Sets

A **set** is a collection of (mathematical) objects.

Each object x is either contained in a set S or not.

We write $x \in S$ to mean 'x is an element of S' or 'S contains x'

We write $x \notin S$ to mean 'x is not an element of S' or 'S does not contain x'

Finite sets:

- $\emptyset$ the **empty set**. It contains no objects.
  - $*$ In particular $\emptyset \notin \emptyset$
- $\{x\}$ the set containing only x
- $\{x, y, z\}$ the set containing x, y, and z, but nothing else.

Some infinite sets:

- $\mathbb{N}$ the natural numbers: 0, 1, 2, 3, etc. **Note 0 is included!**
- $\mathbb{Z}$ the integers: 0, -1, 1, -2, 2, -3, 3, etc.
- $\mathbb{R}$ the real numbers.
- Don't confuse zero with the empty set: $0 \neq \emptyset$.

Equality: Two sets are considered equal ($S = T$) iff they contain exactly the same objects.

- Therefore there is only one empty set (all empty set are equal).

Union: $S \cup T$ is the set that contains all objects contained either in S or in T.

- $x \in (S \cup T)$ exactly if $x \in S$ or $x \in T$

Intersection: $S \cap T$ is the set that contains all objects contained both in S and in T.

- $x \in (S \cap T)$ exactly if $x \in S$ and $x \in T$

Subtraction: $S - T$ is the set that contains all objects in S that are not in T.

- $x \in (S - T)$ exactly if $x \in S$ but $x \notin T$

Subsets:

- $S \subseteq T$ means 'S is a **subset** of T', i.e. every object in S is also in T.
- In particular, $S \subseteq S$ and $\emptyset \subseteq S$, for any set S.
- E.g. $\mathbb{N} \subseteq \mathbb{Z}$

Strict subsets:

- $S \subset T$ means 'S is a subset of T and not equal to T'.
- E.g. $\{1, 2\} \subset \{1, 2, 3\}$ but $\{1, 2\} \not\subset \{1, 2\}$

Contigous sets of integers:

- $\{i, ..j\}$ the set of all integers greater or equal to i and less than j
- E.g. $\{3, ..7\} = \{3, 4, 5, 6\}$.

The size of a set

- $|S|$ the number of members in S.

Don't confuse singleton sets with their single element.

- $1 \in \{1, 2, 3\}$ but $\{1\} \notin \{1, 2, 3\}$
- $\{1\} \in \{\{1\}, \{2\}, \{3\}\}$ but $1 \notin \{\{1\}, \{2\}, \{3\}\}$
- Also $\emptyset \subseteq \{1, 2, 3\}$ but $\emptyset \notin \{1, 2, 3\}$

Set Comprehension

Let $\mathcal{V}$ be a variable (such as x, y, or α) and $\mathcal{S}$ be an expression that describes a set.

Filtering. Let $\mathcal{A}$ be some boolean expression. (I.e. an expression whose value is $\mathfrak{true}$ or $\mathfrak{false}$).

- $\{x \in \mathcal{S} \mid \mathcal{A}\}$ means 'that subset of S containing elements satisfying description $\mathcal{A}$'.
- For example:
  - $\{x \in \mathbb{R} \mid x > 0\}$ is the set of positive real numbers
  - $\{x \in \mathbb{N} \mid x/3 \in \mathbb{N}\}$ is the set of natural number that are multiples of 3.

Mapping. Let $\mathcal{E}$ be some mathematical expression that (typically) depends on x.

- $\{\mathcal{V} \in \mathcal{S} \cdot \mathcal{E}\}$ is 'the set of values of expression $\mathcal{E}$ where x varies over all elements of S.'
- For example:
  - $\{x \in \mathbb{N} \cdot 2 \times x\}$ is the set of all even natural numbers.
  - $\left\{x \in \mathbb{Z} \cdot x^2\right\}$ is the set of all square numbers.
- [The mapping notation is rather uncommon. Most authors would write $\left\{x^2 \mid x \in \mathbb{Z}\right\}$ where I write $\left\{x \in \mathbb{Z} \cdot x^2\right\}$. I use this notation for consistency with other notations used in the course]

Filter and map

- $\{\mathcal{V} \in \mathcal{V} \mid \mathcal{A} \cdot \mathcal{E}\}$ — first filter, then map

- For example:
  - $\left\{x \in \mathbb{Z} \mid 0 \leq x < 8 \cdot x^2\right\} = \{0, 1, 4, 9, 16, 25, 36, 49\}$

Check

What is $\{a \in \mathbb{N} \mid a < 10 \cdot 3 + a\}$

A: $\{1, 2, 3\}$

B: $\{3, ..13\}$

C: $\{-3, ..7\}$

D: $\{3, ..10\}$

Pairs

Pairs, triples, and tuples:

- (x, y) is a **pair** consisting of x on the left and y on the right.
- Note that $(x, y) \neq (y, x)$ unless $x = y$.
- We can also have **triple**s (x, y, z) and in general n-**tuple**s for $n \geq 2$.
- E.g. $(1, \pi, \mathfrak{true}, \text{‘a’}, \emptyset)$ is a 5-tuple.

Cartesian product:

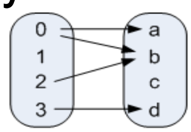
- $S \times T$ is the set of all pairs (x, y) such that $x \in S$ and $y \in T$.
- $S \times T \times U$ is the set of all triples (x, y, z) such that $x \in S$, $y \in T$, and $z \in U$.
- Etc. Note that $S \times T \times U$ is not quite the same as $S \times (T \times U)$ or $(S \times T) \times U$ although all three sets have the same number of members. E.g. $(5, \pi, \mathfrak{false}) \in \mathbb{N} \times \mathbb{R} \times \mathbb{B}$ whereas $(5, (\pi, \mathfrak{false})) \in \mathbb{N} \times (\mathbb{R} \times \mathbb{B})$

Relations and functions.

- A **binary relation** is any triple (S, T, G) where S and T are sets and $G \subseteq S \times T$.
- We call S the **source**, T the **target**, and G the **graph** of the relation.
  - **Notation:** I'll write $x \mapsto y$ for (x, y) when dealing with relations and function.
  - Example: Let $S = \{0, 1, 2, 3\}$, $T = \{$‘a’, ‘b’, ‘c’, ‘d’$\}$,

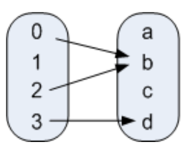
    $$G = \{0 \mapsto \text{‘a’}, 0 \mapsto \text{‘b’}, 2 \mapsto \text{‘b’}, 3 \mapsto \text{‘d’}\}$$

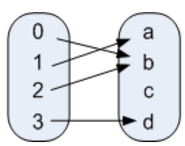
    Then (S, T, G) is a binary relation, illustrated as

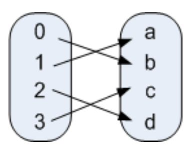


Binary Relation

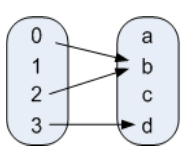
    Here are some more (illustrations of) binary relations.

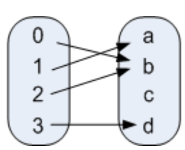


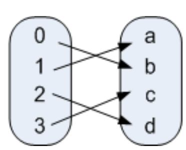




- A **partial function** (S, T, G) is a relation such that, for each $x \in S$, there is at most one y such that $(x, y) \in G$.
  - Example: Here are some examples of partial functions



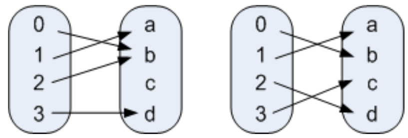




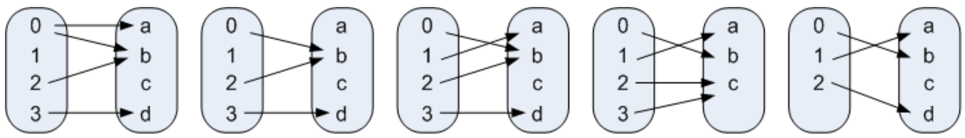
- A **total function** (S, T, G) is a relation such that, for each $x \in S$, there is exactly one y such that $(x, y) \in G$.
- (Note that each ‘total function’ is also a ‘partial

function'!)

* Examples:



- Binary relations, partial functions and total functions



Binary Relation | Binary Relation, Partial Function | Binary Relation, Partial Function, Total Function | Binary Relation, Partial Function, Total Function, Onto function | Binary Relation, Partial Function, Total Function, One-one function

- Let $f = (S, T, G)$ be a partial function and $x \in S$.
  * We say that f is **defined for** x if there is a $y \in T$ such that $(x \mapsto y) \in G$.
  * (Since f is a partial functions such a y will be unique.)
  * (If f is a total function then it defined for all x in S.)
  * If f is defined for x, $f(x)$ means 'that $y \in T$ such that $(x, y) \in G$'.
- $S \overset{\mathrm{tot}}{\to} T$ is the set of all total functions with source S and target T.
- $S \overset{\mathrm{par}}{\to} T$ is the set of all partial functions with source S and target T.

Domain and Range

The **domain** of this relation is the set of elements that map to something

$$\mathrm{dom}(R) = \{x \in S \mid R \text{ is defined for } x\}$$

(For a total function $f : S \overset{\text{tot}}{\to} T$ the domain is the same as the source S, i.e. $\operatorname{dom}(f) = S$)

The **range** of R is the set of elements that appear as the right component of a pair in the graph

$\operatorname{rng}(R) = \{y \in T \mid$ there is an $x \in S$ such that $(x \mapsto y) \in G\}$

(For a total function $f : S \overset{\text{tot}}{\to} T$ the range of f may or may not be T.)

Check

Which of the following statments is **not true**

A: All total functions are also partial functions

B: All partial functions are also binary relations

C: A total function relates every item in its source to exactly one item in its target

D: A total function relates every item in its target to exactly one item in its source

More Examples

- Consider $(\mathbb{Z}, \mathbb{Z}, J)$ and
$$J = \{(a \mapsto b) \in \mathbb{Z} \times \mathbb{Z} \mid b = a \times a\}$$
  - ∗ This is a total function (and hence also a partial function and a relation).
  - ∗ Its domain is $\mathbb{Z}$
  - ∗ Its range is $\{0, 1, 4, 9, ...\}$
- Consider $(\mathbb{R}, \mathbb{R}, K)$ where
$$K = \{(x \mapsto y) \in \mathbb{R} \times \mathbb{R} \mid x \times y = 1\}$$
  - ∗ This is a partial function. It is not total since there is no $(x \mapsto y)$ pair with $x = 0$.
  - ∗ Its domain and range is $\mathbb{R} - \{0.0\}$
- Consider $(\mathbb{R}, \mathbb{R}, L)$ where
$$L = \{(x \mapsto y) \in \mathbb{R} \times \mathbb{R} \mid y \times y = x\}$$
- This is not a function since we have $4 \mapsto 2$ and $4 \mapsto -2$.
  - ∗ It has a domain of $\{x \in \mathbb{R} \mid x \geq 0\}$ and a range of $\mathbb{R}$.

Propositional logic

We assume a set $\mathbb{B}$ of size 2 $\mathbb{B} = \{\mathfrak{true}, \mathfrak{false}\}$

Implication

Define a function $(\Rightarrow) \in \mathbb{B} \times \mathbb{B} \xrightarrow{\text{tot}} \mathbb{B}$. so that, for all $q \in \mathbb{B}$, we have the following laws

$$(\mathfrak{false} \Rightarrow q) = \mathfrak{true} \text{ “False implies anything”}$$
$$(\mathfrak{true} \Rightarrow q) = q \text{ “Identity”}$$

In table form

p	q	$p \Rightarrow q$
$\mathfrak{false}$	$\mathfrak{false}$	$\mathfrak{true}$
$\mathfrak{false}$	$\mathfrak{true}$	$\mathfrak{true}$
$\mathfrak{true}$	$\mathfrak{false}$	$\mathfrak{false}$
$\mathfrak{true}$	$\mathfrak{true}$	$\mathfrak{true}$

$\Rightarrow$ is called **implication**.

Its left operand is called the **antecedant** and its right operand is called the **consequent**.

Some laws

$$(p \Rightarrow p) = \mathfrak{true} \text{ Reflexivity}$$
$$(p \Rightarrow \mathfrak{true}) = \mathfrak{true} \text{ Domination}$$

In most cases $p \Rightarrow q$ corresponds to the English phrase “if p, then q”.

Example: A restaurant has a policy "Anyone drinking wine must be over 18 years of age." We can rephrase this as

"If you drink wine, then you are over 18 years of age."

Consider a family of 4.

	Drink	Age	Rule followed
Alice	Cola	10	Yes
Bob	Tea	42	Yes
Ching	Wine	17	No
Deepa	Wine	43	Yes

Example: A subroutine is documented as follows

/** If x is positive and less than 10^4,
* then the result is the square of x.
*/
int square(int x)

We run 4 tests:

x	Result	Documention obeyed
-23	17	Yes
-8	64	Yes
1	3	No
10	100	Yes

The subroutine passed 3 of 4 tests.

Follows from

$\Leftarrow$ is called **follows from**. It is simply implication turned around

$$(p \Leftarrow q) = (q \Rightarrow p)$$

It corresponds to the English phrase "p if q" or "p follows from q".

p	q	$p \Leftarrow q$
$\mathfrak{false}$	$\mathfrak{false}$	$\mathfrak{true}$
$\mathfrak{false}$	$\mathfrak{true}$	$\mathfrak{false}$
$\mathfrak{true}$	$\mathfrak{false}$	$\mathfrak{true}$
$\mathfrak{true}$	$\mathfrak{true}$	$\mathfrak{true}$

Negation

Define $(\neg) \in \mathbb{B} \overset{\text{tot}}{\to} \mathbb{B}$ such that

$$\neg p = (p \Rightarrow \mathfrak{false}) \text{, for all } p \in \mathbb{B}$$

In table form

p	$\neg p$
$\mathfrak{false}$	$\mathfrak{true}$
$\mathfrak{true}$	$\mathfrak{false}$

Some laws

$$\neg\neg p = p \text{ Involution}$$

$$(p \Rightarrow q) = (\neg q \Rightarrow \neg p) \text{ Contrapositive}$$

$$(p \Rightarrow \mathfrak{false}) = \neg p \text{ Anti-identity}$$

Conjunction (AND) and disjunction (OR)

Define **disjunction** (OR) by

$$(\vee) \ \in \ \mathbb{B} \times \mathbb{B} \stackrel{\text{tot}}{\to} \mathbb{B}$$
$$(p \vee q) \ = \ (\neg p \Rightarrow q)$$

and **conjunction** (AND) by

$$(\wedge) \ \in \ \mathbb{B} \times \mathbb{B} \stackrel{\text{tot}}{\to} \mathbb{B}$$
$$(p \wedge q) \ = \ \neg (p \Rightarrow \neg q)$$

The operands of $\wedge$ are called **conjuncts**.

The operands of $\vee$ are called **disjuncts**.

Some laws

$$\left.\begin{array}{l}(\mathfrak{true} \wedge p) = p \\ (\mathfrak{false} \vee p) = p\end{array}\right\} \text{Identity}$$

$$\left.\begin{array}{l}(\mathfrak{false} \wedge p) = \mathfrak{false} \\ (\mathfrak{true} \vee p) = \mathfrak{true}\end{array}\right\} \text{Domination}$$

$$\left.\begin{array}{l}(p \wedge p) = p \\ (p \vee p) = p\end{array}\right\} \text{Idempotence}$$

$$\left.\begin{array}{l}(p \wedge q) = (q \wedge p) \\ (p \vee q) = (q \vee p)\end{array}\right\} \text{Commutativity}$$

$$\left.\begin{array}{l}(p \wedge q) \wedge r = p \wedge (q \wedge r) \\ (p \vee q) \vee r = p \vee (q \vee r)\end{array}\right\} \text{Associativity}$$

$$\left.\begin{array}{l}(p \wedge (q \vee r)) = ((p \wedge q) \vee (p \wedge r)) \\ (p \vee (q \wedge r)) = ((p \vee q) \wedge (p \vee r))\end{array}\right\} \text{Distributivity}$$

$$\left.\begin{array}{l}(p \wedge \neg p) = \mathfrak{false} \\ (p \vee \neg p) = \mathfrak{true}\end{array}\right\} \left\{\begin{array}{l}\text{law of contradiction} \\ \text{law of excluded middle}\end{array}\right.$$

$$\left.\begin{array}{l}\neg(p \wedge q) = (\neg p \vee \neg q) \\ \neg(p \vee q) = (\neg p \wedge \neg q)\end{array}\right\} \text{De Morgan's laws}$$

$(p \Rightarrow q) = (\neg p \vee q)$ Material implication

$(p \Rightarrow q) = (\neg q \Rightarrow \neg p)$ Contrapositive law

$(p \wedge q \Rightarrow r) = (p \Rightarrow (q \Rightarrow r))$ Shunting

$(p \wedge q \Rightarrow r) = ((p \Rightarrow r) \vee (q \Rightarrow r))$ Distributivity

$(p \vee q \Rightarrow r) = ((p \Rightarrow r) \wedge (q \Rightarrow r))$ Distributivity

$(p \Rightarrow q \wedge r) = ((p \Rightarrow q) \wedge (p \Rightarrow r))$ Distributivity

$(p \Rightarrow q \vee r) = ((p \Rightarrow q) \vee (p \Rightarrow r))$ Distributivity

if $p \Rightarrow q$ and $q \Rightarrow r$ then $p \Rightarrow r$ Transitivity

Check

Which of the following is not equivalent to

Paul likes cats $\Rightarrow$ Paul is a good squash player

A: Paul is not a good squash player $\Rightarrow$ Paul does not like cats

B: Either Paul is a good squash player or Paul does not like cats or both.

C: Paul likes cats and therefore Paul is a good squash player

D: It is not the case that Paul likes cats and is not good squash player.

Equivalence and XOR

Define

$$(p \Leftrightarrow q) \;=\; ((p \Rightarrow q) \wedge (q \Rightarrow p))$$
$$(p \not\Leftrightarrow q) \;=\; \neg(p \Leftrightarrow q)$$

$(p \Leftrightarrow q)$ is often called **equivalence**. It is really just **equality** for Boolean values.

$(p \not\Leftrightarrow q)$ is called **exclusive or**

Notation

This Course	Digital Logic	C/C++/ Java	C/C++/Java bitwise	Other
$\Rightarrow$				$\supset, \rightarrow$
$\wedge$	$\cdot$	&&	&	
$\vee$	$+$	||	|	
$\Leftrightarrow$		==		$\leftrightarrow, \equiv$
$\not\Leftrightarrow$	$\oplus$	!=	^	+
$\neg$	$\overline{}$	!	~	$\sim$

Predicate Logic

In natural language, one often wants to express declarations such as

- All flavours of ice-cream are good.
- Some people like peanut butter.
- The Q output is always equal to the D input of the previous cycle.
- The system will be in the initial state within 5 seconds of the reset button being depressed.

To treat such sentences mathematically we extend logic with "**quantifiers**"

- $\forall$, pronounced "for all", and
- $\exists$, pronounced "exists".

You can say that $\forall$ and $\exists$ have the same relationship to $\wedge$ and $\vee$ (respectively) as $\sum$ has to $+$.

We will extend our 2-valued propositional logic to deal with the quantifiers.

First, though, we look at substitution.

Substitution

Free and bound occurrences of variables

In Engineering, we often use variables to represent quantities in the real-world and boolean expressions containing variables to represent constraints on those quantities, imposed by nature or by an engineered system. For example, we might write

$$0 \le x < 1$$

to express that the x coordinate of the position of something (say a robot's hand) is constrained within certain limits. A constraint

$$0 \le y < 1$$

means something quite different. So we can conclude that the names matter. We call such occurrences of a variable "**free**".

Now consider the following pairs of expressions

- $z = \sum_{i=0}^{N} f(i)$ and $z = \sum_{j=0}^{N} f(j)$
- $z < \int_0^\infty f(u)\, du$ and $z < \int_0^\infty f(v)\, dv$
- $\{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \le 1.0\}$ and $\{(a, b) \in \mathbb{R}^2 \mid a^2 + b^2 \le 1.0\}$

In each case, the two parts of the pair express the same constraint: they are equivalent.

In these cases the variables i, j , u, v, x, y, a, and b are internal to the expression. They don't indicate anything outside of the expression.

Such occurrences of variables are called "**bound**".

An analogous situation comes up in software.

The two subroutines

```
void f() { ++i ; }
```

and

```
void f() { ++j ; }
```

are *not* equivalent. The occurences of `i` and `j` are free.

The two subroutines

```
int g(int i) { return i+1 ; }
```

and

```
int g(int j) { return j+1 ; }
```

are equivalent. The occurences of `i` and `j` are bound.

Single variable substitution

Suppose that $\mathcal{E}$ is an expression and that $\mathcal{V}$ is a variable.

We'll write $\mathcal{E}[\mathcal{V} : \mathcal{F}]$ for the expression obtained by replacing every free occurrence of variable $\mathcal{V}$ in $\mathcal{E}$ with $\mathcal{F}$.

Examples

- $(x/y)[x : y + z]$ is $(y + z)/y$
- $(0 \le i < N \wedge A[i] = 0)[i : i + 1]$ is
$$0 \le i + 1 < N \wedge A[i + 1] = 0$$

Multiple variables

We sometimes need to replace a number of variables at once.

We'll write $\mathcal{E}[\mathcal{V}_0, \mathcal{V}_1, \cdots, \mathcal{V}_{n-1} : \mathcal{F}_0, \mathcal{F}_1, \cdots, \mathcal{F}_{n-1}]$ to mean the simultaneous replacement of n distinct variables by n expressions.

Example

- $(x/y)[x, y : y, x]$ is (y/x)
- whereas $((x/y)[x : y])[y : x]$ is (x/x)

Substitution and bound variables

Making the same substitution in two equivalent expressions must give two equivalent expressions.

Thus we have to be a bit careful about exactly how substitution is defined.

In making substitutions we do not substitute for bound variables. For example in the expression

$$\sum_{i=0}^{N-1} f(i)$$

the variable i is bound, so we don't substitute for it. Thus

$$\left(\sum_{i=0}^{N-1} f(i)\right)[f, i : g, j+1] \text{ is } \sum_{i=0}^{N-1} g(i)$$

Furthermore, it may be necessary to rename bound variables in order to avoid variables free in $\mathcal{F}$ from being "captured". For example

$$\left(\sum_{i=0}^{N-1} (k \times i)\right)[k : i+1] \text{ is } \left(\sum_{j=0}^{N-1} (k \times j)\right)[k : i+1]$$

$$\text{which is } \sum_{j=0}^{N-1} ((i+1) \times j)$$

Note that I had to rename i to j to avoid conflict with the i in the replacement expression.

Notations

Different authors use different notations for substitution.

- In Hoare's axiomatic basis paper, he doesn't use any notation at all.
- In Hehner's practical theory paper, he writes
$$(\text{substitute } \mathcal{F} \text{ for } \mathcal{V} \text{ in } \mathcal{E})$$
- Some writers write $\mathcal{E}(\mathcal{V}/\mathcal{F})$ while others write $\mathcal{E}(\mathcal{F}/\mathcal{V})$
- The most common notation is $\mathcal{E}^{\mathcal{V}}_{\mathcal{F}}$.
- I use $\mathcal{E}[\mathcal{V} : \mathcal{F}]$ because it is hard to mistake for anything else.

One-point laws

The substitution notation lets us express some useful laws called "one-point laws".

Consider $(\mathcal{V} = \mathcal{F}) \Rightarrow \mathcal{E}$ where $\mathcal{V}$ is a variable and $\mathcal{E}$ and $\mathcal{F}$ are expressions. If $\mathcal{V} \neq \mathcal{F}$ then the value of $\mathcal{E}$ doesn't matter, the implication will be $\mathfrak{true}$ regardless of the value of $\mathcal{E}$.

In the case of $\mathcal{V} = \mathcal{F}$, we need only worry about the value of $\mathcal{E}$ under the assumption that $\mathcal{V} = \mathcal{F}$.

The same reasoning applies to an expression $(\mathcal{V} = \mathcal{F}) \wedge \mathcal{E}$.

The one point laws can be expressed as:
$$((\mathcal{V} = \mathcal{F}) \Rightarrow \mathcal{E}) = ((\mathcal{V} = \mathcal{F}) \Rightarrow \mathcal{E}[\mathcal{V} : \mathcal{F}])$$
and
$$((\mathcal{V} = \mathcal{F}) \wedge \mathcal{E}) = ((\mathcal{V} = \mathcal{F}) \wedge \mathcal{E}[\mathcal{V} : \mathcal{F}])$$
Examples:

- $x = y+1 \wedge z = 2x$ is the same as $x = y+1 \wedge z = 2\,(y+1)$

- $x = y + 1 \Rightarrow z = 2x$ is the same as $x = y + 1 \Rightarrow z = 2(y + 1)$

The Quantifiers ∀ and ∃

Suppose that S is a finite set, for example $\{0, 1, 2, 3\}$, and $\mathcal{A}$ is a boolean expression then we write

$$\forall x \in S \cdot \mathcal{A}$$

mean

$$\mathcal{A}[x : 0] \wedge \mathcal{A}[x : 1] \wedge \mathcal{A}[x : 2] \wedge \mathcal{A}[x : 3]$$

and we write

$$\exists x \in S \cdot \mathcal{B}$$

to mean

$$\mathcal{B}[x : 0] \vee \mathcal{B}[x : 1] \vee \mathcal{B}[x : 2] \vee \mathcal{B}[x : 3]$$

just as we would write

$$\sum_{x=0}^{3} \mathcal{E}$$

to mean

$$\mathcal{E}[x : 0] + \mathcal{E}[x : 1] + \mathcal{E}[x : 2] + \mathcal{E}[x : 3]$$

where $\mathcal{E}$ is some numerical expression.

The quantifier $\forall$ is pronounced “for all”.

The quantifier $\exists$ is pronounced “there exists a”.

As long as the set S is finite, $\forall$ and $\exists$ are convenient notations, but not very interesting, as they don’t allow us to do any thing new.

But, if we allow S to be an infinite set, then we have something very interesting.

For example consider the set $\mathbb{N} = \{0, 1, 2, ...\}$ then

$$(\forall x \in \mathbb{N} \cdot \mathcal{A}) = \mathcal{A}[x : 0] \wedge \mathcal{A}[x : 1] \wedge \mathcal{A}[x : 2] \wedge \cdots$$

and

$$(\exists x \in \mathbb{N} \cdot \mathcal{A}) = \mathcal{A}[x : 0] \vee \mathcal{A}[x : 1] \vee \mathcal{A}[x : 2] \vee \cdots$$

In general

- $\forall x \in S \cdot \mathcal{A}$ is $\mathfrak{false}$ if $\mathcal{A}[x : y]$ is $\mathfrak{false}$ for at least one value $y \in S$, otherwise it is $\mathfrak{true}$.
- $\exists x \in S \cdot \mathcal{A}$ is $\mathfrak{true}$ if $\mathcal{A}[x : y]$ is $\mathfrak{true}$ for at least one value $y \in S$, otherwise it is $\mathfrak{false}$.

Some examples:

- All flavours of ice-cream are good:
  $$\forall f \in F \cdot good(iceCream(f))$$
  where F is the set of all flavours of ice-cream, $iceCream$ is a function mapping a flavour to a variety of ice-cream, and $good$ is a “predicate” (boolean function) indicating a variety is good.
- Some people like peanut butter:
  $$\exists p \in P \cdot like(p, peanutButter)$$
  where P is the set of all people and like is a predicate indicating that its first argument likes its second argument.
- The Q output is always equal to the D input of the previous cycle:
  $$\forall t \in \mathbb{N} \cdot Q(t + 1) = D(t)$$
  where Q and D indicate the values of Q and D in a given cycle. We use $\mathbb{N}$ as a time domain, as is

appropriate for discrete time systems.

- The system will be in the initial state within 5 seconds of the reset button being depressed:

$$\forall t \in \mathbb{R}^+ \cdot \mathit{reset}(t) \Rightarrow (\exists u \in \mathbb{R}^+ \cdot t \le u \le t + 5\,\mathrm{s} \wedge \mathit{initial}(u))$$

where reset is a predicate indicating the reset button is depressed and $\mathit{initial}$ indicates that the system is in its initial state. Here I have used $\mathbb{R}^+ = \{x \in \mathbb{R} \mid x \ge 0\}$ to model time, as is appropriate for real-time systems.

Relationship to set theory Recall: The notation $\{x \in S \mid \mathcal{A}\}$ means the subset of S with elements x such that $\mathcal{A}$ is $\mathfrak{true}$.

We can understand $\forall$ and $\exists$ in terms of set notation:

$$(\forall x \in S \cdot \mathcal{A}) = (\{x \in S \mid \mathcal{A}\} = S)$$
$$(\exists x \in S \cdot \mathcal{A}) = (\{x \in S \mid \mathcal{A}\} \ne \emptyset)$$

Therefore

	$\forall x \in S \cdot \mathcal{A}$	$\exists x \in S \cdot \mathcal{A}$
$\emptyset = \{x \in S \mid \mathcal{A}\} = S$	$\mathfrak{true}$	$\mathfrak{false}$
$\emptyset = \{x \in S \mid \mathcal{A}\} \subset S$	$\mathfrak{false}$	$\mathfrak{false}$
$\emptyset \subset \{x \in S \mid \mathcal{A}\} \subset S$	$\mathfrak{false}$	$\mathfrak{true}$
$\emptyset \subset \{x \in S \mid \mathcal{A}\} = S$	$\mathfrak{true}$	$\mathfrak{true}$

Recall: The notation $\{x \in S \cdot \mathcal{E}\}$ means the set of all values of expression $\mathcal{E}[x : y]$ where y is an element of S.

We can understand $\forall$ and $\exists$ in as follows

$$(\forall x \in S \cdot \mathcal{A}) = (\mathfrak{false} \notin \{x \in S \cdot \mathcal{A}\})$$
$$(\exists x \in S \cdot \mathcal{A}) = (\mathfrak{true} \in \{x \in S \cdot \mathcal{A}\})$$

Since $\mathcal{A}$ is boolean, the set $\{x \in S \cdot \mathcal{A}\}$ can have four values (if well defined)

$\{x \in S \cdot \mathcal{A}\}$	$\forall x \in S \cdot \mathcal{A}$	$\exists x \in S \cdot \mathcal{A}$
$\emptyset$	$\mathfrak{true}$	$\mathfrak{false}$
$\{\mathfrak{false}\}$	$\mathfrak{false}$	$\mathfrak{false}$
$\{\mathfrak{true}\}$	$\mathfrak{true}$	$\mathfrak{true}$
$\{\mathfrak{false}, \mathfrak{true}\}$	$\mathfrak{false}$	$\mathfrak{true}$

Laws

There are a number of laws of predicate calculus which are useful to know.

These are some.

Identity laws:

$$
\begin{aligned}
(\forall x \in S \cdot \mathfrak{true}) &= \mathfrak{true} \\
(\exists x \in S \cdot \mathfrak{false}) &= \mathfrak{false} \\
(\forall x \in S \cdot \mathfrak{false}) &= \mathfrak{false}, \text{ provided } S \neq \emptyset \\
(\exists x \in S \cdot \mathfrak{true}) &= \mathfrak{true}, \text{ provided } S \neq \emptyset \\
(\forall x \in \emptyset \cdot \mathcal{A}) &= \mathfrak{true} \\
(\exists x \in \emptyset \cdot \mathcal{A}) &= \mathfrak{false}
\end{aligned}
$$

Change of variable: Provided y does not occur free in $\mathcal{A}$,

$$
\begin{aligned}
(\forall x \in \mathbb{N} \cdot \mathcal{A}) &= (\forall y \in \mathbb{N} \cdot \mathcal{A}[x : y]) \\
(\exists x \in \mathbb{N} \cdot \mathcal{A}) &= (\exists y \in \mathbb{N} \cdot \mathcal{A}[x : y])
\end{aligned}
$$

De Morgan's laws

$$
\begin{aligned}
(\forall x \in S \cdot \mathcal{A}) &= \neg(\exists x \in S \cdot \neg\mathcal{A}) \\
(\exists x \in S \cdot \mathcal{A}) &= \neg(\forall x \in S \cdot \neg\mathcal{A})
\end{aligned}
$$

Example: “It rained every day”, $\forall d \in Day \cdot rain(d)$, is the same as “There was no day when it didn’t rain”, $\neg(\exists d \in Day \cdot \neg rain(d))$.

Domain splitting

$$(\forall x \in S \cup T \cdot \mathcal{A}) = (\forall x \in S \cdot \mathcal{A}) \wedge (\forall x \in T \cdot \mathcal{A})$$
$$(\exists x \in S \cup T \cdot \mathcal{A}) = (\exists x \in S \cdot \mathcal{A}) \vee (\exists x \in T \cdot \mathcal{A})$$

Splitting

$$(\forall x \in S \cdot \mathcal{A} \wedge \mathcal{B}) = (\forall x \in S \cdot \mathcal{A}) \wedge (\forall x \in S \cdot \mathcal{B})$$
$$(\exists x \in S \cdot \mathcal{A} \vee \mathcal{B}) = (\exists x \in S \cdot \mathcal{A}) \vee (\exists x \in S \cdot \mathcal{B})$$

Trading

$$(\forall x \in S \cdot \mathcal{A} \Rightarrow \mathcal{B}) = (\forall x \in \{x \in S \mid \mathcal{A}\} \cdot \mathcal{B})$$
$$(\exists x \in S \cdot \mathcal{A} \wedge \mathcal{B}) = (\exists x \in \{x \in S \mid \mathcal{A}\} \cdot \mathcal{B})$$

One-point laws: Provided x does not appear free in $\mathcal{F}$ and that $\mathcal{F} \in S$,

$$(\forall x \in S \cdot (x = \mathcal{F}) \Rightarrow \mathcal{A}) = \mathcal{A}[x : \mathcal{F}]$$
$$(\exists x \in S \cdot (x = \mathcal{F}) \wedge \mathcal{A}) = \mathcal{A}[x : \mathcal{F}]$$

Commutative: Provided x is not free in T and y is not free in S,

$$(\forall x \in S \cdot \forall y \in T \cdot \mathcal{A}) = (\forall y \in T \cdot \forall x \in S \cdot \mathcal{A})$$
$$(\exists x \in S \cdot \exists y \in T \cdot \mathcal{A}) = (\exists y \in T \cdot \exists x \in S \cdot \mathcal{A})$$

Distributive laws: Provided x is not free in $\mathcal{A}$

$$\mathcal{A} \wedge (\exists x \in S \cdot \mathcal{B}) = (\exists x \in S \cdot \mathcal{A} \wedge \mathcal{B})$$
$$\mathcal{A} \vee (\forall x \in S \cdot \mathcal{B}) = (\forall x \in S \cdot \mathcal{A} \vee \mathcal{B})$$
$$(\mathcal{A} \Rightarrow (\forall x \in S \cdot \mathcal{B})) = (\forall x \in S \cdot \mathcal{A} \Rightarrow \mathcal{B})$$

Distributive laws: Provided $S \neq \emptyset$ and x is not free in $\mathcal{A}$

$$(\mathcal{A} \wedge (\forall x \in S \cdot \mathcal{B})) \;=\; (\forall x \in S \cdot \mathcal{A} \wedge \mathcal{B})$$

$$(\mathcal{A} \vee (\exists x \in S \cdot \mathcal{B})) \;=\; (\exists x \in S \cdot \mathcal{A} \vee \mathcal{B})$$

$$(\mathcal{A} \Rightarrow (\exists x \in S \cdot \mathcal{B})) \;=\; (\exists x \in S \cdot \mathcal{A} \Rightarrow \mathcal{B})$$

Precedence and associativity

As you know, mathematics uses "precedence conventions" to reduce the need for parentheses. For example we all know that

$$w \times x + y \times z$$

means

$$(w \times x) + (y \times z)$$

rather than

$$w \times (x + y) \times z$$

as $\times$ has "higher" precedence than $+$.

Furthermore we know that

$$a - b + c \text{ means } (a - b) + c$$

rather than $a - (b + c)$ as $-$ and $+$ are "left associative".

Some operators are associative meaning it doesn't matter how we add parentheses. E.g.

$$((a \wedge b) \wedge c) = (a \wedge b \wedge c) = (a \wedge (b \wedge c))$$

On the other hand

$$a \leq b < c \text{ means } (a \leq b) \wedge (b < c)$$

and we say that $\leq$, $<$, $=$, etc are "chaining"

The following table shows many of the operators used in the course in order of precedence (highest to lowest)

$x(y)$	LA
$-x \quad \neg x$	
$x \times y \quad x/y$	LA
$x + y \quad x - y$	LA
$\cap$	A
$\cup$	A
$x = y \quad x \leq y \quad x < y \quad x \in y$	Ch
$x \wedge y$	A
$x \vee y$	A
$x \Rightarrow y$ NA $x \Leftrightarrow y \quad x \not\Leftrightarrow y$	A
$x : y$	NA
$\mathbf{if}\ B\ \mathbf{then}\ x\ \mathbf{else}\ y \quad \mathbf{while}\ B\ \mathbf{do}\ x$	
$;$	A
$\sqsubseteq$	Ch
$\forall v \in S \cdot x \quad \exists v \in S \cdot x$	

where

LA	Left associative
RA	Right associative
A	Associative
NA	Nonassociative
Ch	Chaining

The low precedence of the quantifiers means that the scope of a quantified variable extends to the right to the end of the formula, unless there is explicit parenthesization or punctuation to stop it. I recommend putting quantifications in parentheses except when there is no possible confusion.

That $\wedge$ has higher precedence than $\vee$ is conventional, but I recommend using extra parentheses, e.g. to write

$$p \wedge q \vee r \text{ as } (p \wedge q) \vee r$$