

# Formal Specification

## Lecture 2

What will be covered:

- Variables
- Types
- Predicates
- Sets
- Quantifiers

Yeah this seems basic...so feel free to chat amongst yourselves and not pay attention...because you're not going to struggle with the later stuff...it's really a piece of piss...

# Variables

A variable, believe it or not, varies...

You *should* know already that a variable is a value, represented by some other character (usually an alphabetic character) as the value is unknown.

*x*, *y* and *z* are commonly used variables.

# Types

A variable has a **type**.

$\mathbb{Z}$  and  $\mathbb{N}$  are types:

$\mathbb{Z}$  – integers

$\mathbb{N}$  – natural numbers

To assign a variable a type you write **<variable> : <desired type>**

such as...

**$x:\mathbb{Z}$**  ( **$x$**  has type  $\mathbb{Z}$ , so  **$x$**  is an integer)

**$x,y:\mathbb{N}$**  ( **$x$**  and  **$y$**  both have type  $\mathbb{N}$ , so both are natural numbers).

# Predicates

A **predicate** is a proposition with **variables**.

Remember from the previous lecture that a proposition is something that has a truth-value?

So if  $x, y : \mathbb{N}$  then

$x = y + 3$  is a **predicate**.

It has two free variables — guess what they are.

Assign your own values to  $x$  and  $y$  and determine the truth value of the predicate!

# Quantifiers

Sometimes we want to express general truths. Then we use **quantifiers**.

Suppose we want to assert that 'for all  $x$ ,  $x + 1 > 1$ '. We use a quantifier:

$\forall x : \mathbb{N} \bullet x + 1 > 1$ . (Read  $\forall$  as 'for all', read  $\bullet$  as 'it is true that'.)

Notice how careful we are to specify a type. The type can make a difference:

$\forall x : \mathbb{Z} \bullet x + 1 > 1$ .

The truth-value of the  $\mathbb{N}$ -typed version is  $T$ , the truth-value of the  $\mathbb{Z}$ -typed version is  $F$ .

# The existential quantifier

Obviously we may wish not to point to a **general truth** but to a **possibility**.

Then we use  $\exists$  ('there exists'):

$\exists x : \mathbb{N} \cdot x > 5$  is true (there exists a number greater than 5).

$\exists x : \mathbb{N} \cdot x + 3 < 2$  is false.

$\exists x : \mathbb{Z} \cdot x + 3 < 2$  is true.

Finally,  $\exists!$  means 'there exists a unique'.

$\exists! x : \mathbb{N} \cdot x = 25$  is true.

$\exists! x : \mathbb{N} \mid x < 6 \wedge x > 4 \cdot T$  is true (there is just one number less than 6 and more than 4).

$\exists! x : \mathbb{Z} \cdot x^2 = 25$  is false.

# Syntax

Yes, there is a syntax to Z expressions...learn it...practice it and get it bloody right...otherwise you fail...it's quite simple really...

An expression is written in the following format:

$$\forall \langle \text{name} \rangle : \langle \text{type} \rangle [ \mid \langle \text{constraint} \rangle ] \cdot \langle \text{predicate} \rangle$$

This is read as:

“For all  $\langle \text{name} \rangle$  of type  $\langle \text{type} \rangle$  [such that  $\langle \text{constraint} \rangle$ ], it is true that  $\langle \text{predicate} \rangle$ .”

$$\exists \langle \text{name} \rangle : \langle \text{type} \rangle [ \mid \langle \text{constraint} \rangle ] \cdot \langle \text{predicate} \rangle$$

This is read as:

“There exists a  $\langle \text{name} \rangle$  of type  $\langle \text{type} \rangle$  [such that  $\langle \text{constraint} \rangle$ ], it is true that  $\langle \text{predicate} \rangle$ .”

# Quantifiers Summarised

Tell me whether the following are true or false:

- $\forall x : \mathbb{N} \mid x < 10 \cdot x + 9 > 12.$
- $\exists x : \mathbb{N} \mid x < 10 \cdot x + 9 > 12.$
- $\exists_1 x : \mathbb{N} \mid x < 10 \cdot x + 9 > 12.$

That's it for Chapter 2 of "The essence of Z". Do exercises 2.5 and 2.6.

**Warning:** Propositions and predicates are the very foundations of this course.

You will make your life a **lot** easier if you make sure that you are fluent with them **now**.

The following slides are chapter 3 – 3.2 in  
“The Essence of Z”

# Types...defining your own

Everything in the “real-world” can be specified using Z.

As is the same with programming in Java you have objects, and instances of these objects...though in Z we call them *types* (objects) and *variables* (instances of objects). We say that a *variable ranges over a type*.

There are “primitive types” already declared, ready for you to use, in Z - two of which we have met  $\mathbb{Z}$  (integers) and  $\mathbb{N}$  (natural numbers)

**YOU MUST DECLARE THE TYPE OF A VARIABLE  
BEFORE YOU USE IT!  
(LOOK BACK TO SLIDE 3 IN THIS LECTURE!)**

# How to define your own type

To define a type you simply put the name of your “type” in capital letters, followed by '::=' then the values that your “type” can have, separated by '|'.

This is called *free type definition*.

Example:

```
COLOUR ::= red | green | blue
```

This defines type 'COLOUR' and three values possible to any variable of type 'COLOUR'.

# Basic Types

You can add a basic type also to Z.

You can add a basic type **PERSON**.

**[PERSON]**

This just declares a type — and says nothing of what is or is not a person. You can still declare  $x : \mathbf{PERSON}$ , but where your people come from — that's none of Z's business.

# Predicates distinct from types

We can test the value of a variable using a proposition, such as  $x = 2$ , for example.

Then  $x = 2$  is assigned truth-value  $T$  if we assign  $x$  the value 2, otherwise  $x = 2$  is assigned truth-value  $F$ .

$x : \mathbb{Z}$  asserts what  $x$  ranges over.

In a broader philosophical sense, typing declarations  $x : \mathbb{Z}$  are clearly related to predicates, but in  $\mathbb{Z}$  they are formally different.

In English... ' $x : \mathbb{Z}$ ' is not a predicate, it is an assertion, it has no truth value because  $x$  *has* type  $\mathbb{Z}$ , we cannot say this is false.

# Sets

A set is a collection of elements, of the *same* type.

We can declare sets as follows:

```
numset == {4, 5, 6, 7, 8, 9}
```

```
numset == 4..9
```

```
numset == {n : Z | n ≥ 4 ∧ n ≤ 9 • n}
```

```
numset == {n : Z | n ≥ 2 ∧ n ≤ 7 • n + 2}
```

(These are all equivalent.)

# Sets

If the declared variable is 'naked' after the bullet, we may omit it:

```
numset == {n :  $\mathbb{Z}$  |  $n \geq 4 \wedge n \leq 9$ }
```

is shorthand for

```
numset == {n :  $\mathbb{Z}$  |  $n \geq 4 \wedge n \leq 9 \bullet n$ }.
```

We are only returning the value  $n$ , we make no changes to it.

This is merely a "set" of  $n$ 's with value between 4 and 9.

# Sets

If the predicate is just true, we may omit it:

$$\text{evens} == \{n : \mathbb{Z} \cdot 2 * n\}$$

is shorthand for

$$\text{evens} == \{n : \mathbb{Z} | \text{T} \cdot 2 * n\}.$$

# The empty set

The emptyset  $\emptyset$  (or  $\{\}$ ) means

$$\{n : \mathbb{Z} \mid F \cdot n\}.$$

This is the same set as

$$\{n : \mathbb{Z} \mid F \cdot 2 * n\}.$$

Why?

# Murdoch's tips...

On **no account** interrupt me if some totally basic element of notation escapes you — leave that till the end of the course, or even better, till the week before the exams.

Are you remembering to **forget this material** as soon as you leave the lecture theatre? Good.

On **no account** make sure you understand sections 3.1 and 3.2 of the book. Obviously, **ignore** my instructions to do exercises.

# Murdoch's tips...

I hope you have had the sense to ignore me and not buy “The essence of Z”.

You can rely on my 100% sympathy tutoring each of you in turn through the material on the phone when you're in a panic, any time you feel the urge.

If by accident you find yourself reading “The essence of Z”, read it linearly (like novel). At the first word or concept you encounter that you do not immediately understand, stare at it for hours in a half-asleep half-awake trance state, rather than immediately taking five minutes on the web to read around the subject, or phoning a friend.

I had all the advantages when I was an undergrad . . . no mobile phones, no internet, and no good advice.