### 22.3.3 The Reply operation

The **reply** operation is used to reply to a call.

***Syntactical Structure***

*Port* "." **reply** "(" *TemplateInstance* [ **value** *TemplateBody* ] ")"

[ **to** *Address* ]

NOTE 1: *Address* may be an *AddressRef*, a list of *AddressRef*-s or "**all component**".

***Semantic Description***

The **reply** operation is used to reply to a previously accepted call according to the procedure signature.

NOTE 2: The relation between an accepted call and a **reply** operation cannot always be checked statically. For testing it is allowed to specify a **reply** operation without an associated **getcall** operation.

The value part of the **reply** operation consists of a signature reference with an associated actual parameter list and (optional) return value. The signature may either be defined in the form of a signature template or it may be defined in‑line.

Responses to one or more **call** operations may be sent to one, several or all peer entities connected to the addressed port. This can be specified in the same manner as described in clause 22.2.1. This means, the argument of the **to** clause of a **reply** operation is for unicast responses the address of one receiving entity, for multicast responses a list of addresses of a set of receivers and for broadcast responses the **all component** keywords.

In case of one-to-one connections, the **to** clause may be omitted, because the receiving entity is uniquely identified by the system structure.

A return value or template shall be explicitly stated with the **value** keyword and is first evaluated before returning.

***Restrictions***

In addition to the general static rules of TTCN‑3 given in clause 5 and shown in table 15, the following restrictions apply:

1. A **reply** operation shall only be used at a procedure-based port. The type definition of the port shall include the name of the procedure to which the **reply** operation belongs.
2. The *TemplateInstance* in the **reply** operation shall identify the signature definition and all signature parameters of the procedure to which the **reply** operation belongs.

c) All **out** and **inout** parameters of the signature shall have a specific value i.e. the use of matching mechanisms such as *AnyValue* is not allowed.

d) A **to** clause shall be present in case of one-to-many connections.

e)All *AddressRef* items in the **to** clause shall be of type **address**, **component** or of the address type bound to the port type (see clause 6.2.9) of the port instance referenced in the **reply** operation. No *AddressRef* in the **to** clause shall contain the special value **null** at the time of the operation.

f) The optional **value** clause for a return value shall only be present if the signature definition of the procedure to which the **reply** operation belongs defines a **return** type.

g) If a value is to be returned to the calling party, this shall be explicitly stated using the **value** keyword. The *TemplateBody* in the **value** clause shall conform to the template(value) restriction.

h) Applying a **reply** operation to an unmapped or disconnected port shall cause a test case error.

***Examples***

 myPort.**reply**(MyProc2:{ - ,5}); // Replies to an accepted call of MyProc2.

 myPort.**reply**(MyProc2:{ - ,5}) **to** myPeer; // Replies to an accepted call of MyProc2 from myPeer

 myPort.**reply**(MyProc2:{ - ,5}) **to** (myPeer1, myPeer2); // Multicast reply to myPeer1 and myPeer2

 myPort.**reply**(MyProc2:{ - ,5}) **to** **all component**; // Broadcast reply to all entities connected

 // to myPort

 myPort.**reply**(MyProc3:{5, v\_myVar} **value** 20); // Replies to an accepted call of MyProc3.