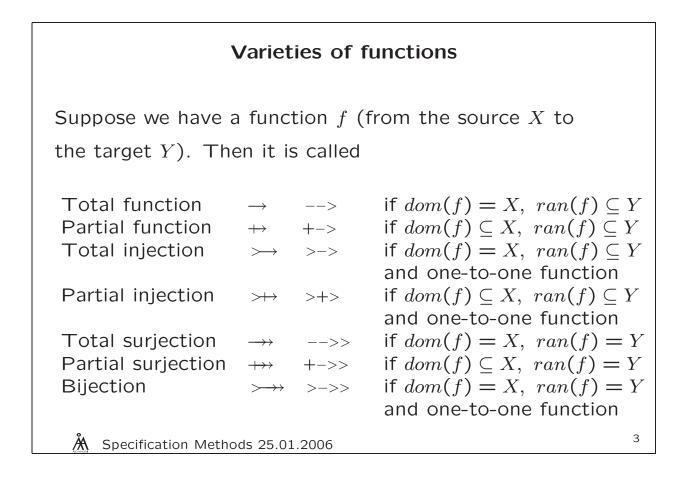
Functions

- Functions form a special class of relations that satisfy additional requirement: any element of the source set can be related to no more than 1 element of the target
- Functionality requirement mathematically:
 (x, y) ∈ R ∧ (x, z) ∈ R ⇒ y = z
- Any operation applicable to a relation or a set is also applicable to a function. For example, we can talk about the domain and the range of a function.
- If f is a function, then f(x) is the result of the function f for the argument x

A Specification Methods 25.01.2006

Functions (cont.)

- Functions are called *total* if their domain is the whole source set.
- Functions are called *partial* if their domain is a subset of the source set.
- Functions are called *injective* or (*one-to-one*) if for every element y from their range exists only one element x from their domain such that f(x) = y.
- Functions are called *surjective* if their range is the whole target set.



Lambda notation for functions

- In addition to defining functions as sets of pairs (relations), *lambda notation* can be used to introduce new functions.
- Lambda notation allows us to define a new function f by describing the result f(x) for any given argument x.
- The general form of a function is then

$$\lambda x \bullet (x \in T \mid E)$$

"the function maps x, of type T, to the value E".

• The corresponding ASCII notation - %x. (x:T|E)

Specification Methods 25.01.2006

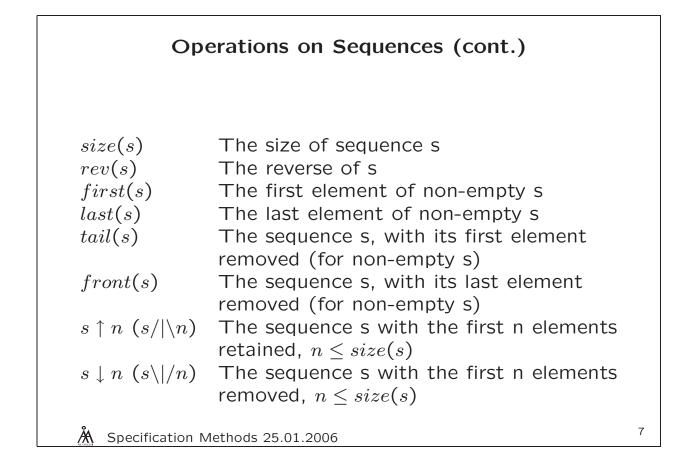
Sequences

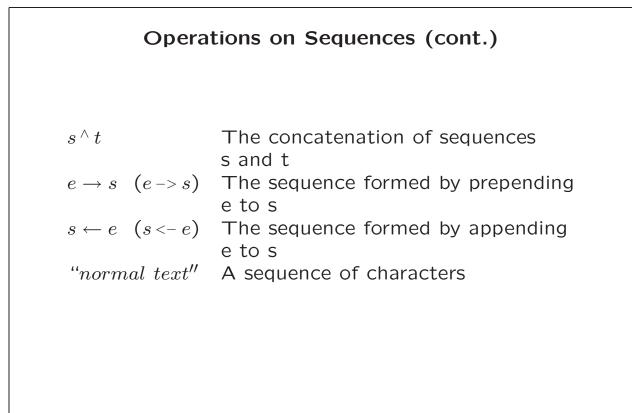
- Sequences are used to describe finite ordered lists of elements of a given type.
- A sequence over a set S is a total function from an interval 1..*n* (for some $n \in NAT$) to *S*.
- In a sequence, elements are ordered and may appear more than once.
- Any operation applicable to a function, a relation, or a set is also applicable to a sequence.

A Specification Methods 25.01.2006

	Operations on Sequences	
$[e_1, \cdots, e_n]$	The sequence containing elements e_1, \cdots, e_n This is the same as $\{(1, e_1), \cdots, (n, e_n)\}$	^{2}n
[e]	The singleton sequence with element e	
[] (<>)	The empty sequence	
seq(S)	The set of finite sequences of elements	
	from S	
seq1(S)	The set of finite non-empty sequences of elements from S: $seq1(S) = seq(S) - \{[]\}$	
iseq(S)	The set of injective sequences of elements	
	from S, i.e. sequences without repetitions	
perm(S)	Permutations of elements from a finite S, i.e. sequences that contain all elements from S without repetitions	
		6

A Specification Methods 25.01.2006





Arrays

- An array is a named, indexed collection of values of a given type.
- The array values can be accessed (read and updated) by using appropriate indexes.
- If we use 1..n (for some n ∈ NAT) as our index set, then an array (of type S) can be modelled as a sequence from 1..n to S.
- In fact, any set can be used as the index set for arrays. Therefore, arrays can be modelled as total functions from S (index set) to T (the type of array values).

A Specification Methods 25.01.2006

Array assignment

• The abstract machine notation allows us to assign values to indexed elements of arrays:

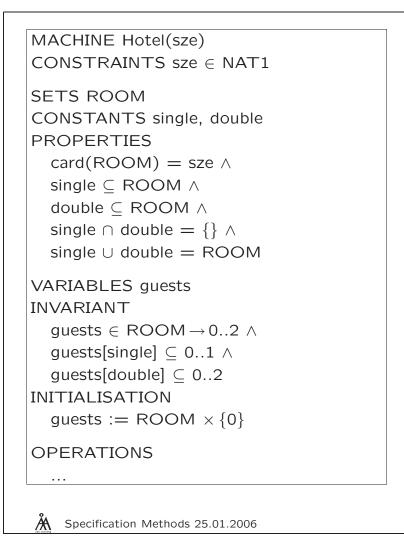
$$a(i) := E$$

• This is the shorthand for the following assignment:

$$a := a <+ \{(i, E)\}$$

• The weakest precondition for the array assignment is calculated then as follows

$$[a(i) := E] P = P[a < + \{(i, E)\} / a]$$



```
checkin(rr,nn) =
   PRE rr \in ROOM \land nn \in 1..2 \land
       guests(rr) = 0 \land
       (rr \in single \Rightarrow nn = 1)
   THEN guests(rr) := nn
   END;
 checkout(rr) =
   PRE rr ∈ ROOM
   THEN guests(rr) := 0
   END;
 change_room(rr1, rr2) =
   PRE
       rr1 \in ROOM \land rr2 \in ROOM \land
       rr1 \neq rr2 \land guests(rr1) > 0 \land
       guests(rr2) = 0 \land
       (rr2 \in single \Rightarrow quests(rr1) = 1)
   THEN
      guests := guests <+
         {(rr1,0), (rr2,guests(rr1))}
   END:
                                                   12
Specification Methods 25.01.2006
```

```
...
nn ← roomquery(rr) =
PRE rr ∈ ROOM
THEN nn := guests(rr)
END;
nn ← vacancies =
BEGIN
nn := card (guests |> {0})
END;
END
```

A Specification Methods 25.01.2006