1 Overview of assignment

This assignment extends the tutorial example of a simple library (see 2.4.2). The extensions are:

1. addition of a borrowing limit;
2. addition of a reservation capability

The assignment archive Library.zip contains the following components:

Library_ctx: context for Library machine;
List_ctx: list context for LibraryR2

Each extension should be modelled as a refinement.

Note To create a refinement of a machine it is best to use the Event-B Explorer in Rodin; don’t create the refinement from scratch by hand.

In the Event-B Explorer right click on the machine you want to refine and then choose Refine from the options. Fill in the name of the refinement machine and Rodin will create a base for your refinement with all events being extended. In some cases you will not want an extension, for example when you want to modify the guards of an event, not simply add more guards. In such cases you will want to turn off extension for such events.

2 Refinements

2.1 LibraryR0: Borrowing limit

LibraryR0 is a very simple refinement that use the constant maxloan to set a borrowing limit on number of books that a member can borrow at any time. The constant does not have a value; it is simply of type N1. For animation with AnimB a value would be required in the AnimB values for Library_ctx.

You should be try to discharge the proof obligations, but some are a little tricky. They probably will not be all auto-proved. You will have to use proof by cases (the dc button in the proof control) as most of the lemmas will have \( m \in \text{members} \) and there will be two cases: \( m = \text{member} \) and \( m \neq \text{member} \).
2.2 Adding book reservation

Book reservation is concerned with reserving a book that is currently borrowed.

The following constraints apply to reservation of book by member:

**Person reserving must be a member** same as constraint for borrowing;

**Book being reserved must be currently onloan**

**Book must not be currently reserved for member** a book may be reserved at most once for any particular member;

**Books may have multiple reservations** by different members.

**Reservations can be cancelled** by the member that requested the reservation.

Modelling of reservation should be done in two stages represented by LibraryR1 and LibraryR2.

LibraryR1 refines Library, and

LibraryR2 refines LibraryR1

2.3 LibraryR1

LibraryR1 should refine LibraryR0 and model reservation with no priority. That is, when a book that has been reserved is returned it can be borrowed by any one of the members who reserved that book.

2.4 LibraryR2

LibraryR2 should refine LibraryR1 and reservations should now satisfy the following:

1. Reservations are queued. That is reservations for the same book are queued in the order in which the reservation requests were made.

2. When a reserved book is returned it is then available for the first member on the queue to borrow. The book is not available for general borrowing until all on the reservation queue have borrowed it.

3. A member who has requested reservation of a book can cancel their reservation, in which case the queue must “close up”.

This refinement will be a data refinement.

2.4.1 The List context

List_ctx contains a list algebra that you should use for the book reservation queue. Lists are modelled as functions, so \( l(i) \) is the i-th element of the list \( l \). There are two types of list models provided: LIST ordinary lists and ILIST injective lists, which are lists in which there are no repeated values. The algebra provides you with the following operations on lists:

**APPEND** \( APPEND(l \mapsto m) \) appends \( m \) to the end of the list \( l \), so you could write \( l := APPEND(l \mapsto m) \);

**DEQUEUE** \( DEQUEUE(l) \) removes the head (first) item on the list, for example \( l := DEQUEUE(l) \);
JOIN $JOIN(l_1 \mapsto l_2)$ joins the two lists $l_1$ and $l_2$, for example

$l := JOIN(l_1 \mapsto l_2)$;

DELETE $DELETE(l \mapsto i)$ deletes the i-th element of the list $l$, for example

$l := DELETE(l \mapsto i)$.

IDELETE a slightly simpler version of DELETE that can be used on injective lists. IDELETE deletes the member $m$ from the injective list $l$

$l := IDELETE(l \mapsto m)$.

The context provides quantifiers for determining $dom$, $ran$ and indexing of list combinations.

2.4.2 What you should do

First, download the archive, Library.zip, containing the contexts Library_ctx, List_ctx and the machine Library.

Partial versions of the refinements are not included as it is best if you use Rodin to generate the initial refinements

Create and complete LibraryR0 to introduce a borrowing limit.

Create and modify LibraryR1 to add simple reservation and cancelling of a reservation.

Create and modify LibraryR2 to add a priority queue for reservation

Discharge or at least review the proof obligations it can be expected that the POs will be generally difficult, but they should be reviewed to detect errors in your models.
CONTEXT Library_ctx

SETS

MEMBER
BOOK

CONSTANTS

maxloan

AXIOMS

axm1: finite(MEMBER)
axm2: finite(BOOK)
axm3: maxloan ∈ N_1

END
MACHINE Library
SEES Library_ctx

VARIABLES

books books contained in the library
members members of the library
borrowed books borrowed by each member

INVARIANTS

inv1: books ⊆ BOOK
inv2: members ⊆ MEMBER
inv3: borrowed ∈ books → members

EVENTS

Initialisation

begin act1: books := ∅
act2: members := ∅
act3: borrowed := ∅
end

Event NewMember ≜ Add a new member of the library

any member
when grd1: member ∈ MEMBER \ members
then act1: members := members ∪ {member}
end

Event AddBook ≜ Add a book to the library; this could be a new book for the library or an extra copy

any book
when grd1: book ∈ BOOK \ books
then act1: books := books ∪ {book}
Event \textit{Borrow} \equiv \text{A member borrows a book}

\textbf{any} book
\textbf{member}
when \textit{grd1}: \text{book} \in \text{books}
\textit{grd2}: \text{member} \in \text{members}
\textit{grd3}: \text{book} \notin \text{dom(borrowed)}
then \textit{act1}: \text{borrowed(book)} := \text{member}
end

Event \textit{Return} \equiv \text{A member returns a book}

\textbf{any} book
when \textit{grd1}: \text{book} \in \text{dom(borrowed)}
then \textit{act1}: \text{borrowed} := \{\text{book}\} \triangle \text{borrowed}
end

END
**CONTEXT**  List.ctx

**EXTENDS**  Library.ctx

**CONSTANTS**

- **LENGTH**  Finite limit on length of lists
- **LIST**  Set of lists
- **ILIST**  Set of injective lists
- **JOIN**  List concatenation operator
- **APPEND**  Append an item to tail of list
- **IAPPEND**  Append maintaining injectivity
- **DEQUEUE**  Delete head of list
- **DELETE**  Delete an element from any position of a list
- **IDELETE**  Delete an element from an injective list

**AXIOMS**

\begin{align*}
\text{axm1:} & \quad \text{LENGTH} \in \mathbb{N} \\
\text{axm2:} & \quad \text{LIST} = \{ f : f \in 1..\text{LENGTH} \rightarrow \text{MEMBER} \land \text{dom}(f) = 1..\text{card}(f)|f \} \\
\text{thm1:} & \quad \emptyset \in \text{LIST} \\
\text{axm3:} & \quad \text{ILIST} = \{ f : f \in \text{LIST} \land f \in 1..\text{LENGTH} \rightarrow \text{MEMBER}|f \} \\
\text{thm2:} & \quad \text{ILIST} \subset \text{LIST} \\
\text{axm4:} & \quad \emptyset \in \text{ILIST} \\
\text{thm3:} & \quad \forall l \in \text{ILIST} \Leftrightarrow l \in \text{LIST} \land l \in 1..\text{LENGTH} \Leftrightarrow \text{MEMBER} \\
\text{thm4:} & \quad \forall l \in \text{LIST} \Rightarrow \text{ran}(l) = l[\text{dom}(l)] \\
\text{thm5:} & \quad \forall l \in \text{LIST} \Rightarrow \text{ran}(l) = l[1..\text{card}(l)] \\
\text{axm5:} & \quad \text{JOIN} \in (\text{LIST} \times \text{LIST}) \rightarrow \text{LIST} \\
\text{thm6:} & \quad \text{dom}(\text{JOIN}) = \text{LIST} \times \text{LIST} \\
\text{axm6:} & \quad \forall l1, l2. l1 \in \text{LIST} \land l2 \in \text{LIST} \\
& \Rightarrow \text{dom}(\text{JOIN}(l1 \leftrightarrow l2)) = 1..\text{card}(l1) + \text{card}(l2)
\end{align*}
axm7: \( \forall l_1, l_2, i : l_1 \in \text{LIST} \land l_2 \in \text{LIST} \land i \in \text{dom} (\text{JOIN}(l_1 \mapsto l_2)) \Rightarrow \)
\( (i \in 1 \ldots \text{card}(l_1) \Rightarrow \text{JOIN}(l_1 \mapsto l_2)(i) = l_1(i)) \)
\( \land \)
\( (i - \text{card}(l_1) \in 1 \ldots \text{card}(l_2) \Rightarrow \text{JOIN}(l_1 \mapsto l_2)(i) = l_2(i - \text{card}(l_1))) \)

thm7: \( \forall l \in \text{LIST} \Rightarrow \text{JOIN}(\emptyset \mapsto l) = l \)

thm8: \( \forall l \in \text{LIST} \Rightarrow \text{JOIN}(\emptyset \mapsto l) = l \)

thm9: \( \forall l_1, l_2 : l_1 \in \text{LIST} \land l_2 \in \text{LIST} \land \text{ran}(l_1) \cap \text{ran}(l_2) = \emptyset \Rightarrow \text{JOIN}(l_1 \mapsto l_2) \in \text{ILIST} \)

thm10: \( \forall l_1, l_2 : l_1 \in \text{LIST} \land l_2 \in \text{LIST} \Rightarrow \text{ran}(\text{JOIN}(l_1 \mapsto l_2)) = \text{ran}(l_1) \cup \text{ran}(l_2) \)

axm8: \( \text{APPEND} \in (\text{LIST} \times \text{MEMBER}) \rightarrow \text{LIST} \)

thm12: \( \text{dom}(\text{APPEND}) = \text{LIST} \times \text{MEMBER} \)

axm9: \( \forall l, m : l \in \text{LIST} \Rightarrow \text{dom}(\text{APPEND}(l \mapsto m)) = 1 \ldots \text{card}(l) + 1 \)

axm10: \( \forall l, m, i : l \in \text{LIST} \land i \in \text{dom}(\text{APPEND}(l \mapsto m)) \Rightarrow \)
\( (i \in \text{dom}(l) \Rightarrow \text{APPEND}(l \mapsto m)(i) = l(i)) \)
\( \land \)
\( (i = \text{card}(l) + 1 \Rightarrow \text{APPEND}(l \mapsto m)(i) = m) \)

thm13: \( \forall l, m : l \in \text{LIST} \land m \in \text{MEMBER} \Rightarrow \text{ran}(\text{APPEND}(l \mapsto m)) = \text{ran}(l) \cup \{m\} \)

thm14: \( \forall l, m : l \in \text{LIST} \land m \in \text{MEMBER} \Rightarrow \text{card}(\text{APPEND}(l \mapsto m)) = \text{card}(l) + 1 \)

thm15: \( \forall l, m : l \in \text{ILIST} \land m \in \text{MEMBER} \land m \notin \text{ran}(l) \Rightarrow \text{APPEND}(l \mapsto m) \in \text{ILIST} \)

axm13: \( \text{IAPPEND} \in (\text{ILIST} \times \text{MEMBER}) \rightarrow \text{ILIST} \)

thm16: \( \text{dom}(\text{IAPPEND}) = \text{ILIST} \times \text{MEMBER} \)

axm14: \( \forall l, m : l \in \text{LIST} \Rightarrow \text{dom}(\text{IAPPEND}(l \mapsto m)) = \text{dom}(\text{APPEND}(l \mapsto m)) \)
axm15: \[ \forall l, m \in I\text{LIST} \land m \notin \text{ran}(l) \Rightarrow I\text{APPEND}(l \mapsto m) = \text{APPEND}(l \mapsto m) \]

thm17: \[ \forall l, m \in I\text{LIST} \land m \in \text{MEMBER} \Rightarrow \text{card}(\text{APPEND}(l \mapsto m)) = \text{card}(l) + 1 \]

axm16: \[ \textsc{DEQUEUE} \in \text{LIST} \to \text{LIST} \]

thm18: \[ \text{dom}(\textsc{DEQUEUE}) = \text{LIST} \]

axm17: \[ \forall l \in \text{LIST} \land l \neq \emptyset \Rightarrow \text{dom}(\textsc{DEQUEUE}(l)) = 1 \ldots \text{card}(l) - 1 \]

axm18: \[ \forall l, i \in \text{LIST} \land l \neq \emptyset \land i \in 1 \ldots \text{card}(l) - 1 \Rightarrow \textsc{DEQUEUE}(l)(i) = l(i + 1) \]

thm19: \[ \forall l \in \text{LIST} \land l \neq \emptyset \Rightarrow \textsc{DEQUEUE}(l) \in \text{LIST} \]

thm20: \[ \forall l \in \text{LIST} \land l \neq \emptyset \Rightarrow \text{ran}(\textsc{DEQUEUE}(l)) = \text{ran}(l) \setminus \{l(1)\} \]

axm19: \[ \textsc{DELETE} \in (\text{LIST} \times (1 \ldots \text{LENGTH})) \to \text{LIST} \]

thm21: \[ \text{dom}(\textsc{DELETE}) = \text{LIST} \times (1 \ldots \text{LENGTH}) \]

axm20: \[ \forall l, i \in \text{LIST} \land i \in \text{dom}(l) \Rightarrow l \mapsto i \in \text{dom}(\textsc{DELETE}) \]

axm21: \[ \forall l, i \in \text{LIST} \land i \in \text{dom}(l) \Rightarrow \text{dom}(\textsc{DELETE}(l \mapsto i)) = 1 \ldots \text{card}(l) - 1 \]

thm22: \[ \forall l, i, j \in \text{LIST} \land i \in \text{dom}(l) \land j \in 1 \ldots \text{card}(l) - 1 \Rightarrow j \in \text{dom}(\textsc{DELETE}(l \mapsto i)) \]

axm22: \[ \forall l, i, j \in \text{LIST} \land i \in \text{dom}(l) \land j \in 1 \ldots i - 1 \Rightarrow \textsc{DELETE}(l \mapsto i)(j) = l(j) \]

axm23: \[ \forall l, i, j \in \text{LIST} \land i \in \text{dom}(l) \land j \in i \ldots \text{card}(l) - 1 \Rightarrow \textsc{DELETE}(l \mapsto i)(j) = l(j + 1) \]

thm23: \[ \forall l, i \in \text{LIST} \land i \in \text{dom}(l) \Rightarrow \text{card}(\textsc{DELETE}(l \mapsto i)) = \text{card}(l) - 1 \]

thm24: \[ \forall l, i \in \text{LIST} \land i \in \text{dom}(l) \Rightarrow \text{ran}(\textsc{DELETE}(l \mapsto i)) = \text{ran}(l) \setminus \{l(i)\} \]
\textbf{thm25}: \ \forall l, i \in \text{LIST} \ \& \ i \in \text{dom}(l) \\
\Rightarrow \\
\text{card}(\text{DELETE}(l \mapsto i)) = \text{card}(l) - 1

\textbf{axm24}: \ IDELETE \in (\text{ILIST} \times \text{MEMBER}) \rightarrow \text{ILIST}

\textbf{axm25}: \ \text{dom}(\text{DELETE}) = \text{ILIST} \times \text{MEMBER}

\textbf{axm26}: \ \forall l, m \in \text{ILIST} \ \& \ m \in \text{MEMBER} \ \& \ m \in \text{ran}(l) \\
\Rightarrow \\
l \mapsto m \in \text{dom}(\text{DELETE})

\textbf{axm27}: \ \forall l, m \in \text{ILIST} \ \& \ m \in \text{ran}(l) \\
\Rightarrow \\
\text{IDELETE}(l \mapsto m) = \text{DELETE}(l \mapsto l^{-1}(m))

\textbf{thm26}: \ \forall l, m \in \text{ILIST} \ \& \ m \in \text{ran}(l) \\
\Rightarrow \\
\text{dom}(\text{IDELETE}(l \mapsto m)) = 1..\text{card}(l) - 1

\textbf{axm28}: \ \forall l, m \in \text{ILIST} \ \& \ m \in \text{ran}(l) \\
\Rightarrow \\
\text{card}(\text{IDELETE}(l \mapsto m)) = \text{card}(l) - 1

\textbf{thm27}: \ \forall l, m \in \text{ILIST} \ \& \ m \in \text{ran}(l) \\
\Rightarrow \\
\text{ran}(\text{IDELETE}(l \mapsto m)) = \text{ran}(l) \setminus \{m\}

\textbf{thm28}: \ \forall l \in \text{LIST} \ \& \ l \neq \emptyset \\
\Rightarrow \\
\text{DEQUEUE}(l) = \text{DELETE}(l \mapsto 1)

\textbf{thm30}: \ \forall l \in \text{LIST} \ \& \ l \neq \emptyset \\
\Rightarrow \\
\text{card}(\text{DEQUEUE}(l)) = \text{card}(l) - 1

\textbf{thm31}: \ \forall l, i \in \text{ILIST} \ \& \ i \in \text{dom}(l) \\
\Rightarrow \\
\text{DEQUEUE}(l) \in \text{ILIST}

\textbf{thm32}: \ \forall l, i \in \text{ILIST} \ \& \ i \in \text{dom}(l) \\
\Rightarrow \\
\text{DELETE}(l \mapsto i) \in \text{ILIST}

\textbf{thm33}: \ \forall l, m \in \text{LIST} \ \& \ m \in \text{MEMBER} \\
\Rightarrow \\
\text{DELETE}(\text{APPEND}(l \mapsto m) \mapsto \text{card}(l) + 1) = l

\textbf{END}