### B.1.2.6 SuperSet

SuperSet is denoted by the keyword **superset**. SuperSet matches a set of values if, and only if, the set of values contains at least all of the elements defined within the SuperSet, and may contain more. This argument may contain templates (including template variables) and matching mechanisms with the restrictions given below. However, the length matching attribute may be attached to the SuperSet itself. In order to produce a successful match, individual items of the SuperSet shall match distinct elements in the set of value.

NOTE: There exists an one-to-one mapping from the superset-elements to the elements of the set-of-value so that the superset-element matches the set-of-element it is mapped to.

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

***Restrictions***

a) SuperSet is an operation for matching that shall be used only on values of **set of** types.

b) Individual members of the SuperSet's argument shall be of the type replicated by the **set of**.

c) The member type of the set of associated with the SuperSet template and the member type of the template in the **all from** clause shall be compatible.

d) The template in the **all from** clause as a whole shall not resolve into a matching mechanism (i.e. its elements may contain any of the matching mechanisms or matching attributes with the exception of those described in the following restriction).

e) The individual members of the SuperSet's argument and the elements of the template in the **all from** clause shall not be the matching mechanisms omit, SuperSet, SubSet and the matching attributes (length restriction and ifpresent). In addition, the individual members shall not resolve to AnyValueOrNone and individual elements of the template in the **all from** clause shall not resolve to AnyElementsOrNone or permutation.

f) If the length matching attribute is attached to the SuperSet, the minimal length allowed by the length attribute shall not be less than the number of the elements in the SuperSet.

***Examples***

EXAMPLE 1:

**type set of integer** MySetOfType (0 .. 10);

**template** MySetOfType MyTemplate1 := **superset** (1, 2, 3);

// matches any sequence of integers which contains at least one occurrences of the numbers

// 1, 2 and 3 in any order and position

**template** MySetOfType MyTemplate2\_AnyValue := **superset** (1, 2, ?)**;**

// matches any sequence of integers which contains at least one occurrences of the numbers

// 1, 2 and at least one more valid integer value (i.e. between 0 and 10, inclusively), in any

// order and position

**template** MySetOfType MyTemplate3 := **superset** (1, 2, (3, 4));

// matches any sequence of integers which contains at least one occurrences of the numbers

// 1, 2 and a number with the value 3 or 4, in any order and position

**template** MySetOfType MyTemplate4 := **superset** (1, 2, **complement**(3, 4));

// any sequence of integers matches which contains at least one occurrences of the numbers

// 1, 2 and a valid integer value which is not 3 or 4, in any order and position

**template** MySetOfType MyTemplate6 := **superset** (1, 2, 3) **length** (7);

// matches any sequence of 7 integers which contains at least one occurrences of the numbers

// 1, 2 and 3 in any order and position

**template** MySetOfType MyTemplate7 := **superset** (1, 2, ?) **length** (7 .. **infinity**);

// matches any sequence of at least 7 integers which contains at least one occurrences of the

// numbers 1, 2 and 3 in any order and position

**template** MySetOfType MyTemplate8 := **superset** (1, 2, 3) **length** (2 .. 7);

// causes an error, the lower bound of the length attribute contradicts to the minimum number

// of elements imposed by the superset argument

EXAMPLE 2:

**type record of integer** RoI;

**type set of integer** SoI;

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

**template** SoIt\_SoI1 := **superset**(**all from** t\_RoI1);

// results in superset(1, 2, ?)

### B.1.2.7 SubSet

SubSet is denoted by the keyword **subset**. SubSet matches a set of values if, and only if, the set of values contains only elements defined within the SubSet, and may contain less. This argument may contain templates (including template variables) and matching mechanisms with the restrictions given below. However, the length matching attribute may be attached to the SubSet itself. In order to produce a successful match, individual items of the SubSet shall match at most one element in the set of values.

NOTE: There exists a one-to-one mapping from the elements of the set-of-value to the superset-elements so that the set-of element is matched by the superset-element it is mapped to.

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

***Restrictions***

a) SubSet is an operation for matching that can be used only on values of **set of** types.

b) Individual members of the SubSet's argument shall be of the type replicated by the **set of**.

c) The member type of the set of type associated with the SubSet and the member type of the template in the **all from** clause shall be compatible.

d) The template in the **all from** clause as a whole shall not resolve into a matching mechanism (i.e. its elements may contain any of the matching mechanisms or matching attributes with the exception of those described in the following restriction).

e) The individual members of the SubSet's argument and the elements of the template in the **all from** clause shall not be the matching mechanisms omit, SuperSet, SubSet and the matching attributes (length restriction and ifpresent). In addition, individual members shall not resolve to AnyValueOrNone and individual fields of the template in the **all from** clause shall not resolve to AnyElementsOrNone or permutation.

f) If the length matching attribute is attached to the SubSet, the maximum length allowed by the length attribute shall not exceed the number of the elements in the SubSet.

***Examples***

EXAMPLE 1:

**template** MySetOfType MyTemplate1:= **subset** (1, 2, 3);

// matches any sequence of integers which contains zero or one occurrences of the numbers

// 1, 2 and 3 in any order and position

**template** MySetOfType MyTemplate1:= **subset** (1, 2, ?);

// matches any sequence of integers which contains zero or one occurrences of the numbers

// 1, 2 and a valid integer value (i.e. between 0 and 10, inclusive) in any order and position

**template** MySetOfType MyTemplate1:= **subset** (1, 2, (3, 4));

// matches any sequence of integers which contains zero or one occurrences of the numbers

// 1, 2 and one of the numbers 3 or 4, in any order and position

**template** MySetOfType MyTemplate1:= **subset** (1, 2, **complement** (3, 4));

// matches any sequence of integers which contains zero or one occurrences of the numbers

// 1, 2 and a valid integer number which is not 3 or 4, in any order and position

**template** MySetOfType MyTemplate1:= **subset** (1, 2, 3) **length** (2);

// matches any sequence of two integers which contains zero or one occurrences of

// the numbers 1, 2 and 3, in any order and position

**template** MySetOfType MyTemplate1:= **subset** (1, 2, ?) **length** (0 .. 2);

// matches any sequence of zero, one or two integers which contains zero or one occurrences of

// the numbers 1, 2 and of a valid integer value, in any order and position

**template** MySetOfType MyTemplate1:= **subset** (1, 2, 3) **length** (0 .. 4);

// causes an error, the upper bound of length attribute contradicts to the maximum number of

// elements imposed by the subset argument

EXAMPLE 2:

**type record of integer** RoI;

**type set of integer** SoI;

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

**template** SoIt\_SoI1 := **subset**(**all from** t\_RoI1);

// results in subset(1, 2, ?)