


Logic

# Discrete Mathematics

## Topic 04 — Relations and Functions

### Lecture 03 — Function Concepts and Definitions

Dr Kieran Murphy 

Department of Computing and Mathematics,  
SETU (Waterford).  
(kieran.murphy@setu.ie)

Autumn Semester, 2022

#### Outline

- Definition of a Function
- Function Properties

Enumeration

# Outline

---

1. Definition of a Function	2
1.1. Definition Based on Relations	3
1.2. Function Notation	11
1.3. An Aside: Interval Notation	13
2. Function Properties	17
2.1. Surjective (Onto)	19
2.2. Injective (One-to-One)	21
2.3. Bijective (Injective and Surjective)	23

# Relation

Recall our definition of a relation,  $R$  from set  $A$  to set  $B$

## Definition 1 (Relation)

A **relation**,  $R$ , from set  $A$  to set  $B$  is any subset of the Cartesian product  $A \times B$

$$R = \{(a, b) \mid a \in A, b \in B\} \quad \subseteq A \times B \quad (1)$$

# Relation

Recall our definition of a relation,  $R$  from set  $A$  to set  $B$

## Definition 1 (Relation)

A **relation**,  $R$ , from set  $A$  to set  $B$  is any subset of the Cartesian product  $A \times B$

$$R = \{(a, b) \mid a \in A, b \in B\} \quad \subseteq A \times B \quad (1)$$

Graphically, this looks like two sets (the source and the target) with arrows leaving elements in the source towards each elements in the target.

# Relation

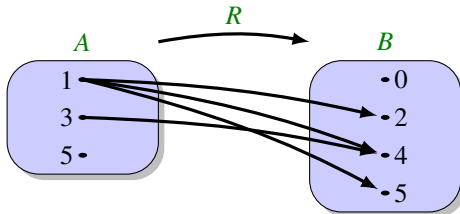
Recall our definition of a relation,  $R$  from set  $A$  to set  $B$

## Definition 1 (Relation)

A **relation**,  $R$ , from set  $A$  to set  $B$  is any subset of the Cartesian product  $A \times B$

$$R = \{(a, b) \mid a \in A, b \in B\} \subseteq A \times B \quad (1)$$

Graphically, this looks like two sets (the source and the target) with arrows leaving elements in the source towards each elements in the target.



# Relation

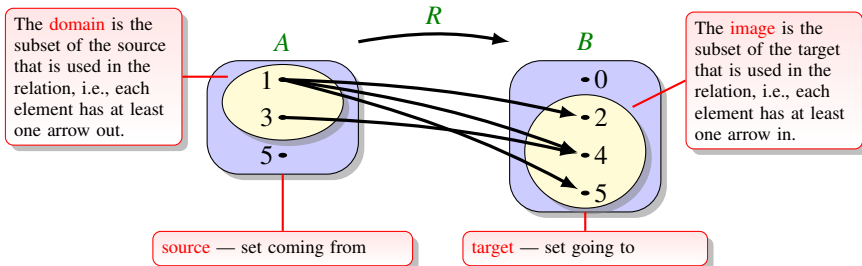
Recall our definition of a relation,  $R$  from set  $A$  to set  $B$

## Definition 1 (Relation)

A **relation**,  $R$ , from set  $A$  to set  $B$  is any subset of the Cartesian product  $A \times B$

$$R = \{(a, b) \mid a \in A, b \in B\} \subseteq A \times B \quad (1)$$

Graphically, this looks like two sets (the source and the target) with arrows leaving elements in the source towards each elements in the target.



# For god's sake, think of the Programmer

## I

We want to restrict our relation definition so that it will make life easier for us as programmers — for example, consider implementing\* “square of” relation over  $\mathbb{R}$  in either java or python.

$$R = \{(a, b) \mid a \in \mathbb{R}, b \in \mathbb{R} \wedge a = b^2\}$$

We have a number of issues (programming wise):

- We can't represent the continuous, infinite set of real numbers,  $\mathbb{R}$  on a discrete, finite device such as our computers.
  - Standard “solution” is to approximate  $\mathbb{R}$  by the `double` data type.
  - Read [What Every Programmer Should Know about Floating-Point Arithmetic](#)

---

\*Typical approach is to implement a `method` in Java, or a `function` in Python which given  $a$  returns  $b$ .

# For god's sake, think of the Programmer

## I

We want to restrict our relation definition so that it will make life easier for us as programmers — for example, consider implementing\* “square of” relation over  $\mathbb{R}$  in either java or python.

$$R = \{(a, b) \mid a \in \mathbb{R}, b \in \mathbb{R} \wedge a = b^2\}$$

We have a number of issues (programming wise):

- We can't represent the continuous, infinite set of real numbers,  $\mathbb{R}$  on a discrete, finite device such as our computers.
  - Standard “solution” is to approximate  $\mathbb{R}$  by the **double** data type.
  - Read [What Every Programmer Should Know about Floating-Point Arithmetic](#)

---

\*Typical approach is to implement a **method** in Java, or a **function** in Python which given  $a$  returns  $b$ .



# For god's sake, think of the Programmer

We want to restrict our relation definition so that it will make life easier for us as programmers — for example, consider implementing\* “square of” relation over  $\mathbb{R}$  in either java or python.

$$R = \{(a, b) \mid a \in \mathbb{R}, b \in \mathbb{R} \wedge a = b^2\}$$

We have a number of issues (programming wise):

- We can't represent the continuous, infinite set of real numbers,  $\mathbb{R}$  on a discrete, finite device such as our computers.
  - Standard “solution” is to approximate  $\mathbb{R}$  by the **double** data type.
  - Read [What Every Programmer Should Know about Floating-Point Arithmetic](#)

```
MyFunction.java
3  double f(double a) {
4      double result = 0.0;
5
6      // do calculation
7
8      return result;
9  }
```

---

\*Typical approach is to implement a **method** in Java, or a **function** in Python which given  $a$  returns  $b$ .

# For god's sake, think of the Programmer

We want to restrict our relation definition so that it will make life easier for us as programmers — for example, consider implementing\* “square of” relation over  $\mathbb{R}$  in either java or python.

$$R = \{(a, b) \mid a \in \mathbb{R}, b \in \mathbb{R} \wedge a = b^2\}$$

We have a number of issues (programming wise):

- We can't represent the continuous, infinite set of real numbers,  $\mathbb{R}$  on a discrete, finite device such as our computers.
  - Standard “solution” is to approximate  $\mathbb{R}$  by the **double** data type.
  - Read [What Every Programmer Should Know about Floating-Point Arithmetic](#)

MyFunction.py

```

1 def f(a):
2     result = 0.0
3
4     # do calculation
5
6     return result

```

MyFunction.java

```

3 double f(double a) {
4     double result = 0.0;
5
6     // do calculation
7
8     return result;
9 }

```

---

\*Typical approach is to implement a **method** in Java, or a **function** in Python which given  $a$  returns  $b$ .

# For god's sake, think of the Programmer

...our issues continued ...

- For some inputs, my relation ( $a = b^2$  on  $\mathbb{R}$ ) generates multiple outputs.

$$(16, 4) \in R \quad \wedge \quad (16, -4) \in R, \quad \dots$$

- For some inputs, my relation generates no outputs.

$$(-1, b) \notin R \quad \forall b \in \mathbb{R}$$

As a result:

- My Java implementation of **double** input **double** output is no good.
- In Python, life is nicer because we are free to return **None** for no result, or multiple **doubles** if needed — but still need special code.

Thinking of the poor programmer, getting minimum wage, etc., we ...

Restrict relations so that:

- All inputs generate at least one output, i.e., domain = source.
- All inputs generate at most one output, i.e., at most one arrow leaving each element

# For god's sake, think of the Programmer

## II

...our issues continued ...

- For some inputs, my relation ( $a = b^2$  on  $\mathbb{R}$ ) generates multiple outputs.

$$(16, 4) \in R \quad \wedge \quad (16, -4) \in R, \quad \dots$$

- For some inputs, my relation generates no outputs.

$$(-1, b) \notin R \quad \forall b \in \mathbb{R}$$

As a result:

- My Java implementation of **double** input **double** output is no good.
- In Python, life is nicer because we are free to return **None** for no result, or multiple **doubles** if needed — but still need special code.

Thinking of the poor programmer, getting minimum wage, etc., we ...

Restrict relations so that:

- All inputs generate at least one output, i.e., domain = source.
- All inputs generate at most one output, i.e., at most one arrow leaving each element

# For god's sake, think of the Programmer

## II

... our issues continued ...

- For some inputs, my relation ( $a = b^2$  on  $\mathbb{R}$ ) generates multiple outputs.

$$(16, 4) \in R \quad \wedge \quad (16, -4) \in R, \quad \dots$$

- For some inputs, my relation generates no outputs.

$$(-1, b) \notin R \quad \forall b \in \mathbb{R}$$

As a result:

- My Java implementation of **double** input **double** output is no good.
- In Python, life is nicer because we are free to return **None** for no result, or multiple **doubles** if needed — but still need special code.

Thinking of the poor programmer, getting minimum wage, etc., we ...

Restrict relations so that:

- All inputs generate at least one output, i.e., domain = source.
- All inputs generate at most one output, i.e., at most one arrow leaving each element

# For god's sake, think of the Programmer

...our issues continued ...

- For some inputs, my relation ( $a = b^2$  on  $\mathbb{R}$ ) generates multiple outputs.

$$(16, 4) \in R \quad \wedge \quad (16, -4) \in R, \quad \dots$$

- For some inputs, my relation generates no outputs.

$$(-1, b) \notin R \quad \forall b \in \mathbb{R}$$

As a result:

- My Java implementation of **double** input **double** output is no good.
- In Python, life is nicer because we are free to return **None** for no result, or multiple **doubles** if needed — but still need special code.

Thinking of the poor programmer, getting minimum wage, etc., we ...

All inputs generate exactly one output.

Restrict relations so that:

- All inputs generate at least one output, i.e., domain = source.
- All inputs generate at most one output, i.e., at most one arrow leaving each element

# Function Definition Based on a Relation

## Definition 2 (Function)

Let  $R$  be a relation from set  $A$  to set  $B$  where

- Each element of  $A$  is in the domain of  $R$ , i.e.,
  - At least one arrow leaving each element in  $A$ .
  - $\exists b \in B$  such that  $(a, b) \in R \quad \forall a \in A$
  - Source of  $R$  is equal to  $\text{Dom}(R)$
- At most one output for each input
  - At most one arrow entering each element in  $B$ .
  - If  $(a, b) \in R$  and  $(a, c) \in R$  then  $b = c$

$\Rightarrow$  We say  $R$  is  
a **function**  
from set  $A$  to  
set  $B$ .

# Function Definition Based on a Relation

## Definition 2 (Function)

Let  $R$  be a relation from set  $A$  to set  $B$  where

- Each element of  $A$  is in the domain of  $R$ , i.e.,
  - At least one arrow leaving each element in  $A$ .
  - $\exists b \in B$  such that  $(a, b) \in R \quad \forall a \in A$
  - Source of  $R$  is equal to  $\text{Dom}(R)$
- At most one output for each input
  - At most one arrow entering each element in  $B$ .
  - If  $(a, b) \in R$  and  $(a, c) \in R$  then  $b = c$

$\Rightarrow$  We say  $R$  is  
a **function**  
from set  $A$  to  
set  $B$ .

Graphically, we have ...



# Function Definition Based on a Relation

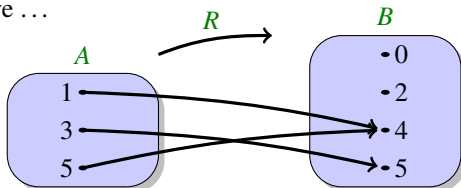
## Definition 2 (Function)

Let  $R$  be a relation from set  $A$  to set  $B$  where

- Each element of  $A$  is in the domain of  $R$ , i.e.,
  - At least one arrow leaving each element in  $A$ .
  - $\exists b \in B$  such that  $(a, b) \in R \quad \forall a \in A$
  - Source of  $R$  is equal to  $\text{Dom}(R)$
- At most one output for each input
  - At most one arrow entering each element in  $B$ .
  - If  $(a, b) \in R$  and  $(a, c) \in R$  then  $b = c$

$\Rightarrow$  We say  $R$  is a **function** from set  $A$  to set  $B$ .

Graphically, we have ...



# Function Definition Based on a Relation

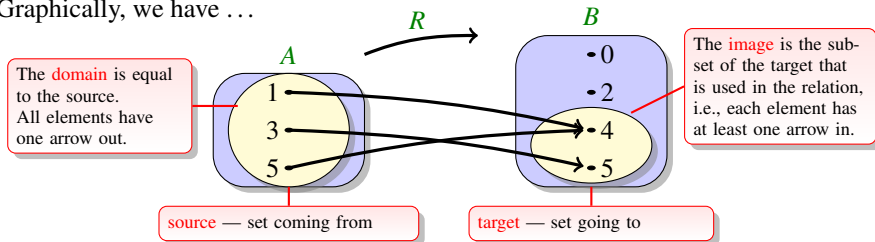
## Definition 2 (Function)

Let  $R$  be a relation from set  $A$  to set  $B$  where

- Each element of  $A$  is in the domain of  $R$ , i.e.,
  - At least one arrow leaving each element in  $A$ .
  - $\exists b \in B$  such that  $(a, b) \in R \quad \forall a \in A$
  - Source of  $R$  is equal to  $\text{Dom}(R)$
- At most one output for each input
  - At most one arrow entering each element in  $B$ .
  - If  $(a, b) \in R$  and  $(a, c) \in R$  then  $b = c$

We say  $R$  is  
a **function**  
from set  $A$  to  
set  $B$ .

Graphically, we have ...



## Example 3

### Example 3 (Specifying a function as a set of ordered pairs)

Let  $S = \{1, 2, 3\}$  and  $T = \{a, b, c\}$ . Set

$$f = \{(1, a), (2, a), (3, b)\}$$

Then  $f$  is a function since

- $f$  is a relation from source set  $S = \{1, 2, 3\}$  to target set  $T = \{a, b, c\}$ .
- The domain of  $f$  is equal to the source of  $f$ .
- Elements in the domain are related to exactly one element in the target.

## Example 3

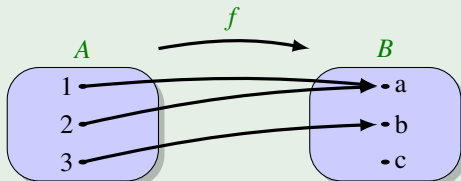
### Example 3 (Specifying a function as a set of ordered pairs)

Let  $S = \{1, 2, 3\}$  and  $T = \{a, b, c\}$ . Set

$$f = \{(1, a), (2, a), (3, b)\}$$

Then  $f$  is a function since

- $f$  is a relation from source set  $S = \{1, 2, 3\}$  to target set  $T = \{a, b, c\}$ .
- The domain of  $f$  is equal to the source of  $f$ .
- Elements in the domain are related to exactly one element in the target.



## Example 3

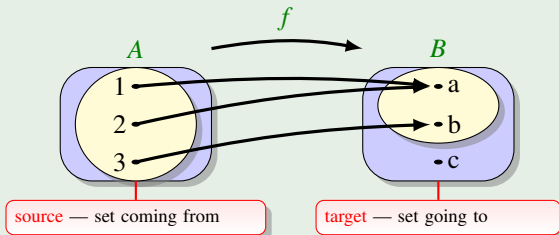
### Example 3 (Specifying a function as a set of ordered pairs)

Let  $S = \{1, 2, 3\}$  and  $T = \{a, b, c\}$ . Set

$$f = \{(1, a), (2, a), (3, b)\}$$

Then  $f$  is a function since

- $f$  is a relation from source set  $S = \{1, 2, 3\}$  to target set  $T = \{a, b, c\}$ .
- The domain of  $f$  is equal to the source of  $f$ .
- Elements in the domain are related to exactly one element in the target.



## Example 3

### Example 3 (Specifying a function as a set of ordered pairs)

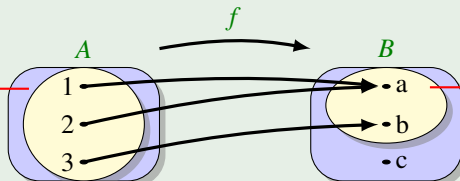
Let  $S = \{1, 2, 3\}$  and  $T = \{a, b, c\}$ . Set

$$f = \{(1, a), (2, a), (3, b)\}$$

Then  $f$  is a function since

- $f$  is a relation from source set  $S = \{1, 2, 3\}$  to target set  $T = \{a, b, c\}$ .
- The domain of  $f$  is equal to the source of  $f$ .
- Elements in the domain are related to exactly one element in the target.

The **domain** is equal to the source.  
All elements have one arrow out.



The **image** is the subset of the target that is used in the relation, i.e., each element has at least one arrow in.

## Example 4

Instead of listing pairs, as in the previous example, we can just give a lookup table.

### Example 4 (Specifying a function using a lookup table)

Let  $f$  be the function defined by

input	output
1	a
2	a
3	b
	c

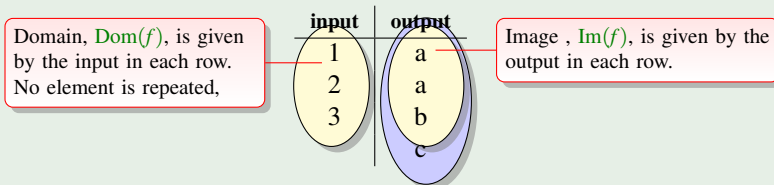
- Lookup tables are frequently used in computing in situations where memory is cheaper than computing cycles.
  - Number theory libraries would store small primes up to, say 1000, and compute others as needed.
  - ASCII table and now unicode.

## Example 4

Instead of listing pairs, as in the previous example, we can just give a lookup table.

### Example 4 (Specifying a function using a lookup table)

Let  $f$  be the function defined by



- Lookup tables are frequently used in computing in situations where memory is cheaper than computing cycles.
  - Number theory libraries would store small primes up to, say 1000, and compute others as needed.
  - ASCII table and now unicode.

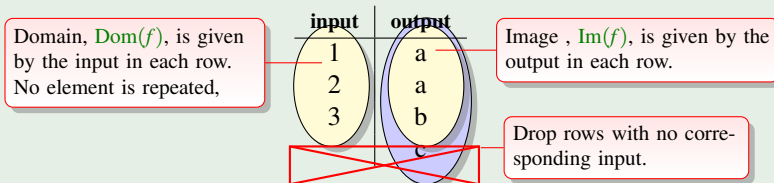


## Example 4

Instead of listing pairs, as in the previous example, we can just give a lookup table.

### Example 4 (Specifying a function using a lookup table)

Let  $f$  be the function defined by



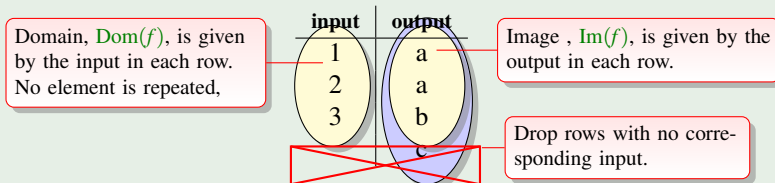
- Lookup tables are frequently used in computing in situations where memory is cheaper than computing cycles.
  - Number theory libraries would small primes up to, say 1000, and compute others as needed.
  - ASCII table and now unicode.

## Example 4

Instead of listing pairs, as in the previous example, we can just give a lookup table.

### Example 4 (Specifying a function using a lookup table)

Let  $f$  be the function defined by



- Lookup tables are frequently used in computing in situations where memory is cheaper than computing cycles.
  - Number theory libraries would small primes up to, say 1000, and compute others as needed.
  - ASCII table and now unicode.

# ASCII Table — A Relation between Integers and Characters

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	\$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

# ASCII Table — A Relation between Integers and Characters

chr				chr					chr					chr				
Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0 0	00	0	<b>ord</b> NULL	32 20	014	0	&#032;	Space	64 40	040	0	&#064;	@	96 60	144	0	&#096;	`
1 1	001	1	Start of Header	33 21	015	1	&#033;	!	65 41	041	1	&#065;	A	97 61	145	1	&#097;	a
2 2	002	2	Start of Text	34 22	042	2	&#034;	"	66 42	102	2	&#066;	B	98 62	142	2	&#098;	b
3 3	003	3	End of Text	35 23	043	3	&#035;	#	67 43	103	3	&#067;	C	99 63	143	3	&#099;	c
4 4	004	4	End of Transmission	36 24	044	4	&#036;	\$	68 44	104	4	&#068;	D	100 64	144	4	&#100;	d
5 5	005	5	Enquiry	37 25	045	5	&#037;	%	69 45	105	5	&#069;	E	101 65	145	5	&#101;	e
6 6	006	6	Acknowledgment	38 26	046	6	&#038;	&	70 46	106	6	&#070;	F	102 66	146	6	&#102;	f
7 7	007	7	Bell	39 27	047	7	&#039;	'	71 47	107	7	&#071;	G	103 67	147	7	&#103;	g
8 8	010	10	Backspace	40 28	050	8	&#040;	(	72 48	110	8	&#072;	H	104 68	150	8	&#104;	h
9 9	011	11	Horizontal Tab	41 29	051	9	&#041;	)	73 49	111	9	&#073;	I	105 69	151	9	&#105;	i
10 A	012	12	Line feed	42 2A	052	10	&#042;	*	74 4A	112	10	&#074;	J	106 6A	152	10	&#106;	j
11 B	013	13	Vertical Tab	43 2B	053	11	&#043;	+	75 4B	113	11	&#075;	K	107 6B	153	11	&#107;	k
12 C	014	14	Form feed	44 2C	054	12	&#044;	,	76 4C	114	12	&#076;	L	108 6C	154	12	&#108;	l
13 D	015	15	Carriage return	45 2D	055	13	&#045;	-	77 4D	115	13	&#077;	M	109 6D	155	13	&#109;	m
14 E	016	16	Shift Out	46 2E	056	14	&#046;	.	78 4E	116	14	&#078;	N	110 6E	156	14	&#110;	n
15 F	017	17	Shift In	47 2F	057	15	&#047;	/	79 4F	117	15	&#079;	O	111 6F	157	15	&#111;	o
16 10	020	20	Data Link Escape	48 30	060	16	&#048;	0	80 50	120	8	&#080;	P	112 70	160	8	&#112;	p
17 11	021	21	Device Control 1	49 31	061	17	&#049;	1	81 51	121	9	&#081;	Q	113 71	161	9	&#113;	q
18 12	022	22	Device Control 2	50 32	062	18	&#050;	2	82 52	122	10	&#082;	R	114 72	162	10	&#114;	r
19 13	023	23	Device Control 3	51 33	063	19	&#051;	3	83 53	123	11	&#083;	S	115 73	163	11	&#115;	s
20 14	024	24	Device Control 4	52 34	064	20	&#052;	4	84 54	124	12	&#084;	T	116 74	164	12	&#116;	t
21 15	025	25	Negative Ack.	53 35	065	21	&#053;	5	85 55	125	13	&#085;	U	117 75	165	13	&#117;	u
22 16	026	26	Synchronous idle	54 36	066	22	&#054;	6	86 56	126	14	&#086;	V	118 76	166	14	&#118;	v
23 17	027	27	End of Trans. Block	55 37	067	23	&#055;	7	87 57	127	15	&#087;	W	119 77	167	15	&#119;	w
24 18	030	30	Cancel	56 38	070	24	&#056;	8	88 58	130	16	&#088;	X	120 78	170	16	&#120;	x
25 19	031	31	End of Medium	57 39	071	25	&#057;	9	89 59	131	17	&#089;	Y	121 79	171	17	&#121;	y
26 1A	032	32	Substitute	58 3A	072	26	&#058;	:	90 5A	132	18	&#090;	Z	122 7A	172	18	&#122;	z
27 1B	033	33	Escape	59 3B	073	27	&#059;	;	91 5B	133	19	&#091;	[	123 7B	173	19	&#123;	{
28 1C	034	34	File Separator	60 3C	074	28	&#060;	<	92 5C	134	20	&#092;	\	124 7C	174	20	&#124;	
29 1D	035	35	Group Separator	61 3D	075	29	&#061;	=	93 5D	135	21	&#093;	]	125 7D	175	21	&#125;	}
30 1E	036	36	Record Separator	62 3E	076	30	&#062;	>	94 5E	136	22	&#094;	^	126 7E	176	22	&#126;	~
31 1F	037	37	Unit Separator	63 3F	077	31	&#063;	?	95 5F	137	23	&#095;	_	127 7F	177	23	&#127;	Del

# ASCII Table — A Relation between Integers and Characters

chr					chr					chr					chr				
Dec	Hex	Oct	Chr		Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0 0	00	0	NULL		32 20	015	032	Space		64 40	040	064	@		96 60	090	144	&#x0060;	`
1 1	01	1	Start of Header		33 21	019	033	!		65 41	041	065	A		97 61	097	142	&#x0097;	a
2 2	02	2	Start of Text		34 22	042	&#x034;	!"		66 42	042	066	B		98 62	142	&#x0098;	b	
3 3	03	3	End of Text		35 23	043	&#x035;	#		67 43	103	&#x067;	C		99 63	143	&#x0099;	c	
4 4	04	4	End of Transmission		36 24	044	&#x036;	\$		68 44	104	&#x068;	D		100 64	144	&#x0100;	d	
5 5	05	5	Enquiry		37 25	045	&#x037;	%		69 45	105	&#x069;	E		101 65	145	&#x0101;	e	
6 6	06	6	Acknowledgment		38 26	046	&#x038;	&		70 46	106	&#x070;	F		102 66	146	&#x0102;	f	
7 7	07	7	Bell		39 27	047	&#x039;	'											
8 8	08	10	Backspace		40 28	050	&#x040;	(	1										
9 9	09	11	Horizontal Tab		41 29	051	&#x041;	)											
10 A	0A	12	Line feed		42 2A	052	&#x042;	*	2										
11 B	0B	13	Vertical Tab		43 2B	053	&#x043;	+											
12 C	0C	14	Form feed		44 2C	054	&#x044;	,	3										
13 D	0D	15	Carriage return		45 2D	055	&#x045;	-											
14 E	0E	16	Shift Out		46 2E	056	&#x046;	.		77 4B	115	&#x077;	M		103 6B	135	&#x006B;	m	
15 F	0F	17	Shift In		47 2F	057	&#x047;	/		78 4E	116	&#x078;	N		110 6E	156	&#x0110;	n	
16 10	020		Data Link Escape		48 30	060	&#x048;	0		79 4F	117	&#x079;	O		111 6F	157	&#x0111;	o	
17 11	021		Device Control 1		49 31	061	&#x049;	1		80 50	120	&#x080;	P		112 70	160	&#x0112;	p	
18 12	022		Device Control 2		50 32	062	&#x050;	2		81 51	121	&#x081;	Q		113 71	161	&#x0113;	q	
19 13	023		Device Control 3		51 33	063	&#x051;	3		82 52	122	&#x082;	R		114 72	162	&#x0114;	r	
20 14	024		Device Control 4		52 34	064	&#x052;	4		83 53	123	&#x083;	S		115 73	163	&#x0115;	s	
										84 54	124	&#x084;	T		116 74	164	&#x0116;	t	
															117 75	165	&#x0117;	u	
															118 76	166	&#x0118;	v	
															119 77	167	&#x0119;	w	
															120 78	170	&#x0120;	x	
															121 79	171	&#x0121;	y	
															122 7A	172	&#x0122;	z	
															123 7B	173	&#x0123;	{	
															124 7C	174	&#x0124;		
															125 7D	175	&#x0125;	}	
															126 7E	176	&#x0126;	~	
															127 7F	177	&#x0127;	Del	

Character 'A' map to 65  
 Character 'l' map to 49  
 Integer 43 maps to character +

```
# convert from character to ASCII
```

```
print ("Character 'A' map to ", ord('A'))
```

```
print ("Character 'l' map to ", ord('l'))
```

```
# convert from ACSII to character
```

```
print ("Integer 43 maps to character ", chr(43))
```

ascii.py

asciichars.com

## Example 5

### Example 5 (Specifying a function using set-builder notation)

The relation

$$L = \{(x, 3x) \mid x \in \mathbb{R}\}$$

is a function from  $\mathbb{R}$  to  $\mathbb{R}$ .

- Alternative notation is typically used when dealing with functions

$$L : \underbrace{\mathbb{R}}_{\substack{\text{source} \\ (= \text{domain})}} \rightarrow \underbrace{\mathbb{R}}_{\text{target}} : \underbrace{x \mapsto 3x}_{\text{rule}}$$

- Since in this example the source is equal to the target we say “ $L$  is a function on  $\mathbb{R}$ ”. (as we did for relations)
- Similarly, the concepts
  - Into vs Onto
  - Injective (one-to-one)

also apply to functions. These properties are important when reversing functions<sup>†</sup>, so we will cover them again using function notation.

<sup>†</sup>decrypting a message, unzipping an archive, etc.

# Function Notation

When defining functions we should be careful and explicitly state the source, the target and the rule. But we are informal (sloppy) and leave detail out assuming the reader will know what is implied. As a result there is large variation in notation. For example, all of the following are intended to define the same function

- *Formal definition using set build notation*

$$f = \{(a, b) | a \in \mathbb{R}, b \in \mathbb{R} \wedge 3a = b\}$$

- *Formal definition using function notation*

$$f : \mathbb{R} \rightarrow \mathbb{R} : x \mapsto 3x$$

or

$$f : \mathbb{R} \rightarrow \mathbb{R} : f(x) = 3x$$

- *Informal definition using function notation*

$$f : x \mapsto 3x$$

or

$$f(x) = 3x$$

or (this last version is horrible but we all do it)

$$f = 3x$$

Based on the context we usually assume  $\mathbb{R} \mapsto \mathbb{R}$ ,  $\mathbb{Z} \mapsto \mathbb{Z}$ , or  $\mathbb{N} \mapsto \mathbb{N}$ . But need to verify that function is **well-defined**.

# Constructing a Well-defined Function

Consider each of the following functions

$$a(x) = x^2 \quad b(x) = \sqrt{x} \quad c(x) = \frac{1}{x-2} \quad d(x) = \log(x)$$

In all four cases, we might start by assuming that the functions are from set  $\mathbb{R}$  to set  $\mathbb{R}$  but, while this works for the first function, we have problems with the others.

Hence

If given just the rule, one must determine what inputs are allowable when specifying the source (domain).

For our four functions above we have

$$a : \mathbb{R} \rightarrow \mathbb{R} : x \mapsto x^2 \quad (\text{no issue})$$

$$b : [0, \infty) \rightarrow \mathbb{R} : x \mapsto \sqrt{x} \quad (\text{cannot get } \sqrt{\phantom{x}} \text{ of negative values})$$

$$c : \mathbb{R} \setminus \{2\} \rightarrow \mathbb{R} : x \mapsto \frac{1}{x-2} \quad (\text{cannot divide by zero})$$










$$d : (0, \infty) \rightarrow \mathbb{R} : x \mapsto \log(x) \quad (\text{cannot log of zero or negative values})$$



# Notation — Open/Closed/Semi-Open Intervals on $\mathbb{R}$

In the previous slide I used interval notation to represent sets involving numbers.

Lets review that notation ...

Interval Notation	Set Notation	Graphical Representation	Informal Description
$[a, b]$	$\{x \in \mathbb{R} : a \leq x \leq b\}$		Closed finite interval <sup>‡</sup>
$(a, b)$	$\{x \in \mathbb{R} : a < x < b\}$		Open finite interval
$[a, b)$	$\{x \in \mathbb{R} : a \leq x < b\}$		Semi-open finite interval
$(a, b]$	$\{x \in \mathbb{R} : a < x \leq b\}$		Semi-open finite interval
$[a, \infty)$	$\{x \in \mathbb{R} : a \leq x < \infty\}$		Semi-open infinite interval
$(a, \infty)$	$\{x \in \mathbb{R} : a < x < \infty\}$		Open infinite interval
$(-\infty, b]$	$\{x \in \mathbb{R} : -\infty < x \leq b\}$		Semi-open infinite interval
$(-\infty, b)$	$\{x \in \mathbb{R} : -\infty < x < b\}$		Open infinite interval
$(-\infty, \infty)$	$\mathbb{R}$		The Real Line

**Table:** Intervals on the real line.

<sup>‡</sup>This is “the set of all real numbers  $x$ , such that  $a$  is less than or equal to  $x$ , and  $x$  is less than or equal to  $b$ .”

# Notation — Open/Closed/Semi-Open Intervals on $\mathbb{Z}$ (or $\mathbb{N}$ )

Similar notation applies to set involving integers, i.e.,  $\mathbb{Z}$  and  $\mathbb{N}$  ...

Interval Notation	Set Notation	Graphical Representation	Informal Description
$[a, b]$	$\{x \in \mathbb{Z} : a \leq x \leq b\}$		Closed finite interval <sup>§</sup>
$(a, b)$	$\{x \in \mathbb{Z} : a < x < b\}$		Open finite interval
$[a, b)$	$\{x \in \mathbb{Z} : a \leq x < b\}$		Semi-open finite interval
$(a, b]$	$\{x \in \mathbb{Z} : a < x \leq b\}$		Semi-open finite interval
$[a, \infty)$	$\{x \in \mathbb{Z} : a \leq x < \infty\}$		Semi-open infinite interval
$(a, \infty)$	$\{x \in \mathbb{Z} : a < x < \infty\}$		Open infinite interval
$(-\infty, b]$	$\{x \in \mathbb{Z} : -\infty < x \leq b\}$		Semi-open infinite interval ( $\mathbb{Z}$ only)
$(-\infty, b)$	$\{x \in -\infty < x < b\}$		Open infinite interval ( $\mathbb{Z}$ only)
$(-\infty, \infty)$	$\mathbb{Z}$		The set of integers ( $\mathbb{Z}$ only)

**Table:** Intervals on integers  $\mathbb{Z}$  (or  $\mathbb{N}$ ).

<sup>§</sup>This is “the set of all integers  $x$ , such that  $a$  is less than or equal to  $x$ , and  $x$  is less than or equal to  $b$ .”

# Notation — Open/Closed/Semi-Open Intervals on $\mathbb{Z}$ (or $\mathbb{N}$ )

Similar notation applies to set involving integers, i.e.,  $\mathbb{Z}$  and  $\mathbb{N}$  ...

Interval Notation	Set Notation	Graphical Representation	Informal Description
$[a, b]$	$\{x \in \mathbb{Z} : a \leq x \leq b\}$		Closed finite interval <sup>§</sup>
$(a, b)$	$\{x \in \mathbb{Z} : a < x < b\}$		Open finite interval
$[a, b)$	$\{x \in \mathbb{Z} : a \leq x < b\}$		Semi-open finite interval
$(a, b]$	$\{x \in \mathbb{Z} : a < x \leq b\}$		Semi-open finite interval
$[a, \infty)$	$\{x \in \mathbb{Z} : a \leq x < \infty\}$		Semi-open infinite interval
$(a, \infty)$	$\{x \in \mathbb{Z} : a < x < \infty\}$		Open infinite interval
$(-\infty, b]$	$\{x \in \mathbb{Z} : -\infty < x \leq b\}$		Semi-open infinite interval ( $\mathbb{Z}$ only)
$(-\infty, b)$	$\{x \in -\infty < x < b\}$		Open infinite interval ( $\mathbb{Z}$ only)
$(-\infty, \infty)$	$\mathbb{Z}$		The set of integers ( $\mathbb{Z}$ only)

Python  
(Why?)

Table: Intervals on integers  $\mathbb{Z}$  (or  $\mathbb{N}$ ).

<sup>§</sup>This is “the set of all integers  $x$ , such that  $a$  is less than or equal to  $x$ , and  $x$  is less than or equal to  $b$ .”

# Type Intervals in Programming Languages

Again, I want to impress on you, that the concept of different intervals is not just something to keep mathematicians awake at night ... it also keeps programmers awake ...

For example, a Google search of [why does python use semi open intervals](#) generates

## Why are Python ranges half-open (exclusive) instead of closed - Quora

<https://www.quora.com/Why-are-Python-ranges-half-open-exclusive-instead-of-close...> ▼

Because half-open intervals are easier to compose and reason with. You never have to think ... But a moderate amount of experience will convince you that they are far more pleasant to ... **Why do many websites use PHP** instead of Python?

## c++ - What is half open range and off the end value - Stack Overflow

<https://stackoverflow.com/questions/.../what-is-half-open-range-and-off-the-end-value> ▼

Oct 25, 2012 - A half-open range is one which includes the first element, but .... we can also use the half-opening range in the function signature which can be

## Why is SQL's BETWEEN inclusive rather than half-open? - Software ...

<https://softwareengineering.stackexchange.com/.../why-is-sqls-between-inclusive-rathe...> ▼

Aug 9, 2012 - ... (and apparently, so did the SQL designers) than a semi-open interval. ... the SQL standard is amended, don't use BETWEEN for dates/times.

# Review Exercises 1 (Definition of a Function)

## Question 1:

Consider the function defined by the rule  $x \mapsto x^2$  with domain of  $f$  equal to  $\{0, 1, 2, 3\}$ . Show that

$$\{(x, f(x)) | x \in \text{Dom}(f)\} \subseteq \mathbb{N} \times \mathbb{N}$$

## Question 2:

For each of the following incomplete function definitions construct a formal definition, assuming input is a real number.

(a)  $f(x) = \frac{1}{x^2-4}$

(b)  $f(x) = \frac{1}{x^2-10}$

(c)  $f(x) = \sqrt{x^2 - x - 6}$

## Question 3:

For each of the following incomplete function definitions construct a formal definition, assuming input is an element of  $\mathbb{N}$ .

(a)  $f(x) = \frac{1}{x^2-4}$

(b)  $f(x) = \frac{1}{x^2-10}$

(c)  $f(x) = \sqrt{x^2 - x - 6}$

# Outline

---

1. Definition of a Function	2
1.1. Definition Based on Relations	3
1.2. Function Notation	11
1.3. An Aside: Interval Notation	13
2. Function Properties	17
2.1. Surjective (Onto)	19
2.2. Injective (One-to-One)	21
2.3. Bijective (Injective and Surjective)	23

# Function Definition

Recall that when properly specifying a function we need the set of allowed inputs (domain) and a set large enough to contain all possible outputs (target) in addition to a rule/table connecting input to output values. So we have definition:

## Definition 6 (Function)

A **function**

$$f : \text{Dom}(f) \rightarrow \text{Target}(f) : x \mapsto f(x)$$

is any process ((multi-)rule, lookup table, etc) that generates a *single* output from every input value. Hence we specify:

- The  $\text{Dom}(f)$  is the set of allowed inputs and is called the “domain of  $f$ ”.
  - If the domain is not specified, then it is assumed to be the largest subset of  $\mathbb{R}$  (or  $\mathbb{Z}$  or  $\mathbb{N}$ ) whose values do not result in an invalid operation.
- The  $\text{Target}(f)$  is any set large enough to contain all possible outputs of  $f$  and is called the “target of  $f$ ”.
  - If the target is not specified, then it is assumed to be  $\mathbb{R}$  (or possibly  $\mathbb{Z}$  or  $\mathbb{N}$ ).
  - We work with the target set of functions because it is often much more difficult to determine the image set — the set of all output values.
- An assignment rule, that associates to every input  $x$  a unique output  $f(x)$ .

# Function Properties — Surjective

Since functions are relations, the relation properties are also function properties ... we just have some extra terminology ...

## Into vs. Onto

With a relation (so also a function) the image set (the set of all actual output) is a subset of the target:



- A function,  $f : A \rightarrow B$ , is **surjective** iff

$$\forall b \in B \quad \exists a \in A \quad (f(a) = b)$$

i.e., there is at least one arrow going to every point in  $B$ .



# Function Properties — Surjective

Since functions are relations, the relation properties are also function properties ... we just have some extra terminology ...

## Into vs. Onto

With a relation (so also a function) the image set (the set of all actual output) is a subset of the target:

$$\text{Im}(R) \subset T$$



$$\text{Im}(R) = T$$

or



- A function,  $f : A \rightarrow B$ , is **surjective** iff

$$\forall b \in B \quad \exists a \in A \quad (f(a) = b)$$

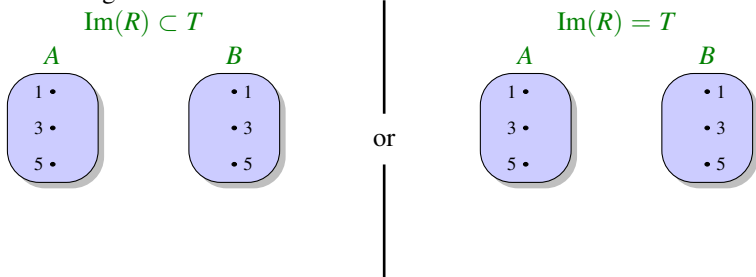
i.e., there is **at least one** arrow going to every point in  $B$ .

# Function Properties — Surjective

Since functions are relations, the relation properties are also function properties ... we just have some extra terminology ...

## Into vs. Onto

With a relation (so also a function) the image set (the set of all actual output) is a subset of the target:



- A function,  $f: A \rightarrow B$ , is **surjective** iff

$$\forall b \in B \quad \exists a \in A \quad (f(a) = b)$$

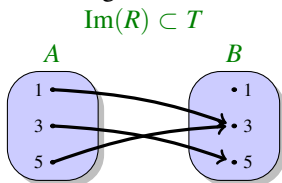
i.e., there is **at least one** arrow going to every point in  $B$ .

# Function Properties — Surjective

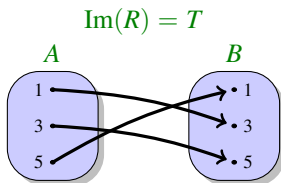
Since functions are relations, the relation properties are also function properties ... we just have some extra terminology ...

## Into vs. Onto

With a relation (so also a function) the image set (the set of all actual output) is a subset of the target:



or



- A function,  $f : A \rightarrow B$ , is **surjective** iff

$$\forall b \in B \quad \exists a \in A \quad (f(a) = b)$$

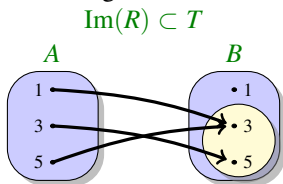
i.e., there is **at least one** arrow going to every point in  $B$ .

# Function Properties — Surjective

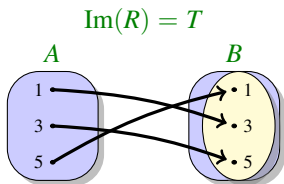
Since functions are relations, the relation properties are also function properties ... we just have some extra terminology ...

## Into vs. Onto

With a relation (so also a function) the image set (the set of all actual output) is a subset of the target:



or



- A function,  $f : A \rightarrow B$ , is **surjective** iff

$$\forall b \in B \quad \exists a \in A \quad (f(a) = b)$$

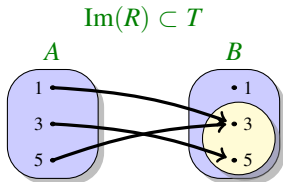
i.e., there is **at least one** arrow going to every point in  $B$ .

# Function Properties — Surjective

Since functions are relations, the relation properties are also function properties ... we just have some extra terminology ...

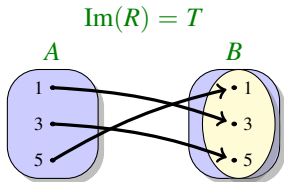
## Into vs. Onto

With a relation (so also a function) the image set (the set of all actual output) is a subset of the target:



A function,  $f$ , in which the image is a proper subset of the target is said to be an **into** function (or **not surjective**).

or



A function,  $f$ , in which the image is equal to the target is said to be an **onto** function (or **surjective**).

- A function,  $f : A \rightarrow B$ , is **surjective** iff

$$\forall b \in B \quad \exists a \in A \quad (f(a) = b)$$

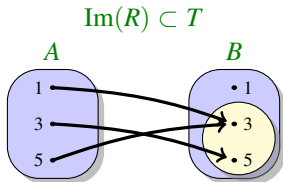
i.e., there is **at least one** arrow going to every point in  $B$ .

# Function Properties — Surjective

Since functions are relations, the relation properties are also function properties ... we just have some extra terminology ...

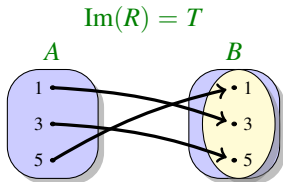
## Into vs. Onto

With a relation (so also a function) the image set (the set of all actual output) is a subset of the target:



A function,  $f$ , in which the image is a proper subset of the target is said to be an **into** function (or **not surjective**).

or



A function,  $f$ , in which the image is equal to the target is said to be an **onto** function (or **surjective**).

- A function,  $f : A \rightarrow B$ , is **surjective** iff

$$\forall b \in B \quad \exists a \in A \quad (f(a) = b)$$

i.e., there is **at least one** arrow going to every point in  $B$ .

# Application

Why is surjective important ?

- If a function is surjective then every element in the target set can be generated/outputted given suitable input.
- If a function is **not** surjective then some elements in the target set **cannot** be generated/outputted regardless of the input — goal of **plausibly deniable encryption**.

## Deniable encryption

From Wikipedia, the free encyclopedia

In **cryptography** and **steganography**, plausibly **deniable encryption** describes **encryption** techniques where the existence of an encrypted file or message is deniable in the sense that an adversary cannot prove that the **plaintext** data exists.<sup>[1]</sup>

Modern deniable encryption techniques exploit the fact that without the key, it is infeasible to distinguish between ciphertext from **block ciphers** and data generated by a **cryptographically secure pseudorandom number generator** (the cipher's **pseudorandom permutation** properties).<sup>[7]</sup>

# Injective (One-to-One)

A relation (or function) from set  $A$  to set  $B$  is one-to-one if every element in  $B$  has at most one incoming arrow.

## Definition 7 (Injective (One-to-One))

A function (or relation) from set  $A$  to set  $B$  is **one-to-one** (or **injective** iff

$$\underbrace{f(a_1) = b_1 \quad \wedge \quad f(a_2) = b_1}_{f(a_1) = f(a_2)} \implies a_1 = a_2$$

- Make sure you are happy with reconciling “most one incoming arrow” with the above definition, which effectually says “if element  $b$  has an incoming arrow from  $a_1$  and an incoming arrow from  $a_2$  then  $a_1 = a_2$ , i.e., the two incoming arrows are the same arrow”.  
Or “equal outputs implies equal inputs”.
- The contrapositive proposition is typically used when proving a function is injective.

$$a_1 \neq a_2 \implies f(a_1) \neq f(a_2)$$

i.e., different inputs implies different outputs.



# Application

I should talk here about injective functions used in encryption or lossless compression (zip, rar, 7z, etc), but instead I will talk about a function that is **not** injective to illustrate the importance of this property.

- A **hash** function is any function that return deterministic<sup>¶</sup> but generally irreversible<sup>||</sup> output values for given inputs.
- Hash functions are a fundamental component in cryptography, and **main attack strategy** is to find two different inputs that generate the same output.

## Hash Collision Attack

A Hash Collision Attack is an attempt to find two input strings of a hash function that produce the same hash result. Because hash functions have infinite input length and a predefined output length, there is inevitably going to be the possibility of two different inputs that produce the same output hash. If two separate inputs produce the same hash output, it is called a **collision**. This collision can then be exploited by any application that compares two hashes together – such as password hashes, file integrity checks, etc.

Practically speaking, there are several ways a hash collision could be exploited. if the attacker was offering a file download and showed the hash to prove the file's integrity, he could switch out the file download for a different file that had the same hash, and the person downloading it would be unable to know the difference. The file would appear valid as it has the same hash as the supposed real file.

---

<sup>¶</sup>means, not random

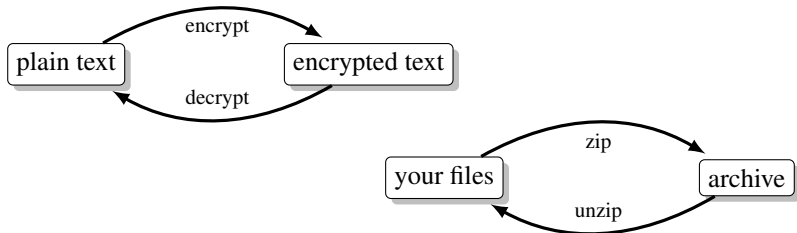
<sup>||</sup>difficult to figure out the input if you only know the output

# Bijective (Injective and Surjective)

## Definition 8 (Bijective (Injective and Surjective))

A function,  $f$ , from set  $A$  to set  $B$  is said to be **bijective** (or a **bijection**) iff it is both injective and surjective.

- In terms of Venn diagram, a bijective function has
  - exactly one arrow leaving every element in the source (always true for a function).
  - exactly one arrow entering every element in the target.
- Bijective functions are reversible\*\*



---

\*\*Just because something is reversible it says nothing about the relative difficulty of computing the different directions.

# Review Exercises 2 (Function Properties)

## Question 1:

Let  $S = \{a, b, c, d\}$  and  $T = \{1, 2, 3, 4, 5, 6, 7\}$ . Which of the following relations on  $S \times T$  is a function.

(a)  $\{(a, 4), (d, 3), (c, 3), (b, 2)\}$

(b)  $\{(a, 5), (c, 4), (d, 3)\}$

## Question 2:

Classify each of the following functions as surjective, injective and bijective.

(a)  $f: \mathbb{R} \rightarrow \mathbb{R} : x \mapsto 3x + 1$

(d)  $f: \mathbb{R} \rightarrow \mathbb{R} : m \mapsto m + 2$

(g)  $f: \mathbb{N} \rightarrow \mathbb{N} : m \mapsto 2m$

(b)  $f: \mathbb{N} \rightarrow \mathbb{N} : x \mapsto 3x + 1$

(e)  $f: \mathbb{N} \rightarrow \mathbb{N} : m \mapsto m + 2$

(h)  $f: \mathbb{R} \rightarrow \mathbb{R} : m \mapsto 2m$

(c)  $f: \mathbb{Q} \rightarrow \mathbb{Q} : x \mapsto 3x + 1$

(f)  $f: \mathbb{Q} \rightarrow \mathbb{Q} : m \mapsto m + 2$

(i)  $g: \mathbb{Z} \rightarrow \mathbb{Z} : m \mapsto 2m^2 - 7$

## Question 3:

Let  $A = \{1, 2, 3, 4\}$  and  $B = \{a, b, c, d\}$ . Determine which of the following are functions. For functions classify as surjective, injective and bijective.

(a)  $f \subseteq A \times B$ , where  $f = \{(1, a), (2, b), (3, c), (4, d)\}$ .

(b)  $g \subseteq A \times B$ , where  $g = \{(1, a), (2, a), (3, b), (4, d)\}$ .

(c)  $h \subseteq A \times B$ , where  $h = \{(1, a), (2, b), (3, c)\}$ .

(d)  $k \subseteq A \times B$ , where  $k = \{(1, a), (2, b), (2, c), (3, a), (4, a)\}$ .

(e)  $L \subseteq A \times A$ , where  $L = \{(1, 1), (2, 1), (3, 1), (4, 1)\}$ .

## Question 4:

If  $|A|$  and  $|B|$  are both finite, how many different functions are there from  $A$  to  $B$ ?