	Binary relations: many-to-many
$X \nrightarrow Y$	Partial functions: many-to-one Some function applications undefined
$X \rightarrow Y$	Total functions: All function applications defined
$X \rightarrowtail\!\!\!\!\!\!\nrightarrow Y$	Partial injections: one-to-one Inverse is also a function
$X \rightarrowtail Y$	Total injections
$X \twoheadrightarrow\!\!\!\!\!\!\rightarrowtail Y$	Bijections: cover entire range (onto)



All have type $\mathbb{P}(X \times Y)$

CLASS WORK

- What kind of function/relation is Abs?
  - Abs = {(x,y) : Z x N | (x < 0 and y = -x) or
    (x ≥ 0 and y = x)}

- How about Squares?
  - Squares : Z x N, Squares = {(-1,1),(2,4)}

EXERCISES