

International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Z.163

(11/2007)

SERIES Z: LANGUAGES AND GENERAL SOFTWARE
ASPECTS FOR TELECOMMUNICATION SYSTEMS

Formal description techniques (FDT) – Testing and Test
Control Notation (TTCN)

Testing and Test Control Notation version 3: TTCN-3 graphical presentation format (GFT)

ITU-T Recommendation Z.163

ITU-T Z-SERIES RECOMMENDATIONS
LANGUAGES AND GENERAL SOFTWARE ASPECTS FOR TELECOMMUNICATION SYSTEMS

FORMAL DESCRIPTION TECHNIQUES (FDT)	
Specification and Description Language (SDL)	Z.100–Z.109
Application of formal description techniques	Z.110–Z.119
Message Sequence Chart (MSC)	Z.120–Z.129
Extended Object Definition Language (eODL)	Z.130–Z.139
User Requirements Notation (URN)	Z.150–Z.159
Testing and Test Control Notation (TTCN)	Z.160–Z.199
PROGRAMMING LANGUAGES	
CHILL: The ITU-T high level language	Z.200–Z.209
MAN-MACHINE LANGUAGE	
General principles	Z.300–Z.309
Basic syntax and dialogue procedures	Z.310–Z.319
Extended MML for visual display terminals	Z.320–Z.329
Specification of the man-machine interface	Z.330–Z.349
Data-oriented human-machine interfaces	Z.350–Z.359
Human-machine interfaces for the management of telecommunications networks	Z.360–Z.379
QUALITY	
Quality of telecommunication software	Z.400–Z.409
Quality aspects of protocol-related Recommendations	Z.450–Z.459
METHODS	
Methods for validation and testing	Z.500–Z.519
MIDDLEWARE	
Processing environment architectures	Z.600–Z.609

For further details, please refer to the list of ITU-T Recommendations.

ITU-T Recommendation Z.163

Testing and Test Control Notation version 3: TTCN-3 graphical presentation format (GFT)

Summary

ITU-T Recommendation Z.163 defines the graphical presentation format for the TTCN-3 core language as defined in ITU-T Recommendation Z.161. This presentation format uses a subset of Message Sequence Charts as defined in ITU-T Recommendation Z.120 with test specific extensions.

This Recommendation is based on the core TTCN-3 language defined in ITU-T Recommendation Z.161. It is particularly suited to display tests as GFTs. It is not limited to any particular kind of test specification.

The specification of other formats is outside the scope of this Recommendation.

Source

ITU-T Recommendation Z.163 was approved on 13 November 2007 by ITU-T Study Group 17 (2005-2008) under the ITU-T Recommendation A.8 procedure.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g. interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2008

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

CONTENTS

Page

1	Scope	1
2	References.....	1
3	Abbreviations.....	1
4	Overview	2
5	GFT language concepts	3
6	Mapping between GFT and TTCN-3 Core language	5
7	Module structure	5
8	GFT symbols	7
9	GFT diagrams	9
9.1	Common properties	9
9.2	Control diagram.....	10
9.3	Test case diagram	11
9.4	Function diagram.....	12
9.5	Altstep diagram	13
10	Instances in GFT diagrams	14
10.1	Control instance.....	14
10.2	Test component instances.....	14
10.3	Port instances.....	15
11	Elements of GFT diagrams.....	15
11.1	General drawing rules.....	15
11.2	Invoking GFT diagrams	17
11.3	Declarations.....	19
11.4	Basic program statements.....	21
11.5	Behavioural program statements	24
11.6	Default handling	29
11.7	Configuration operations	30
11.8	Communication operations.....	33
11.9	Timer operations.....	52
11.10	Test verdict operations	55
11.11	External actions	56
11.12	Specifying attributes.....	56
Annex A	– GFT BNF.....	57
A.1	Meta-language for GFT.....	57
A.2	Conventions for the syntax description	57
A.3	The GFT grammar	58
Annex B	– Reference guide for GFT	81

	Page
Annex C – Examples	110
C.1 The Restaurant example	110
C.2 The INRES example	119

Copyright © 2007 ITU-T. All rights reserved.

Introduction

The graphical presentation format of TTCN-3 (GFT) is based on [ITU-T Z.120] defining Message Sequence Charts (MSC). GFT uses a subset of MSC with test specific extensions. The majority of extensions are textual extensions only. Graphical extensions are defined to ease the readability of GFT diagrams. Where possible, GFT is defined like MSC, so that established MSC tools with slight modifications can be used for the graphical definition of TTCN-3 test cases in terms of GFT.

The core language of TTCN-3 is defined in [ITU-T Z.161] and provides a full text-based syntax, static semantics and operational semantics as well as a definition for the use of the language with ASN.1. The GFT presentation format provides an alternative way of displaying the core language (see Figure 1).

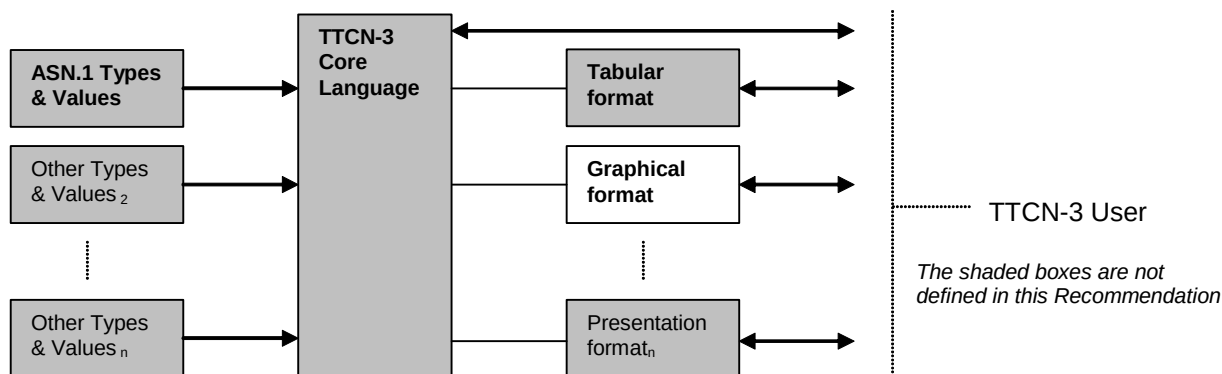


Figure 1 – User's view of the core language and the various presentation formats

The core language may be used independently of GFT. However, GFT cannot be used without the core language. Use and implementation of the GFT shall be done on the basis of the core language.

This Recommendation defines:

- the language concepts of GFT;
- the guidelines for the use of GFT;
- the grammar of GFT;
- the mapping from and to the TTCN-3 core language.

Together, these characteristics form the graphical presentation format of TTCN-3.

Copyright International Telecommunication Union/ITU Telecommunication Sector

ITU-T Recommendation Z.163

Testing and Test Control Notation version 3: TTCN-3 graphical presentation format (GFT)

1 Scope

This Recommendation defines the graphical presentation format for the TTCN-3 core language as defined in [ITU-T Z.161]. This presentation format uses a subset of Message Sequence Charts as defined in [ITU-T Z.120] with test specific extensions.

This Recommendation is based on the core TTCN-3 language defined in [ITU-T Z.161]. It is particularly suited to display tests as GFTs. It is not limited to any particular kind of test specification.

The specification of other formats is outside the scope of this Recommendation.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T X.292] ITU-T Recommendation X.292 (2002), *OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications – The Tree and Tabular Combined Notation (TTCN)*.

ISO/IEC 9646-3:1998, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 3: The Tree and Tabular Combined Notation (TTCN)*.

[ITU-T Z.120] ITU-T Recommendation Z.120 (2004), *Message sequence chart (MSC)*.

[ITU-T Z.161] ITU-T Recommendation Z.161 (2007), *Testing and Test Control Notation version 3: TTCN-3 core language*.

[ITU-T Z.162] ITU-T Recommendation Z.162 (2007), *Testing and Test Control Notation version 3: TTCN-3 tabular presentation format (TFT)*.

3 Abbreviations

This Recommendation uses the following abbreviations and acronyms:

BNF	Backus-Naur Form
CATG	Computer Aided Test Generation
GFT	Graphical presentation Format of TTCN-3
MSC	Message Sequence Chart
MTC	Main Test Component
PTC	Parallel Test Component
SUT	System Under Test

TFT Tabular presentation Format of TTCN-3

TTCN Testing and Test Control Notation

4 Overview

According to the OSI conformance testing methodology defined in [ITU-T X.292], testing normally starts with the identification of test purposes. A test purpose is defined as:

"A prose description of a well-defined objective of testing, focusing on a single conformance requirement or a set of related conformance requirements as specified in the appropriate OSI specification".

Having identified all test purposes an abstract test suite is developed that consists of one or more abstract test cases. An abstract test case defines the actions of the tester processes necessary to validate part (or all) of a test purpose.

Applying these terms to Message Sequence Charts (MSCs) we can define two categories for their usage:

- 1) *Using MSCs for the definition of test purposes* – Typically, an MSC specification that is developed as a use-case or as part of a system specification can be viewed as test purpose, i.e., it describes a requirement for the SUT in the form of a behaviour description that can be tested. For example, Figure 2 presents a simple MSC describing the interaction between instances representing the SUT and its interfaces A, B and C. In a real implementation of such a system the interfaces A, B and C may map onto service access points or ports. The MSC in Figure 2 only describes the interaction with the SUT and does not describe the actions of the test components necessary to validate the SUT behaviour, i.e., it is a test purpose description.

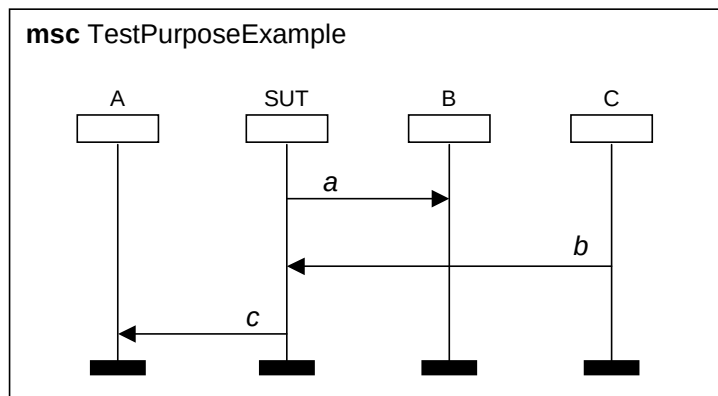


Figure 2 – MSC describing the interaction of an SUT with its interfaces

- 2) *Using MSCs for the definition of abstract test cases* – An MSC specification describing an abstract test case specifies the behaviour of the test components necessary to validate a corresponding test purpose. Figure 3 presents a simple MSC abstract test case description. It shows one Main Test Component (MTC) that exchanges the messages *a*, *b* and *c* with the SUT via the ports PortA, PortB and PortC in order to reach the test purpose shown in Figure 2. The messages *a* and *c* are sent by the SUT via the ports A and B (Figure 2) and received by the MTC (Figure 3) via the same ports. The message *b* is sent by the MTC and received by the SUT.

NOTE – The examples in Figures 2 and 3 are only simple examples to illustrate the different usages of MSC for testing. The diagrams will be more complicated in case of a distributed SUT that consists of several processes or a distributed test configuration with several test components.

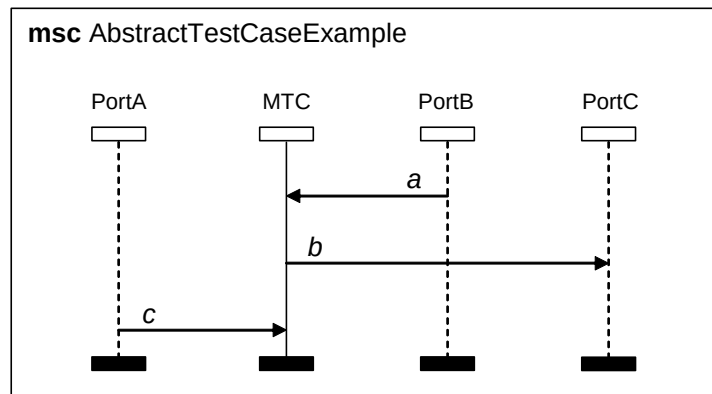


Figure 3 – MSC describing the interaction of an MTC with SUT interfaces

In identifying these two categories of MSC usage two distinct areas of work can be identified (see Figure 4):

- a) *Generation of abstract test cases from MSC test purpose descriptions* – TTCN-3 core language or GFT may be used to represent the abstract test cases. However, it is perceived that test case generation from test purposes is non-trivial and involves the usage and development of Computer Aided Test Generation (CATG) techniques.
- b) *Development of a Graphical presentation format for TTCN-3 (GFT) and definition of the mapping between GFT and TTCN-3.*

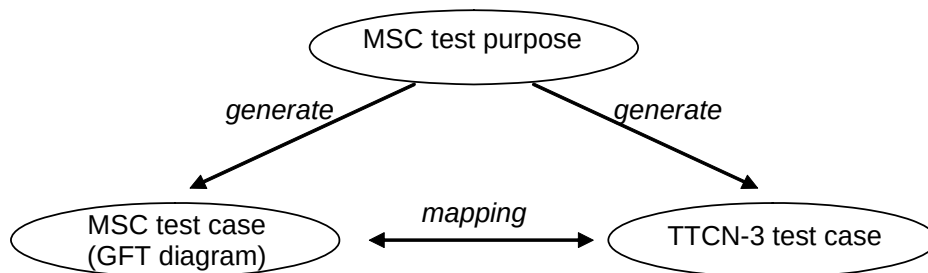


Figure 4 – Relations between MSC test purpose description, MSC test case descriptions and TTCN-3

This Recommendation focuses on item b), i.e., it defines GFT and the mapping between GFT and the TTCN-3 core language.

5 GFT language concepts

GFT represents graphically the behavioural aspects of TTCN-3 like the behaviour of a test case or a function. It does not provide graphics for data aspects like declaration of types and templates.

GFT defines no graphical representation for the structure of a TTCN-3 module, but specifies the requirements for such a graphical representation (see also clause 7).

NOTE – The order and the grouping of definitions and declarations in the module definitions part define the structure of a TTCN-3 module.

GFT defines no graphical representation for:

- module parameter definitions;
- import definitions;
- type definitions;
- signature declarations;
- template declarations;
- constant declarations;
- external constant declarations; and
- external function declarations.

TTCN-3 definitions and declarations without a corresponding GFT presentation may be presented in the TTCN-3 core language or in the tabular presentation format for TTCN-3 (TFT) ([ITU-T Z.162]).

GFT provides graphics for TTCN-3 behaviour descriptions. This means a GFT diagram provides a graphical presentation of either:

- the control part of a TTCN-3 module;
- a TTCN-3 test case;
- a TTCN-3 function; or
- a TTCN-3 altstep.

The relation between a TTCN-3 module and a corresponding GFT presentation is shown in Figure 5.

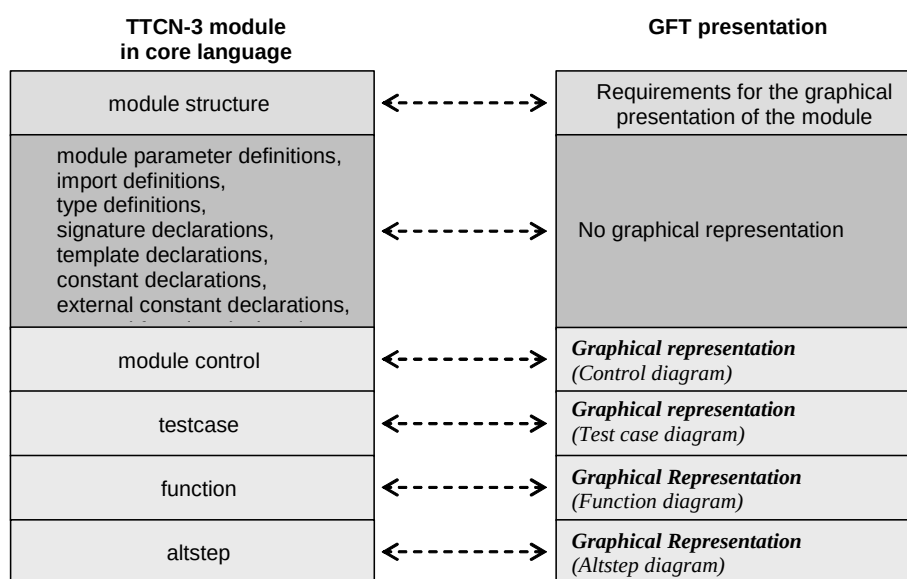


Figure 5 – Relation between TTCN-3 core language and the corresponding GFT description

GFT is based on MSC ([ITU-T Z.120]) and, thus, a GFT diagram maps onto an MSC diagram. Although GFT uses most of the graphical MSC symbols, the inscriptions of some MSC symbols have been adapted to the needs of testing and, in addition, some new symbols have been defined in order to emphasize test specific aspects. Though, the new symbols can be mapped onto valid MSC.

- the representation of port instances;

- the creation of test components;
- the start of test components;
- the return from a function call;
- the repetition of alternatives;
- the time supervision of a procedure-based call;
- the execution of test cases;
- the activation and deactivation of defaults;
- the labelling and goto; and
- the timers within call statements.

A complete list of all symbols used in GFT is presented in clause 8.

6 Mapping between GFT and TTCN-3 Core language

GFT provides graphical means for TTCN-3 behaviour definitions. The control part and each function, altstep and test case of a TTCN-3 core language module can be mapped onto a corresponding GFT diagram and vice versa. This means:

- the module control part can be mapped onto a control diagram (see clause 9.2) and vice versa;
- a test case can be mapped onto a test case diagram (see clause 9.3) and vice versa;
- a function in core language can be mapped onto a function diagram (see clause 9.4) and vice versa;
- an altstep can be mapped onto an altstep diagram (see clause 9.5) and vice versa.

NOTE 1 – GFT provides no graphical presentations for definitions of module parameters, types, constants, signatures, templates, external constants and external functions in the module definitions part. These definitions may be presented directly in core language or by using another presentation format, e.g., the tabular presentation format.

Each declaration, operation and statement in the module control and each test case, altstep or function can be mapped onto a corresponding GFT representation and vice versa.

The order of declarations, operations and statements within a module control, test case, altstep or function definition is identical to the order of the corresponding GFT representations within the related control, test case, altstep or function diagram.

NOTE 2 – The order of GFT constructs in a GFT diagram is defined by the order of the GFT constructs in the diagram header (declarations only) and the order of the GFT constructs along the control instance (control diagram) or component instance (test case diagram, altstep diagram or function diagram).

7 Module structure

As shown in Figure 6, a TTCN-3 module has a tree structure. A TTCN-3 module is structured into a module definitions part and a module control part. The module definitions part consists of definitions and declarations that may be structured further by means of groups. The module control part cannot be structured into sub-structures; it defines the execution order and the conditions for the execution of the test cases.

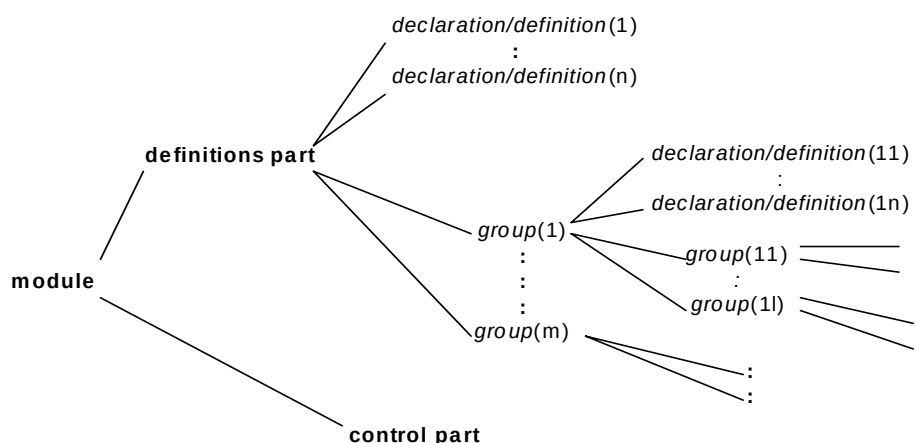


Figure 6 – Structure of TTCN-3 modules

GFT provides diagrams for all "behavioural" leaves of the module tree structure, i.e., for the module control part, for functions, for altsteps and for test cases. GFT defines no concrete graphics for the module tree-structure, however appropriate tool support for GFT requires a graphical presentation of the structure of a TTCN-3 module. The TTCN-3 module structure may be provided in form of an organizer view (Figure 7) or the MSC document-like presentation (Figure 8). An advanced tool may even support different presentations of the same object, e.g., the organizer view in Figure 7 indicates that some definitions are provided within several presentation formats, e.g., function MySpecialFunction is available in core language, in form of a TFT table and as GFT diagram.

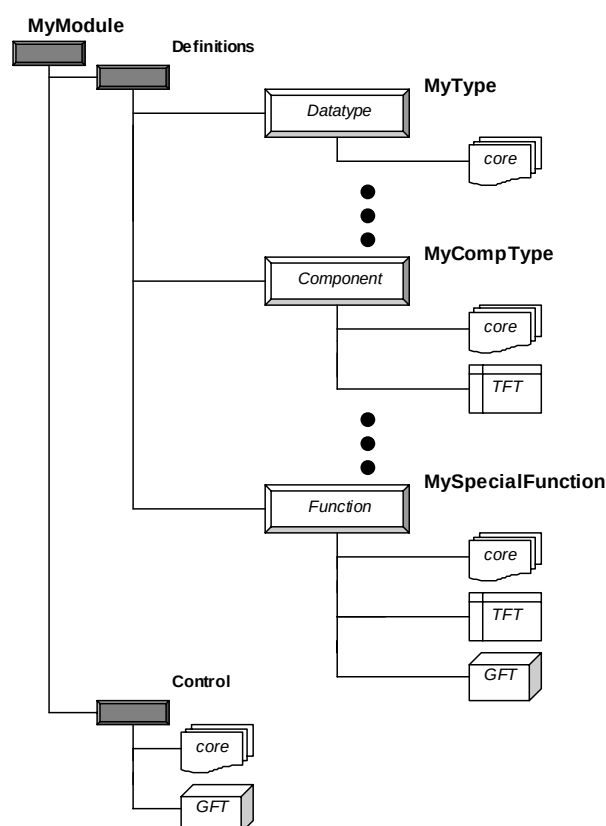


Figure 7 – Various presentation formats in an organizer view of a TTCN-3 module structure

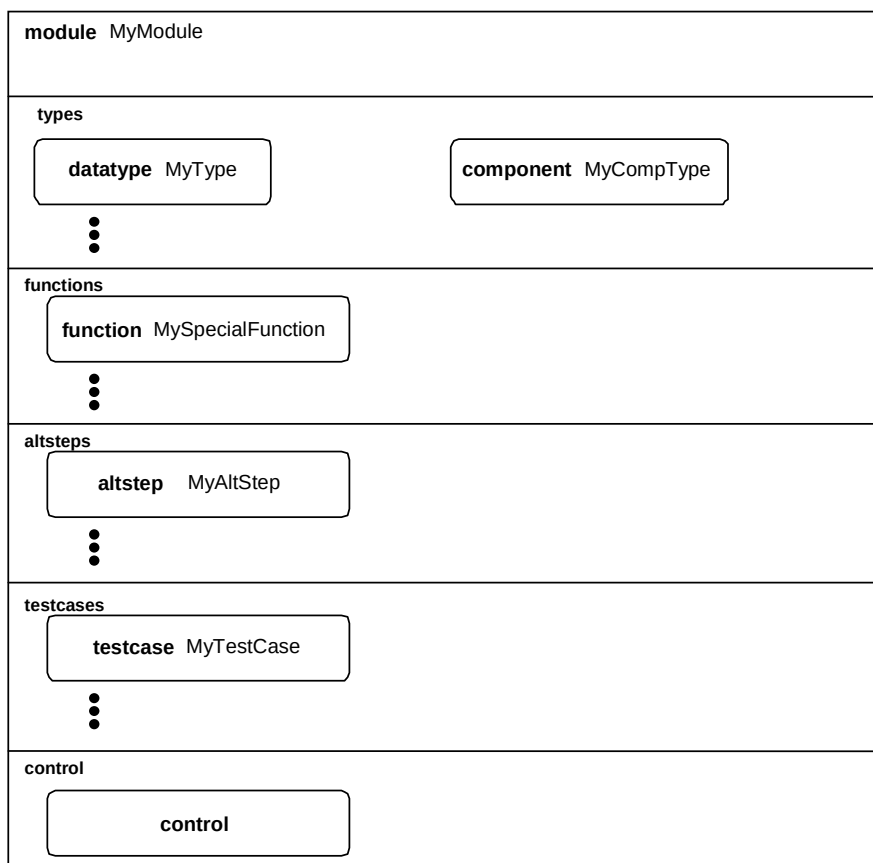


Figure 8 – Graphical MSC document-like presentation of a TTCN-3 module structure

8 GFT symbols

This clause presents all graphical symbols used within GFT diagrams and comments their typical usage within GFT.

Table 1 – GFT symbols

GFT element	Symbol	Description
Frame symbol		Used to frame GFT diagrams
Reference symbol		Used to represent the invocation of functions and altsteps
Port instance symbol		Used to represent port instances

Table 1 – GFT symbols

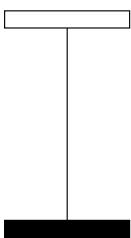
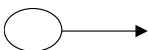
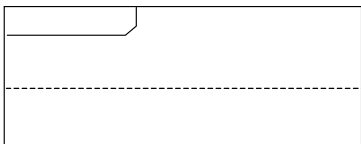
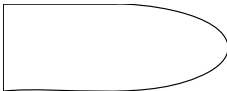
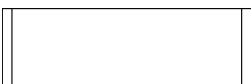
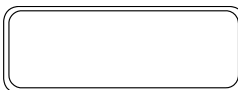
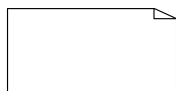
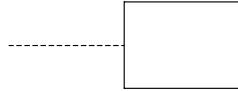
GFT element	Symbol	Description
Component instance symbol		Used to represent test components and the control instance
Action box symbol		Used for textual TTCN-3 declarations and statements, to be attached to a component symbol
Condition symbol		Used for textual TTCN-3 boolean expressions, verdict setting, port operations (start, stop and clear) and the done statement, to be attached to a component symbol
Labelling symbol		Used for TTCN-3 labelling and goto, to be attached to a component symbol
Goto symbol		Used for TTCN-3 labelling and goto, to be attached to a component symbol
Inline expression symbol		Used for TTCN-3 if-else, for, while, do-while, alt, call and interleave statement, to be attached to a component symbol
Default symbol		Used for TTCN-3 activate and deactivate statement, to be attached to a component symbol
Stop symbol		Used for TTCN-3 stop statement, to be attached to a component symbol
Return symbol		Used for TTCN-3 return statement, to be attached to a component symbol
Repeat symbol		Used for TTCN-3 repeat statement, to be attached to a component symbol
Create symbol		Used for TTCN-3 create statement, to be attached to a component symbol
Start symbol		Used for TTCN-3 start statement, to be attached to a component symbol
Message symbol		Used for TTCN-3 send, call, reply, raise, receive, getcall, getreply, catch, trigger and check statement, to be attached to a component symbol and a port symbol
Found symbol		Used for representing TTCN-3 receive, getcall, getreply, catch, trigger and check from any port, to be attached to a component symbol

Table 1 – GFT symbols

GFT element	Symbol	Description
Suspension region symbol		Used in combination with a blocking call, to be within a call inline expression and attached to a component symbol
Start timer symbol		Used for TTCN-3 start timer operation, to be attached to a component symbol
Timeout timer symbol		Used for TTCN-3 timeout operation, to be attached to a component symbol
Stop timer symbol		Used for TTCN-3 stop timer operation, to be attached to a component symbol
Start implicit timer symbol		Used for TTCN-3 implicit timer start in blocking call, to be within a call inline expression and attached to a component symbol
Timeout implicit timer symbol		Used for TTCN-3 timeout exception in blocking call, to be within a call inline expression and attached to a component symbol
Execute symbol		Used for TTCN-3 execute test case statement, to be attached to a component instance symbol
Text symbol		Used for TTCN-3 with statement and comments, to be placed within a GFT diagram
Event comment symbol		Used for TTCN-3 comments associated to events, to be attached to events on component instance or port instance symbols

9 GFT diagrams

GFT provides the following diagram types:

- control diagram* for the graphical presentation of a TTCN-3 module control part;
- test case diagram* for the graphical presentation of a TTCN-3 test case;
- altstep diagram* for the graphical presentation of a TTCN-3 altstep; and
- function diagram* for the graphical presentation of a TTCN-3 function.

The different diagram types have some common properties.

9.1 Common properties

Common properties of GFT diagrams are related to the diagram area, diagram heading and paging.

9.1.1 Diagram area

Each GFT control, test case, altstep and function diagram shall have a frame symbol (also called diagram frame) to define the diagram area. All symbols and text needed to define a complete and syntactically correct GFT diagram shall be made inside the diagram area.

NOTE – GFT has no language constructs like the MSC gates, which are placed outside of, but connected to the diagram frame.

9.1.2 Diagram heading

Each GFT diagram shall have a diagram heading. The diagram heading shall be placed in the upper left-hand corner of the diagram frame.

The diagram heading shall uniquely identify each GFT diagram type. The general rule to achieve this is to construct the heading from the keywords **testcase**, **altstep** or **function** followed by the TTCN-3 signature of the test case, altstep or function that should be presented graphically. For a GFT control diagram, the unique heading is constructed from the keyword **module** followed by the module name.

NOTE – In MSC, the keyword **msc**. always precedes the diagram name to identify MSC diagrams. GFT diagrams do not have such a common keyword to identify GFT diagrams.

9.1.3 Paging

GFT diagrams may be organized in pages and a large GFT diagram may be split into several pages. Each page of a split diagram shall have a numbering in the upper right hand corner that identifies the page uniquely. The numbering is optional if the diagram is not split.

NOTE 1 – The concrete numbering scheme is considered to be a tools issue and is therefore outside the scope of this Recommendation. A simple numbering scheme may only assign a page number, whereas an advanced numbering scheme may support the reconstruction of a diagram only by using the numbering information on the different pages.

NOTE 2 – Paging requirements beyond the general numbering are considered to be tools issues and are therefore outside the scope of this Recommendation. For readability purposes, the diagram heading may be shown on each page, the instance line of an instance that will be continued on another page may be attached to the lower border of the page and the instance head of a continued instance may be repeated on the page that describes the continuation.

9.2 Control diagram

A GFT control diagram provides a graphical presentation of the control part of a TTCN-3 module. The heading of a control diagram shall be the keyword **module** followed by the module name. A GFT control diagram shall only include one component instance (also called control instance) with the instance name **control** without any type information. The control instance describes the behaviour of the TTCN-3 module control part. Attributes associated to the TTCN-3 module control part shall be specified within a text symbol in the control diagram. The principle shape of a GFT control diagram and the corresponding TTCN-3 core description are sketched in Figure 9.

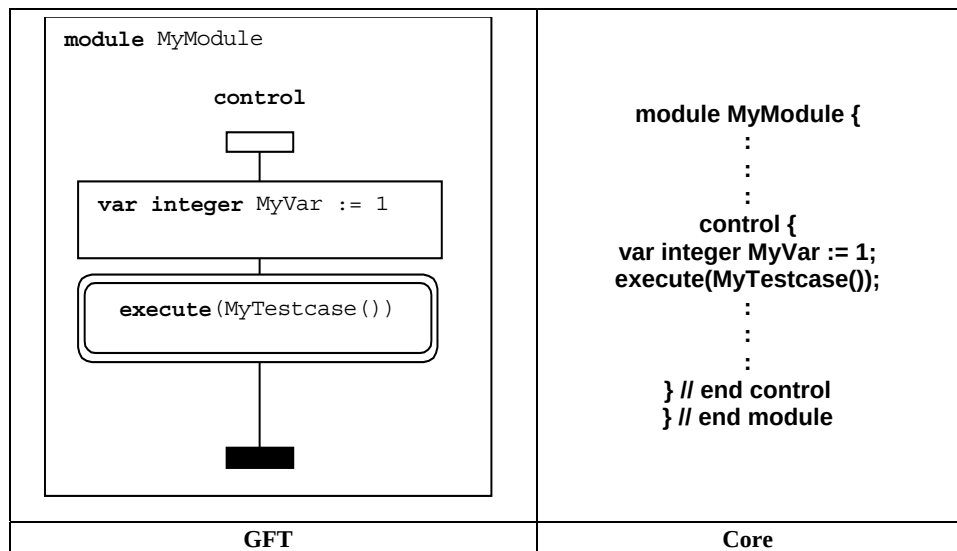


Figure 9 – Principle shape of a GFT control diagram and corresponding core language

Within the control part, test cases can be selected or deselected for the test case execution with the use of Boolean expressions. Expressions, assignments, **log** statements, **label** and **goto** statements, **if-else** statements, **for** loop statements, **while** loop statements, **do while** loop statements, **stop** execution statements, and timer statements can be used to control the execution of test cases. Furthermore, functions can be used to group the test cases together with their preconditions for execution, which are invoked by the module control part.

The GFT representation of those language features is as described in the respective clauses below except that for the module control part the graphical symbols are attached to the control instance and not to a test component instance.

Please refer to clause 11.4 for the GFT representation of expressions, assignments, **log**, **label** and **goto**, **if-else**, **for** loop, **while** loop, **do while** loop, and **stop**, to clause 11.9 for timer operations and to clauses 9.4 and 11.2.2 for functions and their invocation.

9.3 Test case diagram

A GFT test case diagram provides a graphical presentation of a TTCN-3 test case. The heading of a test case diagram shall be the keyword **testcase** followed by the complete signature of the test case. Complete means that at least test case name and parameter list shall be present. The **runs on** clause is mandatory and the **system** clause is optional in the core language. If the system clause is specified in the corresponding core language, it shall also be present in the heading of the test case diagram.

A GFT test case diagram shall include one test component instance describing the behaviour of the **mtc** (also called mtc instance) and one port instance for each port owned by the **mtc**. The name associated with the mtc instance shall be **mtc**. The type associated with the mtc instance is optional, but if the type information is present, it shall be identical to the component type referred to in the **runs on** clause of the test case signature. The names associated with the port instances shall be identical to the port names defined in the component type definition of the **mtc**. The associated type information for port instances is optional. If the type information is present, port names and port types shall be consistent with the component type definition of the **mtc**. The **mtc** and port types are displayed in the component or port instance head symbol.

Attributes associated to the test case presented in GFT shall be specified within a text symbol in the test case diagram. The principle shape of a GFT test case diagram and the corresponding TTCN-3 core description are sketched in Figure 10.

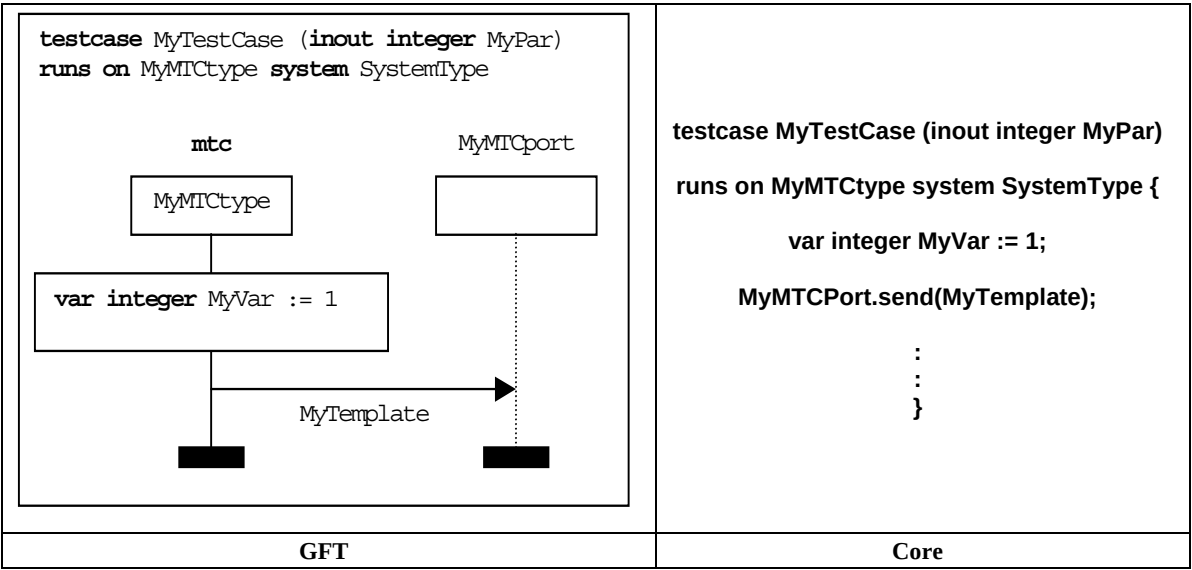


Figure 10 – Principle shape of a GFT test case diagram and corresponding core language

A test case represents the dynamic test behaviour and can create test components. A test case may contain declarations, statements, communication and timer operations and invocation of functions or altsteps.

9.4 Function diagram

GFT presents TTCN-3 functions by means of function diagrams. The heading of a function diagram shall be the keyword **function** followed by the complete signature of the function. Complete means that at least function name and parameter list shall be present. The **return** clause and the **runs on** clause are optional in the core language. If these clauses are specified in the corresponding core language, they shall also be present in the header of the function diagram.

A GFT function diagram shall include one test component instance describing the behaviour of the function and one port instance for each port usable by the function.

NOTE – The names and types of the ports that are usable by a function are passed in as parameters or are the port names and types that are defined in the component type definition referenced in the **runs on** clause.

The name associated with the test component instance shall be **self**. The type associated with the test component instance is optional, but if the type information is present, it shall be consistent with the component type in the **runs on** clause.

The names and types associated with the port instances shall be consistent with the port parameters (if the usable ports are passed in as parameters) or to the port declarations in the component type definition referenced in the **runs on** clause. The type information for port instances is optional.

self and port names are displayed on top of the component and resp. port instance head symbol. The component types and port types are displayed within the component and resp. port instance head symbol.

Attributes associated to the function presented in GFT shall be specified within a text symbol in the function diagram. The principle shape of a GFT function diagram and the corresponding TTCN-3 core description are sketched in Figure 11.

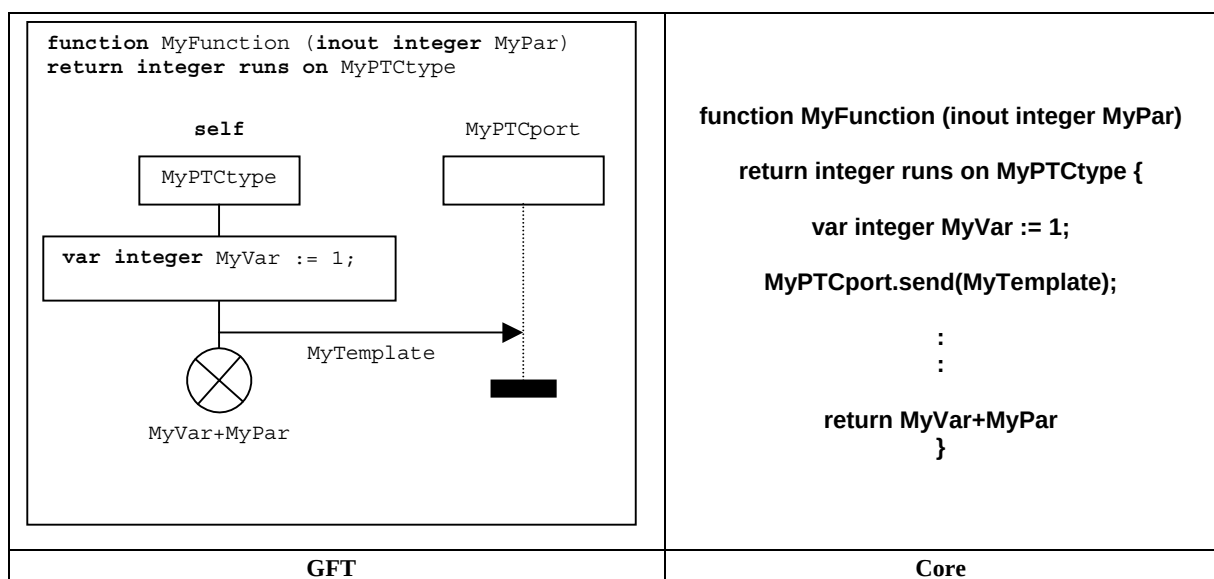


Figure 11 – Principle shape of a GFT function diagram and corresponding core language

A function is used to specify and structure test behaviour, define default behaviour or to structure computation in a module. A function may contain declarations, statements, communication and timer operations and invocation of function or altsteps and an optional return statement.

9.5 Altstep diagram

GFT presents TTCN-3 altsteps by means of altstep diagrams. The heading of an altstep diagram shall be the keyword **altstep** followed by the complete signature of the altstep. Complete means that at least altstep name and parameter list shall be present. The **runs on** clause is optional in the core language. If the **runs on** clause is specified in the corresponding core language, it shall also be present in the header of the altstep diagram.

A GFT altstep diagram shall include one test component instance describing the behaviour of the altstep and one port instance for each port usable by the altstep.

NOTE – The names and types of the ports that are usable by an altstep are passed in as parameters or are the port names and types that are defined in the component type definition referenced in the **runs on** clause.

The name associated with the test component instance shall be **self**. The type associated with the test component instance is optional, but if the type information is present, it shall be consistent with the component type in the **runs on** clause.

The names and types associated with the port instances shall be consistent with the port parameters (if the usable ports are passed in as parameters) or to the port declarations in the component type definition referenced in the **runs on** clause. The type information for port instances is optional.

self and port names are displayed on top of the component and resp. port instance head symbol. The component types and port types are displayed within the component and resp. port instance head symbol.

Attributes associated to the altstep shall be specified within a text symbol in the GFT altstep diagram. The principle shape of a GFT altstep diagram and the corresponding TTCN-3 core language are sketched in Figure 12.

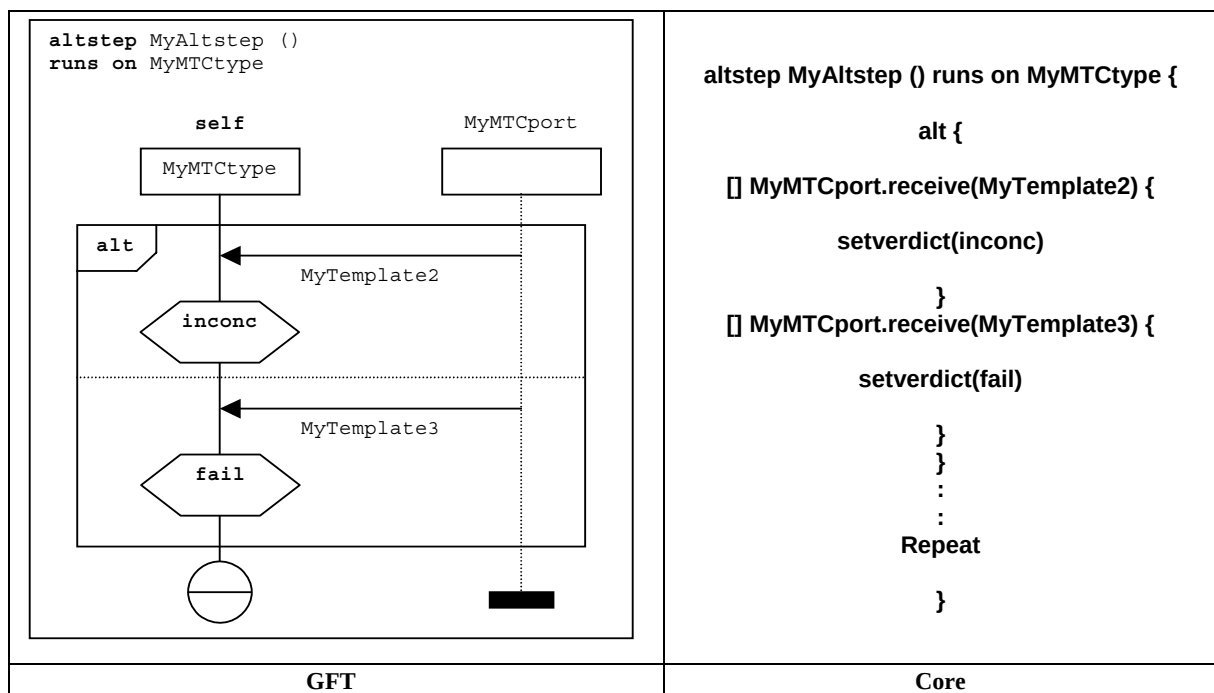


Figure 12 – Principle shape of a GFT altstep diagram and corresponding core language

An altstep is used to specify default behaviour or to structure the alternatives of an **alt** statement. An altstep may contain statements, communication and timer operations and invocation of function or altsteps.

10 Instances in GFT diagrams

GFT diagrams include the following kinds of instances:

- *control instances* describing the flow of control for the module control part;
- *test component instances* describing the flow of control for the test component that executes a test case, function or altstep;
- *port instances* representing the ports used by the different test components.

10.1 Control instance

Only one control instance shall exist within a GFT control diagram (see clause 9.2). A control instance describes the flow of control of a module control part. A GFT control instance shall graphically be described by a component instance symbol with the mandatory name **control** placed on top of the instance head symbol. No instance type information is associated with a control instance. The principle shape of a control instance is shown in Figure 13 a).

10.2 Test component instances

Each GFT test case, function or altstep diagram includes one test component instance that describes the flow of control of that instance. A GFT test component instance shall graphically be described by an instance symbol with:

- the mandatory name **mtc** placed on top of the instance head symbol in the case of a test case diagram;
- the mandatory name **self** placed on top of the instance head symbol in the case of a function or altstep diagram.

The optional test component type may be provided within the instance head symbol. It has to be consistent with the test component type given after the **runs on** keyword in the heading of the GFT diagram.

The principle shape of a test component instance in a test case diagram is shown in Figure 13 b). The principle shape of a test component instance in a function or altstep diagram is shown in Figure 13 c).

10.3 Port instances

GFT port instances may be used within test case, altstep and function diagrams. A port instance represents a port that is usable by the test component that executes the specified test case, altstep or function. A GFT port instance is graphically described by a component instance symbol with a dashed instance line. The name of the represented port is mandatory information and shall be placed on top of the instance head symbol. The port type (optional) may be provided within the instance head symbol. The principle shape of a port instance is shown in Figure 13 d).

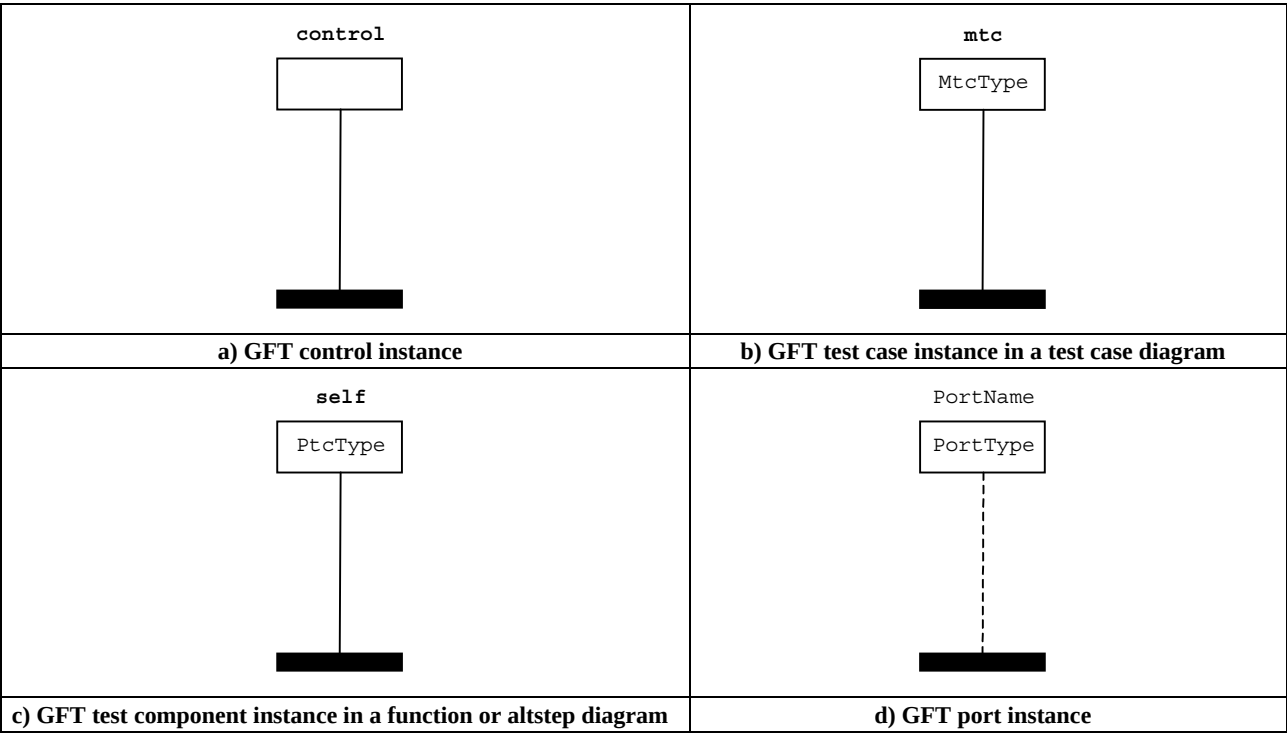


Figure 13 – Principle shape of instance kinds in GFT diagrams

11 Elements of GFT diagrams

This clause defines general drawing rules for the representation of specific TTCN-3 syntax elements (semicolons, comments). It describes how to display the execution of GFT diagrams and the graphical symbols associated with TTCN-3 language elements.

11.1 General drawing rules

General drawing rules in GFT are related to the usage of semicolons, TTCN-3 statements in action symbols and comments.

11.1.1 Usage of semicolons

All GFT symbols with the exception of the action symbol shall include only one statement in TTCN-3 core language. Only an action symbol may include a sequence of TTCN-3 statements (see clause 11.1.2).

The semicolon is optional if a GFT symbol includes only one statement in TTCN-3 core language (see Figure 14 a) and Figure 14 b)).

Semicolons shall separate the statements in a sequence of statements within an action symbol. The semicolon is optional for the last statement in the sequence (Figure 14 c)).

A sequence of variable, constant and timer declarations may also be specified in plain TTCN-3 core language following the heading of a GFT diagram. Semicolons shall also separate these declarations. The semicolon is optional for the last declaration in this sequence.

11.1.2 Usage of action symbols

The following TTCN-3 declarations, statements and operations are specified within action symbols: declarations (with the restrictions defined in clause 11.3), assignments, **log**, **connect**, **disconnect**, **map**, **unmap** and **action**.

A sequence of declarations, statements and operations that shall be specified within action symbols variable can be specified in a single action symbol. It is not necessary to use a separate action symbol for each declaration, statement or operation.

11.1.3 Comments

GFT provide three possibilities to put comments into GFT diagrams:

- Comments may be put into GFT symbols following the symbol inscription and using the syntax for comments of the TTCN-3 core language (Figure 14 d)).
- Comments in the syntax for comments of the TTCN-3 core language can be put into text symbols and freely placed in the GFT diagram area (Figure 14 e)).
- The comment symbol can be used to associate comments to GFT symbols. A comment in a comment symbol can be provided in form of free text, i.e., the comment delimiter `"/*"`, `"*/"` and `"//"` of the core language need not to be used (Figure 14 f)).

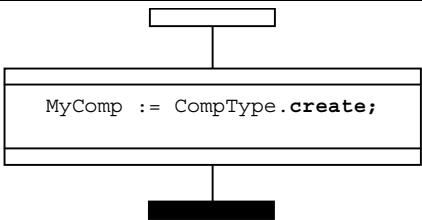
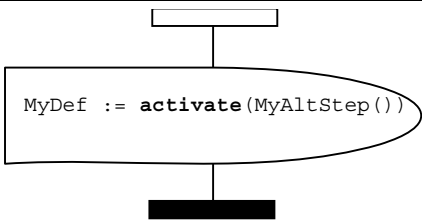
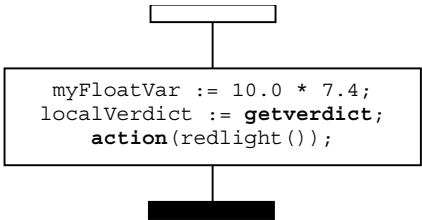
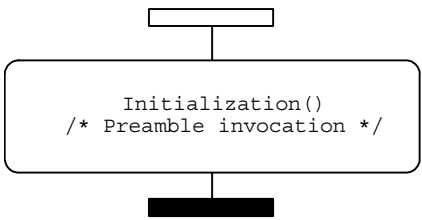
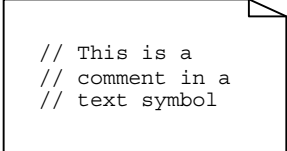
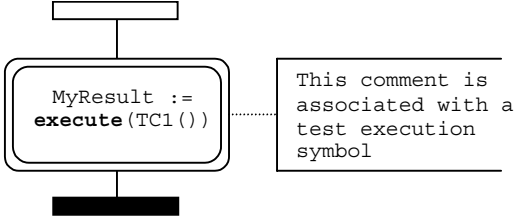
	
a) Component creation with an optional terminating semicolon	b) Default activation without a terminating semicolon
	
c) Sequence of statements in an action symbol	d) Comment within a GFT reference symbol
	
e) Comment in a text symbol	f) Comment within a comment symbol associated with an execution symbol

Figure 14 – Examples for the effects of the general drawing rules

11.2 Invoking GFT diagrams

This clause describes how the individual kinds of GFT diagrams are invoked. Since there is no statement for executing the control part in TTCN-3 (as it is comparable to executing a program via main and out of the scope of TTCN-3), the clause discusses the execution of test cases, functions, and altsteps.

11.2.1 Execution of test cases

The execution of test cases is represented by use of the execute test case symbol (see Figure 15). The syntax of the **execute** statement is placed within that symbol. The symbol may contain:

- an **execute** statement for a test case with optional parameters and time supervision;
- optionally, the assignment of the returned verdict to a **verdicttype** variable; and
- optionally, the inline declaration of the **verdicttype** variable.

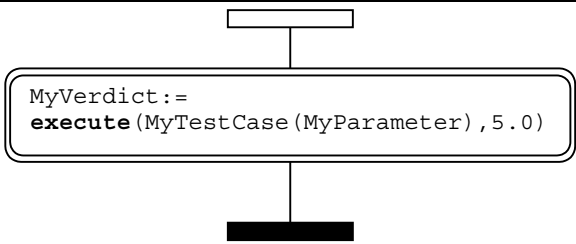
	MyVerdict := execute(MyTestCase(MyParameter),5.0);
GFT	Core

Figure 15 – Test case execution

11.2.2 Invocation of functions

The invocation of functions is represented by the reference symbol (Figure 16), except of external and predefined functions (Figure 17) and except where a function is called inside a TTCN-3 language element that has a GFT representation (Figure 18).

The syntax of the function invocation is placed within the reference symbol. The symbol may contain:

- the invocation of a function with optional parameters;
- an optional assignment of the returned value to a variable; and
- an optional inline declaration of the variable.

The reference symbol is only used for user defined functions defined within the current module. It shall not be used for external functions or predefined TTCN-3 functions, which shall be represented in their text form within an action form (Figure 17) or other GFT symbols (see example in Figure 18).

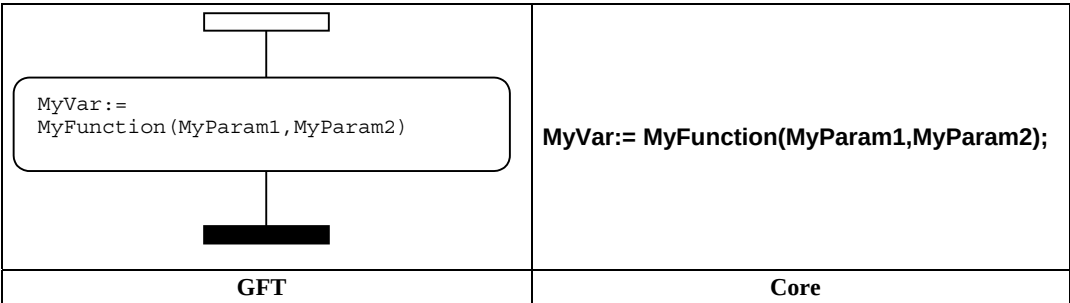


Figure 16 – Invocation of user defined function

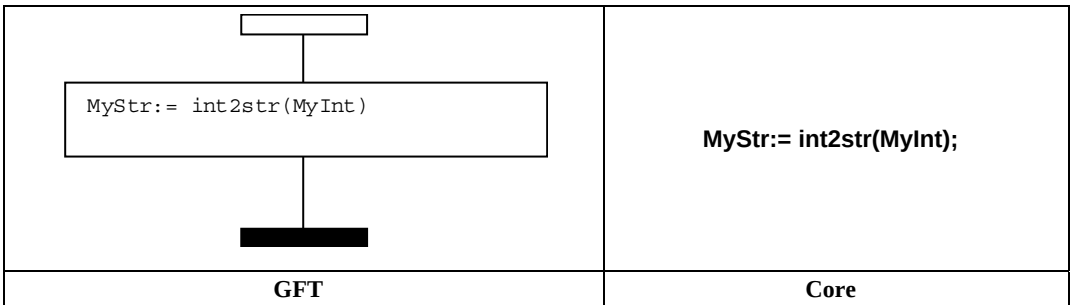


Figure 17 – Invocation of predefined/external function

Functions called inside a TTCN-3 construct with an associated GFT symbol are represented as text within that symbol.

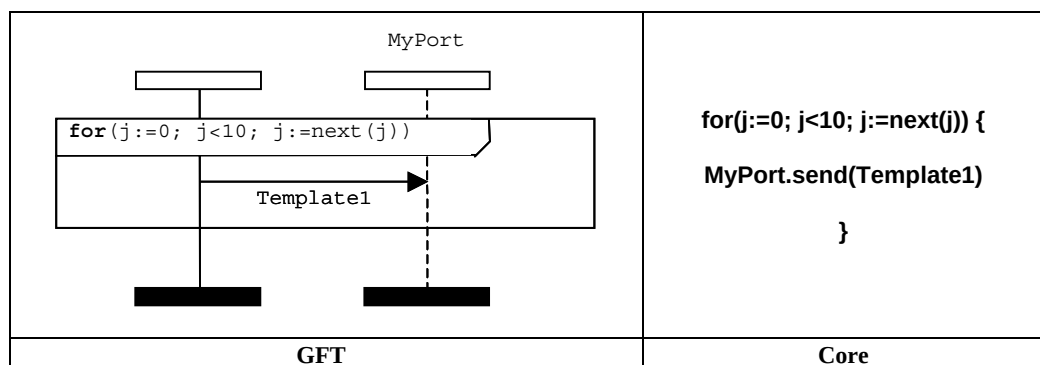


Figure 18 – Invocation of user defined function within GFT symbol

11.2.3 Invocation of altsteps

The invocation of altsteps is represented by use of the reference symbol (see Figure 19). The syntax of the altstep invocation is placed within that symbol. The symbol may contain the invocation of an altstep with optional parameters. It shall be used within alternative behaviour only, where the altstep invocation shall be one of the operands of the alternative statements (see also Figure 32 in clause 11.5.2).

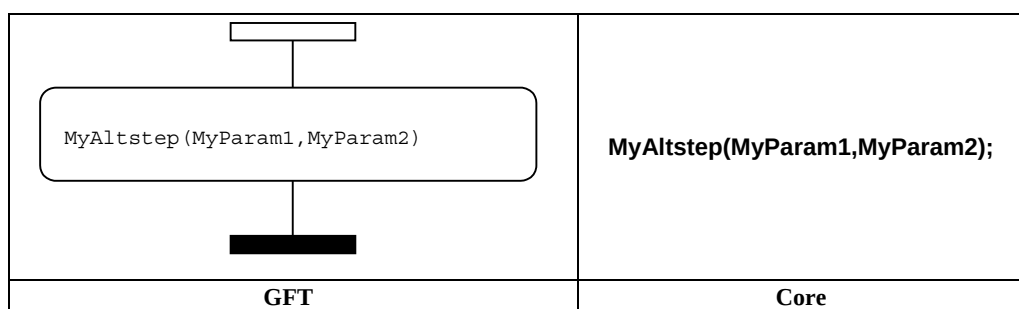


Figure 19 – Altstep invocation

Another possibility is the implicit invocation of altsteps via activated defaults. Please refer to clause 11.6.2 for further details.

11.3 Declarations

TTCN-3 allows the declaration and initialization of timers, constants and variables at the beginning of statement blocks. GFT uses the syntax of the TTCN-3 core language for declarations in several symbols. The type of a symbol depends on the specification of the initialization, e.g., a variable of type **default** that is initialized by means of an **activate** operation shall be specified within a default symbol (see clause 11.6).

11.3.1 Declaration of timers, constants and variables in action symbols

The following declarations shall be made within action symbols:

- timer declarations;
- declarations of variables without initialization;
- declarations of variables and constants with initialization;

- if the initialization is not made by means of functions that include communication functions; or
- if a declaration is:
 - of a component type that is not initialized by means of a **create** operation;
 - of type **default** that is not initialized by means of an **activate** operation;
 - of type **verdicttype** that is not initialized by means of an **execute** statement;
 - of a simple basic type;
 - of a basic string type;
 - of the type **anytype**;
 - of a port type;
 - of the type **address**; or
 - of a user-defined structured type with fields that fulfil all restrictions mentioned in this bullet for "declarations of variables and constants with initialization".

NOTE – Please refer to Table 3 of [ITU-T Z.161] for an overview on TTCN-3 types.

A sequence of declarations that shall be made within action symbols can be put into one action symbol and need not to be made in separate action symbols. Examples for declarations within action symbols can be found in Figures 20 a) and 20 b).

11.3.2 Declaration of constants and variables within inline expression symbols

Constants and variable declarations of a component type that are initialized within an **if-else**, **for**, **while**, **do-while**, **alt** or **interleave** statement shall be presented within the same inline expression symbol.

11.3.3 Declaration of constants and variables within create symbols

Constants and variable declarations of a component type that are initialized by means of **create** operations shall be made within a create symbol. In contrast to declarations within action symbols, each declaration that is initialized by means of a **create** operation shall be presented in a separate create symbol. An example for a variable declaration within a create symbol is shown in Figure 20 c).

11.3.4 Declaration of constants and variables within default symbols

Constants and variable declarations of type **default** that are initialized by means of **activate** operations shall be made within a default symbol. In contrast to declarations within action symbols, each declaration that is initialized by means of an **activate** operation shall be presented in a separate default symbol. An example for a variable declaration within a default symbol is shown in Figure 20 d).

11.3.5 Declaration of constants and variables within reference symbols

Constants and variable declarations that are initialized by means of a function, which includes communication operations, shall be made within reference symbols. In contrast to declarations within action symbols, each declaration that is initialized by means of a function, which includes communication functions, shall be presented in a separate reference symbol. An example for a variable declaration within a reference symbol is shown in Figure 20 e).

11.3.6 Declaration of constants and variables within execute test case symbols

Constants and variable declarations of type **verdicttype** that are initialized by means of **execute** statements shall be made within execute test case symbols. In contrast to declarations within action symbols, each declaration that is initialized by means of an **execute** statement shall be presented in a separate execute test case symbol. An example for a variable declaration within an execute test case symbol is shown in Figure 20 f).

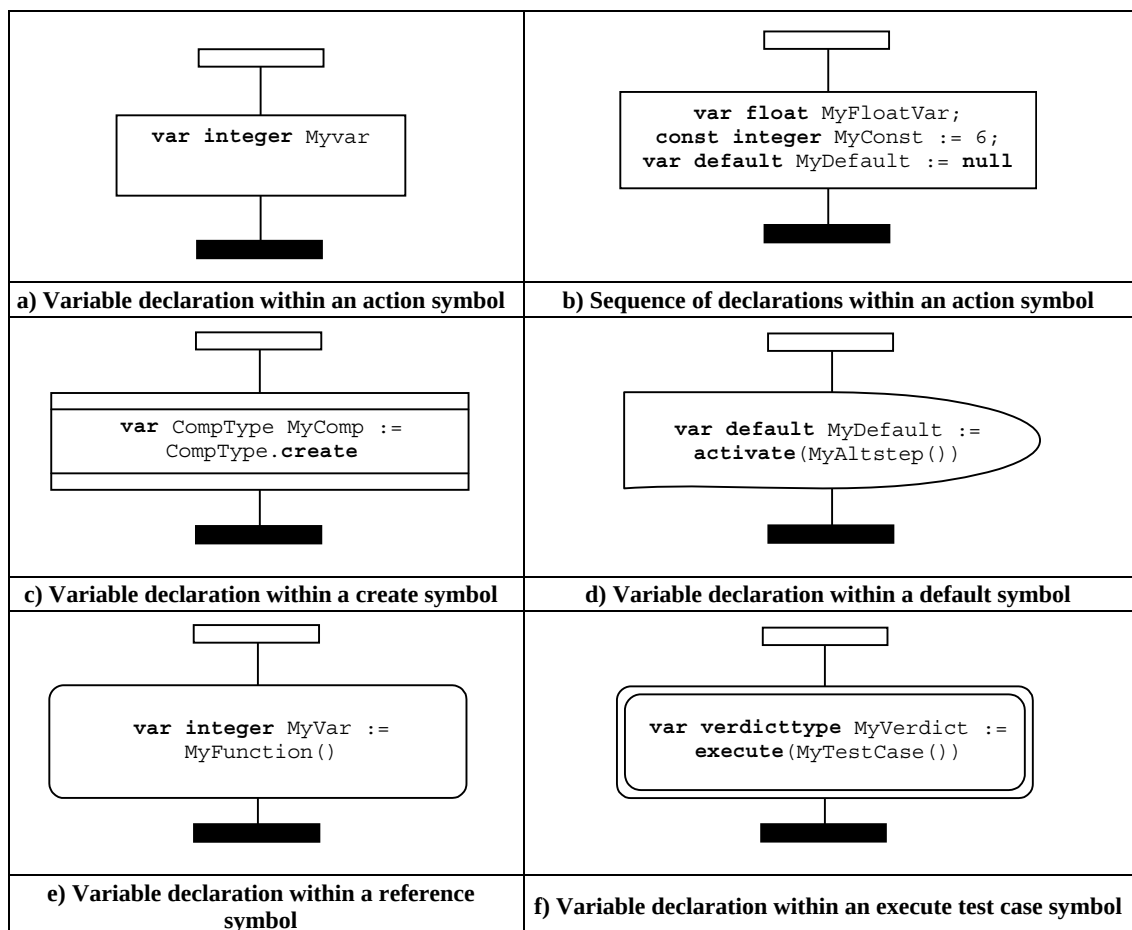


Figure 20 – Examples for declarations in GFT

11.4 Basic program statements

Basic program statements are expressions, assignments, operations, loop constructs, etc. All basic program statements can be used within GFT diagrams for the control part, test cases, functions and altsteps.

GFT does not provide any graphical representation for expressions and assignments. They are textually denoted at the places of their use. Graphics is provided for the **log**, **label**, **goto**, **if-else**, **for**, **while** and **do-while** statement.

11.4.1 The Log statement

The **log** statement shall be represented within an action symbol (see Figure 21).

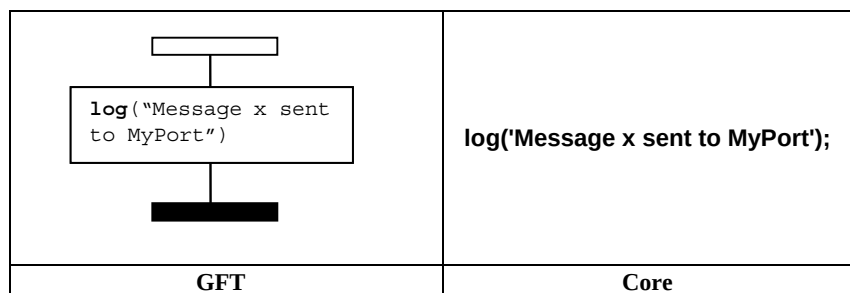


Figure 21 – Log statement

11.4.2 The Label statement

The **label** statement shall be represented with a label symbol, which is connected to a component instance. Figure 22 illustrates a simple example of a **label** named `MyLabel`.

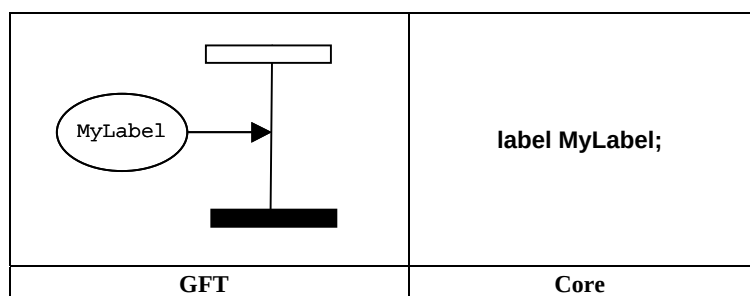


Figure 22 – Label statement

11.4.3 The Goto statement

The **goto** statement shall be represented with a goto symbol. It shall be placed at the end of a component instance or at the end of an operand in an inline expression symbol. Figure 23 illustrates a simple example of a **goto**.

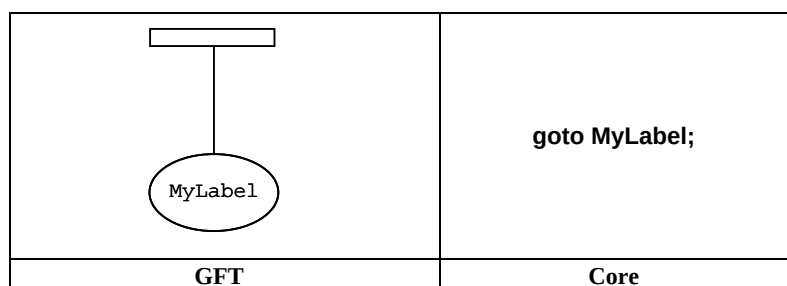


Figure 23 – Goto statement

11.4.4 The If-else statement

The **if-else** statement shall be represented by an inline expression symbol labelled with the **if** keyword and a Boolean expression as defined in clause 19.6 of [ITU-T Z.161]. The if-else inline expression symbol may contain one or two operands, separated by a dashed line. Figure 24 illustrates an **if** statement with a single operand, which is executed when the Boolean expression

$x > 1$ evaluates to true. Figure 25 illustrates an **if-else** statement in which the top operand is executed when the Boolean expression $x > 1$ evaluates to true, and the bottom operand is executed if the Boolean expression evaluates to false.

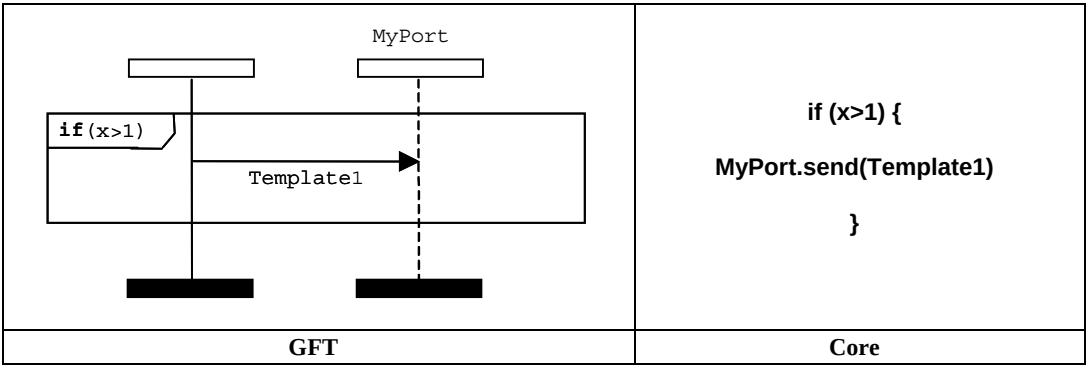


Figure 24 – If-statement

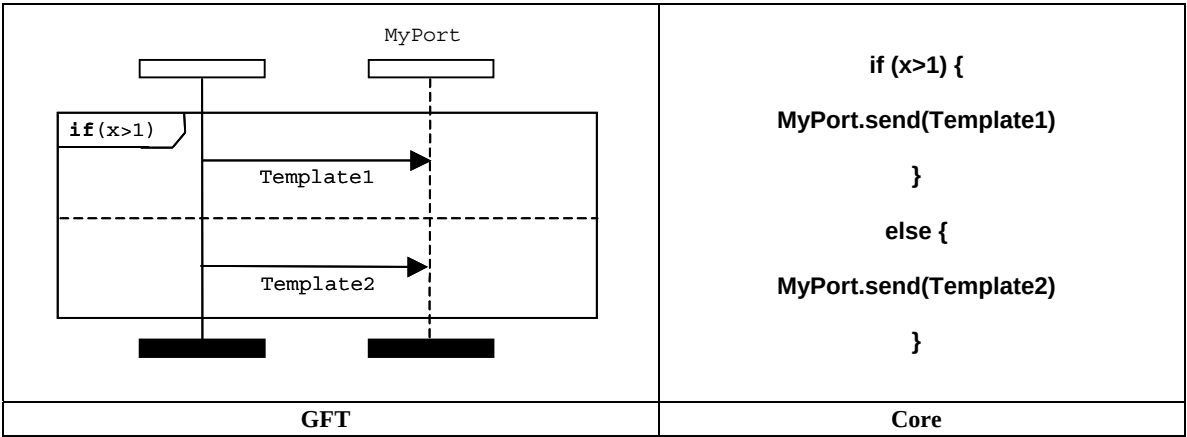


Figure 25 – If-else statement

11.4.5 The For statement

The **for** statement shall be represented by an inline expression symbol labelled with a **for** definition as defined in clause 19.7 of [ITU-T Z.161]. The **for** body shall be represented as the operand of the for inline expression symbol. Figure 26 represents a simple **for** loop in which the loop variable is declared and initialized within the **for** statement.

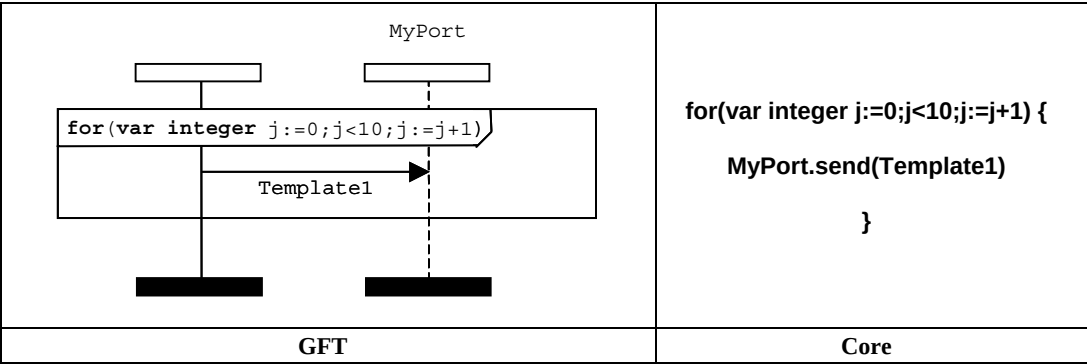


Figure 26 – For statement

11.4.6 The While statement

The **while** symbol shall be represented by an inline expression symbol labelled with a **while** definition as defined in clause 19.8 of [ITU-T Z.161]. The **while** body shall be represented as the operand of the while inline expression symbol. Figure 27 represents an example of a **while** statement.

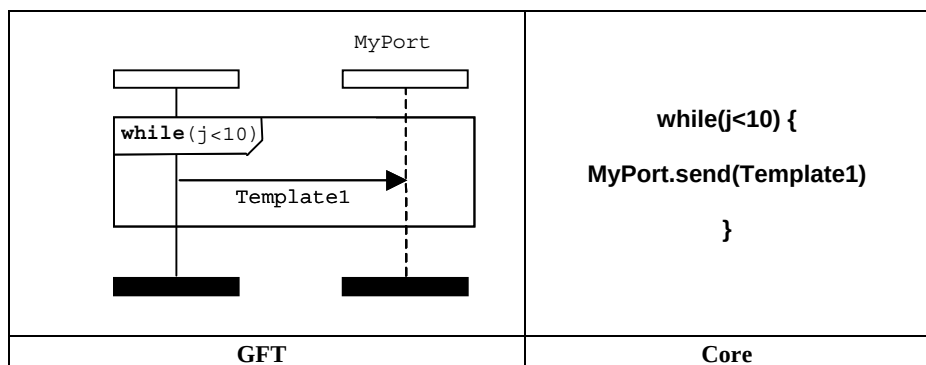


Figure 27 – While statement

11.4.7 The Do-while statement

The **do-while** statement shall be represented by an inline expression symbol labelled with a **do-while** definition as defined in clause 19.9 of [ITU-T Z.161]. The **do-while** body shall be represented as the operand of the do-while inline expression symbol. Figure 28 represents an example of a **do-while** statement.

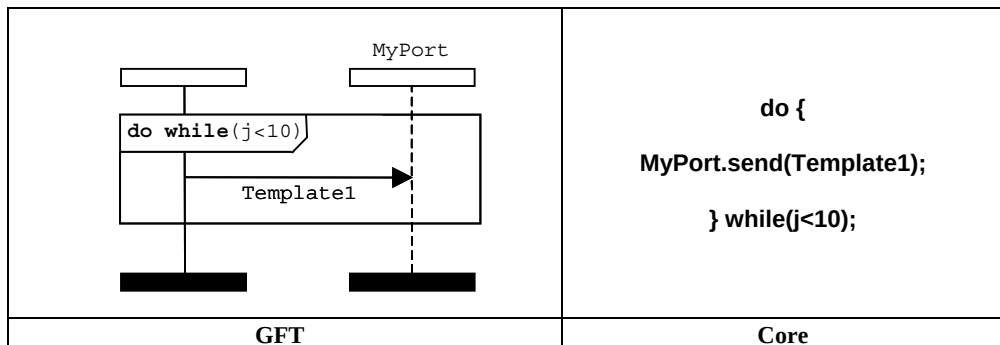


Figure 28 – Do-while statement

11.5 Behavioural program statements

Behavioural statements may be used within test cases, functions and altsteps, the only exception being the return statement, which can only be used within functions. Test behaviour can be expressed sequentially, as a set of alternatives or using an interleaving statement. Return and repeat are used to control the flow of behaviour.

11.5.1 Sequential behaviour

Sequential behaviour is represented by the order of events placed upon a test component instance. The ordering of events is taken in a top-down manner, with events placed nearest the top of the component instance symbol being evaluated first. Figure 29 illustrates a simple case in which the test component firstly evaluates the expression contained within the action symbol and then sends a message to a port **MyPort**.

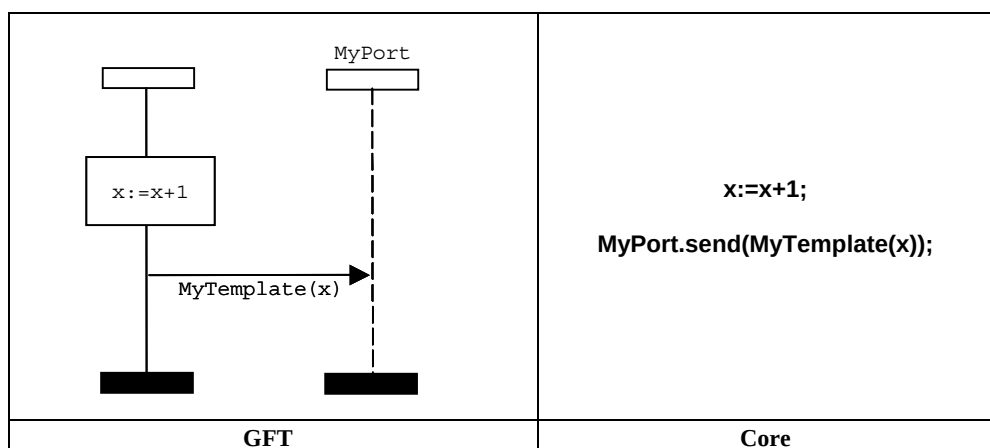


Figure 29 – Sequential behaviour

Sequencing can also be described using references to test cases, functions, and altsteps. In this case, the order in which references are placed upon a component instance axis determines the order in which they are evaluated. Figure 30 represents a simple GFT diagram in which `MyFunction1` is called, followed by `MyFunction2`.

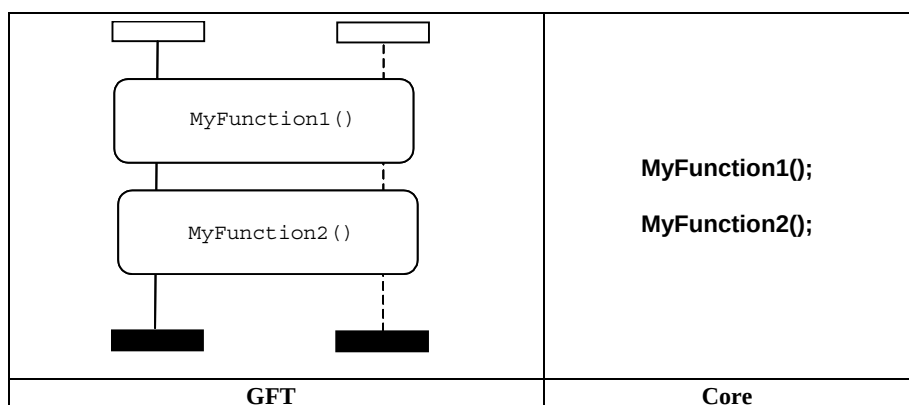


Figure 30 – Sequencing using references

11.5.2 Alternative behaviour

Alternative behaviour shall be represented using inline expression symbol with the `alt` keyword placed in the top left hand corner. Each operand of the alternative behaviour shall be separated using a dashed line. Operands are evaluated top-down.

Note that an alternative inline expression should always cover all port instances, if communication operators are involved. Figure 31 illustrates an alternative behaviour in which either a message event is received with the value defined by `Template1`, or a message event is received with the value defined by `Template2`. The invocation of an altstep in an alternative inline expression is shown in Figure 32.

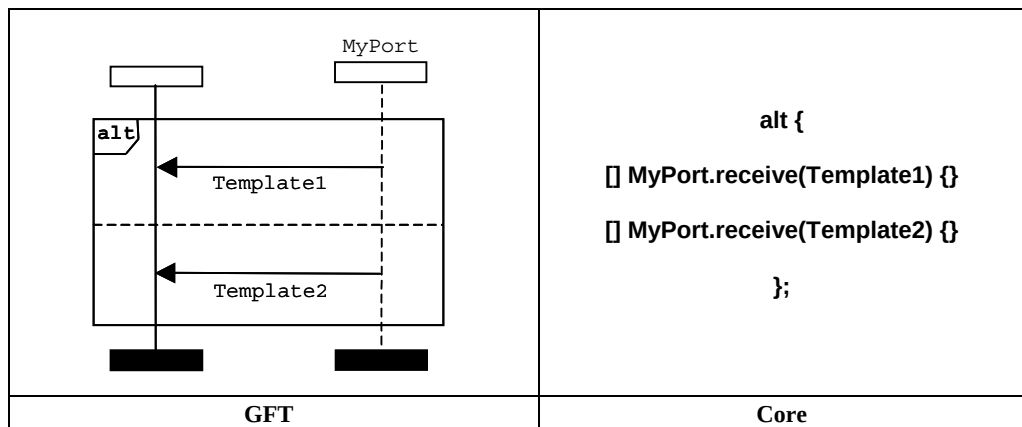


Figure 31 – Alternative behaviour statement

In addition, it is possible to call an altstep as the only event within an alternative operand. This shall be drawn using a reference symbol (see clause 11.2.3).

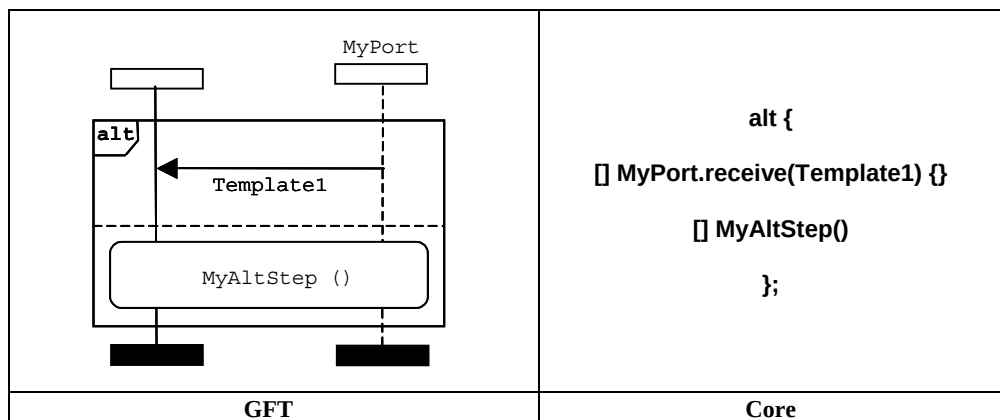


Figure 32 – Alternative behaviour with altstep invocation

11.5.2.1 Selecting/deselecting an alternative

It is possible to disable/enable an alternative operand by means of a Boolean expression contained within a condition symbol placed upon the test component instance. Figure 33 illustrates a simple alternative statement in which the first operand is guarded with the expression $x > 1$, and the second with the expression $x \leq 1$.

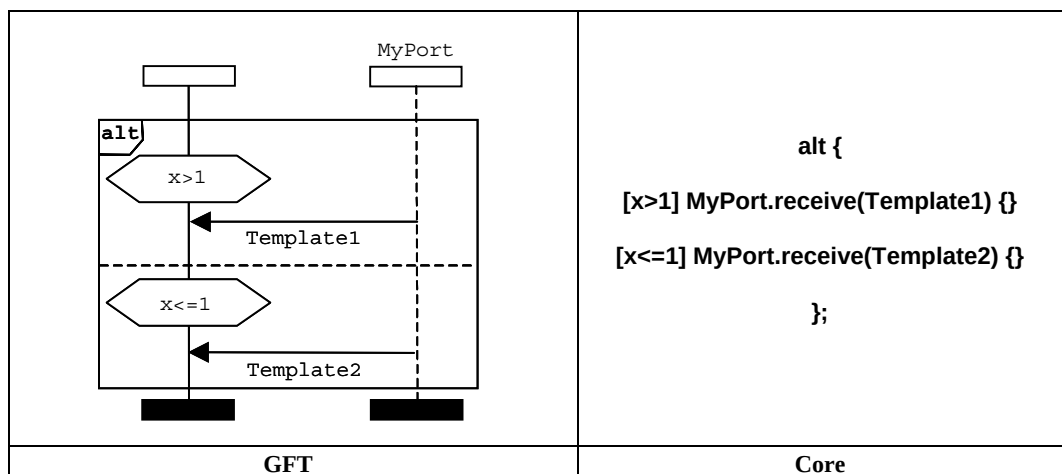


Figure 33 – Selecting/deselecting an alternative

11.5.2.2 Else branch in alternatives

The **else** branch shall be denoted using a condition symbol placed upon the test component instance axis labelled with the **else** keyword. Figure 34 illustrates a simple alternative statement where the second operand represents an **else** branch.

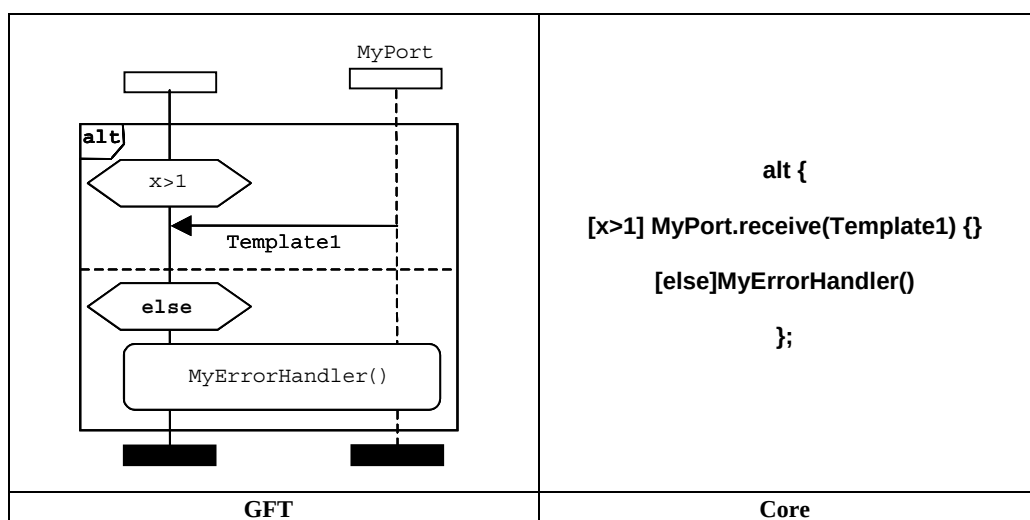


Figure 34 – Else within an alternative

Note that the reference symbol within an else branch should always cover all port instances, if communication operations are involved.

The re-evaluation of an alt statement can be specified using a repeat statement, which is represented by the repeat symbol (see clause 11.5.3).

The invocation of altsteps within alternatives is represented using the reference symbol (see clause 11.2.3).

11.5.3 The Repeat statement

The **repeat** statement shall be represented by a repeat symbol. This symbol shall only be used as last event of an alternative operand in an **alt** statement or as last event of an operand of the top alternative in an altstep definition. Figure 35 illustrates an alternative statement in which the second

operand, having successfully received a message with a value matching `Template2`, causes the alternative to be repeated.

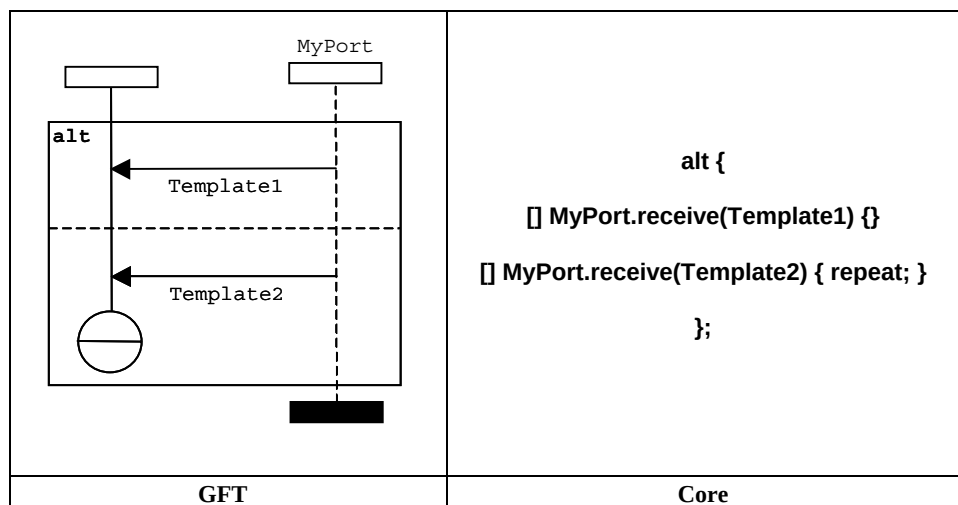


Figure 35 – Repeat within an alternative

11.5.4 Interleaved behaviour

Interleave behaviour shall be represented using an inline expression symbol with the **interleave** keyword placed in the top left hand corner (see Figure 36). Each operand shall be separated using a dashed line. Operands are evaluated in a top-down order.

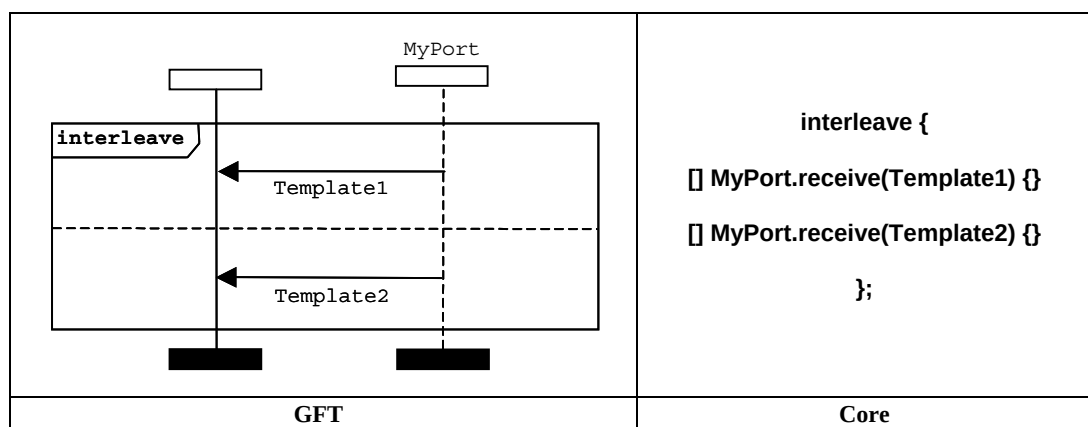


Figure 36 – Interleave statement

11.5.5 The Return statement

The **return** statement shall be represented by a return symbol. This may be optionally associated with a return value. A return symbol shall only be used in a GFT function diagram. It shall only be used as last event of a component instance or as last event of an operand in an inline expression symbol. Figure 37 illustrates a simple function using a return statement without a returning a value, and Figure 38 illustrates a function that returns a value.

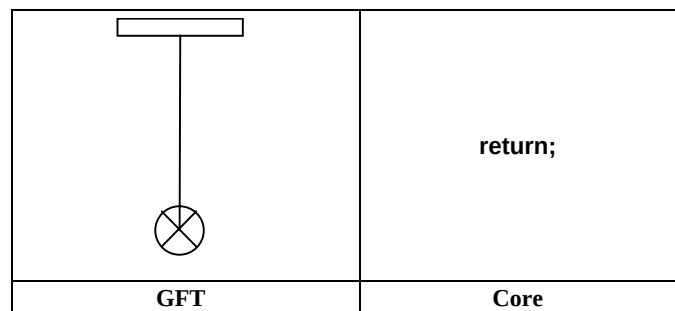


Figure 37 – Return symbol without a return value

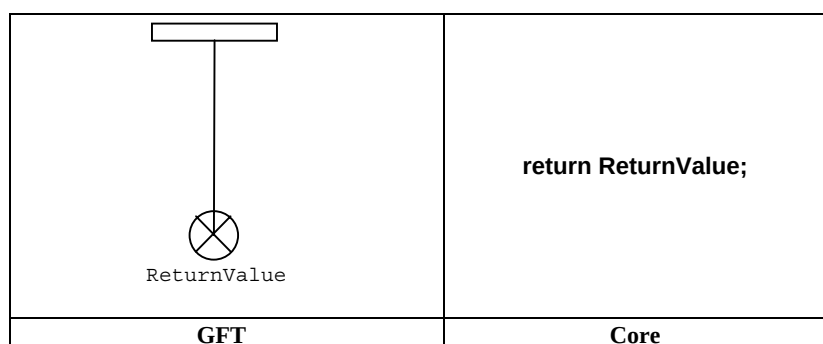


Figure 38 – Return symbol with a return value

11.6 Default handling

GFT provides graphical representation for the activation and deactivation of defaults (see clause 21 of [ITU-T Z.161]).

11.6.1 Default references

Variables of type **default** can either be declared within an action symbol or within a default symbol as part of an activate statement. Clauses 11.3.1 and 11.3.4 illustrate how a variable called **MyDefaultType** is declared within GFT.

11.6.2 The activate operation

The activation of defaults shall be represented by the placement of the **activate** statement within a default symbol (see Figure 39).

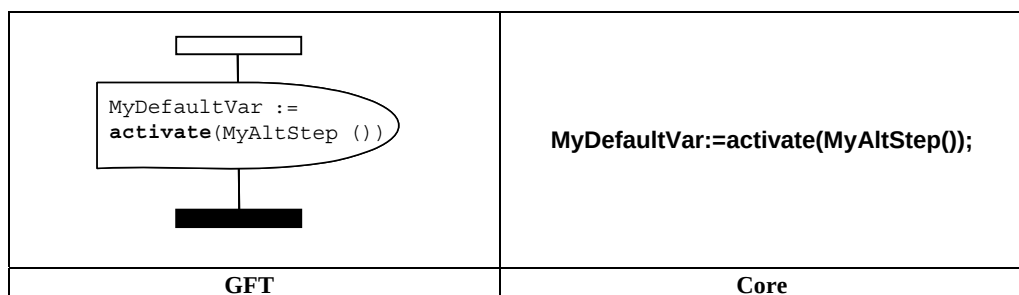


Figure 39 – Default activation

11.6.3 The deactivate operation

The deactivation of defaults shall be represented by the placement of the **deactivate** statement within a default symbol (see Figure 40). If no operands are given to the **deactivate** statement then all defaults are deactivated.

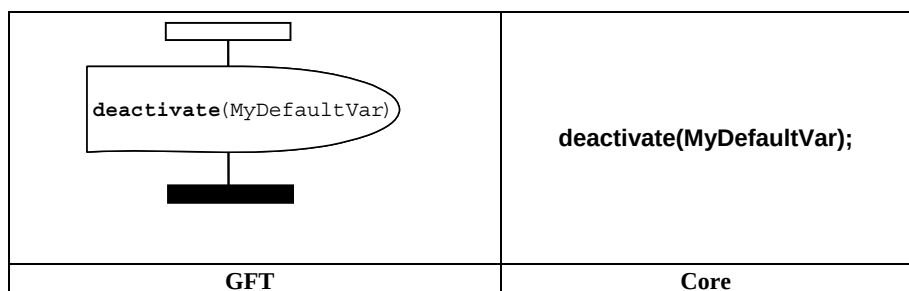


Figure 40 – Deactivation of defaults

11.7 Configuration operations

Configuration operations are used to set up and control test components. These operations shall only be used in GFT test case, function, and altstep diagrams.

The **mtc**, **self**, and **system** operations have no graphical representation; they are textually denoted at the places of their use.

GFT does not provide any graphical representation for the running operation (being a Boolean expression). It is textually denoted at the place where it is used.

11.7.1 The Create operation

The **create** operation shall be represented within the create symbol, which is attached to the test component instance which performs the create operation (see Figure 41). The create symbol contains the create statement.

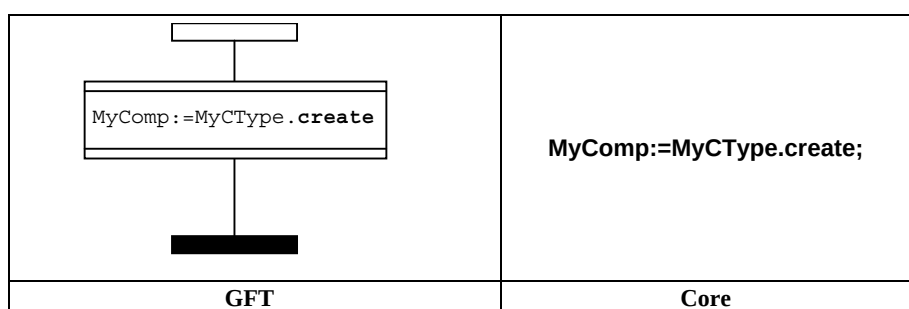


Figure 41 – Create operation

11.7.2 The Connect and Map operations

The **connect** and **map** operations shall be represented within an action box symbol, which is attached to the test component instance which performs the **connect** or **map** operation (see Figure 42). The action box symbol contains the **connect** or **map** statement.

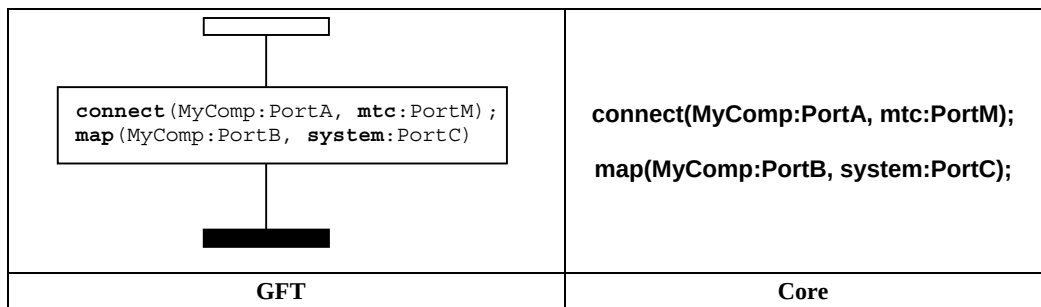


Figure 42 – Connect and map operation

11.7.3 The Disconnect and Unmap operations

The **disconnect** and **unmap** operations shall be represented within an action box symbol, which is attached to the test component instance which performs the **disconnect** or **unmap** operation (see Figure 43). The action box symbol contains the **disconnect** or **unmap** statement.

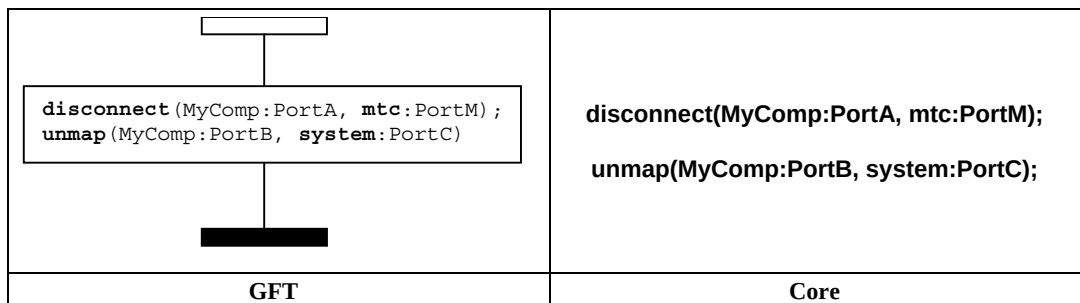


Figure 43 – Disconnect and unmap operation

11.7.4 The Start test component operation

The **start** test component operation shall be represented within the **start** symbol, which is attached to the test component instance that performs the **start** operation (see Figure 44). The start symbol contains the **start** statement.

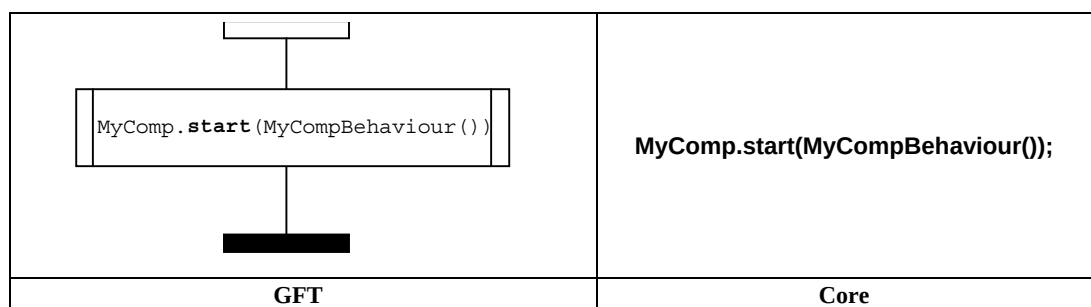


Figure 44 – Start operation

11.7.5 The Stop execution and Stop test component operations

TTCN-3 has two stop operations: The module control and test components may stop themselves by using *stop execution operations*, or a test component can stop other test components by using *stop test component operations*.

The **stop** execution operation shall be represented by a stop symbol, which is attached to the test component instance, which performs the **stop** execution operation (see Figure 45). It shall only be used as last event of a component instance or as last event of an operand in an inline expression symbol.

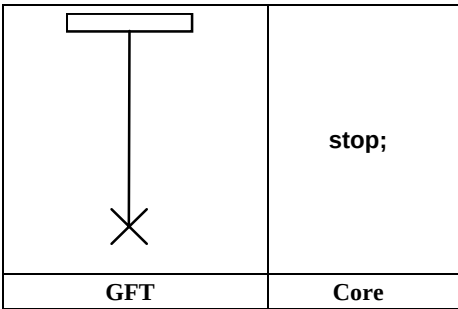


Figure 45 – Stop execution operation

The **stop** test component operation shall be represented by a stop symbol, which is attached to the test component instance, which performs the **stop** test component operation. It shall have an associated expression that identifies the component to be stopped (see Figure 46). The MTC may stop all PTCs in one step by using the stop component operation with the keyword **all** (see Figure 47 a)). A PTC can stop the test execution by stopping the MTC (see Figure 47 b)). The **stop** test component operation shall be used as last event of a component instance or as last event of an operand in an inline expression symbol, if the component stops itself (e.g., **self.stop**) or stops the test execution (e.g., **mtc.stop**) (see Figures 47 c) and d)).

NOTE – The stop symbol has an associated expression. It is not always possible to determine statically, if a stop component operation stops the instance that executes the stop operation or stops the test execution.

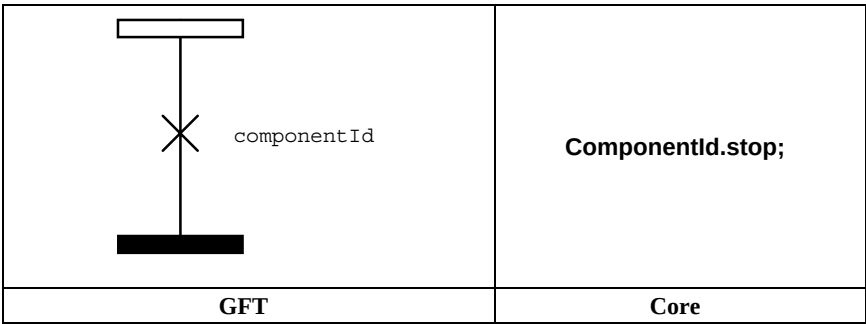


Figure 46 – Stop test component operation

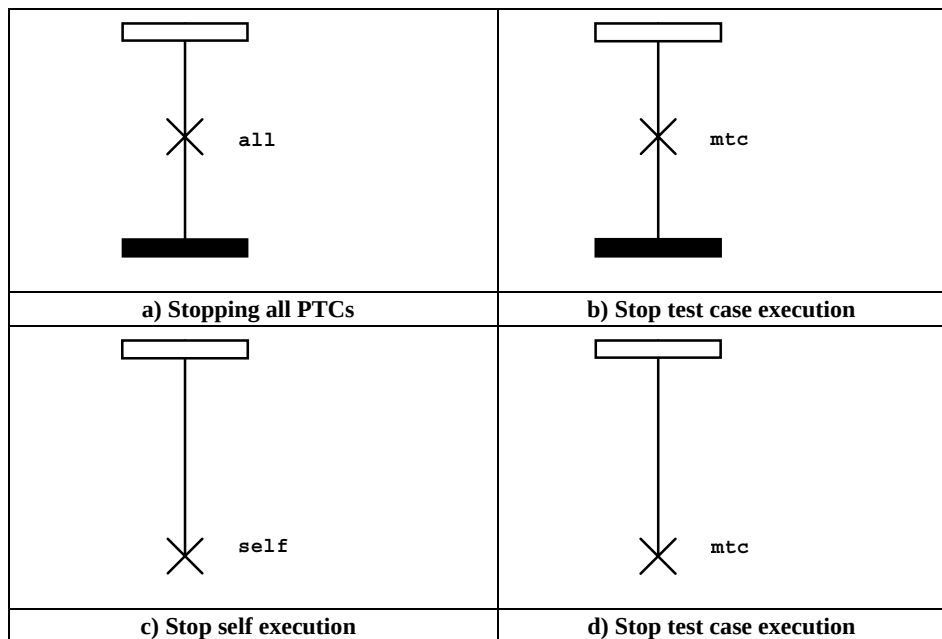


Figure 47 – Special usages of the stop test component operation

11.7.6 The Done operation

The **done** operation shall be represented within a condition symbol, which is attached to the test component instance, which performs the **done** operation (see Figure 48). The condition symbol contains the **done** statement.

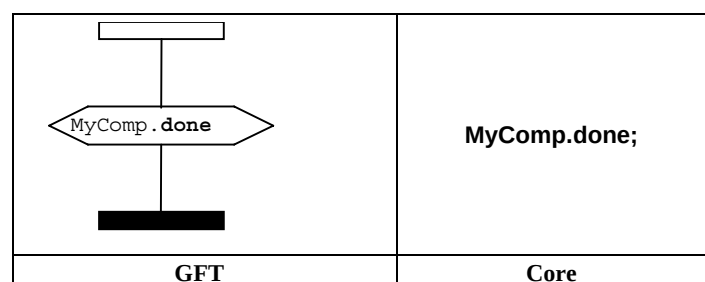


Figure 48 – Done operation

The **any** and **all** keywords can be used for the **running** and **done** operations but from the MTC instance only. They have no graphical representation, but are textually denoted at the places of their use.

11.8 Communication operations

Communication operations are structured into two groups:

- Sending operations:* a test component sends a message (**send** operation), calls a procedure (**call** operation), replies to an accepted call (**reply** operation) or raises an exception (**raise** operation).
- Receiving operations:* a component receives a message (**receive** operation), accepts a procedure call (**getcall** operation), receives a reply for a previously called procedure (**getreply** operation) or catches an exception (**catch** operation).