

Funcons-beta: Sequences *

The P_LanCompS Project

Sequences.cbs | PLAIN | PRETTY

OUTLINE

- Sequences of values
 - Sequence indexing
 - Homogeneous sequences

Sequences of values

```
[ Funcon  length
  Funcon  index
  Funcon  is-in
  Funcon  first
  Funcon  second
  Funcon  third
  Funcon  first-n
  Funcon  drop-first-n
  Funcon  reverse
  Funcon  n-of
  Funcon  intersperse ]
```

Sequences of two or more values are not themselves values, nor is the empty sequence a value. However, sequences can be provided to funcons as arguments, and returned as results. Many operations on composite values can be expressed by extracting their components as sequences, operating on the sequences, then forming the required composite values from the resulting sequences.

A sequence with elements X_1, \dots, X_n is written X_1, \dots, X_n . A sequence with a single element X is identified with (and written) X . An empty sequence is indicated by the absence of a term. Any sequence X^* can be enclosed in parentheses (X^*) , e.g.: $()$, (1) , $(1, 2, 3)$. Superfluous commas are ignored.

The elements of a type sequence T_1, \dots, T_n are the value sequences V_1, \dots, V_n where $V_1 : T_1, \dots, V_n : T_n$. The only element of the empty type sequence $()$ is the empty value sequence $()$.

$(T)^N$ is equivalent to T, \dots, T with N occurrences of T . $(T)^*$ is equivalent to the union of all $(T)^N$ for $N \geq 0$, $(T)^+$ is equivalent to the union of all $(T)^N$ for $N \geq 1$, and $(T)?$ is equivalent to $T \mid ()$. The parentheses around T above can be omitted when they are not needed for disambiguation.

(Non-trivial) sequence types are not values, so not included in **types**.

Meta-variables $T, T' <: \text{values}$

Funcon **length**($_ : \text{values}^*$) : \Rightarrow **natural-numbers**

*Suggestions for improvement: plancomps@gmail.com.
Reports of issues: <https://github.com/plancomps/CBS-beta/issues>.

$\text{length}(V^*)$ gives the number of elements in V^* .

Rule $\text{length}(\) \rightsquigarrow 0$

Rule $\text{length}(V : \text{values}, V^* : \text{values}^*) \rightsquigarrow \text{natural-successor}(\text{length}(V^*))$

Funcon $\text{is-in}(_ : \text{values}, _ : \text{values}^*) : \Rightarrow \text{booleans}$

Rule $\text{is-in}(V : \text{values}, V' : \text{values}, V^* : \text{values}^*) \rightsquigarrow \text{or}(\text{is-equal}(V, V'), \text{is-in}(V, V^*))$

Rule $\text{is-in}(V : \text{values}, (\)) \rightsquigarrow \text{false}$

Sequence indexing

Funcon $\text{index}(_ : \text{natural-numbers}, _ : \text{values}^*) : \Rightarrow \text{values}^?$

$\text{index}(N, V^*)$ gives the N th element of V^* , if it exists, otherwise $(\)$.

Rule $\text{index}(1, V : \text{values}, V^* : \text{values}^*) \rightsquigarrow V$

Rule
$$\frac{\text{natural-predecessor}(N) \rightsquigarrow N'}{\text{index}(N : \text{positive-integers}, _ : \text{values}, V^* : \text{values}^*) \rightsquigarrow \text{index}(N', V^*)}$$

Rule $\text{index}(0, V^* : \text{values}^*) \rightsquigarrow (\)$

Rule $\text{index}(_ : \text{positive-integers}, (\)) \rightsquigarrow (\)$

Total indexing funcons:

Funcon $\text{first}(_ : T, _ : \text{values}^*) : \Rightarrow T$

Rule $\text{first}(V : T, V^* : \text{values}^*) \rightsquigarrow V$

Funcon $\text{second}(_ : \text{values}, _ : T, _ : \text{values}^*) : \Rightarrow T$

Rule $\text{second}(_ : \text{values}, V : T, V^* : \text{values}^*) \rightsquigarrow V$

Funcon $\text{third}(_ : \text{values}, _ : \text{values}, _ : T, _ : \text{values}^*) : \Rightarrow T$

Rule $\text{third}(_ : \text{values}, _ : \text{values}, V : T, V^* : \text{values}^*) \rightsquigarrow V$

Homogeneous sequences

Funcon $\text{first-n}(_ : \text{natural-numbers}, _ : (T)^*) : \Rightarrow (T)^*$

Rule $\text{first-n}(0, V^* : (T)^*) \rightsquigarrow (\)$

Rule
$$\frac{\text{natural-predecessor}(N) \rightsquigarrow N'}{\text{first-n}(N : \text{positive-integers}, V : T, V^* : (T)^*) \rightsquigarrow (V, \text{first-n}(N', V^*))}$$

Rule $\text{first-n}(N : \text{positive-integers}, (\)) \rightsquigarrow (\)$

Funcon $\text{drop-first-n}(_ : \text{natural-numbers}, _ : (T)^*) : \Rightarrow (T)^*$

Rule $\text{drop-first-n}(0, V^* : (T)^*) \rightsquigarrow V^*$

Rule
$$\frac{\text{natural-predecessor}(N) \rightsquigarrow N'}{\text{drop-first-n}(N : \text{positive-integers}, _ : T, V^* : (T)^*) \rightsquigarrow \text{drop-first-n}(N', V^*)}$$

Rule $\text{drop-first-n}(N : \text{positive-integers}, (\)) \rightsquigarrow (\)$

Funcon $\text{reverse}(_ : (T)^*) : \Rightarrow (T)^*$

Rule $\text{reverse}(\) \rightsquigarrow (\)$

Rule $\text{reverse}(V : T, V^* : (T)^*) \rightsquigarrow (\text{reverse}(V^*), V)$

Funcon $\text{n-of}(N : \text{natural-numbers}, V : T) : \Rightarrow (T)^*$

Rule $\text{n-of}(0, _ : T) \rightsquigarrow ()$

Rule
$$\frac{\text{natural-predecessor}(N) \rightsquigarrow N'}{\text{n-of}(N : \text{positive-integers}, V : T) \rightsquigarrow (V, \text{n-of}(N', V))}$$

Funcon $\text{intersperse}(_ : T', _ : (T)^*) : \Rightarrow (T, (T', T)^*)?$

Rule $\text{intersperse}(_ : T', ()) \rightsquigarrow ()$

Rule $\text{intersperse}(_ : T', V) \rightsquigarrow V$

Rule $\text{intersperse}(V' : T', V_1 : T, V_2 : T, V^* : (T)^*) \rightsquigarrow (V_1, V', \text{intersperse}(V', V_2, V^*))$