MODULE *AdditionalSequenceOperators*

Copyright: https://*github.com*/bringhurst/tlaplus/blob/master/*org.lamport.tla.toolbox.uitest*/farsite/*AdditionalSequenceOperat*

EXTENDS *FiniteSets*, *Sequences*, *TLC*, *AdditionalSetOperators*, *AdditionalFunctionOperators*

LOCAL INSTANCE *Naturals*

The TLA+ *Sequences* module defines the operators *Head* and *Tail* for retrieving the first element of a sequence and all-but-the-first elements of a sequence, respectively. This module provides four operators that slightly generalize the notions of *Head* and *Tail*:

First returns the first element of a sequence, equivalently to *Head*. Last returns the last element of a sequence. *AllButFirst* returns all-but-the-first elements of a sequence, equivalently to *Tail*.

AllButLast returns all-but-the-last elements of a sequence.

This module also provides several additional operators on sequences: *IsElementInSeq* is a predicate that is true when the specified value is an element of the specified sequence. *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. *DeleteElement* produces a sequence by deleting an indicated element from another sequence.

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

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

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

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

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

$$\begin{aligned} &DoesSeqPrefixSeq(seq1,\ seq2) \triangleq \\ &\quad \land\ Len(seq1) \leq Len(seq2) \\ &\quad \land\ (\forall\ i \in 1\ ..\ Len(seq1) : seq1[i] = seq2[i]) \end{aligned}$$

$$\begin{aligned} &DoesSeqProperlyPrefixSeq(seq1,\ seq2) \triangleq \\ &\quad \land\ Len(seq1) < Len(seq2) \\ &\quad \land\ (\forall\ i \in 1\ ..\ Len(seq1) : seq1[i] = seq2[i]) \end{aligned}$$

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

$$\begin{aligned} &IsSequenceOfSetElements(seq,\ set) \triangleq \\ &\quad \land\ Len(seq) = Cardinality(set) \\ &\quad \land\ (\forall\ el \in set : IsElementInSeq(el,\ seq)) \end{aligned}$$

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

$$DeleteElement(seq,\ index) \triangleq$$

```
61   [i ∈ 1 .. (Len(seq) − 1) ↦ IF i < index THEN seq[i] ELSE seq[(i + 1)]]
```

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

```
66 RECURSIVE Retain(_, _)
67 Retain(seq, R) ≜
68     IF seq = ⟨⟩
69      THEN ⟨⟩
70      ELSE LET h ≜ Head(seq)
71               IN  IF h ∈ R
72                    THEN ⟨h⟩ ∘ Retain(Tail(seq), R)
73                    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 hengxin; *July* 04, 2018)

```
81 InsertElement(seq, elem, index) ≜
82   [i ∈ 1 .. (Len(seq) + 1) ↦ IF i < index
83                              THEN IF i = (Len(seq) + 1)
84                                   THEN elem
85                                   ELSE seq[i]
86                              ELSE IF i = index
87                                   THEN elem
88                                   ELSE seq[(i − 1)]]  i > index

90 IsSorted2Partition(n, seq1, seq2) ≜
91   ∧ seq1 ∈ Seq(1 .. n)
92   ∧ seq2 ∈ Seq(1 .. n)
93   ∧ n = Len(seq1) + Len(seq2)
94   ∧ (∀ i ∈ DOMAIN seq1, j ∈ DOMAIN seq1 : i < j ⇒ seq1[i] < seq1[j])
95   ∧ (∀ i ∈ DOMAIN seq2, j ∈ DOMAIN seq2 : i < j ⇒ seq2[i] < seq2[j])
96   ∧ (∀ i ∈ DOMAIN seq1, j ∈ DOMAIN seq2 : seq1[i] ≠ seq2[j])

98 IsSequenceInterleaving(seq, subSeq1, subSeq2, indSeq1, indSeq2) ≜
99   ∧ indSeq1 ∈ Seq(Nat)
100  ∧ indSeq2 ∈ Seq(Nat)
101  ∧ IsSorted2Partition(Len(seq), indSeq1, indSeq2)
102  ∧ Len(indSeq1) = Len(subSeq1)
103  ∧ Len(indSeq2) = Len(subSeq2)
104  ∧ (∀ i ∈ DOMAIN indSeq1 : seq[(indSeq1[i])] = subSeq1[i])
105  ∧ (∀ i ∈ DOMAIN indSeq2 : seq[(indSeq2[i])] = subSeq2[i])
```

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

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

```
112 SeqMaxLen(S, n) ≜  UNION {[1 .. m → S] : m ∈ 0 .. n}
```

Map on a sequence.

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

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

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

All possible permutations generated based on sequence T.

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

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

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

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

ADDED by hengxin; *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 hengxin; *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 $seq1$ and $seq2$.

ADDED by hengxin; *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) ≜
  Compatible(seq1, seq2) ≡ LCS(seq1, seq2) = Cardinality(Range(seq1) ∩ Range(seq2))

LCSCompatibleTest(S) ≜
  ∀ seq1, seq2 ∈ PermsWithin(S) : LCSCompatible(seq1, seq2)
```

* Modification History
* *Last* modified *Mon Nov* 05 21:10:44 *CST* 2018 by *hengxin*
* Created *Tue Jul* 03 15:21:02 *CST* 2018 by *hengxin*