6SENG001W Reasoning about Programs

Lecture 2

Sets, Types & Constants in B

Overview of Lecture 2: B Sets & Types

The aim of this lecture is to:

- Outline the *mathematical concepts* used in the B-Method.
- Introduce & review basic *Set Theory*:
  - *Number sets* – types, number operators & relations.
  - *General sets* – types, set operators & relations.
- *Abstract Machine Notation* (AMN) for *numbers* & *sets*.
- Using *Sets*, *Constants* & *Variables* in a *B machine*:
  - *types of sets* that can be used;
  - declaring & using *sets*, *constants* & *variables* in a B machine;
  - B machine *"clauses"* – `SETS`, `CONSTANTS PROPERTIES`, `VARIABLES`, `INVARIANT` & `INITIALISATION`.
- Example Machine using: *sets*, *constants* & *variables*.

PART I

Mathematical Concepts used in the B-Method

B-Method Describing Systems: Precisely, Unambiguously & Concisely

Recall that one of the main aims of the B-Method is to enable developers to *"describe a system's requirements"*:

- in a *structured way*,
- *succinctly*,
- *precisely*,
- *unambiguously*,
- ignore system details by using *abstraction*,
- check for *consistency* &
- *prove* essential & desirable properties about the system.
- *design* & *construct* code that satisfies the system's requirements.

Conclusion

The **ONLY** way all of these aims can be achieved is by the use of a specification language that includes the features that allows & supports them.

Mathematical Concepts used in the B-Method

For the reasons just outlined, the B-Method's specification language *AMN* is a *"wide-spectrum"* specification language.

This means that the *AMN* language includes:

- *mathematical* concepts & notation,
- *programming language* constructs, &
- *structuring* mechanisms.

The *mathematical concepts* included in AMN are:

- set theory,
- predicate logic,
- relations,
- functions &
- sequences.

Remark: we shall cover AMN's *programming constructs* & *structuring mechanisms* in subsequent lectures.

PART II

Set Theory

What is a Set?

Definition: Set

A set is a collection of values called elements or members & all the members of a set must be of the same type.

Where in the above definition the *"same type"* means the *"same kind of thing"*.

For example, you can define *sets* of basically anything by listing the *elements* between set brackets – "{" & "}" separated by commas – "," as follows:

- *"numbers"* — $\{ 0, 3, 45, 999, \ldots \}$
- *"people"* — $\{ Bill, Sue, Joe, Mary, Ian, Jim, \ldots \}$
- *"students"* — $\{ JimJones, MarySmith, \ldots \}$
- *"BEng Level 6 modules"* — $\{ 6SENG001W, 6SENG002W, \ldots \}$
- *"colours"* — $\{ \mathbf{red}, \mathbf{blue}, \mathbf{green}, \mathbf{purple}, \mathbf{orange}, \mathbf{black}, \ldots \}$
- *"days of the week"* — $\{ Monday, Tuesday, Sunday, \ldots \}$

No Duplicate Values in a Set

One of the most important properties of a set is that it **only** records:

*if an individual element is **in** the set or is **not in** the set*,

it **does not** record:

how many times an element occurs in a set.

So sets do not include duplicate values.

So when we write out a set we do **not include duplicate values**, in other words the following sets are all equal:

$$
\begin{aligned}
&\{\, 1,\ 2,\ 2,\ 3,\ 3,\ 3,\ 4,\ 4,\ 4,\ 4,\ 5,\ 5,\ 5,\ 5,\ 5 \,\} \\
=\ &\{\, 1,\ 2,\ 3,\ 3,\ 4,\ 4,\ 4,\ 5,\ 5,\ 5,\ 5 \,\} \\
=\ &\{\, 1,\ 2,\ 3,\ 4,\ 4,\ 5,\ 5,\ 5 \,\} \\
=\ &\{\, 1,\ 2,\ 3,\ 4,\ 5,\ 5 \,\} \\
=\ &\{\, 1,\ 2,\ 3,\ 4,\ 5 \,\}
\end{aligned}
$$

In practice we only use the last version of these sets.

Order of Elements Listed in a Set Does Not Matter

Another important property of a set is that the *order* in which the elements in a set are listed does not matter.

So when we write out a set it does not matter what order we write them in.

Hence the following sets are all equal:

$$\begin{aligned} & \{ 5,\ 4,\ 3,\ 2,\ 1 \} \\ = & \{ 1,\ 3,\ 5,\ 2,\ 4 \} \\ = & \{ 3,\ 5,\ 1,\ 4,\ 2 \} \\ = & \{ 1,\ 2,\ 3,\ 4,\ 5 \} \end{aligned}$$

By convention we usually only use the last version of these sets.

What is **NOT** a Set?

The *elements* in a B set *"must all be the same kind of thing"*.

Therefore, we would **not** consider a set to be *"well defined"* if its elements were different types of things.

To ensure this holds, B uses the notion of "*types*", i.e. where a *type* is a *set*.

All the *elements* in a set are of the *same type* & every *set* is a *subset or equal to* its type.

So the following "sets" are **not** *"well defined"*, i.e. not valid sets:

- *"numbers & colours"* — $\{ 3, \mathit{red}, 45, \mathit{blue}, 999, \mathit{orange}, \ldots \}$
- *"people & days of the week* — $\{ Monday, Sue, Joe, Sunday, \ldots \}$
- *"students & BEng Level 6 modules"* — $\{ JimJones, 6SENG001W, \ldots \}$
- *"numbers, colours & animals"* — $\{ 42, dog, \mathit{green}, 99, penguin, \ldots \}$

Because they contain *elements of different types*.

B's Built in Sets — Number Sets

B has three *"built in"* sets of numbers, that can be used in any B machine.

These sets are: *Integers* & two sets of *Natural numbers*.

Integers: are the negative & positive whole numbers & are defined as:

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

Naturals: are the positive whole numbers & are considered to be a subset of the integers.

B supports two basic sets of natural numbers, one starting from 0 & one starting from 1 that are defined as follows:

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

Arithmetic Operators

The standard *arithmetic operations* are available.

Symbol	Description
$x + y$	x plus y
$x - y$	x minus y
$x \times y$	x multiply y
$x \div y$	x divided by y (integer division)
$x \bmod y$	*remainder* after x divided by y (integer division)
$x \mathbin{\widehat{}} y$	x to the *power* y, x^y
$\mathrm{succ}(x)$	*successor* of x (increment x)
$\mathrm{pred}(x)$	*predecessor* of x (decrement x)

Examples

$$47 \div 5 = 9 \qquad 47 \bmod 5 = 2$$

$$3 \mathbin{\widehat{}} 3 = 3 * 3 * 3 \; = \; 27$$

$$\mathrm{pred}(12) = 11 \qquad \mathrm{pred}(0) = -1$$

$$\mathrm{succ}(34) = 35 \qquad \mathrm{succ}(0) = 1$$

Number Relations

The following standard relationships between numbers are available.

Symbol	Description
$x = y$	x equal to y
$x \neq y$	x not equal to y
$x < y$	x less than y
$x \leq y$	x less than or equal to y
$x > y$	x greater than y
$x \geq y$	x greater than or equal to y

Examples

$11 = 11$

$78 < 99$

$47 > 25$

$23 \neq 42$

$42 \leq 42$

$99 \geq 17$

Miscellaneous Number Operations

The following number operators are also available.

Symbol	Description
$\min(A)$	$\min(A)$ evaluates to the *minimum* number in set A
$\max(A)$	$\max(A)$ evaluates to the *maximum* number in set A
$x \mathinner{..} y$	*range of numbers* from x up to y inclusive, i.e. set of numbers

Examples

Given the set of numbers:

$$A = \{ 2, 5, 9, 23, 45, 89, 101 \}$$

$$\min(A) = 2$$
$$\max(A) = 101$$
$$3 \mathinner{..} 7 = \{ 3, 4, 5, 6, 7 \}$$
$$-2 \mathinner{..} 2 = \{ -2, 1-, 0, 1, 2 \}$$
$$\min(4 \mathinner{..} 6) = 4$$
$$\max(39 \mathinner{..} 99) = 99$$

General Set Definitions

The following general standard set definition operators are available.

Symbol	Description
$x \in A$	*membership* – x is an *element* (or *member*) of A
$x \notin A$	*non-membership* – x is *not* an element (member) of A
$\varnothing, \{\ \}$	*Empty set*
$\{\ 1\ \}$	*Singleton* set (1 element)
$\{\ 1,\ 2,\ 3\ \}$	Set of *elements*: 1, 2, 3
$\mathrm{card}(A)$	*cardinality* – *number of elements* in set A

Examples

$$A = \{\ a,\ b,\ c,\ d,\ e,\ f,\ g,\ h\ \} \qquad B = \{\ a,\ e,\ i,\ o,\ u\ \}$$
$$C = \{\ x,\ y,\ z\ \} \qquad D = \{\ f,\ o,\ r,\ m,\ a,\ l,\ e,\ t,\ h,\ d,\ s\ \}$$

We have:

$$1 \in \{\ 1\ \} \qquad 3 \notin \{\ 1,\ 2,\ 4,\ 9\ \} \qquad 5 \notin \{\ \}$$
$$a \in A \qquad x \notin B \qquad f \in D$$
$$\mathrm{card}(A) = 7 \qquad \mathrm{card}(B) = 5 \qquad \mathrm{card}(C) = 3 \qquad \mathrm{card}(D) = 11$$

Set Operators

New sets can be constructed by using the following set operators to combine existing sets.

Symbol	Description
$A \cup B$	set *union* of A & B
$A \cap B$	set *intersection* of A & B
$A - B$	Set *subtraction* (difference) of A & B

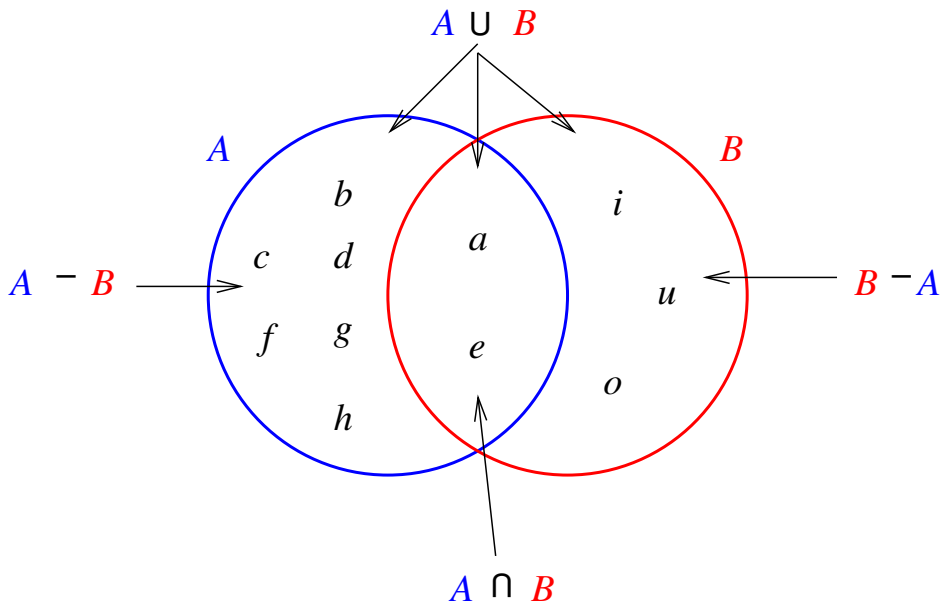
Examples

Using the previous sets A, B, C, D we have:

$$A \cup B = \{\ a, b, c, d, e, f, g, h, i, o, u\ \}$$
$$A \cap B = \{\ a, e\ \}$$
$$D - B = \{\ f, r, m, l, t, h, d, s\ \}$$
$$\{\ 2,\ 4\ \} \cup \{\ 1,\ 3,\ 5\ \} = \{\ 1,\ 2,\ 3,\ 4\ ,5\ \} = 1\,..\,5$$
$$\{\ 2,\ 4\ \} \cap \{\ 1,\ 2,\ 3\ \} = \{\ 2\ \}$$
$$\{\ 1,\ 2,\ 3,\ 4\ ,5\ \} - \{\ 4\ ,5,\ 99\ \} = \{\ 1,\ 2,\ 3\ \} = 1\,..\,3$$

Set Operators Venn Diagram

Using $A = \{ a, b, c, d, e, f, g, h \}$ & $B = \{ a, e, i, o, u \}$.



Examples

$A \cup B = \{ a, b, c, d, e, f, g, h, i, o, u \}$

$A \cap B = \{ a, e \}$

$A - B = \{ b, c, d, f, g, h \}$

$B - A = \{ i, o, u \}$

Set Relations

The following *relations* allow sets to be compared, provided they are the same type.

Symbol	Description
$A = B$	*equality* – A is *equal to* B
$A \neq B$	*non-equality* – A is *not equal to* B
$A \subseteq B$	*subset* – A is a *subset of or equal to* B
$A \subset B$	*strict subset* – A is a *subset of, but not equal to* B

Examples: Where X is any set we have:

$$\{ 1, 2, 3 \} = 1 \mathinner{..} 3 = \{ 3, 2, 1 \}$$
$$\{ 77, 88, 99 \} \neq \{ 33, 66, 99 \}$$
$$\{ 77, 88, 99 \} \subseteq \{ 77, 88, 99 \}$$
$$\{ 3, 4, 5 \} \subseteq \{ 1, 2, 3, 4, 5 \}$$
$$\{ 2, 4 \} \subset \{ 2, 4, 6 \}$$
$$X \subseteq X$$
$$\{ \} \subseteq X$$

Definition: the Power Set of a Set

One of the most common things we need to do in B is use *variables* that contain sets of a particular type.

To be able to do this we need to be able to form the *"power set"* of a set.

Definition: Power Set – $\mathbb{P}(X)$

The "power set" of a set X denoted by:

$$\mathbb{P}(X)$$

is the set containing all of the subsets of X, i.e. a set of sets.

Note: the *power set* of a set X – $\mathbb{P}(X)$ **always includes**:

- the *empty set* "$\{\ \}$", and
- the *set X itself*.

because both are *subsets of any set*.

The cardinality (size) of a power set formed from an arbitrary set X, i.e. the number of sets $\mathbb{P}(X)$ contains, are given by the following formula:

$$\mathrm{card}(\ \mathbb{P}(X)\ ) \ = \ 2^{\mathrm{card}(X)}$$

Examples: Power Sets

The following standard operators are available:

Symbol	Description
$\mathbb{P}(A)$	*Power set* of A

Example

$$
\begin{aligned}
A &= \{ x, y, z \} \\
\mathrm{card}(A) &= 3 \\
\mathbb{P}(A) &= \{ \{\}, \{x\}, \{y\}, \{z\}, \{x, y\}, \{x, z\}, \{y, z\}, \{x, y, z\} \} \\
\mathrm{card}(\ \mathbb{P}(A)\ ) &= 2^{\mathrm{card}(A)} = 2^3 = 8
\end{aligned}
$$

All of the sets that are included in the power set of A, i.e. $\mathbb{P}(A)$, are there because they are all a *subset or equal to* A, in particular all of the following are true:

$$
\begin{aligned}
\{\ \} &\subseteq A \\
\{ x \} &\subseteq A \qquad \{ y \} \subseteq A \qquad \{ z \} \subseteq A \\
\{ x, y \} &\subseteq A \qquad \{ x, z \} \subseteq A \qquad \{ y, z \} \subseteq A \\
\{ x, y, z \} &\subseteq A
\end{aligned}
$$

Set Comprehension

A set can be constructed by giving a *condition* that must hold for all the members of the set.

(Technically, this condition is known as a *"predicate"*.)

A *set comprehension* has the general form:

$$\{ variable \mid variable \in TYPESET \ \wedge \ variableCondition \}$$

Where:

- The $variable$ is *declared*.
  This $variable$ represents an arbitrary *element* of the set being defined.
- $variable \in TYPESET$ defines the *type* of the $variable$, i.e. the set the values of $variable$ are in.
- The $variableCondition$ is the *condition* that elements of $TYPESET$ *must satisfy* to be included in the set being defined.

Examples: Set Comprehension

For example, the set of numbers 0 to 10:

$$\{ x \mid x \in \mathbb{N} \land x \leq 10 \} = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \}$$

Where:

- "x" is the $variable$ part.
- "$x \in \mathbb{N}$" is the $variable \in TYPESET$ part & $\mathbb{N}$ is the $TYPESET$.
- "$x \leq 10$" is the $variableCondition$ part.

The *elements of the set* are the *natural numbers* ($x \in \mathbb{N}$) that satisfy the condition ($x \leq 10$).

Using the *remainder* operator "mod" we can define number sets of *even*, *odd* & multiples of *fives*:

$$\{ x \mid x \in \mathbb{N} \land x \bmod 2 = 0 \} = \{ 0, 2, 4, 6, 8, 10, \ldots \}$$
$$\{ y \mid y \in \mathbb{N} \land y \bmod 2 = 1 \} = \{ 1, 3, 5, 7, 9, 11, \ldots \}$$
$$\{ z \mid z \in \mathbb{N} \land z \bmod 5 = 0 \} = \{ 0, 5, 10, 15, 20, 25, 30, 35, \ldots \}$$

Generalised Set Union & Intersection

There are *"generalised"* versions of the set union & intersection operators that apply to *"sets of sets"*.

Symbol	Description
$\bigcup AA$	*Generalised union* of the set of sets AA
$\bigcap AA$	*Generalised intersection* of the set of sets AA

Examples

$$\bigcup \{ \{\}, B, C \} = (\{\} \cup \{ a, e, i, o, u \}) \cup \{ x, y, z \}$$
$$= \{ a, e, i, o, u, x, y, z \}$$

$$\bigcap \{ \{ p, a, u, l \}, B, D \}$$
$$= (\{ p, a, u, l \} \cap \{ a, e, i, o, u \}) \cap \{ f, o, r, m, a, l, e, t, h, d, s \}$$
$$= (\{ a, u \}) \cap \{ f, o, r, m, a, l, e, t, h, d, s \}$$
$$= \{ a \}$$

$$\bigcup(\mathbb{P}(X)) = X \qquad \bigcap(\mathbb{P}(X)) = \{ \}$$

In Class Set Exercises

Given the following sets:

$$A = \{ 1, 2, 3, 4, 5, 6, 7, 8 \} \qquad B = \{ 1, 4, 7, 9, 10 \}$$
$$C = \{ 2, 4, 6 \} \qquad D = 1 \mathinner{..} 12$$

(1) Evaluate the following set expressions:

$$\mathrm{card}(B) = \qquad B \cup C =$$
$$C \cap 1 \mathinner{..} 5 = \qquad A - B =$$
$$\mathbb{P}(C) = \qquad \bigcup \{ C, \{1, 3, 5\}, \{99\} \} =$$

(2) Using *set comprehension* define the set of even numbers less than 20.

(3) Determine which of the following are *true* & which *false*:

$$5 \in D \qquad 4 \notin B$$
$$A \subseteq D \qquad B \subseteq A$$
$$C \subset B \qquad \{ 2, 4 \} \in \mathbb{P}(C)$$

PART III

Abstract Machine Notation (AMN)

for

Numbers & Sets

Machine Readability AMN: Numbers & Sets

The various B tools that we will use (as well as the ones we will not), require ASCII versions of specifications & symbols.

As we proceed through the module we will introduce the various B symbols & notation.

We will present this in two versions: a “pretty printed” version & an ASCII version that is readable by the B tools.

We begin by introducing the B symbols & ASCII versions of the *Abstract Machine Notation* (AMN) for numbers & sets:

- Numbers: types, operators & relations.
- Sets: types, operators & relations.

Machine Readability AMN: Number Types & Operators

Symbol	ASCII	Description
$\mathbb{N}$	`NAT`, `NATURAL`	Set of natural numbers from 0
$\mathbb{N}_1$	`NAT1`, `NATURAL1`	Set of natural numbers from 1
$\mathbb{Z}$	`INT`, `INTEGER`	Set of integers
$x + y$	`x + y`	x plus y
$x - y$	`x - y`	x minus y
$x \times y$	`x * y`	x multiply y
$x \div y$	`x / y`	x divided by y
$x \bmod y$	`x mod y`	remainder after x divided by y
$x \widehat{} y$	`x ** y`	x to the power y, x^y
$\mathrm{pred}(x)$	`pred(x)`	predecessor of x
$\mathrm{succ}(x)$	`succ(x)`	successor of x

Examples

$47 \div 5 =$ `47 / 5`

$3 \widehat{} 3 =$ `3 ** 3`

Machine Readability AMN: Number Operators & Relations

Symbol	ASCII	Description
$\min(A)$	`min( A )`	minimum number in set A
$\max(A)$	`max( A )`	maximum number in set A
$x \mathinner{..} y$	`x .. y`	numbers from x to y inclusive
$x = y$	`x = y`	x equal to y
$x \neq y$	`x /= y`	x not equal to y
$x < y$	`x < y`	x less than y
$x \leq y$	`x <= y`	x less than or equal to y
$x > y$	`x > y`	x greater than y
$x \geq y$	`x >= y`	x greater than or equal to y

Examples

```
min( { 1, 2, 3 } ) = 1
88 /= 100
47 >= 25
```

```
23 .. 25
78 <= 99
```

Machine Readability AMN: Set Definitions

Symbol	ASCII	Description
$x \in A$	`x : A`	x is an element of A
$x \notin A$	`x /: A`	x is not an element of A
$\varnothing, \{\,\}$	`{}`	Empty set
$\{\,1\,\}$	`{ 1 }`	Singleton set (1 element)
$\{\,1, 2, 3\,\}$	`{ 1, 2, 3 }`	Set of elements
$x\,..\,y$	`x .. y`	Range of integers from x to y inclusive
$\mathbb{P}(A)$	`POW(A)`	Power set of A
$\mathrm{card}(A)$	`card(A)`	Number of elements in set A

Examples

```
3 : { 1, 2, 3 }
{ 42 }
7 .. 10
POW( { 1, 2, 3 } )
```

```
8 /: { 1, 2, 3 }
{ 2017, 2018 }
2000 .. 2099
card( { 1, 2, 3 } )
```

```
99 /: {}
{ 1, 4, 9 }
```

Machine Readability AMN: Set Operators & Relations

Symbol	ASCII	Description
$A \cup B$	`A \/ B`	Union of A and B
$A \cap B$	`A /\ B`	Intersection of A and B
$A - B$	`A - B`	Set difference of A and B
$A = B$	`A = B`	A is equal to B
$A \neq B$	`A /= B`	A is not equal to B
$A \subseteq B$	`A <: B`	A is a subset of or equal to B
$A \subset B$	`A <<: B`	A is a strict subset of B

Examples

```
{ 1, 2, 3 } \/ { 1, 4, 5 }  =  { 1, 2, 3, 4, 5 }
{ 1, 2, 5 } /\ { 1, 4, 5 }  =  { 1, 5 }
{ 1, 2, 3 } - { 1, 2 }      = { 3 }

{ 1, 2, 3 }  /= { 1, 2, 3, 99, 100 }
{ 1, 2, 3 }  <: { 1, 2, 3, 99, 100 }
{ 1, 2 }    <<: { 1, 2, 3 }
```

Machine Readability AMN: Set Operators II

Symbol	ASCII	Description
$\bigcup AA$	`union( AA )`	Generalised union of AA
$\bigcap AA$	`inter( AA )`	Generalised intersection of AA
$\{x \mid x \in TS \wedge C\}$	`{ x | x : TS & C }`	Set comprehension

Examples

```
union( { {1, 2}, {3, 4}, {5} } ) = { 1, 2, 3, 4, 5 }
```

```
inter( { {1, 2}, {1, 3, 4} } ) = { 1 }
```

Set comprehension – numbers 0 to 10:
Symbols: $\{ x \mid x \in \mathbb{N} \wedge x \leq 10 \}$
ASCII: `{ x | x : NAT & x <= 10 }`

Converting B Symbols to Machine Readable ASCII

Using the ASCII machine readable form of AMN we can convert the $PaperRound$ machine into a machine readable version `PaperRound`.

B Symbols: $houseset \subseteq \mathbb{N}_1$
ASCII: `houseset <: NAT1`
Meaning: $houseset$ is a set variable, e.g. $houseset = \{ 21, 42 \}$.

B Symbols: $new \in \mathbb{N}_1 \land new \notin houseset$
ASCII: `new : NAT1 & new /: houseset`
Meaning: new is a number variable & its a "new" house number, i.e. not delivered to. E.g. $new = 84$ and $84 \notin houseset$.

B Symbols: $houseset := houseset \cup \{ new \}$
ASCII: `houseset := houseset \/ { new }`
Meaning: add a new house number to delivered to houses, e.g. $houseset := houseset \cup \{ 84 \}$.

B Symbols: $ans \leftarrow number$
ASCII: `ans <-- number`
Meaning: ans is output by the $number$ operation.

Examples: Converting AMN Symbol to ASCII

Consider statements about the paper round manager example using the $houseset$ variable that keeps track of houses that receive paper deliveries.

For each of the following: explain what it means & convert it to ASCII:

1. $houseset \subseteq \mathbb{N}$
2. $houseset \in \mathbb{P}(\mathbb{N}_1)$
3. $houseset \subseteq 1 \mathinner{..} 163$
4. $houseset \neq \{\}$
5. $139 \notin houseset$
6. $hh \in houseset$
7. $houseset - \{\ hh1,\ hh2\ \} = \{\}$

Answers:

Examples: Converting AMN Symbol to ASCII

Consider statements about the paper round manager example using the $houseset$ variable that keeps track of houses that receive paper deliveries.

For each of the following: explain what it means & convert it to ASCII:

1. $houseset \subseteq \mathbb{N}$
2. $houseset \in \mathbb{P}(\mathbb{N}_1)$
3. $houseset \subseteq 1 \mathinner{..} 163$
4. $houseset \neq \{\}$
5. $139 \notin houseset$
6. $hh \in houseset$
7. $houseset - \{\ hh1,\ hh2\ \} = \{\}$

Answers:

1. `houseset <: NAT`

Examples: Converting AMN Symbol to ASCII

Consider statements about the paper round manager example using the $houseset$ variable that keeps track of houses that receive paper deliveries.

For each of the following: explain what it means & convert it to ASCII:

1. $houseset \subseteq \mathbb{N}$
2. $houseset \in \mathbb{P}(\mathbb{N}_1)$
3. $houseset \subseteq 1 \mathinner{..} 163$
4. $houseset \neq \{\}$
5. $139 \notin houseset$
6. $hh \in houseset$
7. $houseset - \{\ hh1,\ hh2\ \} = \{\}$

Answers:

1. `houseset <: NAT`
2. `houseset : POW(NAT1)`

Examples: Converting AMN Symbol to ASCII

Consider statements about the paper round manager example using the $houseset$ variable that keeps track of houses that receive paper deliveries.

For each of the following: explain what it means & convert it to ASCII:

1. $houseset \subseteq \mathbb{N}$
2. $houseset \in \mathbb{P}(\mathbb{N}_1)$
3. $houseset \subseteq 1 \mathinner{..} 163$
4. $houseset \neq \{\}$
5. $139 \notin houseset$
6. $hh \in houseset$
7. $houseset - \{\ hh1,\ hh2\ \} = \{\}$

Answers:

1. `houseset <: NAT`
2. `houseset : POW(NAT1)`
3. `houseset <: 1..163`

Examples: Converting AMN Symbol to ASCII

Consider statements about the paper round manager example using the $houseset$ variable that keeps track of houses that receive paper deliveries.

For each of the following: explain what it means & convert it to ASCII:

1. $houseset \subseteq \mathbb{N}$
2. $houseset \in \mathbb{P}(\mathbb{N}_1)$
3. $houseset \subseteq 1 \mathinner{..} 163$
4. $houseset \neq \{\}$
5. $139 \notin houseset$
6. $hh \in houseset$
7. $houseset - \{\ hh1,\ hh2\ \} = \{\}$

Answers:

1. `houseset <: NAT`
2. `houseset : POW(NAT1)`
3. `houseset <: 1..163`
4. `houseset /= {}`

Examples: Converting AMN Symbol to ASCII

Consider statements about the paper round manager example using the $houseset$ variable that keeps track of houses that receive paper deliveries.

For each of the following: explain what it means & convert it to ASCII:

1. $houseset \subseteq \mathbb{N}$
2. $houseset \in \mathbb{P}(\mathbb{N}_1)$
3. $houseset \subseteq 1 \mathinner{..} 163$
4. $houseset \neq \{\}$
5. $139 \notin houseset$
6. $hh \in houseset$
7. $houseset - \{\ hh1,\ hh2\ \} = \{\}$

Answers:

1. `houseset <: NAT`
2. `houseset : POW(NAT1)`
3. `houseset <: 1..163`
4. `houseset /= {}`
5. `139 /: houseset`

Examples: Converting AMN Symbol to ASCII

Consider statements about the paper round manager example using the $houseset$ variable that keeps track of houses that receive paper deliveries.

For each of the following: explain what it means & convert it to ASCII:

1. $houseset \subseteq \mathbb{N}$
2. $houseset \in \mathbb{P}(\mathbb{N}_1)$
3. $houseset \subseteq 1 \mathinner{..} 163$
4. $houseset \neq \{\}$
5. $139 \notin houseset$
6. $hh \in houseset$
7. $houseset - \{ hh1, hh2 \} = \{\}$

Answers:

1. `houseset <: NAT`
2. `houseset : POW(NAT1)`
3. `houseset <: 1..163`
4. `houseset /= {}`
5. `139 /: houseset`
6. `hh : houseset`

Examples: Converting AMN Symbol to ASCII

Consider statements about the paper round manager example using the $houseset$ variable that keeps track of houses that receive paper deliveries.

For each of the following: explain what it means & convert it to ASCII:

1. $houseset \subseteq \mathbb{N}$
2. $houseset \in \mathbb{P}(\mathbb{N}_1)$
3. $houseset \subseteq 1 \mathinner{..} 163$
4. $houseset \neq \{\}$
5. $139 \notin houseset$
6. $hh \in houseset$
7. $houseset - \{ hh1, hh2 \} = \{\}$

Answers:

1. `houseset <: NAT`
2. `houseset : POW(NAT1)`
3. `houseset <: 1..163`
4. `houseset /= {}`
5. `139 /: houseset`
6. `hh : houseset`
7. `houseset - { hh1, hh2 } = {}`

PART IV

Using Sets, Constants & Variables in a B Machine

What Sets are used for in the B-Method

Recall that the initial stage of the B-method is to *formalise* the *system requirements* by constructing a *B model* (*abstract machine*).

This initial *abstract machine* describes:

- The main *state variables* of the system & their *types*.
- The *"state invariants"* — *properties* which these state variables **must satisfy at all times**.
- The *operations* that transform & update these variables.

This initial abstract machine is a *formalisation* (i.e. specification) of the system requirements; that ignores as much *implementation detail* as possible using *abstraction*.

Sets are ideally suited to achieving these two goals of: *formalisation* & *abstraction*.

For this reason *sets* are extensively used when defining an abstract machine.

Different Categories of Sets Used in an Abstract Machine

In this part of the lecture we shall show how *sets* can be used within the B-method to define an *abstract machine*.

There are 3 different categories of sets that can be used to define an abstract machine:

- *"Built in"* Sets: $\mathbb{N}$, $\mathbb{N}_1$ & $\mathbb{Z}$. (We have already covered these.)
- *"Enumerated"* Sets – completely defined.
- *"Deferred"* Sets – incomplete & delayed definition.

We have already seen examples of the *number sets*, so will focus on the last two.

Enumerated Sets

Enumerated sets can be used to define types similar to the enumerated types used in many programming languages, e.g. Java.

The syntax of an *enumerated* set is:

$$ENUMERATED\_SET = \{ element_1, element_2, \ldots, element_n \}$$

The elements of $ENUMERATED\_SET$ are the individual elements:

$$element_1, \ldots, element_n$$

Note: that all of the above individual elements are *distinct/unique*.

The B convention is that the name of the set is all in *UPPER CASE*.

"Abstraction Rating"

The use of enumerated sets allows for a certain amount of *abstraction*, since we are able to use words as the elements of the set & we do not have to further define these elements.

Examples: “Enumerated” Sets

$$ANSWER = \{ yes, no \}$$

– set of possible question answers.

$$CURSOR\_KEY = \{ Up\_Key, Down\_Key, Left\_Key, Right\_Key \}$$

– set of possible cursor key presses & movements.

$$DIRECTION = \{ North, South, East, West \}$$

– set of points of the compass & directions of movement.

$$\begin{aligned} MESSAGE = \{ & Success, \\ & Are\_You\_Using\_Your\_Own\_Bag, \\ & Have\_You\_Swiped\_Your\_Nectar\_Card, \\ & Error\_Unknown\_Item\_In\_Bagging\_Area, \\ & Thats\_The\_Third\_Bottle\_Of\_Wine\_This\_Week\_V\_Bad \} \end{aligned}$$

– set of messages output (spoken) by a self service till in a supermarket.

“Deferred” Sets

When a *“deferred”* set is declared the actual elements of the set are **not specified**.

This allows the specifier to *abstract away from* the detail of:

- what the actual elements of the set are; &
- how they are represented.

This is an important & useful feature of B machines.

“Abstraction Rating”

“Deferred” sets allow for a high degree of *abstraction*, since we only need to give the set’s name, not its elements.

However, the *“deferred & missing information”* about its elements **MUST** be supplied by the developer at some point in the *“refinement”* phase, otherwise no program code can be generated.

Examples: “Deferred” Sets

$CAR\_REGISTRATION$

– set of all possible car registrations.

$DRIVERS$

– set of all legal? car drivers.

$PEOPLE$

– set of all uniquely identifiable people.

$UOW\_STUDENTS$

– set of all University of Westminster students.

NOTE: The majority of the *non-number* sets used in a B specification are often initially declared as deferred sets.

Declaring a "Set" in a B Machine

When any *set* is declared in a B machine we must supply a *name* & if it is not a *deferred* set then the list of its elements.

Set declarations are listed in the "SETS" clause of a machine, they are separated by *semi-colons ";"*.

For example, using the sets we have previously defined, we can introduce 3 *enumerated* sets & a *deferred* set as follows in our Sets machine:

Sets

```
MACHINE Sets

SETS
  ANSWER     = { yes, no } ;

  CURSOR_KEY = { Up_Key, Down_Key, Left_Key, Right_Key } ;

  DIRECTION  = { North, South, East, West } ;

  PEOPLE
```

Using Sets to Define *"Type"* Information

We now know how to define *sets* in a B machine.

This is essential since **all** *constants* & *variables* declared in a B machine **must have a type**.

This is achieved by stating that each *constant* & *variable* is either a *member of* a set or a *subset of* a set.

That is achieved by using one of the following:

- *"member of"* — if it is a single valued (non-set) variable or constant then:
  $var \in TYPESET$
  `var : TYPESET.`
- *"subset of"* — if it is a set valued variable or constant then:
  $setvar \subseteq TYPESET$
  `setvar <: TYPESET.`
- An alternative for set valued variables is to state that it is *"an element of a power set"*, i.e. a set, then use:
  $setvar \in \mathbb{P}(TYPESET)$,
  `setvar : POW( TYPESET ).`

Declaring a “Constant” in a B Machine

When a *constant* is declared in a B machine we must supply a:

- *name*,
- its *type* &
- optionally any *properties* its value must satisfy.

For each *constant* its:

- *declaration* (the *constant's name*), is listed in the “`CONSTANTS`” clause. Declarations are separated by *commas* –“,”.
- *type* information for each constant is listed in the “`PROPERTIES`” clause. The types must be either standard types or sets that have been introduced.
  Additional *properties* about its value may optionally be included here as well.
  Since the *type* & *properties* information are *logical statements* these are combined by *“logical and”* using B symbol: “$\wedge$” or ASCII: “`&`”.

Example: Declaring a "Constant" in a Machine

Using the previously defined sets, we can introduce 2 people constants as follows:

Sets

```
MACHINE Sets

  SETS
    PEOPLE

  CONSTANTS
    Jim, Sue

  PROPERTIES
    Jim : PEOPLE  &  Sue : PEOPLE  &  Jim /= Sue
```

Note:

As well as the type information about the 2 constants, we have also included the predicate: "`Jim /= Sue`" (or $Jim \neq Sue$).

This is almost always necessary, since we want `Jim` & `Sue` to represent **two different people** & so we **MUST** explicitly state that they are **not equal**.

Declaring a “Variable” in a Machine

When a *variable* is declared in a B machine we must supply a:

- *name*,
- its *type* & any additional *properties* that it must satisfy,
- its *initial value*.

For each *variable* its:

- *declaration* (the *variable’s name*), is listed in the “`VARIABLES`” clause of the machine.
  Declarations are separated by *commas* –“,”.
- *type* & *properties* information of each variable is listed in the “`INVARIANT`” clause.
  Since the *type* information is a *logical statement* these are combined by *“logical and”* – “$\wedge$” or “`&`”.
- *initialisation* of the variable is done in the “`INITIALISATION`” clause, using *assignment* – “$:=$”.

Example: Declaring a "Variable" in a Machine

Using the previously defined sets, we can introduce 3 variables as follows:

Variables_in_Machine

```
VARIABLES
  dirs , keys , pointing

INVARIANT
  dirs <: DIRECTION  &
  keys <: CURSOR_KEY &
  pointing : dirs

INITIALISATION
  dirs,  keys,  pointing
        :=
  { North, South },  { Up_Key, Down_Key },  North
```

Note: we are using *"multiple" assignment* to simultaneously initialise the 3 variables. *Multiple assignment* has the form:

$$var_1,\ var_2,\ ...,\ var_n\ :=\ val_1,\ val_2,\ ...,\ val_n$$

With val_1 being assigned to variable var_1, etc.

Example: `Sets` Machine

Sets

```
MACHINE Sets

  SETS
    PEOPLE ;
    ANSWER     = { yes, no } ;
    CURSOR_KEY = { Up_Key, Down_Key, Left_Key, Right_Key } ;
    DIRECTION  = { North, South, East, West }

  CONSTANTS
    Jim, Sue
  PROPERTIES
    Jim : PEOPLE & Sue : PEOPLE & Jim /= Sue

  VARIABLES
    dirs , keys , pointing
  INVARIANT
    dirs <: DIRECTION & keys <: CURSOR_KEY & pointing : dirs

  INITIALISATION
    dirs, keys, pointing
          := { North, South }, { Up_Key, Down_Key }, North

  OPERATIONS
    ...
END
```