MODULE *SequenceUtils*

EXTENDS *FiniteSets*, *Sequences*, *SetUtils*, *FunctionUtils*
LOCAL INSTANCE *Naturals*

IsSequenceOfSetElements is a predicate that is true when the specified sequence contains all and only elements of the specified set.

IsSortedSequenceOfSetElements is a predicate that is true when the

IsSequenceOfSetElements is true and the sequence is also sorted in increasing order.

$Prepend(s,\ e) \triangleq \langle e \rangle \circ s$

$First(seq) \triangleq seq[1]$

$Last(seq) \triangleq seq[Len(seq)]$

$AllButFirst(seq) \triangleq [i \in 1\ ..\ (Len(seq) - 1) \mapsto seq[(i + 1)]]$

$AllButLast(seq) \triangleq [i \in 1\ ..\ (Len(seq) - 1) \mapsto seq[i]]$

$DoesSeqPrefixSeq(seq1,\ seq2) \triangleq$
$\quad \land\ Len(seq1) \leq Len(seq2)$
$\quad \land\ (\forall\ i \in 1\ ..\ Len(seq1) : seq1[i] = seq2[i])$

$DoesSeqProperlyPrefixSeq(seq1,\ seq2) \triangleq$
$\quad \land\ Len(seq1) < Len(seq2)$
$\quad \land\ (\forall\ i \in 1\ ..\ Len(seq1) : seq1[i] = seq2[i])$

$IsElementInSeq(el,\ seq) \triangleq \exists\ i \in \text{DOMAIN}\ seq : seq[i] = el$

$IsSequenceOfSetElements(seq,\ set) \triangleq$
$\quad \land\ Len(seq) = Cardinality(set)$
$\quad \land\ (\forall\ el \in set : IsElementInSeq(el,\ seq))$

$IsSortedSequenceOfSetElements(seq,\ set) \triangleq$
$\quad \land\ IsSequenceOfSetElements(seq,\ set)$
$\quad \land\ (\forall\ i \in \text{DOMAIN}\ seq,\ j \in \text{DOMAIN}\ seq : i < j \Rightarrow seq[i] < seq[j])$

$DeleteElement(seq,\ index) \triangleq$
$\quad [i \in 1\ ..\ (Len(seq) - 1) \mapsto \text{IF}\ i < index\ \text{THEN}\ seq[i]\ \text{ELSE}\ \ seq[(i + 1)]]$

Retain only the elements in R in their original order in *seq*.

RECURSIVE $Retain(\_,\ \_)$
$Retain(seq,\ R) \triangleq$
$\quad \text{IF}\ seq = \langle\rangle$
$\quad\ \text{THEN}\ \langle\rangle$
$\quad\ \text{ELSE}\ \ \text{LET}\ h \triangleq Head(seq)$
$\quad\qquad \text{IN}\ \ \text{IF}\ h \in R$

THEN $\langle h \rangle \circ Retain(Tail(seq),\ R)$
ELSE $Retain(Tail(seq),\ R)$

It requires that $index \geq 1$.

If $index > Len(seq) + 1$, then it appends the element to seq.

(ADDED by anonymous; *July* 04, 2018)

$$
\begin{aligned}
&InsertElement(seq,\ elem,\ index) \triangleq \\
&\quad [i \in 1 \,..\, (Len(seq) + 1) \mapsto \text{IF } i < index \\
&\qquad\qquad \text{THEN IF } i = (Len(seq) + 1) \\
&\qquad\qquad\qquad \text{THEN } elem \\
&\qquad\qquad\qquad \text{ELSE } seq[i] \\
&\qquad\qquad \text{ELSE IF } i = index \\
&\qquad\qquad\qquad \text{THEN } elem \\
&\qquad\qquad\qquad \text{ELSE } seq[(i-1)]] \quad i > index
\end{aligned}
$$

$$
\begin{aligned}
&IsSorted2Partition(n,\ seq1,\ seq2) \triangleq \\
&\quad \land\ seq1 \in Seq(1 \,..\, n) \\
&\quad \land\ seq2 \in Seq(1 \,..\, n) \\
&\quad \land\ n = Len(seq1) + Len(seq2) \\
&\quad \land\ (\forall\, i \in \text{DOMAIN } seq1,\ j \in \text{DOMAIN } seq1 : i < j \Rightarrow seq1[i] < seq1[j]) \\
&\quad \land\ (\forall\, i \in \text{DOMAIN } seq2,\ j \in \text{DOMAIN } seq2 : i < j \Rightarrow seq2[i] < seq2[j]) \\
&\quad \land\ (\forall\, i \in \text{DOMAIN } seq1,\ j \in \text{DOMAIN } seq2 : seq1[i] \neq seq2[j])
\end{aligned}
$$

$$
\begin{aligned}
&IsSequenceInterleaving(seq,\ subSeq1,\ subSeq2,\ indSeq1,\ indSeq2) \triangleq \\
&\quad \land\ indSeq1 \in Seq(Nat) \\
&\quad \land\ indSeq2 \in Seq(Nat) \\
&\quad \land\ IsSorted2Partition(Len(seq),\ indSeq1,\ indSeq2) \\
&\quad \land\ Len(indSeq1) = Len(subSeq1) \\
&\quad \land\ Len(indSeq2) = Len(subSeq2) \\
&\quad \land\ (\forall\, i \in \text{DOMAIN } indSeq1 : seq[(indSeq1[i])] = subSeq1[i]) \\
&\quad \land\ (\forall\, i \in \text{DOMAIN } indSeq2 : seq[(indSeq2[i])] = subSeq2[i])
\end{aligned}
$$

Sequences up to length n, including the empty sequence $\langle\rangle$.

Copyright: https://*www.learntla.com*/libraries/sequences/

$$SeqMaxLen(S,\ n) \triangleq \text{UNION } \{[1 \,..\, m \to S] : m \in 0 \,..\, n\}$$

Map on a sequence.

Copyright: https://*www.learntla.com*/libraries/sequences/

$$SeqMap(Op(\_),\ seq) \triangleq [x \in \text{DOMAIN } seq \mapsto Op(seq[x])]$$

$$PermsWithin(S) \triangleq \{s \in \text{UNION } \{[1 \,..\, m \to S] : m \in 0 \,..\, Cardinality(S)\} : Cardinality(Range(s)) = Cardina$$

All possible permutations generated based on sequence T.

Copyright: https://*learntla.com*/tla/functions/

$PermutationKey(n) \triangleq \{key \in [1 \,..\, n \to 1 \,..\, n] : Range(key) = 1 \,..\, n\}$
$PermutationsOf(T) \triangleq \{[x \in 1 \,..\, Len(T) \mapsto T[P[x]]] : P \in PermutationKey(Len(T))\}$

Get the index of the first occurrence of *elem* in *seq*.

Precondition: $elem \in SeqImage(seq)$.

ADDED by anonymous; *Aug.* 12, 2018

```
RECURSIVE FirstIndexOfElement(_, _)
FirstIndexOfElement(seq, elem) ≜
    IF Head(seq) = elem
     THEN 1
     ELSE 1 + FirstIndexOfElement(Tail(seq), elem)
```

Get the index of the first occurence of *elem* in *seq*. It returns 0 if *elem* does not occur in *seq*.

```
RECURSIVE FirstIndexOfElementSafe(_, _)
FirstIndexOfElementSafe(seq, elem) ≜
    LET RECURSIVE FirstIndexOfElementSafeHelper(_, _, _)
        FirstIndexOfElementSafeHelper(seqh, elemh, fail) ≜
            IF seqh = ⟨⟩
             THEN 0 − fail
             ELSE IF Head(seqh) = elemh
                    THEN 1
                    ELSE 1 + FirstIndexOfElementSafeHelper(Tail(seqh), elemh, fail + 1)
     IN FirstIndexOfElementSafeHelper(seq, elem, 0)
```

Check if two sequences are compatible.

Precondition: No duplication in each individual sequence.

Two sequences are compatible if and only if for any two common elements in both sequences, the relative order of them in the two sequences are the same.

ADDED by anonymous; *Aug.* 12, 2018

```
Compatible(seq1, seq2) ≜
    ∨ seq1 = seq2
    ∨ LET commonElements ≜ Range(seq1) ∩ Range(seq2)
      IN ∀ e1, e2 ∈ commonElements :
           ∨ e1 = e2
           ∨ FirstIndexOfElement(seq1, e1) < FirstIndexOfElement(seq1, e2)
             ≡ FirstIndexOfElement(seq2, e1) < FirstIndexOfElement(seq2, e2)
```

The length of the longest common subsequence of two sequences *seq*1 and *seq*2.

ADDED by anonymous; *Aug.* 12, 2018

```
RECURSIVE LCS(_, _)
LCS(seq1, seq2) ≜
    IF seq1 = ⟨⟩ ∨ seq2 = ⟨⟩
     THEN 0
     ELSE IF Last(seq1) = Last(seq2)
            THEN 1 + LCS(AllButLast(seq1), AllButLast(seq2))
```

ELSE $MaxOfSet(\{LCS(AllButLast(seq1),\ seq2),\ LCS(seq1,\ AllButLast(seq2))\})$

$LCSCompatible(seq1,\ seq2) \triangleq$
$Compatible(seq1,\ seq2) \equiv LCS(seq1,\ seq2) = Cardinality(Range(seq1) \cap Range(seq2))$

$LCSCompatibleTest(S) \triangleq$
$\forall\, seq1,\ seq2 \in PermsWithin(S) : LCSCompatible(seq1,\ seq2)$

* Modification History
* *Last* modified *Tue Dec* 04 19:42:23 *CST* 2018 by anonymous
* Created *Tue Jul* 03 15:21:02 *CST* 2018 by anonymous