11 SEQUENCES

In Chapter 10 we saw that a function is a kind of binary relation. In this chapter we see that a sequence is a kind of function.

11.1 SEQUENCE

A *sequence* is just a finite function whose domain is drawn from a sequence of non-zero natural numbers. For example

$$weekday = \{ 1 \mapsto mon, 2 \mapsto tue, 3 \mapsto wed, 4 \mapsto thur, 5 \mapsto fri \}$$

The terms in a sequence are ordered by their first components. $1 \mapsto X$ always comes immediately before $2 \mapsto Y$ whatever X and Y are. The first components of pairs in a sequence are often called indices. In a sequence order matters. And repetitions are allowed - but read on.

Since a sequence is just a function we can write

$$weekday(1) = mon$$
$$weekday(5) = fri$$

and

$weekday(6)$ is undefined.

In Z we might define

$$DAY ::= mon \mid wed \mid fri \mid tue \mid thu \mid sat \mid sun$$

then declare

$$weekday : \text{seq } DAY$$
$$weekday = \langle mon, tue, wed, thu, fri \rangle$$

where *seq* introduces a sequence.

$$weekday = \langle mon, tue, wed, thu, fri \rangle$$

represents the sequence of successive days in a week from Monday to Friday. The angle brackets, < and >, mark both ends of a sequence.

11.2 OPERATIONS ON SEQUENCES

Operations on non-empty sequences include

head *s*	the first element of *s*	head *weekday* = *mon*
last *s*	the last element of *s*	last *weekday* = *fri*
front *s*	*s* without its last element	front *weekday* = < *mon, tue, wed, thu* >
tail *s*	*s* without its head element	tail *weekday* = < *tue, wed, thu, fri* >

The concatenation operator, $\frown$, joins two sequences by appending the second sequence onto the end of the first.

$$< sun > \frown weekday \frown < sat > = < sun, mon, tue, wed, thu, fri, sat >$$

The *filter operator*, $\restriction$, creates a new sequence from an existing sequence. For example

$$weekday \restriction < tue, wed, anyday > = < tue, wed >$$

The new sequence contains just those elements of the original sequence, *weekday*, with the order of *weekday* preserved.

The *extraction operator*, $\upharpoonright$, also creates a new sequence from an existing one.

$$\{ 2, 4, 6 \} \upharpoonright weekday = < tue, wed >$$

The new sequence contains only those elements that appear at the given indices. Order is maintained.

The *squash operator* compacts a function whose domain is from non-zero positive integers into a sequence. For example

$$squash \{ 4 \mapsto c, 2 \mapsto b, 6 \mapsto d \} = < b, c, d >$$

A sequence is a kind of function, and a function is a kind of set. So we can use, with care, the usual set operations. For example

$$\# weekday = \#< mon, tue, wed, thu, fri > = 5$$

The empty sequence is represented by both <> and $\varnothing$.

Repetitions can occur in a sequence.

$$\{ 1 \mapsto a, 2 \mapsto b, 3 \mapsto a \} = < a, b, a >$$

But an *iseq* never contains repetitions.

$$\{ 1 \mapsto a, 2 \mapsto b \} = < a, b >$$

EXERCISE 11.1

1 Evaluate

a. head < *h, e, a, d* >
b. last < *l, a, s, t* >
c. front < *f, r, o, n, t* >
d. tail < *t, a, i, l* >
e. < *m, a, x* > ⌢ < *p, a, i, n* >
f. < *f, i, l, t, e, r* > ↾ < *i, t, s* >
g. { *2, 4, 6* } ⇃ < *e, x, t, r, a* >

2 Given

[*CHAR*]

the set of all printable characters, explain the difference between

a. *text* : *seq CHAR* and
b. *text* : *iseq CHAR*

11.3 QUEUES

Aircraft approach an airport in a random pattern. When the airport is busy arriving aircraft are queued by Air Traffic Control (ATC) in a stack before being instructed to make their final approach.

A stack is a fixed circling pattern in which aircraft fly whilst they wait to land. When airports are busy there can be a build up of aircraft held in several vertical stacks. (In this context, a stack is nothing like the data structure known to programmers.)

ATC ensure that aircraft are sequenced for safe separation by controlling their speed and lengths of routings before being turned onto their final approach. On the final approach the Instrument Landing System (ILS) directs aircraft.

ILS is a beam that is aligned to guide aircraft in a straight-line approach to the runway for landing. The ILS beam has two signals, one giving vertical guidance (the guide slope, typically 3°) and the other indicating whether to fly left or right in order to line up with the runway (the localiser).

ATC ensure a safe separation between aircraft on the final approach by controlling their speed, and by considering the size of the aircraft (a large aircraft could leave a turbulent wake which could cause difficulties for a small following aircraft), by considering the time it should take for an aircraft to vacate the runway after landing, and by the prevailing weather and visibility.

Sometimes an arriving aircraft on a final approach may have to abort landing if it cannot land safely. This is done by applying full take-off power and climbing away from the airport. The landing abort procedure is known as a *go around*. Reasons for a go around include a previously landed aircraft still on the runway, debris on the runway, technical problems experienced by the pilot, passengers not being seated and belted, or adverse weather

conditions such as sudden and severe crosswinds. Aircraft that go-around are re-sequenced.

An aircraft arriving with a technical problem, such as being short on fuel, might be positioned by ATC near the head of the queue in a stack. All aircraft must join the end of the queue on final approach since aircraft must be stabilised on speed and direction before landing

So, essentially we have two queues. The queues in the stacks can have aircraft joining the rear end of the queue, or being inserted at a random position in the queue. The queue on the final approach where aircraft join only at the rear of the queue - but they can leave the queue prematurely. We have a look at the Z that specifies the behaviour of the queue on final approach.

Call sign and flight number, e.g. BA0016, uniquely identify aircraft.

Queue - models the queue of aircraft on their final approach to the runway.

$$limit : \mathbb{N}$$

There is a limit to the number of aircraft that can be queued for landing on the final approach.

$$[FLIGHTID]$$

Queue

$$queue : iseq\ FLIGHTID$$

An aircraft is unquely identified by its FLIGHTID.

InitQueue

$$Queue'$$

$$queue' = \langle\rangle$$

Initially, there are no aircraft in the queue for landing on the final approach.

Join

$$\Delta Queue$$
$$aircraft? : FLIGHTID$$

$$\#queue < limit$$
$$aircraft? \notin ran\ queue \land queue' = queue \frown \langle aircraft? \rangle$$

Of course you cannot have aircraft duplicated in the landing queue.

```
Land
ΔQueue
queue ≠ <> ∧ queue' = tail queue
```

An aircraft leaves the front of the queue when it vacates the runway on which it has landed.

```
GoAround
ΔQueue
aircraft? : FLIGHTID
aircraft? ∈ ran queue ∧ queue' = queue \ ⟨aircraft?⟩
```

An aircraft can prematurely leave the queue when on the final approach.

EXERCISE 11.2

1 Normally, aircraft join the queue in a stack at the tail end, and, by reducing height on each circuit, reach the front of the queue where they are despatched onto the final approach. Sometimes aircraft may be inserted into a random position in the queue. Write the Z specification that describes the behaviour of aircraft queues in the stack.

2 Four stacks of queued aircraft are maintained at a busy airport. Write the Z specification that might describe the behaviour of the aircraft in the four stacks. You might like to declare

[*FLIGHTID, STACKID*]

and define

queue : STACKID ⇸ iseq FLIGHTID

REVIEW

We have seen that a sequence is a finite function whose domain is drawn from a sequence of positive integers. We have looked at some operations on a sequence: *head*, *last*, *front tail, squash* and concatentation. We have seen how a sequence can model the behaviour of queues.

BIBLIOGRAPHY

ba.com accessed 16 Mar 2014
Spivey JM *The Z Notation 2nd* Ed Prentice-Hall International 1992 page 115
Barden, Stepney and Cooper *Z in Practice* Prentice Hall 1994 page 379
Jacky J *The Way of Z* Cambridge University Press 1997 pp 93, 302..304