FMSE: Lecture 2

The Specification Language Z:
Operations, Sequences

Goals of Lecture 2

At the end of this lecture you should be able to:

- specify the initial state of a system
- specify operations using schemas
- use sequences and their operations in schemas

Previous Lecture

In the previous lecture we looked at:

- axiomatic descriptions
- types
- schemas: declarations, invariants

The Library System

$$maxloan : \mathbb{N}_1$$

$$[TITLE, COPY, READER]$$

$$title : COPY \longrightarrow TITLE$$

Library

$$collection : \mathbb{P}\, COPY$$
$$readers : \mathbb{P}\, READER$$
$$issued : COPY \twoheadrightarrow\!\!\!\!\!+\; READER$$

$$\mathrm{dom}\, issued \subseteq collection$$
$$\mathrm{ran}\, issued \subseteq readers$$
$$\forall\, r : \mathrm{ran}\, issued \bullet \#\{c : COPY \mid issued(c) = r\} \leq maxloan$$

Initialisation

We specify a scheme *Init* representing the initial state of the library.

- assume that initially the collection is empty, and there are no registrated readers
- other choices are possible, e.g. starting with a given collection of books
- all the invariants should hold in the initial state

First version of Init

Init

$$collection : \mathbb{P}\, COPY$$
$$readers : \mathbb{P}\, READER$$
$$issued : COPY \rightarrowtail\!\!\!\!\!\!+\; READER$$

$$collection = \varnothing$$
$$readers = \varnothing$$
$$issued = \varnothing$$

but we can do it in a nicer way...

Schema Import

a previously defined schema (called e.g. *State* can be used in the definition of another schema:

```
┌─ NewSchema ──────────────────────
│ State
│ ...
├──────────
│ ...
└──────────────────────────────────
```

Semantics: all state variables and invariants of schema *State* become part of *NewSchema*

A schema import can expanded, i.e. all imported variables and invariants are written out (the tool Z/Eves can do this for you).

Init with Schema Import

Init

Library

$collection = \varnothing$

$readers = \varnothing$

$issued = \varnothing$

but it can be more concise...

Init: Final Version

Init

Library

$collection = \varnothing$

$readers = \varnothing$

The value of *issued* can be deduced with the help of the imported invariants!

Operations on the Library

- issue a copy to a reader
- return a book by a reader
- add/remove a copy to/from the collection
- enquire about the books a reader has on loan
- enquire which reader has a certain copy
- register/cancel a reader
- enquire which titles are in the collection
- enquire for a title which copies are available
- remove a reader who has disappeared, together with the books he has borrowed

Issueing a Copy

Issue

$$\Delta Library$$
$$r? : READER$$
$$c? : COPY$$

$$r? \in readers$$
$$c? \in collection \setminus (\operatorname{dom} issued)$$
$$\#\{c : COPY \mid issued(c) = r?\} < maxloan$$
$$issued' = issued \cup \{(c?, r?)\}$$
$$collection' = collection$$
$$readers' = readers$$

Conventions for Operations

- an unprimed (no ’) variable: before the operation (old)
- a primed (’) variable: after the operation (new)

examples:

$\mathit{collection}' = \mathit{collection}$

$\mathit{issued}' = \mathit{issued} \cup \{(c?, r?)\}$

Note that = is equality and not assignment!

Primed Schema Import

If *State* is a schema, we can also import *State'* : all state variables in declarations and invariants are primed.

We do this typically in operations:

OperationOnState

State
State'

...

Two Shorthands

Δ*State* is shorthand for the import of both *State* and *State'*:

```
── OperationOnState ──────────────────
│ ΔState
├────────
│ ...
└──────────────────────────────────────
```

If a state variable v remains the same, we have to write $v' = v$.
What if all state variables reamain the same (e.g. in a query)?

$\Xi State$ is like $\Delta State$, but all state variables remain the same (so we import $v' = v$ for all state variables v:

OperationOnState

$\Xi State$

$\ldots$

Input and Output

- input variables: end with ”?”,
  e.g. *input?*
- output variables: end with ”!”,
  e.g. *output!*
- need to be declared above the line of an operation schema
- can be constrained by predicates under the line, e.g.
  $r? \in readers$
  $c? \in collection \setminus (\mathrm{dom}\, issued)$

Set Updates

Updates on a set set:

- adding an element new:

  $set' = set \cup \{new\}$

- adding a set s:

  $set \cup s$

- removing an element out:

  $set' = set \setminus \{out\}$

- removing a set s:

  $set' = set \setminus s$

Function updates

Updates on a function f:

- removing a pair (x, y):
  $f' = f \setminus \{(x, y)\}$
- adding (x, y) for $x \notin \operatorname{dom} f$:
  $f' = f \cup \{(x, y)\}$
- changing the value of $f(x)$ into y:
  $f' = f \oplus \{(x, y)\}$
- if f and g are two functions of the same type, then $f \oplus g$ behaves like g on $\operatorname{dom} g$, and like f on $(\operatorname{dom} f) \setminus (\operatorname{dom} g)$

An Enquiry

Which are the books that a reader has on loan?

OnLoan

$$\Xi Library$$
$$r? : READER$$
$$cc! : \mathbb{P}\, COPY$$

$$r? \in readers$$
$$cc! = \{c : COPY \mid issued(c) = r?\}$$

Another Inquiry

Which copies are available for a certain title?

Available

$$\Xi Library$$
$$t? : TITLE$$
$$cc! : \mathbb{P}\, COPY$$

$$cc! = \{c : COPY \mid c \in collection \setminus (\operatorname{dom} issued) \\ \wedge\ title(c) = t?\}$$

Removing a reader

A dubious reader cannot be traced anymore. Remove the reader and the books that he has in his posession.

Remove

$$\Delta Library$$
$$r? : readers$$

$$readers' = readers \setminus \{r?\}$$
$$collection' = collection \setminus \{c : COPY \mid issued(c) = r?\}$$
$$issued' = issued \setminus \{c : COPY, r : READER \mid r = r?\}$$

Sequences

If the order of elements is important we can use *sequences*

- sequences are written using $\langle$ and $\rangle$, e.g. $colorq = \langle red, yellow, green, red \rangle$, and $empty = \langle\rangle$
- a sequence s with elements S has type $\mathrm{seq}\, S$, so $colorq, empty : \mathrm{seq}\, COLOR$
- formally, a sequence s of type $\mathrm{seq}\, S$ is a function from $1..N$ to S for some N, with $\mathrm{dom}\, s = 1..N$ and $\mathrm{ran}\, s$ is the set of elements in the sequence
- we can write $colorq(3) = green$, $\#colorq = 4$, $\#empty = 0$

Note that sequences s, s' are sets of pairs (*sequencenr*, *element*). But in general $s \cup s'$ and $s \cap s'$ are not sequences!

Concatenation of sequences

Suppose $s = \langle 3, 7, 1, 2 \rangle$ and $t = \langle 5, 9 \rangle$.

Then $s \frown t = \langle 3, 7, 1, 2, 5, 9 \rangle$.

Add element 8 to front of s:

$\langle 8 \rangle \frown s$ (and not $8 \frown s$!)

Add element 8 to back of s:

$s \frown \langle 8 \rangle$ (and not $s \frown 8$!)

Other Sequence Operations

Let $s = \langle 3, 7, 1, 2 \rangle$, then:

- $head\ s = 3$
- $tail\ s = \langle 7, 1, 2 \rangle$
- $last\ s = 2$
- $front\ s = \langle 3, 7, 1 \rangle$

Two useful sequence types:
$\mathrm{seq}_1\, X$: non-empty sequences with elements in X
$\mathrm{iseq}\, X$: injective sequence (an element cannot occur more than once)

A Deletion Operation

Specify an operation that deletes an element $el?$ from an injective sequence of integers.

Delete

$$s, s' : \mathrm{iseq}\,\mathbb{Z}$$
$$el? : \mathbb{Z}$$

$$s = l \frown \langle el? \rangle \frown r$$
$$s' = l \frown r$$

What if $el? \notin \mathrm{ran}\, s$? This is treated in the next lecture...