

## Defining Sets

### •Enumerating elements

- Write out all the members of the set
- Works OK for small sets
- Very clumsy for large sets

### •Set comprehension

- Define a set by specifying constraints on its members
- In effect, write membership requirements
- Actual set members are known indirectly

## Set Comprehensions

$$NAT == \{ n: \mathbb{Z} \mid n \geq 0 \cdot n \}$$

Declaration

Predicate

Expression

$$NAT2 == \{ n: \mathbb{Z} \mid n \geq 0 \}$$

$$NATX3 == \{ n: \mathbb{N} \mid n \neq 0 \wedge n \bmod 3 = 0 \cdot n \}$$

$$NATX3B == \{ n: \mathbb{N}_1 \cdot 3 * n \}$$

$$SOMETHING == \{ x, y, z: \mathbb{Z} \mid x > y \wedge y > z \}$$

## Generating Functions

### •Functions

- Used a lot in Z specifications
- Are really sets of ordered pairs (maplets)

### •Generating functions

- Enumerate the maplets (like sets)
- Specify using declaration and predicate
- Create directly with lambda ( $\lambda$ ) notation

## Using Lambda Expressions

$$\text{add3a: } \mathbb{Z} \rightarrow \mathbb{Z}$$

$$\forall j: \mathbb{Z} \cdot \text{add3a } j = j + 3$$

Original function definition

$$\text{add3b} == \{ j: \mathbb{Z} \cdot j \mapsto j + 3 \}$$

Set comprehension form

$$\text{add3c} == (\lambda j: \mathbb{Z} \cdot j + 3)$$

Lambda notation

$$\text{add} == (\lambda x, y: \mathbb{Z} \cdot x + y)$$

What about this one?

## Restricted Quantifier (Universal)

$$\forall i: \mathbb{N} \mid i < 3 \cdot i * i < 10$$

Add restrictive predicate to universal quantifier

$$\forall i: \mathbb{N} \cdot i < 3 \Rightarrow i * i < 10$$

Restriction equivalent to adding implication to predicate

## Another Example

[MODULE]

PROJECT ::= EBT | PALMUI | COURSEDB

project: MODULE  $\rightarrow$  PROJECT  
module\_LOC: MODULE  $\rightarrow$   $\mathbb{N}$

$\forall m: \text{MODULE} \mid \text{module\_LOC } m > 1000 \cdot \text{project } m = \text{COURSEDB}$

$\forall m: \text{MODULE} \cdot \text{module\_LOC } m > 1000 \Rightarrow \text{project } m = \text{COURSEDB}$

## Restricted Quantifier (Existential)

$\exists m: \text{MODULE} \mid \text{project } m = \text{EBT} \cdot \text{module\_LOC } m > 100$

Add restrictive  
predicate to existential  
quantifier

$\exists m: \text{MODULE} \cdot \text{project } m = \text{EBT} \wedge \text{module\_LOC } m > 100$

Restriction equivalent to adding  
conjunction to predicate

## Local Definition

$\forall m: \text{MODULE} \mid \text{project } m = \text{PALMUI}$   
 $\cdot \text{module\_LOC } m > 10 \wedge \text{module\_LOC } m \leq 1000$

Note repeated expression

$\forall m: \text{MODULE} \mid \text{project } m = \text{PALMUI}$   
 $\cdot \text{let } \text{loc} ::= \text{module\_LOC } m \cdot \text{loc} > 10 \wedge \text{loc} \leq 1000$

Include local definition and then  
use locally defined variable

## Conditional Expression

$\text{bigger}: \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$

$\forall x, y: \mathbb{Z} \cdot \text{bigger}(x, y) = \text{if } x > y \text{ then } x \text{ else } y$

How does this function work?