Contents

1 Sets and Types - Basic Notation

Z Specification use mathematics to express the properties of systems. Z is a mathematical language based on standard set notation.

1.1 Sets in Z

A set is a collection of values called elements or members. In Z, all possible values of a set are considered to have something in common and are said to have the same type. Any set is considered to be a subset of its type.

1.2 Variables and Types

It is useful at this stage to distinguish between a variable used in Z which is a mathematical variable and a variable as used in the procedural programming sense. In the latter case, a variable is the name of a location in some store; but in the formal world, a variable is just a symbol that denotes a unique ***element*** i.e. a value that has a type. A Z variable is variable, not because it is updatable, but because the value it denotes is often non-specific, as in a schema, and covers a range of possibilities.

Z adopts a simple approach to typing. A type denotes a non-empty set of values which is treated as ***maximal***(maximal means that suppose that S is a set of type τ, all values of type τ are contained in set S). Moreover, S contains nothing but τ values. This notion of type has two important aspects:

- a type denotes a set that is not contained in any wider set (i.e. subtypes do not exist in Z)
- the type of any Z expression can be determined by an algorithm, irrespective of the value that the expression denotes (type checking is thus possible)

Every Z specification needs one or more basic types with which to describe objects whose internal makeup is of no relevance to the specification. Values of basic types are regarded as atomic. Types can be either

- Built-in types
- Free-types
- Basic types

1.2.1 Built-in types

The Z notation has only one built in type, integer:

$$\mathbb{Z} = \ldots -4, -3, -2, -1, 0, 1, 2, 3, 4, \ldots$$

It is an infinite set.

Natural numbers A widely used set is the set of natural numbers or non-negative integers. It is not a type in Z but a subset of its underlying type, which is integer. It is written

$$\mathbb{N}_1 = \ldots - 4, -3, -2, -1, 0, 1, 2, 3, 4, \ldots$$

In some contexts it is useful to exclude zero from this set. This set is written in Z as follows:

$$\mathbb{N}_1 = \ldots 0, 1, 2, 3, 4, \ldots$$

Operations on integers: The following operators are defined for the type integer and its subsets:

+	addition
-	subtraction
∗	multiplication
div	integer division
mod	modulus

Figure 1: Operations on Integers

integer-ops

Others The Z notation does not include the set of real numbers as a built-in type. If you need real numbers, you could define the type ***REAL*** and regard the integer set as a subset of the type ***REAL***. The set of characters is not a built-in type.

1.2.2 Basic Types

Basic types are also called given sets. The basic types of a specification are declared without further examining what their elements are. For example in a specification about a library, you might need to refer to books and the people who are reading them. We could have two sets, written

$$[BOOK, PERSON]$$

We are not further interested in what information we need about each book (we're looking at the simple case where there is not more than one copy of each book). The type ***PERSON*** frees us from any worry about what information we need about each person. Note that if it was important for us to have a certain piece of information about a person, then we would have to address that. In general, basic types should be chosen to be as widely encompassing as possible. It should be assumed that the elements of the type are uniquely identifiable. [PERSON] is the set of all, uniquely identifiable persons. The problem of how we distinguish people of the same name need not be addressed here.

1.2.3 Free Types

It can be very useful to define a type in terms of the possible values it can have. We can do this with a free type and the general format is:

$$FreeType ::= element1 \mid element2 \mid element3 \mid .... \mid elementn$$

Examples

$$RESPONSE ::= yes \mid no$$
$$REPORT ::= InsertSucess \mid ElementAlreadyThere$$

RESPONSE is a set containing the two values yes, no. We will cover free types more fully later.

1.2.4 Declarations

All names designating values must be declared, i.e. their type must be stated. For example, to introduce a named value smallestCountry to be of the basic type $COUNTRY$, must write

$$[COUNTRY]$$
$$smallestCountry : COUNTRY$$

For the examples following, we will use countries by their abbreviation

Belguim	BE	Greece	EL	Lituania	LT	Portugal	PT
Bulgaria	BG	Spain	ES	Luxembourg	LU	Romania	RO
Czech Republic	CZ	France	FR	Hungary	HU	Slovenia	SI
Denmark	DK	Crotia	HR	Malta	MT	Slovakia	SK
Germany	DE	Italy	IT	Netherlands	NL	Finland	FI
Estonia	EE	Cyprus	CY	Austria	AT	Sweden	SE
Ireland	IE	Latvia	LV	Poland	PL	United Kingdom	UK

Figure 2: List of countries in the EU with their abbreviation
eu-members

To introduce a variable called homecounty to refer to one county, we could write homecounty : COUNTY The following Truth Valued Statement[1] is valid smallestCountry = MT

Sets of Values When a name is to be given to a set of values, the name is declared as a powerset of the type of the elements:

$$smallCountries : \mathbb{P}\, COUNTRY$$

This can be read “smallCountries is a subset of the set of COUNTRY’s ”

[1] A Truth Valued Statement is a statement whose value is either True or False.

Set constants A set of values is written by enumerating the set's values within braces

$$smallCountries = \{MT, LU, CY, SI, NL, EE, SK\}$$

Order is irrelevant i.e.

$$\{MT, LU, CY\} = \{LU, MT, CY\} = \{CY, MT, LU\}$$

Repeating a term does not matter;

$$\{MT, LU, CY\} = \{MT, MT, MT, LU, CY, MT\}$$

The empty set It is possible to have a set with no values. This is called the empty set. It is written

$\varnothing$ or
$\{\ \}$

A Singleton Set A set which contains one element is called a singleton set.

$$\{DE\}$$

Note that *DE* does not have the same type as {DE}

1.2.5 Membership

The membership operator is written

$$\in$$

The expression involving the operator is true if the value is an element of the set, otherwise it is false.

$DE \in \{DE, IE, MT\}$ this statement is true

We use Venn diagrams to illustrate the set operations in these notes

1.2.6 Operators

Equivalence Two values can be tested to check if they have the same type, by using the equals sign as in

$$x = y$$

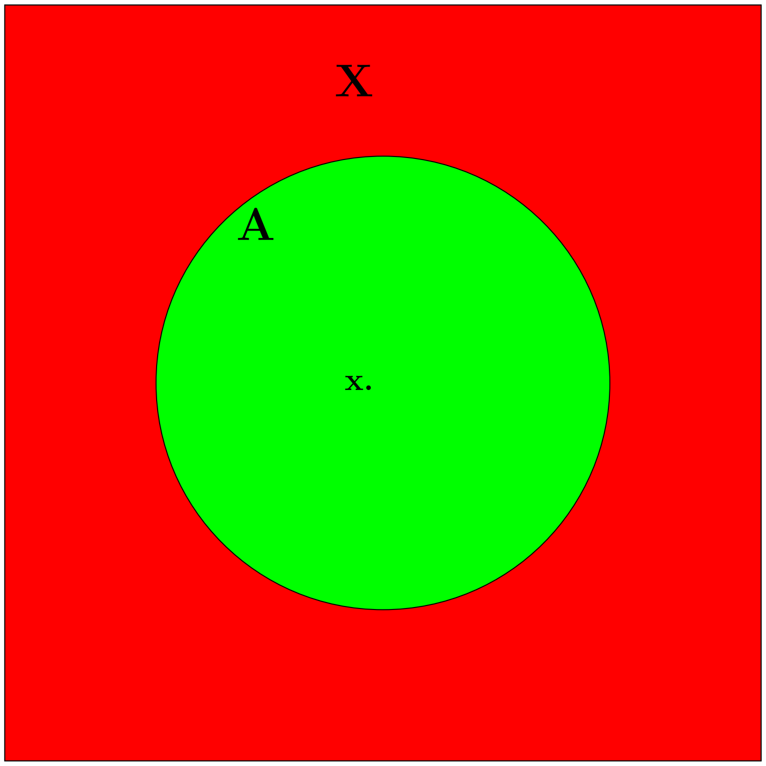
Two sets are equal if they contain exactly the same elements. Note - order does not matter.

Non-equivalence Similarly, two values of the same type can be tested to see if they are not the same by using the not equals sign, as in

$$x \neq y$$

Two sets are not equal if they do not contain exactly the same elements.

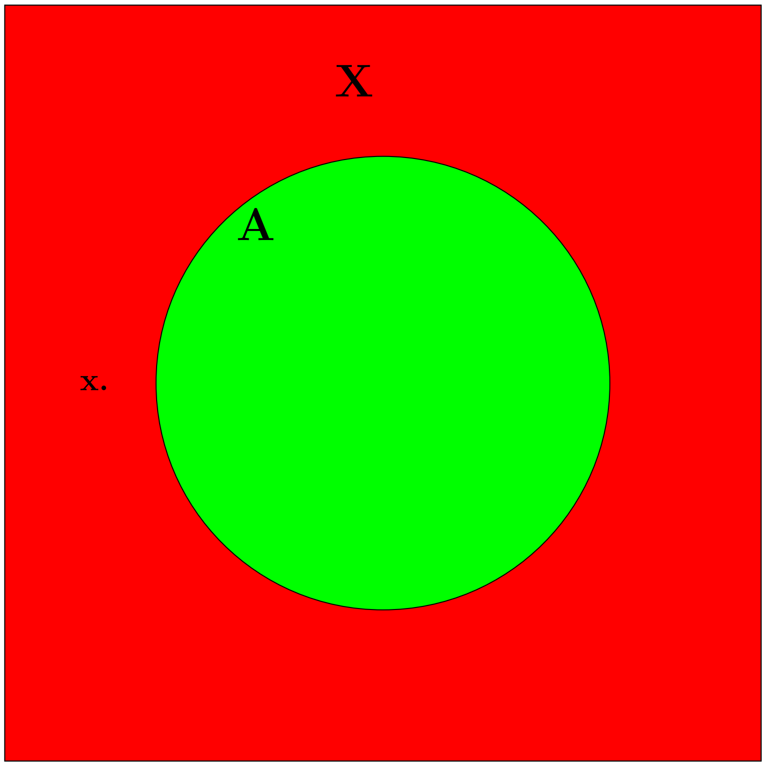
1.2.7 Set Membership



$$[X]$$
$$A : \mathbb{P}\,X$$
$$x : X$$
$$x \in A$$

Figure 3: Set membership

1.2.8 Set non-Membership



$$[X]$$
$$A : \mathbb{P}\, X$$
$$x : X$$
$$x \notin A$$

Figure 4: Set non-membership

Note

$$DE \in \{IE, BE, EE\}$$

is false

$$DE \notin \{IE, BE, EE\}$$

is true
but

$$1 \notin \{IE, BE, EE\}$$

is illegal because 1 in not of type COUNTRY

1.2.9 Size, Cardinality

The number of values in a set is called its size or cardinality and is signified by the hash sign, #, acting as a function.

$$\#\{DE, IE, EE\} = 3 \, \#\{DE\} = 1 \, \# \, IE \text{is illegal as IE is not a set} \# \, \varnothing = 0$$

Note: A set must be finite to legally apply the hash function to it.

1.2.10 Powersets

The powerset of a set S, is written

$$\mathbb{P}\, S$$

and is the set of all its subsets.

$$\begin{array}{ll} \mathbb{P}\{DE, FR, IE\} = & \\ \{ \ \ \varnothing, & \text{the empty set} \\ \quad \{DE\}, \{FR\}, \{IE\}, & \text{all the singletons} \\ \quad \{DE, FR\}, \{FR, IE\}, \{DE, IE\}, & \text{all the pairs} \\ \quad \{DE, FR, IE\} & \text{all three elements} \\ \} & \end{array}$$

This means that a set of values of this powerset type is a subset of the underlying set type.

1.2.11 Finite sets

The cardinality of a set is only defined when the set is finite. This means that if we need to insist that a powerset is finite, we use the Finite set symbol

$$\mathbb{F}$$

e.g.

$$someSetOfNumbers : \mathbb{F}\mathbb{Z}$$

2 Set Operations and Relationships

2.1 Set Relationships

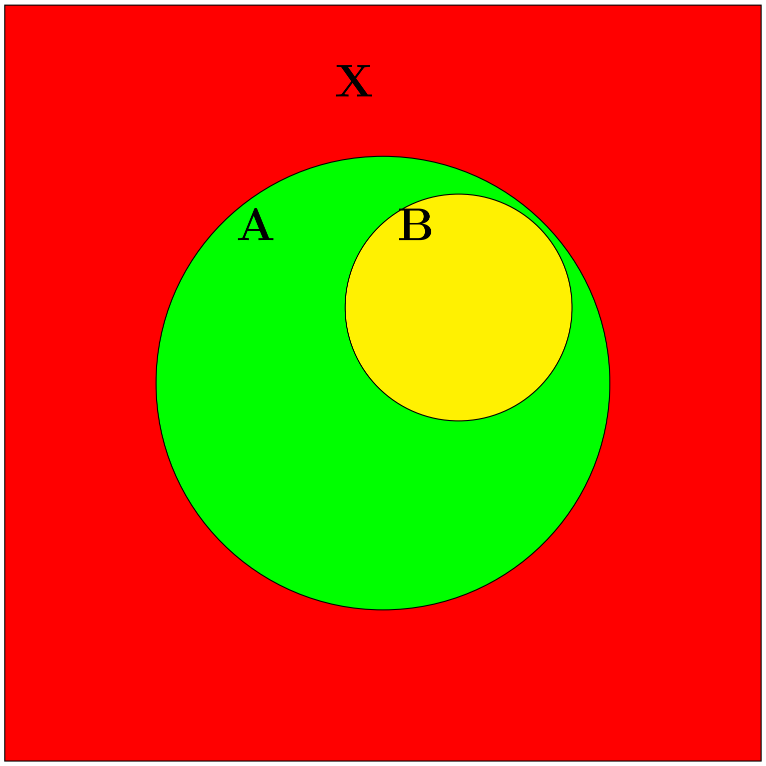
We sometimes need to state relationships between sets. These take the form of Truth Valued Statements.

2.1.1 Set Inclusion

set-incl The operator

$$\subseteq$$

is called ***is included in*** or ***is a subset of***. It tests whether the first set is a subset of the second.



$$[X]$$
$$A : \mathbb{P}\, X$$
$$B : \mathbb{P}\, X$$
$$B \subseteq A$$

Figure 5: Set inclusion

So,

$$\{1,2,3\} \subseteq \{1,2,3,4,5\}$$
$$\{1,2,3\} \subseteq \{1,2,3\}$$
$$\varnothing \subseteq \{1,2,3\}$$

In particular, note that the empty set is a subset of all sets.

2.1.2 Strict Inclusion

set-strict-inclusion The operator

$$\subset$$

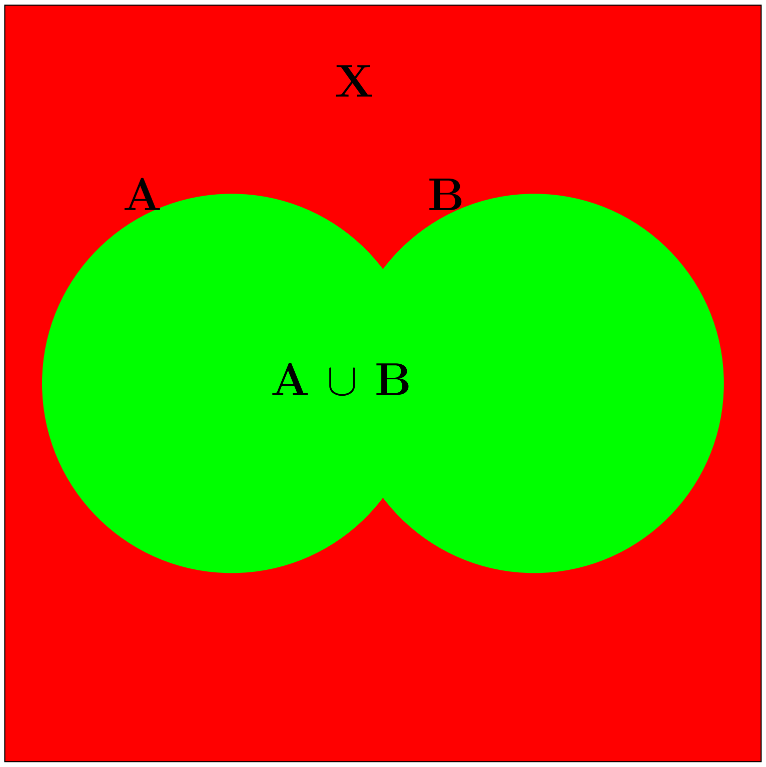
denotes strict inclusion or ***is a proper subset of*** i.e. the first set may not be equal to the second set.

$\{1, 2, 3\} \subset \{1, 2, 3, 4, 5\}$this is true
$\{1, 2, 3\} \subset \{1, 2, 3\}$this is false
$\{1, 2, 3\} \subseteq \{1, 2, 3\}$this is true
$\varnothing \subset \{1, 2, 3\}$this is true

2.2 Set Operations

2.2.1 Set union

The union of two sets is the set containing all the elements that are in either the first set or second set or both.



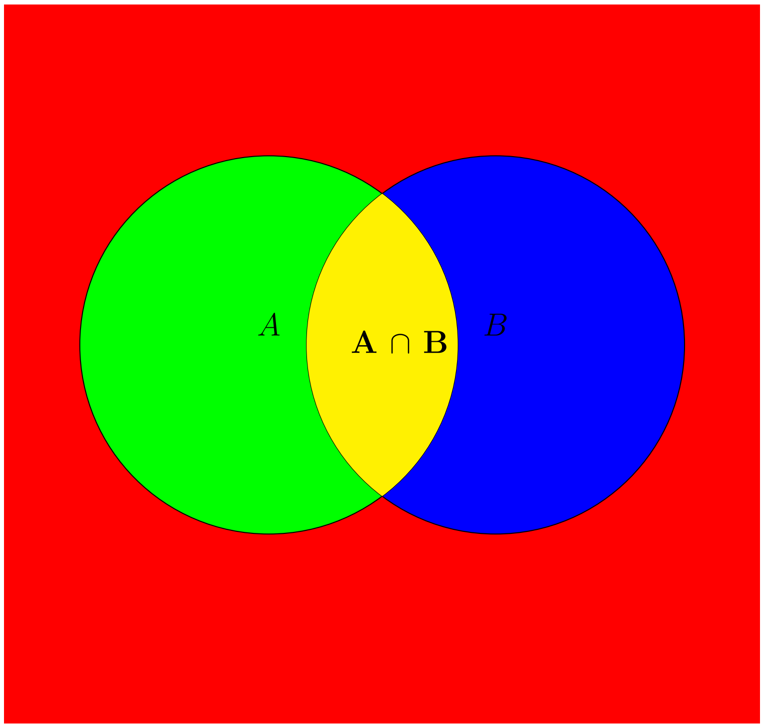
$[X]$
$A : \mathbb{P}\, X$
$B : \mathbb{P}\, X$
$A \cup B$

Figure 6: Set union

These are operations that we can perform on sets (of the same types).

2.2.2 Set intersection

The union of two sets is the set containing all the elements that are in either the first set or second set or both.

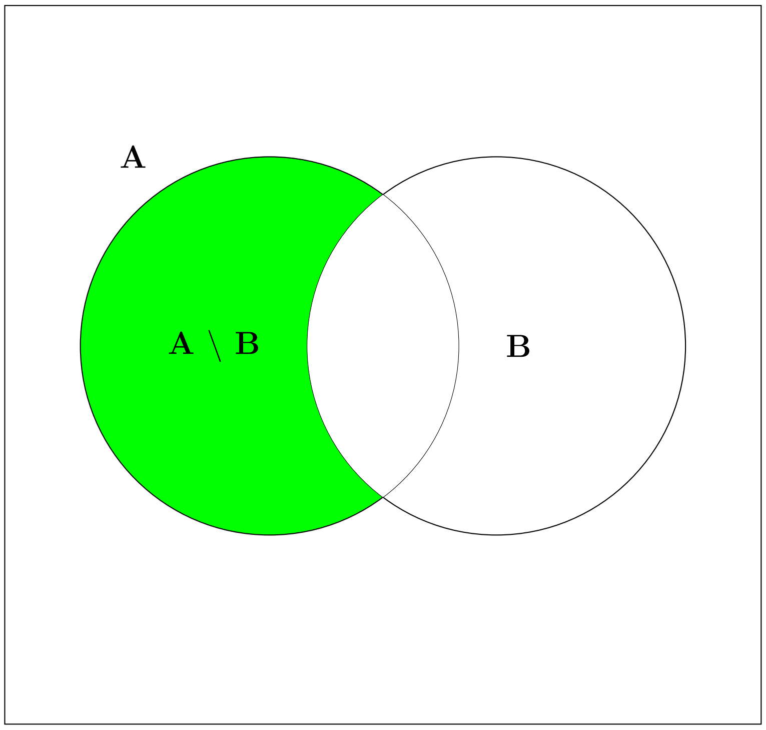


$$[X]$$
$$A : \mathbb{P}\, X$$
$$B : \mathbb{P}\, X$$
$$A \cap B$$

Figure 7: Set Intersection

2.2.3 Set Difference

The set difference of two sets is the set containing all the elements of the first set that are not in the second set.



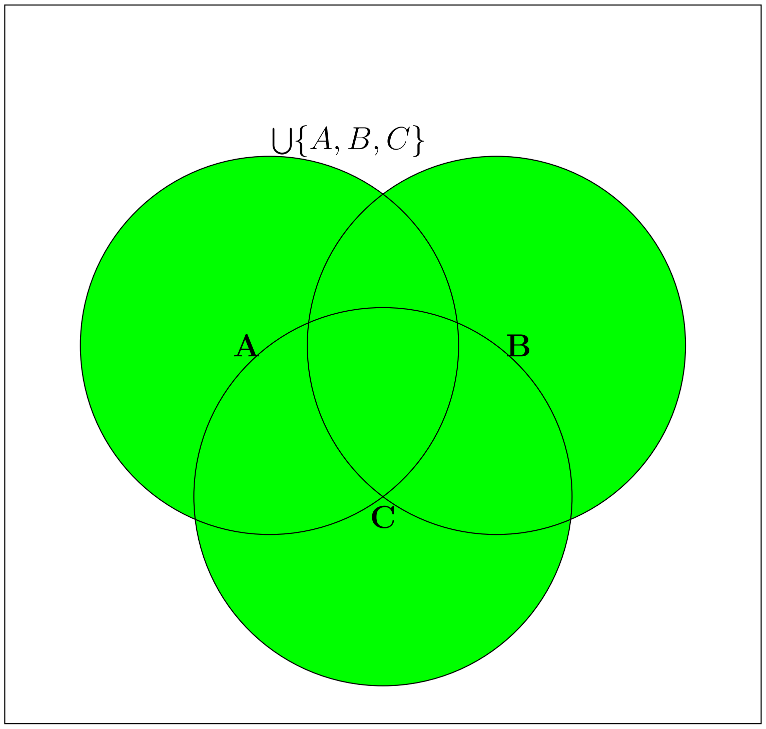
$$[X]$$
$$A : \mathbb{P}\, X$$
$$B : \mathbb{P}\, X$$
$$A \setminus B$$

Figure 8: Set Difference

2.2.4 Distributed Union

The distributed intersection of a set is the set of elements which are in all of the component sets.

$$\bigcup \{\{DE, IE\}, \{IE, EE\}, \{BE, EE, GB, IE\}\} = \{DE, IE, EE, BE, GB\}$$



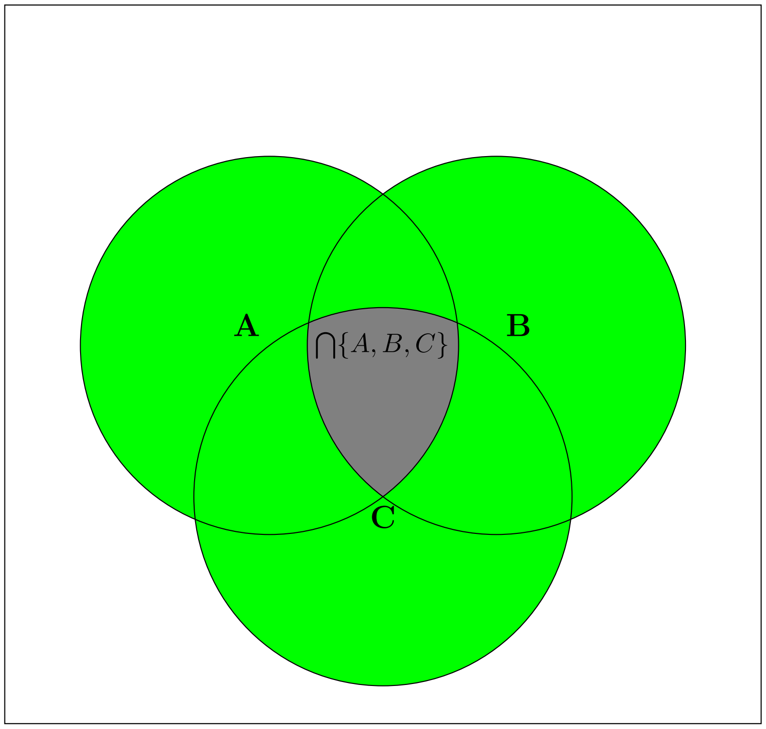
$$[X]$$
$$A : \mathbb{P}\,X$$
$$B : \mathbb{P}\,X$$
$$C : \mathbb{P}\,X$$
$$\bigcup\{A, B, C\}$$

Figure 9: Distributed Union

2.2.5 Distributed Intersection

The operations described so far have been applied to two sets. Sometimes it is useful to be able to refer to the union of several sets or more precisely to a set of sets. This can be done with the distributed union operator. The operator is written as an large version of the union operator and is applied to a set of sets and results in a set.

$$\bigcap\{\{DE, IE\}, \{IE, EE\}, \{BE, EE, GB, IE\}\} = \{IE\}$$

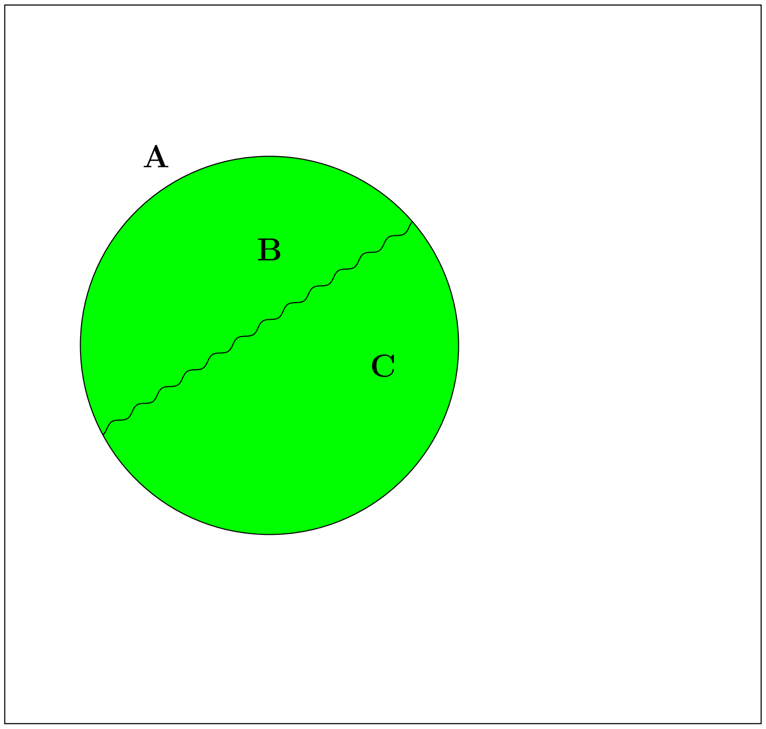


$$[X]$$
$$A : \mathbb{P}\,X$$
$$B : \mathbb{P}\,X$$
$$C : \mathbb{P}\,X$$
$$\cap\{A, B, C\}$$

Figure 10: Distributed Intersection

2.2.6 Set partition

The union of two sets is the set containing all the elements that are in either the first set or second set or both.



$[X]$
$A : \mathbb{P}\, X$
$B : \mathbb{P}\, X$
$C : \mathbb{P}\, X$
$< B, C > \underline{partition}\ A$

Figure 11: Set Partition

Note that

$A = B \cup C$
$B \cap C = \varnothing$

In this case, B and C are said to be disjoint i.e. they have no elements in common.

$\underline{disjoint} < B, C >$

Also, the distributed union of B and C makes up another set, A. The sets B and C are said to partition the set A because

$A = B \cup C\ and$
$B \cap C = \varnothing$

More than two sets can partition a set. This can be a useful way of describing interset relationships. More formally, this is known as partition

$$< B, C > \underline{partition}\ A$$

2.2.7 Ranges of numbers

The range of values m..n is the set of all integers between and including m and n e.g. 2..5 = {2,3,4,5}. The notation m..n therefore denotes a valid set and can be used as such.

3 Set construction

Up to now, we have needed to enumerate the elements in a set. Another way of constructing a set is using set comprehension i.e. don't explicitly enumerate the elements, instead define the set as the set of all elements that obey a given set of rules. Set comprehension is a more powerful and elegant way of constructing sets.

3.1 Set Comprehension Notation

The set of values

$$\{1, 2, 3, 4, 5, 6, 7, 8\}$$

could also be written using set comprension

$$\{x : \mathbb{N} \mid 1 \leq x \leq 8 \bullet x\}$$

The structure is below:

$$Set - Exp = \{declaration \mid constraint(or\ TruthValuedStatement) \bullet expression\ or\ term\}$$

Example

$$Squares = \{x : \mathbb{N} \mid 1 \leq x \leq 20 \bullet x^2\}$$

is the set of squares of every integer from 1 to 20.

The members of the set

$$\{Declaration \mid constraint \bullet expression\}$$

are the values taken by the expression when the variables introduced by the declaration take all possible values which make the constraint true.

The above set Squares might be read as: "select all those integers which lie between 1 and 20 inclusive, and form the set of their squares". In future examples, we will use more advanced constraints. Any valid Truth Valued statement is valid here.

3.2 Omissions

If the constraint is omitted it is taken to be True i.e. it does not constrain the variables in the expression.

$$T = \{x : \mathbb{Z} \bullet x^2\}$$

is the set of squares of all integers.

3.3 Tuples and Product Types

We will meet tuples later in a discussion on relations. A tuple is an ordered collection of two or more objects. Examples are common in records : (Name, Address) is an ordered couple; and in such mathematical constructs as vectors and co-ordinate points . If to the couple we add a third component PhoneNo we get an ordered triple (Name, Address, PhoneNo), and so on.

Characteristic Tuple
The characteristic tuple is the tuple formed from the variables in the declaration part of a set comprehension, in the order of declaration. In

$$K = \{a : \mathbb{N}, b : \mathbb{N} \mid a \leq 10 \wedge b \leq 10\}$$

the characteristic tuple is (a,b).

The default Expression in a set comprehension is the characteristic tuple of the set:

$$K = \{a : \mathbb{N}, b : \mathbb{N} \mid a \leq 10 \wedge b \leq 10 \ \bullet (a, b)\}$$