Discrete Mathematics

Topic 04 : Relations and Functions

Lecture 03 : Function Concepts and Definitions

Dr Kieran Murphy

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

Autumn Semester, 2025/26

Outline

- Definition of a Function
- Function Properties

Outline

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 $A \times B$

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

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 $A \times B$

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

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

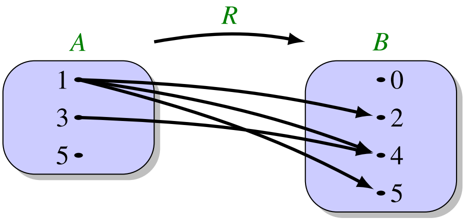
Recall our definition of a relation, R from set A to set B

Definition 1 (Relation)

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

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

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

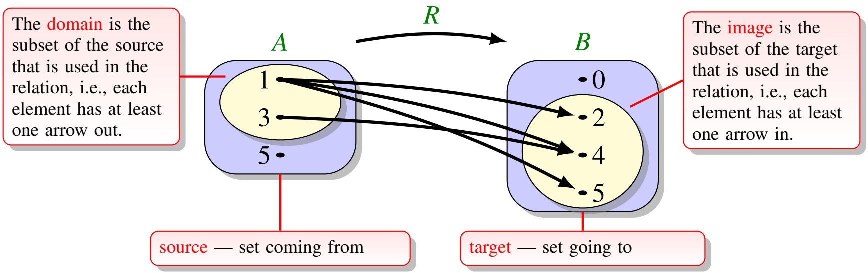
Recall our definition of a relation, R from set A to set B

Definition 1 (Relation)

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

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

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 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

```
double f(double a) {
    double result = 0.0;

    // do calculation

    return result;
}
```

*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

```python
def f(a):
    result = 0.0

    # do calculation

    return result
```

```java
double f(double a) {
    double result = 0.0;

    // do calculation

    return result;
}
```

*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 II

…our issues continued …

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

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

- 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.

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

- 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.

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

- 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.

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

- 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 `double`s 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 $\mathrm{Dom}(R)$
- At most one output for each input
  - At most one arrow leaving each element in A.
  - If $(a, b) \in R$ and $(a, c) \in R$ then $b = c \quad \forall a \in A$

$\Rightarrow$ 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 $\mathrm{Dom}(R)$
- At most one output for each input
  - At most one arrow leaving each element in A.
  - If $(a, b) \in R$ and $(a, c) \in R$ then $b = c \quad \forall a \in A$

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

Graphically, we have ...

Function Definition Based on a Relation

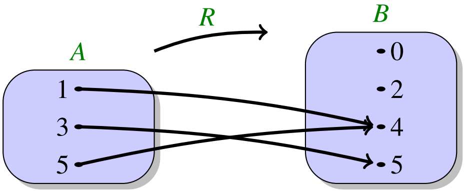
Definition 2 (Function)

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 $\mathrm{Dom}(R)$
- At most one output for each input
  - At most one arrow leaving each element in A.
  - If $(a, b) \in R$ and $(a, c) \in R$ then $b = c \quad \forall a \in A$

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

Graphically, we have ...



Function Definition Based on a Relation

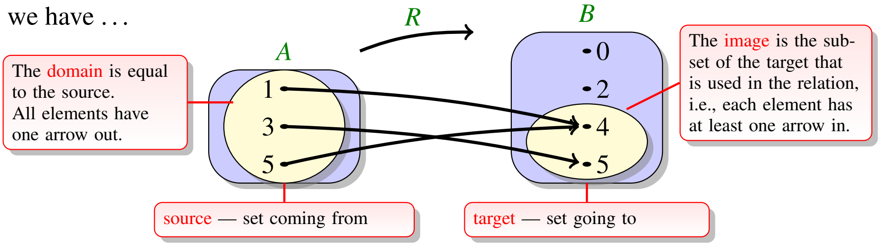
Definition 2 (Function)

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 $\mathrm{Dom}(R)$
- At most one output for each input
  - At most one arrow leaving each element in A.
  - If $(a, b) \in R$ and $(a, c) \in R$ then $b = c \quad \forall a \in A$

$\Rightarrow$ 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

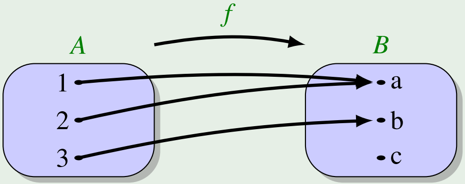
Example 3 (Specifying a function as a set of ordered pairs)

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

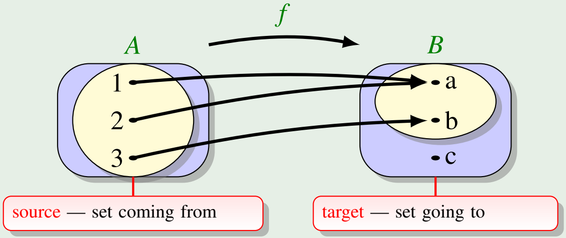
Example 3 (Specifying a function as a set of ordered pairs)

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

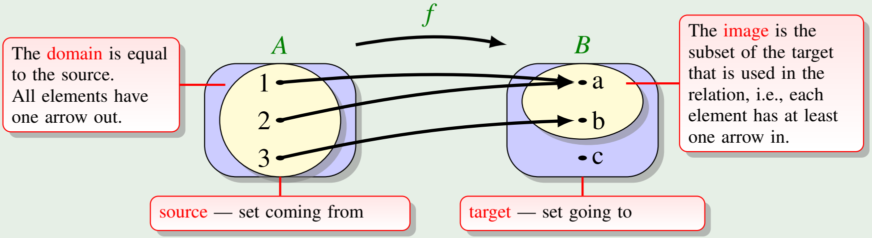
Example 3 (Specifying a function as a set of ordered pairs)

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.



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 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

Domain, $\mathrm{Dom}(f)$, is given by the input in each row. No element is repeated,

input	output
1	a
2	a
3	b
	c

Image , $\mathrm{Im}(f)$, is given by the output in each row.

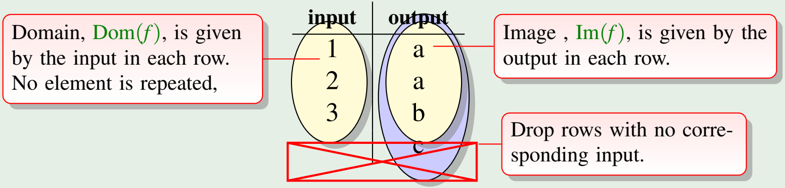
- Lookup tables are frequently used in computing in situations where memory is cheaper than computing cycles.
  - Number theory libraries would small primes up to, say 1000, and compute others as needed.
  - ASCII table and now unicode.

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



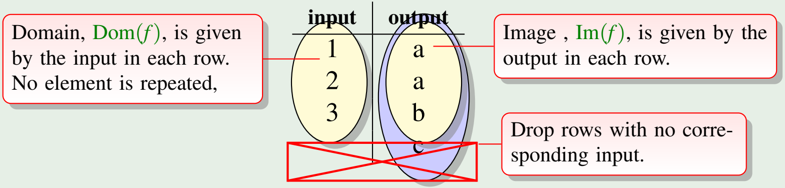
- Lookup tables are frequently used in computing in situations where memory is cheaper than computing cycles.
  - Number theory libraries would small primes up to, say 1000, and compute others as needed.
  - ASCII table and now unicode.

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		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	Start of Header	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	Start of Text	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	End of Text	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	End of Transmission	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	Enquiry	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	Acknowledgment	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	Bell	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	Backspace	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	Horizontal Tab	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	Line feed	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	Vertical Tab	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	Form feed	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	Carriage return	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	Shift Out	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	Shift In	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	Data Link Escape	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	Device Control 1	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	Device Control 2	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	Device Control 3	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	Device Control 4	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	Negative Ack.	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	Synchronous idle	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	End of Trans. Block	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	Cancel	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	End of Medium	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	Substitute	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z

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		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	Start of Header	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	Start of Text	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	End of Text	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	End of Transmission	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	Enquiry	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	Acknowledgment	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	Bell	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	Backspace	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	Horizontal Tab	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	Line feed	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	Vertical Tab	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	Form feed	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	Carriage return	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	Shift Out	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	Shift In	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	Data Link Escape	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	Device Control 1	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	Device Control 2	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	Device Control 3	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	Device Control 4	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	Negative Ack.	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	Synchronous idle	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	End of Trans. Block	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	Cancel	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	End of Medium	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	Substitute	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z

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		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	Start of Header	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	Start of Text	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	End of Text	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	End of Transmission	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	Enquiry	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	Acknowledgment	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	Bell	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	Backspace	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	Horizontal Tab	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A																j	j
11	B																	
12	C																	
13	D																	
14	E																	
15	F																o	o
16	10																p	p
17	11																q	q
18	12																r	r
19	13	023	Device Control 3	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	Device Control 4	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	Negative Ack.	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	Synchronous idle	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	End of Trans. Block	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	Cancel	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	End of Medium	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	Substitute	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z

```python
# convert from character to ASCII

print ("Character 'A' map to ", ord('A'))
print ("Character '1' map to ", ord('1'))

# convert from ACSII to character
print ("Integer 43 maps to character ", chr(43))
```

```
Character 'A' map to  65
Character '1' map to  49
Integer 43 maps to character +
```

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

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

- 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†, so we will cover them again using function notation.

† 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*

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

or

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

- *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

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

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

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

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

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

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

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[‡]
(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, ∞)	$\{x \in \mathbb{R} : a < x < \infty\}$		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.

[‡] 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[§]
(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, ∞)	$\{x \in \mathbb{Z} : a < x < \infty\}$		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 $\mathbb{N}$).

[§] 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 $\mathbb{N}$ …

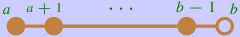
Interval Notation	**Set Notation**	**Graphical Representation**	**Informal Description**
$[a, b]$	$\{x \in \mathbb{Z} : a \leq x \leq b\}$		Closed finite interval[§]
(a, b)	$\{x \in \mathbb{Z} : a < x < b\}$		Open finite interval
$[a, b)$	$\{x \in \mathbb{Z} : a \leq x < b\}$		Semi-open finite interval Python (Why?)
$(a, b]$	$\{x \in \mathbb{Z} : a < x \leq b\}$		Semi-open finite interval
$[a, \infty)$	$\{x \in \mathbb{Z} : a \leq x < \infty\}$		Semi-open infinite interval
(a, ∞)	$\{x \in \mathbb{Z} : a < x < \infty\}$		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 $\mathbb{N}$).

[§] 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 inst

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

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

$$f : \mathrm{Dom}(f) \to \mathrm{Target}(f) : x \mapsto f(x)$$

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

- The $\mathrm{Dom}(f)$ 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 $\mathrm{Target}(f)$ 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:

or

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

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

i.e., there is **al 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:

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

or

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

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

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

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

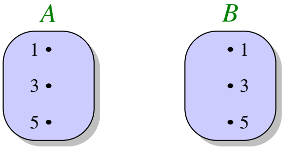
Function Properties — Surjective

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

Into vs. Onto

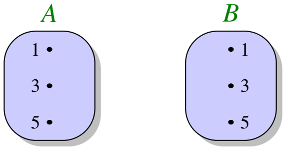
With a relation (so also a function) the image set (the set of all actual output) is a subset of the target:

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



or

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



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

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

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

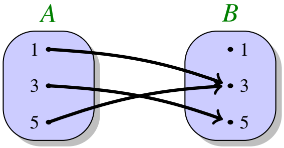
Function Properties — Surjective

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

Into vs. Onto

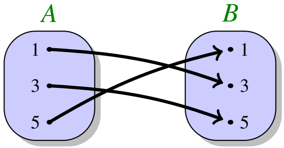
With a relation (so also a function) the image set (the set of all actual output) is a subset of the target:

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



or

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



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

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

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

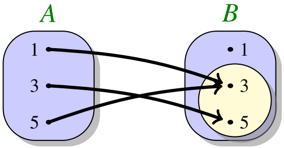
Function Properties — Surjective

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

Into vs. Onto

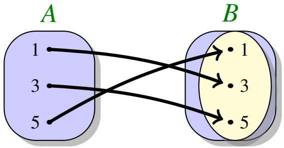
With a relation (so also a function) the image set (the set of all actual output) is a subset of the target:

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



or

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



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

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

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

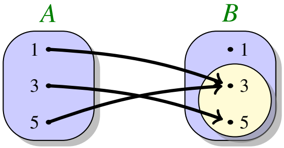
Function Properties — Surjective

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:

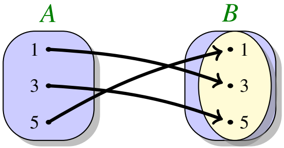
$$\mathrm{Im}(R) \subset T$$



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

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



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

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

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

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

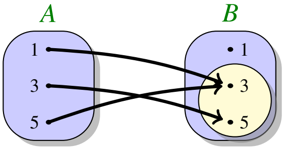
Function Properties — Surjective

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:

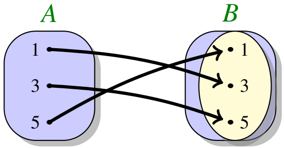
$$\mathrm{Im}(R) \subset T$$



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

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



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

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

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

i.e., there is **al 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 at least one element 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.[1]

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).[7]

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

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

- Make sure you are happy with reconciling "most one incoming arrow" with the above definition, which 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[¶] but generally irreversible[‖] 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.

[¶] means, not random

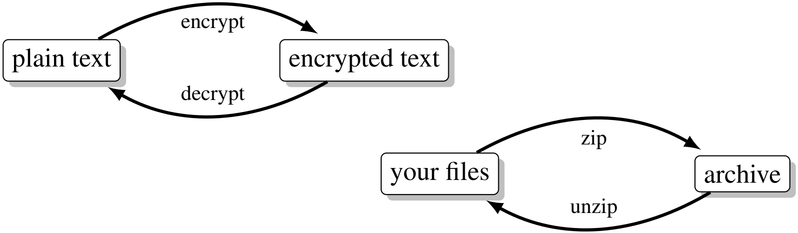
[‖] difficult to figure out the input if you only know the output

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) $f : \mathbb{R} \to \mathbb{R} : x \mapsto 3x + 1$

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

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

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

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

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

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

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

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

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 $L = \{(1, 1), (2, 1), (3, 1), (4, 1)\}$.