

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.

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

$$\text{First}(seq) \triangleq seq[1]$$

$$\text{Last}(seq) \triangleq seq[\text{Len}(seq)]$$

$$\text{AllButFirst}(seq) \triangleq [i \in 1 \dots (\text{Len}(seq) - 1) \mapsto seq[(i + 1)]]$$

$$\text{AllButLast}(seq) \triangleq [i \in 1 \dots (\text{Len}(seq) - 1) \mapsto seq[i]]$$

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

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

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

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

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

$$\begin{aligned} \text{DeleteElement}(seq, index) &\triangleq \\ [i \in 1 \dots (\text{Len}(seq) - 1) \mapsto \text{IF } i < index \text{ THEN } seq[i] \text{ ELSE } seq[(i + 1)]] \end{aligned}$$

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

RECURSIVE *Retain*(-, -)

$$\begin{aligned} \text{Retain}(seq, R) &\triangleq \\ \text{IF } seq &= \langle \rangle \\ \text{THEN } &\langle \rangle \\ \text{ELSE LET } h &\triangleq \text{Head}(seq) \\ \text{IN IF } h &\in R \end{aligned}$$

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

It requires that  $\text{index} \geq 1$ .

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

(ADDED by hengxin; July 04, 2018)

$\text{InsertElement}(\text{seq}, \text{elem}, \text{index}) \triangleq$   
 $[i \in 1 \dots (\text{Len}(\text{seq}) + 1) \mapsto \text{IF } i < \text{index}$   
    THEN IF  $i = (\text{Len}(\text{seq}) + 1)$   
        THEN  $\text{elem}$   
        ELSE  $\text{seq}[i]$   
    ELSE IF  $i = \text{index}$   
        THEN  $\text{elem}$   
        ELSE  $\text{seq}[(i - 1)]]$   $i > \text{index}$

$\text{IsSorted2Partition}(n, \text{seq1}, \text{seq2}) \triangleq$   
 $\wedge \text{seq1} \in \text{Seq}(1 \dots n)$   
 $\wedge \text{seq2} \in \text{Seq}(1 \dots n)$   
 $\wedge n = \text{Len}(\text{seq1}) + \text{Len}(\text{seq2})$   
 $\wedge (\forall i \in \text{DOMAIN } \text{seq1}, j \in \text{DOMAIN } \text{seq1} : i < j \Rightarrow \text{seq1}[i] < \text{seq1}[j])$   
 $\wedge (\forall i \in \text{DOMAIN } \text{seq2}, j \in \text{DOMAIN } \text{seq2} : i < j \Rightarrow \text{seq2}[i] < \text{seq2}[j])$   
 $\wedge (\forall i \in \text{DOMAIN } \text{seq1}, j \in \text{DOMAIN } \text{seq2} : \text{seq1}[i] \neq \text{seq2}[j])$

$\text{IsSequenceInterleaving}(\text{seq}, \text{subSeq1}, \text{subSeq2}, \text{indSeq1}, \text{indSeq2}) \triangleq$   
 $\wedge \text{indSeq1} \in \text{Seq}(\text{Nat})$   
 $\wedge \text{indSeq2} \in \text{Seq}(\text{Nat})$   
 $\wedge \text{IsSorted2Partition}(\text{Len}(\text{seq}), \text{indSeq1}, \text{indSeq2})$   
 $\wedge \text{Len}(\text{indSeq1}) = \text{Len}(\text{subSeq1})$   
 $\wedge \text{Len}(\text{indSeq2}) = \text{Len}(\text{subSeq2})$   
 $\wedge (\forall i \in \text{DOMAIN } \text{indSeq1} : \text{seq}[(\text{indSeq1}[i])] = \text{subSeq1}[i])$   
 $\wedge (\forall i \in \text{DOMAIN } \text{indSeq2} : \text{seq}[(\text{indSeq2}[i])] = \text{subSeq2}[i])$

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

Copyright: <https://www.learntla.com/libraries/sequences/>

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

Map on a sequence.

Copyright: <https://www.learntla.com/libraries/sequences/>

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

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

All possible permutations generated based on sequence  $T$ .

Copyright: <https://learntla.com/tla/functions/>

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

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

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

ADDED by hengxin; Aug. 12, 2018

RECURSIVE *FirstIndexOfElement*( $\_, \_$ )  
 $FirstIndexOfElement(seq, elem) \triangleq$   
 IF *Head(seq)* = *elem*  
 THEN 1  
 ELSE 1 + *FirstIndexOfElement*(*Tail(seq)*, *elem*)

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

RECURSIVE *FirstIndexOfElementSafe*( $\_, \_$ )  
 $FirstIndexOfElementSafe(seq, elem) \triangleq$   
 LET RECURSIVE *FirstIndexOfElementSafeHelper*( $\_, \_, \_$ )  
 $FirstIndexOfElementSafeHelper(seqh, elemh, fail) \triangleq$   
 IF *seqh* =  $\langle \rangle$   
 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

$Compatible(seq1, seq2) \triangleq$   
 $\vee seq1 = seq2$   
 $\vee$  LET *commonElements*  $\triangleq Range(seq1) \cap Range(seq2)$   
 IN  $\forall e1, e2 \in commonElements :$   
 $\vee e1 = e2$   
 $\vee FirstIndexOfElement(seq1, e1) < FirstIndexOfElement(seq1, e2)$   
 $\equiv FirstIndexOfElement(seq2, e1) < FirstIndexOfElement(seq2, e2)$

The length of the longest common subsequence of two sequences *seq1* and *seq2*.

ADDED by hengxin; Aug. 12, 2018

RECURSIVE *LCS*( $\_, \_$ )  
 $LCS(seq1, seq2) \triangleq$   
 IF *seq1* =  $\langle \rangle$   $\vee$  *seq2* =  $\langle \rangle$   
 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 hengxin
\ * Created Tue Jul 03 15:21:02 CST 2018 by hengxin

```