### B.1.3.3 Permutation

Permutation is an operation for matching that shall be used only on values of **record of** and array types. Permutation is denoted by the keyword **permutation**. Expressions, templates and *AnyElement* and *AnyElementsOrNone* are allowed as permutation elements. Permutation elements shall obey the restrictions given below.

A permutation without *AnyElementsOrNone* in place of a single record of element means that any series of elements is acceptable provided that there is a one to one mapping between elements in the record of and in the permutation list such that each element matches its corresponding element in the permutation list.

*AnyElementsOrNone* used inside permutation (directly or via reference) replaces none or any number of elements within the segment of the record of value matched by permutation. The permutation matching is successful, if a subset of the elements in the record of matches the permutation list without the *AnyElementsOrNone*. If both permutation and *AnyElementsOrNone* are used in a record of template, they shall be evaluated jointly.

NOTE 1: *AnyElementsOrNone* used inside permutation has a different effect as *AnyElementsOrNone* used in conjunction with permutation as in the latter *AnyElementsOrNone* replaces consecutive elements only. For example, {**permutation**(1,2,\*)} is equivalent to ({\*,1,\*,2,\*},{\*,2,\*,1,\*}), while {**permutation**(1,2),\*} is equivalent to ({1,2,\*},{2,1,\*}).

NOTE 2: When *AnyElementsOrNone* is inside a permutation, a length attribute may be applied to *AnyElementsOrNone* to restrict the number of elements matched by *AnyElementsOrNone* (see also clause B.1.4.1).

Besides specifying all individual values, it is possible to add all elements of a **record of** or **set of** template into permutations using an **all from** clause.

***Restrictions***

a) Each individual member listed in the permutation shall be of the type replicated by the **record of** or array type.

a) The member type of the permutation and the member type of the template in the **all from** clause shall be compatible.

b) The template in the **all from** clause as a whole shall not resolve into a matching mechanism other than a *SpecificValue* (see clause B.1.1), and its elements may resolve to the matching mechanisms *SpecificValue*, *AnyElement* and *AnyElementsOrNone* only.

c) Individual members of a permutation and elements of the template in the **all from** clause shall only be expressions, templates obeying to restriction b) above, and the *AnyElement* and *AnyElementsOrNone* matching mechanisms.

***Examples***

EXAMPLE 1:

 **type record of integer** MySequenceOfType;

 **template** MySequenceOfType MyTemplate1 := { **permutation** ( 1, 2, 3 ), 5 };

 // matches any of the following sequences of 4 integers: 1,2,3,5; 1,3,2,5; 2,1,3,5;

 // 2,3,1,5; 3,1,2,5; or 3,2,1,5

 **template** MySequenceOfType MyTemplate2 := { **permutation** ( 1, 2, ? ), 5 };

 // matches any sequence of 4 integers that ends with 5 and contains 1 and 2 at least once in

 // other positions

 **template** MySequenceOfType MyTemplate3 := { **permutation** ( 1, 2, 3 ), \* };

 // matches any sequence of integers starting with 1,2,3; 1,3,2; 2,1,3; 2,3,1; 3,1,2 or 3,2,1

 **template** MySequenceOfType MyTemplate4 := { \*, **permutation** ( 1, 2, 3 )};

 // matches any sequence of integers ending with 1,2,3; 1,3,2; 2,1,3; 2,3,1; 3,1,2 or 3,2,1

 **template** MySequenceOfType MyTemplate5 := { \*, **permutation** ( 1, 2, 3 ),\* };

 // matches any sequence of integers containing any of the following substrings at any position:

 // 1,2,3; 1,3,2; 2,1,3; 2,3,1; 3,1,2 or 3,2,1

 **template** MySequenceOfType MyTemplate6 := { **permutation** ( 1, 2, \* ), 5 };

 // matches any sequence of integers that ends with 5 and containing 1 and 2 at least once in

 // other positions

 **template** MySequenceOfType MyTemplate7 := { **permutation** ( 1, 2, 3 ), \* length (0..5)};

 // matches any sequence of three to eight integers starting with 1,2,3; 1,3,2; 2,1,3; 2,3,1;

 // 3,1,2 or 3,2,1

 **template integer** MyInt1 := (1,2,3);

 **template integer** MyInt2 := (1,2,?);

 **template integer** MyInt3 := ?;

 **template integer** MyInt4 := \*;

 **template** MySequenceOfType MyTemplate10 := { **permutation** (MyInt1, 2, 3 ), 5 };

 // matches any of the sequences of 4 integers:

 // 1,3,2,5; 2,1,3,5; 2,3,1,5; 3,1,2,5; or 3,2,1,5;

 // 2,3,2,5; 2,2,3,5; 2,3,2,5; 3,2,2,5; or 3,2,2,5;

 // 3,3,2,5; 2,3,3,5; 2,3,3,5; 3,3,2,5; or 3,2,3,5;

 **template** MySequenceOfType MyTemplate11 := { **permutation** (MyInt2, 2, 3 ), 5 };

 // matches any sequence of 4 integers that ends with 5 and contains 2 and 3 at least once in

 // other positions

 **template** MySequenceOfType MyTemplate12 := { **permutation** (MyInt3, 2, 3 ), 5 };

 // matches any sequence of 4 integers that ends with 5 and contains 2 and 3 at least once in

 // other positions

 **template** MySequenceOfType MyTemplate13 := { **permutation** (MyInt4, 2, 3 ), 5 };

 // matches any sequence of integers that ends with 5 and containing 2 and 3 at least once in

 // other positions

 **template** MySequenceOfType MyTemplate14 := { **permutation** (MyInt3, 2, ? ), 5 };

 // matches any sequence of 4 integers that ends with 5 and contains 2 at least once in

 // other positions

 **template** MySequenceOfType MyTemplate15 := { **permutation** (MyInt4, 2, \* ), 5 };

 // matches any sequence of integers that ends with 5 and contains 2 at least once in

 // other positions

EXAMPLE 2:

 **type record of integer** RoI;

 **template** RoIt\_RoI1 := {1, 2, \*};

 **template** RoIt\_RoI2 := {**permutation**(0, **all from** t\_RoI1), 4, 5};

 // results in {permutation(0, 1, 2, \*), 4, 5}