Note: This annotated commentary on the list theory is based on a slightly earlier version of the context. The main difference you will notice is that LIST is defined slightly differently, but equivalently. You should have no difficulty applying this commentary to the newer context.

Explanation of the annotation

The following explains a theory of lists (or sequences) presented as axioms and theorems that could be presented in an Event-B context. The theory is presented in what is frequently called a *declarative* or *axiomatic* style. That means that the descriptions of the sets and functions show how we want to manipulate lists and does not present the detail as to how the underlying sets are modified. The presentation adopts the following general scheme:

- a type definition of a set or function;
- one or more axioms (generally quantified) that describe the required properties and capabilities of the set or function;
- one or more theorems that present other views on the properties and capabilities that are implied by the axioms

Lists are modelled as functions with a coherent domain: 1..length where length is the length of the list. In order to shorten the presentation the following notation will be used for lists:

\[ [A, B, C] \equiv \{1 \mapsto A, 2 \mapsto B, 3 \mapsto C\} \]

with the ordering implied by the position of the list elements within the square brackets.
Usage of this Theory

To use this theory to manage lists, it is only necessary to use the defined sets and functions. It is not necessary to be concerned with the underlying function structure and the changes of that structure. Essentially this theory provides a list language. For example, if there is a variable list, where \( list \in LIST \), then appending the value \( m \), where \( m \in MEMBER \), to list is achieved with

\[
list := \text{APPEND}(list, m)
\]

Of course list is a function and values in list can be referenced in the normal way, e.g \( list(i) \) for any value \( i \in \text{dom}(list) \).

**CONTEXT** Library2.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** Library1.ctx

**CONSTANTS**

- \( LIST \): Set of lists
- \( ILIST \): Set of injective lists
- \( JOIN \): List concatenation operator
- \( APPEND \): Append an item to tail of list
- \( DEQUEUE \): Remove head of list
- \( DELETE \): Remove an element from any position of a list
- \( UNIT \): A unit list

**AXIOMS**

**LIST**

\[ \text{axm1} : LIST \subseteq \mathbb{N}_i \rightarrow MEMBER \]

Axiom 1 models list as a partial function. Lists modelled by LIST can have repeated elements in the list. This axiom provides a type signature.

\[ \text{axm2} : \forall l \cdot l \in LIST \leftrightarrow l \in \mathbb{N}_i \rightarrow MEMBER \land \text{finite}(l) \land \text{dom}(l) = 1 \ldots \text{card}(l) \]

Notice that axiom 2 defines a relationship between the cardinality and the domain of a list.

\[
\text{dom}(l) = 1 \ldots \text{card}(l)
\]

This property holds for all lists, and so for consistency in the following the cardinality of lists produced by functions is defined in axioms. Cardinality is intuitively connected with length. The domain is determined by the cardinality —as shown above— and is thus presented in a theorem.
thm1: \( \forall l \in LIST \Rightarrow \text{dom}(l) = 1 .. \text{card}(l) \)

axm3: \( ILIST \subseteq \mathbb{N} \mapsto MEMBER \)

Axiom 3 defines a set of injective lists. Injective lists do not contain repeated elements.
injective list: \([A, B, C]\); non-injective list: \([A, B, A]\)

axm4: \( \forall l \in ILIST \leftrightarrow l \in LIST \land l \in \mathbb{N} \mapsto MEMBER \)

Axiom 4 completes the definition of injective lists as a subset of ordinary lists but modelled by an injective function. Notice that the axiom defines an injective list to also be an ordinary list.

thm2: \( \varnothing \in ILIST \)
Theorem 2 claims that the empty set is an empty injective list.

thm3: \( ILIST \subset LIST \)
Theorem 3 claims the set of injective lists is a strict subset of the set of lists.

JOIN We now define a function JOIN that is given a pair of lists and joins them to produce a single list.
The intention:

\[
\text{JOIN}([A, B, C] \mapsto [D, E]) = [A, B, C, D, E]
\]

First the type signature,

axm5: \( \text{JOIN} \in (LIST \times LIST) \rightarrow LIST \)

and then the semantics:

axm6: \( \forall l1, l2 \cdot l1 \in LIST \land l2 \in LIST \)

\( \Rightarrow \)

\[
\text{card}(\text{JOIN}(l1 \mapsto l2)) = \text{card}(l1) + \text{card}(l2)
\]

Joining two lists produces a new list whose length (cardinality) is the sum of the lengths of the component lists.

thm4: \( \forall l1, l2 \cdot l1 \in LIST \land l2 \in LIST \)

\( \Rightarrow \)

\[
\text{dom}(\text{JOIN}(l1 \mapsto l2)) = 1 .. \text{card}(l1) + \text{card}(l2)
\]

axm7: \( \forall l1, l2, i \cdot l1 \in LIST \land l2 \in LIST \land i \in \text{dom}(\text{JOIN}(l1 \mapsto l2)) \)

\( \Rightarrow \)

\[
(i \in \text{dom}(l1) \Rightarrow \text{JOIN}(l1 \mapsto l2)(i) = l1(i))
\]

\(\land\)

\[
(i - \text{card}(l1) \in \text{dom}(l2) \Rightarrow \text{JOIN}(l1 \mapsto l2)(i) = l2(i - \text{card}(l1)))
\]

Indexing a joined-list before the join is equivalent to indexing the first list of the join:

\[
(\text{JOIN}([A, B, C] \mapsto [D, E])(2) = [A, B, C](2)
\]
while indexing a joined list after the join is equivalent to indexing the second list of the join with the index corrected for the length of the first list:

\[(JOIN([A, B, C] \mapsto [D, E])(4) = [D, E](1)\]

**thm5**: \(\forall l \cdot l \in LIST \Rightarrow JOIN(l \mapsto \emptyset) = l\)

**thm6**: \(\forall l \cdot l \in LIST \Rightarrow JOIN(\emptyset \mapsto l) = l\)

Theorems 4 & 5 deduce that if either of the lists being joined is the empty list then the join is simply the other list.

**thm7**: \(\forall l_1, l_2 \cdot l_1 \in LIST \land l_2 \in LIST \Rightarrow ran(JOIN(l_1 \mapsto l_2)) = ran(l_1) \cup ran(l_2)\)

Theorem 6 deduces that the range of a joined list is the union of the ranges of the lists being joined

\[ran(JOIN([A, B, C] \mapsto [D, E]) = \{A, B, C\} \cup \{D, E\}\]

**thm7**: \(\forall l_1, l_2 \cdot l_1 \in ILIST \land l_2 \in ILIST \land ran(l_1) \cap ran(l_2) = \emptyset \Rightarrow JOIN(l_1 \mapsto l_2) \in ILIST\)

Theorem 7 deduces that the join of two injective lists that are disjoint (no common elements) is an injective list.

**APPEND**: APPEND adds a single element to the end of a list

\[APPEND([A, B, C] \mapsto D) = [A, B, C, D]\]

**axm8**: \(APPEND \in (LIST \times MEMBER) \rightarrow LIST\)

Axioms 9 & 10 define some of the behaviour of APPEND.

**axm9**: \(\forall l, m \cdot l \in LIST \Rightarrow card(APPEND(l \mapsto m)) = card(l) + 1\)

**thm9**: \(\forall l, m \cdot l \in LIST \Rightarrow dom(APPEND(l \mapsto m)) = 1 \ldots card(l) + 1\)

**axm10**: \(\forall l, m, i \cdot l \in LIST \land i \in dom(APPEND(l \mapsto m)) \Rightarrow (i \in dom(l) \Rightarrow APPEND(l \mapsto m)(i) = l(i))\)

\(^\wedge\)

\((i = card(l) + 1 \Rightarrow APPEND(l \mapsto m)(i) = m)\)

\[APPEND([A, B, C] \mapsto D)(2) = [A, B, C](2)\]

\[APPEND([A, B, C] \mapsto D)(4) = D\]

Theorem 8 deduces the value of \(ran(APPEND(l \mapsto m))\).
thm10: \( \forall l, m \cdot l \in LIST \land m \in MEMBER \Rightarrow \)  
\[
\text{ran}(APPEND(l \mapsto m)) = \text{ran}(l) \cup \{m\}
\]
\[
\text{ran}(APPEND([A, B, C] \mapsto D)) = \{A, B, C\} \cup \{D\}
\]

Hence, we deduce the condition for APPEND to yield an injective list.

thm11: \( \forall l, m \cdot l \in ILIST \land m \in MEMBER \land m \notin \text{ran}(l) \Rightarrow \)  
\[
\text{APPEND}(l \mapsto m) \in ILIST
\]

The cardinality of the list produced by APPEND is clearly one larger than the cardinality of list to which the append applies.

thm12: \( \forall l, m \cdot l \in LIST \land m \in MEMBER \Rightarrow \)  
\[
\text{card}(APPEND(l \mapsto m)) = \text{card}(l) + 1
\]

DEQUEUE: DEQUEUE is an operation that removes the first item of a list. Notice that APPEND and DEQUEUE support the use of a list to model a queue: a first-in-last-out structure.

axm11: \( \text{DEQUEUE} \in LIST \to LIST \)

axm12: \( \forall l \cdot l \in LIST \land l \neq \emptyset \Rightarrow \)  
\[
\text{card}(\text{DEQUEUE}(l)) = \text{card}(l) - 1
\]

DEQUEUE can only be used on a non-empty list and the cardinality of the resultant list is decreased by one.

thm13: \( \forall l \cdot l \in LIST \land l \neq \emptyset \Rightarrow \)  
\[
\text{dom}(\text{DEQUEUE}(l)) = 1 \ldots \text{card}(l) - 1
\]

axm13: \( \forall l \cdot l \in LIST \land l \neq \emptyset \land i \in 1 \ldots \text{card}(l) - 1 \Rightarrow \)  
\[
\text{DEQUEUE}(l)(i) = l(i + 1)
\]

Referencing the i-th element in the dequeued list is the same as referencing the i+1-th element of the list before the dequeue.

thm14: \( \forall l \cdot l \in ILIST \land l \neq \emptyset \Rightarrow \)  
\[
\text{ran}(\text{DEQUEUE}(l)) = \text{ran}(l) \setminus \{l(1)\}
\]

Dequeueing an element from an injective list will remove that element from the range of the list.

thm15: \( \forall l \cdot l \in ILIST \land l \neq \emptyset \Rightarrow \)  
\[
\text{DEQUEUE}(l) \in ILIST
\]

Dequeueing an element from an injective list produces an injective list.
**DELETE**: an operation for deleting an element from a list.

**axm14**: \( \text{DELETE} \in (\text{LIST} \times \mathbb{N}) \rightarrow \text{LIST} \)

DELETE takes a pair consisting of a list and an index and produces a list.

**axm15**: \( \forall \, l, \, i \cdot l \in \text{LIST} \land i \in \text{dom}(l) \)

\( l \mapsto i \in \text{dom}(\text{DELETE}) \)

For \( l \mapsto i \) to be in the domain of DELETE the index must be in the domain of the list, \( i \in \text{dom}(l) \)

**axm16**: \( \forall \, l, \, i \cdot l \in \text{LIST} \land i \in \text{dom}(l) \)

\( \Rightarrow \)

\( \text{card}(\text{DELETE}(l \mapsto i)) = \text{card}(l) - 1 \)

Deleting an element from a list decreases the cardinality of the list by one.

**thm16**: \( \forall \, l, \, i \cdot l \in \text{LIST} \land i \in \text{dom}(l) \)

\( \Rightarrow \)

\( \text{dom}(\text{DELETE}(l \mapsto i)) = 1..\text{card}(l) - 1 \)

**axm17**: \( \forall \, l, \, i, \, j \cdot l \in \text{LIST} \land i \in \text{dom}(l) \)

\( \Rightarrow \)

\( j \in 1..i - 1 \Rightarrow \text{DELETE}(l \mapsto i)(j) = l(j) \)

\( \land \)

\( j \in i..\text{card}(l) - 1 \Rightarrow \text{DELETE}(l \mapsto i)(j) = l(j + 1) \)

The index of an element in the deleted list that is positioned before the element that was deleted is unchanged, while the index of an element positioned after the element that was deleted decreases by one.

**thm17**: \( \forall l \cdot l \in \text{LIST} \land l \neq {} \)

\( \Rightarrow \)

\( \text{DEQUEUE}(l) = \text{DELETE}(l \mapsto 1) \)

Clearly, DELETE can be used in place of DEQUEUE.

**thm18**: \( \forall l, \, i \cdot l \in \text{ILIST} \land i \in \text{dom}(l) \)

\( \Rightarrow \)

\( \text{ran}(\text{DELETE}(l \mapsto i)) = \text{ran}(l) \setminus \{l(i)\} \)

Deleting an element from an injective list removes that element from the range of the list.

**thm19**: \( \forall l, \, i \cdot l \in \text{ILIST} \land i \in \text{dom}(l) \)

\( \Rightarrow \)

\( \text{DELETE}(l \mapsto i) \in \text{ILIST} \)

Deleting an element from an injective list produces an injective list.

**thm20**: \( \forall l, \, m \cdot l \in \text{LIST} \land m \in \text{MEMBER} \)

\( \Rightarrow \)

\( \text{DELETE}(\text{APPEND}(l \mapsto m) \mapsto \text{card}(l) + 1) = l \)
Appending an element to a list and then deleting the last element from the resulting list should produce the original list, before the append.

**UNIT:** UNIT is a singleton list

- **axm18:** \( UNIT \in MEMBER \rightarrow LIST \)
- **axm19:** \( \forall m. m \in MEMBER \Rightarrow UNIT(m) = \{1 \mapsto m\} \)

A singleton list can be created using APPEND, so UNIT is not really necessary.

- **thm21:** \( \forall m. m \in MEMBER \Rightarrow UNIT(m) = \text{APPEND}(\emptyset \mapsto m) \)

END