Chapter 7 - **Sequences**

We will look at

- the idea of and need for a sequence
- how to construct a sequence, different kinds of sequences
- operations on sequences
- examples of use of sequences

The need for sequences

There are times when the properties of sets make them inappropriate to model certain kinds of information. For example, sometimes order is important and it is important to include each occurrence of an element. In sets order is not important and duplicates are ignored i.e.

{1,2,3 } = {3, 2, 1} = { 1, 3, 2, 1}

Sequences are ordered collections of objects.
For example when talking about days of the week, their order is important. We write the sequence using the angle bracket notation.

daysinweek = < sun, mon, tue, wed, thurs, fri, sat >

This could be equivalently displayed without sequence notation as

$\{ 1 \mapsto sun, 2 \mapsto mon, 3 \mapsto tue, 4 \mapsto wed, 5 \mapsto thu, 6 \mapsto fri, 7 \mapsto sat \}$

Thus, a sequence holds not only the elements but also the position they occupy.

The sequence denoted <mon, tue, wed> contains information as follows:

position	element
1	mon
2	tue
3	wed

Note from this, it follows that :

<mon, tue, wed> $\neq$ <wed, tue, mon>
<mon, tue, wed> $\neq$ <mon, tue, wed, mon>

How to construct a sequence

Using the example as above:

DAYS ::= sun| mon| tue| wed| thurs| fri| sat

$$daysGoneBy : seq\ DAYS$$
$$daysGoneBy = \langle mon, tue, wed \rangle$$

More generally (and formally)

$seq\ X == \{ f : \mathbb{N} \pfun X \mid dom\ f = 1 .. \#f \}$

Note the use of the finite function symbol $\pfun$ to indicate that sequences are finite functions. Infinite sequences are not part of standard Z

Special types of sequences

Non-empty sequences
A non-empty sequence is a simple specialisation of sequences. It represents sequences with at least one element. It is written seq_1 and can be defined as :

$seq_1\ X == \{seq\ X \mid \#\ f > 0\}$

Injective Sequences
Sometimes it is important to have an injective (one-to-one) sequence. This will not allow any element in the range to be used more than once in the sequence. The *daysinweek* sequence seen above is an example of an injective function. It is written iseq and is defined as:

$iseq\ X == seq\ X \cap (\mathbb{N} \rightarrowtail X)$

It is precisely the set of finite of sequences over X which contain no repetitions.

Example of use
ACTIVITY ::= plan | code | test | review | deliver

$$Project : seq\ ACTIVITY$$
$$Project = \langle plan, code, deliver \rangle$$

Because a sequence is a special kind of function, all normal functions operations can be used on sequences. ‘dom’ can be used to deliver the set of index numbers for the sequence and ‘ran’ delivers the set of items in the sequence.

Project (1) = plan
Project 2 = code

ran Project = { plan, code, deliver}
dom Project = {1, 2, 3}
Project = 3

Operations on Sequences
Concatenation
The main operator for constructing sequences is concatenation, written _⌢_ where both of its arguments must be sequences. Note that a simple union operators between the two sets containing sequences is not sufficient. Concatenation has the effect of joining two sequences by taking two sequences of the same type and producing a sequence numbered from one to the combined size as follows :

<mon, tue, wed> ⌢ < thu, fri, sat> == <mon, tue, wed, thu, fri, sat>
$\{1 \mapsto mon, 2 \mapsto tue, 3 \mapsto wed\} \frown \{ 1 \mapsto thu, 2 \mapsto fri, 3 \mapsto sat\}$
$== \{ 1 \mapsto mon, 2 \mapsto tue, 3 \mapsto wed, 4 \mapsto thu, 5 \mapsto fri, 6 \mapsto sat\}$

The formal definition for the infix concatenation operation is as follows :

$$[X]$$
$$\_ \frown \_ : seq\ X \times seq\ X \rightarrow seq\ X$$
$$\forall s, t : seq\ X \bullet$$
$$s \frown t = s \cup \{ n : 1..\#t \bullet (n + \#s) \mapsto t\ n \}$$

Reverse
The reverse operation takes a sequence and produces a sequence with the order of the elements reversed. Its type is therefore seq X →seq X.
Example
e.g. rev <mon, tue, wed> = <wed, tue, mon>

Splitting Sequences
Head and Tail
The head of a sequence is its first element. The tail of a sequence the rest of the sequence after you have removed the head.
Examples
head <mon, tue, wed> = mon
(Note that the head operation returns an element of the sequence, not a sequence.)

tail <mon, tue, wed> = <tue, wed>
(Note that tail returns a sequence)

Front and Last

These are similar to head and tail except that the last of a sequence is the last element in a sequence and the front of a sequence is the sequence containing everything except the last element.

Examples

front<mon, tue, wed> = <mon, tue>

last <mon, tue, wed> = wed

More formally, the operations are defined as follows:

$$[X]$$

$$head, last : seq_1\ X \rightarrow X$$

$$tail, front : seq_1\ X \rightarrow seqX$$

$$\forall\ s : seq_1\ X \bullet$$

$$head\ s = s\ 1$$

$$s = \langle head\ s \rangle \frown tail\ s$$

$$last\ s = s\ \#s$$

$$front\ s = (1\ ..\ \#s - 1) \lhd s$$

Distributed Concatenation

Analogous to the distributed union and intersection that we have already seen and used, there is a distributed concatenation operator. It takes a sequence of sequences and 'puts' them all together. It is written $\frown/$

$$\frown/\_ : seq(seqX) \rightarrow seq\ X$$

Example

ss = < <a,b> , <c, d>, <e> , <f,g,h> >

$\frown/ss = \langle a, b, c, d, e, f, g, h \rangle$

Selecting from sequences

There are operations **Extract** ^{1} and **Filter** which are analogous to domain restriction and range restriction on relations and functions. Squash compacts back to a sequence.

Filter is written $\upharpoonright$. Is s is sequence over X and V is a subset of X , then s $\upharpoonright$ V is a sequence which contains just those elements of s which are members of V in the same order as in s.

Example

<a,r,i,t,h,m,e,t,i,c> $\upharpoonright$ <a,e,i,o,u> = <a, i, e, i>

Extract is written ^{1} . Given U as a set of indices and s as a sequence, then U ^{1} is sequence that contains exactly those elements of s that appear at an index in U, in the same order as in s.

Squash takes a finite function defined strictly on the positive integers and compacts it into a sequence, i.e. there are no 'holes' left. (Squash is used in the definition of the extract and filter operations).

Let test = < u, p, c, u, p, c, u, p, c >

then : $\{1,5,3\}$ ^{1} test = <u, c, p>

test $\upharpoonright \{u\}$ = <u, u, u >

Partition/Disjoint

We have already looked at the concept of sets partitioning another set.
Simply, a sequence of sets is disjoint if they have no elements in common(i.e. the distributed intersection operator yields the empty set).

disjoint_ : $\mathbb{P}(seq\ \mathbb{P}X)$

disjoint <A, B> $\Leftrightarrow A \cap B = \{\}$

e.g.
boys, girls, children : $\mathbb{P}\ NAME$
$disjoint < boys, girls>$

A sequence of sets **partition** a set S if they are disjoint and the distributed union of the sequence of sets is S

partition : (seq $\mathbb{P}X$) $\leftrightarrow \mathbb{P}X$

e.g.

<boys, girls> **partition** children