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 $\text{maxloan}$ 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 $\mathbb{N}_1$. 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.

A book cannot be reserved by the current borrower of the 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 \( \text{APPEND}(l \mapsto m) \) appends \( m \) to the end of the list \( l \), so you could write \( l := \text{APPEND}(l \mapsto m); \)
DEQUEUE \( \text{DEQUEUE}(l) \) removes the head (first) item on the list, for example
\[
l := \text{DEQUEUE}(l);
\]

JOIN \( \text{JOIN}(l_1 \mapsto l_2) \) joins the two lists \( l_1 \) and \( l_2 \), for example
\[
l := \text{JOIN}(l_1 \mapsto l_2);
\]

DELETE \( \text{DELETE}(l \mapsto i) \) deletes the \( i \)-th element of the list \( l \), for example
\[
l := \text{DELETE}(l \mapsto i);
\]

IDELETE a slightly simpler version of DELETE that can be used on injective lists. IDELETE deletes the member \( m \) from the injective list \( l \)
\[
l := \text{IDELETE}(l \mapsto m).
\]

The context provides quantifiers for determining \( \text{dom}, \text{ran} \) 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,
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
where
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
where
grd1: book ∈ BOOK \ books
then
act1: books := books ∪ {book}
end
**Event**  \( \text{Borrow} \triangleq \)

A member borrows a book

any

\( \text{book} \)

\( \text{member} \)

where

\( \text{grd1}: \text{book} \in \text{books} \)

\( \text{grd2}: \text{member} \in \text{members} \)

\( \text{grd3}: \text{book} \notin \text{dom}(\text{borrowed}) \)

then

\( \text{act1}: \text{borrowed}(\text{book}) := \text{member} \)

end

**Event**  \( \text{Return} \triangleq \)

A member returns a book

any

\( \text{book} \)

where

\( \text{grd1}: \text{book} \in \text{dom}(\text{borrowed}) \)

then

\( \text{act1}: \text{borrowed} := \{ \text{book} \} \cup \text{borrowed} \)

end

END
CONTEXT List_ctx
This context presents a small theory of lists.
Lists might also be called sequences.
Both injective and non-injective lists will be modelled.
All elements of an injective list are distinct.

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

axm1: LENGTH ∈ \mathbb{N}
axm2: LIST = \{ l | l ∈ \mathbb{N} \rightarrow MEMBER ∧ \text{dom}(l) = 1 .. \text{card}(l) \}
axm60: finite(LIST)
axm3: \emptyset ∈ LIST
axm4: ILIST = \{ l | l ∈ LIST ∧ l ∈ \mathbb{N} \rightarrow MEMBER \}
axm5: ILIST ⊂ LIST
axm6: \emptyset ∈ ILIST
axm7: \forall l1 · l1 ∈ ILIST ⇒ l1 ∈ LIST ∧ l1 ∈ 1 .. LENGTH ⇒ MEMBER
axm8: \forall l1 · l1 ∈ LIST ⇒ \text{ran}(l1) = l1[\text{dom}(l1)]
axm9: \forall l1 · l1 ∈ LIST ⇒ \text{ran}(l1) = l1[1 .. \text{card}(l1)]
axm10: \forall l1 · l1 ∈ LIST ∧ \text{card}(\text{ran}(l1)) = \text{card}(\text{dom}(l1)) ⇒ l1 ∈ ILIST
axm11: JOIN ∈ (LIST × LIST) → LIST
axm12: \text{dom}(JOIN) = LIST × LIST
axm13: \forall l1, l2 · l1 ∈ LIST ∧ l2 ∈ LIST
⇒ \text{dom}(JOIN(l1 ↦ l2)) = 1 .. \text{card}(l1) + \text{card}(l2)
axm14: \forall l1, l2, i · l1 ∈ LIST ∧ l2 ∈ LIST ∧ i ∈ \text{dom}(JOIN(l1 ↦ l2))
⇒ (i ∈ 1 .. \text{card}(l1) ⇒ JOIN(l1 ↦ l2)(i) = l1(i))
∧ (i − \text{card}(l1) ∈ 1 .. \text{card}(l2) ⇒ JOIN(l1 ↦ l2)(i) = l2(i − \text{card}(l1)))
\textbf{axm15:} \( \forall l \cdot l \in \text{LIST} \)  
\( \Rightarrow \)  
\( \text{JOIN}(l \mapsto \varnothing) = l \) 

\textbf{axm16:} \( \forall l \cdot l \in \text{LIST} \)  
\( \Rightarrow \)  
\( \text{JOIN}(\varnothing \mapsto l) = l \) 

\textbf{axm17:} \( \forall l_1, l_2 \cdot l_1 \in \text{ILIST} \land l_2 \in \text{ILIST} \land \text{ran}(l_1) \cap \text{ran}(l_2) = \varnothing \)  
\( \Rightarrow \)  
\( \text{JOIN}(l_1 \mapsto l_2) \in \text{ILIST} \) 

\textbf{axm18:} \( \forall l_1, l_2 \cdot l_1 \in \text{LIST} \land l_2 \in \text{LIST} \)  
\( \Rightarrow \)  
\( \text{ran}(\text{JOIN}(l_1 \mapsto l_2)) = \text{ran}(l_1) \cup \text{ran}(l_2) \) 

\textbf{axm19:} \( \forall l_1, l_2 \cdot l_1 \in \text{LIST} \land l_2 \in \text{LIST} \)  
\( \Rightarrow \)  
\( \text{card}(\text{JOIN}(l_1 \mapsto l_2)) = \text{card}(l_1) + \text{card}(l_2) \) 

\textbf{axm20:} \( \text{APPEND} \in (\text{LIST} \times \text{MEMBER}) \rightarrow \text{LIST} \) 

\textbf{axm21:} \( \text{dom}(\text{APPEND}) = \text{LIST} \times \text{MEMBER} \) 

\textbf{axm22:} \( \forall l, m \cdot l \in \text{LIST} \)  
\( \Rightarrow \)  
\( \text{dom}(\text{APPEND}(l \mapsto m)) = 1 \ldots \text{card}(l) + 1 \) 

\textbf{axm23:} \( \forall l, m, i \cdot l \in \text{LIST} \land i \in \text{dom}(\text{APPEND}(l \mapsto m)) \)  
\( \Rightarrow \)  
\( (i \in \text{dom}(l) \Rightarrow \text{APPEND}(l \mapsto m)(i) = l(i)) \)  
\&  
\( (i = \text{card}(l) + 1 \Rightarrow \text{APPEND}(l \mapsto m)(i) = m) \) 

\textbf{axm24:} \( \forall l, m \cdot l \in \text{LIST} \land m \in \text{MEMBER} \)  
\( \Rightarrow \)  
\( \text{ran}(\text{APPEND}(l \mapsto m)) = \text{ran}(l) \cup \{m\} \) 

\textbf{axm25:} \( \forall l, m \cdot l \in \text{LIST} \land m \in \text{MEMBER} \)  
\( \Rightarrow \)  
\( \text{card}(\text{APPEND}(l \mapsto m)) = \text{card}(l) + 1 \) 

\textbf{axm26:} \( \forall l, m \cdot l \in \text{ILIST} \land m \in \text{MEMBER} \land m \notin \text{ran}(l) \)  
\( \Rightarrow \)  
\( \text{APPEND}(l \mapsto m) \in \text{ILIST} \) 

\textbf{axm27:} \( \text{IAPPEND} \in (\text{ILIST} \times \text{MEMBER}) \rightarrow \text{ILIST} \) 

\textbf{axm28:} \( \text{dom}(\text{IAPPEND}) = \text{ILIST} \times \text{MEMBER} \) 

\textbf{axm29:} \( \forall l, m \cdot l \in \text{LIST} \)  
\( \Rightarrow \)  
\( \text{dom}(\text{IAPPEND}(l \mapsto m)) = \text{dom}(\text{APPEND}(l \mapsto m)) \) 

\textbf{axm30:} \( \forall l, m \cdot l \in \text{ILIST} \land m \notin \text{ran}(l) \)  
\( \Rightarrow \)  
\( \text{IAPPEND}(l \mapsto m) = \text{APPEND}(l \mapsto m) \) 

\textbf{axm31:} \( \forall l, m \cdot l \in \text{LIST} \land m \in \text{MEMBER} \)  
\( \Rightarrow \)  
\( \text{card}(\text{APPEND}(l \mapsto m)) = \text{card}(l) + 1 \)
axm32: \( \text{DEQUEUE} \in \text{LIST} \rightarrow \text{LIST} \)

axm33: \( \text{dom}(\text{DEQUEUE}) = \text{LIST} \)

axm34: \( \forall l \cdot l \in \text{LIST} \land l \neq \emptyset \Rightarrow \text{dom}(\text{DEQUEUE}(l)) = 1 .. \text{card}(l) - 1 \)

axm35: \( \forall l, i \cdot l \in \text{LIST} \land l \neq \emptyset \land i \in 1 .. \text{card}(l) - 1 \Rightarrow \text{DEQUEUE}(l)(i) = l(i + 1) \)

axm36: \( \forall l \cdot l \in \text{ILIST} \land l \neq \emptyset \Rightarrow \text{DEQUEUE}(l) \in \text{ILIST} \)

axm37: \( \forall l \cdot l \in \text{LIST} \land l \neq \emptyset \Rightarrow \text{ran}(\text{DEQUEUE}(l)) = \text{ran}(l) \setminus \{l(1)\} \)

axm38: \( \text{DELETE} \in (\text{LIST} \times (1 .. \text{LENGTH})) \rightarrow \text{LIST} \)

axm39: \( \text{dom}(\text{DELETE}) \subseteq \text{LIST} \times (1 .. \text{LENGTH}) \)

axm40: \( \forall l, i \cdot l \in \text{LIST} \land i \in \text{dom}(l) \Rightarrow l \mapsto i \in \text{dom}(\text{DELETE}) \)

axm41: \( \forall l, i \cdot l \in \text{LIST} \land i \in \text{dom}(l) \Rightarrow \text{dom}(\text{DELETE}(l \mapsto i)) = 1 .. \text{card}(l) - 1 \)

axm42: \( \forall l, i, j \cdot l \in \text{LIST} \land i \in \text{dom}(l) \land j \in 1 .. \text{card}(l) - 1 \Rightarrow j \in \text{dom}(\text{DELETE}(l \mapsto i)) \)

axm43: \( \forall l, i, j \cdot l \in \text{LIST} \land i \in \text{dom}(l) \land j \in 1 .. i - 1 \Rightarrow \text{DELETE}(l \mapsto i)(j) = l(j) \)

axm44: \( \forall l, i, j \cdot l \in \text{LIST} \land i \in \text{dom}(l) \land j \in i .. \text{card}(l) - 1 \Rightarrow \text{DELETE}(l \mapsto i)(j) = l(j + 1) \)

axm45: \( \forall l, i \cdot l \in \text{LIST} \land i \in \text{dom}(l) \Rightarrow \text{card}(\text{DELETE}(l \mapsto i)) = \text{card}(l) - 1 \)

axm46: \( \forall l, i \cdot l \in \text{LIST} \land i \in \text{dom}(l) \Rightarrow \text{ran}(\text{DELETE}(l \mapsto i)) = \text{ran}(l) \setminus \{l(i)\} \)

axm47: \( \forall l, i \cdot l \in \text{LIST} \land i \in \text{dom}(l) \Rightarrow \text{card}(\text{DELETE}(l \mapsto i)) = \text{card}(l) - 1 \)

axm48: \( \text{IDELETE} \in (\text{ILIST} \times \text{MEMBER}) \rightarrow \text{ILIST} \)

axm49: \( \text{dom}(\text{IDELETE}) = \text{ILIST} \times \text{MEMBER} \)

axm50: \( \forall l, m \cdot l \in \text{ILIST} \land m \in \text{MEMBER} \land m \in \text{ran}(l) \Rightarrow l \mapsto m \in \text{dom}(\text{IDELETE}) \)
\textbf{axm51}: \forall l, m \cdot l \in \text{ILIST} \land m \in \text{ran}(l) \\
\Rightarrow \qquad \text{dom}(\text{DELETE}(l \mapsto m)) = 1 .. \text{card}(l) - 1

\textbf{axm52}: \forall l, m \cdot l \in \text{ILIST} \land m \in \text{ran}(l) \\
\Rightarrow \qquad \text{IDELETE}(l \mapsto m) = \text{DELETE}(l \mapsto l^{-1}(m))

\textbf{axm53}: \forall l, m \cdot l \in \text{ILIST} \land m \in \text{ran}(l) \\
\Rightarrow \qquad \text{card}(\text{IDELETE}(l \mapsto m)) = \text{card}(l) - 1

\textbf{axm54}: \forall l, m \cdot l \in \text{ILIST} \land m \in \text{ran}(l) \\
\Rightarrow \qquad \text{ran}(\text{IDELETE}(l \mapsto m)) = \text{ran}(l) \setminus \{m\}

\textbf{axm55}: \forall l \cdot l \in \text{LIST} \land l \neq \emptyset \\
\Rightarrow \qquad \text{DEQUEUE}(l) = \text{DELETE}(l \mapsto 1)

\textbf{axm56}: \forall l \cdot l \in \text{LIST} \land l \neq \emptyset \\
\Rightarrow \qquad \text{card}(\text{DEQUEUE}(l)) = \text{card}(l) - 1

\textbf{axm57}: \forall l, i \cdot l \in \text{ILIST} \land i \in \text{dom}(l) \\
\Rightarrow \qquad \text{DEQUEUE}(l) \in \text{ILIST}

\textbf{axm58}: \forall l, i \cdot l \in \text{ILIST} \land i \in \text{dom}(l) \\
\Rightarrow \qquad \text{DELETE}(l \mapsto i) \in \text{ILIST}

\textbf{axm59}: \forall l, m \cdot l \in \text{LIST} \land m \in \text{MEMBER} \\
\Rightarrow \qquad \text{DELETE}(\text{APPEND}(l \mapsto m) \mapsto \text{card}(l) + 1) = l

\textbf{END}