MIRPUR UNIVERSITY OF SCIENCE AND TECHNOLOGY (MUST), MIRPUR

DEPARTMENT OF SOFTWARE ENGINEERING

Formal Methods in Software Engineering

Lecture [14]: Elements and Structure Z Language

Engr. Samiullah Khan

(Lecturer)

Topics discussed in Today's Lectures

- Elements of Z Language
- Structure of Z Language

3 basic constructs of Z Language

- Declarations introduce variables
- Expressions describe values that variables can assume
- Predicates place constraints on the values that variables do assume

Sets in Z Language

Set display

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

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

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

{ red, yellow, green }

{ yellow, red, green, 1, 2 } Type error!

Set names

$\mathbb{Z}$

DICE

LAMP

Set expressions

$1 \mathrel{..} 6$

$\{ i : \mathbb{Z} \mid 1 \leq i \leq 6 \}$

Types in Z Language

Every object belongs to a set called its *type*.

1, 2, ... all belong to the type $\mathbb{Z}$.

red, green belong to the type COLOR.

Every type must be introduced in a *declaration*. There are two ways to declare types.

Free types are like enumerations.

COLOR ::= red | green | blue | yellow
| cyan | magenta | white | black

Basic types can include indefinitely many elements.

$\mathbb{Z}$
[NAME]

VARIABLES in Z Language

A *variable* is a name for an object: its *value*. Variables are introduced in *declarations*.

x: S The value of x belongs to set S

Axiomatic definitions declare *global variables* and can include optional *constraints*.

$$\begin{array}{|l} d_1, d_2 : DICE \\ \hline d_1 + d_2 = 7 \\ d_1 < d_2 \end{array}$$

Constants are variables that are constrained to one value (ℙ S means *set of* S).

$$\begin{array}{|l} DICE : \mathbb{P}\,\mathbb{Z} \\ \hline DICE = 1 \mathinner{..} 6 \end{array}$$

Abbreviation definitions can also declare constants.

$$DICE == 1 \mathinner{..} 6$$

TYPES, SETS AND NORMALIZATION in Z Language

Types are sets, but not all sets are types.

ODD, EVEN, PRIME are just sets, $\mathbb{Z}$ is their type.

Any set can appear in a declaration.

| e: EVEN
| o: ODD
| p: PRIME

In a *normalized* declaration, we write the *signature* to show the type.

| e,o,p: $\mathbb{Z}$
|―――――――――
| $e \in EVEN$
| $o \in ODD$
| $p \in PRIME$

The type determines which variables can be combined in expressions.

EXPRESSIONS AND OPERATORS, ARITHMETIC in Z Language

Expressions have values. The simplest expressions are constants and variables.

1, 2, red, x, d_1, DICE, $\mathbb{Z}$, ...

Operators build larger expressions from smaller ones. Arithmetic provides familiar examples.

m+n	Addition
m-n	Subtraction
m*n	Multiplication
m **div** n	Division
m **mod** n	Remainder (modulus)
m ≤ n	Less than or equal
m .. n	Number range (up to)
min A	Minimum of a set of numbers
max A	Maximum of a set of numbers

EXPRESSIONS AND OPERATORS, ARITHMETIC in Z Language

SET OPERATORS

The *size* operator # counts elements.

$\# \{ red, yellow, blue, green, red \} = 4$

The *union* operator $\cup$ combines sets.

$\{ 1, 2, 3 \} \cup \{ 2, 3, 4 \} = \{ 1, 2, 3, 4 \}$

The *difference* operator \ removes the elements of one set from another.

$\{ 1, 2, 3, 4 \} \setminus \{ 2, 3 \} = \{ 1, 4 \}$

The *intersection* operator $\cap$ finds the elements common to both sets.

$\{ 1, 2, 3 \} \cap \{ 2, 3, 4 \} = \{ 2, 3 \}$

Set operators work with sets of any type, but

$\{ 1, 2, 3 \} \cup \{ red, green \}$ Type error!

EXPRESSIONS AND OPERATORS, ARITHMETIC in Z Language

PREDICATES

Predicates *constrain* values. Many have the form e_1 R e_2, where e_1 and e_2 are expressions.

Equality, x and y have the same value.

$x = y$

Arithmetic relations, n is less than m.

$n < m$

Set membership, x is a member of S.

$x \in S$

Subset, members of S are members of T.

$S \subseteq T$

Predicates are not expressions. They do not have values, they are *true* or *false*.

EXPRESSIONS AND OPERATORS, ARITHMETIC in Z Language

PUTTING THE ELEMENTS TOGETHER

A train moves at a constant velocity of sixty miles per hour for four hours.

$$
\begin{array}{|l}
\text{distance, velocity, time} : \mathbb{N} \\
\hline
\text{distance} = \text{velocity} * \text{time} \\
\text{velocity} = 60 \\
\text{time} = 4
\end{array}
$$

How far does the train travel?

Philip works on the adhesives team in the materials group, which is part of the research division.

$$
\begin{array}{|l}
\text{philip} : \text{PERSON} \\
\text{adhesives, materials, research,} \\
\quad \text{manufacturing} : \mathbb{P}\ \text{PERSON} \\
\hline
\text{adhesives} \subseteq \text{materials} \\
\text{materials} \subseteq \text{research} \\
\text{philip} \in \text{adhesives}
\end{array}
$$

Is Philip in the research division?

ELEMENTS OF Z: DISCUSSION

1. Compare this Z

$DICE == 1 \mathrel{..} 6$

$$
\begin{array}{|l}
d_1, d_2 : DICE \\
\hline
d_1 + d_2 = 7 \\
d_1 < d_2
\end{array}
$$

with this C.

```
typedef int DICE;
DICE d1, d2;
```

2. Compare this basic type and definition

$[X]$

$$
\begin{array}{|l}
x, y : X
\end{array}
$$

with this free type.

$X ::= x \mid y$

STRUCTURE OF Z

TUPLES

Tuples can resemble C *structures* or Pascal *records*.

Tuples are instances of *Cartesian product types*.

First declare types for each component.

[NAME]

$ID == \mathbb{N}$

DEPT ::= admin | manufacturing | research

Define the Cartesian product type EMPLOYEE.

$EMPLOYEE == ID \times NAME \times DEPT$

Declare tuples which are instances of the type.

Frank, Aki: EMPLOYEE

Frank = (0019, frank, admin)
Aki = (7408, aki, research)

STRUCTURE OF Z

RELATIONS

Relations are sets of tuples. They can resemble *tables* or *databases*.

ID	NAME	DEPT
0019	Frank	Admin
0308	Philip	Research
7408	Aki	Research
...	...	...

In Z this can be expressed

```
Employee: ℙ EMPLOYEE
―――――――――――――――――――
Employee = {
   (0019, frank, admin),
   (0308, philip, research),
   (7408, aki, research),
   ...
   }
```

STRUCTURE OF Z

PAIRS

Pairs are tuples with just two components.

(aki, 4117)

The *maplet* arrow provides alternate syntax without parentheses.

aki $\mapsto$ 4117

The *projection* operators *first* and *second* extract the components of a pair.

first(aki,4117) = aki

second(aki, 4117) = 4117

STRUCTURE OF Z

BINARY RELATIONS (1)

Binary relations are sets of pairs.

$\mathbb{P}\,(NAME \times PHONE)$

or

$NAME \leftrightarrow PHONE$

Binary relations can model lookup tables.

NAME	PHONE
Aki	4019
Philip	4107
Doug	4107
Doug	4136
Philip	0113
Frank	0110
Frank	6190
...	...

In Z this can be expressed

phone: NAME ↔ PHONE

phone = {
...
aki ↦ 4019,
philip ↦ 4107,
doug ↦ 4107,
doug ↦ 4136,
philip ↦ 0113,
frank ↦ 0110,
frank ↦ 6190,
...
}

THANKS