Department of Computer Science, CUI Lahore Campus

Formal Methods

By

Farooq Ahmad

Sequence

- In this chapter we will study a different type of collection, known as a **sequence**.
- A sequence differs from a set in two principal ways:
  - A sequence is an *ordered* collection of objects.
  - In a sequence, repetitions *are* significant.

Notation

- A sequence is specified by enclosing its members in square brackets. In general terms we could define a particular sequence, *s*, as follows:
  - $s = [\, a, d, f, a, d, d, c \,]$
- A sequence representing a queue of people, say, at a bus-stop could be defined as
  - *queue* = [MICHAEL, VARINDER, ELIZABETH, WINSTON, JUDITH]

Notation

- It is important to note that because a sequence is an *ordered* collection, then, for example:
  - [a, d, f] ≠ [a, f, d]
- The empty sequence is expressed as:
  - [ ]

Elements

- The elements of a sequence are numbered, starting from 1, from left to right. We can refer to a particular element of a sequence by placing the position of the element in brackets. For example, using the above sequences:
  - $s(3)= f$
  - *queue*(4) = WINSTON
  - $s(10)$ is undefined

Sequence operators: Length and Elements operator

The length operator:

- The **len** operator gives us the length of the sequence. Using the above examples:
  - **len** *s* = 7
  - **len** *queue* = 5

The Elements operator:

- The **elems** operator returns a set that contains all the members of the sequence (it therefore removes the duplicates):
  - **elems** *s* = {*a*, *d*, *f*, *c*}
  - **elems** *queue* = {MICHAEL, VARINDER, ELIZABETH, WINSTON, JUDITH}

Head and tail operator

- The *head* (**hd**) operator gives us the first element in the sequence; the *tail* (**tl**) operator gives us a sequence containing all but not the first element:
  - **hd** $s = s(1) = a$
  - **tl** $s = [d, f, a, d, d, c]$
  - **hd** *queue* = MICHAEL
  - **tl** *queue* = [VARINDER, ELIZABETH, WINSTON, JUDITH]

Head, tail and concatenation operator

- The result of both **hd**[] and **tl**[] is undefined. Notice that **hd** returns an element, whereas **tl** returns a sequence.

Concatenation operator:

- The **concatenation** operator (^) operates on two sequences, and returns a sequence that consists of the two sequences joined together:
  - if **first =** [*w*, *e*, *r*, *w*]
  - and **second =** [*t*, *w*, *q*]

  then
  - **first^second =** [*w*, *e*, *r*, *w*, *t*, *w*, *q*]

Override operator

- The **override** operator, †, takes a sequence and gives us a new sequence with a particular element of the old sequence overridden by a new element. The generalized form of this expression is:
  - $s \dagger m$
- where *s* is a sequence and *m* is a **map**

Indices and subsequence operator

- The **inds** operator returns a set of all the indices of the sequence. Thus, using the previous examples:
  - **inds** *s* = {1, 2, 3, 4, 5 , 6, 7}
  - **inds** *queue* = {1, 2, 3, 4, 5}
- A **subsequence** operator is defined to allow us to extract a part of a sequence between two indices. For example, using, *s* above:
  - *subseq*(*s*, 2, 5) = [*d*, *f*, *a*, *d*]
- The language allows us to write this in the following, more convenient, way:
  - *s*(2, … , 5) = [*d*, *f*, *a*, *d*]

Subsequence operators

Invalid!

- $s(1, \ldots ,0) = []$

and

- $s(8, \ldots , 7) = []$

Note:

You should also note that:

- $s(2, \ldots , 2) = [d]$

Comprehension

- [*expression*(*a*) | *a* Є *SomeSet* • *test* (*a*)]
- [*a* | *a* Є {1, … , 20} • *is-odd*(*a*)]
- Often sequence comprehension is used to 'filter' a sequence. For example, if the sequence *s1* were defined as follows:
- *s1* = [2, 3, 4, 7, 9, 11, 6, 7, 8, 14, 39, 45, 3]

and *s2* were defined as

  - *s2* = [*s1*(*i*) | *i* Є **inds** *s1* • *s1*(*i*) > *10*]
- then *s2* would evaluate to the sequence
  - [11, 14, 39, 45].

Reference and reading material

- Formal Software Development From VDM to Java, Chapter # 7: **Sequences**