UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T X.292

(05/2002)

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

SERIE X: REDES DE DATOS Y COMUNICACIÓN ENTRE SISTEMAS ABIERTOS

Interconexión de sistemas abiertos – Pruebas de conformidad

Metodología y marco de las pruebas de conformidad para interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del UIT-T – Notación combinada arborescente y tabular

Recomendación UIT-T X.292

RECOMENDACIONES UIT-T DE LA SERIE X

REDES DE DATOS Y COMUNICACIÓN ENTRE SISTEMAS ABIERTOS

REDES PÚBLICAS DE DATOS	
Servicios y facilidades	X.1–X.19
Interfaces	X.20–X.49
Transmisión, señalización y conmutación	X.50–X.89
Aspectos de redes	X.90–X.149
Mantenimiento	X.150–X.179
Disposiciones administrativas	X.180–X.199
INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Modelo y notación	X.200–X.209
Definiciones de los servicios	X.210–X.219
Especificaciones de los protocolos en modo conexión	X.220–X.229
Especificaciones de los protocolos en modo sin conexión	X.230–X.239
Formularios para declaraciones de conformidad de implementación de protocolo	X.240–X.259
Identificación de protocolos	X.260–X.269
Protocolos de seguridad	X.270–X.279
Objetos gestionados de capa	X.280–X.289
Pruebas de conformidad	**X.290–X.299**
INTERFUNCIONAMIENTO ENTRE REDES	
Generalidades	X.300–X.349
Sistemas de transmisión de datos por satélite	X.350–X.369
Redes basadas en el protocolo Internet	X.370–X.399
SISTEMAS DE TRATAMIENTO DE MENSAJES	X.400–X.499
DIRECTORIO	X.500–X.599
GESTIÓN DE REDES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS Y ASPECTOS DE SISTEMAS	
Gestión de redes	X.600–X.629
Eficacia	X.630–X.639
Calidad de servicio	X.640–X.649
Denominación, direccionamiento y registro	X.650–X.679
Notación de sintaxis abstracta uno	X.680–X.699
GESTIÓN DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Marco y arquitectura de la gestión de sistemas	X.700–X.709
Servicio y protocolo de comunicación de gestión	X.710–X.719
Estructura de la información de gestión	X.720–X.729
Funciones de gestión y funciones de arquitectura de gestión distribuida abierta	X.730–X.799
SEGURIDAD	X.800–X.849
APLICACIONES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Compromiso, concurrencia y recuperación	X.850–X.859
Procesamiento de transacciones	X.860–X.879
Operaciones a distancia	X.880–X.899
PROCESAMIENTO DISTRIBUIDO ABIERTO	X.900–X.999

Para más información, véase la Lista de Recomendaciones del UIT-T.

RECOMENDACIÓN UIT-T X.292

Metodología y marco de las pruebas de conformidad para interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del UIT-T – Notación combinada arborescente y tabular

Resumen

En esta Recomendación UIT-T se define una notación de prueba informal, denominada notación combinada arborescente y tabular (TTCN), para las series de pruebas de conformidad de OSI, que es independiente de los métodos de prueba, capas y protocolos y que refleja la metodología de las pruebas abstractas definida en las Recomendaciones UIT-T X.290 y X.291. Se ha añadido a esta edición las correcciones de los defectos recibidas. Esta versión es equivalente a ETSI TR 101 666, que también se conoce como TTCN2++.

Orígenes

La Recomendación UIT-T X.292, preparada por la Comisión de Estudio 17 (2001-2004) del UIT-T, fue aprobada por el procedimiento de la Resolución 1 de la AMNT el 22 de mayo de 2002.

PREFACIO

La UIT (Unión Internacional de Telecomunicaciones) es el organismo especializado de las Naciones Unidas en el campo de las telecomunicaciones. El UIT-T (Sector de Normalización de las Telecomunicaciones de la UIT) es un órgano permanente de la UIT. Este órgano estudia los aspectos técnicos, de explotación y tarifarios y publica Recomendaciones sobre los mismos, con miras a la normalización de las telecomunicaciones en el plano mundial.

La Asamblea Mundial de Normalización de las Telecomunicaciones (AMNT), que se celebra cada cuatro años, establece los temas que han de estudiar las Comisiones de Estudio del UIT-T, que a su vez producen Recomendaciones sobre dichos temas.

La aprobación de Recomendaciones por los Miembros del UIT-T es el objeto del procedimiento establecido en la Resolución 1 de la AMNT.

En ciertos sectores de la tecnología de la información que corresponden a la esfera de competencia del UIT-T, se preparan las normas necesarias en colaboración con la ISO y la CEI.

NOTA

En esta Recomendación, la expresión "Administración" se utiliza para designar, en forma abreviada, tanto una administración de telecomunicaciones como una empresa de explotación reconocida de telecomunicaciones.

PROPIEDAD INTELECTUAL

La UIT señala a la atención la posibilidad de que la utilización o aplicación de la presente Recomendación suponga el empleo de un derecho de propiedad intelectual reivindicado. La UIT no adopta ninguna posición en cuanto a la demostración, validez o aplicabilidad de los derechos de propiedad intelectual reivindicados, ya sea por los miembros de la UIT o por terceros ajenos al proceso de elaboración de Recomendaciones.

En la fecha de aprobación de la presente Recomendación, la UIT no ha recibido notificación de propiedad intelectual, protegida por patente, que puede ser necesaria para aplicar esta Recomendación. Sin embargo, debe señalarse a los usuarios que puede que esta información no se encuentre totalmente actualizada al respecto, por lo que se les insta encarecidamente a consultar la base de datos sobre patentes de la TSB.

ÍNDICE

Página

1 Alcance ... 1

2 Referencias normativas ... 2
- 2.1 Recomendaciones | Normas Internacionales idénticas ... 2
- 2.2 Pares de Recomendaciones | Normas Internacionales cuyo contenido técnico es equivalente ... 3
- 2.3 Referencias adicionales ... 3

3 Definiciones ... 3
- 3.1 Términos básicos extraídos de la Rec. UIT-T X.290 ... 3
- 3.2 Términos de la Rec. UIT-T X.200 ... 5
- 3.3 Términos de la Rec. UIT-T X.210 ... 5
- 3.4 Términos de la Rec. UIT-T X.680 ... 5
- 3.5 Términos de la Rec. UIT-T X.690 ... 5
- 3.6 Términos específicos de la TTCN ... 5

4 Abreviaturas ... 9
- 4.1 Abreviaturas definidas en la Rec. UIT-T X.290 ... 9
- 4.2 Abreviaturas definidas en la Rec. UIT-T X.291 ... 10
- 4.3 Otras abreviaturas ... 10

5 Las formas de sintaxis de la TTCN ... 10

6 Cumplimiento ... 11

7 Convenios ... 12
- 7.1 Introducción ... 12
- 7.2 Metanotación sintáctica ... 12
- 7.3 Formularios de cuadro en TTCN.GR ... 12
  - 7.3.1 Introducción ... 12
  - 7.3.2 Cuadro para objeto TTCN único ... 12
  - 7.3.3 Cuadro de objeto TTCN múltiple ... 14
  - 7.3.4 Cuadros compactos alternativos ... 14
  - 7.3.5 Especificación de formularios ... 14
- 7.4 Texto libre y texto libre limitado ... 15

8 Concurrencia en la TTCN ... 15
- 8.1 Componentes de prueba ... 15
- 8.2 Configuraciones de componentes de prueba ... 15

9 Estructura de la serie de pruebas con TTCN ... 16
- 9.1 Introducción ... 16
- 9.2 Referencias de grupos de prueba ... 17
- 9.3 Referencias de grupo de pasos de prueba ... 17
- 9.4 Referencias de grupo de valores por defecto ... 17
- 9.5 Partes de una serie de pruebas TTCN ... 17

10 Visión general de la serie de pruebas ... 18
- 10.1 Introducción ... 18
- 10.2 Índice de series de pruebas ... 18
- 10.3 Estructura de la serie de pruebas ... 19
- 10.4 Índice de casos de prueba ... 20
- 10.5 Índice de pasos de prueba ... 21
- 10.6 Índice de valores por defecto ... 22
- 10.7 Exportaciones de serie de pruebas ... 23
- 10.8 La parte Importación ... 24
  - 10.8.1 Introducción ... 24
  - 10.8.2 Importaciones ... 24

11 La parte Declaraciones ... 25
- 11.1 Introducción ... 25
- 11.2 Tipos de TTCN ... 26
  - 11.2.1 Introducción ... 26

Página
11.2.2 Tipos TTCN predefinidos ... 26
11.2.3 Definiciones de tipos de series de pruebas ... 27
11.3 Operadores de TTCN y operaciones de TTCN ... 32
11.3.1 Introducción ... 32
11.3.2 Operadores de TTCN ... 32
11.3.3 Operaciones predefinidas ... 34
11.3.4 Definiciones y descripciones de operaciones serie de pruebas ... 36
11.4 Declaraciones de parámetros de serie de pruebas ... 40
11.5 Definiciones de expresión de selección de caso de prueba ... 41
11.6 Declaraciones de constantes de series de pruebas ... 42
11.7 Declaraciones de constantes de series de pruebas por referencia ... 43
11.8 Variables de TTCN ... 43
11.8.1 Declaraciones de variables de series de pruebas ... 43
11.8.2 Acotación de variables de series de pruebas ... 44
11.8.3 Declaraciones de variables de casos de prueba ... 44
11.8.4 Acotación de variables de casos de prueba ... 45
11.9 Declaraciones de tipos de PCO ... 45
11.10 Declaraciones de PCO ... 46
11.11 Declaraciones de CP ... 48
11.12 Declaraciones de temporizador ... 49
11.13 Componentes de prueba y declaraciones de configuración ... 50
11.13.1 Componentes de prueba ... 50
11.13.2 Declaraciones de configuraciones de componentes de prueba ... 51
11.14 Definiciones de tipos de ASP ... 53
11.14.1 Introducción ... 53
11.14.2 Definiciones de tipos de ASP mediante cuadros ... 53
11.14.3 Utilización de tipos estructurados en las definiciones de tipo de ASP ... 54
11.14.4 Definiciones de tipos de ASP con ASN.1 ... 55
11.14.5 Definiciones de tipos de ASP en ASN.1 por referencia ... 56
11.15 Definiciones de tipos de PDU ... 56
11.15.1 Introducción ... 56
11.15.2 Definición de tipos de PDU mediante cuadros ... 56
11.15.3 Utilización de tipos estructurados dentro de las definiciones de PDU ... 58
11.15.4 Definiciones de tipo de PDU con ASN.1 ... 58
11.15.5 Definiciones de tipo de PDU en ASN.1 por referencia ... 60
11.16 Información de codificación de serie de pruebas ... 61
11.16.1 Definiciones de codificación ... 61
11.16.2 Variaciones de codificación ... 62
11.16.3 Definiciones de codificación de campo no válidas ... 63
11.16.4 Aplicación de las reglas de codificación ... 64
11.17 Definiciones de tipos de CM ... 65
11.17.1 Introducción ... 65
11.17.2 Definiciones de tipos de CM mediante cuadros ... 66
11.17.3 Definiciones de tipos de CM con ASN.1 ... 66
11.18 Especificaciones de longitud de cadena ... 67
11.19 Definiciones de ASP, PDU y CM para eventos SEND (enviar) ... 68
11.20 Definiciones de ASP, PDU y CM para eventos RECEIVE (recibir) ... 68
11.21 Definiciones de alias ... 69
11.21.1 Introducción ... 69
11.21.2 Expansión de alias ... 69
12 Parte constricciones ... 70
12.1 Introducción ... 70
12.2 Principios generales ... 70
12.3 Parametrización de constricciones ... 71
12.4 Encadenamiento de constricciones ... 71
12.5 Constricciones para eventos SEND ... 71
12.6 Constricciones para eventos RECEIVE ... 72
12.6.1 Valores de concordancia ... 72

Página

12.6.2 Mecanismos de concordancia ... 72
12.6.3 Specific Value (valor específico) ... 73
12.6.4 Instead of value (valor en vez de) ... 73
12.6.5 Inside Values (valores interiores) ... 76
12.6.6 Atributes of values (atributos de valores) ... 77

13 Especificación de constricciones mediante cuadros (o tablas) ... 77
13.1 Introducción ... 77
13.2 Declaraciones de constricciones de tipo estructurado ... 78
13.3 Declaraciones de constricciones de ASP ... 79
13.4 Declaraciones de constricciones de PDU ... 80
13.5 Parametrización de constricciones ... 82
13.6 Constricciones de base y constricciones modificadas ... 82
13.7 Listas de parámetros formales en las constricciones modificadas ... 82
13.8 Declaraciones de constricciones de CM ... 83

14 Especificación de constricciones mediante ASN.1 ... 83
14.1 Introducción ... 83
14.2 Declaraciones de constricciones de tipo en ASN.1 ... 83
14.3 Declaraciones de constricciones de ASP en ASN.1 ... 85
14.4 Declaraciones de constricciones de PDU en ASN.1 ... 85
14.5 Constricciones en ASN.1 parametrizadas ... 86
14.6 Constricciones en ASN.1 modificadas ... 86
14.7 Listas de parámetros formales en las constricciones en ASN.1 modificadas ... 87
14.8 Nombres de parámetro ASP y campo de PDU en las constricciones en ASN.1 ... 87
14.9 Declaraciones de constricción de CM en ASN.1 ... 88

15 La parte dinámica ... 88
15.1 Introducción ... 88
15.2 Comportamiento dinámico de caso de prueba ... 88
15.2.1 Especificación del cuadro de comportamiento dinámico de caso de prueba ... 88
15.2.2 El formulario de comportamiento dinámico de caso de prueba ... 89
15.2.3 Estructura del comportamiento de caso de prueba ... 90
15.2.4 Descripción del comportamiento de caso de prueba concurrente ... 90
15.2.5 Numeración y continuación de líneas ... 91
15.3 Comportamiento dinámico de paso de prueba ... 92
15.3.1 Especificación del cuadro de comportamiento dinámico de los pasos de prueba ... 92
15.3.2 El formulario de comportamiento dinámico de paso de prueba ... 92
15.4 Comportamiento dinámico por defecto ... 92
15.4.1 Comportamiento por defecto ... 92
15.4.2 Especificación del cuadro de comportamiento dinámico por defecto ... 93
15.4.3 El formulario de comportamiento dinámico por defecto ... 93
15.5 La descripción de comportamiento ... 94
15.6 La notación arborescente ... 94
15.7 Nombres de árbol y lista de parámetros ... 95
15.7.1 Introducción ... 95
15.7.2 Árboles con parámetros ... 95
15.8 Enunciados en TTCN ... 95
15.9 Eventos de prueba en TTCN ... 96
15.9.1 Eventos de enviar y recibir ... 96
15.9.2 Eventos de recibir ... 96
15.9.3 Eventos de enviar ... 96
15.9.4 Tiempo de vida de los eventos ... 97
15.9.5 Ejecución del árbol de comportamiento ... 97
15.9.6 El evento IMPLICIT SEND ... 99
15.9.7 El evento OTHERWISE ... 100
15.9.8 El evento OTHERWISE y la TTCN concurrente ... 101
15.9.9 El evento TIMEOUT ... 101
15.9.10 Eventos y constructivos en la TTCN concurrente ... 102

Página

15.10 Expresiones en TTCN ... 102
15.10.1 Introducción ... 102
15.10.2 Referencias a objetos de datos definidos en ASN.1 ... 103
15.10.3 Referencias para objetos de datos definidos con cuadros ... 105
15.10.4 Asignaciones ... 105
15.10.5 Calificadores ... 106
15.10.6 Líneas de evento con asignaciones y calificadores ... 106
15.11 Seudoeventos ... 107
15.12 Gestión de temporizador ... 107
15.12.1 Introducción ... 107
15.12.2 La operación START ... 108
15.12.3 La operación CANCEL ... 108
15.12.4 La operación READTIMER ... 109
15.13 El constructivo ATTACH ... 109
15.13.1 Introducción ... 109
15.13.2 Ámbito de la adjunción de árbol ... 109
15.13.3 Aspectos básicos de la adjunción de árbol ... 110
15.13.4 El significado de adjunción de árbol ... 110
15.13.5 Paso de constricciones parametrizadas ... 112
15.13.6 Adjunción recursiva de árbol ... 112
15.13.7 Adjunción de árboles y valores por defecto ... 113
15.14 Etiquetas y constructivo GOTO ... 113
15.15 El constructivo REPEAT ... 114
15.16 La referencia a constricciones ... 115
15.16.1 Finalidad de la columna referencia a constricciones ... 115
15.16.2 Paso de parámetros en referencias a constricciones ... 115
15.16.3 Constricciones y calificadores y asignaciones ... 116
15.17 Veredictos ... 116
15.17.1 Introducción ... 116
15.17.2 Resultados preliminares ... 116
15.17.3 Veredicto final ... 117
15.17.4 Veredictos y OTHERWISE ... 117
15.17.5 Asignación de veredicto en la TTCN concurrente ... 118
15.18 El significado de valores por defecto ... 118
15.18.1 Introducción ... 118
15.18.2 Referencias de valores por defecto ... 119
15.18.3 El enunciado RETURN (devolución) ... 120
15.18.4 El enunciado ACTIVATE ... 120
15.18.5 Valores por defecto y adjunción de árbol ... 120
15.18.6 Adjunción de árboles, valores por defecto, Activate y Return ... 122
15.18.7 Valores por defecto y CREATE ... 127
15.18.8 Valores por defecto y mensajes CM ... 127

16 Continuación de página ... 128
16.1 Continuación de página en los cuadros de TTCN ... 128
16.2 Continuación de página en los cuadros de comportamiento dinámico ... 129

Anexo A – Sintaxis y semántica estática de la TTCN ... 130
A.1 Introducción ... 130
A.2 Convenciones de la descripción de sintaxis ... 130
A.2.1 Metanotación sintáctica ... 130
A.2.2 Definiciones de sintaxis de TTCN.MP ... 130
A.3 Las producciones de sintaxis TTCN.MP en BNF ... 131
A.3.1 Especificación de TTCN ... 131
A.3.2 Módulo de TTCN ... 131
A.3.3 Serie de pruebas ... 132
A.4 Requisitos generales de semántica estática ... 155
A.4.1 Introducción ... 155
A.4.2 Exclusividad de los identificadores ... 155
A.5 Diferencias entre TTCN.GR y TTCN.MP ... 159
A.5.1 Diferencias de sintaxis ... 159

Página
A.5.2 Semántica estática adicional en la TTCN.MP ... 160
A.6 Lista de números de producción en BNF ... 160
Anexo B – Semántica operacional de la TTCN ... 161
B.1 Introducción ... 161
B.2 Precedencia ... 161
B.3 Procesamiento de errores de caso de prueba ... 161
B.4 Conversión de una serie de pruebas modularizada a una serie de pruebas expandida equivalente ... 161
B.5 Semántica operacional de la TTCN ... 162
B.5.1 Introducción ... 162
B.5.2 Notación de seudocódigo ... 162
B.5.3 Ejecución de un caso de prueba ... 163
B.5.4 Ejecución de un caso de prueba ... 164
B.5.5 Expansión de un conjunto de alternativas ... 165
B.5.6 Evaluación de una línea de evento ... 168
B.5.7 Funciones para eventos de TTCN ... 168
B.5.8 Ejecución del evento SEND ... 168
B.5.9 Ejecución del evento RECEIVE ... 169
B.5.10 Ejecución del evento OTHERWISE ... 171
B.5.11 Ejecución del evento TIMEOUT ... 171
B.5.12 Ejecución del evento DONE ... 173
B.5.13 Ejecución del evento IMPLICIT SEND ... 173
B.5.14 Ejecución de un pseudo-event ... 174
B.5.15 Ejecución de expresiones BOOLEAN ... 174
B.5.16 Ejecución de asignaciones ... 174
B.5.17 Ejecución de operaciones TIMER ... 175
B.5.18 Funciones para constructivos de TTCN ... 175
B.5.19 Ejecución del constructivo ACTIVATE ... 176
B.5.20 Ejecución del constructivo CREATE ... 176
B.5.21 Ejecución del constructivo GOTO ... 177
B.5.22 Ejecución del constructivo RETURN ... 177
B.5.23 Veredicto ... 177
B.5.24 Registro cronológico de conformidad ... 178
B.5.25 Funciones y procedimientos de tratamiento del árbol ... 179
B.5.26 Funciones diversas utilizadas por el seudocódigo ... 180
Anexo C – Módulos TTCN ... 182
C.1 Introducción ... 182
C.2 Parte visión general del módulo TTCN ... 182
C.2.1 Introducción ... 182
C.2.2 Exportaciones del módulo TTCN ... 182
C.2.3 Estructura del módulo TTCN ... 184
C.2.4 Índice de casos de prueba ... 184
C.2.5 Índice de pasos de prueba ... 184
C.2.6 Índice de valores por defecto ... 184
C.3 Parte importación ... 184
C.3.1 Introducción ... 184
C.3.2 Externos ... 184
C.3.3 Importación ... 185
Anexo D – Índice de serie de pruebas ... 186
Anexo E – Formularios compactos ... 187
E.1 Introducción ... 187
E.2 Formularios compactos para constricciones ... 187
E.2.1 Requisitos ... 187
E.2.2 Formularios compactos para las constricciones de ASP ... 187
E.2.3 Formularios compactos para constricciones de PDU ... 188
E.2.4 Formularios compactos para constricciones de tipo estructurado ... 190
E.2.5 Formularios compactos para constricciones en ASN.1 ... 192
E.3 Formulario compacto para casos de prueba ... 193
E.3.1 Requisitos ... 193
E.3.2 Formulario compacto para comportamiento dinámico de casos de prueba ... 193

Página

Anexo F – Ejemplos ... 195
F.1 Ejemplos de constricciones en forma de cuadro ... 195
F.1.1 Definiciones de ASP y PDU ... 195
F.1.2 Constricciones de ASP/PDU ... 196
F.2 Ejemplos de constricciones en ASN.1 ... 199
F.2.1 Definiciones de ASP y PDU ... 199
F.2.2 Constricciones de ASP/PDU en ASN.1 ... 201
F.2.3 Otros ejemplos de constricciones en ASN.1 ... 204
F.3 Constricciones de base y modificadas ... 206
F.4 Definición de tipo con macros ... 208
F.5 Utilización de REPEAT ... 209
F.6 Operaciones series de pruebas ... 210
F.7 Ejemplo de visión general de una serie de pruebas ... 210
F.8 Ejemplo de un caso de prueba en forma TTCN.MP ... 212
F.9 Utilización de referencia a componentes para la asignación de valores de campo en constricciones ... 214
F.10 Pruebas multiparte ... 216
F.11 Multiplexación/demultiplexación ... 217
F.12 Partición y recombinación ... 218
F.13 Casos de prueba multiprotocolo ... 218
F.14 Ejemplo de TTCN modular ... 219

Anexo G – Guía de estilo ... 220
G.1 Introducción ... 220
G.2 Estructura de caso de prueba ... 220
G.3 Utilización de la TTCN con diferentes métodos de prueba abstracta ... 221
G.3.1 Introducción ... 221
G.3.2 La TTCN y el método de prueba LS ... 221
G.3.3 La TTCN y el método de prueba DS ... 221
G.3.4 La TTCN y el método de prueba CS ... 221
G.3.5 La TTCN y el método de prueba RS ... 221
G.4 Utilización de valores por defecto ... 222
G.5 Limitación del tiempo de ejecución de un caso de prueba ... 222
G.6 Tipos estructurados ... 222
G.7 Abreviaturas ... 223
G.8 Descripciones de prueba ... 223
G.9 Asignación en caso de eventos SEND ... 223
G.10 PCO multiservicio ... 223

Anexo H – Índice ... 224
H.1 Introducción ... 224
H.2 Índice ... 224

Introducción

En esta Recomendación UIT-T se define una notación de prueba informal, denominada "notación combinada arborescente y tabular"(TTCN, *tree and tabular combined notation*), que se utilizará en la especificación de las secuencias (en adelante "series") de pruebas abstractas de conformidad de OSI.

En el diseño de una serie de pruebas abstractas normalizada se utiliza una notación de prueba con el fin de efectuar la descripción de casos de pruebas abstractas. La notación de prueba puede ser una notación informal (sin una semántica definida formalmente) o una técnica de descripción formal (FDT, *formal description technique*). La TTCN es una notación informal con una semántica claramente definida, pero no formalmente definida.

Se ha diseñado la TTCN con miras a alcanzar los siguientes objetivos:

a) proporcionar una notación con la que poder expresar casos de pruebas abstractas en series de pruebas normalizadas;

b) proporcionar una notación que sea independiente de los métodos de pruebas, capas y protocolos;

c) proporcionar una notación que refleje la metodología de comprobación abstracta definida en las Recomendaciones de la serie X.290.

d) proporcionar una capacidad para utilizar la concurrencia en la especificación de los casos de pruebas abstractas, cuando resulte apropiado, tanto en la comprobación multiparte como en la uniparte.

En la metodología de comprobación abstracta la serie de pruebas se considera como una jerarquía que se extiende desde la serie de pruebas completa hasta los eventos de prueba, pasando por grupos de prueba, casos de prueba y pasos de prueba. La TTCN proporciona una estructura de denominación para reflejar la situación de los casos de prueba en esta jerarquía. Proporciona, asimismo, el modo de estructurar casos de prueba en forma de jerarquía de pasos de prueba, que culmina en los eventos de prueba. En la TTCN, los eventos de prueba básicos envían y reciben primitivas de servicio abstractas (ASP, *abstract service primitives*), unidades de datos de protocolo (PDU, *protocol data units*) y eventos de temporización.

Se han previsto dos formas de notación, a saber: una forma tabular interpretable por personas denominada TTCN.GR, para su utilización en las normas de series de pruebas de conformidad de OSI, y una forma procesable por máquina, denominada TTCN.MP, que se utiliza para representar la TTCN de forma canónica en el entorno de sistemas de computador y como sintaxis para su utilización cuando deban transferirse casos de prueba con TTCN entre diferentes sistemas de computadora. Ambas formas son semánticamente equivalentes.

En esta edición de la Rec. UIT-T X.292 se añaden las correcciones de defectos recibidas para la Rec. UIT-T X.292 (1998). Desde el punto de vista técnico, es equivalente a ETSI TR 101 666 (1999-05), que se conoce también como TTCN2++.

TTCN2++, como se define en la Rec. UIT-T X.292 (2002) es una definición intermedia de TTCN, entre la Rec. UIT-T X.292 (1998) y la Rec. UIT-T Z.140 (2001), y su objetivo es satisfacer la necesidad de mantener las series de pruebas creadas sobre la base de la Rec. UIT-T X.292 (1998). Ahora bien, se recomienda que las nuevas series de pruebas utilicen la TTCN-3 definida en las Recomendaciones UIT-T de la serie Z.140.

RECOMENDACIÓN UIT-T X.292

Metodología y marco de las pruebas de conformidad para interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del UIT-T – Notación combinada arborescente y tabular[1]

El UIT-T,

considerando

a) que la Rec. UIT-T X.200 define el modelo de referencia a interconexión de sistemas abiertos (OSI, *open systems interconnection*) para aplicaciones del UIT-T;

b) que el objetivo de OSI no se logrará completamente hasta que puedan probarse los sistemas para determinar si son conformes o no con las Recomendaciones pertinentes sobre protocolos de OSI;

c) que se deben elaborar series de pruebas normalizadas para cada Recomendación UIT-T sobre protocolos de OSI para:

- obtener una amplia aceptación y confianza en los resultados de las pruebas de conformidad elaboradas por diferentes probadores;
- proporcionar seguridad en la interoperabilidad de los equipos que pasan las pruebas de conformidad normalizadas;

d) la necesidad de normalizar el proceso de prueba de conformidad para lograr un grado aceptable y útil de comparación de los resultados de las evaluaciones de conformidad de productos similares,

recomienda por unanimidad

que la notación en la que se especifiquen los casos de pruebas abstractas y genéricas se elija con arreglo a esta Recomendación.

1 Alcance

En esta Recomendación se define una notación de prueba informal denominada "notación combinada arborescente y tabular"(TTCN) para las series de pruebas de conformidad de OSI, que es independiente de los métodos de prueba, capas y protocolos, y que refleja la metodología de las pruebas abstractas definida en las Recomendaciones UIT-T X.290 y X.291.

También se especifican los requisitos y se proporcionan orientaciones para la utilización de la TTCN en la especificación de series de pruebas de conformidad independientes del sistema, para una o más normas OSI. Se especifican dos formas de notación: una de ellas interpretable por personas, que resulta aplicable a la elaboración de normas de series de pruebas de conformidad para protocolos OSI y la otra, en forma procesable por máquina, aplicable al procesamiento dentro de sistemas de computador y entre ellos.

Esta Recomendación se aplica a la especificación de casos de prueba de conformidad que se pueden formular de forma abstracta en términos de control y la observación de unidades de datos de protocolo y primitivas de servicio abstractas. Sin embargo, para algunos protocolos, puede ser necesario utilizar casos de prueba que no se pueden formular en estos términos. La especificación de tales casos de prueba queda fuera del ámbito de esta Recomendación, aunque puede ser necesaria la inclusión de esos casos en una Recomendación norma de serie de pruebas de conformidad.

Por ejemplo, algunos requisitos de conformidad estática relacionados con un servicio de aplicación pueden exigir técnicas de prueba específicas de esa aplicación particular.

---

[1] La Rec. UIT-T X.292 (2002) es una actualización de la Rec. UIT-T X.292 (1998) y la norma ISO/CEI 9646-3:1998 técnicamente alineadas, Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 3: The Tree and Tabular Combined Notation (TTCN).

La especificación de casos de prueba en los cuales se deben ejecutar en paralelo varias descripciones de comportamiento, se trata mediante las características de concurrencia (particularmente mediante la definición de los componentes de pruebas y configuraciones de componentes de prueba).

Esta Recomendación especifica los requisitos que una norma de serie de pruebas podría especificar para una realización conforme de la serie de pruebas, incluida la semántica operacional de las series de pruebas con TTCN.

Esta Recomendación se aplica a la especificación de las series de pruebas de conformidad para los protocolos OSI de las capas OSI 2 a 7, incluidos específicamente los protocolos basados en la notación de sintaxis abstracta uno (ASN.1, *abstract syntax notation one*). Los siguientes aspectos están fuera del alcance de esta Recomendación:

a) la especificación de series de pruebas de conformidad para protocolos de la capa física;

b) la relación entre la TTCN y las técnicas de descripción formal;

c) los medios para realizar series de pruebas ejecutables (ETS, *executable test suites*) a partir de series de pruebas abstractas.

En esta Recomendación se definen los mecanismos para el uso de la concurrencia en la especificación de los casos de pruebas abstractas. La concurrencia en TTCN es aplicable a la especificación de los casos de prueba:

a) en un contexto de comprobación multiparte;

b) que manejan la multiplexación y la demultiplexación tanto en un contexto uniparte como multiparte;

c) que manejan el fraccionamiento y la recombinación tanto en un contexto uniparte como multiparte;

d) en un contexto de comprobación uniparte, cuando la complejidad del protocolo o conjunto de protocolos manejados por la implementación sometida a prueba es tal que la concurrencia puede simplificar la especificación del caso de prueba.

Se definen los módulos TTCN para permitir la compartición entre series de pruebas de las especificaciones TTCN comunes.

2 Referencias normativas

Las siguientes Recomendaciones del UIT-T y otras referencias contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación. Al efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y otras referencias son objeto de revisiones por lo que se preconiza que los usuarios de esta Recomendación investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y otras referencias citadas a continuación. Se publica periódicamente una lista de las Recomendaciones UIT-T actualmente vigentes. En esta Recomendación, la referencia a un documento, en tanto que autónomo, no le otorga el rango de una Recomendación.

2.1 Recomendaciones | Normas Internacionales idénticas

– Recomendación UIT-T X.200 (1994) | ISO/CEI 7498-1:1994, *Tecnología de la información – Interconexión de sistemas abiertos – Modelo de referencia básico: El modelo básico*.

– Recomendación UIT-T X.210 (1993) | ISO/CEI 10731:1994, *Tecnología de la información – Interconexión de sistemas abiertos – Modelo de referencia básico: Convenios para la definición de servicios en la interconexión de sistemas abiertos*.

– Recomendación UIT-T X.680 (1997) | ISO/CEI 8824-1:1998, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de la notación básica*.

– Recomendación UIT-T X.681 (1997) | ISO/CEI 8824-2:1998, *Tecnología de la información – Notación de sintaxis abstracta uno – Especificación de objetos de información*.

– Recomendación UIT-T X.682 (1997) | ISO/CEI 8824-3:1998, *Tecnología de la información – Notación de sintaxis abstracta uno – Especificación de constricciones*.

– Recomendación UIT-T X.683 (1997) | ISO/CEI 8824-4:1998, *Tecnología de la información – Notación de sintaxis abstracta uno – Parametrización de las especificaciones de la notación de sintaxis abstracta uno*.

– Recomendación UIT-T X.690 (1997) | ISO/CEI 8825-1:1998, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno – Especificación de las reglas de codificación básica, de las reglas de codificación canónica y de las reglas de codificación distinguida*.

– Recomendación UIT-T X.691 (1997) | ISO/CEI 8825-2:1998, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno – Especificación de las reglas de codificación compactada*.

2.2 Pares de Recomendaciones | Normas Internacionales cuyo contenido técnico es equivalente

– Recomendación UIT-T X.290 (1995), *Metodología y marco de las pruebas de conformidad de interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del UIT-T – Conceptos generales*.

ISO/CEI 9646-1:1994, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 1: General concepts*.

– Recomendación UIT-T X.291 (1995), *Metodología y marco de las pruebas de conformidad de interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del UIT-T – Especificación de sucesiones de pruebas abstractas*.

ISO/CEI 9646-2:1994, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 2: Abstract test suite specification*.

– Recomendación UIT-T X.293 (1995), *Metodología y marco de las pruebas de conformidad de interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del UIT-T – Realización de pruebas*.

ISO/CEI 9646-4:1994, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 4: Test realization*.

– Recomendación UIT-T X.294 (1995), *Metodología y marco de las pruebas de conformidad de interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del UIT-T – Requisitos que deberán cumplir los laboratorios de pruebas y los clientes en el proceso de evaluación de la conformidad*.

ISO/CEI 9646-5:1994, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 5: Requirements on test laboratories and clients for the conformance assessment process*.

– Recomendación UIT-T X.295 (1995), *Metodología y marco de las pruebas de conformidad de interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del UIT-T – Especificación de pruebas de perfil de protocolo*.

ISO/CEI 9646-6:1994, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 6: Protocol profile test specification*.

– Recomendación UIT-T X.296 (1995), *Metodología y marco de las pruebas de conformidad de interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del UIT-T – Declaraciones de conformidad de implementación.*

ISO/CEI 9646-7:1995, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 7: Implementation conformance statements*.

2.3 Referencias adicionales

– ISO/CEI 646:1991, *Information technology – ISO 7-bit coded character set for information interchange*.

– ISO/CEI 10646-1:2000, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*.

3 Definiciones

3.1 Términos básicos extraídos de la Rec. UIT-T X.290

A los efectos de esta Recomendación se aplican los siguientes términos definidos en la Rec. UIT-T X.290:

a) *primitiva de servicio abstracta;*

b) *metodología de comprobación abstracta;*

c) *caso de prueba abstracta;*

d) *método de prueba abstracta;*

e) *serie (secuencia) de pruebas abstractas;*

f) *registro cronológico de conformidad;*

g) *serie (secuencia) de pruebas de conformidad;*
h) *método de prueba coordinada;*
i) *método de prueba distribuida;*
j) *caso de prueba ejecutable;*
k) *error de caso de prueba ejecutable;*
l) *serie (secuencia) de pruebas ejecutables;*
m) *veredicto de fracaso;*
n) *estado de comprobación en reposo;*
o) *implementación sometida a prueba;*
p) *veredicto de no concluyente;*
q) *evento de prueba no válido;*
r) *método de prueba local;*
s) *probador inferior;*
t) *medio(s) de comprobación;*
u) *veredicto de éxito;*
v) *formulario de PICS;*
w) *formulario de PIXIT;*
x) *declaración de conformidad de implementación de protocolo;*
y) *información suplementaria de implementación de protocolo para pruebas;*
z) *punto de control y observación;*
aa) *método de prueba a distancia;*
ab) *estado de comprobación estable;*
ac) *serie (secuencia) de pruebas abstractas normalizadas;*
ad) *requisitos de conformidad estática;*
ae) *evento de prueba sintácticamente no válido;*
af) *sistema sometido a prueba;*
ag) *cuerpo de prueba;*
ah) *caso de prueba;*
ai) *error de caso de prueba;*
aj) *procedimientos de coordinación de pruebas;*
ak) *evento de prueba;*
al) *grupo de prueba;*
am) *objetivo de grupo de prueba;*
an) *laboratorio de pruebas;*
ao) *protocolo de gestión de prueba;*
ap) *resultado de la prueba;*
aq) *epílogo (de una prueba);*
ar) *prólogo (de una prueba);*
as) *finalidad de la prueba;*
at) *realización de la prueba;*
au) *realizador de la prueba;*
av) *paso de prueba;*
aw) *serie (secuencia) de pruebas;*
ax) *sistema de prueba;*
ay) *probador superior;*
az) *veredicto (de una prueba);*
ba) *estado de comprobación.*

3.2 Términos de la Rec. UIT-T X.200

A los efectos de esta Recomendación se aplican los siguientes términos definidos en la Rec. UIT-T X.200:

a) *capa (en particular para las capas de aplicación, de sesión y de transporte);*

b) *unidad de datos de protocolo;*

c) *punto de acceso al servicio;*

d) *subred;*

e) *sintaxis de transferencia.*

3.3 Términos de la Rec. UIT-T X.210

A los efectos de esta Recomendación se aplica el siguiente término definido en la Rec. UIT-T X.210:

a) *proveedor del servicio OSI.*

3.4 Términos de la Rec. UIT-T X.680

A los efectos de esta Recomendación se aplican los siguientes términos definidos en la Rec. UIT-T X.680:

a) *tipo cadena de bits (bitstring);*

b) *tipo cadena de caracteres (characterstring);*

c) *tipo enumerado (enumerated);*

d) *tipo externo (external);*

e) *identificador de objeto (object identifier);*

f) *tipo cadena de octetos (octectstring);*

g) *tipo real (real type);*

h) *tipo selección (selection);*

i) *tipo secuencia (sequence);*

j) *tipo secuencia-de (sequence-of);*

k) *tipo conjunto (set);*

l) *tipo conjunto-de (set-of);*

m) *subtipo (subtype).*

NOTA – Cuando pueda existir ambigüedad en los términos de la TTCN, esos términos llevarán el término en ASN.1 como prefijo.

3.5 Términos de la Rec. UIT-T X.690

A los efectos de esta Recomendación se aplica el siguiente término definido en la Rec. UIT-T X.690:

a) *codificación.*

3.6 Términos específicos de la TTCN

En esta Recomendación se definen los términos siguientes:

3.6.1 reglas de codificación aplicables: Reglas de codificación reales que se utilizarán en la emisión o recepción de una PDU después de que se hayan combinado, si existen, todas las codificaciones de sustitución y por defecto pertinentes.

3.6.2 constructivo adjuntar: Enunciado en notación combinada arborescente y tabular que adjunta un paso de prueba a un árbol llamante.

3.6.3 constricción de base: Especifica un conjunto de valores por defecto para todos y cada uno de los campos en una definición de tipo de primitiva de servicio abstracto o de unidad de datos de protocolo.

3.6.4 tipo de base: Tipo del que se deriva un tipo definido en una serie de pruebas.

3.6.5 línea de comportamiento: Entrada de un cuadro de comportamiento dinámico que representa un evento de prueba u otro enunciado en notación combinada arborescente y tabular, junto con una etiqueta asociada, veredicto, referencia a constricción e información de comentario, según sea aplicable.

3.6.6 árbol de comportamiento: Especificación de un conjunto de secuencias de eventos de prueba y otros enunciados en notación combinada arborescente y tabular.

3.6.7 casilla en blanco: En un cuadro compacto de constricciones modificadas, una casilla en blanco en un parámetro o campo de constricción o indica que se hereda un valor de constricción.

3.6.8 bol llamante: Árbol de comportamiento al que se adjunta un paso de prueba.

3.6.9 cuadro compacto de constricciones: Declaración de un conjunto de constricciones para una primitiva de servicio abstracta, una unidad de datos de protocolo o un tipo estructurado, dispuesta en forma de un solo cuadro.

3.6.10 cuadro compacto de casos de prueba: Declaración de un conjunto de casos de prueba para un grupo de prueba determinado, dispuesta en forma de un solo cuadro.

3.6.11 caso de prueba concurrente: Caso de prueba especificado mediante la notación combinada arborescente y tabular concurrente.

3.6.12 notación combinada arborescente y tabular concurrente: Notación combinada arborescente y tabular que utiliza componentes de prueba y configuraciones de componentes de prueba para expresar la concurrencia en el comportamiento dinámico de casos de prueba.

3.6.13 parte constricciones: Parte de una serie de pruebas con notación combinada arborescente y tabular relacionada con la especificación de los valores de parámetros de primitiva de servicio abstracta y campos de unidad de datos de protocolo enviados a la implementación sometida a prueba y condiciones relativas a los parámetros de primitiva de servicio abstracta y campos de unidad de datos de protocolo recibidas de la implementación sometida a prueba.

3.6.14 referencia a constricción: Referencia a una constricción contenida en una línea de comportamiento.

3.6.15 mensaje de coordinación (CM, *co-ordination message*): Elemento de información estructurada que puede ser transferido de un componente de prueba a otro en un punto de coordinación.

3.6.16 punto de coordinación (CP, *co-ordination point*): Punto dentro de un entorno de prueba asignado a dos componentes de prueba en una configuración de componentes de prueba, en el que se puede intercambiar de manera asíncrona mensajes de coordinación entre estos componentes de prueba.

3.6.17 parte declaraciones: Parte de una serie de pruebas con notación combinada arborescente y tabular con la definición y/o declaración de todos los componentes no predefinidos que se usan en la serie de pruebas.

3.6.18 comportamiento por defecto: Eventos y otros enunciados en notación combinada arborescente y tabular que se pueden producir a cualquier nivel del árbol asociado y que se indican en el formulario de comportamiento por defecto.

3.6.19 grupo de valores por defecto: Conjunto denominado de comportamientos por defecto.

3.6.20 referencia a grupo de valores por defecto: Trayecto que especifica la ubicación lógica de un elemento por defecto en la biblioteca de valores.

3.6.21 identificador de valor por defecto: Nombre exclusivo de un valor por defecto.

3.6.22 biblioteca de valores por defecto: Conjunto de comportamientos por defecto de una serie de pruebas.

3.6.23 referencia a valor por defecto: Referencia a un valor de la biblioteca de valores por defecto de un caso de prueba o un cuadro de pasos de prueba.

3.6.24 trayecto de derivación: Identificador que consta de un identificador de constricción de base, concatenado con uno o más identificadores de constricción modificados, separados por puntos y que terminan con un punto.

3.6.25 encadenamiento dinámico: Vinculación, mediante parametrización, de las declaraciones de constricción de un parámetro primitiva de servicio abstracta o de un campo de unidad de datos de protocolo a la declaración de constricción de otra unidad de datos de protocolo. Las unidades de datos de protocolo que son encadenadas se especifican en la referencia a constricción de una línea de comportamiento.

3.6.26 parte dinámica: Parte de una serie de pruebas con notación combinada arborescente y tabular relacionada con la especificación de descripciones de caso de prueba, paso de prueba y comportamiento dinámico por defecto.

3.6.27 serie de pruebas expandida: Serie de pruebas ampliada con todos los objetos importados. Es el resultado de convertir una serie de pruebas modularizada con el algoritmo del anexo B.

3.6.28 externo explícito: Objeto denominado en el cuadro de externos. Un objeto que es declarado explícitamente como externo en un módulo será explícitamente definido o exportado como un objeto externo.

3.6.29 objeto definido explícitamente: Objeto para el cual existe una definición o declaración en el módulo o serie de pruebas.

3.6.30 objeto exportado explícitamente: Objeto denominado en los cuadros de exportaciones disponible para su utilización. Si el objeto es un objeto importado, se le dará el nombre del objeto fuente.

3.6.31 objeto importado explícitamente: Objeto denominado en los cuadros de importaciones que se encuentra disponible para referencias explícitas.

3.6.32 objeto exportado: Objeto definido explícitamente u objeto importado explícitamente de un objeto fuente, puesto a disposición para su utilización en cualquier otro módulo o serie de pruebas. Un objeto exportado es un objeto exportado explícitamente o un objeto exportado implícitamente.

3.6.33 objeto externo: Objeto al que se hace referencia por el nombre en un módulo, pero que no es ni importado ni definido explícitamente. Un objeto externo se declara en el cuadro de externos. Un objeto externo puede ser externo explícitamente o externo implícitamente.

3.6.34 variable de resultado global: Variable de caso de prueba predefinida mantenida por un componente de prueba principal en el contexto MP y T o por el caso de prueba en el contexto SP y T para registrar el efecto acumulado de todos los resultados preliminares del caso de prueba con el fin de determinar el veredicto de prueba.

3.6.35 externo implícito: Objeto declarado externamente en un cuadro de exportaciones que no se menciona en el cuadro de importaciones correspondiente.

3.6.36 objeto exportado implícitamente: Objeto definido explícitamente u objeto importado explícitamente, que no es en sí mismo exportado explícitamente pero al que se hace referencia por un objeto exportado explícitamente.

3.6.37 objeto importado implícitamente: Objeto al que se hace referencia mediante algún objeto importado explícitamente. El uso de un objeto importado implícitamente está restringido a los objetos importados explícitamente (a partir de algún objeto fuente) que hacen referencia al mismo.

3.6.38 evento enviar implícito: Mecanismo utilizado en los métodos de prueba a distancia para especificar que la implementación sometida a prueba debe iniciar una unidad de datos de protocolo o una primitiva de servicio abstracta concreta.

3.6.39 objeto importado: Objeto copiado de algún otro objeto fuente, que se encuentra disponible para su utilización. Un objeto importado es un objeto importado explícitamente o un objeto importado implícitamente.

3.6.40 nivel de sangrado: Indica la estructura de árbol de una descripción de comportamiento. Se refleja en la descripción de comportamiento mediante el sangrado de un texto.

3.6.41 variable de resultado local: Variable predefinida mantenida por un componente de prueba para registrar el efecto acumulado de sus resultados preliminares.

3.6.42 árbol local: Árbol de comportamiento definido en el mismo formulario que su árbol llamante.

3.6.43 componente de prueba principal *(*MTC*, main test component)*: Componente de prueba único en una configuración de componentes de prueba, que se encarga de la creación y control de los componentes de prueba paralelos y de calcular y asignar el veredicto de prueba.

3.6.44 constricción modificada: Constricción establecida para una primitiva de servicio abstracta o una unidad de datos de protocolo que tiene ya una constricción de base y que efectúa modificaciones a esa constricción de base.

3.6.45 serie de pruebas modularizada: Serie de pruebas que contiene cuadros de importaciones.

3.6.46 módulo: Colección autónoma de objetos en notación combinada arborescente y tabular. Todos los objetos referenciados se definen explícitamente en el módulo, se importan de otras fuentes o se definen como objetos externos en el módulo.

3.6.47 caso de prueba no concurrente: Caso de prueba especificado en notación combinada arborescente y tabular pero que no utiliza notación combinada arborescente y tabular concurrente.

3.6.48 objeto: Elemento de una de las categorías de objetos relacionadas en A.4.2 para objetos en TTCN con un identificador globalmente único y para identificadores ASN.1 que son globalmente exclusivos a lo largo de la serie de pruebas.

3.6.49 semántica operacional: Semántica que explica la ejecución de un árbol de comportamiento en notación combinada arborescente y tabular.

3.6.50 objeto fuente original: Módulo o serie de pruebas en el que un objeto se define explícitamente.

3.6.51 evento en otro caso: Mecanismo de notación combinada arborescente y tabular previsto para manejar eventos de prueba imprevistos de una manera controlada.

3.6.52 parte visión general: Parte de una serie de pruebas en notación combinada arborescente y tabular relacionada con la presentación de una visión general de la estructura de la serie de pruebas, la estructura (si existe) de la biblioteca de pasos de prueba, la estructura (si existe) de la biblioteca de valores por defecto y la asociación entre las expresiones de selección (si existen) y los casos de prueba y/o grupos de prueba. Esta parte proporciona, asimismo, índices de los casos de prueba, pasos de prueba y valores por defecto.

3.6.53 componente de prueba paralelo (PTC, *parallel test component*): Componente de prueba creado por el componente de prueba principal.

3.6.54 resultado preliminar: Resultado registrado antes de la finalización de un caso de prueba, que indica si la parte asociada del caso de prueba ha pasado la prueba, ha fallado o no hay conclusión al respecto.

3.6.55 seudoevento: Un seudoevento es una expresión en notación combinada arborescente y tabular o una operación de temporizador que aparece en una línea de enunciado en la descripción de comportamiento sin ningún evento asociado.

3.6.56 evento calificado: Evento que tiene asociada una expresión booleana.

3.6.57 evento recibir: Recepción de una primitiva de servicio abstracta o una unidad de datos de protocolo en un punto de control y observación denominado o implicado.

3.6.58 variable de resultado: Variable de caso de prueba predefinida para el almacenamiento de los resultados preliminares. En notación combinada arborescente y tabular no concurrente hay una variable de resultado llamada R. En notación combinada arborescente y tabular concurrente, hay una variable de resultado global llamada R, cada PTC tiene una variable resultado local llamada R y el MTC tiene una variable resultado local llamada MTC_R.

3.6.59 árbol raíz: Árbol de comportamiento principal de un caso de prueba que se presenta en el nivel de entrada del caso de prueba.

3.6.60 evento enviar: Envío de una primitiva de servicio abstracta o una unidad de datos de protocolo o un punto de control y observación denominado o implicado.

3.6.61 conjunto de alternativas: Enunciados en notación combinada arborescente y tabular codificados al mismo nivel de sangrado y pertenecientes al mismo nodo predecesor. Representan los posibles eventos, seudoeventos y constructivos que han de considerarse en el punto pertinente de la ejecución del caso de prueba.

3.6.62 cuadro único de constricciones: Declaración de una constricción para una sola primitiva de servicio abstracta o unidad de datos de protocolo de un tipo determinado, dispuesta en forma de un solo cuadro.

3.6.63 semántica instantánea: Modelo semántico previsto para eliminar el efecto de la temporización en la ejecución de un caso de prueba, definido en términos de instantáneas del entorno de prueba, durante las cuales te entorno se congela efectivamente durante un periodo prescrito.

3.6.64 objeto fuente: Módulo o serie de pruebas que es importado y tiene un cuadro de importaciones correspondiente.

3.6.65 valor específico: Valor en notación combinada arborescente y tabular que no contiene ningún mecanismo de concordancia o variable no acotada.

3.6.66 encadenamiento estático: Vinculación de declaraciones de constricciones de un parámetro una primitiva de servicio abstracta o campo de una unidad de datos de protocolo a la declaración de constricción de otra unidad de datos de protocolo, mediante una referencia explícita a la constricción como su valor.

3.6.67 semántica estática: Reglas semánticas que restringen la utilización de la sintaxis de notación combinada arborescente y tabular.

3.6.68 tipo estructurado: Colección de uno o más parámetros de primitiva de servicio abstracta o campos de unidad de datos de protocolo que pueden existir en una o más definiciones de tipo de primitiva de servicio abstracta o de unidad de datos de protocolo y que se han definido en una declaración separada, que se puede usar para especificar una porción de estructura plana o subestructura dentro de la primitiva de servicio abstracta o la unidad de datos de protocolo.

3.6.69 submódulo: Módulo incluido en otro módulo.

3.6.70 identificador de caso de prueba: Nombre exclusivo de un caso de prueba.

3.6.71 variable de caso de prueba: Una de un conjunto de variables, declarada globalmente para la serie de pruebas, cuyo valor se mantiene solamente para la ejecución de un solo caso de prueba.

3.6.72 componente de prueba: Subdivisión denominada de un caso de prueba concurrente que se puede ejecutar en paralelo con otros componentes de prueba y de la que se declara que dispone de un número fijo de PCO y de un número fijo o un número máximo de puntos de coordinación.

3.6.73 configuración de componentes de prueba: Disposición fija de componentes de prueba, puntos de control y observación (PCO) y puntos de coordinación (CP) que se declara para su utilización en casos de prueba concurrentes.

3.6.74 referencia a grupo de prueba: Trayecto que especifica la ubicación lógica de un caso de prueba dentro de la estructura de la serie de pruebas abstractas.

3.6.75 grupo de pasos de prueba: Un conjunto denominado de pasos de prueba.

3.6.76 referencia a grupo de pasos de prueba: Trayecto que especifica la ubicación lógica de un paso de prueba en la biblioteca de pasos de prueba.

3.6.77 identificador de paso de prueba: Nombre exclusivo de un paso de prueba.

3.6.78 biblioteca de pasos de prueba: Conjunto de descripciones de comportamiento dinámico de pasos de prueba en la serie de pruebas, que no son pasos de prueba locales.

3.6.79 objetivo de paso de prueba: Declaración informal de lo que se pretende que realice el paso de prueba.

3.6.80 constante de serie de pruebas: Una de un conjunto de constantes no derivadas del enunciado conformidad de implementación de protocolo o información suplementaria de implementación de protocolo para pruebas, que permanecerá constante en la serie de pruebas.

3.6.81 parámetro serie de pruebas: Una de un conjunto de constantes derivadas del enunciado conformidad de implementación de protocolo o de la información suplementaria de implementación de protocolo para pruebas, que parametriza globalmente una serie de pruebas.

3.6.82 variable de serie de pruebas: Una de un conjunto de variables declaradas globalmente para la serie de pruebas y que mantiene su valor en los casos de prueba.

3.6.83 evento de temporización: Evento utilizado dentro de un árbol de comportamiento para verificar la expiración de un temporizador especificado.

3.6.84 adjunción de árbol: Método para indicar que hay que incluir un árbol de comportamiento especificado en cualquier otra parte (ya sea en un punto diferente del formulario en vigor o como paso de prueba en biblioteca de pasos de prueba) en el árbol de comportamiento vigente.

3.6.85 encabezamiento de árbol: Identificador de un árbol local seguido de una lista optativa de los parámetros formales del árbol.

3.6.86 identificador de árbol: Nombre que identifica un árbol local.

3.6.87 hoja de árbol: Enunciado en notación combinada arborescente y tabular de un árbol de comportamiento o paso de prueba que no ha especificado un comportamiento subsiguiente.

3.6.88 nodo de árbol: Enunciado en notación combinada arborescente y tabular única.

3.6.89 notación de árbol: Notación utilizada en notación combinada arborescente y tabular para representar los casos de prueba como árboles.

3.6.90 enunciado en TTCN: Evento, seudoevento o constructivo especificado en una descripción de comportamiento.

3.6.91 evento de prueba imprevisto: Evento de prueba no identificado como evento de prueba en el resultado de la prueba previsto en la serie de pruebas. Se maneja habitualmente con el evento OTHERWISE (en otro caso).

3.6.92 evento no calificado: Evento que no tiene asociada una expresión booleana.

4 Abreviaturas

4.1 Abreviaturas definidas en la Rec. UIT-T X.290

A los efectos de esta Recomendación se aplican las siguientes siglas definidas en la cláusula 4/X.290:

ASP	Primitiva de servicio abstracta (*abstract service primitive*)
ATS	Sucesión de pruebas abstractas (*abstract test suite*)
ETS	Sucesión de pruebas ejecutables (*executable test suite*)
IUT	Implementación sometida a prueba (*implementation under test*)
LT	Probador inferior (*lower tester*)

LTCF Función de control de probador inferior (*lower tester control function*)

MOT Medio de pruebas (*means of testing*)

PCO Punto de control y observación (*point of control and observation*)

PICS Declaración de conformidad de implementación de protocolo (*protocol implementation conformance statement*)

PIXIT Información suplementaria de implementación de protocolo para pruebas (*protocol implementation extra information for testing*)

SUT Sistema sometido a prueba (*system under test*)

TMP Protocolo de gestión de las pruebas (*test management protocol*)

UT Probador superior (*upper tester*)

UTCF Función de control de probador superior (*upper tester control function*)

4.2 Abreviaturas definidas en la Rec. UIT-T X.291

A los efectos de esta Recomendación se aplican las siguientes siglas definidas en la cláusula 4/X.291:

CS Método de pruebas monocapa coordinada (*co-ordinated single-layer (test method)*

DS Método de pruebas monocapa distribuido [*distributed single-layer (test method)*]

LS Método de pruebas monocapa local [*local single-layer (test method)*]

RS Método de pruebas monocapa a distancia [*remote single-layer (test method)*]

TTCN Notación combinada arborescente y tabular (*tree and tabular combined notation*)

4.3 Otras abreviaturas

En esta Recomendación se utilizan las siguientes siglas:

ASN.1 Notación de sintaxis abstracta uno (*abstract syntax notation one*)

BNF Forma Backus-Naur ampliada que se usa en la TTCN (*the extended Backus-Naur form used in TTCN*)

CM Mensaje de coordinación (*co-ordination message*)

CP Punto de coordinación (*co-ordination point*)

FDT Técnica de descripción formal (*formal description technique*)

FIFO Primero en entrar, primero en salir (*first in first out*)

MTC Componente de prueba principal (*main test component*)

OSI Interconexión de sistemas abiertos (*open systems interconnection*)

PDU Unidad de datos de protocolo (*protocol data unit*)

PTC Componente de prueba paralelo (*parallel test component*)

SAP Punto de acceso al servicio (*service access point*)

TCP Procedimientos de coordinación de las pruebas (*test coordination procedures*)

TTCN.GR Notación combinada arborescente y tabular, forma gráfica (*tree and tabular combined notation, graphical form*)

TTCN.MP Notación combinada arborescente y tabular, forma procesable por máquina (*tree and tabular combined notation, machine processable form*)

5 Las formas de sintaxis de la TTCN

La TTCN se proporciona en dos formas:

- una forma gráfica (TTCN.GR)que pueden leer las personas;
- una forma procesable por máquina (TTCN.MP), adecuada para la transmisión de descripciones con TTCN entre máquinas y posiblemente idónea para otro tipo de procesamiento automatizado.

La TTCN.GR se define mediante formularios tabulares. La TTCN.MP se define mediante producciones de sintaxis que tienen palabras clave de TTCN.MP especiales como símbolos terminales, en vez de las partes fijas de los formularios tabulares (por ejemplo, líneas de casilla y encabezamientos).

En el anexo A se especifican las producciones de sintaxis de TTCN.MP.

Se ha previsto que la descripción textual de la TTCN.GR sea coherente con la sintaxis subyacente tal y como se define en las producciones de sintaxis de la TTCN.MP, excepto las diferencias indicadas en A.5 y las constricciones de semántica estática especificadas en el anexo A (que son comunes a la TTCN.MP y la TTCN.GR).

Si hay alguna contradicción entre la sintaxis de TTCN.GR, por un lado, y las semánticas estática y operacional, por otro, descritas en el texto y descritas en el anexo A, se observará lo siguiente:

a) excepto en el caso de las diferencias especificadas en A.5, las producciones de sintaxis de la TTCN.MP tendrán precedencia sobre el texto y producciones de sintaxis contenidas en el cuerpo de esta Recomendación

b) las restricciones de semántica estática especificadas en A.4 y en los comentarios de semántica estática (denominados SEMÁNTICA ESTÁTICA) relativos a las producciones de sintaxis de A.3 especifican restricciones de la parte válida de la TTCN, restringiendo lo que es admisible según las producciones de la sintaxis;

c) de manera similar, las restricciones de semántica operacional especificadas en los comentarios de semántica operacional (denominados SEMÁNTICA OPERACIONAL) relativos a las producciones de sintaxis de A.3 especifican restricciones de la parte válida de la TTCN en el momento de la ejecución, restringiendo lo que es admisible según las producciones de la sintaxis;

d) las restricciones de semántica estática y operacional especificadas en el anexo A tendrán precedencia sobre el texto del cuerpo de esta Recomendación.

Si una ATS se especifica en TTCN.GR de conformidad con esta Recomendación, existe entonces una sola representación correspondiente con TTCN.MP de esa ATS que comparte la misma sintaxis subyacente. Estas dos representaciones tienen semánticas operacionales idénticas. Dos representaciones diferentes de una ATS son equivalentes si y sólo si tienen idénticas semánticas operacionales.

NOTA – Si hay una ATS normalizada especificada en TTCN.GR y una representación con TTCN.MP aparentemente equivalente, pero hay una discrepancia en la interpretación de la semántica operacional de las dos, tendrá precedencia la semántica operacional de la TTCN.GR, porque la ATS normalizada es la versión en TTCN.GR.

6 Cumplimiento

Las ATS que cumplan esta Recomendación satisfarán los requisitos de la TTCN.GR o la TTCN.MP.

NOTA 1 – Para la explicación del uso del término "cumplimiento" en las Recomendaciones UIT-T de la serie X.290, véase la cláusula 10/X.290.

Una ATS que cumpla los requisitos de la TTCN.GR satisfará los requisitos de sintaxis de la TTCN.GR establecidos en las cláusulas 9 a 16 y A.4.

Una ATS que cumpla los requisitos de la TTCN.MP satisfará los requisitos de sintaxis de la TTCN.MP establecidos en A.3.

Las ATS que cumplan esta Recomendación satisfarán los requisitos de semántica estática establecidos en las cláusulas 7 a 16 y anexo A y tendrán una semántica operacional conforme con la definición de semántica operacional del anexo B, junto con las restricciones de semántica operacional especificadas en A.3, de forma que sean semánticamente válidas.

Una ATS normalizada que cumpla esta Recomendación tiene por requisito que toda realización de una serie de pruebas que pretenda ser conforme con la ATS normalizada:

a) tenga una semántica operacional equivalente a la semántica operacional de la serie de pruebas tal y como se define en el anexo B;

b) cumpla los requisitos de la semántica operacional adicionales especificados en A.3;

c) cumpla la Rec. UIT-T X.293.

NOTA 2 – Si durante la ejecución del caso de prueba ejecutable, que cumple la especificación con TTCN del caso de prueba abstracta correspondiente, se detecta un error semántico estático o semántico operacional, el laboratorio de pruebas que cumple la Rec. UIT-T X.294 deberá registrar un error de caso de prueba ejecutable o de caso de prueba abstracta según donde se haya producido el error.

7 Convenios

7.1 Introducción

Para la definición de los formularios de cuadro en TTCN.GR y de la gramática en TTCN.MP se han utilizado los siguientes convenios.

7.2 Metanotación sintáctica

En el cuadro 1 se define la metanotación utilizada para especificar la forma ampliada de gramática BNF para la TTCN (en adelante BNF).

Cuadro 1/X.292 – Metanotación sintáctica en TTCN.MP

::=	Definición
abc xyz	abc seguido de xyz
|	Alternativa
[abc]	0 ó 1 instancia de abc
{abc}	0 o más instancias de abc
{abc}+	1 o más instancias de abc
(...)	Agrupación textual
abc	El símbolo no terminal abc
abc	Un símbolo terminal abc
"abc"	Un símbolo terminal abc

EJEMPLO 1 – Utilización de la metanotación BNF

FormalParList ::= "(" FormalPar&Type {SemiColon FormalPar&Type} ")"

En el texto de los formularios de cuadro se emplearán los siguientes convenios:

a) El texto en negrita (**como éste**) aparecerá palabra por palabra en cada cuadro real de una serie de pruebas con TTCN.

b) El texto en cursiva (*como éste*) no aparecerá exactamente palabra por palabra en una serie de pruebas con TTCN. Este tipo de caracteres se utiliza para indicar que el texto real debe sustituir al símbolo en cursiva. Los requisitos de sintaxis para el texto real figuran en la producción con TTCN.MP BNF correspondiente.

EJEMPLO 2 – SuiteIdentifier corresponde a la producción 3 del anexo A.

7.3 Formularios de cuadro en TTCN.GR

7.3.1 Introducción

La TTCN.GR se define mediante dos tipos de cuadro:

a) cuadro de objeto TTCN único (véase 7.3.2),

utilizado para definir, declarar o describir un solo objeto TTCN, como una declaración de PDU o un comportamiento dinámico de caso de prueba;

b) cuadro de objeto TTCN múltiple (véase 7.3.3),

utilizado para definir cierto número de objetos TTCN del mismo tipo en un solo cuadro, como las definiciones de tipo simple o las variables de caso de prueba.

7.3.2 Cuadro para objeto TTCN único

En la figura 1 se muestra la disposición general de un cuadro para un objeto TTCN único.

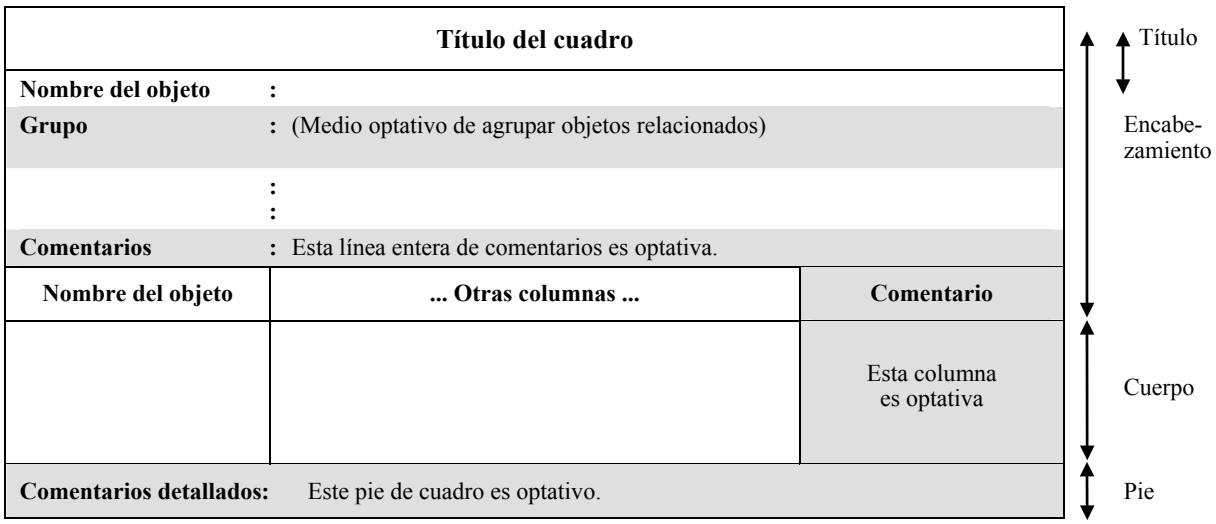


Figura 1/X.292 – Disposición general del cuadro para declarar un objeto único

El encabezamiento del cuadro contiene información general sobre el objeto definido en el cuadro. El primer elemento del encabezamiento, denominado *Nombre del objeto*, contiene un identificador del objeto. Un segundo elemento, denominado *Grupo,* puede emplearse para proporcionar un identificador para agrupar objetos relacionados de la misma categoría. Este elemento es optativo. El último elemento, denominado *Comentarios* contiene una descripción informal del objeto. Este elemento es optativo.

El cuerpo del cuadro consta de una o más columnas. Cada columna tiene un título. La columna situada más a la derecha denominada *Comentarios*, contiene descripciones informales de los componentes del objeto especificado en el cuerpo. No existe en todos los formularios. En los formularios que contengan una columna de comentarios se puede omitir esta columna.

El pie de cuadro contiene un elemento, denominado *Comentarios detallados*. Se puede utilizar este pie para los mismos fines que la columna comentarios del cuerpo del cuadro. El especificador de la serie de pruebas puede utilizar los comentarios detallados que figuran en el pie en combinación con la columna de comentarios, en vez de la columna de comentarios, o no utilizarlos en absoluto, en cuyo caso se puede omitir el pie del cuadro.

7.3.3 Cuadro de objeto TTCN múltiple

A continuación se muestra la disposición general del cuadro para objeto TTCN múltiple.

Título del cuadro		
Grupo : (Medio optativo de agrupar conjuntos de objetos relacionados)		
Comentario colectivo: *Comentario válido para los objetos definidos/declarados a continuación. Este comentario tiene un ámbito que abarca hasta el siguiente Comentario colectivo o hasta el final de este cuadro.*		
Nombre del objeto	**... Otras columnas ...**	**Comentarios**
Comentario colectivo: *Comentario válido para los objetos definidos/declarados a continuación. Este comentario tiene un ámbito que abarca hasta el siguiente Comentario colectivo o hasta el final de este cuadro.*		
Nombre del objeto	**... Otras columnas ...**	**Comentarios**
Comentarios detallados:		

Figura 2/X.292 – Disposición general del cuadro para declarar un objeto múltiple

Los C*omentarios colectivos* optativos se pueden usar antes de un grupo de objetos relacionados declarados en un cuadro de objeto múltiple, tanto para indicar la agrupación como para presentar un comentario que se aplica a cada miembro del grupo o al grupo en su conjunto.

Este tipo de cuadro tiene solamente una sección de encabezamiento optativa mínima, que consta de un identificador de *Grupo* y un *Comentario colectivo*. El cuerpo del cuadro consta de una o más columnas. Cada columna tiene un título. La columna situada a la izquierda, titulada *Nombre del objeto*, contiene identificadores de los objetos definidos o declarados en el cuadro. La columna situada más a la derecha titulada *Comentarios*, contiene descripciones informales de los objetos definidos o declarados en el cuadro. No existe en todos los formularios. Cuando existe, su uso es optativo por parte del especificador de la serie de pruebas. El pie del cuadro es idéntico al del cuadro de objeto único.

7.3.4 Cuadros compactos alternativos

En algunos casos, se permite la representación de varios cuadros de objeto TTCN único en un formato compacto alternativo para ahorrar espacio. Es decir, se puede representar varios cuadros de objeto TTCN único en forma de un solo cuadro compacto. Los únicos cuadros que se presentan con este formato son:

- constricciones de ASP (tabular y ASN.1);
- constricciones de PDU (tabular y ASN.1);
- constricciones del tipo estructurado;
- constricciones del tipo ASN.1;
- comportamientos dinámicos de casos de prueba.

En el anexo E se definen los formatos de estos formularios compactos alternativos.

7.3.5 Especificación de formularios

En esta Recomendación se especifican tipos de cuadro con TTCN.GR y se proporciona una visión gráfica de los formularios correspondientes. Estos formularios se conforman a la disposición general de 7.3.2 y 7.3.3. Cuando una columna está sombreada significa que es optativa.

7.4 Texto libre y texto libre limitado

Algunas entradas de los cuadros permiten la utilización de texto libre, es decir, caracteres de cualquiera de los conjuntos de caracteres definidos en ISO/CEI 10646-1. Se aplican las siguientes restricciones:

a) El texto libre no podrá contener la combinación de caracteres "*/", a menos que estén precedidos por el carácter barra inclinada invertida (\), ya que aquellos se utilizan en la TTCN.MP para indicar el final de una cadena de texto libre. Esto significa que la doble barra inclinada invertida (\\) quiere decir barra inclinada invertida.

b) Las combinaciones de caracteres "/*" y "*/" con que empiezan y terminan las cadenas de texto libre limitado en la TTCN.MP, no aparecerán en la TTCN.GR, es decir que cuando aparezca una cadena de texto libre limitado, tanto en un campo de un cuadro como en un identificador completo, no se imprimirán estas combinaciones de caracteres.

8 Concurrencia en la TTCN

8.1 Componentes de prueba

La TTCN permite especificar componentes de prueba que se pueden ejecutar de manera concurrente. Esta cláusula proporciona una visión general de los mecanismos y formularios adicionales disponibles en la TTCN concurrente. Estos formularios y mecanismos no se utilizarán en las ATS que no apliquen la concurrencia (es decir, el uso de la concurrencia es optativo).

Un probador consta de un componente de prueba principal (MTC, *main test component*) y cero o más componentes de prueba paralelos (PTC, *parallel test components*). En la TTCN no concurrente no es preciso declarar el componente de prueba principal, ya que solo existe un componente de prueba, y el valor por defecto es el componente de prueba principal.

Los componentes de prueba se declaran en el cuadro de declaraciones de componentes de prueba. Un componente de prueba puede comunicar con la implementación sometida a prueba (IUT) mediante uno o más puntos de control y observación (PCO, *points of control and observation*). Los componentes de prueba pueden comunicar con cada uno de los demás mediante el intercambio de mensajes de coordinación (CM, *co-ordination message*) a través de puntos de coordinación (CP, *co-ordination point*). Los componentes de prueba paralelos pueden también comunicar con el componente de prueba principal implícitamente, por medio de asignaciones a la variable de resultado global y encontrándose el componente de prueba principal en situación de disponible para supervisar si uno o más componentes de prueba paralelos han finalizado la ejecución. Los cuadros de declaraciones de configuración de componentes de prueba se utilizan para especificar configuraciones abstractas de componentes de prueba. Estas declaraciones (una para cada configuración) muestran los puntos de control y observación y los puntos de coordinación utilizados, si existen, por los componentes de prueba. Los mensajes de coordinación (CM) se especifican por un método similar al utilizado en la especificación de las primitivas de servicio abstractas (ASP). Para la especificación de los mensajes de coordinación se puede utilizar ASN.1. Las constricciones de CM son también muy similares a las constricciones de ASP. Para la definición de tipos de CM y la declaración de constricciones de CM se proporcionan formularios especiales. Para el envío y recepción de los mensajes de coordinación se utilizan los enunciados de TTCN SEND y RECEIVE normales.

Resumiendo, cuando se usa la TTCN concurrente se emplearán los siguientes formularios:

a) declaraciones de componentes de prueba;

b) declaraciones de configuración de componente de prueba.

Además, cuando se usa la TTCN concurrente se pueden emplear los formularios siguientes:

c) declaraciones de CP;

d) definiciones de tipo de CM y/o definiciones de tipo de CM en ASN.1, en el supuesto de que se utilicen declaraciones de CP;

e) declaraciones de constricciones de CM, cuando se utilicen definiciones de tipo de CM;

f) declaraciones de constricciones de CM en ASN.1, siempre que se utilicen definiciones de tipo de CM en ASN.1.

8.2 Configuraciones de componentes de prueba

En las figuras 3 y 4 se muestran algunas configuraciones posibles de componentes de prueba. En la realización de estas configuraciones abstractas, los componentes de prueba pueden residir en una sola máquina o estar distribuidos en varias máquinas.

Se pueden utilizar diferentes configuraciones de PTC en diferentes casos de prueba de una serie de pruebas abstractas. Cada caso de prueba abstracta que utiliza concurrencia empleará una de las configuraciones de componentes de prueba declaradas.

Se señalan los siguientes casos válidos, aunque inusuales:

a) un PTC no necesita tener ningún PCO;

b) un PTC no necesita tener un CP a un MTC. En tales casos la única interacción entre el PTC y el MTC será la creación del PTC y los informes de resultados implícitos procedentes del PTC, es decir, el MTC no tiene un control explícito sobre el PTC después de su creación;

c) se pueden conectar dos PTC por más de un CP;

d) un caso de prueba cuya configuración de componentes de prueba haga referencia a un PTC no necesita contener ningún enunciado CREATE para arrancar este PTC;

e) un caso de prueba cuya configuración de componentes de prueba haga referencia a un CP no necesita contener ningún enunciado SEND o RECEIVE con este CP.

Los ítems a), b) y c) se ilustran en las figuras 3 y 4.

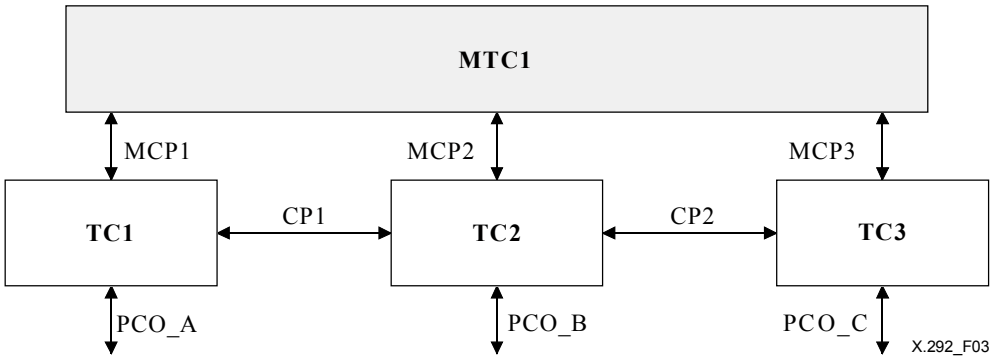


Figura 3/X.292 – Configuración de componentes de prueba CONFIG1

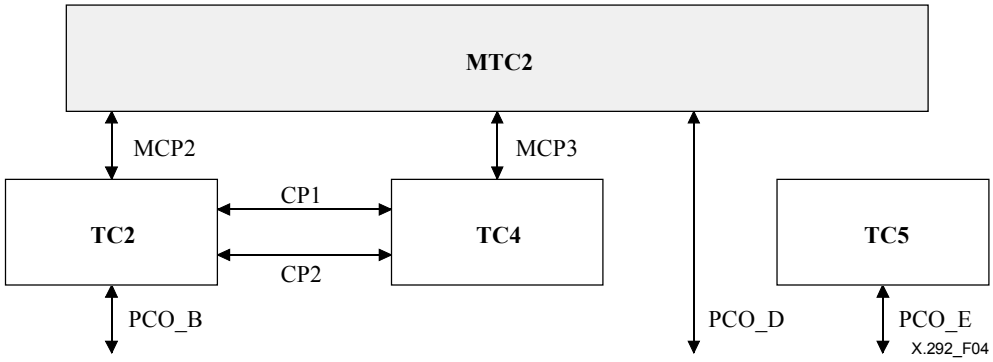


Figura 4/X.292 – Configuración de componentes de prueba CONFIG2

9 Estructura de la serie de pruebas con TTCN

9.1 Introducción

La TTCN permite estructurar jerárquicamente una serie de pruebas, de conformidad con 8.1/X.290. Los componentes de esta estructura son:

a) grupos de prueba;

b) casos de prueba;

c) pasos de prueba.

Una serie de pruebas con TTCN puede ser completamente plana (es decir, que carece de estructura), en cuyo caso no hay grupos de prueba.

La TTCN permite el empleo de grupos de paso de prueba y grupos por defecto similares al concepto de grupos de prueba, a fin de estructurar jerárquicamente los pasos de prueba y los valores por defecto. Esta estructura jerárquica es optativa.

9.2 Referencias de grupos de prueba

La TTCN soporta una estructura de denominación que muestra una agrupación conceptual de los casos de prueba. Los grupos de prueba pueden estar anidados. Los casos de prueba también pueden ser autónomos (véase la figura 9 de la cláusula 8/X.290). Las referencias de grupo de prueba definen la estructura de la serie de pruebas.

EJEMPLO 3 – Referencia a grupo de transporte: TRANSPORT/CLASS0/CONN_ESTAB/

9.3 Referencias de grupo de pasos de prueba

En la TTCN, los pasos de prueba se pueden identificar explícitamente y utilizar para estructurar casos de prueba y otros pasos de prueba. Otra posibilidad es que los pasos de prueba estén implícitos en la descripción de comportamiento de un caso de prueba. Los pasos de prueba explícitos se pueden especificar:

- localmente dentro de una descripción de comportamiento de caso de prueba o de paso de prueba; o
- globalmente dentro de una biblioteca de pasos de prueba, que puede estar jerárquicamente estructurada en grupos de pasos de prueba.

NOTA – Por ejemplo, un prólogo puede consistir en sólo unas cuantas líneas de enunciado en una descripción de comportamiento del caso de prueba, en cuyo caso es implícito. Alternativamente, un prólogo puede estar especificado explícitamente con su propia descripción de comportamiento. Si el prólogo explícito sólo se utiliza dentro de un caso de prueba, puede estar especificado localmente dentro de ese caso de prueba; en cambio, si se utiliza en varios casos de prueba, se tiene que especificar en la biblioteca de pasos de prueba.

Los pasos de prueba locales se identifican simplemente por un identificador de árbol. Los pasos de prueba globales se identifican por un identificador de paso de prueba. Los pasos de prueba globales tienen, además, una referencia a grupo de pasos de prueba, que muestra la posición de un paso de prueba en la biblioteca de pasos de prueba. La estructura de la biblioteca de pasos de prueba es independiente de la estructura de la serie de pruebas.

EJEMPLO 4 – Referencia a grupo de pasos de prueba de transporte: TRANSPORT/STEP_LIBRARY/ CLASS0/CONN_ESTAB/

9.4 Referencias de grupo de valores por defecto

Los comportamientos por defecto (si existen) están situados en una biblioteca de valores por defecto.

Una referencia a grupo de valores por defecto especifica la situación del valor por defecto en la biblioteca de valores por defecto, que puede estar estructurada jerárquicamente. La biblioteca de valores por defecto no tiene ninguna influencia sobre la propia estructura de la serie de pruebas.

EJEMPLO 5 – Referencia al grupo de valores por defecto de transporte: TRANSPORT/DEFAULT_LIBRAR/ CLASS0/

9.5 Partes de una serie de pruebas TTCN

Una ATS escrita en TTCN tendrá las cuatro secciones siguientes, en el orden indicado:

a) Visión general de la serie (véase la cláusula 10),

   que contiene la información necesaria para la presentación general y la comprensión de la serie de pruebas, tal como las referencias de pruebas y una descripción de su finalidad general.

b) Parte importación (véase 10.8),

   que contiene las declaraciones de los objetos utilizados en el módulo o la serie de pruebas que se importan de un objeto fuente.

c) Parte declaraciones (véase la cláusula 11),

   que contiene las definiciones o declaraciones de todos los componentes que constituyen la serie de pruebas (por ejemplo PCO, temporizadores, ASP, PDU, así como sus parámetros o campos).

d) Parte constricciones (véanse las cláusulas 12, 13 y 14),

que contiene las declaraciones de valores de las ASP, PDU y sus parámetros utilizados en la parte dinámica. Las constricciones se especificarán mediante:

1) cuadros de TTCN; o
2) la notación de valor ASN.1; o
3) los cuadros de TTCN y la notación de valor ASN.1.

e) Parte dinámica (véase la cláusula 15),

que consta de tres secciones, que contienen cuadros que especifican el comportamiento de la prueba, expresado sobre todo en términos de la aparición de ASP, o PDU en los PCO. Estas secciones son:

1) las descripciones de comportamiento dinámico del caso de prueba; o
2) una biblioteca que contiene las descripciones de comportamiento dinámico del paso de prueba (si existen);
3) una biblioteca que contiene las descripciones de comportamiento dinámico de los valores por defecto (si existen).

10 Visión general de la serie de pruebas

10.1 Introducción

La finalidad de la parte "Visión general de la serie de pruebas" de la ATS es proporcionar la información necesaria para la presentación general y la comprensión de la serie de pruebas. Dicha información comprende:

a) índice de series de pruebas (véase 10.2);
b) estructura de la serie de pruebas (véase 10.3);
c) índice de casos de prueba (véase 10.4);
d) índice de pasos de prueba (véase 10.5);
e) índice de valores por defecto (véase 10.6);
f) exportaciones de la serie de pruebas (véase 10.7).

10.2 Índice de series de pruebas

La finalidad del Índice de series de pruebas es dar la información necesaria para todos los objetos importados de una serie de pruebas ampliada. Esta información sirve para encontrar fácilmente la definición de un objeto.

El Índice de series de pruebas es una lista completa de todos los objetos de una serie de pruebas ampliada, y se obtiene mediante la conversión de una serie de pruebas modularizada en una serie de pruebas ampliada. La lista contiene información sobre cada objeto (el objeto fuente o el nombre de la serie de pruebas, el nombre original y el número de página del objeto fuente verdaderamente original).

El formulario Índice de series de pruebas identifica todos los objetos que se usan en una serie de pruebas. Para cada objeto se proporcionará la información siguiente:

a) Nombre del objeto

nombre que sirve para referirse al objeto (es decir un nombre generado).

b) Tipo de objeto

que será igual al tipo dado cuando se define el objeto.

c) Nombre del objeto fuente o de la serie de pruebas

donde se define el objeto.

d) Nombre original del objeto

nombre dado cuando se define explícitamente el objeto.

e) Número de página optativo

que proporciona la situación del objeto en el objeto fuente original.

Esta información se dará con el formato del formulario siguiente:

Índice de series de pruebas					
Nombre del objeto	**Tipo de objeto**	**Nombre fuente**	**Ref. objeto original**	**N.º de página**	**Comentarios**
⋮ *ObjectIdentifier* ⋮	⋮ *ObjectType* ⋮	⋮ *SourceIdentifier* ⋮	⋮ *[ObjectReference]* ⋮	⋮ *[Number]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*					

Formulario 0: Índice de series de pruebas

El número de página se da cuando el objeto fuente original estándar y la situación del objeto es inequívoca.

10.3 Estructura de la serie de pruebas

La estructura de la serie de pruebas contiene la identificación de los documentos de referencia pertinentes, la especificación de la estructura de la serie de pruebas, una breve descripción de su finalidad general y las referencias a los criterios de selección del grupo de prueba.

Esta sección incluirá, al menos, la siguiente información:

a) Nombre de la serie de pruebas.

b) Referencias a las normas de base pertinentes.

c) Referencia al formulario de PICS.

d) Referencia al formulario de PIXIT parcial (véase 14.1/X.291 y apéndice V/X.296).

e) Una indicación del método o métodos de prueba al que se aplica la serie de pruebas y además, en el caso de métodos de prueba coordinados, una referencia al lugar en que está especificado el TMP.

f) Cualquier otra información que pueda ayudar a la comprensión de la serie de pruebas, por ejemplo su número de versión o cómo se ha derivado. Esta información se proporcionará como comentario.

g) Una lista de los grupos de prueba de la serie de pruebas (si existen),

en cuyo caso se facilitará para cada grupo la siguiente información:

1) Referencia al grupo de prueba,

   en la que el primer identificador es el nombre de la serie y cada identificador sucesivo representa un orden conceptual ulterior de la serie de pruebas. Los grupos de prueba se numerarán en el orden en que sus casos de prueba correspondientes aparecen en la ATS. Además, se ordenarán de forma que cada grupo de un solo grupo siga inmediatamente a ese grupo. Se hará una lista con todos los grupos de prueba de la serie de pruebas.

   Los casos de prueba importados se pueden incluir en cualquier grupo, con independencia del grupo en el que están definidos en el objeto fuente original. Se puede hacer una lista con un nuevo grupo que no ocurra en la parte dinámica. Este grupo sólo contendrá los casos de prueba importados.

   Los grupos de la parte dinámica ocurrirán en el mismo orden en que aparecen, pero la lista puede estar precedida, interrumpida o seguida por nuevos grupos de casos de prueba importados. Para estos nuevos grupos no se suministrará el número de página.

   La columna "Ref. de selección" puede contener el identificador de una expresión de selección aplicable a los nuevos grupos de prueba. La nueva expresión de selección reemplazará a la expresión de selección especificada en el grupo de prueba original (si existe). La ausencia en esta columna del identificador de expresión de selección indica que se omite la expresión de selección especificada en el grupo de prueba original (si existe).

   La columna "Objetivo de grupo de prueba" puede contener un nuevo enunciado informal del objetivo del nuevo grupo de prueba. Este nuevo objetivo reemplazará al objetivo en el grupo de prueba importado (si existe). La ausencia en esta columna del objetivo de grupo de prueba indica que se omite el objetivo de grupo de prueba especificado.

2) Un identificador de expresión de selección optativo,

que hace referencia a una entrada en el cuadro de definiciones de expresión de selección de casos de prueba utilizado para determinar si se aplican los casos de prueba del grupo a una IUT específica. Esta columna puede contener el identificador de una expresión de selección aplicable al grupo de prueba. Si, para un grupo se proporciona un identificador de expresión de selección, y la expresión de selección referenciada toma el valor FALSO (FALSE), no se seleccionará para su ejecución ningún caso de prueba de ese grupo. Si la expresión toma el valor VERDADERO (TRUE), se seleccionarán los casos de prueba de ese grupo para su ejecución, según la evaluación de las expresiones de selección pertinentes de los subgrupos de ese grupo y/o casos de prueba individuales. La omisión de un identificador de expresión de selección es equivalente al valor booleano TRUE.

3) Objetivo del grupo de prueba,

que es una declaración informal del objetivo del grupo de prueba.

4) Número de página,

que proporciona la situación del primer caso de prueba del grupo en la ATS. El número de página inscrito en lista con cada referencia a grupo de prueba en el cuadro de estructura de la serie de pruebas será el número de página de la primera descripción de comportamiento de caso de prueba de ese grupo.

Esta información se proporcionará con el formato del formulario siguiente:

Estructura de la serie de pruebas			
Nombre de la serie : *SuiteIdentifier*			
Ref. de norma : *Free Text*			
Ref. de PICS : *Free Text*			
Ref. de PIXIT : *Free Text*			
Método(s) de prueba : *FreeText*			
Comentarios : *[FreeText]*			
Referencia a grupo de prueba	**Ref. de selección**	**Objetivo del grupo de prueba**	**N.º de página**
⋮ *TestGroupReference* ⋮	⋮ *[SelectExprIdentifier]* ⋮	⋮ *FreeText* ⋮	⋮ *Number* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 1: Estructura de la serie de pruebas

10.4 Índice de casos de prueba

El Índice de casos de prueba contiene una lista completa de todos los casos de prueba de la ATS. Para cada caso de prueba se facilitará la siguiente información:

a) Una referencia a un grupo de prueba optativa (si la ATS está estructurada en grupos de prueba),

que define el lugar que ocupa el caso de prueba en la estructura de grupo de la serie de pruebas.

b) Nombre del caso de prueba,

que es el identificador proporcionado en el cuadro de comportamiento dinámico del caso de prueba. Los casos de prueba se pondrán en lista en el orden en el que figuran en la ATS.

c) Un identificador de expresión de selección optativo,

que hace referencia a una entrada en el cuadro de definiciones de expresión de selección de casos de prueba utilizado para determinar si se puede seleccionar el caso de prueba para su ejecución. Esta columna puede contener el identificador de una expresión de selección aplicable al caso de prueba. Si se proporciona el identificador de expresión de selección, y la expresión de selección referenciada toma el valor FALSE, no se seleccionará el caso de prueba para su ejecución. Si la expresión de selección toma el valor TRUE, se seleccionará para su ejecución el caso de prueba, según la evaluación de la expresión de selección para los grupos de prueba contenidos en el caso de prueba. Se selecciona un caso de prueba si la expresión de selección para el caso de prueba y todos los grupos contenidos en el caso de prueba toman el

valor TRUE. La omisión de un identificador de expresión de selección es equivalente al valor booleano TRUE.

d) Una descripción del caso de prueba,

que posiblemente es una forma abreviada de la finalidad de la prueba.

e) Un número de página,

que proporciona la situación del caso de prueba en la ATS. El número de página indicado en lista con cada identificador de caso de prueba en el cuadro de índice de caso de prueba será el número de página de la descripción de comportamiento del caso de prueba correspondiente.

Esta información se proporcionará con el formato del formulario siguiente.

Índice de casos de prueba				
Referencia a grupo de pruebas	**Id. de caso de prueba**	**Ref. de selección**	**Descripción**	**N.º de página**
⋮ *TestGroupReference* ⋮	⋮ *TestCaseIdentifier* ⋮	⋮ *[SelectExprIdentifier]* ⋮	⋮ *FreeText* ⋮	⋮ *Number* ⋮
Comentarios detallados: *[FreeText]*				

Formulario 2: Índice de casos de prueba

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

La lista completa de casos de prueba incluirá los casos de prueba importados. Los casos de prueba definidos explícitamente se relacionarán en el orden en que se encuentran en la ATS. Para los casos de prueba importados no se proporcionarán los números de página.

La columna "Ref. de selección" tiene una semántica similar a la dada en 10.3.

La columna "Descripción" puede contener una forma abreviada nueva de la finalidad de la prueba. Esta nueva descripción reemplazará a la descripción en el caso de prueba importado (si existe). La ausencia de la descripción indica que se omite la descripción especificada.

10.5 Índice de pasos de prueba

El Índice de pasos de prueba contiene una lista completa de todos los pasos de prueba de la ATS. Para cada paso de prueba se facilitará la siguiente información:

a) Una referencia optativa al grupo de pasos de prueba, (si la ATS está estructurada en grupos de pasos de prueba),

que define en qué lugar de la estructura de la biblioteca de pasos de prueba reside ese paso de prueba. Si falta la referencia al grupo para un paso de prueba, se supone que dicho paso de prueba reside en el mismo grupo que el paso de prueba anterior en el índice. Los grupos de pasos de prueba se pondrán en lista en el mismo orden en el que figuran en la ATS. Para el primer paso de prueba de cada grupo se proporcionará una referencia explícita al grupo de paso de prueba. Se proporcionará, asimismo, una referencia explícita al grupo de paso de prueba para cada paso de prueba que siga inmediatamente al último paso de prueba del grupo; esto es necesario si un grupo de pasos de prueba contiene grupos de pasos de prueba y pasos de prueba.

b) Nombre del paso de prueba,

que será el identificador proporcionado en el cuadro de comportamiento dinámico del paso de prueba. Los pasos de prueba se pondrán en lista en el mismo orden en el que figuran en la ATS.

c) Una descripción del paso de prueba,

que posiblemente es una forma abreviada del objetivo del paso de prueba.

d) Un número de página,

que proporciona la situación del paso de prueba en la ATS. El número de página indicado en lista con cada identificador de paso de prueba en el cuadro de índices de paso de prueba, será el número de página de la descripción de comportamiento del paso de prueba correspondiente.

Esta información se proporcionará con el formato del formulario siguiente.

Índice de pasos de prueba			
Referencia a grupo de paso de prueba	**Id. paso de prueba**	**Descripción**	**N.º de página**
⋮ *TestStepGroupReference* ⋮	⋮ *TestStepIdentifier* ⋮	⋮ *FreeText* ⋮	⋮ *Number* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 3: Índice de pasos de prueba

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

La lista completa de pasos de prueba incluirá los pasos de prueba importados. Los pasos de prueba definidos explícitamente aparecerán en la lista con el orden que tienen en la ATS. Para los pasos de prueba importados no se proporcionará el número de página.

La columna "Descripción" puede contener una forma abreviada nueva del objetivo de paso de prueba. Esta nueva descripción reemplazará a la descripción en el paso de prueba importado (si existe). La ausencia de la descripción indica que se omite la descripción especificada.

10.6 Índice de valores por defecto

El Índice de valores por defecto contiene una lista completa de todos los valores por defecto de la ATS. Para cada valor por defecto, se proporcionará la información siguiente:

a) Una referencia a grupo optativa (si la ATS está estructurada en grupos por defecto),

que define en qué lugar de la estructura de la biblioteca de valores por defecto reside ese valor por defecto. Si falta la referencia a grupo de valor por defecto, se supone que dicho valor por defecto reside en el mismo grupo que el valor por defecto anterior en el índice. Los valores por defecto se pondrán en lista en el mismo orden en el que figuran en la ATS. Para el primer valor por defecto de cada grupo se proporcionará una referencia explícita al grupo de valores por defecto. Igualmente, se proporcionará una referencia explícita al grupo de valores por defecto para cada valor por defecto que siga inmediatamente al último valor por defecto del grupo.

b) Nombre del valor por defecto,

que es el identificador proporcionado en el cuadro de comportamiento dinámico del valor por defecto. Los valores por defecto se pondrán en lista en el mismo orden en el que figuran en la ATS.

c) Una descripción del valor por defecto,

que posiblemente es una forma abreviada del objetivo por defecto.

d) Un número de página,

que proporciona la situación del valor por defecto en la ATS. El número de página indicado en lista con cada identificador del valor por defecto en el cuadro de índice de valores por defecto será el número de página de la descripción de comportamiento del valor por defecto correspondiente.

La información se proporcionará con el formato del formulario siguiente.

Índice de valores por defecto			
Referencia a grupo de valores por defecto	**Id. del valor por defecto**	**Descripción**	**N.º de página**
⋮ *DefaultGroupReference* ⋮	⋮ *DefaultIdentifier* ⋮	⋮ *FreeText* ⋮	⋮ *Number* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 4: Índice de valores por defecto

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

La lista completa de valores por defecto incluirá los valores por defecto importados. Los valores por defecto definidos explícitamente se relacionarán en el orden en que se encuentran en la ATS. Para los valores por defecto importados no se proporcionará el número de página.

La columna "Descripción" puede contener una forma abreviada nueva del objetivo de la prueba. Esta nueva descripción reemplazará a la descripción del valor por defecto importado (si existe). La ausencia de una descripción indica que se omite la descripción especificada.

10.7 Exportaciones de serie de pruebas

El cuadro Exportaciones de serie de pruebas se puede utilizar para especificar explícitamente qué objetos de la serie de pruebas se pueden reutilizar y, por ello, se pueden importar en otras series de pruebas o módulos TTCN.

El formulario de exportaciones de serie de pruebas se utiliza para identificar los objetos que se pueden exportar.

Si el objeto mismo es importado, se dará el nombre del objeto fuente original.

Si el objeto se declara como un objeto externo (externo explícito), o si se trata de un objeto que se omite en el objeto fuente importado (externo implícito), se da la palabra clave EXTERNAL en lugar del nombre del objeto fuente.

La exportación de un objeto del tipo Enumeración (Enumeration) o Número Denominado (Named Number) exige que se dé el tipo correspondiente. Los demás objetos que se definen en el tipo correspondiente tampoco se exportan. Sin embargo, se los exporta implícitamente y se puede hacer referencia a ellos en otros objetos exportados. El nombre del tipo se añade como sufijo al nombre del objeto encerrado entre corchetes.

En el cuadro Exportaciones de serie de pruebas se suministrará la siguiente información para cada uno de los objetos exportados:

a) Nombre del objeto,

   si el objeto es del tipo NamedNumber o Enumeration, se da el tipo correspondiente como sufijo del nombre del objeto encerrado entre corchetes.

b) Tipo de objeto.

c) Nombre del objeto fuente original si el objeto es importado, o la instrucción de objeto EXTERNAL.

d) Número de página,

   que proporciona la situación del objeto en la serie de pruebas (los objetos importados no llevan número de página).

e) Comentario optativo.

Esta información se proporcionará con el formato del formulario siguiente.

Exportaciones de la serie de pruebas				
Nombre del objeto	**Tipo del objeto**	**Nombre de la fuente**	**N.º de página**	**Comentarios**
⋮ *ObjectIdentifier* ⋮	⋮ *TTCN_ObjectType* ⋮	⋮ *[SourceIdentifier | ObjectDirective]* ⋮	⋮ *Number* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*				

Formulario 5: Exportaciones de la serie de pruebas

EJEMPLO 6 – Exportaciones de la serie de pruebas:

Exportaciones de la serie de pruebas				
Nombre del objeto	**Tipo del objeto**	**Nombre de la fuente**	**N.º de página**	**Comentarios**
String5	SimpleType_Object		3	
wait	Timer_Object	Module_B		
INTC	TTCN_PDU_Type_Object		13	
DEF1	Default_Object	TestSuite_1		
TC_2	TestCase_Object	TestSuite_2		
TC_3	TestCase_Object		33	
Preamble	TestStep_Object	EXTERNAL		
Comentarios detallados:				

10.8 La parte Importación

10.8.1 Introducción

La finalidad de la parte Importación es declarar los objetos utilizados en la serie de pruebas que se importan de un objeto fuente. El efecto que producen las importaciones equivale a disponer de una copia de los objetos importados dentro de la serie de pruebas.

Un objeto sólo se puede importar si lo exporta un objeto fuente. Una serie de pruebas sin un cuadro Exportaciones exporta todos los objetos que tienen un nombre global. Un módulo y una serie de pruebas con al menos un cuadro Exportaciones, exportan los objetos contenidos en los cuadros Exportaciones. Un objeto que no es importado explícitamente es importado implícitamente cuando está referenciado por un objeto importado.

10.8.2 Importaciones

El cuadro Importaciones identifica el objeto fuente y proporciona información acerca del objetivo global del objeto fuente. En el cuadro Importaciones se suministrará la siguiente información:

a) nombre del objeto fuente;

b) descripción del objetivo del objeto fuente;

c) referencia completa al objeto fuente, con un identificador de documento y otra información, como la versión y la fecha;

d) referencia a las normas a las que se aplica la fuente;

e) otra información, incluida como comentario, que pueda ayudar a comprender el objeto fuente;

f) lista de los objetos procedentes del objeto fuente importado; para cada objeto se proporcionará la siguiente información:

   1) nombre del objeto utilizado en el objeto fuente;

   2) tipo del objeto, que es igual al tipo que se da en el objeto fuente;

   3) nombre del objeto fuente original cuando el objeto se importa de otro objeto fuente, la instrucción de objeto OMIT o "-" cuando hay que omitir el objeto del conjunto de objetos importados del objeto fuente, o la instrucción de objeto EXTERNAL cuando se declara el objeto como externo en el objeto fuente.

Esta información se proporcionará con el formato del formulario siguiente.

Importaciones			
Nombre de la fuente : *SuiteIdentifier*			
Grupo : *[ImportsGroupReference]*			
Ref. de fuente : *[FreeText]*			
Ref. de normas : *[FreeText]*			
Comentarios : *[FreeText]*			
Nombre del objeto	**Tipo del objeto**	**Nombre de la fuente**	**Comentarios**
⋮ *ObjectIdentifier* ⋮	⋮ *TTCN_ObjectType* ⋮	⋮ *[SourceIdentifier | ObjectDirective]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 6: Importaciones

EJEMPLO 7 – Cuadro de importaciones:

Importaciones			
Nombre de la fuente : Module A			
Ref. de fuente : {ISO standard 1234}			
Ref. de norma : ISO 300 313			
Comentarios : Layer 2 Test Suite			
Nombre del objeto	**Tipo del objeto**	**Nombre de la fuente**	**Comentarios**
String5	SimpleType_Object		
Wait	Timer_Object	Module B	1)
R1_POSTAMBLE	TestStep_Object	EXTERNAL	2)
TSAP	PCO_Type_Object		3)
blue[ColorEnum]	Enumeration_Object	OMIT	
a[NN_type1]	NamedNumber_Object		4)

Comentarios detallados:

1) La fuente original de este temporizador es Module B.
2) Este paso de prueba se declara como externo en Module A y se definirá explícitamente o se importará cuando se utiliza este módulo.
3) TSAP se definirá en el cuadro de declaraciones de tipo de PCO.
4) Este Named Number se omite de las importaciones y por ello hay que redefinirlo explícitamente en la serie de pruebas.

11 La parte Declaraciones

11.1 Introducción

La finalidad de la parte "Declaraciones" de la ATS es definir y declarar todos los objetos que se usan en la serie de pruebas. En la parte Declaraciones se declararán los objetos de una ATS referenciados de la parte general, la parte constricciones y la parte dinámica que se indican a continuación. Los objetos son:

a) definiciones:
   1) tipos de series de pruebas (véase 11.2.3);
   2) operaciones series de pruebas (véase 11.3.4).
b) parametrización y selección de casos de prueba:
   1) parámetros de series de pruebas (véase 11.4);
   2) expresiones de selección de caso de prueba (véase 11.5).

c) declaraciones/definiciones:

1) constantes de serie de pruebas (véanse 11.6 y 11.7);
2) variables de serie de pruebas (véase 11.8.1);
3) variables de casos de prueba (véase 11.8.3);
4) tipos de PCO (véase 11.9);
5) PCO (véase 11.10);
6) CP (véase 11.11);
7) temporizadores (véase 11.12);
8) componentes de prueba (véase 11.13.1);
9) configuraciones de componentes de prueba (véase 11.13.2);
10) tipos de ASP (véase 11.14);
11) tipos de PDU (véase 11.15);
12) reglas de codificación (véase 11.16.1);
13) variaciones de codificación (véase 11.16.2);
14) codificaciones de campo no válidas (véase 11.16.3);
15) tipos de CM (véase 11.17);
16) alias (véase 11.21).

11.2 Tipos de TTCN

11.2.1 Introducción

La TTCN soporta cierto número de tipos y mecanismos predefinidos, que permiten definir los tipos de series de pruebas específicas. Estos tipos se pueden utilizar en toda la serie de pruebas y referenciarlos cuando se declaran parámetros de serie de pruebas, constantes de serie de pruebas, variables de serie de pruebas, parámetros de ASP, campos de PDU, etc.

La TTCN es un lenguaje con pocos tipos, porque dos tipos cualesquiera que tienen el mismo tipo de base se consideran compatibles (por ejemplo, cuando se hacen asignaciones o se transfieren parámetros).

11.2.2 Tipos TTCN predefinidos

Se han predefinido diversos tipos utilizados habitualmente para usarlos en la TTCN. Todos los tipos definidos en ASN.1 y en esta cláusula se pueden referenciar, aunque no aparezcan en una definición de tipo en una serie de pruebas. Los demás tipos utilizados en la serie de pruebas se declararán en las definiciones de tipo de serie de pruebas, de ASP o de PDU y se referenciarán por nombre.

Los siguientes tipos TTCN predefinidos se consideran iguales a sus homólogos en ASN.1:

a) **Tipo predefinido INTEGER** (entero): Tipo con valores distinguidos, que son los números enteros, positivos y negativos, incluido el cero.

   Los valores de tipo INTEGER se designarán mediante uno o más dígitos. El primer dígito no será cero, a menos que el valor sea 0; el valor cero se representará por un solo cero.

b) **Tipo predefinido BOOLEAN** (booleano): Tipo que consta de dos valores distinguidos.

   Los valores del tipo BOOLEAN son TRUE y FALSE (verdadero y falso).

c) **Tipo predefinido BITSTRING** (cadena de bits): Tipo cuyos valores distinguidos son secuencias ordenadas de cero, uno, o más bits.

   Los valores del tipo BITSTRING se designarán mediante un número arbitrario (quizás ninguno) de ceros y unos, precedidos por comillas sencillas (') y seguidos del par de caracteres ('B):

   EJEMPLO 8 – '01101'B

d) **Tipo predefinido HEXSTRING** (cadena hexadecimal): Tipo cuyos valores distinguidos son secuencias ordenadas de cero, uno o más dígitos hexadecimales, cada uno de los cuales corresponde a una secuencia ordenada de cuatro bits.

   Los valores de tipo HEXSTRING se designarán mediante un número arbitrario (quizás ninguno) de dígitos hexadecimales:

   0 1 2 3 4 5 6 7 8 9 A B C D E F

precedidos por comillas sencillas (') y seguidos del par de caracteres ('H). Se utiliza cada dígito HEX, para representar el valor de un semiocteto con la notación hexadecimal:

EJEMPLO 9 – 'AB01D'H

e) **Tipo predefinido OCTETSTRING** (cadena de octetos): Tipo cuyos valores distinguidos son una secuencia ordenada de cero o un número par positivo de dígitos HEX (cada par de dígitos corresponde a una secuencia ordenada de 8 bits).

Los valores de tipo OCTETSTRING se indicarán mediante un número par arbitrario (quizás ninguno) de dígitos HEX:

0 1 2 3 4 5 6 7 8 9 A B C D E F

precedidos por comillas sencillas (') y seguidos del par de caracteres ('O). Se utiliza cada dígito HEX para designar el valor de un semiocteto con la notación hexadecimal:

EJEMPLO 10 – 'FF96'O

f) **Tipo predefinido OBJECTIDENTIFIER** (identificador de objeto): Tipo cuyos valores distinguidos son el conjunto de todos los identificadores de objeto asignados de conformidad con las reglas de la Rec. UIT-T X.680.

g) **Tipo predefinido R_TYPE** (tipo R): Tipo constituido por los valores distinguidos siguientes:

pass (éxito), fail (fracaso), inconc (no concluyente) y none (ninguno)

Estos valores son identificadores predefinidos y por ello dependen del caso. Este tipo predefinido se utiliza con los veredictos; véase 15.17.

h) **Tipos predefinidos CharacterString** (cadena de caracteres): Tipos cuyos valores distinguidos son cero, uno, o más caracteres tomados de un conjunto de caracteres; se pueden utilizar los tipos CharacterString del cuadro 2, y se definen en la cláusula 31/X.680.

Cuadro 2/X.292 – Tipo predefinido CharacterString

NumericString
PrintableString
TeletexString
T61String
VideotexString
VisibleString
ISO646String
IA5String
GraphicString
GeneralString
BMPString
UniversalString

Los valores de los tipos CharacterString se designarán mediante un número arbitrario (quizás ninguno) de caracteres de un conjunto de caracteres referenciado por el tipo CharacterString, precedidos y seguidos por comillas ("). Si el tipo CharacterString incluye comillas, la denotación de cualquier valor se encerrará entre comillas ("").

11.2.3 Definiciones de tipos de series de pruebas

11.2.3.1 Introducción

Las definiciones de tipos que se utilizarán como tipos para objetos de datos y subtipos de ASP, PDU estructuradas, etc., se pueden introducir en formato tabular y/o ASN.1. Cuando los tipos se referencian en las definiciones de tipos de series de pruebas, dichas referencias no serán recursivas (ni directa ni indirectamente).

11.2.3.2 Definiciones de tipo simple mediante cuadros

Para definir un nuevo tipo simple se facilitará la siguiente información:

a) Nombre del tipo.

b) Tipo de base,

donde el tipo de base será un tipo predefinido o un tipo simple. El tipo de base va seguido de la restricción de tipo, que tendrá una de las formas siguientes:

1) Lista de valores distinguidos del tipo de base; estos valores comprenden el nuevo tipo.

2) Especificación de una gama de valores del tipo INTEGER. El nuevo tipo comprende todos los valores especificados en la gama, incluidos el límite inferior y el límite superior. Para especificar una gama infinita se puede utilizar la palabra clave INFINITY (infinito) en lugar de un valor, con lo que se indica que no hay límite superior ni límite inferior.

3) Especificación de una longitud concreta o una gama de longitudes de un tipo predefinido o un tipo cadena de series de pruebas. El valor o valores de la longitud se interpretarán de conformidad con el cuadro 5 de 11.18. Para el límite superior solo se utilizarán literales INTEGER no negativos o la palabra clave INFINITY.

c) Optativamente, un identificador de codificación específico seguido de una lista de parámetros reales necesarios, a fin de especificar una codificación explícita para el tipo simple, que sustituye las reglas de codificación y las variaciones de codificación aplicables a cualquier PDU en la que se utilice este tipo simple; el identificador de codificación, si existe, identificará una de las variaciones de codificación o una definición de codificación de campo no válida definida en la serie de pruebas (por ejemplo, LD(10)); véase 11.16.4.

Esta información se proporcionará con el formato del formulario siguiente.

Definiciones de tipo simple			
Grupo : *[SimpleTypeGroupReference]*			
Nombre del tipo	**Definición del tipo**	**Codificación del tipo**	**Comentarios**
⋮ *SimpleTypeIdentifier* ⋮	⋮ *Type&Restriction* ⋮	⋮ *[PDU_FieldEncodingCall]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 7: Definiciones de tipo simple

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

Cuando se utiliza una gama en una definición de tipo, ya sea una gama de valores o una gama de longitudes (para cadenas), esto se indicará colocando el menor de los dos valores a la izquierda. Una gama de enteros sólo se utilizará con un tipo de base INTEGER o un tipo derivado de INTEGER. En este último caso, la gama de enteros será una subgama del conjunto de valores definidos por el tipo de base.

Cuando se utiliza una lista de valores, los valores serán del tipo de base y constituirán un subconjunto verdadero de los valores definidos por el tipo de base. Cuando se emplea una restricción de longitud, el conjunto de valores para el tipo definido por esta restricción será un subconjunto verdadero de los valores definidos por el tipo de base.

Los valores de dos tipos simples cualesquiera que tienen el mismo tipo de base se consideran compatibles en cuanto al tipo (por ejemplo, para hacer asignaciones o transferir parámetros).

EJEMPLO 11 – Definiciones de tipo simple para las serie de pruebas

Definiciones de tipo simple		
Nombre del tipo	**Definición del tipo**	**Comentarios**
Transport_classes	INTEGER(0, 1, 2, 3, 4)	Clases que se pueden utilizar para la conexión en la capa de transporte
String5	IA5String[5]	Cadena de longitud 5
SeqNumbers	INTEGER(0..127)	Todos los números de 0 a 127
PositiveNumbers	INTEGER(1..INFINITY)	Todos los números enteros positivos
String 10to20	IA5String[10..20]	Cadena, long. mín. 10 caracteres; long. máx. 20 caracteres

11.2.3.3 Definiciones de tipo estructurado mediante cuadros

Los tipos estructurados se pueden definir de forma tabular con el fin de utilizarlos para declarar objetos estructurados como subtipos dentro de las definiciones de ASP y de PDU y en otros tipos estructurados, etc.

Para cada tipo estructurado se facilitará la siguiente información:

a) nombre,

   según convenga, será el nombre completo que figura en la norma del protocolo pertinente; si se emplea una abreviatura, se colocará a continuación el nombre completo entre paréntesis;

b) las variaciones de codificación que se utilizarán en estructuras de este tipo en una PDU,

   con el fin de especificar variaciones de codificación explícitas para tipos estructurados completos, que sustituyan a las variaciones de codificación aplicables a cualquier PDU en la que se utilice este tipo estructurado, esta entrada optativa referenciará una entrada del cuadro de variaciones de codificación pertinente (por ejemplo, para cambiar de SD a LD(3)). Si no se utiliza esta entrada, las variaciones de codificación aplicables son las variaciones que son aplicables a cada PDU en cuyo interior se utiliza este tipo estructurado. Véase 11.16.4;

c) lista de los elementos asociados al tipo estructurado,

   en la que para cada elemento se facilitará la siguiente información:

   1) nombre,

      según convenga, será el nombre completo que figura en la norma del protocolo pertinente; si se emplea una abreviatura, se colocará a continuación el nombre completo entre paréntesis.

   2) tipo y un atributo optativo,

      cuyos elementos pueden ser de un tipo de estructura compleja arbitraria; no podrá haber referencias recursivas (ni directa ni indirectamente);

      como atributo optativo se puede usar la restricción de longitud a fin de proporcionar las longitudes máxima y mínima de un elemento del tipo cadena (véase 11.18);

   3) optativamente, un identificador de codificación específico seguido de una lista de parámetros reales necesarios, a fin de especificar una codificación explícita para el tipo estructurado, que sustituya a las reglas de codificación y las variaciones de codificación aplicables a cualquier PDU en la que se utilice este tipo estructurado; el identificador de codificación, si existe, identificará una de las variaciones de codificación o una definición de codificación de campo no válida definida en la serie de pruebas (por ejemplo, LD(10)); véase 11.16.4.

Se considera que los elementos de las definiciones de tipo estructurado son optativos, es decir que puede que no haya elementos completos en instancias de estos tipos.

Esta información se proporcionará con el formato del formulario siguiente.

Definiciones de tipo estructurado			
Nombre del tipo : *StructId&FullId*			
Grupo : *[StructTypeGroupReference]*			
Variación de codificación : *[EncVariationCall]*			
Comentarios : *[FreeText]*			
Nombre del elemento	**Definición del tipo**	**Codificación del tipo**	**Comentarios**
⋮ *ElemId&FullId* ⋮	⋮ *Type&Attributes* ⋮	⋮ *[PDU_FieldEncodingCall]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 8: Definiciones de tipo estructurado

11.2.3.4 Definiciones de tipo de series de pruebas mediante ASN.1

Se puede especificar el tipo de serie de pruebas mediante ASN.1. Esto se hace aplicando una definición ASN.1 con la sintaxis ASN.1 definida en la Rec. UIT-T X.680. Para cada tipo ASN.1 se facilitará la siguiente información:

a) nombre,

   según convenga, se usará el nombre completo que figura en la norma del protocolo pertinente; si se emplea una abreviatura se pondrá a continuación el nombre completo entre paréntesis;

b) las variaciones de codificación que se vayan a utilizar en estructuras de este tipo en una PDU;

   con el fin de especificar las variaciones de codificación explícitas para tipos ASN.1 completos, que sustituyan a las variaciones de codificación aplicables a cualquier PDU en la que se utilice este tipo ASN.1, esta entrada optativa deberá hacer referencia a una entrada en el cuadro de variaciones de codificación pertinente (por ejemplo, para cambiar de SD a LD(3)). Si no se utiliza esta entrada, las variaciones de codificación que se pueden aplicar son las variaciones que son aplicables a cada PDU en cuyo interior se utiliza este tipo ASN.1. Véase 11.16.4;

c) definición del tipo ASN.1,

   que se atendrá a la sintaxis definida en la Rec. UIT-T X.680, salvo que exista la opción adicional de especificar una variación de codificación o codificación de campo no válida asociada con el tipo ASN.1 completo o con cualquier tipo ASN.1 dentro de ASN.1_Type. Esto se lleva a cabo proporcionando un identificador de codificación específico, seguido de una lista de parámetros reales necesarios, al objeto de especificar codificaciones explícitas para campos individuales u otros subtipos de una PDU, que sustituya a las reglas de codificación y a las variaciones de codificación aplicables a la PDU como un conjunto; el identificador de codificación, si existe, deberá identificar, bien una de las variaciones de codificación, bien una definición de codificación de campo no válida definida en la serie de pruebas (por ejemplo, LD(10)); véase 11.16.4.

Dentro de esa definición no se puede usar el símbolo guión (-) para los identificadores, pero se puede usar el símbolo subrayado (_). El identificador de tipo del encabezamiento del cuadro es el nombre del primer tipo definido en el cuerpo del cuadro.

Los tipos a los que hace referencia la definición de tipo se definirán en otros cuadros de definición de tipo ASN.1 mediante referencias en el cuadro de referencias de tipo ASN.1 o de forma local en el mismo cuadro, siguiendo la definición del primer tipo. Los tipos definidos localmente no se utilizarán en otras partes de la serie de pruebas.

Las definiciones de tipo ASN.1 utilizadas en la TTCN no harán uso de referencias de tipo externas, como las definidas en la Rec. UIT-T X.680, pero dentro del cuerpo del cuadro se pueden hacer comentarios. En este cuadro no hay una columna "Comentarios".

Los comentarios en ASN.1 empiezan con "--" y terminan con el "--" siguiente o con "end of line", el primero que aparezca. Ello evita que un solo comentario ASN.1 se extienda a lo largo de varias líneas. Se recomienda el uso de los especificadores de ATS para facilitar el intercambio de ATS en la TTCN.MP, y terminar siempre los comentarios ASN.1 con "--".

Esta información se proporcionará con el formato del formulario siguiente.

Definición de tipo ASN.1		
Nombre del tipo	**:**	*ASN1_TypeId&FullId*
Grupo	**:**	*[ASN1_TypeGroupReference]*
Variación de codificación	**:**	*[EncVariationCall]*
Comentarios	**:**	*[FreeText]*
Definición del tipo		
⋮ *ASN1_Type&LocalTypes* ⋮		
Comentarios detallados:	*[FreeText]*	

Formulario 9: Definición de tipo ASN.1

EJEMPLO 12 – Definición de tipo ASN.1 para serie de pruebas:

Definición de tipo ASN.1
Nombre del tipo : DATE_type **Comentarios** : Para ilustrar la estructura de definiciones de tipo ASN.1.
Definición del tipo

SEQUENCE {
día DAY_type,
mes MONTH_type,
año YEAR_type
}
-- *local DAY_type* --
DAY_type::= INTEGER {first(), last(31)}
-- *MONTH_type y YEAR_type están definidos en otros cuadros de definición de tipo ASN.1* --

11.2.3.5 Definiciones de tipo ASN.1 por referencia

Los tipos se pueden especificar mediante una referencia precisa a un tipo ASN.1, definido en una norma OSI o mediante referencia a un tipo ASN.1 definido en un módulo ASN.1 adjuntado a la serie de pruebas. Para cada tipo se facilitará la siguiente información:

a) nombre,

que se podrá usar en toda la serie de pruebas. El nombre se especificará sin un FullIdentifier;

b) la referencia a tipo,

que se atendrá a las normas de identificador establecidas en la Rec. UIT-T X.680;

c) identificador de módulo,

consistente en una referencia a módulo, que se atendrá a las reglas de identificador establecidas en la Recomendación UIT-T X.680, y un ObjectIdentifier optativo; el módulo será exclusivo dentro del dominio de interés;

d) las variaciones de codificación que se vayan a utilizar en tales tipos ASN.1 en una PDU,

con el fin de especificar las variaciones de codificación explícitas para tipos ASN.1 completos, que sustituyan a las variaciones de codificación aplicables a cualquier PDU en la que se utilice este tipo ASN.1, esta entrada optativa deberá hacer referencia a una entrada en el cuadro de variaciones de codificación pertinente (por ejemplo, para cambiar de SD a LD(3)). Si no se utiliza esta entrada, las variaciones de codificación que se pueden aplicar son las variaciones que son aplicables a cada PDU en cuyo interior se utiliza este tipo ASN.1. Véase 11.16.4.

Esta información se proporcionará con el formato del formulario siguiente.

Definición de tipo ASN.1 por referencia				
Grupo : *[ASN1_TypeGroupReference]*				
Nombre del tipo	**Referencia al tipo**	**Identificador del módulo**	**Variación de codificación**	**Comentarios**
⋮ *ASN1_TypeId&FullId* ⋮	⋮ *TypeReference* ⋮	⋮ *ASN1_ModuleIdentifier* ⋮	⋮ *[EncVariationCall]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*				

Formulario 10: Definiciones de tipo ASN.1 por referencia

En este cuadro se puede hacer comentarios colectivos de conformidad con la figura 2.

Los tipos ASN.1 importados desde módulos ASN.1 pueden contener identificadores, por lo que las referencias de tipo y las referencias de valor que siguen las reglas de identificador de la Rec. UIT-T X.680, pueden contener guiones. Para poder utilizar las definiciones importadas en TTCN es necesario sustituir los guiones de los identificadores importados por el símbolo subrayado. Esto se efectúa en el proceso de importación.

EJEMPLO 13 – La siguiente definición de tipo en módulo ASN.1:

```
module-1 DEFINITIONS BEGIN
    Type-1 ::= SEQUENCE     {     field1    Sub-Type-1,
                                  field2    BIT STRING {first-bit(0), second-bit(1) } }
END
```

se puede importar en TTCN con:

Definiciones de tipo ASN.1 por referencia			
Nombre del tipo	**Referencia al tipo**	**Identificador del módulo**	**Comentarios**
Type_1	Type-1	module-1	
Sub_Type_1	Sub-Type-1	module-1	

La definición de Type-1 por referencia dada más arriba es equivalente a la siguiente definición:

Definición de tipo ASN.1		
Nombre del tipo	**:**	Type_1
Comentarios	**:**	
Definición del tipo		
SEQUENCE {	field1	Sub_Type_1,
	field2	BIT STRING {first_bit(0), second_bit(1) } }

11.3 Operadores de TTCN y operaciones de TTCN

11.3.1 Introducción

La TTCN soporta cierto número de operadores, operaciones y mecanismos predefinidos, que permiten definir las operaciones serie de pruebas. Estos operadores y operaciones se pueden usar en cualesquiera descripciones de comportamiento dinámico y constricciones.

11.3.2 Operadores de TTCN

11.3.2.1 Introducción

Hay tres categorías de operadores predefinidos:

a) aritméticos;

b) relacionales;

c) booleanos.

En el cuadro 3 se indica la precedencia de estos operadores. Se pueden usar paréntesis para agrupar operandos en expresiones. Una expresión que figura entre paréntesis tiene la máxima precedencia a efectos de evaluación.

En cualquier fila del cuadro 3, los operadores indicados tienen la misma precedencia. Si en una expresión aparece más de un operador de la misma precedencia, se evalúan las operaciones de izquierda a derecha.

Cuadro 3/X.292 – Precedencia de los operadores

Máxima ↓ Mínima	( ) Unarios: + – NOT Binarios: * / MOD AND + – OR = < > <> >= <=

11.3.2.2 Operadores aritméticos predefinidos

Los operadores aritméticos predefinidos son:

"+", "-", "*", "/", **MOD**

Representan las operaciones de adición, sustracción, multiplicación, división y módulo. Los operandos de esos operadores pueden ser de tipo INTEGER (esto es, predefinidos en TTCN o en ASN.1) o derivaciones de INTEGER (por ejemplo, subgama). En expresiones aritméticas no se pueden usar valores denominados de ASN.1 como operandos de operaciones.

El tipo resultante de operaciones aritméticas es INTEGER.

Las reglas de los operandos se aplican también en el caso en que el operador más (+) o menos (–) se utilice como operador unario. El resultado de la utilización del operador "menos" es el valor negativo del operando si éste era positivo, y a la inversa.

El resultado de efectuar la operación de división (/) sobre dos valores INTEGER proporciona el valor INTEGER resultante de dividir el primer INTEGER por el segundo (es decir, se descartan las fracciones).

El resultado de efectuar la operación MOD sobre dos valores INTEGER proporciona el resto de dividir el primer INTEGER por el segundo.

11.3.2.3 Operadores relacionales predefinidos

Los operadores relacionales predefinidos son los siguientes:

"=" | "<" | ">" | "<>" | ">=" | "<="

Representan las relaciones de igualdad, menor que, mayor que, no igual a, mayor o igual que, menor o igual que. Los operandos de igual (=) y no igual (<>), pueden ser de tipo arbitrario. Los dos operandos deben ser compatibles. Los demás operadores relacionales tendrán operandos del tipo INTEGER solamente o derivados de INTEGER. El tipo de resultado de estas operaciones es BOOLEAN.

En comparaciones de cadenas, BITSTRING, HEXSTRING, OCTETSTRING y todos los géneros de CharacterStrings podrán contener los caracteres comodín AnyOrNone (*) y AnyOne (?). En estos casos, se efectúa la comparación según las reglas de concordancia de patrones definidas en 12.6.2.

11.3.2.4 Operadores booleanos predefinidos

Los operadores booleanos predefinidos son:

NOT AND OR

Representan las operaciones de negación "Y lógico" y "O lógico". Sus operandos serán de tipo BOOLEAN (TTCN o ASN.1 o predefinidos). El tipo de resultado de los operadores booleanos es BOOLEAN.

El Y lógico devuelve el valor TRUE si ambos operandos tienen el valor TRUE; en cualquier otro caso, devuelve el valor FALSE. El O lógico devuelve el valor TRUE si al menos uno de los operandos toma el valor TRUE; y devuelve el valor FALSE sólo en el caso en que ambos operandos tomen el valor FALSE. El NO lógico es un operador unario que devuelve el valor TRUE si el operando toma el valor FALSE y devuelve el valor FALSE si el operador toma el valor TRUE.

11.3.3 Operaciones predefinidas

11.3.3.1 Introducción

Las operaciones predefinidas son de dos categorías:

a) operaciones de conversión;

b) otras operaciones.

Las operaciones predefinidas se pueden usar en cualquier serie de pruebas. No requieren una definición explícita mediante un cuadro de definición de operación serie de pruebas. Cuando se invoque una operación predefinida:

a) el número de parámetros reales será igual al número de parámetros formales; y

b) cada parámetro real tomará el valor de un elemento de su tipo de parámetro formal correspondiente; y

c) todas las variables que aparecen en la lista de parámetros tienen que estar acotadas.

Cada una de las operaciones predefinidas se presenta con el siguiente formato:

OPERATION_NAME (FORMAL_PARAMETER_LIST) ⇒ RESULT_TYPE

11.3.3.2 Operaciones de conversión predefinidas

11.3.3.2.1 Introducción

La TTCN soporta las siguientes operaciones predefinidas para las conversiones de tipos:

a) HEX_TO_INT convierte HEXSTRING en INTEGER;

b) BIT_TO_INT convierte BITSTRING en INTEGER;

c) INT_TO_HEX convierte INTEGER en HEXSTRING;

d) INT_TO_BIT convierte INTEGER en BITSTRING.

Estas operaciones proporcionan reglas de codificación en el contexto de las operaciones solamente. No es admisible suponer que estas reglas de codificación se aplican fuera del dominio de las operaciones en la TTCN.

11.3.3.2.2 HEX_TO_INT

HEX_TO_INT(hexvalue:HEXSTRING) ⇒ INTEGER

Esta operación convierte un valor HEXSTRING único en un valor INTEGER único.

A efectos de esta conversión, un HEXSTRING se interpretará como un valor INTEGER positivo de base 16. El dígito HEX más a la derecha es el menos significativo. El dígito HEX más a la izquierda es el más significativo. Los dígitos HEX 0 ... F representan los valores decimales 0 ... 15, respectivamente.

11.3.3.2.3 BIT_TO_INT

BIT_TO_INT(bitvalue:BITSTRING) ⇒ INTEGER

Esta operación convierte un valor BITSTRING único en un valor INTEGER único.

A efectos de esta conversión, un BITSTRING se interpretará como un valor INTEGER positivo de base 2. El BIT más a la derecha es el menos significativo y el BIT más a la izquierda es el más significativo. Los bits 0 y 1 representan los valores decimales 0 y 1, respectivamente.

11.3.3.2.4 INT_TO_HEX

INT_TO_HEX(intvalue, slength:INTEGER) ⇒ HEXSTRING

Esta operación convierte un valor INTEGER único en un valor HEXSTRING único. La cadena resultante tiene una longitud igual a *slength* (longitud de cadena) dígitos HEX.

A efectos de esta conversión, un HEXSTRING se interpretará como un valor INTEGER positivo de base 16. El dígito HEX más a la derecha es el menos significativo, el dígito HEX más a la izquierda es el más significativo. Los dígitos HEX 0 ... F representan los valores decimales 0 ... 15, respectivamente.

Si la conversión genera un valor con menos dígitos HEX que los especificados en el segundo parámetro, deberá rellenarse el HEXSTRING con ceros por la izquierda.

Se producirá un error de caso de prueba si *intvalue* (valor entero) es negativo o si el HEXSTRING resultante contiene más dígitos HEX que los especificados en el segundo parámetro.

11.3.3.2.5 INT_TO_BIT

INT_TO_BIT(intvalue, slength:INTEGER) ⇒ BITSTRING

Esta operación convierte un valor INTEGER único en un valor BITSTRING único. La cadena resultante tiene una longitud de *slength* bits.

A efectos de esta conversión un BITSTRING se interpretará como un valor INTEGER positivo de base 2. El bit más a la derecha es el menos significativo, el bit más a la izquierda es el más significativo. Los bits 0 y 1 representan los valores decimales 0 y 1, respectivamente.

Si la conversión genera un valor con un número de bits menor que el especificado en el segundo parámetro, deberá rellenarse el BITSTRING con ceros por la izquierda.

Se producirá un error de caso de prueba si *intvalue* es negativo o si el BITSTRING resultante contiene más bits que los especificados en el segundo parámetro.

11.3.3.3 Otras operaciones predefinidas

11.3.3.3.1 IS_PRESENT

IS_PRESENT(DataObjectReference) ⇒ BOOLEAN

Como argumento, la operación tomará una referencia a un campo dentro de un objeto de datos solamente si se ha definido como OPTIONAL o si tiene un valor DEFAULT. El campo puede ser de cualquier tipo. El resultado de aplicar la operación es el valor BOOLEAN TRUE si, y sólo si, el valor de este campo está presente en la instancia efectiva del objeto de datos. En cualquier otro caso el resultado es FALSE.

El argumento de la operación tendrá el formato definido en 15.10.2.

EJEMPLO 14 – Utilización de IS_PRESENT:

```
si la unidad de datos de protocolo recibida (received_PDU) es del tipo ASN.1
      SEQUENCE      {       field_1   INTEGER OPTIONAL,
                            field_2   SEQUENCE OF INTEGER          }
entonces la llamada de operación
      IS_PRESENT(received_PDU.field_1)
tomará el valor TRUE si está presente field_1 en la instancia de received_PDU.
```

11.3.3.3.2 IS_CHOSEN

IS_CHOSEN(DataObjectReference) ⇒ BOOLEAN

La operación devuelve el valor BOOLEAN TRUE si, y sólo si, la referencia al objeto de datos especifica la variante del tipo CHOICE seleccionada realmente para un objeto de datos determinado. En cualquier otro caso, el resultado es FALSE. La operación no se puede aplicar a objetos de datos o campos de objetos de datos diferentes del tipo ASN.1 CHOICE. El argumento de la operación tendrá el formato definido en 15.10.2.

EJEMPLO 15 – Utilización de IS_CHOSEN:

```
si received_PDU es del tipo ASN.1
      CHOICE        {       p1 PDU_type1,
                            p2 PDU_type2,
                            p3 PDU_type }
entonces la llamada de operación
      IS_CHOSEN(received_PDU.p2)
devuelve TRUE si la instancia de received_PDU transporta una PDU del tipo PDU_type2.
```

11.3.3.3.3 NUMBER_OF_ELEMENTS

NUMBER_OF_ELEMENTS(Value) ⇒ INTEGER

La operación devuelve el número real de elementos de un valor que sea del tipo ASN.1 SEQUENCE OF o SET OF. Su resultado es totalmente compatible con el de la constricción ASN.1 SIZE equivalente aplicada a objetos de estos tipos. La operación no se puede aplicar a valores diferentes del tipo ASN.1 SEQUENCE OF o SET OF. El argumento de la operación tendrá el formato definido en 15.10.2.

EJEMPLO 16 – Utilización de NUMBER_OF_ELEMENTS:

```
si received_PDU es del tipo ASN.1
     SEQUENCE      {        field_1    INTEGER OPTIONAL,
                            field_2    SEQUENCE OF INTEGER }
entonces la llamada de operación
     NUMBER_OF_ELEMENTS(received_PDU.field_2)
devuelve el número de elementos de SEQUENCE OF INTEGER dentro del objeto de datos real received_PDU.
Además, NUMBER_OF_ELEMENTS ({3, 0, 5}) devuelve 3.
```

11.3.3.3.4 LENGTH_OF

LENGTH_OF(Value) ⇒ INTEGER

La operación devuelve la longitud real de un valor que es del tipo BITSTRING, HEXSTRING, OCTETSTRING o CharacterString, o del tipo ASN.1 BIT STRING u OCTET STRING. En el cuadro 5 de 11.18.2, se definen las unidades de longitud de cada tipo cadena.

NOTA – Estas unidades de longitud son compatibles con las utilizadas en las constricciones SIZE en ASN.1 para objetos de tipos ASN.1, pero no para los valores literales que en este contexto de TTCN se consideran del tipo TTCN correspondiente. De este modo, una cadena string tal como 'F3'H, que podría ser del tipo BIT STRING u OCTET STRING en ASN.1, será interpretada como la HEXSTRING de tipo TTCN.

El argumento de la operación tendrá el formato definido en 15.10.2.

La operación no se puede aplicar a valores diferentes de BITSTRING, HEXSTRING, OCTETSTRING o CharacterString, o del tipo ASN.1 BIT STRING u OCTET STRING.

EJEMPLO 17 – Utilización de LENGTH_OF:

```
Si S es del tipo BITSTRING o tipo ASN.1 BIT STRING e ='010'B, entonces LENGTH_OF(S) devuelve 3.
Si S es del tipo HEXSTRING e ='F3'H, entonces LENGTH_OF(S) devuelve 2.
Si S es del tipo OCTETSTRING e ='F2'O, entonces LENGTH_OF(S) devuelve 1.
Si S es del tipo CharacterString e ="EXAMPLE", entonces LENGTH_OF(S) devuelve 7.
Si S es del tipo ASN.1 BIT STRING e ='F3'H, entonces LENGTH_OF(S) devuelve 8.
Si S es del tipo ASN.1 OCTET STRING e ='F3'H, entonces LENGTH_OF(S) devuelve 1.
Si S es del tipo ASN.1 OCTET STRING e ='01010011'B, entonces LENGTH_OF(S) devuelve 1.
También, LENGTH_OF (INT_TO_HEX (26, 4)) devuelve 4.
LENGTH_OF ('F3'H) devuelve 2
y, LENGTH_OF ("Length_of Example") devuelve 17.
```

11.3.4 Definiciones y descripciones de operaciones serie de pruebas

11.3.4.1 Introducción

El especificador de ATS puede definir operaciones específicas de una serie de pruebas. Para definir una nueva operación se facilitará la siguiente información:

a) Nombre de la operación.

b) Lista de parámetros de entrada y sus tipos.

Ésta es una lista de los nombres y tipos de los parámetros formales. Cada nombre de parámetro irá seguido del carácter dos puntos (:), y a continuación el nombre del tipo de parámetro.

Cuando se utilice más de un parámetro del mismo tipo se especificarán los parámetros mediante una sublista de parámetros. Cuando se emplee la sublista de parámetros, los nombres de los parámetros irán separados por coma (,). El parámetro final de la lista irá seguido por dos puntos (:), y a continuación el nombre del tipo de parámetro.

Cuando se utilice más de un par de parámetro y tipo (o par de lista de parámetros y tipo), los pares irán separados entre sí mediante punto y coma (;).

Como tipos de parámetro formal solamente se podrán utilizar tipos predefinidos y tipos de datos tal y como están definidos en las definiciones de tipos de serie de pruebas, definiciones de tipo de ASP o definiciones de tipo de PDU. No podrán utilizarse los tipos de PCO como tipos de parámetro formal. Se darán valores a todos los parámetros, lo que significa que al evaluar una llamada de operación serie de pruebas, se asignarán los parámetros reales a los correspondientes parámetros formales, al igual que en un enunciado asignación.

EJEMPLO 18 – Listas de parámetros

Los métodos que siguen son formas equivalentes de especificación de una lista de parámetros con dos parámetros INTEGER y un parámetro BOOLEAN:

(A:INTEGER; B:INTEGER; C:BOOLEAN)

(A, B:INTEGER; C:BOOLEAN)

c) Tipo del resultado,

que se atendrá a las reglas para los tipos de parámetros indicadas en b).

d) Definición de la operación,

que consistirá en:

1) una definición de procedimiento, cuya evaluación produce la evaluación de un enunciado RETURNVALUE que proporciona el resultado de la operación, y que incluye comentarios explicativos insertados en lugares apropiados de la definición de procedimiento como texto delimitado por "/*" y "*/"; o

2) una descripción de la operación en forma de texto, que puede incluir una referencia a una especificación de disponibilidad general del algoritmo aplicable al invocar la operación, seguida al menos de un ejemplo que muestre una invocación y su resultado correspondiente. La explicación comenzará con el nombre de la operación, seguido de una lista entre paréntesis que contiene los nombres de los parámetros de la operación; esto es una invocación "patrón" de la operación.

e) Optativamente, otros comentarios que describen la operación, recogidos en la parte "Comentarios" del encabezamiento del cuadro o en la parte "Comentarios detallados" del cuadro.

Se recomienda el uso de definiciones de procedimiento para aportar precisión a la definición de las operaciones, pero se permite una explicación textual como alternativa para la compatibilidad hacia atrás.

Para una definición de procedimiento, esta información se proporcionará con el formato del formulario siguiente.

Definición de procedimiento de la operación serie de pruebas	
Nombre de la operación	: *TS_ProcId&ParList*
Grupo	: *[TS_ProcGroupReference]*
Tipo de resultado	: *Type*
Comentarios	: *[FreeText]*
Definición	
TS_OpProcDef	
Comentarios detallados:	*[FreeText]*

Formulario 11: Definición de procedimiento de la operación serie de pruebas

En el caso de una descripción textual, esta información se proporcionará con el formato del formulario siguiente:

Descripción de la operación serie de pruebas	
Nombre de la operación	: *TS_OpId&ParList*
Grupo	: *[TS_OpGroupReference]*
Tipo de resultado	: *Type*
Comentarios	: *[FreeText]*
Descripción	
FreeText	
Comentarios detallados:	*[FreeText]*

Formulario 12: Descripción de la operación serie de pruebas

11.3.4.2 Parámetros

Se puede comparar una operación serie de pruebas una función en un lenguaje de programación ordinario. Los valores solo deberán transferirse a la operación mediante parámetros formales. Cada parámetro formal deberá declararse como un tipo predefinido, un identificador de tipo de serie de pruebas, identificador de tipo de ASP, identificador de tipo de CM o el metatipo **PDU**. En una definición de procedimiento de una operación serie de pruebas, no se utilizarán directamente variables de serie de pruebas, variables de caso de prueba, constantes de serie de pruebas, parámetros de serie de pruebas y constricciones, pero, si resulta necesario en la operación serie de pruebas, deberán transferirse como parámetros reales.

No habrá efectos secundarios, es decir, los parámetros de la operación no se alterarán como consecuencia de una llamada de operación. En una definición de procedimiento de una operación serie de pruebas se pueden utilizar operaciones predefinidas y otras operaciones serie de pruebas, sin que sea necesario transferirlas como parámetros reales.

Cuando se invoque una operación serie de pruebas:

a) el número de parámetros reales será el mismo que el número de parámetros formales;

b) cada parámetro real tomará el valor de un elemento de su tipo de parámetro formal correspondiente;

c) todas las variables que aparezcan en la lista de parámetros estar acotadas; y

d) se deberá asignar valor a los parámetros reales.

11.3.4.3 Variables e identificadores

Cuando se utilice una definición de procedimiento, se puede incluir la declaración de variables locales, situada al principio de la definición de procedimiento, entre las palabras clave **VAR** y **ENDVAR**. Estas variables pueden ser de cualquiera de los tipos admitidos en la TTCN. El ámbito de estas variables locales la definición de procedimiento en sí misma. Estas declaraciones declaran listas de identificadores de variables, cada uno de un tipo determinado, y cada lista puede ser declarada o no declarada como **STATIC**. Se puede dar un valor inicial optativo tanto a las variables declaradas como **STATIC** como a las no declaradas.

NOTA – Se recomienda proporcionar siempre las variables **STATIC** con un valor inicial.

Las variables no declaradas como **STATIC** se inicializan cada vez que se invoca la operación, con el valor inicial especificado, si existe, y por tanto las variables no deberán transportar un valor de una evaluación de la operación serie de pruebas a otra. Las variables declaradas como **STATIC** son inicializadas con el valor inicial especificado, si existe, la primera vez que se invoca la operación dentro de un componente de prueba dado, o dentro de un caso de prueba si no se utilizan componentes de prueba, y mantienen sus valores de una invocación a la siguiente dentro de ese componente de prueba o caso de prueba.

Se considera que las variables a las que no se asigna un valor inicial no están acotadas, y habrá que acotarlas explícitamente a un valor mediante una asignación en el cuerpo de la operación antes de utilizarlas en una expresión. Cuando se utiliza una variable no acotada en una expresión, se trata de un error de caso de prueba.

Cada identificador utilizado en la definición de procedimiento de una operación serie de pruebas será uno de los siguientes:

a) nombre de variable declarada localmente;

b) nombre del tipo utilizado en una declaración de variable;

c) nombre de parámetro formal declarado en una lista de parámetros formales de la operación;

d) nombre de la operación serie de pruebas.

El ámbito de los nombres de parámetros formales y de los nombres de variables declaradas localmente es la definición de procedimiento de la operación serie de pruebas. De este modo, los valores de todos los demás tipos de identificador no son directamente accesibles dentro de la definición de procedimiento de una operación serie de pruebas. Para acceder a tales valores, éstos se transferirán como parámetros reales a la operación serie de pruebas.

11.3.4.4 Enunciados de procedimiento

En una definición de procedimiento, después de la declaración de variables locales, si existe, habrá un enunciado procedimiento de alguna de las siguientes clases:

a) un enunciado devolución (*Return statement*);

b) un enunciado asignación (*Assignment statement*);

c) un enunciado If (*If statement*);

d) un bucle While (*While loop*);

e) un enunciado caso (*Case statement*);

f) un bloque que contiene una secuencia de enunciados de procedimiento separados por punto y coma, todos ellos encerrados por las palabras clave **BEGIN** y **END**.

Se puede insertar comentarios como texto dentro de los enunciados de procedimientos, delimitados por "/*" y "*/". No se insertarán comentarios dentro de otros comentarios.

11.3.4.5 Enunciados Return Value (devolución de valor)

Cada evaluación de una operación serie de pruebas terminará con la evaluación de un enunciado ReturnValue, que se compone de la palabra clave **RETURNVALUE** seguida de una expresión. Este enunciado volverá el valor de la expresión dada como resultado de la operación serie de pruebas. El tipo de este resultado deberá concordar con el Result Type especificado en el encabezamiento del cuadro de definiciones de operaciones serie de pruebas.

11.3.4.6 Enunciados Assignment (asignación)

La forma de Assignment es igual a la de las descripciones de comportamiento en TTCN (véase 15.10.4), salvo que no va encerrado entre paréntesis. El DataObjectReference izquierda deberá comenzar con una variable local. Si el tipo de variable local es un tipo estructurado, entonces DataObjectReference puede acceder a un componente de la estructura (con una referencia a registro, una referencia a matriz, o una referencia a bit, según convenga; véanse 15.10.2 y 15.10.3).

11.3.4.7 Enunciados If

Existen dos formas de enunciado If:

- **IF** expression **THEN** procedure-statement **ELSE** procedure-statement **ENDIF**;
- **IF** expression **THEN** procedure-statement **ENDIF**.

La expresión que sigue a la palabra clave **IF** se evaluará en primer lugar y tomará un valor booleano. Si toma el valor **TRUE**, deberá entonces evaluarse el enunciado procedimiento que sigue a la palabra clave **THEN**. Si la expresión toma el valor **FALSE**, entonces se evalúa, si existe, el enunciado procedimiento que sigue a la palabra clave **ELSE**. La utilización de la palabra clave **ENDIF** para finalizar el enunciado If permite que los enunciados de procedimiento que siguen a **THEN** y **ELSE** sean enunciados If sin que tengan que estar encerrados en un bloque.

11.3.4.8 While Loop (bucle While)

El bucle While tiene la forma:

- **WHILE** expression **DO** procedure-statement **ENDWHILE**.

La expresión que sigue a la palabra clave **WHILE** se evaluará en primer lugar y tomará un valor booleano. Si toma el valor **TRUE**, deberá entonces evaluarse el enunciado procedimiento que sigue a la palabra clave **DO** y a continuación, si no ha sido evaluado el enunciado ReturnValue, deberá repetirse el proceso arrancando de nuevo con la evaluación de la expresión. Tan pronto como la expresión toma el valor **FALSE** se completa la evaluación del bucle While.

11.3.4.9 Enunciado Case (caso)

Un enunciado caso toma una de las dos formas siguientes:

- **CASE** expression **OF**
  integer-label_1: procedure-statement_1;
  integer-label_2: procedure-statement_2;
  ...
  integer-label_n: procedure-statement_n;
  ELSE
  procedure-statement
  ENDCASE
- **CASE** expression **OF**
  integer-label_1: procedure-statement_1;
  integer-label_2: procedure-statement_2;
  ...
  integer-label_n: procedure-statement_n;
  ENDCASE

La expresión que sigue a la palabra clave **CASE** se evaluará en primer lugar y tomará un valor entero positivo que habrá de concordar como máximo con una de las etiquetas integer del cuerpo del enunciado del caso. El enunciado procedimiento que se encuentra a continuación de la etiqueta integer concordante, si existe, será evaluado y con ello se completa la evaluación del enunciado procedimiento. Sin embargo, si el resultado de la evaluación de la expresión no

concuerda con ninguna de las etiquetas integer, el enunciado procedimiento que sigue a la palabra clave **ELSE**, si existe, será evaluado y con ello se completa el enunciado procedimiento. Pero cuando no hay concordancia con una etiqueta integer o con una cláusula **ELSE**, entonces el resultado del enunciado caso es un error de caso de prueba. Por tanto, el enunciado caso es equivalente a una secuencia anidada de enunciados If, cada uno de los cuales comprueba la expresión "(expression) = integer-label_i", que puede ir seguida de una cláusula **ELSE** en el nivel más interno del anidado.

11.3.4.10 Utilización de las operaciones serie de pruebas

Una operación serie de pruebas, junto con su lista de parámetros reales, se puede utilizar dondequiera que esté permitida una expresión.

Cada operación serie de pruebas debe incluir la comprobación de errores apropiada. Si durante la evaluación de una operación serie de pruebas se detecta un error (por ejemplo, una división por cero, un parámetro no válido, una discordancia de tipos o la evaluación de una variable no acotada), el resultado será un error de caso de prueba.

EJEMPLO 19 – Definición de la operación SUBSTR:

Descripción de la operación serie de pruebas
Nombre de la operación : SUBSTR (source:IA5String; start_index, length:INTEGER) **Tipo de resultado :** IA5String
Descripción
SUBSTR(source, start_index, length) es la cadena de longitud *len* que comienza en el índice *start_index* de la cadena origen *source*. Por ejemplo: SUBSTR("abcde",3,2) = "cd" SUBSTR("abcde",1,3) = "abc" *SUBSTR(source, start_index, len)* solamente se definirá si: *start_index* >= 1, *len* >= 0, y *start_index* + *len* <= *(length of source)* + 1. Cualquier tentativa de evaluar SUBSTR aplicada a argumentos en los cuales no está definida dará como resultado un error de caso de prueba.

EJEMPLO 20 – Definición de la operación NUMBER_OF_INVOCATIONS:

Definición de procedimiento de la operación serie de pruebas
Nombre de la operación : NUMBER_OF_INVOCATIONS **Tipo de resultado :** INTEGER
Definición
VAR STATIC COUNT : INTEGER : 0 **ENDVAR** **BEGIN** COUNT := COUNT + 1; **RETURNVALUE** COUNT **END**
Comentarios detallados: NUMBER_OF_INVOCATIONS() da un valor entero igual al número de veces que se ha invocado la operación en el componente de prueba actual, o el caso de prueba si no se utilizan componentes de prueba.

11.4 Declaraciones de parámetros de serie de pruebas

La finalidad de esta parte de la ATS es declarar constantes derivadas de PICS y/o PIXIT que se utilizan para parametrizar globalmente la serie de pruebas. Se denomina a estas constantes parámetros de serie de pruebas y se utilizan como base para la selección de casos de prueba y la parametrización de casos de prueba.

Se facilitará la siguiente información relativa a cada parámetro la serie de pruebas:

a) nombre;

b) tipo,

que será un tipo predefinido, un tipo ASN.1 un tipo de serie de pruebas o un tipo PDU;

c) valor por defecto, si existe,

que se puede utilizar para proponer valores adecuados para algunos parámetros de la serie de pruebas, tales como la duraciones de la temporización;

d) referencia a entrada PICS/PIXIT,

que es una referencia a una entrada de formulario PICS/PIXIT individual que identificará con claridad dónde se encuentra el valor que se utilizará para este parámetro serie de pruebas.

Esta información se proporcionará con el formato del formulario siguiente.

Declaraciones de parámetro serie de pruebas				
Grupo : *[TS_ParGroupReference]*				
Nombre del parámetro	**Tipo**	**Valor por defecto**	**Ref. de PICS/PIXIT**	**Comentarios**
⋮ *TS_ParIdentifier* ⋮	⋮ *Type* ⋮	⋮ *[DefaultValue]* ⋮	⋮ *FreeText* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*				

Formulario 13: Declaraciones de parámetro serie de pruebas

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

EJEMPLO 21 – Declaración de parámetros de serie de pruebas:

Declaraciones de parámetros de serie de pruebas			
Nombre del parámetro	**Tipo**	**Ref. de PICS/PIXIT**	**Comentarios**
PAR1	INTEGER	PICS question xx	
PAR2	INTEGER	PICS question yy	
PAR3	INTEGER	PICS question zz	

11.5 Definiciones de expresión de selección de caso de prueba

La finalidad de esta parte de la ATS es la definición de expresiones de selección para su uso en el proceso de selección de casos de prueba. Esta parte de la ATS deberá cumplir los requisitos de la Rec. UIT-T X.291.

Una expresión de selección se asocia a uno o más grupos de prueba y/o casos de prueba situando su identificador en la columna de referencia a selección de casos de prueba de la estructura de serie de pruebas y/o Índice de casos de prueba. Una expresión se puede referenciar mediante más de un grupo de prueba y/o caso de prueba.

La utilización de una expresión de selección se interpretará como que hay que ejecutar el caso de prueba si la expresión de selección toma el valor TRUE.

Se facilitará la siguiente información relativa a cada expresión de selección de caso de prueba:

a) nombre;

b) una expresión de selección,

que tomará un valor booleano y utilizará en sus términos únicamente valores literales, parámetros de serie de pruebas, constantes de serie de pruebas y otros identificadores de expresión de selección.

Esta información se proporcionará con el formato del formulario siguiente.

Definiciones de expresión de selección de caso de prueba		
Grupo : *[SelectExprGroupReference]*		
Nombre de la expresión	**Expresión de la selección**	**Comentarios**
⋮ *SelectExprIdentifier* ⋮	⋮ *SelectionExpression* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*		

Formulario 14: Definiciones de expresión de selección de caso de prueba

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

11.6 Declaraciones de constantes de series de pruebas

La finalidad de esta parte de la ATS es declarar un conjunto de nombres para valores *no* derivados del PICS o PIXIT que permanecerán constantes en la serie de pruebas.

Se facilitará la siguiente información relativa a cada constante de serie de pruebas:

a) nombre;

b) tipo,

que será un tipo predefinido, un tipo sencillo o un tipo ASN.1 (que incluye los tipos de las PDU, las ASP y los CM expresados en ASN.1);

c) valor,

que no puede contener los términos de la expresión de valor variables de serie de pruebas o variables de casos de prueba. El valor tomará el valor de un elemento del tipo indicado en la columna de tipo.

Esta información se proporcionará con el formato del formulario siguiente.

Declaraciones de constantes de serie de pruebas			
Grupo : *[TS_ConstGroupReference]*			
Nombre de la constante	**Tipo**	**Valor**	**Comentarios**
⋮ *TS_ConstIdentifier* ⋮	⋮ *Type* ⋮	⋮ *ConstantExpression* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 15: Declaraciones de constantes de serie de pruebas

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

EJEMPLO 22 – Declaración de constantes de serie de pruebas:

Declaraciones de constantes de serie de pruebas			
Nombre de la constante	**Tipo**	**Valor**	**Comentarios**
TS_CONST1 TS_CONST2	BOOLEAN IA5String	TRUE "A string"	

11.7 Declaraciones de constantes de series de pruebas por referencia

La finalidad de esta parte de la ATS es declarar un conjunto de nombres de valores *no* derivados del PICS o PIXIT que permanecerán constantes en la serie de pruebas.

Se facilitará la siguiente información relativa a cada constante de serie de pruebas:

a) nombre;

b) tipo,

que será un tipo predefinido o un tipo ASN.1 (que incluye los tipos de PDU, ASP o CM expresados en ASN.1) importado por una definición de tipo ASN.1 por referencia desde el módulo ASN.1 identificado por el identificador de módulo especificado;

c) valor de referencia,

que deberá corresponder a un elemento del tipo indicado en la columna de tipo;

d) identificador de módulo,

consistente en una referencia a módulo, que se atendrá a las reglas de identificador establecidas en la Rec. UIT-T X.680, y un ObjectIdentifier optativo; el módulo será exclusivo dentro del dominio de interés.

Esta información se proporcionará con el formato del formulario siguiente.

Declaraciones de constantes de series de pruebas por referencia				
Grupo : *[TS_ConstGroupReference]*				
Nombre de la constante	**Tipo**	**Referencia de valor**	**Identificador de módulo**	**Comentarios**
⋮ *TS_ConstIdentifier* ⋮	⋮ *Type* ⋮	⋮ *ValueReference* ⋮	⋮ *ASN1_ModuleIdentifier* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*				

Formulario 16: Declaraciones de constantes de series de pruebas por referencia

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

11.8 Variables de TTCN

11.8.1 Declaraciones de variables de series de pruebas

Una serie de pruebas puede utilizar un conjunto de variables definidas globalmente para la serie de pruebas y mantener sus valores en toda la serie de pruebas. Estas variables se denominan variables de serie de pruebas.

Se utiliza una variable de serie de pruebas siempre que sea necesario transferir información de un caso de pruebas a otro. En la TTCN concurrente, las variables de series de pruebas sólo serán utilizadas por el MTC.

Para cada declaración de variable, se facilitará la siguiente información:

a) nombre;

b) tipo,

que será un tipo predefinido, un tipo ASN.1, un tipo de serie de pruebas o un tipo de PDU;

c) valor inicial (si existe),

se utiliza la columna de valor inicial cuando se desee asignar un valor inicial a una variable de serie de pruebas en su punto de declaración; los términos de la expresión de valor no podrán contener variables de serie de pruebas ni variables de casos de pruebas. El valor tomará el valor de un elemento del tipo indicado en la columna de tipo. La especificación de un valor inicial es optativa.

Esta información se proporcionará con el formato del formulario siguiente.

Declaraciones de variables de series de pruebas			
Grupo : *[TS_VarGroupReference]*			
Nombre de la variable	**Tipo**	**Valor**	**Comentarios**
⋮ *TS_VarIdentifier* ⋮	⋮ *Type* ⋮	⋮ *[ConstantExpression]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 17: Declaraciones de variables de series de pruebas

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

Es posible que cada caso de prueba particular se ejecute con independencia de otros en la serie de pruebas, por lo que la utilización que se haga de las variables de casos de prueba no debe establecer supuestos en cuanto a la ordenación de la ejecución del caso de prueba.

EJEMPLO 23 – Declaración de las variables de serie de pruebas:

Declaraciones de variables de series de pruebas			
Nombre de la variable	**Tipo**	**Valor**	**Comentarios**
state	IA5String	"idle"	Se utiliza como indicación del estado estable final del caso de prueba precedente, si existe, para ayudar a la determinación del prólogo que ha de utilizarse.

11.8.2 Acotación de variables de series de pruebas

Inicialmente, las variables de series de pruebas son no acotadas. Dichas variables pueden devenir acotadas (o ser reacotadas) en los siguientes contextos:

a) en el punto de declaración, si se especifica un valor inicial;

b) cuando la variable de serie de pruebas aparezca en el lado izquierdo de un enunciado asignación (véase 15.10.4).

Una vez que una variable de serie de pruebas ha sido acotada a un valor, dicha variable mantendrá ese valor hasta que sea acotada a un valor diferente o termine la ejecución de la serie de pruebas, lo que ocurra primero.

Si una variable de serie de pruebas no acotada se utiliza en el lado derecho de una asignación, se produce un error de caso de prueba.

11.8.3 Declaraciones de variables de casos de prueba

Una serie de pruebas puede utilizar un conjunto de variables declaradas globalmente para esta serie de pruebas, pero cuyo ámbito se ha definido de manera que sea local al caso de prueba.

En la TTCN concurrente, cada componente de prueba, incluido el MTC, recibe, cuando es creado, una copia reciente de todas las variables de casos de prueba. Estas variables se denominan variables de caso de pruebas.

Para cada declaración de variable se facilitará la siguiente información:

a) nombre;

b) tipo,

que será un tipo predefinido, un tipo ASN.1, un tipo de serie de pruebas o un tipo de PDU;

c) valor inicial (si existe),

se utiliza la columna de valor inicial cuando se desea asignar un valor inicial a una variable de caso de prueba en su punto de declaración; los términos de la expresión de valor no podrán contener variables de serie de pruebas ni variables de casos de prueba. El valor tomará el valor de un elemento del tipo indicado en la columna de tipo. La especificación de un valor inicial es optativa.

Esta información se proporcionará con el formato del formulario siguiente.

Declaraciones de variables de casos de prueba			
Grupo : *[TC_VarGroupReference]*			
Nombre de la variable	**Tipo**	**Valor**	**Comentarios**
⋮ *TC_VarIdentifier* ⋮	⋮ *Type* ⋮	⋮ *[ConstantExpression]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 18: Declaraciones de variables de casos de prueba

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

NOTA – Cuando se utilicen variables de casos de prueba como variables locales en un paso de prueba hay que tomar precauciones para evitar la aparición de contradicciones en la utilización con otros pasos de prueba o variables de casos de prueba. Un especificador de serie de pruebas puede evitar tales problemas mediante la adopción de una convención de denominación, que haga que todas esas variables resulten denominadas de forma unívoca dentro de una serie de pruebas.

11.8.4 Acotación de variables de casos de prueba

Inicialmente, las variables de casos de prueba son no acotadas. Podrán devenir acotadas (o ser reacotadas), en los siguientes contextos:

a) en el punto de declaración, si se especifica un valor inicial;

b) cuando aparezca la variable de caso de prueba en el lado izquierdo de un enunciado asignación (véase 15.10.4).

Una vez que una variable de caso de prueba haya sido acotada a un valor, dicha variable mantendrá ese valor hasta que sea acotada a un valor diferente o termine la ejecución del caso de prueba, lo que ocurra primero. A la conclusión del caso de prueba, las variables de casos de pruebas serán reacotadas a su valor inicial, si se ha especificado; en caso contrario deviene no acotada.

Si una variable de caso de prueba no acotada se utiliza en el lado derecho de una asignación, se produce un error de caso de prueba.

11.9 Declaraciones de tipos de PCO

Esta parte de la ATS reseña el conjunto de fronteras de servicio en que están ubicados los puntos de control y observación (PCO)

Para cada tipo de PCO utilizado en la serie de pruebas se facilitará la siguiente información:

a) nombre,

que es el que se usa para identificar la frontera de servicio donde está ubicado el PCO;

b) cometido,

que se declara como probador superior (UT, *upper tester*) o probador inferior (LT, *lower tester*) en la columna "Cometido"; o por un texto descriptivo en la columna "Comentarios"; el identificador predefinido **UT** indica que el punto PCO es un punto PCO de probador superior y el identificador **LT** especifica que se trata de un punto PCO de probador inferior; si se utiliza la columna de Cometido, su contenido será coherente con el cometido, si existe, dado en el cuadro de declaraciones de PCO.

NOTA – En una serie de pruebas que utiliza la concurrencia, puede ser necesario describir el cometido de un tipo de PCO en términos de la naturaleza del componente de prueba y el proveedor de servicio subyacente que ha de enlazarse mediante puntos PCO de este tipo.

Esta información se proporcionará con el formato del formulario siguiente.

Declaraciones de tipos de PCO		
Grupo : *[PCO_GroupReference]*		
Tipo de PCO	**Cometido**	**Comentarios**
⋮ *PCO_TypeIdentifier* ⋮	⋮ *[PCO_Role]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*		

Formulario 19: Declaraciones de tipos de PCO

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

11.10 Declaraciones de PCO

Esta parte de la ATS reseña el conjunto de puntos de control y observación (PCO) que han de utilizarse en una serie de pruebas y explica dónde existen tales PCO en un entorno de prueba.

NOTA 1 – El número de PCO debe ser, cuando es aplicable, el definido en la Recomendaciones UIT-T X.290 y UIT-T X.291, para el método o métodos de prueba indicados en el cuadro de estructura de series de pruebas. En la TTCN, también se pueden usar los PCO de formas no descritas en la Rec. UIT-T X.291, por ejemplo para comunicar con partes del sistema de pruebas o entorno de pruebas no definidas en la serie de pruebas (por ejemplo, manipular frecuencias o simular conmutaciones de transferencia en la comprobación de protocolos de radio).

NOTA 2 – Los enunciados de comportamiento en TTCN especificados para su ejecución en el PCO del UT no impondrán requisitos adicionales a los especificados por la Rec. UIT-T X.291.

En la TTCN, el modelo de PCO se basa en colas del tipo "primero en entrar, primero en salir" (FIFO, *first in first out*):

- una cola de salida, para el envío de las ASP y/o PDU;
- una cola de entrada, para la recepción de las ASP y/o PDU.

Se supone que la cola de salida está situada en el proveedor del servicio subyacente o, en el caso de las UT, en la IUT.

Un evento SEND (enviar) en el PCO discurre con éxito cuando se transfiere del LT al proveedor del servicio o del UT a la IUT.

El probador tiene una cola de entrada, con el fin de recibir eventos. Todos los eventos de entrada son puestos en cola y procesados por el probador en el mismo orden en que fueron recibidos, sin pérdida de ningún evento.

NOTA 3 – El modelo de cola es tan solo un modelo abstracto y no se pretende que implique una implementación específica.

Para cada PCO utilizado en la serie de pruebas se facilitará la siguiente información:

a) nombre,

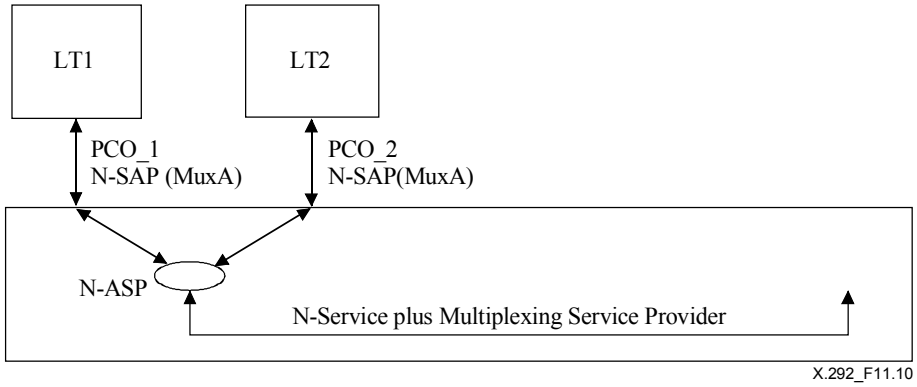
   utilizado en las descripciones de comportamiento para especificar dónde se producen los eventos particulares;

b) tipo,

   el declarado en los cuadros Declaración de tipo PCO, y que puede ir seguido en caso necesario de información relacionada con los requisitos de multiplexación que deben cumplirse inmediatamente debajo de este PCO pero encima de la frontera del servicio; si la actividad en dos o más PCO ha de ser multiplexada en conjunto por el proveedor del servicio (por ejemplo, sobre un punto extremo de conexión único), en las declaraciones de PCO para estos PCO, el tipo de PCO deberá ir seguido del mismo MuxValue (es decir, un parámetro serie de pruebas) indicado entre paréntesis; el significado preciso de este parámetro serie de pruebas se especificará en la PIXIT pertinente.

   NOTA 4 – Para una explicación más detallada de MuxValue, véase F.11.

EJEMPLO 24 – Utilización de MuxValue:



c) Cometido,

que se puede omitir si está especificado en el cuadro de declaraciones de tipo de PCO para cada uno de los tipos de PCO utilizados; si el cometido no está especificado en el cuadro de declaraciones de tipos de PCO, entonces se declarará, bien como UT o LT en la columna de Cometido, o por medio de un texto descriptivo en la columna de Comentarios; el identificador predefinido **UT** indica que se trata de un PCO de probador superior y **LT** especifica un PCO de probador inferior. Si se utiliza la columna de cometido, entonces su contenido será coherente con el cometido, si existe, dado en el cuadro de declaraciones de tipos de PCO.

NOTA 5 – En una serie de pruebas que utiliza la concurrencia, puede ser necesario describir el cometido de un PCO en términos de la naturaleza del componente de prueba y mediante este PCO enlazar al proveedor de servicio subyacente.

Esta información se proporcionará con el formato del formulario siguiente.

Declaraciones de PCO			
Grupo : *[PCO_GroupReference]*			
Nombre del PCO	**Tipo**	**Cometido**	**Comentarios**
⋮ *PCO_Identifier* ⋮	⋮ *PCO_TypeIdentifier [(MuxValue)]* ⋮	⋮ *[PCO_Role]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 20: Declaraciones de PCO

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

EJEMPLO 25 – Declaraciones de PCO:

Declaraciones de PCO			
Nombre del PCO	**Tipo de PCO**	**Cometido**	**Comentarios**
L	TSAP	LT	Punto de acceso al servicio de transporte en el probador inferior.
U	SSAP	UT	Punto de acceso al servicio de sesión en el probador superior.

Normalmente, los puntos de control y observación son simplemente SAP pero, de una manera general, pueden ser cualesquiera puntos apropiados en los que sea posible controlar y observar los eventos de prueba. Se puede, no obstante, definir un PCO de forma que corresponda a un *conjunto* de SAP, siempre que la totalidad de los puntos de acceso al servicio (SAP, *service access point*) que constituyen ese PCO:

– estén en el mismo lugar (esto es, en el LT o en el UT);

– sean SAP del mismo servicio.

Cuando un PCO corresponda a varios SAP, se utiliza una dirección apropiada para identificar cada SAP individual. Normalmente, los PCO se asocian con un punto de acceso al servicio del (N – 1) proveedor de servicio o la IUT.

NOTA 6 – Puede suceder que un PCO no esté relacionado con ningún SAP. Tal sería el caso cuando una capa estuviera constituida por subcapas (por ejemplo, en la capa de aplicación o en capas inferiores, donde un punto de adjunción de subred no es un SAP).

11.11 Declaraciones de CP

Los puntos de coordinación (CP) se utilizan para facilitar el intercambio de mensajes de coordinación (CM) entre componentes de prueba. Los puntos de coordinación se modelan como dos colas, una para cada dirección de comunicación (véase la figura 5). A este respecto los puntos de coordinación son similares a los PCO (véase la figura 3). Los CP se diferencian de los CPO en que los CP conectan dos componentes de prueba, mientras que los PCO conectan un componente de prueba con el entorno exterior, normalmente la IUT o un proveedor del servicio.

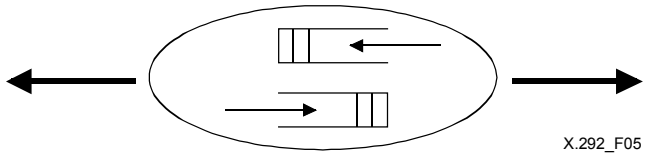


Figura 5/X.292 – Modelo de CP

Los puntos de coordinación se pueden realizar por una comunicación local o por una comunicación que atraviesa fronteras físicas.

A través de los CP la comunicación es asíncrona, esto es, la comunicación es efectuada por un componente de prueba que envía un mensaje de coordinación a su colateral recibe el mensaje de coordinación cuando está preparado. El componente de prueba que inició el CM continúa, sin embargo, su ejecución inmediatamente después del envío del CM. Si se necesita que el componente de prueba emisor suspenda su actividad hasta que el mensaje de coordinación se haya recibido, un especificador de serie de pruebas debe utilizar un mecanismo de toma de contacto. En la figura 6 se muestra un ejemplo de especificación de esta toma de contacto.

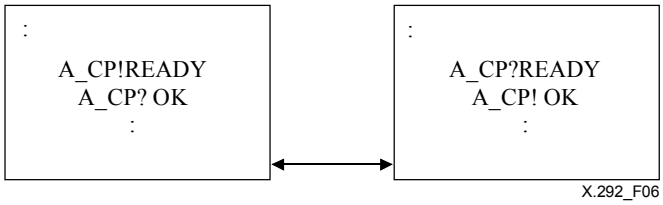


Figura 6/X.292 – Ejemplo de una toma de contacto simple

Todos los CP deberán ser declarados. Cada CP tendrá un nombre exclusivo dentro de la serie de pruebas.

Esta información se proporcionará con el formato del formulario siguiente.

Declaraciones de CP	
Grupo : *[CP_GroupReference]*	
Nombre del CP	**Comentarios**
⋮ *CP_Identifier* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*	

Formulario 21: Declaraciones de CP

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

11.12 Declaraciones de temporizador

Una serie de pruebas puede utilizar temporizadores. Para cada temporizador, se facilitará la siguiente información:

a) el nombre del temporizador;

b) la duración del temporizador optativo,

la duración por defecto del temporizador es una expresión que podrá omitirse si no se puede establecer el valor antes de la ejecución de la serie de pruebas. Los términos de la expresión de valor no contendrán variables de series de pruebas ni variables de casos de prueba; la duración del temporizador tomará a un valor INTEGER positivo;

c) la unidad de tiempo,

que será una de las siguientes:

1) **ps** (picosegundo);
2) **ns** (nanosegundo);
3) **µs** (microsegundo);
4) **ms** (milisegundo);
5) **s** (segundo);
6) **min** (minuto).

Dentro de la misma serie de pruebas, temporizadores diferentes pueden utilizar unidades diferentes. Si existe una entrada PICS o PIXIT, la declaración del temporizador especificará las mismas unidades que figuran en la entrada PICS/PIXIT.

Esta información se proporcionará con el formato del formulario siguiente.

Declaraciones de temporizador			
Grupo : *[TimerGroupReference]*			
Nombre del temporizador	**Duración**	**Unidad**	**Comentarios**
⋮ *TimerIdentifier* ⋮	⋮ *[ConstantExpression]* ⋮	⋮ *TimeUnit* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 22: Declaraciones de temporizador

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

Cada componente de prueba obtiene una copia reciente de todos los temporizadores cuando arranca la ejecución de su comportamiento.

EJEMPLO 26 – Declaración de temporizador:

Declaraciones de temporizador			
Nombre del temporizador	**Duración**	**Unidad**	**Comentarios**
wait	15	s	Espera de finalidad general
no_response	A	min	Se utiliza para esperar a que la IUT se conecte o reaccione al establecimiento de la conexión. Duración mayor que la de la espera de finalidad general
delay_time		ms	Su duración se establecerá durante la ejecución de la serie de pruebas

11.13 Componentes de prueba y declaraciones de configuración

11.13.1 Componentes de prueba

11.13.1.1 Componente de prueba principal

El componente de prueba principal realiza la función de control de probador inferior (LTCF, *lower tester control function*), tal como se define en 11.5.2/X.291. Su comportamiento se describe en el primer árbol del cuadro de descripción de caso de prueba y en todos los árboles adjuntados al mismo. Se ocupa de:

a) la creación de todos los PTC necesarios dentro de la configuración vigente y la supervisión de su terminación;

b) la gestión de los CP existentes entre el MTC y los PTC;

c) el cálculo y la asignación del veredicto de prueba con su conocimiento del efecto combinado de los resultados preliminares procedentes de los PTC.

Además, un componente de prueba principal puede gestionar uno o varios PCO.

Solamente el componente de prueba principal utilizará directamente las variables de serie de pruebas. Las variables de serie de pruebas se pueden transferir a los PTC en el constructivo CREATE. Se asignan valores a los parámetros para impedir los efectos secundarios.

11.13.1.2 Componentes de prueba paralelos

Los componentes de prueba paralelos realizan el cometido de los probadores inferiores o los probadores superiores. Su comportamiento se describe en el árbol referenciado en un enunciado CREATE en el MTC, y en todos los árboles adjuntados al mismo. Un PTC asigna resultados preliminares pero no asigna veredictos de prueba.

Un PTC no deberá:

a) utilizar variables de serie de pruebas;

b) crear otros componentes de prueba.

11.13.1.3 Declaraciones de componentes de prueba

Si se utiliza la TTCN concurrente, esta sección de la ATS deberá declarar todos los componentes de prueba que se están con. Estos componentes de prueba son posteriormente referenciados a partir de las declaraciones de configuraciones de componentes de prueba que definen configuraciones específicas.

Para cada componente de prueba se proporcionará la información siguiente:

a) nombre,

que será exclusivo a lo largo de la serie de pruebas;

b) cometido,

que indicará si el componente de prueba es el componente de prueba principal o un componente de prueba paralelo, y donde al menos un componente de prueba será un componente de prueba principal, y al menos un componente de prueba será un componente de prueba paralelo;

c) número de PCO utilizados,

donde cero o más PCO pueden estar asociados con el componente de prueba;

d) número de CP utilizados,

donde cero o más CP pueden estar asociados con el componente de prueba.

Esta información se proporcionará con el formato del formulario siguiente.

Declaraciones de componentes de prueba				
Grupo : *[TcompGroupReference]*				
Nombre del componente	**Cometido del componente**	**N.º de PCO**	**N.º de CP**	**Comentarios**
⋮ *TcompIdentifier* ⋮	⋮ *TCompRole* ⋮	⋮ *Num_PCOs* ⋮	⋮ *Num_CPs* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*				

Formulario 23: Declaraciones de componentes de prueba

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

EJEMPLO 27 – Declaración de componentes de prueba:

Este cuadro de declaraciones de componentes de prueba se puede utilizar junto con las configuraciones de componentes de prueba CONFIG1 y CONFIG2, que se presentan en las figuras 3 y 4 y se declaran en el ejemplo 28 y el ejemplo 29.

Declaraciones de componentes de prueba				
Nombre del componente	**Cometido del componente**	**N.º de PCO**	**N.º de CP**	**Comentarios**
MTC1	MTC	0	3	Se utiliza en Config 1
MTC2	MTC	1	2	Se utiliza en Config 2, con un PCO
TC1	PTC	1	2	Se utiliza en Config 1
TC2	PTC	1	3	Se utiliza en Config 1 y Config 2
TC3	PTC	1	2	Se utiliza en Config 1
TC4	PTC	0	3	Se utiliza en Config 2
TC5	PTC	1	0	Se utiliza en Config 2, sin un CP

11.13.2 Declaraciones de configuraciones de componentes de prueba

Los componentes de prueba se utilizan para construir una arquitectura o configuración lógica que facilite la ejecución concurrente de árboles de comportamiento dinámico en la TTCN. Debe declararse cada configuración de componentes de prueba que se utilice en un caso de prueba abstracta que emplea concurrencia.

Para cada configuración de componentes de prueba se proporcionará la siguiente información:

a) nombre,

que será exclusivo dentro de la serie de pruebas, y deberá referenciarse a partir de un encabezamiento del cuadro de comportamiento dinámico de casos de prueba;

b) una lista de los componentes de prueba pertenecientes a la configuración de prueba,

que debe proporcionar para cada componente de prueba la información siguiente:

1) nombre,

que habrá sido declarado como nombre de componente de prueba. Solo uno exactamente de los componentes de prueba de la configuración deberá declarase como MTC.

2) PCO utilizados,

donde se asocia con cada componente de prueba una lista de cero o más PCO declarados. El número de PCO de la lista será el mismo que el número de PCO declarados en la declaración de componentes de prueba pertinente. En una configuración única, ningún PCO se utilizará más de una vez (es decir, los componentes de prueba en una configuración no deberán compartir puntos PCO).

3) CP utilizados,

donde se asocia con cada componente de prueba una lista de cero o más CP declarados. El número de CP en la lista de un PTC será el mismo que el número de CP declarados en la declaración de componentes de prueba pertinente. El número de CP en la lista de un MTC no será superior al número de CP declarados. Ningún nombre de CP deberá aparecer más de una vez en cada lista de CP. Cada nombre de CP en la lista de un componente de prueba deberá aparecer en la lista de exactamente otro componente de prueba de la configuración. En otras palabras, cada nombre de CP utilizado en la configuración aparecerá exactamente dos veces en el cuadro de configuración. Estas parejas de CP se utilizan para especificar la conectividad de los componentes de prueba de la configuración.

Esta información se proporcionará con el formato del formulario siguiente.

Declaración de configuración de componentes de prueba			
Nombre de la configuración : *TCompConfigIdentifier*			
Grupo : *[TCompConfigGroupReference]*			
Comentarios : *[FreeText]*			
Componentes utilizados	**PCO utilizados**	**CP utilizados**	**Comentarios**
⋮ *TcompIdentifier* ⋮	⋮ *[PCO_List]* ⋮	⋮ *[CP_List]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 24: Declaración de configuración de componentes de prueba

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

EJEMPLO 28 – Declaración de configuración de componentes de prueba correspondiente a la figura 3:

Declaración de configuración de componentes de prueba		
Nombre de la configuración : CONFIG_1		
Componentes utilizados	**PCO utilizados**	**CP utilizados**
MTC1		MCP1, MCP2, MCP3
TC1	PCO_A	MCP1, CP1
TC2	PCO_B	MCP2, CP1, CP2
TC3	PCO_C	MCP3, CP2

EJEMPLO 29 – Declaración de configuración de componentes de prueba correspondiente a la figura 4:

Declaración de configuración de componentes de prueba		
Nombre de la configuración : CONFIG_2		
Componentes utilizados	**PCO utilizados**	**CP utilizados**
MTC2	PCO_D	MCP2, MCP3
TC2	PCO_B	MCP2, CP1, CP2
TC4		MCP3, CP1, CP2
TC5	PCO_E	

11.14 Definiciones de tipos de ASP

11.14.1 Introducción

La finalidad de esta parte de la serie de pruebas con TTCN abstractas es declarar los tipos de ASP que pueden enviarse o recibirse en los PCO declarados. Las definiciones de tipo de ASP pueden incluir definiciones de tipo ASN.1, si procede.

11.14.2 Definiciones de tipos de ASP mediante cuadros

Para cada ASP se facilitará la siguiente información:

a) nombre,

   se utilizará el nombre completo como figura en las normas sobre protocolo apropiadas. Si se emplea una abreviatura, deberá seguir el nombre completo entre paréntesis;

b) el tipo de PCO asociado con la ASP,

   Si dentro de una serie de pruebas se define un solo PCO, la especificación del tipo de PCO en la definición de tipo de ASP es optativa;

c) una lista de los parámetros asociados con la ASP,

   debiendo proporcionarse la siguiente información para cada parámetro:

   1) nombre,

      que podrá ser:

      – el nombre completo que figura en la norma del protocolo pertinente. Si se emplea una abreviatura, deberá seguir el nombre completo entre paréntesis; o

      – el símbolo macro (<-), para indicar que la entrada en la columna de tipo identifica un conjunto de parámetros que ha de insertarse directamente en la lista de parámetros de ASP. El símbolo macro se utilizará solamente con tipos estructurados definidos en las definiciones de tipos estructurados;

   2) su tipo y un atributo optativo,

      cuyos parámetros pueden ser un tipo de estructura arbitrariamente compleja, incluido el especificado como tipo de serie de pruebas (ya sea predefinido, tipo simple, tipo estructurado o tipo ASN.1). Si un parámetro ha de estructurarse como PDU, su tipo podrá enunciarse en cualquiera de las formas siguientes:

      – como un identificador de PDU, para indicar que, en la constricción para la ASP, este parámetro puede encadenarse a una constricción de PDU de un tipo de PDU específico; o

      – como una **PDU**, para indicar que en la constricción para la ASP, este parámetro puede encadenarse a una constricción de PDU de cualquier tipo de PDU; y cuyo atributo optativo es longitud;

      en cuyo caso, la especificación puede restringir el parámetro a una longitud o gama particular, de acuerdo con 11.18. Los valores de longitud se interpretarán de conformidad con el cuadro 5, en 11.18. Los límites se especificarán en términos de literales INTEGER no negativos, parámetros de serie de pruebas, constantes de series de pruebas o la palabra clave INFINITY.

      No habrá contradicción entre las especificaciones de longitud definidas para el tipo de parámetro ASP en las definiciones de tipo de serie de pruebas y las especificaciones de longitud en la definición de tipo de ASP, esto es, el conjunto de cadenas definidas por una restricción de longitud en una

definición de ASP será un verdadero subconjunto del conjunto de cadenas definidas por la definición de tipo de serie de pruebas.

Se puede utilizar la palabra clave INFINITY, como valor de límite superior, para indicar que la longitud no tiene límite superior.

NOTA – Normalmente no es necesario restringir la longitud de los parámetros de ASP, pero en algunos casos, quizás haga falta para restringir efectivamente la longitud de un campo de PDU correspondiente en un protocolo subyacente.

Se considera que los parámetros de las definiciones de tipos de ASP son optativos, es decir, que en instancias de estos tipos pueden no estar presentes parámetros completos.

Esta información se proporcionará con el formato del formulario siguiente.

Definición de tipo de ASP		
Nombre de la ASP : *ASP_Id&FullId*		
Grupo : *[ASP_GroupReference]*		
Tipo de PCO : *[PCO_TypeIdentifier]*		
Comentarios : *[FreeText]*		
Nombre del parámetro	**Tipo de parámetro**	**Comentarios**
⋮ *ASP_ParIdOrMacro* ⋮	⋮ *Type&Attributes* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*		

Formulario 25: Definición de tipo de ASP

Las columnas de nombre del parámetro y tipo de parámetro estarán presentes o ausentes a la vez.

EJEMPLO 30 – Primitiva de servicio abstracta, petición T_CONEXIÓN:

En la figura que sigue, se muestra un ejemplo del servicio de transporte [ISO/CEI 8072]. Podría ser parte del conjunto de ASP utilizadas para describir el comportamiento de un UT abstracto en una serie de pruebas DS para el transporte de clase 0. CDA, CGA y QOS son tipos de series de pruebas [ISO/CEI 8073].

Definición de tipo de ASP		
Nombre de la ASP : CONreq (T_CONNECTrequest)		
Tipo de PCO : *TSAP*		
Comentarios :		
Nombre del parámetro	**Tipo de parámetro**	**Comentarios**
Cda (Called Address, dirección de llamada)	CDA	… del probador superior
Cga (Calling Address, dirección de llamada)	CGA	… del probador inferior
QoS (Quality of Service, calidad de servicio)	QoS	deberá asegurar que se utiliza la clase cero
Comentarios detallados: ASP a enviar al punto de acceso al servicio de transporte.		

11.14.3 Utilización de tipos estructurados en las definiciones de tipo de ASP

Hay dos relaciones posibles entre un tipo estructurado y las definiciones de ASP relativas al mismo, a saber:

a) Si en la definición figura un nombre de parámetro, el tipo estructurado referenciado es una subestructura. Esto permite la definición de ASP que contengan una subestructura multinivel de parámetros.

b) Si se utiliza el símbolo macro (<-) en vez de un nombre de parámetro, ello equivale a una expansión de macro. La entrada en la definición de tipo de ASP se expande directamente a una lista de parámetros sin la introducción de un nivel adicional de subestructura.

No se utilizará el símbolo macro en la misma línea que la de referencias a los tipos definidos en ASN.1 tipos simples, es decir, sólo tipos estructurados definidos en forma tabular pueden ser expandidos a otros tipos estructurados como expansiones de macro.

11.14.4 Definiciones de tipos de ASP con ASN.1

Donde sea más apropiado, se pueden especificar las ASP empleando la ASN.1. Esto se logrará mediante una definición ASN.1 que utilice la sintaxis ASN.1 definida en la Rec. UIT-T X.680. Para cada ASP en ASN.1 se proporcionará la siguiente información:

a) nombre,

   se utilizará el nombre completo como figura en la norma del protocolo pertinente. Si se emplea una abreviatura deberá seguir el nombre completo entre paréntesis;

b) el tipo de PCO asociado con la ASP,

   si dentro de una serie de pruebas se define un solo PCO, la especificación del tipo de PCO en la definición de tipo de ASP es optativa;

c) la definición de tipo de ASP en ASN.1,

   que seguirá la sintaxis definida en la Rec. UIT-T X.680. En los identificadores internos a esa definición no se utilizará el símbolo (-). En su lugar podrá emplearse el símbolo de subrayado (_). El identificador de ASP en el encabezamiento del cuadro es el nombre del primer tipo definido en el cuerpo del cuadro.

   Los tipos a que hace referencia la definición de la ASP deberán estar definidos en otros cuadros de definición de tipo en ASN.1, definidos por referencia en el cuadro de referencias del tipo ASN.1 o definidos localmente en el mismo cuadro, a continuación de la primera definición de tipo. Los tipos definidos localmente no se utilizarán en otras partes de la serie de pruebas.

   Se puede hacer comentarios en ASN.1 en el cuerpo del cuadro. En el cuadro no estará presente la columna de comentarios.

Los Comentarios en ASN.1 comienzan con "--" y terminan con la siguiente aparición de "--" o con "end of line" (fin de línea), el primero que aparezca. Esto impide que un sólo comentario en ASN.1 ocupe varias líneas. Para facilitar el intercambio de series ATS en TTCN.MP, se recomienda utilizar especificadores de ATS, y encerrar siempre entre "--" los comentarios en ASN.1.

Esta información se proporcionará con el formato del formulario siguiente.

Definición de tipo de ASP en ASN.1	
Nombre de la ASP	**:** *ASP_Id&FullId*
Grupo	**:** *[ASN1ASP_GroupReference]*
Tipo de PCO	**:** *[PCO_TypeIdentifier]*
Comentarios	**:** *[FreeText]*
Definición de tipo	
ASN1_Type&LocalTypes	
Comentarios detallados:	*[FreeText]*

Formulario 26: Definición de tipo de ASP en ASN.1

11.14.5 Definiciones de tipos de ASP en ASN.1 por referencia

Las ASP se pueden especificar mediante una referencia precisa a una ASP en ASN.1 definida en una norma OSI o mediante referencia a un tipo ASN.1 definido en un módulo ASN.1 adjuntado a una serie de pruebas. Para cada ASP se facilitará la siguiente información:

a) nombre,

   que se utilizará a lo largo de toda la serie de pruebas;

b) el tipo de PCO asociado con la ASP,

   si dentro de una serie de pruebas se define un solo PCO, la especificación del tipo de PCO en la definición de tipo de ASP es optativa;

c) la referencia a tipo,

   que seguirá las reglas de identificador establecidas en la Rec. UIT-T X.680;

d) el identificador del módulo,

   que consiste en una referencia a módulo que seguirá las reglas de identificador establecidas en la Rec. UIT-T X.680 y un ObjectIdentifier optativo.

La información se proporcionará con el formato del formulario siguiente.

Definiciones de tipo de ASP en ASN.1 por referencia				
Grupo : *[ASN1ASP_GroupReference]*				
Nombre de la ASP	**Tipo de PCO**	**Referencia a tipo**	**Identificador de módulo**	**Comentarios**
⋮ *ASP_Id&FullId* ⋮	⋮ *[PCO_TypeIdentifier]* ⋮	⋮ *TypeReference* ⋮	⋮ *ModuleIdentifier* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*				

Formulario 27: Definiciones de tipo de ASP en ASN.1 por referencia

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

Las referencias de tipos de identificadores en ASN.1 y las referencias de valor pueden contener guiones. Para poder utilizar definiciones importadas en la TTCN es preciso cambiar los guiones por caracteres de subrayado (véase 11.2.3.5).

11.15 Definiciones de tipos de PDU

11.15.1 Introducción

La finalidad de esta parte de la serie de pruebas con TTCN abstractas es declarar los tipos de las PDU que pueden enviarse o recibirse ya sea directamente o insertadas en las ASP de los PCO declarados. Las definiciones de tipo de PDU pueden incluir definiciones de tipo ASN.1, si procede. Las definiciones de PDU definen el conjunto de PDU intercambiadas con la IUT que son sintácticamente válidas con respecto a la ATS, pero no necesariamente válidas con respecto a la especificación de protocolo.

Es preciso declarar todos los campos de las PDU que están definidos en la norma del protocolo pertinente tanto explícita como implícitamente, haciendo referencia a reglas de codificación (reglas de codificación ASN.1, si son aplicables).

La codificación de los campos de las PDU se ajustará a la definida en la especificación de protocolos pertinente, salvo que se incluya información de codificación en la serie de pruebas.

11.15.2 Definición de tipos de PDU mediante cuadros

Las definiciones de PDU son similares a las de ASP. Para cada PDU se facilitará la siguiente información:

a) nombre,

   se utilizará el nombre completo como figura en la norma del protocolo pertinente. Si se emplea una abreviatura deberá seguir el nombre completo entre paréntesis;

b) el tipo de PCO asociado con la PDU,

si una PDU solamente se envía o se recibe insertada en las ASP, en la totalidad de la serie de pruebas, la especificación de las PCO es optativa. Si en una serie de pruebas se define solamente un PCO único, la especificación del tipo de PCO en la definición de tipo de PDU, es optativa;

c) las reglas de codificación que se utilizarán para una PDU de este tipo.

con el fin de especificar las codificaciones explícitas para las PDU completas, que sustituyan a las reglas de codificación global por defecto de la serie de pruebas en conjunto, esta entrada optativa deberá hacer referencia a una entrada en el cuadro de definiciones de codificación pertinente (por ejemplo, cambiar de BER a DER). Si no se utiliza esta entrada, se aplican entonces las reglas de codificación global por defecto. Véase 11.16.4;

d) las variaciones de codificación que se utilizarán para PDU de este tipo;

con el fin de especificar las variaciones de codificación explícitas para las PDU completas, que sustituyen a las variaciones de codificación global por defecto de la serie de pruebas en conjunto, esta entrada optativa deberá hacer referencia a una entrada en el cuadro de variaciones de codificación pertinente [por ejemplo, cambiar de SD a LD(3)]. Si no se utiliza esta entrada, se aplican entonces las variaciones de codificación global por defecto. Véase 11.16.4;

e) lista de los campos asociados con la PDU,

debiendo proporcionarse la siguiente información para cada campo:

1) nombre,

que podrá ser:

– el nombre completo que figura en la norma del protocolo pertinente. Si se emplea una abreviatura, deberá seguir el nombre completo entre paréntesis; o

– el símbolo macro (<-), para indicar que la entrada en la columna de tipo identifica un conjunto de campos que han de insertarse directamente en la lista de campos de la PDU. El símbolo macro se utilizará solamente con los tipos estructurados definidos en las definiciones de tipo estructurado;

2) su tipo y un atributo optativo,

pudiendo ser los campos de un tipo de estructura arbitrariamente compleja, incluso ser especificados como tipo de serie de pruebas (ya sea tipo predefinido, tipo simple, tipo estructurado o tipo ASN.1). Si un campo ha de estructurarse como PDU, podrá establecerse su tipo de cualquiera de las formas siguientes:

– como un identificador de PDU, para indicar que en la constricción aplicable a la PDU, este campo puede encadenarse a una constricción PDU de un tipo de PDU específico; o

– como una **PDU**, para indicar que en la constricción aplicable a la PDU, este campo puede encadenarse a una constricción PDU de cualquier tipo de PDU;

y siendo el atributo optativo longitud;

en cuyo caso la especificación puede restringir el campo a una longitud particular o a una gama de acuerdo con 11.18. Los valores se interpretarán longitud de conformidad con el cuadro 5, en 11.18. Los límites se especificarán en términos de literales INTEGER no negativos, parámetros de series de pruebas, constantes de series de pruebas o la palabra clave INFINITY.

No habrá contradicción entre las especificaciones de longitud definidas para el tipo de campo de PDU en las definiciones de tipos de series de pruebas y las especificaciones de longitud en la definición de tipo de PDU, es decir, el conjunto de cadenas definidas por una restricción de longitud en una definición de PDU será un verdadero subconjunto del conjunto de cadenas definidas por la definición de tipo de serie de pruebas.

Se puede utilizar la palabra clave INFINITY, como valor del límite superior, para indicar que la longitud no tiene límite superior;

3) optativamente, un identificador de codificación específico seguido de alguna lista de parámetros reales necesarios, con el fin de especificar codificaciones explícitas para campos individuales de una PDU, que sustituirán a las reglas de codificación y a las variaciones de codificación aplicables a la PDU en su conjunto. Este identificador de codificación, si existe, deberá identificar una de las variaciones de codificación o una definición de codificación de campo no válida definida en la serie de pruebas (por ejemplo, LD(10)); véase 11.16.4.

Se considera que los campos de las definiciones de tipos de PDU son optativos, es decir, que en instancias de estos tipos pueden no estar presentes campos completos.

Esta información se proporcionará con el formato del formulario siguiente.

Definición de tipo de PDU			
Nombre de la PDU : *PDU_Id&FullId*			
Grupo : *[PDU_GroupReference]*			
Tipo de PCO : *[PCO_TypeIdentifier]*			
Nombre de la regla de codificación : *[EncodingRuleIdentifier]*			
Variación de codificación : *[EncVariationCall]*			
Comentarios : *[FreeText]*			
Nombre del campo	**Tipo de campo**	**Codificación del tipo**	**Comentarios**
⋮ *PDU_FieldIdOrMacro* ⋮	⋮ *Type&Attributes* ⋮	⋮ *[PDU_FieldEncodingCall]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 28: Definición de tipo de PDU

Las columnas Nombre del campo y Tipo del campo deberán estar presentes o ausentes a la vez.

EJEMPLO 31 – Definición de tipo de PDU típica:

Definición de tipo de PDU		
Nombre de la PDU : *INTC (interrupt Confirm)* **Tipo de PCO** : *NSAP*		
Nombre del campo	**Tipo de campo**	**Comentarios**
GFI	BITSTRING	Identificador de formato general
LCGN	BITSTRING	Número de grupo de canales lógicos
LCN	BITSTRING	Identificador de canal lógico
PTI	OCTETSTRING	Identificador de tipo de paquete
EXTRA	OCTETSTRING	Para crear paquetes INTC largos

11.15.3 Utilización de tipos estructurados dentro de las definiciones de PDU

Hay dos relaciones posibles entre un tipo estructurado y las definiciones de PDU que se refieren al mismo, a saber:

a) Si en la definición se da un nombre de campo, el tipo estructurado referenciado es una subestructura. Esto permite la definición de PDU, que contengan una estructura multinivel de campos.

b) Si se utiliza el símbolo macro (<-) en lugar del nombre de campo, ello equivale a una expansión de macro. La entrada en la definición de tipo de la PDU se expande directamente a una lista de campos sin la introducción de un nivel adicional de subestructura.

El símbolo macro no se utilizará en la misma línea que las referencias a tipos definidos en ASN.1 o tipos simples; es decir, solamente los tipos estructurados definidos de forma tabular pueden ser expandidos a otros tipos estructurados como expansiones de macro.

11.15.4 Definiciones de tipo de PDU con ASN.1

Donde sea más apropiado, se pueden especificar PDU en ASN.1. Esto se consigue mediante una definición ASN.1 que emplee la sintaxis ASN.1 definida en la Rec. UIT-T X.680. Para cada PDU de tipo ASN.1 se facilitará la siguiente información:

a) nombre,

que se utilizará completo como figura en la norma del protocolo pertinente. Si se emplea una abreviatura deberá seguir el nombre completo entre paréntesis;

b) tipo de PCO asociado con la PDU,

si un PDU siempre se envía o se recibe insertado en las ASP, la especificación del tipo de PCO en la definición de tipo de la PDU es optativa. Si en una serie de pruebas se define solamente un PCO único, la especificación del tipo de PCO en la definición de tipo de la PDU es optativa;

c) las reglas de codificación que se utilizarán para una PDU de este tipo

con el fin de especificar las codificaciones explícitas para PDU completas, que sustituyan a las reglas de codificación global por defecto de la serie de pruebas en conjunto, esta entrada optativa deberá hacer referencia a una entrada en el cuadro de definiciones de codificación pertinente (por ejemplo, cambiar de BER a DER). Si no se utiliza esta entrada, se aplican entonces las reglas de codificación global por defecto. Véase 11.16.4;

d) las variaciones de codificación que se utilizarán para PDU de este tipo,

con el fin de especificar las variaciones de codificación explícitas para PDU completas, que sustituyen a las variaciones de codificación global por defecto de la serie de pruebas en conjunto, esta entrada optativa deberá hacer referencia a una entrada en el cuadro de variaciones de codificación pertinente [por ejemplo, cambiar de SD a LD(3)]. Si no se utiliza esta entrada, se aplican entonces las variaciones de codificación global por defecto. Véase 11.16.4;

e) definición de tipo de PDU en ASN.1,

que seguirá la sintaxis definida en la Rec. UIT-T X.680, excepto cuando existe la opción adicional de especificar una variación de codificación o codificación de campo no válida asociada con el ASN1_Type completo o con cualquier tipo ASN.1 dentro del ASN1_Type. Esto se lleva a cabo dando un identificador de codificación específico seguido de una lista de parámetros reales adicional, con el fin de especificar codificaciones explícitas para campos individuales u otros subtipos de una PDU, que sustituirán a las reglas de codificación y las variaciones de codificación aplicables a la PDU en conjunto; el identificador de codificación, si existe, deberá identificar una de las variaciones de codificación o una definición de codificación de campo no válida definida en la serie de pruebas (por ejemplo, LD(10)); véase 11.16.4.

En los identificadores internos a esa definición no se utilizará el símbolo guión (-). En su lugar podrá emplearse el símbolo de subrayado (_). El identificador de PDU en el encabezamiento del cuadro es el nombre del primer tipo definido en el cuerpo del cuadro.

Los tipos a que hace referencia la definición de PDU deberán estar definidos en otros cuadros de definición de tipo en ASN.1, definidos por referencia en el cuadro de referencias del tipo ASN.1 o definidos localmente en el mismo cuadro, a continuación de la primera definición de tipo. Los tipos definidos localmente se utilizarán en otras partes de la serie de pruebas.

Se puede hacer comentarios en ASN.1 en el cuerpo del cuadro. En el cuadro no estará presente la columna de comentarios.

Los Comentarios en ASN.1 empiezan con "--" y terminan con la siguiente aparición de "--" o con "end of line" (fin de línea), el primero que llegue. Esto impide que un sólo comentario en ASN.1 ocupe varias líneas. Para facilitar el intercambio de series ATS en TTCN.MP, se recomienda utilizar especificadores de ATS, encerrando siempre entre "--" los comentarios en ASN.1.

La información se proporcionará con el formato del formulario siguiente.

Definición de tipo de PDU en ASN.1		
Nombre de la PDU	**:**	*PDU_Id&FullId*
Grupo	**:**	*[ASN1_PDU_GroupReference]*
Tipo de PCO	**:**	*[PCO_TypeIdentifier]*
Nombre de la regla de codificación	**:**	*[EncodingRuleIdentifier]*
Variación de codificación	**:**	*[EncVariationCall]*
Comentarios	**:**	*[FreeText]*
Definición de tipo		
ASN1_Type&LocalTypes		
Comentarios detallados: *[FreeText]*		

Formulario 29: Definición de tipo de PDU en ASN.1

EJEMPLO 32 – Una definición ASN.1 relativa a la FTAM:

Definición de tipo de PDU en ASN.1	
Nombre de la PDU	**:** *F_INIT (F_INITIALIZE_response)*
Tipo de PCO	**:**
Comentarios	**:**
Definición de tipo	

SEQUENCE {
  state_result State_result DEFAULT success,
  action_result Action_Result DEFAULT success,
  protocol_id Protocol_Version,
  --etc. --
}	

11.15.5 Definiciones de tipo de PDU en ASN.1 por referencia

Las PDU se pueden especificar mediante una referencia precisa a una PDU en ASN.1 definida en una norma OSI o referenciando un tipo ASN.1 definido en un módulo ASN.1 adjuntado a la serie de pruebas. Los identificadores ASN.1, referencias de tipo y referencias de valor pueden contener guiones. Para posibilitar el uso de definiciones importadas en la TTCN es necesario cambiar los guiones por subrayado (véase 11.2.3.5).

Para cada PDU se facilitará la siguiente información:

a) nombre,

   que se utilizará en toda la serie de pruebas;

b) el tipo de PCO asociado con la PDU,

   si una PDU solamente se envía o se recibe insertada en las ASP dentro de la totalidad de la serie de pruebas, la especificación del tipo de PCO es optativa. Si, en una serie de pruebas, se define solamente un PCO único, la especificación del tipo de PCO en una definición de tipo de PDU es optativa;

c) la referencia a tipo,

   que deberá seguir las reglas de identificador establecidas en la Rec. UIT-T X.680.

d) el identificador de módulo,

   consistente en la referencia a módulo que deberá seguir las reglas establecidas en la Rec. UIT-T X.680 para el identificador y a un ObjectIdentifier optativo;

e) las reglas de codificación que se utilizarán para unidades PDU de este tipo,

   a fin de especificar las codificaciones explícitas para PDU enteras, que sustituyan a las reglas de codificación global por defecto para la serie de pruebas como un conjunto, esta entrada optativa deberá hacer referencia a una entrada en el cuadro de definiciones de codificación pertinente (por ejemplo, para cambiar de BER a DER). Si no se utiliza esta entrada, se aplican entonces las reglas de codificación global por defecto. Véase 11.16.4;

f) las variaciones de codificación que se han de utilizar para unidades PDU de este tipo,

   a fin de especificar variaciones de codificación explícitas para PDU enteras, que sustituyan a las variaciones de codificación global por defecto para la serie de pruebas como un conjunto, esta entrada optativa deberá hacer referencia a una entrada en el cuadro de variaciones de codificación pertinente (por ejemplo, para cambiar de SD a LD(3)). Si no se utiliza esta entrada, se aplican entonces las variaciones de codificación global por defecto. Véase 11.16.4.

La información se proporcionará con el formato del formulario siguiente.

Definiciones de tipo de PDU en ASN.1 por referencia						
Grupo : *[ASN1_PDU_GroupReference]*						
Nombre de la PDU	**Tipo de PCO**	**Referencia de tipo**	**Identificador del módulo**	**Reglas de codificación**	**Variación de codificación**	**Comentarios**
⋮ *PDU_Id&FullId* ⋮	⋮ *[PCO_TypeIdentifier]* ⋮	⋮ *TypeReference* ⋮	⋮ *ModuleIdentifier* ⋮	⋮ *[EncodingRuleIdentifier]* ⋮	⋮ *[EncVariationCall]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*						

Formulario 30: Definiciones de tipo de PDU en ASN.1 por referencia

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

11.16 Información de codificación de serie de pruebas

11.16.1 Definiciones de codificación

Para facilitar la especificación y comprobación de las reglas de codificación de un protocolo OSI, si se admite alguna flexibilidad en las reglas de codificación aplicables al protocolo, debe proporcionarse una definición de codificación. Si se facilita una definición de codificación, se da una referencia, en la ATS, a la especificación en la que se especifican las reglas de codificación. La referencia puede hacerse a la propia especificación del protocolo, o a una especificación de reglas de codificación separada. Si no se puede proporcionar esta referencia, es decir, si no están normalizadas las reglas de codificación del protocolo, no deberán probarse las reglas de codificación.

Para cada juego de reglas de codificación correspondientes al protocolo deberá proporcionarse la siguiente información:

a) el nombre de la regla de codificación, que es un identificador exclusivo que se utilizará a lo largo de la serie de pruebas para referirse a una definición de codificación;

b) la referencia a la norma pertinente que define las reglas de codificación;

c) una expresión por defecto, que identifica las reglas de codificación que se utilizarán por defecto; esta expresión por defecto deberá tomar un valor booleano y utilizará solamente Literal Values, Test Suite Parameters, y Test Suite Constants en sus términos;

d) optativamente, un comentario adicional, proporcionado en la columna Comentarios o en el área de Comentarios detallados del cuadro.

Si se puede utilizar más de un juego de reglas de codificación en un protocolo, los nombres de las reglas de codificación deberán relacionarse en la columna de regla de codificación del cuadro de definiciones de codificación. El nombre de la regla de codificación asociado con la expresión por defecto que toma el valor TRUE deberá elegirse como el conjunto por defecto para la serie de pruebas. Si más de una expresión por defecto o ninguna expresión por defecto en el cuadro de definiciones de codificación toma el valor TRUE, se tratará de un error de caso de prueba. La no especificación de ninguna expresión por defecto, es equivalente a la especificación del valor FALSE.

La información se proporcionará con el formato indicado en el formulario siguiente:

Definiciones de codificación			
Grupo : *[EncodingGroupReference]*			
Nombre de la regla de codificación	**Referencia**	**Valor por defecto**	**Comentarios**
⋮ *EncodingRuleIdentifier* ⋮	⋮ *EncodingReference* ⋮	⋮ *[ConstantExpression]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 31: Definiciones de codificación

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

Las reglas de codificación especificadas en este formulario solamente se aplican a PDU.

EJEMPLO 33 – Definiciones de codificación:

Definiciones de codificación			
Nombre de la regla de codificación	**Referencia**	**Valor por defecto**	**Comentarios**
BER	Rec. UIT-T X.690	TRUE	Reglas de codificación básica
PER	Rec. UIT-T X.690		Reglas de codificación compactada
DER	Rec. UIT-T X.690		Reglas de codificación distinguida
Comentarios detallados: *[FreeText]*			

11.16.2 Variaciones de codificación

Se pueden proporcionar variaciones admisibles de cada definición de codificación que se pueden usar en la serie de pruebas.

Para definir estas variaciones de codificación, se facilitará la siguiente información:

a) un nombre de regla de codificación, que es el nombre de las reglas de codificación identificadas en el cuadro de definiciones de codificación al que se aplica dicha variación;

b) una lista de tipos optativa, que relaciona los tipos a los que puede aplicarse esta variación de codificación; si la lista está vacía significa que las variaciones de codificación pueden aplicarse a cualquier campo de PDU. Los tipos pueden ser cualquier tipo de PDU o cualquier tipo que pueda ocurrir dentro de una PDU;

c) una lista de variaciones de codificación,

en la cual suministrarse la siguiente información para cada variación de codificación:

1) el nombre de la variación de codificación, que es un identificador exclusivo referente a una definición de codificación admitida para un tipo específico, como se recoge en la especificación pertinente de reglas de codificación;

2) una referencia, la cual se utiliza para identificar la sección de la especificación de las reglas de codificación que describe este conjunto de variaciones de codificación;

3) una expresión por defecto, que identifica la variación de codificación utilizada por defecto; esta expresión por defecto deberá tomar un valor booleano y deberá utilizar solamente Literal Values, Test Suite Parameters y Test Suite Constants en sus términos;

d) optativamente, un comentario adicional, proporcionado en la parte de Comentarios del encabezamiento del cuadro, en la columna Comentarios o en el área de comentarios detallados del cuadro.

La variación de codificación asociada con la expresión que toma el valor TRUE deberá elegirse como variación de codificación por defecto para la lista de tipos dada, si existe, o en los demás casos para todos los tipos dentro de la serie de pruebas. Si en el cuadro de variaciones de codificación más de una expresión por defecto toma el valor TRUE, se tratará de un error de caso de prueba. La no especificación de ninguna expresión por defecto para una variación de codificación, es equivalente a la especificación del valor FALSE. Si no se especifica ninguna expresión por defecto, o si todas toman el valor FALSE, se tomará como valor por defecto la primera variación de codificación.

Las variaciones de codificación se proporcionarán con el formato indicado en el formulario siguiente.

Variaciones de codificación			
Grupo : *[EncVariationGroupReference]*			
Nombre de la regla de codificación : *EncodingRuleIdentifier*			
Lista de tipos : *[TypeList]*			
Comentarios : *[FreeText]*			
Variación de codificación	**Referencia**	**Valor por defecto**	**Comentarios**
⋮ *EncVariationId&ParList* ⋮	⋮ *VariationReference* ⋮	⋮ *[ConstantExpression]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 32: Variaciones de codificación

EJEMPLO 34 – Variaciones de codificación:

Variaciones de codificación			
Nombre de la regla de codificación : BER			
Lista de tipos : Length			
Comentarios : Length se define como un tipo INTEGER.			
Variación de codificación	**Referencia**	**Valor por defecto**	**Comentarios**
SD *LD(len: INTEGER)*	6.3.3.1 6.3.3.2	*TRUE*	
Comentarios detallados:			

11.16.3 Definiciones de codificación de campo no válidas

A fin de probar a fondo las reglas de codificación, será probablemente necesario definir las variaciones ilegales de las definiciones de codificación utilizadas por el protocolo. Pueden proporcionarse definiciones de codificación de campo no válidas para cualquiera de los tipos utilizados en campos de PDU en la serie de pruebas. Una vez definida, una definición de codificación de campo no válida se puede utilizar para sustituir a la codificación normal de un valor de campo de constricción de PDU específico del mismo tipo (véase 13.4).

Se proporcionará la siguiente información relativa a una definición de codificación de campo no válida:

a) un nombre de codificación de campo no válida, identificador exclusivo que se utilizará en toda la serie de pruebas para referirse a esta definición de codificación de campo no válida, seguido de una lista de parámetros formales optativa;

b) una lista de tipos optativa, para relacionar los tipos a lo que se puede aplicar esta codificación; una lista vacía significa que la definición de codificación puede aplicarse a cualquier campo de una PDU;

c) una definición de codificación que contiene la definición de cómo se codificarán los valores,

consistente en una definición de procedimiento de la misma forma que una definición de procedimiento de una operación serie de pruebas (11.3.4), y que al ser evaluada da como resultado la evaluación de un enunciado ReturnValue (valor de devolución*)* para proporcionar el resultado de la operación, incluyendo comentarios explicativos incrustados dentro de la definición de procedimiento en lugares apropiados como texto delimitado por "/*" y "*/"; los comentarios explicativos deberán incluir un ejemplo que muestre una invocación. El resultado de la operación de codificación será una cadena de bits con un orden definido de transmisión, que será la codificación del valor pertinente;

d) optativamente, comentarios adicionales que describan la operación, proporcionados en la parte de Comentarios del encabezamiento del cuadro o en el área de Comentarios detallados del mismo cuadro.

Se recomienda el uso de definiciones de procedimientos a fin de proporcionar en la definición de las operaciones.

Si se especifica una lista de parámetros formales, los valores transferidos a la operación de codificación se utilizan para afectar a la codificación del campo de PDU. Cada parámetro formal deberá declararse como un tipo predefinido, un identificador de tipo de serie de pruebas o un identificador de tipo de PDU. Por ejemplo, puede transferirse un valor entero a una operación de codificación que calcula la longitud de un campo de PDU. El modo en que se utilizarán los parámetros transferidos a la operación deberá explicarse en la definición de la operación de codificación.

Se utilizará un formulario para cada definición de codificación de campo no válida.

Las definiciones de operación de codificación campo no válida se proporcionarán en el formulario siguiente.

Definición de codificación de campo no válida	
Grupo	**:** *[InvalidFieldEncodingGroupReference]*
Nombre de la operación	**:** *InvalidFieldEncodingId&ParList*
Tipo de resultado	**:** *[TypeList]*
Comentarios	**:** *[FreeText]*
Definición	
TS_OpProcDef	
Comentarios detallados:	*[FreeText]*

Formulario 33: Definición de codificación de campo no válida

11.16.4 Aplicación de las reglas de codificación

Las reglas de codificación especificadas en la serie de pruebas se aplican a todas las PDU enviadas o recibidas en la parte Comportamiento. Las reglas de codificación se pueden especificar para la serie de pruebas en conjunto o para declaraciones de tipo o declaraciones de constricción, como se señala en el cuadro 4. Las casillas del cuadro 4 señaladas por ✓ identifican el ámbito de aplicación permitido para cada clase de información de codificación.

Cuadro 4/X.292 – Aplicabilidad de las definiciones de codificación

		Definiciones de codificación				
		Reglas de codificación		Variaciones de codificación		Codificaciones de campo no válidas
Precedencia	**Ámbito de aplicación**	**Por defecto**	**Otra**	**Por defecto**	**Otra**	
Inferior	Serie de pruebas	✓		✓		
	Declaraciones de tipo					
	PDU		✓	✓	✓	
	Tipos estructurados o tipos ASN.1			✓	✓	
	Tipos simples o campos/ elementos de PDU			✓	✓	✓
	Declaraciones de constricción					
	PDU		✓	✓	✓	
Superior	Tipos estructurados o tipos ASN.1			✓	✓	
	Campos/ elementos de PDU			✓	✓	✓
Precedencia dentro de una fila		Inferior				Superior

Las reglas de codificación se aplicarán de conformidad con los valores de precedencia de las filas mostrados en la primera columna del cuadro 1, siendo "(4)" la prioridad más elevada y "(1)" la prioridad más baja. Dentro de cada fila el orden de precedencia es de izquierda a derecha, siendo la entrada situada más a la derecha la de precedencia superior. De este modo, las reglas de codificación de campo de constricción tienen precedencia sobre las demás, mientras que las reglas de codificación por defecto aplicadas al nivel de serie de pruebas pueden ser sustituidas por cualquiera de los otros métodos de especificación. Las reglas de codificación reales que se utilizarán para una PDU después de haberse aplicado todas las sustituciones, serán referidas como reglas de codificación aplicables.

Si no se especifica ninguna información de codificación sobre una constricción de tipo ASN.1 o estructurado, ésta hereda las reglas de codificación aplicadas en el nivel de PDU. Con ello, las reglas de codificación aplicadas a una constricción de tipo ASN.1 o estructurado pueden variar, de acuerdo con la PDU en la cual se utilizan. A la inversa, si se especifica la información de codificación en una constricción de tipo ASN.1 o estructurado, esta información sustituirá a la información de codificación de la PDU en la que se utilice. Si tal constricción de tipo ASN.1 o estructurado se utiliza en un ASP, se pasa por alto la información de codificación.

Los eventos RECEIVE en los que no se aplican reglas de codificación específicas a la PDU entrante, se pueden codificar en cualquier variación permitida por la definición de codificación aplicable (por ejemplo, cualquier forma de codificación de longitud admitida por BER).

11.17 Definiciones de tipos de CM

11.17.1 Introducción

Los parámetros de un mensaje de coordinación (CM) puede ser de cualquiera de los tipos susceptibles de especificación en TTCN. Los CM simples no pueden contener parámetros asociados o contener sólo uno, por ejemplo, un número natural, un resultado preliminar o una cadena de caracteres del tipo de "suspend" o "continue". Los CM más complejos pueden transportar información adicional, por ejemplo una PDU completa, un campo de PDU o el valor de lectura de un temporizador. No existen CM predefinidos.

11.17.2 Definiciones de tipos de CM mediante cuadros

Los tipos de CM se pueden declarar mediante cuadros de TTCN. Para cada tipo de CM se proporcionará la información siguiente:

a) nombre,

   que será único dentro de la serie de pruebas;

b) una lista de parámetros asociada con el CM,

   que deberá proporcionar la información siguiente para cada parámetro:

   1) nombre,

      que será único dentro del CM;

   2) su tipo y un atributo optativo,

      del mismo modo que en los campos de PDU,

      en cuyo caso la especificación puede restringir el campo a una longitud particular o a una gama de longitudes con arreglo a 11.18. Los valores de longitud deberá interpretarse de conformidad con el cuadro 5 en 11.18. Los límites se especificarán en términos de literales INTEGER no negativos, parámetros de serie de pruebas, constantes de series de pruebas o la palabra clave INFINITY.

      No habrá contradicción entre las especificaciones de longitud definidas para el tipo de campo de PDU en las definiciones de tipo de serie de pruebas y las especificaciones de longitud en la definición de tipo de PDU, esto es, el conjunto de cadenas definidas por una restricción de longitud en una definición de PDU será un verdadero subconjunto del conjunto de cadenas definidas por la definición de tipo de serie de pruebas.

      Se puede utilizar la palabra clave INFINITY como valor de límite superior para indicar que la longitud no tiene límite superior.

Todos los parámetros de CM son optativos, es decir, se pueden omitir cuando se utiliza un CM.

Esta información se proporcionará con el formato del formulario siguiente.

Definición de tipo de CM		
Nombre del CM : *CM_Identifier*		
Grupo : *[CM_GroupReference]*		
Comentarios : *[FreeText]*		
Nombre del parámetro	**Tipo de parámetro**	**Comentarios**
⋮ *CM_ParIdOrMacro* ⋮	⋮ *Type&Attributes* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*		

Formulario 34: Definición de tipo de CM

Las columnas Nombre del parámetro y Tipo de parámetro estarán presentes o ausentes a la vez.

11.17.3 Definiciones de tipos de CM con ASN.1

Los tipo de CM se pueden declarar mediante la ASN.1. Para cada tipo de CM en ASN.1 se proporcionará la información siguiente:

a) nombre,

   que será único dentro de la serie de pruebas;

b) la definición de tipo de CM en ASN.1,

   que seguirá la sintaxis definida en la Rec. UIT-T X.680. En los identificadores internos a esa definición no se utilizará el símbolo (-). En su lugar podrá emplearse el símbolo de subrayado (_). El identificador de PDU en el encabezamiento del cuadro es el nombre del primer tipo definido en el cuerpo del cuadro.

Los tipos a que hace referencia la definición de la PDU deberán estar definidos en otros cuadros de definición de tipo en ASN.1, definidos por referencia en el cuadro de referencias del tipo ASN.1 o definidos localmente en el mismo cuadro, a continuación de la primera definición de tipo. Los tipos definidos localmente no se utilizarán en otras partes de la serie de pruebas.

Se puede hacer comentarios en ASN.1 en el cuerpo del cuadro. En el cuadro no estará presente la columna de comentarios.

Los Comentarios en ASN.1 comienzan con "--" y terminan con la siguiente aparición de "--" o con "end of line", el primero que llegue. Esto impide que un sólo comentario en ASN.1 ocupe varias líneas. Para facilitar el intercambio de series ATS en TTCN.MP, se recomienda utilizar especificadores de ATS, y encerrar siempre entre "--" los comentarios en ASN.1.

Esta información se proporcionará con el formato del formulario siguiente.

Definición de tipo de CM en ASN.1	
Nombre del CM	**:** *CM_Identifier*
Grupo	**:** *[ASN1_CM_GroupReference]*
Comentarios	**:** *[FreeText]*
Definición de tipo	
ASN1_Type&LocalTypes	
Comentarios detallados:	*[FreeText]*

Formulario 35: Definición de tipo de CM en ASN.1

11.18 Especificaciones de longitud de cadena

La TTCN permite la especificación de restricciones de longitud a tipos de cadena (esto es, BITSTRING, HEXSTRING, OCTETSTRING y todos los tipos de CharacterString, más los tipos de ASN.1 BIT STRING y OCTET STRING) en las siguientes circunstancias:

a) cuando se declaren tipos de series de pruebas como en una restricción de tipo;

b) cuando se declaren parámetros de ASP simples, campos de PDU y elementos de tipos estructurados como un atributo del tipo de elemento, campo o parámetro;

c) cuando se definan constricciones de ASP/PDU o tipos estructurados como un atributo del valor de constricción.

Las especificaciones de longitud pueden tener los siguientes formatos:

a) [Length]

que restringe la longitud de los valores de cadena posibles de un tipo exactamente a *Length;*

b) [MinLength TO MaxLength] o [MinLength ... MaxLength]

que especifica una longitud máxima y una mínima para los valores de un tipo de cadena particular.

Los límites de longitud: *Length, MinLength* y *MaxLength* son de complejidad diferente, según el lugar donde se utilizan. En todos los casos, estos límites tomarán valores INTEGER no negativos. Para el límite superior, se puede utilizar también la palabra INFINITY con la que se indica que no existe límite superior de la longitud. Cuando se especifique una gama de longitudes, se indicará a la izquierda el menor de los dos valores.

En el contexto de las constricciones, se pueden especificar también restricciones de longitud para valores de tipo SEQUENCE OF o SET OF, limitando de este modo el número de sus elementos.

En el cuadro 5 siguiente se especifican las unidades de longitud de los diferentes tipos de cadenas.

Cuadro 5/X.292 – Unidades de longitud utilizadas en especificaciones de longitud de campo

Tipo	Unidades de longitud
BITSTRING o BIT STRING	Bits
HEXSTRING	Dígitos Hex
OCTETSTRING u OCTET STRING	Octetos
CharacterString	Caracteres
SEQUENCE OF	Elementos de su tipo de base
SET OF	Elementos de su tipo de base

No habrá contradicción entre las especificaciones de longitud, es decir, una restricción a un tipo (conjunto de valores) que ya está restringido especificará una subgama de valores de su tipo de base.

EJEMPLO 35 – Especificación de longitud:

Supóngase las siguientes definiciones de tipo en ASN.1:

type1 ::= OCTETSTRING [0 .. 25]

type2 ::= type1 [15 .. 24]

La restricción de longitud a type2, es correcta porque type2 comprende todos los valores de OCTETSTRING que tienen una longitud mínima de 15 y una longitud máxima de 24, lo que constituye un subconjunto verdadero de todos los valores de OCTETSTRING de longitud máxima 25. Por otra parte:

type2 ::= type1[15 .. 30]

no es válido, porque contiene valores no incluidos en type1.

11.19 Definiciones de ASP, PDU y CM para eventos SEND (enviar)

En las ASP y/o las PDU enviadas desde el probador, los valores de los parámetros de ASP y/o de los campos de PDU definidos en la parte constricciones (véanse las cláusulas 12,13 y 14) se corresponderán con la definición del parámetro o campo. Esto significa que:

a) el valor será del tipo especificado para ese parámetro ASP o campo de PDU; y

b) los valores satisfarán cualquier tipo de restricciones de longitud pertinentes asociadas con el tipo;

c) los valores de campo de PDU deberán codificarse de conformidad con las reglas de codificación aplicables.

Las operaciones de codificación definidas en la serie de pruebas se ejecutan implícitamente como parte del suceso SEND. En caso necesario se aplicarán los valores por defecto o de sustitución. De este modo, la salida del suceso SEND es la información codificada que se ha de transferir al proveedor del servicio pertinente.

11.20 Definiciones de ASP, PDU y CM para eventos RECEIVE (recibir)

El tipo de ASP y/o PDU define para las ASP y PDU recibidas por el probador, las clases de ASP y/o PDU entrantes que pueden concordar con una especificación de evento de ese tipo. Una ASP o PDU entrante se considera que pertenece a esa clase si y sólo si:

a) los valores de parámetro ASP y/o campo de PDU son del tipo especificado en la definición de ASP y/o PDU; y

b) el valor satisface cualquier restricción de longitud pertinente asociada con el tipo;

c) los valores de campo de PDU deberán codificarse de conformidad con las reglas de codificación aplicables.

En todos los demás casos, una ASP y/o una PDU entrantes no concuerdan con una especificación de evento de ese tipo.

En el caso de ASP y/o PDU subestructuradas ya sea con tipos estructurados o ASN.1, las reglas anteriores son aplicables a los campos de la subestructura o subestructuras recursivamente.

11.21 Definiciones de alias

11.21.1 Introducción

Para acrecentar la legibilidad de las descripciones de comportamiento en TTCN, se puede utilizar un alias que facilite la redenominación de los identificadores de ASP y/o PDU en descripciones de comportamiento. La redenominación puede efectuarse resaltando el intercambio de las PDU insertadas en las ASP.

Para cada alias se facilitará la siguiente información:

a) un identificador de alias;

b) su expansión,

que, en sí misma, es un identificador.

La información se proporcionará con el formato del formulario siguiente.

Definiciones de alias		
Grupo : *[AliasGroupReference]*		
Nombre de alias	**Expansión**	**Comentarios**
⋮ *AliasIdentifier* ⋮	⋮ *Expansion* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*		

Formulario 36: Definiciones de alias

En este cuadro se pueden utilizar comentarios colectivos de conformidad con la figura 2.

11.21.2 Expansión de alias

Se aplicarán las siguientes reglas:

a) Un alias es un identificador que deberá seguir las reglas de sintaxis para los identificadores definidas en la TTCN.MP.

b) Los alias no son transitivos – Si un alias aparece como la expansión de otro alias, no podrá ser a su vez expandido (esto es, se trata de una expansión de un solo paso).

c) Se podrá usar un alias solamente para sustituir un identificador de ASP o un identificador de PDU dentro de un solo enunciado TTCN en un árbol de comportamiento. Solamente se podrá usar en una columna de descripción de comportamiento.

d) La expansión de un alias deberá seguir las reglas de sintaxis para identificadores definidas en la TTCN.MP.

EJEMPLO 36 – Definición de alias desde una serie de pruebas de transporte:

Definiciones de alias		
Nombre de alias	**Desarrollo**	**Comentarios**
CR	N_DATArequest	Alias de la ASP de petición N_DATA utilizado para transportar una CR_TPDU
DR	N_DATArequest	Alias de la ASP de petición N_DATA utilizado para transportar una DR_TPDU
CC	N_DATAindication	Alias de la ASP de indicación N_DATA utilizado para transportar una CC_TPDU

NOTA – Puesto que los alias se tratan como expansiones de macro, el término AliasIdentifier no aparece en la BNF para las líneas de evento TTCN.

12 Parte constricciones

12.1 Introducción

Una ATS especificará los valores de los parámetros de ASP y campos de PDU que han de ser enviados o recibidos por el sistema de prueba. En la TTCN, este cometido lo cumple la parte constricciones.

Las descripciones de comportamiento dinámico (véase la cláusula 15) referenciarán constricciones para construir ASP y/o PDU salientes en eventos SEND y para especificar los contenidos esperados de ASP y/o PDU entrantes en eventos RECEIVE.

Las constricciones se pueden especificar en cualquiera de las dos formas siguientes:

a) constricciones tabulares (véase la cláusula 13);

b) constricciones en ASN.1 (véase la cláusula 14).

Los valores reales o constricciones impuestos a los valores de un CM se declararán del mismo modo que las constricciones de PDU.

12.2 Principios generales

En esta subcláusula se describen los principios generales y se seleccionan los mecanismos relativos a la construcción de constricciones para eventos SEND y se explica cómo efectuar la concordancia de eventos RECEIVE. Estos principios son comunes a ambas formas de constricciones, la tabular y la ASN.1.

Las constricciones son especificaciones detalladas de ASP y/o PDU. Normalmente cada constricción se define de forma específica para su uso con eventos SEND o eventos RECEIVE. No será necesario especificar una constricción si una ASP o un CM no tienen parámetros o si la PDU no tiene campos. En cada contexto se puede utilizar cualquier constricción dada, siempre que se cumplan las restricciones semánticas operacionales definidas en el anexo B.

La especificación de constricción de una ASP y/o PDU tendrá la misma estructura que la definición de tipo de esa ASP o PDU.

Si una ASP y/o PDU está subestructurada, las constricciones de esa ASP y/o PDU de dicho tipo tendrán la misma estructura tabular o una estructura ASN.1 compatible (es decir, posiblemente con algunas agrupaciones).

Se considera que los tipos estructurados, expandidos a una definición de ASP o PDU mediante la utilización del símbolo macro (<-) no son subestructuras. Las constricciones para esas ASP o PDU tendrán una estructura completamente plana (es decir, que la relación de los elementos de una estructura expandida figura de manera explícita en la constricción de ASP o PDU) o harán referencia a una constricción de estructura correspondiente para expansión de macro.

Las constricciones especifican valores de parámetros de ASP y de campo de PDU con diversas combinaciones de valores literales, referencias de objetos de datos, expresiones, valores construidos en ASN.1, mecanismos de concordancia especiales y referencias a otras constricciones. Las constricciones que se aplican a la totalidad de una PDU o a una parte de la misma pueden también especificar reglas de codificación para sustituir a las reglas de codificación que se están aplicando en la serie de pruebas. Tales reglas de codificación se pueden especificar para la constricción completa o para un solo campo de la constricción.

En las constricciones se pueden usar valores de todos los tipos de TTCN o de ASN.1. Las expresiones utilizadas en las constricciones tomarán un valor específico cuando se utilice la constricción para el envío o la recepción de eventos.

Cualquiera que sea la forma en que se obtengan los valores, corresponderán a las inscripciones de parámetro o de campo en las definiciones de tipo de ASP o de PDU. Esto significa que:

a) el valor será del tipo especificado para ese parámetro o campo; y

b) la longitud satisfará toda restricción asociada con el tipo.

Una expresión en una constricción solamente contendrá valores (incluidos, por ejemplo ConstraintValue&Attributes), parámetros de series de pruebas, constantes de series de pruebas, parámetros formales, referencias de componentes y operaciones serie de pruebas.

Se admite también como valor de parámetro o valor de campo (encadenamiento estático) una referencia a constricción (posiblemente parametrizada).

En las constricciones, no se utilizarán variables de series de pruebas ni variables de caso de prueba, a menos que se pasen como parámetros reales. En este último caso deberán estar acotadas a un valor y no serán modificadas por la ocurrencia de un evento SEND o RECEIVE.

En 12.6.2 se definen los mecanismos de concordancia.

12.3 Parametrización de constricciones

Las constricciones se pueden parametriza. En esos casos, el nombre de la constricción irá seguido de una lista de parámetros formales encerrados entre paréntesis. Los parámetros formales se utilizarán para especificar valores de parámetros de ASP o de campos de PDU en la constricción.

Cada nombre de parámetro formal irá seguido del símbolo dos puntos (:) y del nombre del tipo de parámetro. Cuando se utilice una sublista de parámetros, los nombres de parámetro irán separados por coma (,). El parámetro final de la sublista irá seguido por el símbolo dos puntos y el nombre del tipo de la sublista de parámetros. Cuando se utilicen más de un par parámetro y tipo (o par sublista de parámetros y tipos), los pares deberán estar separados entre sí mediante el símbolo punto y coma (;).

En la referencia a constricciones efectuada desde una descripción de comportamiento, pueden transferirse como parámetros reales de la constricción valores literales, parámetros de serie de pruebas, constantes de serie de pruebas, variables de serie de pruebas, variables de caso de prueba y PDU o constricciones de tipo de serie de pruebas. Estos parámetros no podrán ser de tipo PCO ni de tipo ASP.

12.4 Encadenamiento de constricciones

Las constricciones se pueden encadenar referenciando una constricción como el valor de un parámetro o campo de otra constricción. Por ejemplo, el valor del parámetro datos de una ASP petición N-DATOS (petición de datos de red) podría ser una referencia a una constricción de PDU de T-CRPDU (PDU de petición de conexión de transporte), es decir, la T-CRPDU es encadenada a la ASP petición N-DATOS.

Las constricciones pueden encadenarse de una de las dos formas siguientes, ya sea por:

a) encadenamiento estático, en el que un valor del parámetro una ASP o un valor de campo de una PDU en una constricción es una referencia explícita a otra constricción; o

b) encadenamiento dinámico, en el que un valor del parámetro una ASP o un valor de campo de una PDU en una constricción constituye un parámetro formal de la constricción. Cuando tal constricción se referencia desde un comportamiento dinámico, el parámetro real correspondiente a la constricción es una referencia a otra constricción (en el anexo F se presentan ejemplos de encadenamiento estático y dinámico).

Cuando se haga referencia a las constricciones en declaraciones de constricción, tales referencias no serán recursivas (ni directa ni indirectamente).

El encadenamiento de constricciones solo se puede utilizar si se han establecido las declaraciones apropiadas para permitir el encadenamiento. Por ejemplo, si un parámetro ASP ha de encadenarse a una constricción de PDU deberá declararse que el parámetro ASP sea de un tipo de PDU apropiado o el metatipo **PDU**. En declaraciones de PDU en ASN.1, el tipo de PDU puede ser perfectamente un tipo definido como una CHOICE de todos los tipos de PDU individuales válidos, mientras que en las declaraciones de PDU en forma de tabla habría de utilizarse el metatipo **PDU** para lograr un efecto similar. De manera análoga, si un campo de PDU se va a encadenar a una constricción de estructura, deberá declararse que el campo de la PDU sea de un tipo de estructura apropiado.

12.5 Constricciones para eventos SEND

Las constricciones que son referenciadas para eventos SEND no incluirán comodines (es decir, AnyValue (?) o AnyOrOmit (*)) a menos que se les hayan asignado explícitamente valores pecíficos en la línea de eventos SEND, en la descripción de comportamiento.

En constricciones en forma tabular, todos los parámetros de ASP y campos de PDU son opcionales y, por esa razón, pueden ser omitidos con el símbolo Omit, para indicar que el parámetro ASP o campo de PDU estará ausente en el evento de enviar.

En constricciones en ASN.1, sólo podrán omitirse parámetros de ASP y campos de PDU declarados como OPTIONAL. Estos parámetros pueden omitirse ya sea con el símbolo Omit o, simplemente, dejando fuera el parámetro ASP o campo de PDU pertinente.

Ninguno de los mecanismos de concordancia definidos en 12.6.2, excepto SpecificValue, proporciona un valor para un parámetro ASP o campo de PDU sobre un evento de SEND.

En aquellos casos en que se utilicen valores de ASN.1 de tipo SET o SET OF en una constricción, los valores de los elementos del conjunto se enviarán en el orden especificado por la constricción pertinente.

12.6 Constricciones para eventos RECEIVE

12.6.1 Valores de concordancia

Si se utiliza una constricción para construir los valores de parámetros de ASP o de campos de PDU con los que deberá concordar una ASP o una PDU recibida, aquélla contendrá los valores específicos evaluados como se indica en 12.6.3 o mecanismos de concordancia especiales, siempre que no sea deseable o posible la especificación de valores concretos. Los mecanismos de concordancia especifican otras formas de concordancia diferentes de "igual a un valor específico".

Una ASP y/o una PDU entrante concuerdan con una constricción utilizada en un evento RECEIVE si y sólo si se cumplen todas las condiciones siguientes:

a) los parámetros de la ASP y/o los campos de la PDU son del tipo especificado en las definiciones de ASP y/o PDU;

b) el valor, el alfabeto y la longitud satisfacen cualquier limitación asociada al tipo;

c) los valores del parámetro ASP y/o del campo de PDU concuerdan correctamente con los de la constricción;

d) para las unidades PDU, si se ha efectuado la decodificación correcta de las mismas, teniendo en cuenta los valores por defecto y de sustitución de las reglas de codificación aplicables; si para codificar la PDU recibida se han utilizado reglas de codificación distintas de las especificadas para la constricción, esa PDU recibida no concordará.

En el caso de ASP y/o PDU subestructuradas, mediante tipos estructurados o ASN.1, las reglas anteriores se aplicarán recursivamente a los campos de la subestructura o subestructuras.

NOTA – Si un evento RECEIVE es calificado por una expresión booleana, una concordancia correcta significa que la ASP y/o la PDU entrantes deben concordar con la constricción y que el calificador debe tomar el valor TRUE.

12.6.2 Mecanismos de concordancia

En el cuadro 6, se presenta una visión general de los mecanismos de concordancia soportados, incluyendo los símbolos especiales y el ámbito de su aplicación. En la columna izquierda del cuadro se enumeran todos los tipos de ASN.1 y tipos de TTCN equivalentes a los que se aplican estos mecanismos de concordancia. Los mecanismos de concordancia situados en los encabezamientos horizontales se han dividido en cuatro grupos:

a) valores específicos;

b) símbolos especiales que se pueden usar *en vez de (instead of)* valores;

c) símbolos especiales que se pueden usar *dentro de (inside)* valores;

d) símbolos especiales que describen *atributos (attributes)* de valores.

Algunos de los símbolos se pueden usar combinados, como se indica en las cláusulas que siguen.

La zona sombreada del cuadro 6 indica los mecanismos que se aplican a los tipos de ASN.1 y de TTCN predefinidos.

Cuadro 6/X.292 – Mecanismos de concordancia en TTCN

	VALOR	VALORES EN VEZ DE								VALORES DENTRO DE			ATRIBUTOS	
TYPE	**Specific Value**	**Complement**	**Omit(-)**	**AnyValue (?)**	**AnyOrOmit(*)**	**ValueList**	**Range**	**SuperSet**	**SubSet**	**AnyValue (?)**	**AnyOrNone(*)**	**Permutation**	**Length**	**IfPresent**
BOOLEAN	•	•	•	•	•	•								•
INTEGER	•	•	•	•	•	•	•							•
ENUMERATED	•	•	•	•	•	•								•
BITSTRING	•	•	•	•	•	•				•	•		•	•
OCTETSTRING	•	•	•	•	•	•				•	•		•	•
HEXSTRING	•	•	•	•	•	•				•	•		•	•
CHARSTRINGS	•	•	•	•	•	•				•	•		•	•
SEQUENCE		•	•	•	•	•								•
SEQUENCE OF	•	•	•	•	•	•				•	•	•	•	•
SET	•	•	•	•	•	•								•
SET OF	•	•	•	•	•	•		•	•	•	•		•	•
ANY	•	•	•	•	•	•								•
CHOICE	•	•	•	•	•	•								•
OBJECT ID	•	•	•	•	•	•								•

En una especificación de constricción, los mecanismos de concordancia pueden reemplazar valores de parámetros de ASP o campos de PDU únicos o incluso de la totalidad del contenido de una ASP o una PDU.

NOTA – Cuando estos mecanismos de concordancia se utilicen solos o en combinación, se pueden especificar en las constricciones numerosas restricciones de protocolo, evitando de este modo detalles de computación no deseables en la parte comportamiento.

12.6.3 Specific Value (valor específico)

Este es el mecanismo de concordancia básico. Los valores específicos de constricciones son expresiones. A menos que se especifique otra cosa, un parámetro una ASP o un campo de una PDU de constricción, concuerda con el parámetro ASP o el campo de PDU correspondiente si, y sólo si, el parámetro ASP o el campo de PDU entrante tiene exactamente el mismo valor que el valor que toma la expresión en la constricción.

Se considera que los dos valores de una ASP, PDU o tipo estructurado expresado en forma tabular o de SEQUENCE o SEQUENCE OF de ASN.1, son el mismo si cada uno de sus campos de parámetros o elementos concuerdan entre sí y están en el mismo orden. En el caso de tipos SET y SET OF de ASN.1, se considera que dos valores son el mismo si tienen igual número de elementos y cada elemento de un valor concuerda exactamente con un elemento del otro valor. Para que exista concordancia, no es necesario que los elementos de un valor de tipo SET o SET OF estén en el mismo orden.

12.6.4 Instead of value (valor en vez de)

12.6.4.1 Complement (complemento)

Complement es una operación de concordancia que se puede utilizar con todos los valores de todos los tipos. El complemento se denota por la palabra clave COMPLEMENT, seguida de una lista de valores de constricción. Cada valor de constricción de la lista será del tipo declarado para el parámetro ASP o campo de PDU en el que se utilice el mecanismo de complemento.

Un parámetro ASP o campo de PDU de constricción que utilice Complement concuerda con el parámetro ASP o el campo de PDU correspondiente si, y sólo si, el parámetro ASP o el campo de PDU entrante no concuerda con ninguno de los valores enumerados en ValueList.

EJEMPLO 37 – Constricciones con Complement en vez de un valor y con una lista de valores:

Tipo	*Constricción*
INTEGER	COMPLEMENT(5)
INTEGER	COMPLEMENT(1, 3, 5)

12.6.4.2 Omit (omitir)

Omit es un símbolo especial para concordancia que se puede utilizar con valores de todos los tipos, siempre que el parámetro ASP o campo de PDU sea opcional.

En las constricciones en ASN.1 también es posible dejar fuera un parámetro ASP o campo de PDU OPTIONAL, en vez de utilizar explícitamente OMIT.

NOTA – En las constricciones en forma tabular, todos los parámetros, campos y elementos se consideran optativos implícitamente, y en consecuencia pueden ser omitidos con Omit. En las constricciones en ASN.1, los parámetros, campos y elementos que no están explícitamente señalados como OPTIONAL en la definición de tipo son obligatorios y no pueden omitirse sin violar la definición del tipo. Si es necesario que ese parámetro, campo o elemento sea omitido de una constricción particular, se necesita, bien definir otro tipo en el cual dicho parámetro, campo o elemento sea explícitamente señalado como OPTIONAL (quizás mediante la marcación de todo como OPTIONAL), o bien aplicar una codificación de campo no válida al parámetro, campo o elemento, con la consecuencia de que se le omite de la codificación.

En las constricciones en forma tabular, Omit deberá denotarse por un guión (-). En constricciones en ASN.1, Omit se denota por OMIT.

Se utiliza un símbolo Omit en una constricción para indicar la ausencia de un parámetro ASP o un campo de PDU opcional.

EJEMPLO 38 – Constricciones con Omit en vez de un valor en el nivel superior:

Tipo	*Constricción*
INTEGER OPTIONAL	OMIT

12.6.4.3 AnyValue (cualquier valor)

AnyValue es un símbolo especial para concordancia que se puede utilizar con valores de todos los tipos. Tanto para las constricciones en ASN.1 como en forma tabular, AnyValue se denota por "?".

Un parámetro ASP o campo de PDU de constricción que utiliza AnyValue, concuerda con el parámetro ASP o campo de PDU entrante correspondiente si, y sólo si, el parámetro ASP o campo de PDU toma el valor de un único elemento del tipo especificado.

EJEMPLO 39 – Constricciones con Value en combinación con AnyValue:

Tipo	*Constricción*
SEQUENCE OF SET OF INTEGER	{ {1, 2}, ?, {1, 2, ?} }

12.6.4.4 AnyOrOmit (cualquiera u omitir)

AnyOrOmit es un símbolo especial para concordancia que se puede utilizar con valores de todos los tipos, siempre que se hayan declarado como opcionales el parámetro ASP o el campo de PDU. Tanto para las constricciones en forma tabular como en ASN.1, AnyOrOmit se denota por "*".

NOTA – Se utiliza el símbolo "*" para representar tanto a AnyOrOmit como a AnyOrNone. La ambigüedad de la interpretación se resuelve mediante los requisitos establecidos en 12.6.4.4 y 12.6.5.2.

Un parámetro ASP o campo de PDU de constricción que utiliza AnyOrOmit concuerda con el parámetro ASP o campo de PDU entrante correspondiente únicamente si, o bien el parámetro ASP, o el campo de PDU entrante toma el valor de cualquier elemento del tipo especificado, o bien está ausente el parámetro ASP o campo de PDU entrante.

EJEMPLO 40 – Constricciones con Value en combinación con AnyOrOmit:

Tipo		*Constricción*	
SEQUENCE OF {	id1 SET OF INTEGER	{ id1 {2, 5},	
	id2 SET OF INTEGER }	id2 *	}

12.6.4.5 ValueList (lista de valores)

ValueList se puede utilizar con valores de todos los tipos. Tanto en las constricciones en forma tabular como en ASN.1. ValueList se representa mediante una lista de valores separados por comas y encerrados entre paréntesis.

Un parámetro ASP o campo de PDU de constricción que utiliza una ValueList concuerda con el parámetro ASP o campo de PDU entrante correspondiente si, y sólo si, el valor del parámetro ASP o campo de PDU entrante concuerda con uno cualquiera de los valores de ValueList. Cada valor de ValueList será del tipo declarado para el parámetro ASP o campo de PDU en el que se utiliza el mecanismo ValueList.

EJEMPLO 41 – Constricciones con ValueList en vez de un valor específico, para tipo INTEGER:

Tipo	*Constraint*
INTEGER	(2, 4, 6)

EJEMPLO 42 – Constricciones con ValueList en vez de un valor específico, para el tipo CHOICE:

Tipo	*Constricción*
CHOICE { a INTEGER, b BOOLEAN }	(a 2, b TRUE)

12.6.4.6 Range (gama o rango)

Solamente podrán utilizarse gamas con valores de tipo INTEGER. Una gama se designa mediante dos valores límites separados por ".." o TO, encerrados entre paréntesis. Un valor límite será:

a) INFINITY o –INFINITY; o bien

b) una expresión que toma a un valor INTEGER específico.

El límite inferior se colocará a la izquierda de ".." o TO, situándose el límite superior a la derecha. El límite inferior será menor que el límite superior.

Un parámetro ASP o campo de PDU de constricción que utiliza una gama concuerda con el parámetro ASP o campo de PDU entrante correspondiente si, y sólo si, el valor del parámetro ASP o campo de PDU entrante es igual a uno de los valores de la gama.

EJEMPLO 43 – Constricciones con Range en vez de valor:

Tipo	*Constricción*
INTEGER	(1 .. 6) (-INFINITY .. 8) (12 .. INFINITY)

12.6.4.7 SuperSet (superconjunto)

SuperSet es una operación para concordancia que se utilizará solamente con valores del tipo SET OF. Solamente se utilizará SuperSet con constricciones en ASN.1. SuperSet se denota por SUPERSET.

Un parámetro ASP o campo de PDU de constricción que utiliza SuperSet concuerda con el parámetro ASP o campo de PDU entrante correspondiente si, y sólo si, el parámetro ASP o campo de PDU contiene, al menos, todos los elementos definidos en SuperSet, aunque puede contener más. El argumento de SuperSet será del tipo declarado para el parámetro ASP o el campo de PDU en el que se utiliza el mecanismo SuperSet.

EJEMPLO 44 – Constricciones con SuperSet en vez de un valor específico:

Tipo	*Constricción*
SET OF INTEGER	SUPERSET({1, 2, 3})

12.6.4.8 SubSet (subconjunto)

SubSet es una operación de concordancia que se puede utilizar solamente con valores de tipo SET OF. Solamente se utilizará SubSet con constricciones en ASN.1. SubSet se denota por SUBSET.

Un parámetro ASP o campo de PDU de constricción que utiliza SubSet concuerda con el parámetro ASP o campo de PDU entrante correspondiente si, y sólo si, el parámetro ASP o campo de PDU entrante contiene únicamente elementos definidos en el SubSet, aunque puede contener menos. El argumento de SubSet será del tipo declarado para el parámetro ASP o campo de PDU en el que se utiliza el mecanismo SuperSet.

EJEMPLO 45 – Constricciones con SuperSet en vez de un valor específico:

Tipo	*Constricción*
SET OF INTEGER	SUBSET({2, 4, 6, 8, 10})

12.6.5 Inside Values (valores interiores)

12.6.5.1 AnyOne (cualquiera)

AnyOne es un símbolo especial para concordancia que se puede utilizar con valores de tipos cadena, SEQUENCE OF y SET OF. En las constricciones en forma tabular y en ASN.1, AnyOne se denota por "?".

Dentro de una cadena, SEQUENCE OF o SET OF, una "?" en lugar de un solo elemento significa que se aceptará cualquier elemento aislado. Si resulta necesario utilizar el símbolo "?" dentro de una CharacterString como un carácter se representará mediante "\?". Si dentro de una CharacterString se necesita utilizar como carácter el símbolo "\" se representará mediante "\\".

EJEMPLO 46 – Constricciones con AnyOne:

Tipo	*Constricción*
IA5String	"a?cd"
SEQUENCE OF INTEGER	{1, 2, ? }

NOTA – El "?" del segundo ejemplo puede interpretarse como un AnyValue que reemplaza un valor INTEGER o un valor AnyOne dentro de un valor SEQUENCE OF INTEGER. Como ambas interpretaciones conducen al mismo conjunto de eventos que hacen concordar la constricción, no se plantea ningún problema.

12.6.5.2 AnyOrNone (cualquiera o ninguno)

AnyOrNone es un símbolo especial para concordancia que se puede utilizar con valores de tipos cadena, SEQUENCE OF y SET OF. En las constricciones en forma tabular y en ASN.1, AnyOrNone se denota por "*".

Si aparece un "*" en el nivel superior dentro de un valor de tipo cadena, SEQUENCE OF o SET OF, se interpretará como AnyOrNone.

NOTA – Esta regla evita la en otro caso posible interpretación de "*" como AnyOrOmit que reemplaza un elemento dentro de la cadena SEQUENCE OF o SET OF.

Dentro de una cadena, SEQUENCE OF o SET OF, un "*" en lugar de un solo elemento significa que o bien no se aceptará ninguno o bien se aceptará cualquier número de elementos consecutivos. El símbolo "*" hace concordar la secuencia más larga posible de elementos de conformidad con el patrón especificado por los símbolos que rodean al "*". Si, dentro de una CharacterString, es necesario utilizar como carácter el símbolo "*" éste se representará por "*". Si, dentro de una CharacterString, es necesario utilizar como carácter el símbolo "\", se representará mediante "\\".

EJEMPLO 47 – Constricciones con AnyOrOne:

Tipo	*Constricción*
IA5String	"ab*z"
SEQUENCE OF INTEGER	{1, 2, *, 10 }
SEQUENCE OF IA5String	{ "ab*z", *, "abc" }

12.6.5.3 Permutation (permutación)

Permutation es una operación para concordancia que se puede utilizar solamente con valores interiores a un valor de tipo SEQUENCE OF. Solamente se utilizará Permutation con constricciones en ASN.1. Permutation se denota por PERMUTATION.

Permutation en lugar de un solo elemento significa que es aceptable cualquier serie de elementos, siempre que contenga los mismos elementos que la lista de valores en Permutation, aunque posiblemente en un orden diferente. Si, dentro de un valor, se utilizan Permutation y AnyOrNone, se evaluará en primer lugar AnyOrNone. Cada uno de los elementos enumerados en Permutation serán del tipo declarado dentro del tipo SEQUENCE OF del parámetro ASP o campo de PDU.

EJEMPLO 48 – Constricciones con Permutation:

Tipo	*Constricción*
SEQUENCE OF INTEGER	{PERMUTATION (1, 2, 3), 5}

EJEMPLO 49 – Constricciones con Permutation en combinación con AnyOrNone:

Tipo	*Constricción*
SEQUENCE OF INTEGER	{PERMUTATION (1,2,3), *} {PERMUTATION (1,2,3,*)}

Obsérvese que la primera constricción concuerda con ASP y/o PDU entrantes que constan de una secuencia de valores INTEGER que se inicia con 1,2,3; 1,3,2; 2,1,3; 2,3,1; 3,1,2; ó 3,2,1 y va seguida por cualquier número de valores de tipo INTEGER. La segunda constricción concuerda con cualquier ASP y/o PDU entrante de tipo SEQUENCE OF INTEGER

que contiene los elementos 1, 2, 3 en cualquier orden y en cualquier posición. Por ejemplo, hace concordar {5,2,7,1,3} y {9,3,7,2,12,1,17}.

12.6.6 Atributes of values (atributos de valores)

12.6.6.1 Length (longitud)

Length es una operación para concordancia que se puede utilizar solamente como atributo de los siguientes mecanismos: Complement, AnyValue, AnyOrOmit, AnyOne, AnyOrNone, Permutation, SuperSet y SubSet. Se puede utilizar junto con el atributo IfPresent.

En las constricciones en forma tabular y en ASN.1 puede especificarse la longitud como un valor exacto o una gama de valores cadena y de valores SEQUENCE OF y SET OF, de conformidad con 11.18. Las unidades de longitud se interpretarán según el cuadro 5. Los límites pueden designarse mediante valores INTEGER específicos no negativos. Alternativamente, se puede utilizar la palabra clave INFINITY como valor del límite superior, para indicar que no tiene límite superior.

No habrá contradicción entre las especificaciones de longitud definidas para el tipo de parámetro ASP o campo de PDU, en las definiciones de tipo de series de pruebas y las especificaciones de longitud en las constricciones de ASP o PDU; esto es, el conjunto de cadenas definidas por una restricción de longitud en una constricción de ASP o PDU será un subconjunto verdadero del conjunto de cadenas definidas por la definición de ASP o PDU.

Un parámetro ASP o campo de PDU de constricción que utiliza Length como atributo de un símbolo, concuerda con el parámetro ASP o campo de PDU entrante correspondiente si, y sólo si, el parámetro ASP o campo de PDU concuerda tanto con el símbolo como con su atributo asociado. El atributo de longitud concuerda si la longitud del parámetro ASP o campo de PDU entrante es mayor o igual que el límite inferior y menor o igual que el límite superior especificados. En el caso de un valor de longitud único, el atributo de longitud concuerda si la longitud del parámetro ASP o campo de PDU recibidos es exactamente el valor especificado.

En el caso de un parámetro, campo o elemento omitido, Length se considera siempre concordante. Por consiguiente, con Omit es redundante y con AnyOrOmit y IfPresent sitúa una restricción en el valor entrante, si existe.

EJEMPLO 50 – Constricciones con Value en combinación con Length:

Tipo	*Constricción*
IA5String	"ab*ab" [13]

12.6.6.2 IfPresent (si presente)

IfPresent es un símbolo especial para concordancia que se puede utilizar como un atributo de todos los mecanismos de concordancia, siempre que se declare el tipo como opcional. En constricciones en forma tabular y en ASN.1, IfPresent se denota por IF_PRESENT.

Un parámetro ASP o campo de PDU de constricción que utiliza un símbolo IfPresent como atributo de otro símbolo, concuerda con el parámetro ASP o campo de PDU entrante correspondiente si, y sólo si, el parámetro ASP o campo de PDU entrante concuerda con el símbolo o si está ausente el parámetro ASP o campo de PDU entrante.

NOTA – El símbolo AnyOrOmit (*) tiene exactamente el mismo significado que ? IF_PRESENT.

EJEMPLO 51 – Constricciones con Value en combinación con IfPresent:

Tipo	*Constricción*
IA5String OPTIONAL	"abcdef" IF_PRESENT

13 Especificación de constricciones mediante cuadros (o tablas)

13.1 Introducción

En esta cláusula se describe la especificación de constricciones en forma tabular impuestas a tipos estructurados, ASP y PDU. Se indica cómo se pueden usar los cuadros de constricciones simples para especificar constricciones impuestas a ASP o PDU planas (no estructuradas) y cómo se pueden especificar constricciones estructuradas declarando constricciones impuestas a tipos estructurados, definidos en los tipos de series de pruebas.

En el anexo C se definen cuadros adicionales que permiten efectuar muchas declaraciones de constricciones simples en un solo cuadro.

13.2 Declaraciones de constricciones de tipo estructurado

Si se define una ASP o PDU con tipos estructurados ya sea como expansiones de macro o subestructuras, deberán subestructurarse análogamente las constricciones para esas ASP o PDU. Para cada constricción de tipo estructurado deberá suministrarse la siguiente información:

a) El nombre de la constricción,

   que puede ir seguido de una lista de parámetros formales optativos.

b) El nombre del tipo estructurado.

c) El trayecto de derivación (véase 13.6).

d) Las variaciones de codificación que han de utilizarse para la constricción.

   A fin de especificar las variaciones de codificación explícita para constricciones de tipo estructurado enteras, que sustituyen a las reglas de codificación y variaciones de codificación aplicables a la constricción de PDU en la cual se utiliza esta constricción de tipo estructurado, esta entrada optativa hará referencia a una entrada en el cuadro de variaciones de codificación pertinente [por ejemplo, para cambiar de SD a LD(3)]. Si no se utilizan esta entrada, las reglas de codificación y las variaciones de codificación aplicables a la constricción de PDU se aplica también a esta constricción de tipo estructurado. Véase 11.16.4.

e) Un valor de constricción para cada elemento,

   donde se suministrará la siguiente información para cada elemento:

   1) nombre,

      cada entrada en la columna de nombre del elemento deberá declararse en la definición de tipo estructurado pertinente. Si alguno de los elementos originales se ha definido de modo que tenga un nombre abreviado y un identificador completo, la constricción no deberá repetir el identificador completo.

      Si la definición de tipo estructurado se refiere a otro tipo estructurado por una expansión de macro (es decir, con "<-" en vez del nombre del nombre del elemento), en una constricción correspondiente:

      – se incluirán los elementos individuales del tipo estructurado directamente dentro de las constricciones; o bien

      – se situará el símbolo macro (<-) en la posición correspondiente de la columna de nombre del elemento de la constricción, y el valor constituirá una referencia a una constricción para el tipo estructurado referenciado a partir de esta definición de tipo estructurado.

      No deberán utilizarse constricciones estructuradas por expansión de macro en una constricción, a menos que la definición de tipo estructurado correspondiente haga también referencia al tipo estructurado interno por expansión de macro;

   2) su valor y un atributo opcional;

   3) optativamente, un identificador de codificación específico seguido de una lista de parámetros reales necesarios, para especificar la codificación explícita para el elemento individual de una constricción de tipo estructurado, que sustituya a las reglas de codificación y a las variaciones de codificación aplicables a la constricción de tipo estructurado global, y que sustituya también a cualquier codificación especificada para este elemento en la declaración del tipo estructurado. El identificador de codificación, si existe, deberá identificar una de las variaciones de codificación o una definición de codificación de campo no válida definida en la serie de pruebas [por ejemplo, LD(10)]; véase 11.16.4.

Los valores de elementos de las constricciones de estructura se proporcionarán con el formato del formulario siguiente.

Declaración de constricción de tipo estructurado			
Nombre de la constricción : *ConsId&ParList*			
Grupo : *[StructTypeConstraintGroupReference]*			
Tipo estructurado : *StructIdentifier*			
Trayecto de derivación : *[DerivationPath]*			
Variación de codificación : *[EncVariationCall]*			
Comentarios : *[FreeText]*			
Nombre del elemento	**Valor del elemento**	**Codificación del elemento**	**Comentarios**
⋮ *ElemIdentifier* ⋮	⋮ *ConstraintValue&Attributes* ⋮	⋮ *[PDU_FieldEncodingCall]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 37: Declaración de constricción de tipo estructurado

Este formulario se utiliza del mismo modo que se utiliza para las PDU el formulario de declaración de constricción de PDU (véase 13.4).

Si una definición de ASP o de PDU se refiere a un tipo estructurado como una subestructura de un parámetro o campo (esto es, con un nombre de parámetro o nombre de campo especificado para ella), la constricción correspondiente tendrá el mismo nombre de parámetro o de campo en la posición correspondiente de la columna de nombre de parámetro o nombre de campo de la constricción y el valor será una referencia a una constricción para ese parámetro o campo (es decir, para esa subestructura de acuerdo con la definición del tipo estructurado). Si la definición de ASP o PDU se refiere a un parámetro o campo especificado como del metatipo PDU, en la constricción correspondiente se especificará el valor de ese parámetro o campo como el nombre de una constricción de PDU o un parámetro formal.

13.3 Declaraciones de constricciones de ASP

Los valores de parámetros para constricciones de ASP se proporcionarán con el formato del formulario siguiente.

Declaración de constricción de ASP		
Nombre de la constricción : *ConsId&ParList*		
Grupo : *[ASP_ConstraintGroupReference]*		
Tipo de ASP : *ASP_Identifier*		
Trayecto de derivación : *[DerivationPath]*		
Comentarios : *[FreeText]*		
Nombre del parámetro	**Valor del parámetro**	**Comentarios**
⋮ *ASP_ParIdOrMacro* ⋮	⋮ *ConstraintValue&Attributes* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*		

Formulario 38: Declaración de constricción de ASP

Las columnas de Nombre de parámetro y Valor de parámetro deben estar presentes o ausentes a la vez.

Este formulario se utiliza para las ASP del mismo modo que se utiliza el formulario de declaración de constricción de PDU (véase 13.4), con la salvedad de que esta información de codificación no es pertinente, por lo que no se especificará.

13.4 Declaraciones de constricciones de PDU

Una constricción en formato tabular se define especificando un valor y los atributos opcionales de cada campo de PDU. Para cada constricción de PDU se facilitará la siguiente información:

a) el nombre de la constricción,

   que puede ir seguido de una lista de parámetros formales opcionales;

b) el nombre de tipo de la PDU;

c) el trayecto de derivación (véase 13.6);

d) las reglas de codificación que se utilizarán para la constricción,

   a fin de especificar codificaciones explícitas para constricciones de PDU enteras, que sustituyan a las reglas de codificación aplicables al tipo de PDU dado, esta entrada optativa hará referencia a una entrada en el cuadro de definiciones de codificación pertinente (por ejemplo cambiar de BER a DER). Si no se utiliza esta entrada, se aplican entonces las reglas de codificación aplicables al tipo de PDU. Véase 11.16.4;

e) las variaciones de codificación que se utilizarán para la constricción,

   a fin de especificar variaciones de codificación explícitas para constricciones de PDU enteras, que sustituyan a las variaciones de codificación aplicables al tipo de PDU dado, esta entrada optativa hará referencia a una entrada en el cuadro de definiciones de variaciones de codificación pertinente [por ejemplo cambiar de SD a LD(3)]. Si no se utiliza esta entrada, se aplican entonces las variaciones de codificación aplicables al tipo de PDU. Véase 11.16.4;

f) un valor de constricción para cada campo,

   para cada campo se suministrará la siguiente información:

   1) Nombre,

      cada una de las inscripciones del campo en la columna de nombre del campo se tiene que declarar en la definición de tipo de PDU pertinente. Si alguno de los campos de la PDU originales se ha definido de modo que tenga un nombre abreviado y un identificador completo, la constricción no deberá repetir el identificador completo.

      Si la definición de PDU se refiere a un tipo estructurado por una expansión de macro (es decir, "<-" en vez del nombre del campo de la PDU), en una constricción correspondiente:

      – se incluirán los elementos individuales del tipo estructurado directamente dentro de las constricciones; o bien

      – se situará el símbolo macro (<-) en la posición correspondiente de la columna de nombre del campo de la PDU de la constricción y el valor constituirá una referencia a una constricción para el tipo estructurado referenciado desde la definición de la PDU.

      No deberán utilizarse constricciones estructuradas por expansión de macro en una constricción, a menos que la definición de las PDU correspondiente haga también referencia al mismo tipo estructurado por expansión de macro.

   2) Su valor y un atributo opcional.

   3) Optativamente, un identificador de codificación específico seguido de una lista de parámetros reales necesarios, para especificar las codificaciones explícitas para los campos individuales de una constricción de PDU, que sustituya a las reglas de codificación y a las variaciones de codificación aplicables a la constricción de PDU en conjunto y que sustituya también a cualquier codificación de campo específica aplicable a este campo para unidades PDU de este tipo de PDU; el identificador de codificación, si existe, deberá identificar una de las variaciones de codificación o una definición de codificación de campo no válida definida en la serie de pruebas (por ejemplo, LD(10)); véase 11.16.4.

Con las constricciones ASP no se utilizará mecanismo de codificación.

Esta información se proporcionará con el formato del formulario siguiente.

Declaración de constricción de PDU			
Nombre de la constricción : *ConsId&ParList*			
Grupo : *[PDU_ConstraintGroupReference]*			
Tipo de PDU : *PDU_Identifier*			
Trayecto de derivación : *[DerivationPath]*			
Nombre de la regla de codificación : *[EncodingRuleIdentifier]*			
Variación de codificación : *[EncVariationCall]*			
Comentarios : *[FreeText]*			
Nombre del campo	**Valor del campo**	**Codificación del tipo**	**Comentarios**
⋮ *PDU_FieldIdOrMacro* ⋮	⋮ *ConstraintValue&Attributes* ⋮	⋮ *[PDU_FieldEncodingCall]* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario 39: Declaración de constricción de PDU

Las columnas de Nombre del campo y de Valor del campo estarán presentes o ausentes a la vez. La columna de Codificación del campo no estará presente.

EJEMPLO 52 – Constricción, denominada C1, impuesta a la PDU denominada PDU_A:

Declaración de constricción de PDU		
Nombre de la constricción : C1		
Tipo de PDU : PDU_A		
Trayecto de derivación :		
Comentarios :		
Nombre del campo	**Valor del campo**	**Comentarios**
FIELD1 FIELD2 FIELD3	(4 .. INFINITY) TRUE "A STRING"	

13.5 Parametrización de constricciones

Las constricciones se pueden parametrizar con una lista de parámetros formales. Los parámetros reales se pasan a una constricción desde una referencia a constricciones de una descripción de comportamiento.

EJEMPLO 53 – Constricción parametrizada:

Declaración de constricción de PDU			
Nombre de la constricción	**:**	C2 (P1:INTEGER; P2:BOOLEAN)	
Tipo de PDU	**:**	PDU_B	
Trayecto de derivación	**:**		
Comentarios	**:**		
Nombre del campo		**Valor del campo**	**Comentarios**
FIELD1 FIELD2 FIELD3		P1 P2 "A STRING"	

13.6 Constricciones de base y constricciones modificadas

Para cada definición de tipo de ASP, PDU o CM se especificará, al menos, una constricción de base. En el caso en que una ASP o un CM no tienen parámetros o en que una PDU no tiene campos, las constricciones no son procedentes y por tanto las constricciones de base son innecesarias. Una constricción de base especifica un conjunto de valores de base o por defecto o símbolos de concordancia para cada uno de los campos definidos en la definición apropiada. Por cada PDU en concreto, puede haber un número cualquiera de constricciones de base (véanse ejemplos en el anexo F).

Cuando se especifica una constricción como una modificación de una constricción de base, todo campo que no se haya especificado de nuevo en la constricción modificada tomará los valores por defecto o símbolos de concordancia especificados en la constricción de base. El nombre de la constricción modificada será un identificador exclusivo. El nombre de la constricción de base que vaya a ser modificada se indicará en la entrada del trayecto de derivación en el encabezamiento de la constricción. Esta entrada se dejará en blanco en el caso de una constricción de base. Una constricción modificada puede ser modificada a su vez. En tal caso, el trayecto de derivación indica la concatenación de nombres de las constricciones de base y modificada previamente, separadas por puntos (.). El nombre de la última constricción modificada irá seguido por un punto. Las reglas para construir una constricción modificada a partir de una constricción de base son las siguientes:

- Si en la constricción modificada no están especificados un parámetro o campo y su valor o símbolo de concordancia correspondiente, se utilizará el valor o símbolo de concordancia de la constricción progenitora (es decir, se hereda el valor).
- Si en la constricción modificada están especificados un parámetro o campo y su valor o símbolo de concordancia correspondiente, el valor o símbolo de concordancia especificado sustituye al contenido en la constricción progenitora.

13.7 Listas de parámetros formales en las constricciones modificadas

Si se establece que una constricción de base tiene una lista de parámetros formales, se aplican las normas que siguen a todas las constricciones modificadas derivadas de esa constricción de base, tanto si se han derivado en uno o en varios pasos de modificación como si no ha sido así:

a) La constricción modificada tendrá la misma lista de parámetros que la constricción de base. No habrá, en particular, parámetros omitidos o añadidos a esta lista.

b) Para cada constricción modificada, el nombre de la constricción irá seguido de la lista de parámetros formales.

c) Los parámetros de ASP o PDU parametrizados de los campos de una constricción de base, no serán modificados ni explícitamente omitidos en una constricción modificada.

13.8 Declaraciones de constricciones de CM

Los valores de campo para constricciones de CM se proporcionarán con el formato del formulario siguiente:

Declaración de constricción de CM		
Nombre de la constricción :	*ConsId&ParList*	
Grupo :	*[CM_ConstraintGroupReference]*	
Tipo de CM :	*CM_Identifier*	
Trayecto de derivación :	*[DerivationPath]*	
Comentarios :	*[FreeText]*	
Nombre del parámetro	**Valor del parámetro**	**Comentarios**
⋮ *CM_ParIdOrMacro* ⋮	⋮ *ConstraintValue&Attributes* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*		

Formulario 40: Declaración de constricción de CM

Las columnas de Nombre del parámetro y de Valor del parámetro estarán presentes o ausentes a la vez.

Este formulario se utiliza para mensajes CM del mismo modo que se utiliza el formulario de la declaración de constricción de PDU para las PDU.

14 Especificación de constricciones mediante ASN.1

14.1 Introducción

En esta cláusula se describe un método de especificación de constricciones en ASN.1 de tipo, ASP y PDU similar a la definición de las constricciones en forma tabular. Se amplía la declaración de valor ASN.1 normal de modo que sea posible el uso de mecanismos de concordancia. Se definen también mecanismos con los que sustituir u omitir partes de constricciones en ASN.1 para utilizarlas en constricciones modificadas.

En otros aspectos, ASN.1 se utiliza en las constricciones del mismo modo que en los tipos. En particular,

a) para identificadores dentro de una constricción en ASN.1 no se utilizará el símbolo de guión ("-"); en su lugar se puede utilizar el símbolo de subrayado ("_");

b) las constricciones en ASN.1 no utilizarán referencias de valores externos como se define en la Rec. UIT-T X.680;

c) se pueden hacer comentarios en ASN.1 dentro del cuerpo del cuadro. La columna de Comentarios no deberá estar presente en este cuadro. Los comentarios en ASN.1 empiezan con "--" y finalizan con la siguiente ocurrencia de "--" o con "end of line", el primero que aparezca. Esto evita que un comentario único en ASN.1 se extienda varias líneas. Se recomienda utilizar los especificadores de ATS para facilitar el intercambio de ATS en TTCN.MP y cerrar siempre los comentarios en ASN.1 con "--".

14.2 Declaraciones de constricciones de tipo en ASN.1

Tanto las constricciones de ASP en ASN.1 como las constricciones de PDU en ASN.1, pueden estructurarse con referencias a constricciones de tipo de series de pruebas en ASN.1, para valores de campos complejos. Los tipos de series de pruebas en ASN.1 se definen en la parte declaraciones de las ATS.

Para cada declaración de constricción de tipo ASN.1 se deberá suministrar la siguiente información:

a) el nombre de la constricción,

   que puede ir seguido de una lista de parámetros formales optativos;

b) el nombre del tipo en ASN.1,

c) el trayecto de derivación (véase 13.6 y 14.6),

   a fin de especificar variaciones de codificación explícitas para constricciones de tipo en ASN.1 enteras, que sustituyan tanto a las variaciones de codificación de la constricción de PDU que referencia esta constricción de tipo en ASN.1 como a las variaciones de codificación global por defecto de la serie de pruebas, esta entrada optativa deberá referenciar una entrada en el cuadro de variaciones de codificación pertinente (por ejemplo, cambiar de SD a LD(3)). Si no se utiliza esta entrada, las variaciones de codificación por defecto se aplican entonces a todas las constricciones de tipo en ASN.1 de este tipo, a menos que sean específicamente sustituidas dentro de una constricción particular;

d) las variaciones de codificación que se han de utilizar para la constricción,

   si una declaración de constricción en ASN.1 es una modificación de una constricción ASN.1 existente, el nombre de la constricción ASN.1 que se toma como base de esta modificación deberá referenciarse en la entrada trayecto de derivación del cuadro;

e) el valor de la constricción,

   donde el cuerpo del cuadro de constricciones de tipo en ASN.1 contiene la declaración de constricción en ASN.1 con atributos optativos; todos los valores y atributos de la constricción definidos en 12.6 se pueden utilizar en ASN.1.

A fin de especificar codificaciones explícitas para valores individuales dentro de una constricción de tipo en ASN.1, que sustituya a todas las variaciones de codificación de las codificaciones de constricción de tipo en ASN.1 específicas (véase c) arriba), se utiliza la palabra clave **ENC** después del valor pertinente, seguido de un identificador de la codificación específica y de una lista de parámetros reales necesarios. El identificador de codificación deberá identificar una de las variaciones de codificación o una definición de codificación de campo no válida definida en la serie de pruebas.

Las declaraciones de constricciones de tipo en ASN.1 se especificarán con el formato del formulario siguiente:

Declaración de constricción de tipo en ASN.1		
Nombre de la constricción	**:**	*ConsId&ParList*
Grupo	**:**	*[ASN1_TypeConstraintGroupReference]*
Tipo estructurado	**:**	*ASN1_TypeIdentifier*
Trayecto de derivación	**:**	*[DerivationPath]*
Variación de codificación	**:**	*[EncVariationCall]*
Comentarios	**:**	*[FreeText]*
Valor de constricción		
ConstraintValue&AttributesOrReplace		
Comentarios detallados:	*[FreeText]*	

Formulario 41: Declaración de constricción de tipo en ASN.1

Este formulario se utiliza para tipos en ASN.1 del mismo modo que se utiliza el formulario de declaración de constricción del PDU en ASN.1 (véase 14.4).

14.3 Declaraciones de constricciones de ASP en ASN.1

Para cada declaración de constricción de ASP en ASN.1 se facilitará la siguiente información:

a) el nombre de la constricción,

   que puede ir seguido de una lista de parámetros formales optativos;

b) el nombre de tipo de la ASP,

c) el trayecto de derivación (véanse 13.6 y 14.6),

   si una declaración de constricción en ASN.1 es una modificación de una constricción en ASN.1 existente, el nombre de la constricción en ASN.1 adoptado como base de esta modificación se referenciará en el cuadro, en la entrada del trayecto de derivación;

d) el valor de la constricción,

   que contiene el cuerpo del cuadro de constricción de la ASP, la declaración de constricción en ASN.1 con atributos optativos. En las constricciones en ASN.1 Se pueden usar todos los valores y atributos de la constricción definidos en 12.6.

Las declaraciones de constricciones de ASP en ASN.1 se especificarán con el formato del formulario siguiente.

Declaración de constricción de ASP en ASN.1		
Nombre de la constricción	**:**	*ConsId&ParList*
Grupo	**:**	*[ASN1_ASP_ConstraintGroupReference]*
Tipo de ASP	**:**	*ASP_Identifier*
Trayecto de derivación	**:**	*[DerivationPath]*
Comentarios	**:**	*[FreeText]*
Valor de constricción		
ConstraintValue&AttributesOrReplace		
Comentarios detallados: *[FreeText]*		

Formulario 42: Declaración de constricción de ASP en ASN

Este formulario se utiliza para las ASP en ASN.1 del mismo modo que se utiliza el formulario de declaración de constricción de PDU en ASN.1 (véase 14.4).

14.4 Declaraciones de constricciones de PDU en ASN.1

Para cada declaración de constricción de PDU en ASN.1 se facilitará la siguiente información:

a) el nombre de la constricción,

   que puede ir seguido de una lista de parámetros formales optativos;

b) el nombre de tipo de la PDU;

c) el trayecto de derivación (véanse 13.6 y 14.6);

d) las reglas de codificación que se utilizarán para la constricción,

   a fin de especificar las codificaciones explícitas para constricción de PDU en ASN.1 enteras, que sustituyan a reglas de codificación global por defecto para la serie de pruebas, esta entrada optativa deberá hacer referencia a una entrada en el cuadro de definiciones de codificación pertinente (por ejemplo, cambiar de BER a DER). Si no se utiliza esta entrada, se aplicarán entonces las reglas de codificación por defecto a todas las constricciones de tipo de PDU en ASN.1 de este tipo, a menos que se haya sustituido específicamente en una constricción particular;

e) las variaciones de codificación que se utilizarán para la constricción,

a fin de especificar las variaciones de codificación explícitas para constricción de PDU en ASN.1 enteras, que sustituyan a las variaciones de codificación global por defecto para la serie de pruebas, esta entrada optativa deberá hacer referencia a una entrada en el cuadro de variaciones de codificación pertinente [por ejemplo, cambiar de SD a LD(3)]; si no se utiliza esta entrada, se aplican entonces las variaciones de codificación por defecto a todas las constricciones de tipo de PDU en ASN.1 de este tipo, a menos que se haya sustituido en una constricción particular;

si una declaración de constricción en ASN.1 es una modificación de una constricción en ASN.1 existente, el nombre de la constricción en ASN.1 que se toma como base de esta modificación será referenciado en la entrada del trayecto de derivación;

f) el valor de la constricción,

que contiene el cuerpo del cuadro de constricción de la PDU, la declaración de constricción en ASN.1 con atributos optativos. En las constricciones en ASN.1 Se pueden usar todos los valores y atributos de la constricción definidos en 12.6.

A fin de especificar codificaciones explícitas para valores individuales dentro de una constricción de PDU en ASN.1, que sustituya a las reglas de codificación global por defecto o a las codificaciones de constricción de PDU en ASN.1 específicas [véanse c) y d) anteriormente], se utiliza la palabra clave **ENC** después del valor pertinente, seguida de un identificador de la codificación específica y de una lista de parámetros reales necesaria. El identificador de codificación deberá identificar una de las variaciones de codificación o una definición de codificación de campo no válida definida en la serie de pruebas.

Las declaraciones de constricciones de PDU se especificarán con el formato del formulario siguiente.

Declaración de constricción de PDU en ASN.1		
Nombre de la constricción	**:**	*ConsId&ParList*
Grupo	**:**	*[ASN1_PDU_ConstraintGroupReference]*
Tipo de PDU	**:**	*PDU_Identifier*
Trayecto de derivación	**:**	*[DerivationPath]*
Nombre de la regla de codificación	**:**	*[EncodingRuleIdentifier]*
Variación de codificación	**:**	*[EncVariationCall]*
Comentarios	**:**	*[FreeText]*
Valor de constricción		
ConstraintValue&AttributesOrReplace		
Comentarios detallados: *[FreeText]*		

Formulario 43: Declaración de constricción de PDU en ASN.1

14.5 Constricciones en ASN.1 parametrizadas

Las constricciones en ASN.1 se pueden parametrizar (véase 13.5).

14.6 Constricciones en ASN.1 modificadas

Se pueden especificar constricciones en ASN.1 modificando una constricción en ASN.1 existente. En el proceso de creación de una nueva constricción es posible mantener partes de una constricción con el mecanismo REPLACE/OMIT (reemplazar/omitir).

Los parámetros o campos concretos de una constricción de base o modificada, pueden identificarse mediante una lista de selectores de campo, para sustituir su valor definido por uno nuevo u omitir el valor definido. Una ReferenceList consta de los identificadores de selector de campo (definidos en la definición de tipo correspondiente), separados por puntos, que identifican inequívocamente un campo determinado (posiblemente estructurado) dentro de una PDU (o una ASP). Los campos de primer nivel pueden identificarse mediante un selector único, en tanto que los campos anidados requieren el trayecto completo.

Se utilizarán valores de reemplazar sólo en el caso en que se especifique un trayecto de derivación. Se utilizarán valores completos en ASN.1 sólo en el caso en que no se especifique un trayecto de derivación. Los valores REEMPLAZADOS u OMITIDOS pueden ser estructurados.

Si un campo pertenece a una estructura SEQUENCE, SET o CHOICE, se puede utilizar la posición del campo entre paréntesis para reemplazar al identificador de selector de campo. Este método se utilizará cuando en la declaración del campo no se incluya el identificador.

14.7 Listas de parámetros formales en las constricciones en ASN.1 modificadas

Los requisitos de 13.7 se aplican, también, a las constricciones en ASN.1 modificadas.

14.8 Nombres de parámetro ASP y campo de PDU en las constricciones en ASN.1

Cuando se especifique una constricción para una ASP o una PDU en ASN.1, el parámetro o los identificadores de campo definidos en la definición de tipo en ASN.1 para los tipos SEQUENCE, SET y CHOICE podrán utilizarse para identificar los parámetros o campos específicos de la ASP o la PDU representados por un valor. En el caso de tipos CHOICE, deberán emplearse los identificadores que identifican la variante. Para los tipos SEQUENCE, deberán utilizarse identificadores de parámetro o campo siempre que la definición de valor resulte ambigua, debido a los valores omitidos de los parámetros o campos OPTIONAL. En el caso de tipos SET, los identificadores de campo o de parámetro deberán utilizarse siempre.

EJEMPLO 54 – Valores de campo en una constricción de PDU en ASN.1:

Supóngase la definición de tipo:

Declaración de tipo PDU en ASN.1
Nombre de la PDU : XY_PDU **Tipo de PCO** : **Comentarios** :
Definición de tipo
SET { field_1 INTEGER OPTIONAL, field_2 BOOLEAN, field_3 INTEGER OPTIONAL, field_4 INTEGER OPTIONAL }

Entonces, una posible constricción es:

Declaración de la constricción de PDU en ASN.1
Nombre de la constricción : CONS1 **Tipo de PDU** : XY_PDU **Trayecto de derivación** : **Comentarios** :
Valor de la constricción
{ field_1 5, field_2 TRUE, field_3 3 } *-- field_4 is not specified => omitted when sending --* *-- if identifier field_3 was not used it would be ambiguous whether 3 was the value of --* *-- field_3 or field_4, since both are OPTIONAL. --*

14.9 Declaraciones de constricción de CM en ASN.1

Los valores de parámetro para las constricciones de CM se especificarán con el formato del formulario siguiente:

Declaración de constricción de CM en ASN.1		
Nombre de la constricción	**:**	*ConsId&ParList*
Grupo	**:**	*[ASN1_CM_ConstraintGroupReference]*
Tipo de CM	**:**	*CM_Identifier*
Trayecto de derivación	**:**	*[DerivationPath]*
Comentarios	**:**	*[FreeText]*
Valor de la constricción		
ConstraintValue&AttributesOrReplace		
Comentarios detallados: *[FreeText]*		

Formulario 44: Declaración de constricción de CM en ASN.1

Este formulario se utiliza para los CM del mismo modo que se utiliza la declaración de constricción de PDU para las PDU.

15 La parte dinámica

15.1 Introducción

La parte dinámica contiene el cuerpo principal de la serie de pruebas, es decir las descripciones de caso de prueba, de paso de prueba y de comportamiento por defecto.

15.2 Comportamiento dinámico de caso de prueba

15.2.1 Especificación del cuadro de comportamiento dinámico de caso de prueba

El título del cuadro es "Comportamiento dinámico de caso de prueba".

El encabezamiento contendrá la siguiente información:

a) nombre del caso de prueba,

que dará un identificador exclusivo para el caso de prueba descrito en el cuadro;

b) referencia al grupo de prueba,

que contendrá el nombre completo del nivel más bajo del grupo que contiene el caso de prueba. Ese nombre completo será conforme con los requisitos de 9.2 y finalizará con el carácter barra inclinada (/);

c) finalidad de la prueba,

enunciado informal de la finalidad del caso de prueba, según se indique en la norma pertinente (si existe) sobre finalidades de la prueba y estructuras de las series de pruebas pertinentes o en la parte equivalente de la norma de serie de pruebas (si existe);

d) referencia a comportamiento por defecto,

una lista de identificadores de descripción de comportamiento por defecto (incluida una lista de parámetros reales, si es necesario) de una descripción de comportamiento por defecto, si existe, que se aplica a la descripción del comportamiento del caso de prueba (véase 15.4).

El cuerpo del cuadro contendrá las siguientes columnas con la información correspondiente:

a) una columna (optativa) con el número de línea (véase 15.2.5),

que, de estar presente, deberá estar situada en el lado izquierdo del cuadro;

b) una columna de etiqueta,

en la que se colocarán etiquetas para identificar los enunciados en TTCN que permitan efectuar bifurcaciones con el constructivo GOTO (véase 15.14);

c) una descripción de comportamiento,

que describe el comportamiento del LT y/o el UT, en términos de enunciados en TTCN y sus parámetros, con la notación arborescente (véase 15.6);

d) una columna de referencias de constricciones,

en la que se sitúan las referencias a constricciones para asociar los enunciados TTCN de un árbol de comportamiento con referencia a valores de ASP y/o PDU específicos definidos en la parte constricciones (véase la cláusula 12);

e) una columna de veredicto,

donde se sitúa la información de resultado o de veredicto en asociación con enunciados en TTCN del árbol de comportamiento (véase 15.17);

f) una columna de comentarios (optativa),

que se utiliza para insertar comentarios que faciliten la comprensión de los enunciados en TTCN, y las notas breves o referencias al texto adicional se proporcionan en el campo de comentarios detallados optativo.

Las columnas c), d), e) y f) se situarán en ese orden de izquierda a derecha. Se recomienda que la columna de etiqueta obligatoria se sitúe a la izquierda de la descripción de comportamiento. Como alternativa, puede colocarse a la derecha de la descripción de comportamiento.

Un pie (optativo) puede contener comentarios detallados.

15.2.2 El formulario de comportamiento dinámico de caso de prueba

El comportamiento dinámico de caso de prueba se proporcionará con el formato del formulario siguiente.

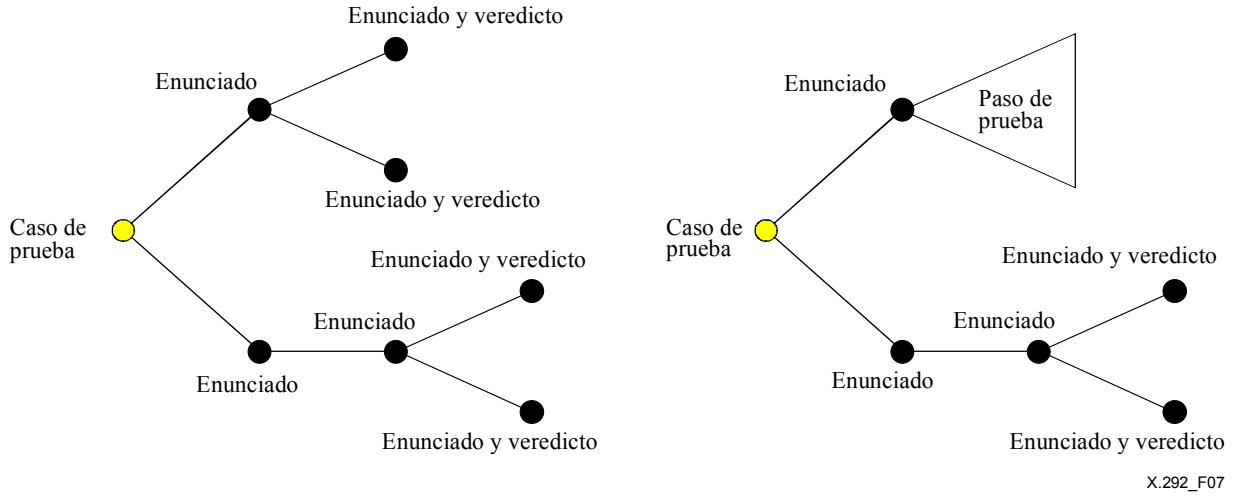
Comportamiento dinámico de caso de prueba					
Nombre del caso de prueba : *TestCaseIdentifier*					
Grupo : *TestGroupReference*					
Finalidad : *FreeText*					
Configuración : *TCompConfigIdentifier*					
Valores por defecto : *[DefaultRefList]*					
Comentario : *[FreeText]*					
N.º	**Etiqueta**	**Descripción de comportamiento**	**Ref. de constricciones**	**Veredicto**	**Comentarios**
1 *2* . . . . *n*	. . *[Label]*	. . *StatementLine* . . . *TreeHeader* *StatementLine* . .	. . *[ConstraintReference]*	. . *[Verdict]*	. . *[FreeText]* . . . .
Comentarios detallados: *[FreeText]*					

Formulario 45: Comportamiento dinámico de caso de prueba

Los encabezamientos de columna de este formulario se pueden abreviar así (L) **E**, (Cref) **RefC**, **V** y **C**. Esto permite que la columna del árbol de comportamiento sea lo más amplia posible en aquellos casos en que existan limitaciones físicas debidas al papel.

15.2.3 Estructura del comportamiento de caso de prueba

Cada caso de prueba contiene una descripción precisa de las secuencias de eventos (anticipados) y veredictos conexos. Esta descripción se estructura en forma de árbol, en el que los nodos son enunciados en TTCN y las asignaciones de veredicto son las hojas. En muchos casos es más eficaz utilizar pasos de prueba como un medio para subestructurar este árbol:



1: Comportamiento de caso de prueba no estructurado **2: Comportamiento de caso de prueba estructurado**

Figura 7/X.292 – Estructura del comportamiento de caso de prueba

En TTCN se expresa esta modularización explícita con pasos de prueba y el constructivo ATTACH.

15.2.4 Descripción del comportamiento de caso de prueba concurrente

Cuando se emplean PTC en un caso de prueba, el encabezamiento deberá contener la entrada adicional, Configuración, que identificará una configuración de componente de prueba declarada en la parte declaración.

El comportamiento del MTC se describe mediante el primer árbol en el cuadro de comportamiento de caso de prueba, más todos los árboles adjuntados. El árbol de comportamiento de MTC crea los PTC cuando es necesario, y asocia cada PTC con su propio árbol de comportamiento.

Si un comportamiento de PTC se especifica como árbol local en el comportamiento de caso de prueba, la referencia por defecto deberá estar vacía. Esta restricción impide que un PTC herede el comportamiento por defecto del MTC.

Un caso de prueba solamente utilizará los componentes de prueba que estén presentes en la configuración de comportamiento de prueba referenciada. La configuración elegida determinará el conjunto de PCO y CP que se pueden usar en el caso de prueba. Cuando se utilice la entrada Configuración en el encabezamiento del comportamiento dinámico de caso de prueba, deberá proporcionarse en el formato que se muestra en el formulario 45.

15.2.5 Numeración y continuación de líneas

Al imprimir las líneas de una descripción de comportamiento, éstas pueden resultar demasiado largas para estar contenidas en un renglón, por lo que es necesario utilizar símbolos adicionales que indiquen la extensión de una sola línea de comportamiento. Se dispone de dos métodos:

a) Indicar el comienzo de una nueva línea de comportamiento. Para ello se añade una columna de línea adicional como columna situada más a la izquierda en el cuerpo del cuadro. En esta columna existirá una sola entrada aplicable a los renglones en los que se inicie una nueva línea de comportamiento. Los números de línea utilizados serán 1, 2, 3, ... no debiendo reiniciarse la numeración cuando se definan árboles locales; es decir, existirá un número de línea exclusivo para cada línea de comportamiento del árbol de comportamiento.

   NOTA 1 – Se puede utilizar números de línea a efectos de registro cronológico para registrar inequívocamente qué línea de comportamiento ha sido ejecutada.

   NOTA 2 – Los números de línea se pueden usar como referencias en la sección de comentarios detallados.

b) Indicar la continuación de las líneas. Si una línea debe continuarse dentro de una columna de descripción de comportamiento se utilizará el símbolo número (#) situándolo en la posición más a la izquierda de la línea de comportamiento sobre la línea del texto continuado. Se recomienda que el texto de la parte continuada adopte el mismo nivel de sangrado que la línea a la que continúa.

Si una línea es continuada en cualquier columna distinta de la columna de descripción de comportamiento, no es necesario utilizar el símbolo de número.

EJEMPLO 55 – Impresión de líneas de comportamiento largas:

Estilo recomendado:

N.º	Etiqueta	Descripción de comportamiento	Ref. de constricciones	Veredicto	Comentarios
1		Enunciado en TTCN demasiado largo para su impresión en un único renglón porque la columna es demasiado estrecha	Ref1		
2		Línea del enunciado siguiente	Referencia a constricción demasiado larga para su impresión en un renglón		
3		Línea de enunciado alternativa	Ref2		

Estilo alternativo:

Etiqueta	Descripción de comportamiento	Ref. de constricciones	Veredicto	Comentarios
	Enunciado en TTCN demasiado largo para su impresión en un único renglón porque la columna es demasiado estrecha	Ref1		
	Línea del enunciado siguiente	Referencia a constricción demasiado larga para su impresión en un renglón		
	Línea de enunciado alternativa	Ref2		

15.3 Comportamiento dinámico de paso de prueba

15.3.1 Especificación del cuadro de comportamiento dinámico de los pasos de prueba

El comportamiento dinámico de los pasos de prueba se define con el mismo método de los casos de prueba, con la excepción de que los pasos de prueba se pueden parametrizar (véase 15.7). Los cuadros de comportamiento dinámico de paso de prueba son idénticos a los cuadros de comportamiento dinámico de caso de prueba, con las siguientes diferencias:

a) el cuadro tiene como título "Comportamiento dinámico de paso de prueba";

b) el primer elemento del encabezamiento es el nombre de paso de prueba,

que es un identificador exclusivo del paso de prueba, que va seguido de una lista optativa de parámetros formales y de sus tipos asociados. Estos parámetros Se pueden usar para pasar PCO, constricciones u otros objetos de datos al árbol raíz del paso de prueba;

c) el segundo elemento del encabezamiento es la referencia a grupo de pasos de prueba,

que proporciona el nombre completo del nivel más bajo del grupo biblioteca de pasos de prueba que contiene ese paso de prueba. Dicho nombre completo deberá ajustarse a los requisitos de referencias de grupo de pasos de prueba (véase 9.3) y finalizar con el carácter (/);

d) el tercer elemento del encabezamiento es el objetivo de paso de prueba,

que es una declaración informal del paso de prueba buscado.

15.3.2 El formulario de comportamiento dinámico de paso de prueba

El comportamiento dinámico del paso de prueba se proporcionará con el formato del formulario siguiente.

Comportamiento dinámico del paso de prueba					
Nombre del paso de prueba	**:**	*TestStepId&ParList*			
Grupo	**:**	*TestStepGroupReference*			
Objetivo	**:**	*FreeText*			
Valores por defecto	**:**	*[DefaultRefList]*			
Comentarios	**:**	*[FreeText]*			
N.º	**Etiqueta**	**Descripción de comportamiento**	**Ref. de constricciones**	**Veredicto**	**Comentarios**
1	.	.	.	.	.
2	.	.	.	.	.
.	*[Label]*	*StatementLine*	*[ConstraintReference]*	*[Verdict]*	*[FreeText]*
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	*TreeHeader*	.	.	.
.	.	*StatementLine*	.	.	.
.	.	.	.	.	.
n	.	.	.	.	.
Comentarios detallados:	*[FreeText]*				

Formulario 46: Comportamiento dinámico del paso de prueba

Los encabezamientos de columna de este formulario se pueden abreviar así: **E**, **RefC**, **V** y **C**.

15.4 Comportamiento dinámico por defecto

15.4.1 Comportamiento por defecto

Un caso de prueba en TTCN especificará el comportamiento alternativo para *cada* evento posible (incluyendo los que no sean válidos). Ocurre a menudo que en un árbol de comportamiento, cada serie de alternativas finaliza en el mismo comportamiento. Dicho comportamiento puede sacarse (a modo de factor común) como comportamiento por defecto de ese árbol. Esas descripciones de comportamiento por defecto se colocan en la biblioteca de valores por defecto global.

El comportamiento dinámico de los valores por defecto se define con los mismos mecanismos que para los pasos de prueba, con excepción de las siguientes restricciones:

a) no está permitida la especificación de comportamiento por defecto del comportamiento por defecto;

b) la descripción del comportamiento por defecto puede adjuntar árboles locales (véase 15.7.1) pero no deberá adjuntar pasos de prueba;

c) si se emplean árboles locales en la descripción del comportamiento por defecto, no adjuntarán pasos de prueba;

d) el árbol (o árboles) en la descripción del comportamiento no utilizarán la operación ACTIVATE (véase 15.18.4).

Los PCO y otros parámetros reales pueden pasarse a descripciones de comportamiento por defecto de la misma forma en que pueden pasarse a pasos de prueba. Las normas aplicables al ámbito y a la sustitución textual de estos parámetros son las mismas que las correspondientes a la adjunción de árbol (véase 15.13).

15.4.2 Especificación del cuadro de comportamiento dinámico por defecto

Los cuadros de comportamiento dinámico por defecto son idénticos a los cuadros de comportamiento dinámico de pasos de prueba, con las siguientes diferencias:

a) el cuadro se titula "Comportamiento dinámico por defecto";

b) el primer elemento del encabezamiento es el nombre por defecto,

que es un identificador exclusivo del comportamiento por defecto, que va seguido de una lista optativa de parámetros formales y de sus tipos asociados. Estos parámetros Se pueden usar para pasar PCO, constricciones u otros objetos de datos al árbol raíz del comportamiento por defecto;

c) el segundo elemento del encabezamiento es la referencia a grupo de valores por defecto,

que proporciona el nombre completo del nivel más bajo del grupo de valores por defecto que contiene el comportamiento por defecto. Ese nombre completo se ajustará a los requisitos de la referencia a grupo de valores por defecto (véase 9.4) y finalizar con el carácter (/);

d) el tercer elemento del encabezamiento es el objetivo por defecto,

que es una declaración informal del objetivo del comportamiento por defecto.

15.4.3 El formulario de comportamiento dinámico por defecto

El comportamiento dinámico por defecto se proporcionará con el formato del formulario siguiente.

Comportamiento dinámico por defecto					
Nombre por defecto : *DefaultId&ParList*					
Grupo : *DefaultGroupReference*					
Objetivo : *FreeText*					
Comentarios : *[FreeText]*					
N.º	**Etiqueta**	**Descripción de comportamiento**	**Ref. de constricciones**	**Veredicto**	**Comentarios**
1 *2* *.* *.* *.* *.* *n*	. . *[Label]*	. . *StatementLine* . . *TreeHeader* *StatementLine* .	. . *[ConstraintReference]*	. . *[Verdict]*	. . *[FreeText]* . . . .
Comentarios detallados: *[FreeText]*					

Formulario 47: Comportamiento dinámico por defecto

Los encabezamientos de columna de este formulario se pueden abreviar así: **E**, **RefC**, **V** y **C**.

15.5 La descripción de comportamiento

La columna de descripción de comportamiento de un cuadro de comportamiento dinámico contiene la especificación de las combinaciones de enunciados en TTCN que considera posibles el especificador de la serie de pruebas. Se denomina árbol de comportamiento al conjunto de esas combinaciones. Cada enunciado en TTCN es un nodo del árbol de comportamiento.

15.6 La notación arborescente

Cada enunciado en TTCN se representará en una línea de enunciado separada. Los enunciados pueden relacionarse entre sí de dos maneras posibles:

- como secuencias de enunciados en TTCN;
- como enunciados en TTCN alternativos.

Las secuencias de enunciados en TTCN se representan en líneas de enunciado sucesivas, estando cada nuevo enunciado en TTCN sangrado una vez, de izquierda a derecha, con respecto al anterior.

EJEMPLO 56 – Enunciados en TTCN en secuencia:

```
EVENT_A
    CONSTRUCT_B
        EVENT_C
```

Los enunciados situados al mismo nivel de sangrado y que pertenecen al mismo nodo predecesor constituyen los enunciados alternativos que se pueden producir en ese momento. En consecuencia, este conjunto de enunciados en TTCN se definirá como *conjunto de alternativas* o, simplemente *alternativas*.

EJEMPLO 57 – Enunciados en TTCN alternativos:

```
CONSTRUCT_A1
STATEMENT_A2
EVENT_A3
```

EJEMPLO 58 – Combinación de secuencias y de alternativas para construir un árbol:

```
EVENT_A
    CONSTRUCT_B
        EVENT_C
    STATEMENT_D1
    EVENT_D2
```

El que la evaluación de un enunciado en TTCN sea exitosa o no depende de diversas condiciones asociadas a la línea de enunciado. Tales condiciones no tienen por qué ser mutuamente excluyentes, es decir, es posible que, en un momento dado, pueda evaluarse exitosamente más de una línea de enunciado. Como las líneas de enunciado se evalúan según el orden de su aparición en el conjunto de alternativas, el primer enunciado que cumpla una condición se evaluará como exitoso. Esto puede conducir a un comportamiento inalcanzable, en especial si los enunciados se codifican como alternativas siguiendo a enunciados cuya evaluación es siempre exitosa.

REPEAT y GOTO son siempre exitosos. Adicionalmente, SEND, IMPLICIT SEND, asignaciones y operaciones de temporizador son exitosos siempre que el calificador acompañante, si existe, tome el valor TRUE.

El sangrado gráfico de líneas de enunciado en la TTCN.GR se corresponde con valores de sangrado en TTCN.MP. Los enunciados del primer nivel de alternativas que no tengan predecesores en el árbol raíz o local al que pertenezca, tendrán un valor de sangrado igual a 0. Los enunciados que contengan predecesor deberán tener un valor de sangrado igual al valor de sangrado del predecesor incrementado en uno.

15.7 Nombres de árbol y lista de parámetros

15.7.1 Introducción

Cada descripción de comportamiento contendrá, al menos, un árbol de comportamiento. Para poder hacer referencia inequívoca a esos árboles (como en un constructivo ATTACH), cada árbol deberá poseer un nombre de árbol.

El primer árbol que aparece en una descripción de comportamiento se denomina árbol raíz. El nombre de un árbol raíz es el identificador que figura en el encabezamiento de su cuadro de comportamiento dinámico. Esto es, el nombre de árbol del árbol raíz de un paso de prueba es el identificador de paso de prueba para dicho paso de prueba, y del mismo modo para los árboles raíz de los comportamientos dinámicos del caso de prueba y comportamientos dinámicos por defecto.

Los árboles distintos del árbol raíz que aparecen dentro de los cuadros de comportamiento dinámico se denominan árboles locales. A los árboles locales les antecede un prefijo en forma de encabezamiento de árbol que contiene el nombre del árbol.

15.7.2 Árboles con parámetros

Todos los árboles se pueden parametrizar, con excepción de los árboles raíz de caso de prueba. Los parámetros pueden proporcionar PCO, constricciones, variables y elementos similares para su utilización dentro del árbol. Los árboles raíz de caso de prueba no podrán estar parametrizados.

Si un árbol es parametrizado (es decir, si tiene parámetros), inmediatamente después del nombre del árbol deberá aparecer, entre paréntesis, una lista de los parámetros formales y sus tipos. Por ejemplo, la lista de parámetros formales de un árbol raíz de paso de prueba deberá figurar entre paréntesis inmediatamente después del identificador del paso de prueba, en el encabezamiento del cuadro de comportamiento dinámico del paso de prueba. De forma análoga, la lista de parámetros formales de un árbol local deberá aparecer en el encabezamiento del árbol, inmediatamente después del nombre del árbol.

En la construcción de la lista de parámetros formales, cada parámetro formal deberá ir seguido de una coma y del nombre del tipo de parámetro formal. Si hay más de un parámetro formal del mismo tipo, tales parámetros podrán combinarse en una sublista. Cuando se utilice una sublista, los parámetros formales contenidos en la sublista deberán estar separados entre sí por comas. El parámetro formal final de la sublista deberá ir seguido por el carácter dos puntos (:) y por el tipo de parámetro formal.

Cuando haya más de un par parámetro formal y tipo (o más de un par sublista y tipo), los pares irán separados entre sí por el carácter punto y coma.

Los parámetros formales pueden ser de tipo PCO, ASP, PDU, estructura o de uno de los restantes tipos predefinidos o tipos de series de pruebas.

Si el parámetro formal de un árbol es un tipo PDU, los campos específicos de la PDU no serán referenciados en el árbol. Si el parámetro formal es un identificador de una PDU específica, podrán referenciarse en el árbol campos específicos de la PDU.

EJEMPLO 59 – Vacío

EJEMPLO 60 – Paso de prueba que utiliza parámetros formales: EXAMPLE_TREE (L:TSAP; X:INTEGER; Y:INTEGER)

EJEMPLO 61 – Paso de prueba que utiliza parámetros formales con una sublista: EXAMPLE_TREE (L:TSAP; X, Y:INTEGER)

15.8 Enunciados en TTCN

La notación en forma de árbol permite la especificación de eventos de prueba iniciados por el probador (o probadores) inferior(es) o el probador (o probadores) superior(es) (eventos SEND e IMPLICIT SEND), eventos de prueba recibidos por el probador (o probadores) inferiores o el probador (o probadores) superiores (RECEIVE, OTHERWISE, TIMEOUT y DONE), constructivos (GOTO, ATTACH, REPEAT, CREATE, RETURN y ACTIVATE), y seudoeventos que comprenden combinaciones de calificadores, asignaciones y operaciones de temporizador. Todo este conjunto se denomina, colectivamente, enunciados en TTCN.

Los eventos de prueba pueden ir acompañados de calificadores (expresiones booleanas), asignaciones y operaciones de temporizador. Los calificadores, asignaciones y operaciones de temporizador pueden, asimismo, ser autónomos, en cuyo caso se denominan seudoeventos.

15.9 Eventos de prueba en TTCN

15.9.1 Eventos de enviar y recibir

La TTCN soporta la iniciación (envío) de ASP y PDU a PCO denominados y la aceptación (recepción) de ASP y PDU en PCO denominados. El modelo de PCO se define en 11.10 y 15.9.5.3. La TTCN concurrente soporta la emisión y recepción de CM a CP denominados. El modelo de CP se define en 11.11.

En su forma más simple, un identificador de ASP o un identificador de PDU sigue al símbolo SEND (!) para aquellos eventos que hayan de ser iniciados por el LT o el UT, o al símbolo RECEIVE (?) para aquellos eventos cuya aceptación es posible por parte del LT o del UT. No se proporciona el nombre de PCO optativo. Esta forma es válida cuando sólo hay un PCO en la serie de pruebas.

EJEMPLO 62 – !CONreq o ?CONind

Si en una serie de pruebas hay más de un PCO, al símbolo SEND o al símbolo RECEIVE deberá antecederle como prefijo un nombre de PCO que aparezca en la parte declaraciones o en la lista de parámetros formales del árbol. El nombre de PCO se utiliza para indicar el PCO en el que puede producirse el evento de prueba.

EJEMPLO 63 – L! CONreq o L? CONind

Cuando se trate de los CP, se utilizará el identificador de CP, el cual deberá anteceder como prefijo al símbolo SEND en el caso de emisión de un CM y deberá anteceder como prefijo al símbolo RECEIVE en el caso de recepción de un CM.

EJEMPLO 64 – A_CP!A_CM o A_CP?A_CM

15.9.2 Eventos de recibir

Una línea de evento RECEIVE evalúa exitosamente si una ASP o una PDU entrante en el PCO especificado concuerda con la línea de evento. Se produce concordancia cuando se satisfacen las siguientes condiciones:

a) La PDU entrante puede decodificarse de conformidad con las reglas de codificación aplicables.

b) La ASP o la PDU entrante es válida de conformidad con la definición de tipo de la ASP o la PDU a la que hace referencia el nombre de evento en la línea de evento. En especial, todos los valores de parámetros y/o campos deberán ser del tipo definido y satisfarán todas las restricciones de longitud especificadas.

c) La ASP o la PDU concuerda con la referencia a constricciones en la línea de evento.

d) En los casos en que se especifique un calificador en la línea de evento, el calificador adoptará el valor TRUE. El calificador puede contener referencias a parámetros de la ASP y/o campos de la PDU.

El evento entrante se retira de la cola del PCO sólo cuando concuerde correctamente con una línea de evento RECEIVE.

En la TTCN concurrente la recepción y concordancia de un CM en un CP se trata del mismo modo descrito anteriormente.

15.9.3 Eventos de enviar

Una línea de evento SEND con un calificador es exitosa si la expresión del calificador toma el valor TRUE. Los eventos SEND no calificados son siempre exitosos. La ASP o la PDU saliente resultante de un evento SEND deberá construirse como sigue:

a) Todos los parámetros de la ASP y campos de la PDU serán del tipo especificado en las definiciones correspondientes y satisfarán las restricciones de longitud que figuran en las definiciones.

b) Los valores de los parámetros de la ASP y de los campos de la PDU, se fijarán según se especifique en la constricción referenciada en la línea de evento (véanse las cláusulas 12, 13 y 14, donde se explica la construcción de las ASP o los PDU con constricciones).

c) Cualesquiera asignaciones directas de parámetros de ASP o de campos de PDU en la línea de eventos reemplazarán los valores correspondientes especificados en la constricción, si existen.

d) Todos los parámetros y/o campos de las ASP o las PDU salientes contendrán valores específicos o se omitirán explícitamente, antes de la compleción del evento SEND.

e) La PDU totalmente construida se deberá codificar de conformidad con las reglas de codificación aplicables.

La generación de un parámetro ASP o un valor de campo de PDU, tanto por constricciones como por asignaciones, que viole el tipo declarado y las constricciones de longitud, ocasionarán un error de caso de prueba.

En la TTCN concurrente la emisión de un CM en un CP se trata del mismo modo descrito anteriormente.

15.9.4 Tiempo de vida de los eventos

Los identificadores de parámetros de ASP y campos de PDU asociados con SEND y RECEIVE se utilizarán únicamente para referenciar valores de parámetros de ASP o campos de PDU en la propia línea de enunciado.

En el caso de eventos SEND podrán fijarse, si es necesario, valores apropiados de parámetros de ASP y de campos de PDU en asignaciones adecuadas en la línea de SEND.

EJEMPLO 65 – !A_PDU (A_PDU.FIELD := 3)

Los efectos de tales asignaciones no irán más allá de la línea de eventos en la que se produjeron.

En el caso de eventos RECEIVE, si es necesaria una referenciación ulterior de valores de parámetros de ASP o de campos de PDU pertinentes, se asignará a variables, en la propia línea de RECEIVE, la totalidad de las ASP o las PDU o una parte pertinente de las mismas. Tales variables podrán entonces referenciarse en líneas subsiguientes.

EJEMPLO 66 – ?A_PDU (VAR := A_PDU.FIELD)

pudiéndose utilizar VAR en líneas de evento subsiguientes a la recepción de A_PDU.

El tiempo de vida de los CM se restringe también al enunciado RECEIVE pertinente. A los identificadores de campos de CM se puede acceder de un modo similar que a los identificadores de campos de PDU.

EJEMPLO 67 – A_CP!A_CM o A_CP?A_CM

15.9.5 Ejecución del árbol de comportamiento

15.9.5.1 Introducción

El especificador de la serie de pruebas organizará el árbol de comportamiento que representa un caso de prueba o un paso de prueba con arreglo a las normas siguientes, en lo que respecta a la ejecución de las pruebas:

a) comenzando en la raíz del árbol, el LT o el UT se mantiene en el primer nivel de sangrado hasta que concuerda un evento. Si debe comenzar un evento, lo inicia el LT o el UT. Si debe recibirse un evento, se dice que éste concuerda sólo cuando acaece un evento real recibido que concuerda con la línea de eventos;

b) una vez que un evento ha concordado, el LT o el UT pasa al siguiente nivel de sangrado. No puede efectuarse un retorno al nivel previo de sangrado, salvo si se utiliza el constructivo GOTO;

c) las líneas de evento al mismo nivel de sangrado y que siguen a la misma línea de evento predecesora representan las posibles alternativas que pueden concordar en ese momento. Las alternativas se darán en el orden en el que el especificador de la serie de pruebas exija que el LT o el UT intente iniciarlas o recibirlas, de ser necesario, repetidamente, hasta que una de ellas concuerde.

EJEMPLO 68 – Ilustración de un árbol de comportamiento TTCN:

Supóngase la posible producción de la siguiente secuencia de eventos durante una prueba, cuya finalidad es el establecimiento de una conexión, el intercambio de algunos datos y la liberación de la conexión. Los eventos se producen en el PCO L del probador inferior:

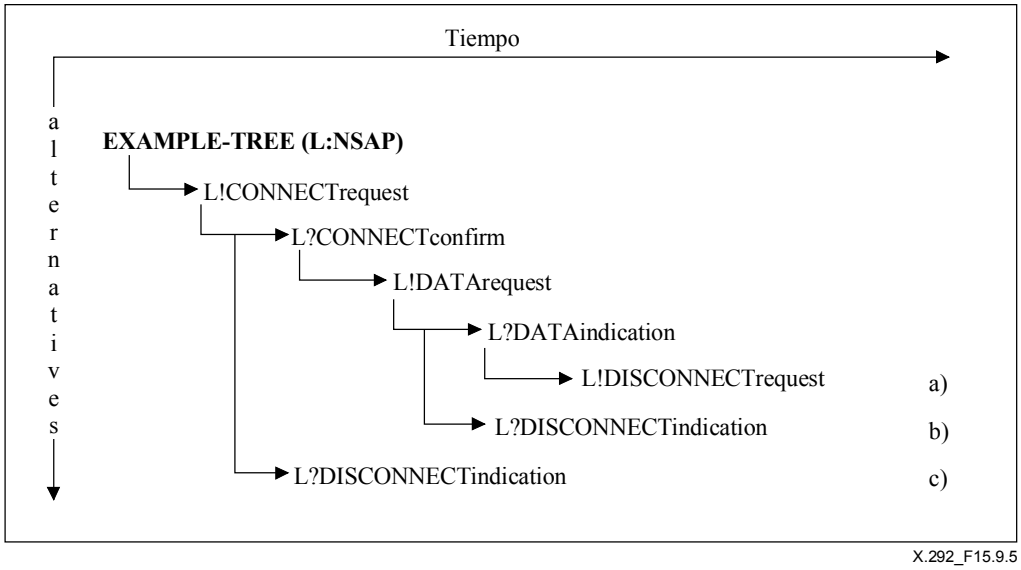
a) Petición CONEXIÓN, confirmación CONEXIÓN, petición DATOS, indicación DATOS, petición DESCONEXIÓN.

   La progresión puede ser alterada en cualquier momento por la IUT o el proveedor del servicio. Esto genera dos o más secuencias:

b) Petición CONEXIÓN, confirmación CONEXIÓN, petición DATOS, indicación DESCONEXIÓN.

c) Petición CONEXIÓN, indicación DESCONEXIÓN.

   Las tres secuencias de eventos pueden expresarse mediante un árbol de comportamiento de TTCN. Hay cinco niveles de alternativas y sólo tres hojas [a) a c)] porque los eventos SEND L! son siempre exitosos. La ejecución progresará de izquierda a derecha (secuencia) y de arriba abajo (alternativas). En la figura que sigue se ilustra esta progresión así como el principio del árbol de comportamiento de TTCN:



X.292_F15.9.5

En la TTCN no hay líneas, flechas ni nombres de hoja. El árbol de comportamiento del ejemplo anterior se puede representar así:

EJEMPLO 69 – Árbol de comportamiento de TTCN:

Comportamiento dinámico de paso de prueba					
Nombre del paso de prueba : TREE_EX_1 (L:NSAP)					
Grupo : TTCN_EXAMPLES/TREE_EXAMPLE_1/					
Objetivo : Ilustración del uso de los árboles.					
Valor por defecto :					
Comentario : NOTA – Puede simplificarse este ejemplo con valores por defecto.					
N.º	**Etiqueta**	**Descripción de comportamiento**	**Ref. de restricciones**	**Veredicto**	**Comentarios**
1		L ! CONNECTrequest	CR1		Petición …
2		L ? CONNECTconfirm	CC1		…Confirmación
3		L ! DATArequest	DTR1		Enviar datos
4		L ? DATAindication	DTI1		Recibir datos
5		L ! DISCONNECTrequest	DSCR1	PASS	Aceptar
6		L ? DISCONNECTindication	DSCI1	INCONC	Prematuro
7		L ? DISCONNECTindication	DSCR1	INCONC	Prematuro
Comentarios detallados:					

15.9.5.2 El concepto de semántica instantánea

Los enunciados alternativos en el nivel actual de sangrado se procesan según su orden de aparición. La semántica operacional de la TTCN (véase el anexo B) presupone que la situación de cualquiera de los eventos permanece invariable en el proceso de tentativa de concordancia con una de las alternativas de un conjunto. Esto implica que se utilice una semántica instantánea para los eventos recibidos y temporizaciones, es decir, en cada intervalo de tiempo en torno a un conjunto de alternativas, se toma una instantánea de los eventos recibidos y de las temporizaciones iniciadas. Solamente los eventos y las temporizaciones de la instantánea pueden concordar en el ciclo siguiente a través de las alternativas.

15.9.5.3 Restricciones a la utilización de eventos

Para evitar errores en casos de prueba se aplican las siguientes restricciones:

a) Un caso de prueba o paso de prueba no debe contener un comportamiento en el cual la velocidad de procesamiento relativa del MOT pudiera influir en los resultados. En evitación de tales problemas una

línea de eventos RECEIVE, OTHERWISE o TIMEOUT sólo deberá ir seguida por otras líneas de eventos RECEIVE, OTHERWISE y TIMEOUT en un conjunto de alternativas. En consecuencia, los árboles por defecto contendrán solamente líneas de eventos RECEIVE, OTHERWISE y TIMEOUT sobre el primer conjunto de alternativas.

b) Una vez que existe un evento en una cola de PCO o una temporización en la lista de temporizaciones, dicho evento sólo podrá ser sacado de la cola o lista por una concordancia correcta del enunciado en TTCN correspondiente. En el caso de un conjunto de alternativas que incluya enunciados RECEIVE, el conjunto de eventos entrantes esperados deberá estar totalmente especificado. Esto significa que se producirá un error de caso de prueba si, durante la ejecución, no se produce la concordancia de ninguno de los enunciados RECEIVE y, a pesar de ello, la ejecución avanza al nivel siguiente de alternativas porque una temporización (TIMEOUT) que ocurrió después de una ASP o PDU, y que no estaba especificada en el conjunto de enunciados RECEIVE, fue recibida en cualquiera de las colas de PCO o CP pertinentes. No se utilizará IMPLICIT SEND con CM.

c) Al utilizar la TTCN concurrente hay que adoptar precauciones a fin de evitar que se produzcan resultados no fiables causados por situaciones en las cuales el orden de recepción de los eventos en puntos PCO diferentes o en CP diferentes se emplea para determinar la asignación de veredicto. El instante real en el que se recibe una PDU o un CM, con respecto al de la recepción de otra PDU o CM, no se puede reflejar exactamente cuando se ejecutan componentes de prueba paralelos.

EJEMPLO 70 – Conjunto incompleto de eventos RECEIVE:

```
PARTIAL_TREE
! A START T
        ? B
        ? TIMEOUT T
            !C
                ?D
```

a)

```
PARTIAL_TREE
! A START T
        ? B
        ? OTHERWISE
            FAIL
        ? TIMEOUT T
            !C
                ?D
```

b)

Si en a), se recibe D en respuesta a !A el caso de prueba asignará un veredicto PASS erróneo en virtud de la TIMEOUT. Esto puede evitarse con el enunciado OTHERWISE.

d) En la TTCN concurrente, el orden relativo de los eventos en PCO diferentes o CP diferentes no debe afectar al veredicto asignado, ya que ello conduciría a la irrepetibilidad de los resultados causada por las diferencias en las velocidades de procesamiento y de transmisión.

15.9.5.4 Precauciones que han de adoptarse cuando se utiliza la TTCN concurrente

Al utilizar la TTCN concurrente se deben adoptar precauciones para evitar que se produzcan resultados no repetibles causados por situaciones en las cuales el orden de recepción de los eventos en PCO diferentes o en CP diferentes se emplea para determinar la asignación de veredicto. El instante real en el que se recibe una PDU o un CM, con respecto al instante de la recepción de otra PDU u otro CM, no se puede reflejar exactamente cuando se ejecutan componentes de prueba paralelos.

15.9.6 El evento IMPLICIT SEND

En los métodos de prueba a distancia, aunque no haya un PCO explícito por encima de la IUT, es necesario disponer de alguna forma de especificar, en un punto dado de la descripción del comportamiento del LT, que la IUT debe iniciar una PDU o una ASP determinada (pero no un CM). Para ello, se define el evento enviar implícito.

No se especifica lo que se le hace a la IUT para desencadenar esta reacción, sino sólo la reacción que se necesita; la IUT enviará ASP o PDU en el PCO indicado. La **IUT** sustituye al identificador de PCO cuando no hay ambigüedad (por ejemplo, cuando hay un solo PCO).

Se considera que un evento IMPLICIT SEND siempre es exitoso, en el sentido de que todas las alternativas codificadas después y al mismo nivel de sangrado que IMPLICIT SEND, son inalcanzables.

Solamente se utilizará IMPLICIT SEND cuando la norma o normas OSI pertinentes permitan a la IUT enviar las ASP o las PDU especificadas en ese punto en su proceso de comunicación con el LT.

Para cada IMPLICIT SEND de una serie de pruebas, el especificador de la serie de pruebas deberá crear y referenciar una cuestión en el formulario de PIXIT parcial, que permita la indicación de si puede invocarse IMPLICIT SEND por demanda.

No se utilizará un evento IMPLICIT SEND a menos que el método de prueba utilizado sea uno de los métodos de prueba a distancia. No se utilizará un evento IMPLICIT SEND a menos que pudiera haberse conseguido el mismo efecto con el método de prueba DS.

NOTA – Por ejemplo, cuando se pruebe una implementación de protocolo de transporte con conexión, si no existiera esta restricción, sería admisible utilizar IMPLICIT SEND para hacer que la IUT iniciara una CR TPDU, ya que en el método de prueba DS podría conseguirse dicho efecto haciendo que el UT enviara una ASP petCON-T. En cambio, no sería admisible utilizar IMPLICIT SEND para hacer que la IUT iniciara una ASP PetRst-N, ya que tal efecto no podría ser controlado a través de la frontera del servicio de transporte. El motivo de esta restricción es evitar que los casos de prueba requieran un mayor control exterior sobre una IUT que el proporcionado por la norma del protocolo pertinente.

Cuando se especifique un evento IMPLICIT SEND, también se efectuarán los eventos internos asociados dentro de la IUT, necesarios para el cumplimiento de los requisitos de la norma del protocolo sometido a prueba es decir, la fijación de temporizador y las variables de inicialización de estado.

La semántica de IMPLICIT SEND es tal que, si fuese necesario, se puede controlar el SUT para provocar la iniciación de la ASP o PDU especificada. En la PIXIT (o en la documentación referenciada por la PIXIT), se especificará la manera de controlar el SUT.

En el evento IMPLICIT SEND no podrán codificarse ni un veredicto final ni un resultado preliminar.

En un punto apropiado que siga a IMPLICIT SEND habrá un evento RECEIVE que concuerde con la ASP o la PDU, el cual debería haber sido enviado como resultado por la IUT.

EJEMPLO 71 – Ejemplo de utilización de IMPLICIT SEND:

Comportamiento dinámico de caso de prueba					
Nombre del caso de prueba : IMPI					
Grupo : TTCN_EXAMPLES/IMPLICIT_SEND/					
Finalidad : Árbol parcial para ilustrar la utilización de IMPLICIT SEND					
Valor por defecto :					
Comentarios :					
N.º	**Etiqueta**	**Descripción de comportamiento**	**Ref. de constricciones**	**Veredicto**	**Comentarios**
:		:			
5		<IUT ! CR>	CR1		
6		L ? CR	CR1		
7		L ! CC	CC1		
:		:			
12		L ? OTHERWISE			
:		:			
Comentarios detallados:					

15.9.7 El evento OTHERWISE

El evento predefinido OTHERWISE es el mecanismo de la TTCN utilizado para tratar eventos de prueba imprevistos de una forma controlada.

Se utiliza OTHERWISE para indicar que el LT o el UT, aceptará *cualquier* evento entrante que no haya concordado anteriormente con una de las alternativas OTHERWISE. El probador aceptará cualquier información entrante que no se haya podido decodificar o que no haya concordado anteriormente con una alternativa anterior a este evento OTHERWISE.

En la TTCN no concurrente, si en una serie de pruebas hay más de un PCO, el prefijo de OTHERWISE será un nombre del PCO que figure en la parte declaraciones, o en un parámetro formal de la lista de parámetros formales del árbol

cuando se utiliza al parámetro formal para cursar un nombre de PCO. El nombre del PCO se utiliza para indicar el PCO en el que puede producirse el evento de pruebas. Los eventos entrantes, incluido OTHERWISE, se consideran solamente en términos del PCO dado.

EJEMPLO 72 – Utilización de OTHERWISE con identificadores de PCO:

PARTIAL_TREE	
PCO1 ? A	
PCO2 ? B	PASS
PCO1 ? C	INCONC
PCO2 ? OTHERWISE	FAIL

Suponiendo que no se haya recibido ningún evento en PCO1, la recepción del evento B en PCO2 producirá un veredicto de éxito (PASS). La recepción de cualquier otro evento en PCO2 producirá un veredicto de fracaso (FAIL).

Debido a la significación de la ordenación de alternativas, los eventos entrantes que sean alternativas que siguen a un OTHERWISE incondicional en el mismo PCO nunca concordarán entre sí.

EJEMPLO 73 – Eventos entrantes que siguen a un OTHERWISE:

PARTIAL_TREE	
PCO1 ? A	PASS
PCO1 ? OTHERWISE	FAIL
PCO1 ? C	INCONC

El OTHERWISE concordará con cualquier evento entrante diferente de A. La última alternativa, ?C, no puede concordar nunca.

15.9.8 El evento OTHERWISE y la TTCN concurrente

En la TTCN concurrente, el evento OTHERWISE se puede utilizar con puntos CP al igual que con puntos PCO. Se admite OTHERWISE en CP para proporcionar un modo eficiente de manejar "los demás CM en este CP".

15.9.9 El evento TIMEOUT

El evento TIMEOUT permite verificar, en un caso de prueba, la expiración de un temporizador o de todos los temporizadores. Cuando expira un temporizador (conceptualmente inmediatamente antes del procesamiento instantáneo de un conjunto de eventos alternativos), se sitúa un evento TIMEOUT en una lista de temporizaciones. Inmediatamente después el temporizador queda inactivo. En un momento, cualquiera sólo puede aparecer en la lista una única entrada, para cualquier temporizador concreto. Como TIMEOUT no está asociado con un PCO, se utiliza una sola lista de temporizaciones.

Cuando se procesa un evento TIMEOUT, si se indica un nombre de temporizador, se efectúa una búsqueda en la lista de temporizaciones y, en el caso en que exista un evento de temporización que concuerde con el nombre del temporizador, se retira dicho evento de la lista, y el evento TIMEOUT es exitoso.

Si no se indica ningún nombre de temporizador, cualquier evento TIMEOUT de la lista de temporizaciones exitoso. El evento TIMEOUT es exitoso cuando la lista no está vacía. Cuando así ocurre, se vacía la lista completa de temporizaciones.

EJEMPLO 74 – Utilización de TIMEOUT:

	? TIMEOUT T			

Puesto que los eventos TIMEOUT no son eventos RECEIVE, las alternativas OTHERWISE antes indicadas no las hacen inalcanzables.

15.9.10 Eventos y constructivos en la TTCN concurrente

En la TTCN se utilizan el constructivo CREATE y el evento DONE.

15.9.10.1 El constructivo CREATE

El componente de prueba principal arranca al principio de la ejecución del caso de prueba. El componente de prueba principal arranca componentes de prueba paralelos, en caso necesario, mediante el constructivo CREATE.

Este constructivo invoca un conjunto de componentes de prueba paralelos (PTC). Hay dos argumentos para cada PTC. El primero es el identificador del PTC que se crea, el cual concordará con el identificador de un PTC de la configuración de componente de prueba referenciada en el encabezamiento del caso de prueba. El segundo es una referencia a un árbol de comportamiento (es decir, paso de prueba o árbol local), posiblemente con una lista de parámetros que contiene valores reales (por ejemplo, PCO y CP). El efecto del constructivo CREATE es que cada PTC listado arranque la ejecución de su descripción de comportamiento en paralelo con la ejecución del componente de prueba principal.

NOTA – La transferencia de identificadores de PCO y CP a un árbol de comportamiento como parámetros reales que se pueda utilizar el mismo árbol de comportamiento en más de un componente de prueba.

Los PCO y CP que se utilizan en la ejecución de la descripción de comportamiento asociada con un PTC mediante el constructivo CREATE deberán ser solamente los determinados por la configuración de componente de prueba para ese caso de prueba.

La ejecución de un constructivo CREATE en un PTC que ya ha sido creado deberá dar como resultado un error de caso de prueba. La ejecución de un CREATE por cualquier otro componente de prueba distinto del MTC deberá dar como resultado un error de caso de prueba.

En el constructivo CREATE, los identificadores de PCO y los identificadores de CP son transferidos a un PTC por sustitución textual, como es usual en la ADJUNción (ATTACHment) de pasos de prueba. Los parámetros restantes son puestos en valor. Esto se realiza para evitar efectos secundarios en las variables que pudieran afectar al procesamiento de otros PTC, produciendo resultados no repetibles.

15.9.10.2 El evento DONE

Cuando el MTC termina, asigna el veredicto final calculado hasta ese momento (15.17.5). El evento DONE se puede utilizar en el MTC y en los PTC para averiguar si los PTC ya han terminado. Los componentes de prueba pueden emplear esta información para determinar sus propios resultados preliminares y para otras acciones; en especial, el MTC puede eludir la terminación antes de que todos los PTC hayan terminado (15.17.5).

La falta de una lista de argumentos se interpreta es un lista de todos los PTC indicados en un constructivo CREATE ejecutado antes de la ejecución del evento DONE. Un evento DONE sin una lista de argumentos sólo será utilizado por el MTC.

EJEMPLO 75 – Utilización del evento DONE:

```
PARTIAL_MTC_TREE
CREATE (PTC1 : TREEA , PTC2 : TREEB)
        ? DONE (PTC1, PTC2)
```

NOTA 1 – Si DONE es la única alternativa, ello equivale a una petición de espera para la terminación del PTC especificado.

NOTA 2 – DONE no significa para el MTC la terminación de coordinación del PTC.

15.10 Expresiones en TTCN

15.10.1 Introducción

Hay dos clases de expresiones en TTCN: asignaciones y expresiones booleanas. Tanto las asignaciones como las expresiones booleanas pueden contener valores explícitos y las siguientes formas de referencia a objetos de datos:

a) parámetros de series de pruebas;

b) constantes de series de pruebas;

c) variables de caso de prueba y de series de pruebas;

d) parámetros formales de un paso de prueba, árbol por defecto o árbol local;

e) ASP y PDU (en líneas de evento).

Todas las variables que aparezcan en expresiones booleanas y/o al lado derecho de una asignación, deberán estar acotadas. Si se utiliza una variable no acotada, se tratará de un error de caso de prueba.

15.10.2 Referencias a objetos de datos definidos en ASN.1

15.10.2.1 Introducción

Para permitir hacer referencias a componentes de objetos de datos definidos mediante ASN.1, la TTCN proporciona tres mecanismos de acceso: referencias de registro, referencias de matriz y referencias de bit.

15.10.2.2 Referencias de registro

Para hacer referencia a un componente de objeto de datos del tipo SEQUENCE, SET o CHOICE se puede utilizar una referencia a registro. Una referencia a registro se construye con una notación de punto, añadiendo un punto y el nombre (identificador del componente) o número (posición del componente) del componente deseado al identificador del objeto de datos. Si se ha identificado, se utilizará el identificador del componente con preferencia a la posición del componente. Las referencias a componentes no designados, se construyen indicando entre paréntesis el número de la posición del componente dentro de la definición del tipo. Por definición, la numeración implícita de componentes empieza con cero; en consecuencia, el tercer componente tiene la posición número 2.

En la Rec. UIT-T X.680, se definen tipos SET con componentes no ordenados. Esto solamente tiene importancia si se codifican valores de este tipo y se envían a través del proveedor de servicio subyacente. En consecuencia, la TTCN trata los objetos de datos del tipo SET de la misma forma que los objetos del tipo SEQUENCE, es decir, que la referencia a componentes con número *i* significa siempre una referencia al campo *i*-ésimo declarado en el tipo.

Una vez que se hayan recibido una ASP o una PDU o un CM, la referencia al componente con el índice *i* devolverá siempre el mismo valor. El orden de los elementos en un SET no se alteraría en cualquier operación realizada en la TTCN.

EJEMPLO 76 – Referencias de registro de componente:

```
Example_type ::= SEQUENCE {
                        field_1   INTEGER
                        field_2   BOOLEAN,
                                  OCTET STRING }
```

Si var1 es de tipo ASN.1 Example_type, se podría escribir:

var1.field_1	que se refiere al primer campo (INTEGER)
var1.(3)	que se refiere al tercer campo (no designado)

EJEMPLO 77 – Referencias de campo de PDU:

```
XY_PDUtype ::= SEQUENCE {
                        ... ,
                        user-data   OCTET STRING,
                        ...  }
```

En una línea de enunciado que contuviera XY_PDUtype, se podría escribir lo siguiente:

L? XY_PDU (buffer := XY_PDUtype.user_data)

para cargar la memoria intermedia variable con el contenido del campo user_data de la PDU entrante.

Cuando una PDU o un parámetro, campo o elemento de tipo en ASN.1 se encadena a una ASP, a otra PDU o a un CM, se puede utilizar una referencia a registro para identificar un componente de esa PDU o tipo en ASN.1. La referencia a

registro identificará la secuencia completa pertinente de nombres de parámetro, campo o elemento separados por puntos, y arrancando con un identificador de objeto de datos que resuelve al identificador de ASP, al identificador de CM o (en el caso en que no se utilicen ASP en la serie de pruebas) al identificador de PDU pertinente. Más allá de este identificador de objeto de datos inicial, la secuencia no deberá contener ningún identificador de PDU o identificador de tipo en ASN.1, sino, en su lugar, sólo los identificadores de los parámetros, campos y elementos pertinentes. Este mecanismo no se utilizará si existe alguna ambigüedad acerca de la identidad de una constricción de PDU o constricción de tipo en ASN.1 en la secuencia. En el siguiente ejemplo se ilustra el uso de referencias de registro cuando se emplea el encadenamiento de constricciones (véase 12.4).

EJEMPLO 78 – Referencias de registro con encadenamiento:

Definición de ASP en ASN.1

```
ASP1_type ::= SEQUENCE  {
                        par_1   OCTET STRING,
                        par_2   OCTET STRING,
                        pdu1    PDU1_type
                            }
```

Definición de tipo PDU en ASN.1

```
PDU1_type ::= SEQUENCE  {
                        field1  OCTET STRING,
                        field2  OCTET STRING,
                        f       F_type
                        }
```

Definición de tipo structure en ASN.1

```
F_type ::= SEQUENCE     {
                        data1   IA5STRING,
                        data2   IA5STRING
                        }
```

Cuando se utilizan constricciones de tipo ASP1_type, PDU1_type y F_type, los valores de data1 y data2 pueden referenciarse como sigue:

ASP1_type.pdu1.f.data1

ASP1_type.pdu1.f.data2

De manera análoga, el campo F de la PDU completa se puede referenciar como:

ASP1_type.pdu1.f

o la PDU completa puede referenciarse como:

ASP1_type.pdu1

Hay que señalar que las declaraciones empleadas en este ejemplo se pueden aplicar tanto al encadenamiento estático como al encadenamiento dinámico, ya que las diferencias entre los dos tipos encadenamiento sólo son visibles en las constricciones. Por tanto, la referencia a registro es independiente de la variedad de encadenamientos utilizados.

15.10.2.3 Referencias de matriz

Se puede emplear una referencia a matriz para hacer referencia a un componente de objeto de datos del tipo SEQUENCE OF o SET OF. Una referencia a matriz deberá construirse con una notación de punto, añadiendo un punto y el índice del componente deseado al identificador de objeto de datos. El índice, indicando la posición del componente dentro del objeto de datos (cuando el objeto es examinado como una matriz lineal), está encerrado entre corchetes. Por definición, dentro de ASN.1 la indexación de componentes se inicia con cero. El índice puede ser una expresión, en cuyo caso deberá tomar un valor INTEGER no negativo.

En la Rec. UIT-T X.680, se definen tipos SET OF con componentes no ordenados. Esto solamente tiene importancia si se codifican valores de ese tipo y se envían a través del proveedor de servicio subyacente. En consecuencia, la TTCN trata los objetos de datos del tipo SET OF de la misma forma que objetos de tipo SEQUENCE OF; es decir, que la referencia a componentes con número *i* significa siempre una referencia al campo *i*-ésimo declarado en el tipo.

Una vez recibida una ASP o una PDU, la referencia al componente con el índice *i* devolverá siempre el mismo valor. El orden de los elementos en un SET OF, no se alteraría en cualquier operación realizada en la TTCN.

EJEMPLO 79 – Referencias de matriz de componente:

```
Array_type ::= SEQUENCE OF {BOOLEAN}
```

Si var2 es del tipo ASN.1 Array_type, se podría escribir lo siguiente para referirse al primer BOOLEAN en la secuencia:

var2.[0]

var1.[1-1]

15.10.2.4 Referencias de bit

Una referencia a bit se puede utilizar para hacer referencia a bits particulares dentro de un tipo BITSTRING. Con esta finalidad, se supone que los objetos de datos del tipo BITSTRING se definen como SEQUENCE OF {BOOLEAN}. Así, una referencia a bit se puede construir con la notación de índice al igual que las referencias de matriz. El bit situado más a la izquierda tiene el índice cero. Una expresión utilizada como un índice en una referencia a bit deberá tomar un valor INTEGER no negativo. Alternativamente, si determinados bits de una BITSTRING están asociados con un identificador (bits denominados), el identificador se puede utilizar para hacer referencia al bit.

EJEMPLO 80 – Referencias de bit:

```
B_type ::= BIT STRING { ack(0), poll(3) }
```

Define un B_type de tipo BITSTRING, donde el bit cero se denomina "ack" y el bit tres se denomina "poll".

Si b_str es de tipo B_type en ASN.1, se puede escribir;

b_str.ack := TRUE

b_str.[2] := FALSE

Obsérvese que b_str.poll := TRUE y b_str [3] := TRUE asignan ambos el valor TRUE al bit "poll".

15.10.3 Referencias para objetos de datos definidos con cuadros

Para construir referencias de registro a componentes de ASP, PDU, CM y tipos estructurados definidos en forma tabular, se utilizará la sintaxis definida en 15.10.2.2. El encadenamiento de tipos de ASP, PDU, CM y estructurados en forma tabular repercute en las referencias de registro exactamente del mismo modo que en los definidos en ASN.1.

Cuando se defina un parámetro, campo o elemento para incluir un ítem que es una subestructura verdadera de un tipo definido, en un cuadro de tipos estructurados, la referencia al ítem en la subestructura consistirá en la referencia a registro al parámetro, campo o elemento seguida de un punto, con el identificador del ítem dentro de esa estructura.

Cuando se utilice una estructura como expansión de macro, se hará referencia a los elementos de la estructura como si esa estructura hubiera sido expandida a la estructura que se refiere a ella.

Si un parámetro, campo o elemento se define de forma que sea de metatipo **PDU**, no se harán referencias a campos de esa subestructura.

15.10.4 Asignaciones

15.10.4.1 Introducción

Pueden asociarse eventos de prueba con una lista de asignaciones y/o un calificador. Las asignaciones se separan entre sí mediante coma y la lista se encierra entre paréntesis.

Durante la ejecución de una asignación, el segundo miembro tomará el valor de un elemento del tipo del primer miembro.

El efecto de una asignación es el de vincular la variable de la serie de pruebas o del caso de prueba (o el parámetro la ASP o el campo de la PDU) al valor de la expresión. La expresión no podrá contener variables no acotadas.

Todas las asignaciones se producen según el orden en que aparecen, es decir, el procesamiento se efectúa de izquierda a derecha.

EJEMPLO 81 – Utilización de asignaciones con líneas de evento:

```
(X:=1)
  (Y:=2)
    L!A (Y:=0, X:=Y, A.field1:=y)
      L?B (Y:=B.field2, X:=X+1)
```

Cuando se haya transmitido la PDU A exitosamente, el contenido de las variables de caso de prueba X e Y será cero y el campo 1 de la PDU A contendrá, asimismo, el valor cero. Tras recibir la PDU B, se asignará a la variable Y de caso de prueba el contenido del campo 2 de la PDU B y se incrementará la variable X del caso de prueba.

15.10.4.2 Normas de asignación para tipos cadena

Si, dentro de una asignación, se utilizan tipos cadena restringidos en longitud se aplicarán las siguientes reglas:

a) Si se define que el tipo cadena destino es más corto que el tipo cadena fuente, el tipo cadena fuente se trunca, por la derecha, hasta un valor igual a la longitud máxima del tipo cadena destino.

b) Si la cadena fuente es más corta que el valor admitido por el tipo cadena destino, la cadena fuente se alinea a la izquierda y se completa con caracteres de relleno hasta un valor igual al máximo del tipo cadena destino.

Como caracteres de relleno se utilizan:

" " (blanco) para todas las CharacterString;

"0" (cero) para BITSTRING, HEXSTRING y OCTETSTRING.

Cuando en el lado izquierdo de una asignación se utilice una variable de tipo cadena no acotada (es decir, de una longitud arbitraria), deberá quedar acotada al valor del lado derecho sin relleno. El relleno sólo será necesario cuando la variable sea de tipo cadena de longitud fija.

15.10.5 Calificadores

Un evento puede calificarse disponiendo una expresión booleana encerrada entre corchetes tras el evento. Esta calificación se interpreta en el sentido de que el enunciado se ejecutará solamente en el caso en que concuerde el evento y el calificador tome el valor TRUE.

Si a un mismo evento se le asocian un calificador y una asignación, aparecerá primero el calificador, tomando cualquier término del mismo los valores existentes con anterioridad a la ejecución de la asignación.

15.10.6 Líneas de evento con asignaciones y calificadores

A un evento se le puede asociar una asignación, un calificador o ambos. Si a un evento se le asocia una asignación, esa asignación solamente se ejecuta si el evento concuerda. Si a un evento se le asocia un calificador, aquél solamente podrá concordar si el calificador toma el valor TRUE. Si a un evento se le asocian una asignación y un calificador, el evento sólo concordará si el calificador toma el valor TRUE y la asignación sólo se ejecutará si el evento concuerda.

Si se califica un evento RECEIVE y el evento que ha ocurrido concuerda potencialmente con el evento especificado, el calificador se evaluará en el contexto del evento acaecido. Si el calificador contiene una referencia a parámetros de ASP y/o campos de PDU, se tomarán los valores de esos parámetros y/o campos del evento que ha acaecido.

Las reglas para la utilización de asignaciones en los eventos son las siguientes:

a) En el caso de un evento SEND, todas las asignaciones se efectúan *después* de evaluar el calificador y *antes* de transmitir ASP o PDU.

b) En el caso de eventos SEND, se permiten asignaciones para los campos de la ASP o PDU que se está transmitiendo.

c) En el caso de un evento RECEIVE, las asignaciones se efectúan *después* de ocurrido el evento y no pueden hacerse para campos de la ASP o PDU que acaba de recibirse.

Una asignación a un parámetro ASP, campo de PDU o elemento de estructura de una constricción en la parte comportamiento sobreescribirá los valores de constricción en una línea de evento SEND.

EJEMPLO 82 – Utilización de un evento SEND calificado:

```
PARTIAL_TREE
!A[X:=3]
!B
```

En el procesamiento de esos eventos SEND alternativos, el probador enviará A solamente en el caso en que la variable X tome el valor 3. En cualquier otro caso, enviará B.

El evento OTHERWISE se puede utilizar junto con calificadores y/o asignaciones. Si se utiliza un calificador, esta variable booleana se convierte en una condición adicional para la aceptación de cualquier evento entrante. Si se utiliza un enunciado asignación, esa asignación solamente tendrá lugar si se satisfacen todas las condiciones para que concuerde el OTHERWISE.

EJEMPLO 83 – Utilización de calificadores y asignaciones OTHERWISE:

```
PARTIAL_TREE (PCO1:XSAP; PCO2:YSAP)
PCO1 ? A                                                    PASS
PCO2 ? B  [X=2]                                             INCONC
PCO1 ? C                                                    PASS
PCO2 ? OTHERWISE [X<>2] (Reason:="X not equal 2")           FAIL
PCO2 ? OTHERWISE (Reason:="X equals 2 but event not B")     FAIL
```

Supóngase que en el PCO1 no se ha recibido ningún evento. La recepción del evento B en el PCO2, cuando X=2 produce un veredicto no concluyente. La recepción de cualquier otro evento en el PCO2 cuando X<>2 produce un veredicto de FAIL y asigna un valor de "X distinto de 2" a la variable motivo CharacterString. Si en PCO2 se recibe un evento que no satisface ninguno de estos escenarios, el OTHERWISE final concordará.

Los eventos que incluyen CM que ocurren en CP pueden asociarse también con una asignación, un calificador, o ambos, del mismo modo que para las PDU, como se ha descrito anteriormente.

EJEMPLO 84 – CM asociados con un calificador:

```
A_CP!A_CM                   [X=2]
```

15.11 Seudoeventos

Se permite la utilización de asignaciones, calificadores y operaciones de temporización por sí mismos en una línea de enunciado un árbol de comportamiento, sin ningún evento asociado. Estas expresiones autónomas se denominan seudoeventos.

El significado de tales seudoeventos es el siguiente:

a) Si solamente se especifica un calificador: éste se evalúa y continúa la ejecución con el comportamiento subsiguiente, si la evaluación del calificador es TRUE; si la evaluación es FALSE, se intenta la siguiente alternativa. Si no hay alternativa, se trata de un error de caso de prueba.

b) Si solamente se especifican asignaciones y/u operaciones de temporizador: se ejecutarán de izquierda a derecha las asignaciones y/o se ejecutarán también de izquierda a derecha las operaciones de temporizador.

c) Si se especifican asignaciones y/u operaciones de temporizador precedidas de un calificador: éste se evaluará en primer lugar, procediéndose después a la evaluación de las asignaciones y/u operaciones de temporizador solamente en el caso en que el valor del calificador sea TRUE.

15.12 Gestión de temporizador

15.12.1 Introducción

Para modelar la gestión de temporizador se utiliza un conjunto de operaciones. Tales operaciones pueden aparecer combinadas con eventos o en forma de seudoeventos autónomos.

Se puede aplicar operaciones de temporizador a:

- un temporizador individual, especificado mediante la operación del temporizador seguida del nombre del temporizador;
- la totalidad de los temporizadores, lo que se especifica omitiendo el nombre del temporizador.

Se supone que los temporizadores utilizados en una serie de pruebas están en marcha o inactivos. Todos los temporizadores en marcha se cancelan automáticamente al final de cada caso de prueba. Hay tres operaciones de temporizador predefinidas, a saber: START, CANCEL y READTIMER. En una línea de evento puede especificarse, si es necesario, más de una operación de temporizador. Esto se indica separando las operaciones mediante comas.

Cuando en una misma línea de enunciado aparezca una operación de temporizador como un evento y/o un calificador se ejecutará la operación de temporizador si, y sólo si, el evento concuerda y/o el calificador toma el valor TRUE.

15.12.2 La operación START

La operación START se utiliza para indicar que un temporizador tiene que arrancar.

Se utilizará el parámetro valor de temporizador optativo cuando no se proporcione una duración por defecto o cuando se desee asignar un tiempo de expiración (es decir, de duración) a un temporizador, que sustituya al valor por defecto especificado en las declaraciones del temporizador.

Los valores del temporizador serán del tipo INTEGER. El escritor del caso de prueba deberá asegurarse de que el parámetro valor de temporizador optativo tenga como evaluación, un valor INTEGER positivo no nulo. Si el temporizador se arranca con un valor negativo o nulo, se producirá un error de caso de prueba.

Todas las variables que aparezcan en la expresión que especifica el valor del temporizador optativo deberán estar acotadas. Si se utiliza una variable no acotada, se producirá un error de caso de prueba.

Cuando se reemplaza una duración de temporizador, el nuevo valor se aplica solamente a la instancia vigente del temporizador. Es decir, cualesquiera operaciones START posteriores para ese temporizador que no especifiquen una duración utilizarán la duración indicada en la parte declaraciones de temporizador.

EJEMPLO 85 – Utilizaciones del temporizador START:

donde Ti son identificadores de temporizador y Vi son valores de temporizador:

```
START T0
START T0 (V0)
START T1, START  T2 (V2)
```

La operación START puede aplicarse a un temporizador en marcha, en cuyo caso se cancela, repone y rearranca el temporizador. Cualquier entrada en la lista de duraciones de temporización de este temporizador se eliminará de dicha lista.

15.12.3 La operación CANCEL

La operación CANCEL se utiliza para detener un temporizador en marcha.

Un temporizador cancelado queda inactivo. Si en la lista de duraciones de temporización aparece un evento TIMEOUT para ese temporizador, se elimina dicho evento de la citada lista. Si se omite el nombre del temporizador en la operación CANCEL, quedan inactivos todos los temporizadores en marcha y se vacía la lista de temporizaciones.

La cancelación de un temporizador inactivo es una operación válida, aunque no produce ningún efecto.

EJEMPLO 86 – Algunos usos del temporizador CANCEL:

donde Ti son identificadores de temporizador:

```
CANCEL
CANCEL T0
CANCEL T1, CANCEL T2
CANCEL T1, START T3
```

15.12.4 La operación READTIMER

La operación READTIMER se utiliza para recuperar el tiempo transcurrido desde que se arrancó el temporizador especificado y para almacenar dicho tiempo en la variable de caso de prueba o de serie de pruebas especificada. Esta variable será de tipo INTEGER. Se considera que el valor temporal asignado a la variable tiene la misma unidad de tiempo que la especificada en la declaración del temporizador. Por convención, la aplicación de la operación READTIMER a un temporizador inactivo devolverá el valor cero.

EJEMPLO 87 – Utilización de READTIMER:

```
:
START TimerName (TimerVal)
      ?EVENT_A
            +Tree_A
      ?EVENT_B
            +Tree_B
      ?EVENT_C
            READTIMER TimerNAME(CurrTime)
            +Tree_C
      ?TIMEOUT TimerName
:
```

Si antes de la expiración del temporizador designado por TimerName se recibe EVENT_C, el intervalo de tiempo transcurrido desde el arranque del temporizador se almacenará en la variable CurrTime de la serie de pruebas, o caso de prueba. El comportamiento contenido en Tree_C puede utilizar el valor de esta variable de caso de prueba o de serie de pruebas.

EJEMPLO 88 – READTIMER utilizado en combinación con otras operaciones de temporizador:

READTIMER T1 (PASSED_TIME), CANCEL T1

READTIMER T1 (V1), START NEW_TIMER (V1)

15.13 El constructivo ATTACH

15.13.1 Introducción

Mediante el constructivo ATTACH se puede adjuntar unos árboles a otros.

Las variables de caso de prueba o serie de pruebas son globales tanto para el árbol que efectúa la adjunción (árbol principal) como para el árbol adjuntado, es decir, todos los cambios que se efectúen en variables de un árbol adjuntado, se aplican también al árbol principal. Los constructivos de adjunción de árbol deberán aparecer en una línea de enunciado por sí mismos.

15.13.2 Ámbito de la adjunción de árbol

Las descripciones de comportamiento pueden contener más de un árbol. Sin embargo, solamente el *primer* árbol de la descripción de comportamiento es accesible desde el exterior de dicha descripción de comportamiento. Se considera que todos los demás árboles son pasos de prueba locales a la descripción del comportamiento y que, por consiguiente, no resultan accesibles externamente.

Cabe observar que solamente los casos de prueba son ejecutables directamente, en tanto que los pasos de prueba sólo son ejecutables si están adjuntados a un caso de prueba o a un paso de prueba en el que pueda efectuarse el seguimiento de su punto de adjunción hasta un caso de prueba (ya sea de una forma directa o a través de otros pasos de prueba adjuntados). No es posible efectuar la adjunción entre casos de prueba.

La referencia a árbol puede consistir en identificadores de paso de prueba o identificadores de árbol, donde:

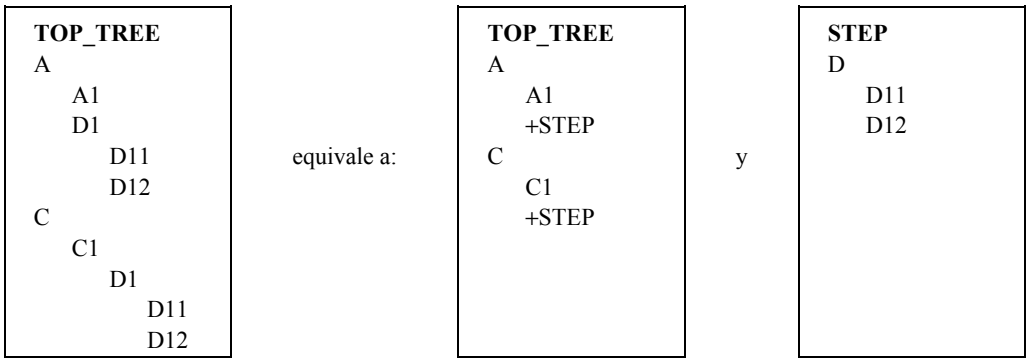
a) Un identificador de paso de prueba representa la adjunción de un paso de prueba que reside en la biblioteca de pasos de prueba. El paso de prueba se referencia mediante su identificador exclusivo.

b) Un identificador de árbol será el nombre de uno de los árboles de la descripción de comportamiento actual. Se trata de una adjunción de un árbol local.

15.13.3 Aspectos básicos de la adjunción de árbol

Dado un árbol de comportamiento, se puede segregar partes de dicho árbol en forma de árboles de comportamiento separados, por ejemplo, pasos de prueba. Los puntos en los que se ha desgajado un paso de prueba del árbol original se señalan mediante el símbolo de adjuntar (+), seguido del nombre asignado al paso de prueba.

EJEMPLO 89 – Partición de un árbol grande en dos árboles más pequeños:



Esta operación se puede realizar no sólo en el árbol de comportamiento principal del caso de prueba (árbol raíz), sino también en los pasos de prueba segregados de él. El árbol adjuntado puede ser un árbol local o un elemento de la biblioteca de pasos de prueba.

La operación de adjunción puede definirse de una forma más general que la mera reinserción de pasos de prueba completos:

- Un árbol adjuntado no tiene por qué contener trayectos completos hasta las hojas del árbol al que está adjuntado (su *árbol llamante*). En su lugar, se pueden especificar en el árbol llamante algunos comportamientos ulteriores comunes a todos los trayectos del árbol adjuntado, por ejemplo como comportamiento subsiguiente a la línea de adjunción.
- Algunas líneas (incluso hasta el nivel máximo) del paso de pruebas adjuntado pueden, de nuevo, tener la forma +SOME_SUBTREE que indican la adjunción de ulteriores pasos de prueba.
- Los pasos de prueba adjuntados se pueden parametrizar.

15.13.4 El significado de adjunción de árbol

En la lista que sigue se define la semántica de ejecución de la adjunción de árbol:

a) La línea de adjunción (por ejemplo +STEP) del árbol de comportamiento (por ejemplo, TOP_TREE) es, formalmente, una alternativa (por ejemplo Ai) de un conjunto ordenado de alternativas:

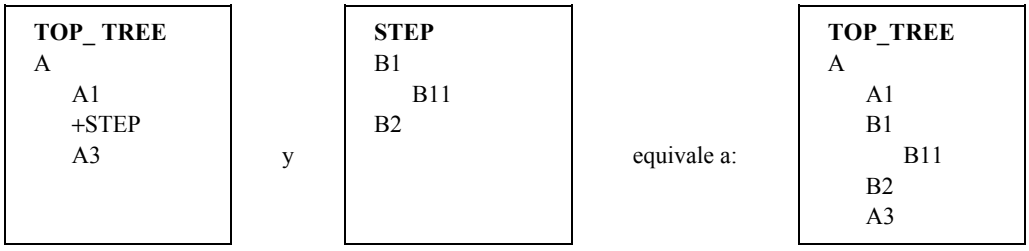
(A1, ... , Ai, ... , An)

La adjunción de STEP en esta posición implica la expansión de TOP_TREE insertando las alternativas STEP superiores del paso de prueba (por ejemplo, B_1, ... , B_m) en esta secuencia, lo que conduce a una nueva secuencia de alternativas:

(A1, ... , A(i-1), B1, ... , Bm, A(i+1), ... , An)

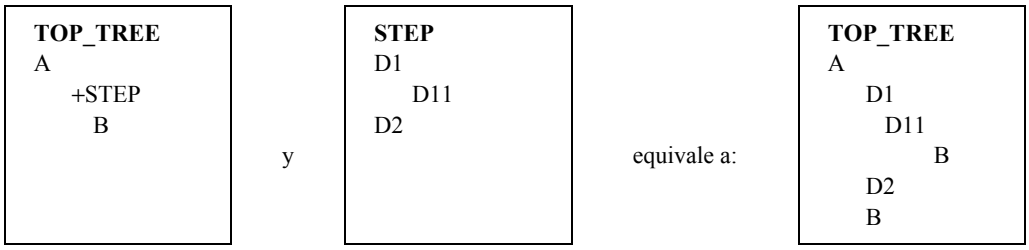
Cualquier comportamiento subsiguiente a las B se adjuntará a ellas.

EJEMPLO 90 – Expansión de un paso de prueba:



b) Cualquier comportamiento subsiguiente a la línea +STEP en el árbol se transformará en un comportamiento subsiguiente a todas las hojas del STEP adjuntado, expandidas en el árbol.

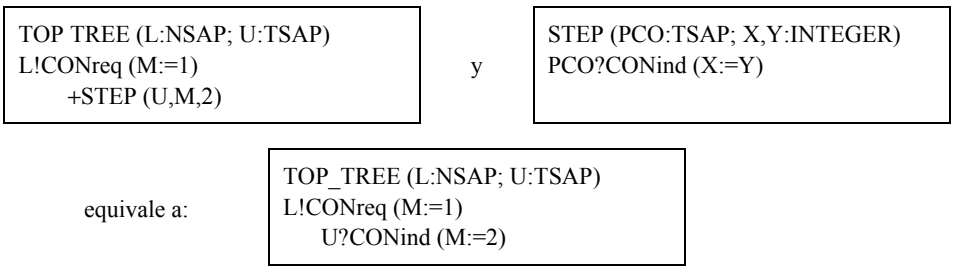
EJEMPLO 91 – Comportamiento subsiguiente a un ATTACH:



c) Cuando en un constructivo ATTACH se utilice una lista de parámetros reales, cada uno de los parámetros reales deberá sustituir a cada uno de los parámetros formales correspondientes, haciendo uso de una sustitución textual simple. Esta sustitución se realizará de conformidad con las siguientes reglas de determinación de ámbito:

1) Los parámetros reales del ATTACH de un árbol local, reemplazarán los parámetros formales correspondientes solamente en forma directa dentro de ese árbol local.
2) Los parámetros reales del ATTACH del árbol raíz de un paso de prueba reemplazarán todas las ocurrencias de los correspondientes parámetros formales dentro del árbol raíz y de todos los árboles locales directamente dentro del paso de prueba.
3) Cuando se efectúe la adjunción de un árbol parametrizado:
   - el número de parámetros reales será igual al número de parámetros formales;
   - cada parámetro real tomará el valor de un elemento de su tipo de parámetro formal correspondiente; y
   - los parámetros reales y formales de pasos de prueba se utilizarán de modo que sólo la TTCN válida es creada por sustitución textual.

EJEMPLO 92 – Sustitución de parámetros:



EJEMPLO 93 – Reglas de determinación de ámbito para la sustitución de los parámetros:

Comportamiento dinámico de caso de prueba					
Nombre del caso de prueba : TEST_STEP_1 (X,Y : INTEGER)					
Grupo : TTCN_EXAMPLES/PARAMS/STEPS/					
Finalidad : Ilustrar las reglas de determinación de ámbito para la sustitución de parámetros					
Valor por defecto :					
Comentarios :					
N.º	**Etiqueta**	**Descripción de comportamiento**	**Ref. de constricciones**	**Veredicto**	**Comentarios**
1		? A	A1		
2		+ TEST_STEP_2 (X)			
3		+ LOCAL (5)			
		LOCAL (F : INTEGER)			
4		! B	B1		
5		(TC_VAR = F + Y)		PASS	
Comentarios detallados: Cuando TEST_STEP_1 es adjuntado por un árbol llamante, todas las apariciones de los parámetros formales X e Y en la totalidad del paso de prueba (incluido dentro del árbol local LOCAL) se sustituyen por los valores reales suministrados. Obsérvese que los parámetros formales X e Y no se sustituyen automáticamente por los reales dentro TEST_STEP_2. Sin embargo, en el constructivo ATTACH "+TEST_STEP_2 (X)" el valor del parámetro real sustituye al valor formal X. Esto hace que se utilice el valor X del parámetro real (en TEST_STEP_1) en lugar de cualesquiera parámetros formales que aparezcan en la declaración TEST_STEP_2. Obsérvese por último, que el parámetro real (constante) 5 sustituye al formal "F", cuando se adjunta el árbol LOCAL. Dicha sustitución sólo tiene lugar dentro del árbol local.					

15.13.5 Paso de constricciones parametrizadas

Pueden pasarse constricciones a los pasos de prueba en forma de parámetros. Si la constricción posee una lista de parámetros formales, se transferirá esa constricción junto con una lista de parámetros reales. Los parámetros reales de la constricción estarán ya vinculados en el punto de adjunción.

EJEMPLO 94 – Paso de una constricción parametrizada:

Supóngase que la constricción C1 posee un solo parámetro formal del tipo INTEGER. TOP_TREE anexiona STEP y pasa C1 como parámetro. Obsérvese que la referencia a constricciones en STEP no está parametrizada:

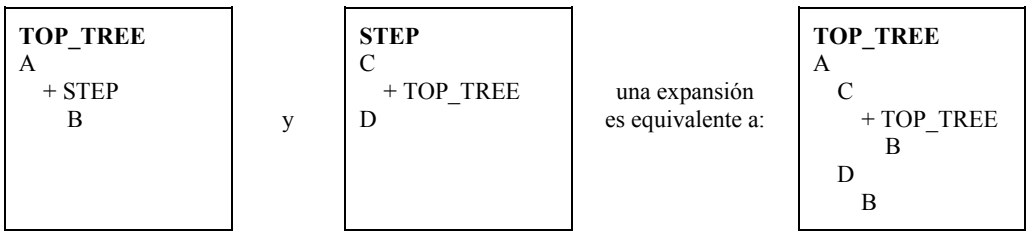
```
TOP TREE
    :
    + STEP (C1(3))
    :
```

```
STEP (PARA:A_PDU)
    :
    ! A_PDU              PAR
    :
```

15.13.6 Adjunción recursiva de árbol

Puesto que una adjunción de árbol funciona recursivamente (STEP puede contener una línea +SOME_OTHER_TREE), la semántica de expansión del árbol nunca puede llegar a un árbol exento de líneas de adjunción.

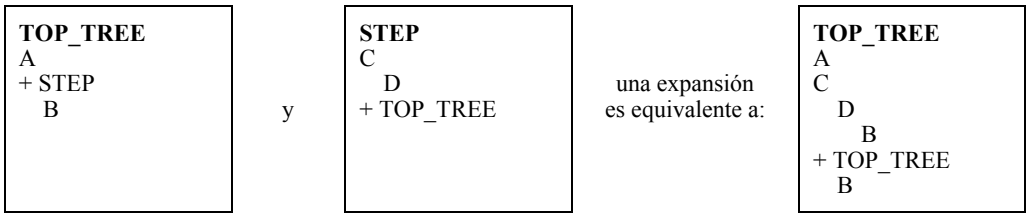
EJEMPLO 95 – Adjunción recursiva legal de un árbol:



Un árbol no podrá adjuntarse a sí mismo directa ni indirectamente en su nivel máximo de sangrado.

NOTA – No es necesario expandir paso de prueba que no vaya a ser ejecutado o cualesquiera alternativas más allá del nivel vigente hasta que se haya seleccionado una alternativa desde el nivel vigente.

EJEMPLO 96 – Adjunción recursiva ilegal de un árbol:



15.13.7 Adjunción de árboles y valores por defecto

Antes de adjuntar un árbol en cualquier lugar hay que completar la expansión de los valores por defecto en el árbol (véase 15.18.5).

NOTA – Cuando en una descripción de comportamiento se utilicen adjunciones de árboles y valores por defecto se pondrá especial cuidado.

15.14 Etiquetas y constructivo GOTO

Puede colocarse una etiqueta en la columna de etiquetas de cualquier línea de enunciado del árbol de comportamiento.

NOTA 1 – Cuando se ejecute una entrada en el árbol de comportamiento para la que se haya especificado una etiqueta, deberá anotarse dicha etiqueta en el registro cronológico de conformidad, de forma que pueda asociarse con el registro de la ejecución de esa entrada.

Dentro de un árbol de comportamiento, se puede especificar un GOTO a una etiqueta siempre que ésta se asocie con la primera de un conjunto de alternativas, una de las cuales un nodo predecesor del punto desde el cual se va a efectuar el GOTO. El constructivo GOTO sólo se podrá usar para efectuar saltos dentro de un árbol, a saber, un árbol raíz de caso de prueba, un árbol de paso de prueba, un árbol por defecto o un árbol local. En consecuencia, una etiqueta utilizada en un constructivo GOTO aparecerá dentro del mismo árbol en que se utiliza GOTO. No podrá efectuarse ningún GOTO al primer nivel de alternativas de árboles locales, pasos de prueba o por defecto.

Un GOTO no se referirá a una etiqueta antes que a un constructivo ACTIVATE que sea un nodo predecesor del GOTO.

Un GOTO se especificará colocando una flecha (→) o la palabra clave GOTO, seguida del nombre de la etiqueta, en una línea de enunciado propia, del árbol de comportamiento.

Cualquier etiqueta será exclusiva dentro de un árbol. Si se ejecuta un GOTO, el caso de prueba proseguirá con el conjunto de alternativas a que hace referencia la etiqueta.

Los GOTO serán siempre incondicionales y, por consiguiente, se ejecutarán siempre.

NOTA 2 – Puede colocarse una expresión booleana como antecedente inmediato de un GOTO, con lo cual se obtiene el efecto de un salto condicional.

EJEMPLO 97 – Utilización de GOTO:

Comportamiento dinámico de caso de prueba					
Nombre de caso de prueba : GOTO_EX1					
Grupo : TTCN_EXAMPLES/GOTO_EXAMPLE1/					
Objeto : Ilustrar la utilización de etiquetas y GOTO					
Valor por defecto :					
Comentarios :					
N.º	**Etiqueta**	**Descripción de comportamiento**	**Ref. de constricciones**	**Veredicto**	**Comentarios**
1	LA	! A	A1		
2	LB	? B	B1		
3	LB2	+ B-tree			
4	LC	? C	C1		
5	LD	[D=1]			
6		GOTO LA			
7	LE	[E=1]			
8	LF	! F	F1	FAIL	
Comentarios detallados: En este ejemplo se muestra un salto a LA. Desde la misma posición en ese árbol, también sería admisible saltar a LB o a LD, pero no a LB2 o a LF (porque el conjunto de alternativas no contiene un nodo predecesor del punto desde el que salta) ni a LC o a LE (porque no son la primera del conjunto de alternativas).					

15.15 El constructivo REPEAT

En esta cláusula se describe el mecanismo que se utilizará en las descripciones de comportamiento para repetir un paso de prueba cierto número de veces.

La referencia al árbol será una referencia a un árbol local o a un paso de prueba definido en la biblioteca de pasos de prueba. Por lo que respecta a las reglas de adjunción, véase 15.13. El constructivo REPEAT tiene el siguiente significado: en primer lugar se ejecuta el árbol referenciado por la referencia a árbol y a continuación se evalúa el calificador. Si la evaluación del calificador es TRUE, se completa la ejecución del constructivo REPEAT. En caso contrario, se ejecuta de nuevo el árbol y se evalúa nuevamente el calificador. Se repite el proceso hasta que la evaluación del calificador sea TRUE.

El constructivo REPEAT puede ejecutarse siempre, y normalmente será la última alternativa de una serie de enunciados en TTCN con el mismo nivel de sangrado, según permite el apartado a) de 15.9.5.3.

NOTA – Se recomienda utilizar el constructivo REPEAT, si es aplicable, en lugar de GOTO.

EJEMPLO 98 – Utilizaciones de REPEAT:

Comportamiento dinámico de caso de prueba					
Nombre del caso de prueba : RPT_EX1					
Grupo : TTCN_EXAMPLES/REPEAT_EXAMPLE1/					
Objeto : Ilustrar la utilización de REPEAT					
Valor por defecto :					
Comentarios :					
N.º	**Etiqueta**	**Descripción de comportamiento**	**Ref. de constricciones**	**Veredicto**	**Comentarios**
1		(FLAG:=FALSE)			
2		! A	A1		
3		REPEAT STEP1 UNTIL [FLAG]			
4		! D	D1	PASS	
		STEP1			
5		? B (FLAG:=TRUE)	B1		
6		? C (FLAG:=FALSE)	C1		
Comentarios detallados: En este ejemplo se describe una prueba capaz de recibir un número arbitrario de eventos C en el PCO probador inferior, hasta que se reciba el mensaje B esperado.					

15.16 La referencia a constricciones

15.16.1 Finalidad de la columna referencia a constricciones

Esta columna permite efectuar referencias a una constricción específica situada en una ASP, una PDU o un CM. Tales constricciones se definen en la parte constricciones (véanse las cláusulas 12, 13 y 14). La referencia a constricciones deberá estar presente junto con SEND, IMPLICIT SEND y RECEIVE. Si una ASP o CM carecen de parámetros, la referencia a constricciones optativa y no estará presente con ningún otro tipo de enunciado en TTCN.

La entrada de la columna Referencia a constricciones puede ser una referencia a constricciones real, el símbolo AnyValue ("?"), o un parámetro formal cuyo parámetro real contendrá una referencia a constricciones o un símbolo AnyValue. Si AnyValue se utiliza en lugar de una referencia a constricciones ello significa una constricción "don't care" ("no importa"), equivalente a una constricción con AnyOrNone ("*") en cada parámetro, campo o elemento.

EJEMPLO 99 – Referencia a constricciones sin lista de parámetros:

N_SAP? CR_PDU CRI

15.16.2 Paso de parámetros en referencias a constricciones

Una referencia a constricciones puede tener una lista de parámetros optativa para permitir la manipulación de valores de constricciones específicos desde el árbol de comportamiento.

La lista de parámetros reales deberá cumplir lo siguiente:

– el número de parámetros reales será igual al número de parámetros formales; y

– el valor de cada parámetro real será el de un valor de su tipo formal correspondiente o un símbolo de concordancia que pueda concordar con un valor de ese tipo de formato.

Si se pasa una constricción en forma de parámetro real y se declara esa constricción con una lista de parámetros formales, la constricción poseerá, asimismo, una lista de parámetros reales (que pueden estar anidados). Cuando se

utilice la constricción, todas las variables que aparezcan en la lista de parámetros deberán estar acotadas. Si se utiliza una variable no acotada, se produce un error de caso de prueba.

EJEMPLO 100 – Referencia a constricciones con lista de parámetros:

N_SAP?	N_DATAreq	D1(P1,CR1(P2))

Donde D1 es una constricción de petición N_DATOS con dos parámetros (parámetros reales P1 y CR1) y CR1 es una constricción con un solo parámetro (parámetro real P2).

15.16.3 Constricciones y calificadores y asignaciones

Si un evento es calificado y además contiene una referencia a constricciones, esto se interpreta en el sentido de que el evento concuerda si, y sólo si, se cumplen el calificador *y* la constricción.

Si un evento va seguido de una asignación y posee una referencia a constricciones y/o un calificador, esto se interpreta en el sentido de que la asignación se efectúa si, y sólo si, el evento se produce de conformidad con la definición dada anteriormente.

15.17 Veredictos

15.17.1 Introducción

Las inscripciones en la columna de veredicto de los cuadros de comportamiento dinámico podrán ser:

- un resultado preliminar que figurará entre paréntesis; o
- un veredicto final explícito.

Una entrada de cualquiera de estos tipos no podrá producirse en una línea vacía, ni en los siguientes enunciados en TTCN.

a) un constructivo ATTACH;

b) un constructivo REPEAT;

c) un GOTO;

d) un IMPLICIT SEND.

NOTA – Durante la ejecución del caso de prueba, cada vez que se produzca una entrada en un árbol de comportamiento para el que exista una entrada correspondiente en la columna de veredicto del caso de prueba abstracta, la información de esa columna de veredicto ha de anotarse en el registro cronológico de conformidad, de forma que quede asociada con el registro de esa entrada en el árbol de comportamiento.

15.17.2 Resultados preliminares

En cada caso de prueba se dispone de una variable predefinida, llamada R, del tipo predefinido R_TYPE, para el almacenamiento de todos los resultados intermedios. Estos valores son identificadores predefinidos y como tales son sensibles al tipo de letra, mayúscula o minúscula, con que se escriban.

R se puede utilizar dondequiera que puedan utilizarse otras variables de caso de prueba, salvo en el primer miembro de una sentencia de asignación. Se trata, por tanto, de una variable de lectura solamente, excepto en el caso de modificaciones de su valor ocasionadas por inscripciones en la columna de veredictos (como se especifica más adelante).

Si en la columna de veredicto ha de especificarse un resultado preliminar, éste será uno de los siguientes:

a) (P) o (PASS), indicando que se han cumplido algunos aspectos de la finalidad de la prueba.

b) (I) o (INCONC), indicando que ha ocurrido algo que hace que el caso de prueba no sea concluyente para algún aspecto de la finalidad de la prueba.

c) (F) o (FAIL), indicando que se ha producido algún error de protocolo o que se ha registrado un fallo en algún aspecto de la finalidad de la prueba.

NOTA 1 – Los términos PASS o P, FAIL o F e INCONC o I, son palabras clave utilizadas en la columna de veredictos solamente. Los identificadores predefinidos (*éxito*) (*pass*), (*fracaso*) (*fail*), (*no concluyente*) (*inconc*) y (*ninguno*) (*none*), son valores que representan los posibles contenidos de la variable predefinida R. Estos identificadores predefinidos se utilizarán para la comprobación de la variable R en líneas de comportamiento solamente.

Cuando se registre un resultado preliminar, puesto a que se ha ejecutado la entrada correspondiente en el árbol de comportamiento, se modificará el valor de la variable de caso de prueba predefinida R, según el cuadro 7:

Cuadro 7/X.292 – Cálculo de la variable R

Valor vigente de R	Entrada en la columna veredicto		
	(PASS)	(INCONC)	(FAIL)
Ninguno	Éxito	No concluyente	Fracaso
Éxito	Éxito	No concluyente	Fracaso
No concluyente	No concluyente	No concluyente	Fracaso
Fail	Fracaso	Fracaso	Fracaso

NOTA 2 – El orden de precedencia de más bajo a más alto es, en consecuencia, N, P, I, F. Aun cuando R tome el valor *fracaso* puede ser útil para registrar un resultado preliminar de P o I, a fin de inscribir en el cuaderno de conformidad que una P o I es apropiada para algún aspecto de la finalidad de la prueba, a pesar de que esto no cambie el valor de R.

15.17.3 Veredicto final

Cuando hay que especificar un veredicto final explícito en la columna de veredicto, éste será uno de los siguientes:

a) P o PASS indica que debe registrarse un veredicto de éxito.

b) I o INCONC, indica que debe registrarse un veredicto no concluyente.

c) F o FAIL, indica que debe registrarse un veredicto de fracaso.

d) La variable predefinida R indica que el valor de R se tomará como veredicto final, a menos que dicho valor sea (*ninguno*) (*none*), en cuyo caso se registra un error de caso de prueba en lugar de un veredicto final.

Cuadro 8/X.292 – Cálculo del veredicto final R

Valor vigente de R	Entrada en la columna veredicto			
	PASS	INCONC	FAIL	R
Ninguno	Éxito	No concluyente	Fracaso	*Error*
Éxito	Éxito	No concluyente	Fracaso	Éxito
No concluyente	*Error*	No concluyente	Fracaso	No concluyente
Fracaso	*Error*	*Error*	Fracaso	Fracaso

Si durante la ejecución de un caso de prueba se especifica un veredicto final, dicha especificación concluirá el caso de prueba. Para asegurar la conformidad con la Rec. UIT-T X.291, sólo se podrá especificar un veredicto final explícito cuando el caso de prueba haya vuelto a un estado estable adecuado (por ejemplo, el estado de comprobación en reposo).

NOTA 1 – La conclusión del caso de prueba originada por la especificación de un veredicto final explícito es necesaria, por ejemplo, si se ha alcanzado el estado estable en un paso de prueba agregado, cuando en el árbol llamante se especifica el comportamiento subsiguiente.

Si se alcanza la hoja del árbol de comportamiento sin que se haya especificado un veredicto final explícito, se determinará el veredicto final como en el caso d) anterior (esto es, como si se hubiese puesto R en la columna de veredicto).

Cuando haya que registrar un veredicto final explícito distinto de R, se comparará ese veredicto con el valor de R, para determinar si son o no coherentes. Si el valor de R es *fail*, se considerará que un veredicto final PASS o INCONC no es coherente. Si el valor de R es *inconc*, se considerará un veredicto final PASS como incoherente. Si se produce alguna de estas incoherencias, ello equivale a un error de caso de prueba.

NOTA 2 – En tales casos, se inscribirá "error de caso de prueba" en el registro cronológico de conformidad.

15.17.4 Veredictos y OTHERWISE

Un enunciado OTHERWISE no podrá conducir a un veredicto PASS, sino a un veredicto FAIL, ya que OTHERWISE podría concordar con un evento de prueba no válido.

15.17.5 Asignación de veredicto en la TTCN concurrente

En la TTCN concurrente, el veredicto final es asignado por el MTC, bien explícitamente en la columna de veredicto, o bien implícitamente como consecuencia de la terminación del MTC. Los resultados de prueba preliminares se mantienen en la variable de resultado global, que es accesible al MTC como la variable de caso de prueba R. La variable de resultado global es actualizada siempre que se registra un resultado preliminar o veredicto en la columna de veredicto mediante una línea de comportamiento de MTC en concordancia. Si el MTC termina sin la asignación de un veredicto explícito, el veredicto deberá entonces determinarse como si se hubiera colocado R en la columna de veredicto (15.17.3 d).

Además, cada PTC registrará al menos un resultado preliminar. Este resultado preliminar se mantiene en su variable de resultado local, la cual es accesible al PTC como su variable de caso de prueba R. Cuando un PTC asigna un resultado preliminar, mediante una entrada en la columna de veredicto de una línea de comportamiento de PTC en concordancia (esté, o no, la entrada encerrada entre paréntesis), tanto su variable de resultado local como la variable de resultado global se actualizan automáticamente mediante el algoritmo especificado en 15.17.2. En un PTC, una entrada en la columna de veredicto que no está encerrada entre paréntesis no es un veredicto final, pero deberá producir la terminación del PTC si esa línea de comportamiento concuerda.

La terminación del MTC antes de la terminación de todos los PTC dará como resultado un error de caso de prueba.

Cuando el MTC utiliza la variable R en una expresión booleana o en una asignación, tiene acceso a la variable de resultado global. Cuando el PTC utiliza la variable R en una expresión booleana o en una asignación, tiene acceso a su variable de resultado local. El MTC puede también acceder a una variable de resultado local propia con la variable de caso de prueba predefinida MTC_R en lugar de R. MTC_R es del tipo predefinido R_TYPE. MTC_R se actualiza siempre que una línea de comportamiento de MTC en concordancia registra en la columna de veredicto un resultado preliminar, pero no se ve afectada por los resultados preliminares de los PTC.

El valor de una variable de resultado local de PTC sólo puede comunicarse a otro PTC mediante mensajes de coordinación. El valor de las variables de resultado global y de resultado local del MTC sólo pueden comunicarse a un PTC a través de CM.

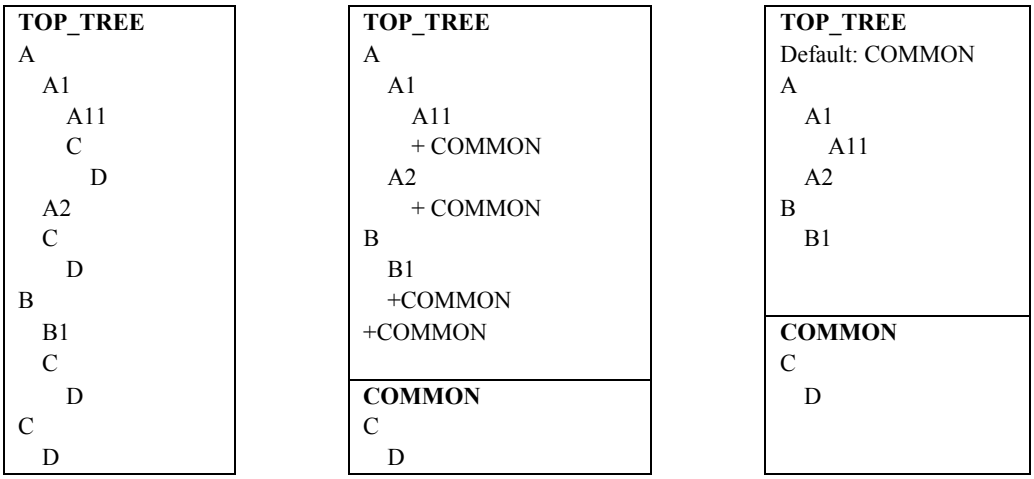
15.18 El significado de valores por defecto

15.18.1 Introducción

En muchos casos se utilizará el comportamiento por defecto para destacar un conjunto de trayectos interesantes a través de una prueba, mediante la declaración de las alternativas comunes menos interesantes (más su comportamiento subsiguiente) como comportamiento por defecto.

Podría lograrse el mismo efecto, aunque con menos concisión, mediante la adjunción del paso de prueba (por ejemplo, +DEFAULT) como una última alternativa general adicional. Por oposición a la adjunción de árbol, el comportamiento por defecto se expande a muchos puntos del árbol al cual está asociado. Esta propiedad requiere una utilización cuidadosa de los valores por defecto.

EJEMPLO 101 – Identificación de un árbol por defecto:



A un comportamiento por defecto no se le podrá especificar ningún otro comportamiento por defecto; es decir, un comportamiento por defecto no podrá tener, él mismo, un comportamiento por defecto. En los árboles de comportamiento por defecto no podrán utilizarse adjunciones de árbol; esto es, los árboles de comportamiento por defecto no adjuntarán casos de prueba. Los casos de prueba o pasos de prueba no serán referidos como valores por defecto.

Para la ejecución de un caso de prueba no es necesario expandir valores por defecto por cualquier parte en todos los árboles que se refieren a ellos. Esto puede verse a partir de una descripción operacional del significado de los valores por defecto: al intentar satisfacer una secuencia de alternativas (lo que puede requerir intentos repetidos) cada vez que tales alternativas no concuerdan, se intenta, asimismo, el primer nivel de alternativas del comportamiento por defecto. Si tampoco concuerda ninguna de éstas, se reintenta la secuencia con los nuevos estados de los temporizadores y las colas en todos los PCO afectados. Si se produce una concordancia en el valor por defecto, se prosigue en este punto con el comportamiento por defecto.

Para asegurar la no ocurrencia de comportamiento subsiguiente alguno tras la ejecución de un comportamiento por defecto, la ejecución de una hoja de un árbol por defecto, diferente de un enunciado RETURN, deberá causar la terminación del caso de prueba. Para llevar a cabo esta terminación, en un árbol por defecto, las hojas que no tienen veredicto o resultado preliminar en la columna de veredicto son implícitamente provistas de una entrada de "R" en la columna de veredicto, y las hojas que tienen un resultado preliminar en la columna de veredicto tienen ese resultado preliminar implícitamente transformado en un veredicto final.

15.18.2 Referencias de valores por defecto

Los comportamientos de caso de prueba y paso de prueba hacen referencia a una lista de comportamientos por defecto en la biblioteca de valores por defecto a través de la entrada de valores por defecto en el encabezamiento del cuadro.

Cada referencia en esta lista localiza un valor por defecto mediante su identificador exclusivo. El identificador de valores por defecto será una referencia a un valor por defecto definido en la biblioteca de valores por defecto.

Los valores por defecto se pueden parametrizar. La lista de parámetros reales cumplirá lo siguiente:

a) igualdad entre el número de parámetros reales y el número de parámetros formales;

b) cada parámetro real tomará el valor de un elemento de su tipo formal correspondiente; y

c) todas las variables que figuren en la lista de parámetros estarán acotadas cuando se invoque la constricción.

EJEMPLO 102 – Referencia a valores por defecto:

Comportamiento dinámico de caso de prueba					
Nombre del caso de prueba : DEF_EX1					
Grupo : TTCN_EXAMPLES/DEFAULT_EXAMPLE1/					
Finalidad : Ilustrar la utilización de valores por defecto.					
Valor por defecto : DEF1 (L)					
Comentarios : El árbol del ejemplo 69 puede subdividirse en este caso de prueba con el comportamiento por defecto DEF1.					
N.º	**Etiqueta**	**Descripción de comportamiento**	**Ref. de constricciones**	**Veredicto**	**Comentarios**
1		L ! CONNECTrequest	CR1		Petición …
2		L ? CONNECTconfirm	CC1		… Confirmación
3		L ! DATArequest	DTR1		Enviar datos
4		L ? DATAindication	DTI1		Recibir datos
5		L ! DISCONNECTrequest	DSCR1	PASS	Aceptar
Comentarios detallados:					

Comportamiento dinámico por defecto					
Nombre por defecto : DEF1 (X : XSAP) **Grupo** : TTCN_EXAMPLES/DEFAULTS_LIB/DEFAULT_1/ **Objeto** : Ilustración de un comportamiento por defecto simple **Valor por defecto** : **Comentarios** :					
N.º	**Etiqueta**	**Descripción de comportamiento**	**Ref. de constricciones**	**Veredicto**	**Comentarios**
1		X ? DISCONNECTindication	DSC2	INCONC	Prematuro

NOTA – Sintácticamente, el comportamiento por defecto del segundo cuadro del ejemplo anterior adjunta X?DISCONNECTindication como una alternativa a cada uno de los enunciados L! y L? del primer cuadro. Sin embargo, la adjunción del árbol por defecto como alternativa a un enunciado L! que siempre se produce con éxito no tiene sentido.

15.18.3 El enunciado RETURN (devolución)

El enunciado RETURN es una extensión de las capacidades de la descripción del comportamiento por defecto. Un enunciado RETURN sólo se utilizará en un árbol por defecto.

Cuando se lleva a cabo la expansión por defecto de un árbol, la ejecución de un enunciado RETURN hará que se continúe en la primera alternativa del conjunto de alternativas que ha hecho que se intente el comportamiento por defecto.

15.18.4 El enunciado ACTIVATE

El enunciado ACTIVATE permite la activación de un conjunto de comportamientos por defecto. En vez de encontrarse implícitamente activos durante el caso de prueba, los valores por defecto se pueden activar selectivamente mediante el enunciado ACTIVATE. El comportamiento por defecto así activado se intenta de este modo en el orden especificado por el enunciado ACTIVATE, por ejemplo, ACTIVATE (Def_1, Def_2) hará que se ejecute Def_1 antes de Def_2 cuando se precisa el comportamiento por defecto.

El comportamiento por defecto especificado en un enunciado ACTIVATE reemplaza a cualquier comportamiento por defecto activo, incluido el comportamiento por defecto especificado en un encabezamiento de caso de prueba o paso de prueba.

Un ACTIVATE que tenga una lista de referencias por defecto vacía, es decir, ACTIVATE(), desactiva todos los comportamientos por defecto.

15.18.5 Valores por defecto y adjunción de árbol

Cuando se utiliza la adjunción de árbol, es importante tener un conocimiento claro de cómo se aplican los valores por defecto tanto al árbol llamante como al paso de prueba adjuntado. Para evitar efectos colaterales ocultos se definen los valores por defecto aplicables dentro de un paso de prueba adjuntado como aquellos que se han especificado en el cuadro que define a ese paso de prueba. Por consiguiente, si en la biblioteca de pasos de prueba se define el paso de prueba, los valores por defecto se especifican en el encabezamiento del cuadro de comportamiento del paso de prueba. De manera alternativa, si el paso de prueba se define localmente, en el mismo cuadro de comportamiento que el del árbol llamante, se aplicarán los mismos valores por defecto tanto al árbol llamante como al paso de prueba adjuntado.

El valor por defecto especificado para un árbol particular no se aplica al nivel superior de las alternativas de ese árbol, a menos que el árbol sea el árbol raíz de un caso de prueba, a fin de evitar múltiples inserciones de valores por defecto dentro de un conjunto de alternativas.

Para generar el desarrollo correcto de un árbol, es necesario expandir los valores por defecto:

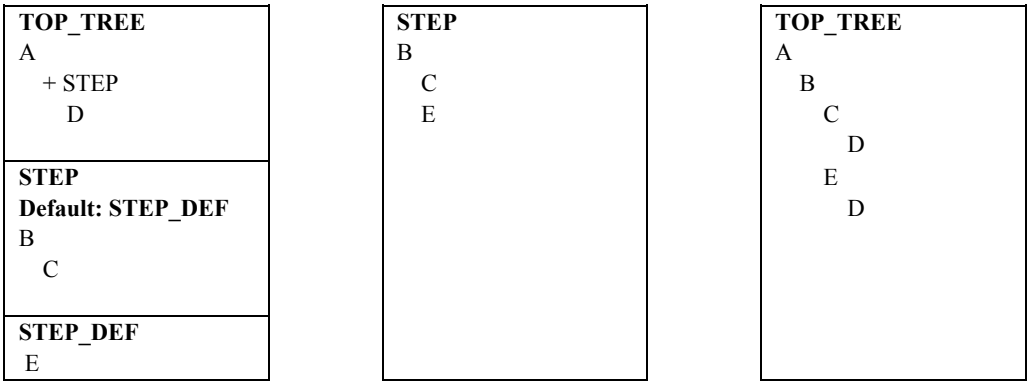
a) antes de que se expanda el árbol como árbol adjuntado; y

b) antes de que se expandan cualesquiera de los pasos de prueba adjuntados al árbol.

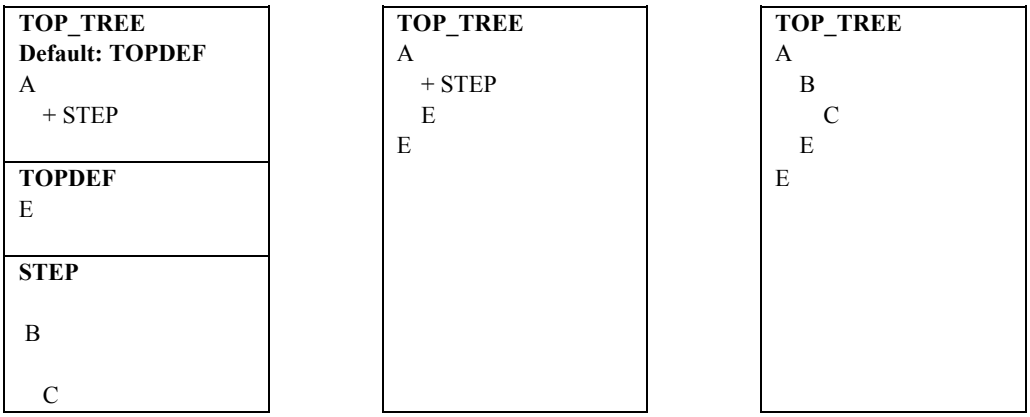
La expansión de los valores por defecto es, en consecuencia, una actividad local de un árbol aislado y comprende la adjunción del árbol por defecto a la parte inferior de cada conjunto de alternativas dentro del árbol (salvo el conjunto superior de alternativas de cualquier árbol distinto del árbol raíz de un caso de prueba).

Las normas de expansión de valores por defecto se aplican igualmente en el caso en que un conjunto de alternativas contenga un evento OTHERWISE.

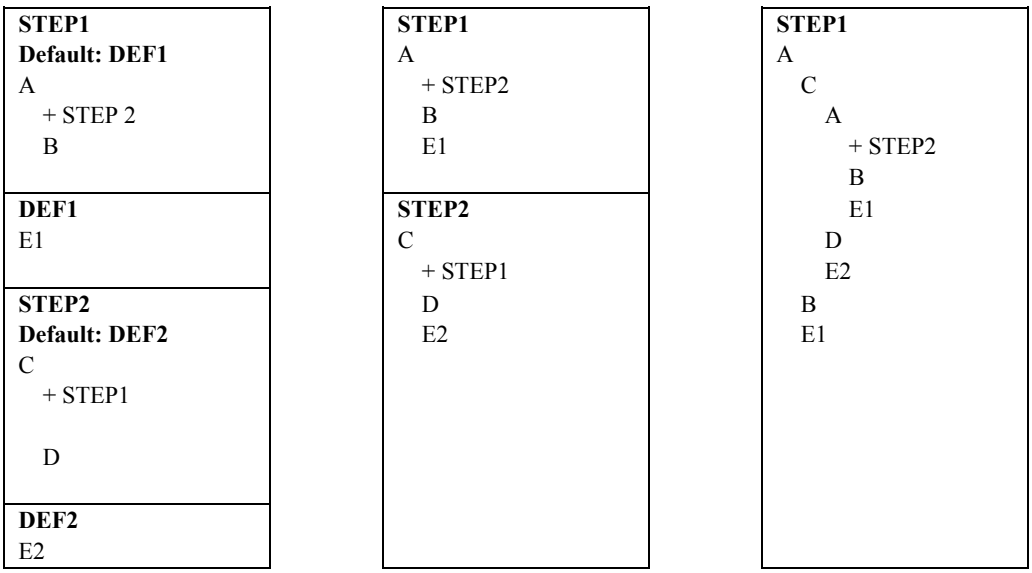
EJEMPLO 103 – Ubicación de un valor por defecto frente a un paso de prueba:



EJEMPLO 104 – Ubicación de un valor por defecto frente a un árbol llamante:



EJEMPLO 105 – Caso de adjudicación cíclica de árbol:



NOTA – No se aconsejan tales adjunciones cíclicas.

15.18.6 Adjunción de árboles, valores por defecto, Activate y Return

Si se utiliza la operación ACTIVATE dentro de un caso de prueba, la semántica de valores por defecto y adjunción de árbol sólo se puede describir dinámicamente, y no estáticamente. En efecto, la semántica operacional de valores por defecto del anexo B se especifica en términos de expansión de árbol dinámica, un nivel en un momento dado.

En este modelo de semántica dinámica, la especificación en el encabezamiento de una lista de valores por defecto equivale a prefijar el árbol de comportamiento con una operación ACTIVATE de esa lista de árboles por defecto. En un paso de prueba, la colocación en el encabezamiento de una lista de valores por defecto equivale a colocar una operación ACTIVATE de esa lista de árboles por defecto entre cada alternativa del primer nivel de alternativas y su comportamiento subsiguiente. Si se ha adjuntado un paso de prueba que no tiene valores por defecto especificados en el encabezamiento, las operaciones ACTIVATE implicadas no tienen parámetros y por consiguiente desactivan todos los valores por defecto.

Como el comportamiento subsiguiente a una adjunción de árbol toma sus valores por defecto del contexto del árbol llamante en lugar de tomarlos del paso de prueba adjuntado, la adjunción de árbol implica la inserción de una ACTIVATE después de cada nodo de hoja de no terminación (es decir, que no asigna un veredicto) para restaurar los valores por defecto a los valores del contexto en el cual se ha efectuado la adjunción. Si el nodo de hoja es una operación RETURN, ello implica que ACTIVATE tiene que ir antes de RETURN para asegurar que tenga efecto antes de retroceder en el contexto exterior.

La consecuencia de una combinación de valores por defecto y adjunción de árbol se ilustra en el caso de prueba que se muestra en el ejemplo 106.

EJEMPLO 106 – Ejemplo de caso de prueba X-Def1 para ilustrar el significado de los valores por defecto:

Comportamiento dinámico de caso de prueba			
Nombre de paso de prueba : X_Def1 **Grupo** : **Finalidad** : **Valor por defecto** : D1, D2			
E	**Descripción de comportamiento**	**RefC**	**V**
	X +T1 Y Z +T2		

Comportamiento dinámico de caso de prueba			
Nombre de paso de prueba : T1 **Grupo** : **Finalidad** : **Valor por defecto** : D3, D4			
E	**Descripción de comportamiento**	**RefC**	**V**
	A B C		

Comportamiento dinámico de caso de prueba			
Nombre de paso de prueba : T2 **Grupo** : **Finalidad** : **Valor por defecto** :			
E	**Descripción de comportamiento**	**RefC**	**V**
	D E F		

Este ejemplo de caso de prueba es equivalente al del ejemplo 107, en el cual se ha reemplazado la lista de valores por defecto en el encabezamiento de caso de prueba por una ACTIVATE de la misma lista de valores por defecto como primer enunciado TTCN del árbol de comportamiento.

EJEMPLO 107 – Especificación alternativa del ejemplo de caso de prueba X-Def1 que utiliza ACTIVATE:

Comportamiento dinámico de caso de prueba			
Nombre de paso de prueba : X_Def1 **Grupo** : **Finalidad** : **Valor por defecto** :			
E	**Descripción de comportamiento**	**RefC**	**V**
	ACTIVATE(D1,D2) X +T1 Y Z +T2		

El procesamiento de una ACTIVATE fija el contexto por defecto vigente. La progresión al siguiente nivel de alternativas adjunta la lista de árboles por defecto en el contexto por defecto vigente al siguiente nivel de alternativas.

De este modo, la evaluación del caso de prueba que se presenta en el ejemplo 107 puede proseguir como se ilustra en la figura 8. En primer lugar, se evalúa el enunciado ACTIVATE(D1,D2) para fijar el contexto por defecto a D1 y D2. Luego, suponiendo que X concuerda, se adjuntan D1 y D2 en el mismo nivel de alternativas que T1. Cuando T1 se expande a continuación, ACTIVATE(D3,D4) es insertado después del primer nivel de alternativas del caso de prueba, y ACTIVATE(D1,D2) es insertado después de dos nodos de hoja a fin de restaurar el contexto por defecto antes de que se alcance el comportamiento subsiguiente, Y. Suponiendo entonces que A concuerda, se adjuntan redundantemente los valores por defecto D1 y D2 en el mismo nivel de alternativas que la ACTIVATE. Sucede así porque el contexto por defecto vigente se añade siempre al nivel siguiente de alternativas, de manera indiscriminada, incluso en el caso en que el nivel siguiente de alternativas consta de un constructivo o seudoevento que concuerda siempre. Cuando se evalúa el nuevo enunciado ACTIVATE, se cambia el contexto por defecto al contexto aplicable al paso de prueba T1. Si entonces B concuerda, la evaluación avanza a la ACTIVATE que restaura el contexto por defecto al contexto aplicable al árbol raíz.

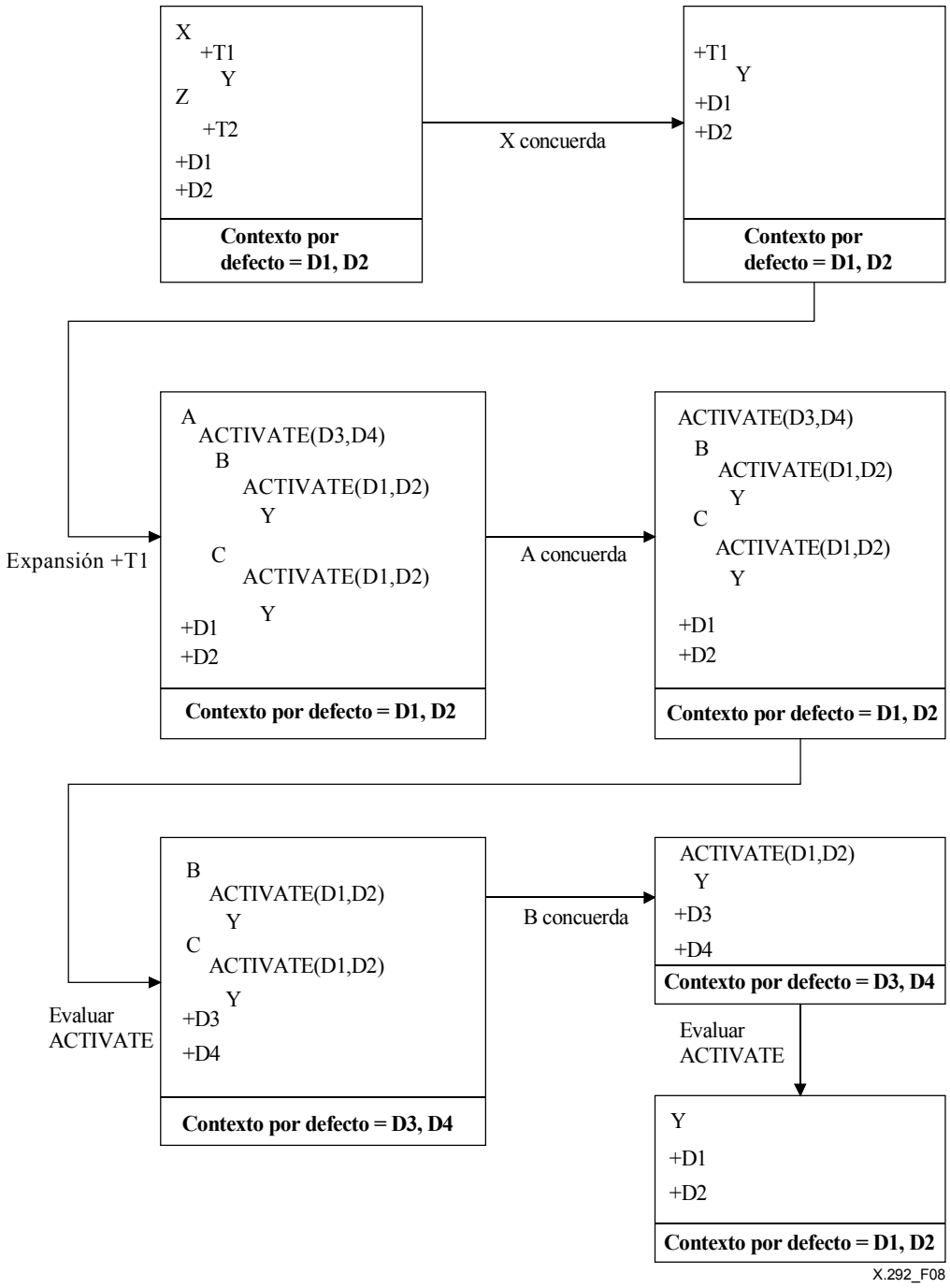


Figura 8/X.292 – Posible progresión de la evaluación de ejemplo en caso de prueba X_Def1

En el ejemplo 108 se presenta otro ejemplo de caso de prueba en el que se funden los valores por defecto especificados en los encabezamientos con un enunciado ACTIVATE explícito y adjunción de árbol.

EJEMPLO 108 – Ejemplo de caso de prueba X-Def2 para ilustrar el significado de valores por defecto y ACTIVATE:

Comportamiento dinámico de caso de prueba			
Nombre de paso de prueba : X_Def2 **Grupo** : **Finalidad** : **Valor por defecto** : D1			
E	**Descripción de comportamiento**	**RefC**	**V**
	X ACTIVATE(D2) +T S +T S		

Comportamiento dinámico de caso de prueba			
Nombre de paso de prueba : T **Grupo** : **Finalidad** : **Valor por defecto** : D3			
E	**Descripción de comportamiento**	**RefC**	**V**
	Y Z		

La progresión de la evaluación de este caso de prueba se ilustra en la Figura 9. En ella se presenta la progresión de la evaluación a lo largo de los dos trayectos principales del caso de prueba, y se muestra mostrando que el contexto por defecto aplicable al primer enunciado S se determina mediante ACTIVATE, mientras que el contexto por defecto aplicable al segundo S viene determinado por los valores por defecto especificados en el encabezamiento del caso de prueba; ninguno de estos dos contextos por defecto para los enunciados S se ve afectado por las adjunciones de árbol precedentes.

En la figura 9 se comienza mostrando las consecuencias de la expansión de la adjunción de T en el primer nivel de alternativas más la adición de los valores por defecto iniciales. Si X concuerda, la evaluación prosigue, vía el ACTIVATE(D2), hasta la segunda ocurrencia de la adjunción de T, con el contexto por defecto cambiado a D2 y la adjunción de D2 añadida en el mismo nivel de alternativas que T. Se expande entonces T, recordando insertar los dos enunciados ACTIVATE para fijar el contexto por defecto de caso de prueba y restaurar a continuación el contexto por defecto de árbol raíz. Estos cambios en el contexto por defecto se muestran en las dos etapas siguientes de la evaluación, en las que se supone en primer lugar que Y concuerda y seguidamente que concuerda Z. El resultado es S con la evaluación de una alternativa de la adjunción de D2 en el contexto por defecto D2.

El trayecto alternativo que se presenta en la figura 9 comienza con la concordancia de Y en vez de la concordancia de X. Esto provoca la progresión en el contexto por defecto D3, con lo cual, si Z concuerda, se restablece el contexto por defecto a D1. De este modo, este trayecto de la progresión ha descendido a S con una alternativa de la adjunción de D1 que se evalúa en el contexto por defecto D1.

En la progresión de la evaluación de los ejemplos de casos de prueba de las figura 8 y 9 no se ha mostrado la expansión de los árboles por defecto. Si al expandir el árbol por defecto se encuentra que éste, o cualquier árbol local asociado, contiene un constructivo RETURN, ello equivale a la colocación de una etiqueta en cabeza del conjunto vigente de alternativas, con el reemplazamiento de cada constructivo RETURN por una ACTIVATE a fin de restaurar el contexto por defecto del árbol llamante, seguida por un constructivo GOTO para ir a la nueva etiqueta.

Todos los nodos de hoja distintos de RETURN de un árbol de comportamiento por defecto en el cual se han adjuntado todos los subárboles locales, no tienen un comportamiento subsiguiente, por lo que deberán fijar un veredicto o dar como resultado un error de caso de prueba.

Para ilustrar lo anterior se utilizará el caso de prueba del ejemplo 109.

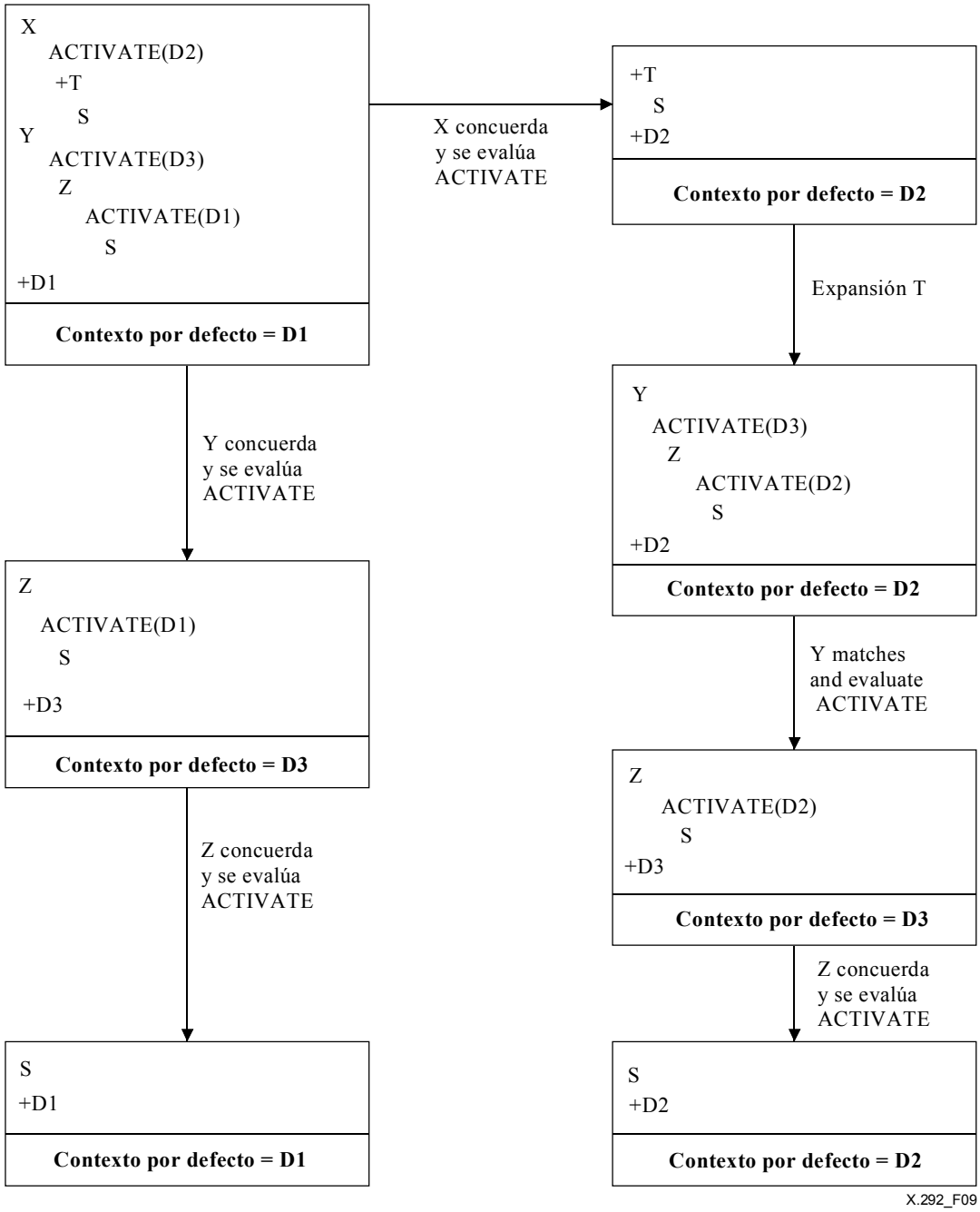


Figura 9/X.292 – Posible progresión de la evaluación de ejemplo en caso de prueba X_Def2

EJEMPLO 109 – Ejemplo de caso de prueba X_Def3 para ilustrar el significado de valores por defecto y RETURN:

Comportamiento dinámico de caso de prueba			
Nombre de paso de prueba : X_Def3 **Grupo** : **Finalidad** : **Valor por defecto** : D1			
E	**Descripción de comportamiento**	**RefC**	**V**
	X Y		P

Comportamiento dinámico por defecto			
Nombre por defecto : D1 **Nombre de paso de prueba** : **Objetivo** :			
E	**Descripción de comportamiento**	**RefC**	**V**
	C D RETURN E		 F

En la figura 10 se muestra la progresión de la evaluación de este ejemplo de caso de prueba. En primer lugar, se adjunta el árbol por defecto D1 en el primer nivel de alternativas del árbol raíz. A continuación se expande D1. Puesto que D1 contiene un enunciado RETURN, se trata de una expansión bastante compleja. El evento superior del nivel de alternativas en el cual tiene lugar la adjunción, se etiqueta con una etiqueta exclusiva, L. Como el árbol adjuntado es un valor por defecto, su contexto por defecto interno propio se encuentra vacío debido a que los valores por defecto no tienen sus propios valores, y por consiguiente se inserta un ACTIVATE sin argumentos después del primer nivel de alternativas del árbol adjuntado. Además, se reemplaza el enunciado RETURN por un ACTIVATE para restablecer el contexto por defecto D1, seguido en el nivel siguiente por GOTO L. En este momento, cuando se ha evaluado este árbol expandido, si C concuerda, continúa al enunciado ACTIVATE() junto con la adjunción redundante del contexto por defecto, D1. La evaluación de ACTIVATE() vacía el contexto por defecto. A continuación, si D concuerda, se evalúa el ACTIVATE(D1) para restaurar el contexto por defecto D1. Esto conduce al enunciado GOTO junto con otra adjunción redundante del contexto por defecto D1. La evaluación del GOTO devuelve entonces el proceso al estado en el cual se añadió la etiqueta L. La evaluación continuará para cerrar el bucle hasta que X, seguido de Y, concuerde para un éxito, o C, seguido de E, concuerde para un fracaso.

15.18.7 Valores por defecto y CREATE

El comportamiento por defecto no es heredado por los pasos de prueba que se utilizan en una operación CREATE, es decir, los pasos de prueba que ejecutan sus descripciones de comportamiento en paralelo con el MTC. Por ello, el ámbito del comportamiento por defecto en la TTCN concurrente es siempre local al MTC o un PTC.

Cuando se utiliza un paso de prueba en una operación CREATE, el comportamiento por defecto especificado en el encabezamiento del paso de prueba deberá aplicarse en el primer nivel de sangrado. Este uso de los valores por defecto es coherente con la aplicación de valores por defecto en casos de prueba.

15.18.8 Valores por defecto y mensajes CM

El comportamiento por defecto se aplica a cada conjunto de alternativas, incluso a los que sólo reciben mensajes CM. Esto puede ocasionar que las PDU que lleguen antes a la recepción del CM ejecutado, o las PDU que se encuentren ya en la cola del PCO pero que no hayan sido recibidas todavía, se eliminen de la cola del PCO. Para evitar la eliminación de unidades PDU de la cola del PCO, se desactivarán los valores por defecto mediante el constructivo ACTIVATE() como evento inmediatamente anterior al conjunto de alternativas que sólo recibe el o los CM.

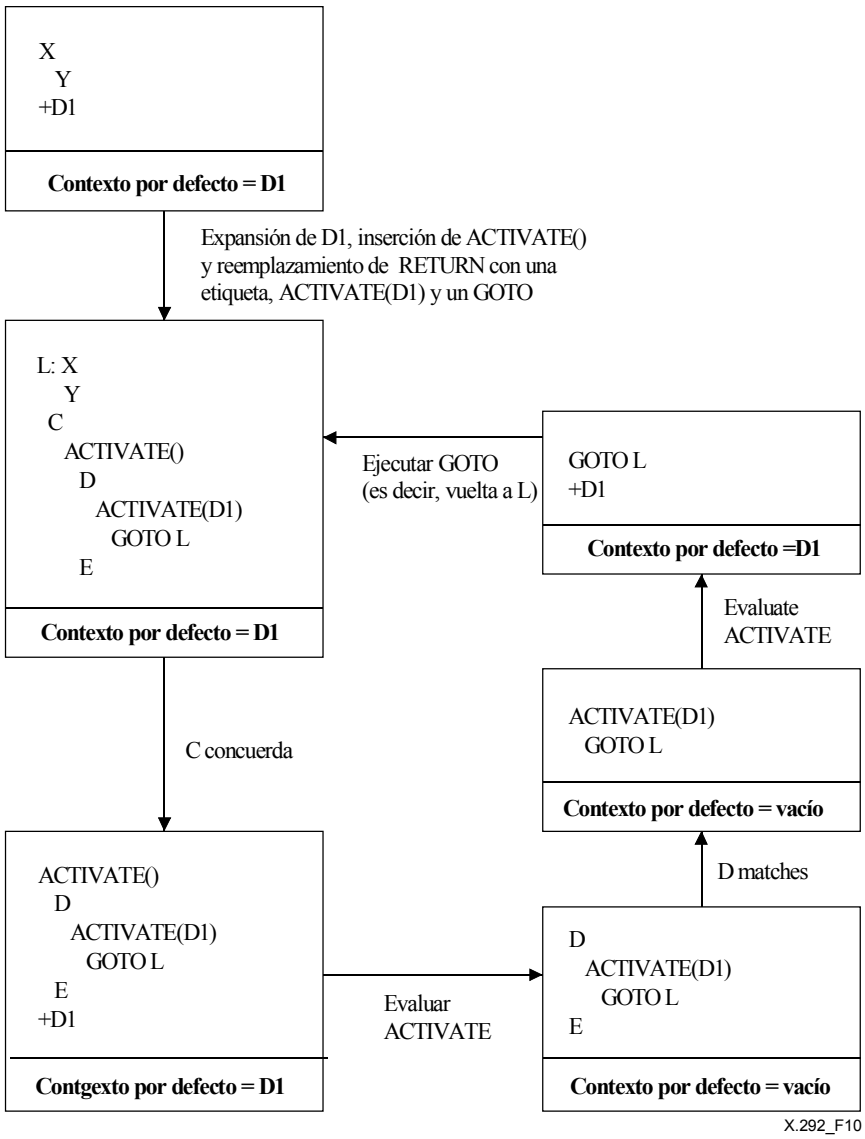


Figura 10/X.292 – Posible progresión de la evaluación de ejemplo en caso de prueba X_Def3

16 Continuación de página

16.1 Continuación de página en los cuadros de TTCN

Cuando, por su longitud, algún cuadro TTCN no quepa en una sola página, se utilizará el siguiente mecanismo:

a) *A continuación* de la línea del cuadro en donde se produzca la división se imprimirán las palabras "Continúa en la página siguiente".

b) *Antes* de la continuación del cuadro en la página siguiente se imprimirán las palabras "Continuación de la página anterior".

Los cuadros se pueden dividir en cualquier lugar, esto es, en sus secciones de encabezamiento, de cuerpo o de pie. En todos los casos, los títulos de las secciones (por ejemplo, encabezamiento de las columnas) deberán repetirse en la página siguiente. No es necesario efectuar la repetición del encabezamiento completo.

EJEMPLO 110 – Cuadro de parámetros de una serie de pruebas continuada:

Declaraciones de parámetros de serie de pruebas			
Nombre del parámetro	**Tipo**	**Ref. de PICS/PIXIT**	**Comentarios**
PAR1	INTEGER	Cuestión aa de la PICS	
PAR2	BOOLEAN	Cuestión bb de la PICS	
PAR3	IA5String	Cuestión cc de la PIXIT	

Continúa en la página siguiente página n

Continuación de la página anterior página n + 1

Declaraciones de parámetros de serie de pruebas			
Nombre del parámetro	**Tipo**	**Ref. de PICS/PIXIT**	**Comentarios**
PAR4	BOOLEAN	Cuestión dd de la PICS	
PAR5	HEXSTRING	Cuestión ee de la PICS	

16.2 Continuación de página en los cuadros de comportamiento dinámico

Cuando sea necesario continuar un cuadro de comportamiento dinámico, se podrá usar cualquiera de los dos mecanismos siguientes:

a) modularización,

   cuando se especifique alguna parte del comportamiento del árbol como un paso de prueba de biblioteca (no local), modularizando, en consecuencia, el árbol y reduciendo la magnitud del comportamiento, del formulario de que se trate, de manera que quepa en una sola página; o

b) mecanismo de continuación de página,

   cuando, en el caso de un cuadro de comportamiento dinámico y para facilitar la alineación de los niveles de sangrado, tenga que presentarse la siguiente información adicional:

   1) el nivel de sangrado, impreso entre corchetes antes de la frase "Continúa en la página siguiente", del último enunciado la TTCN anterior a la división de la página;

   2) en la página siguiente, y tras la frase "Continuación de la página anterior", el nivel de sangrado, impreso entre corchetes, del primer enunciado la TTCN del cuadro que sigue.

En el caso de casos de prueba largos, puede ser necesario sangrar a un nivel diferente del establecido. En tales casos, el nivel de sangrado enunciado, encerrado entre corchetes, deberá alinearse con el sangrado elegido de la primera línea de enunciado en el cuadro siguiente. Para facilitar ulteriormente la alineación de los niveles de sangrado, pueden, asimismo, facilitarse indicaciones adicionales de esos niveles de sangrado.

Anexo A

Sintaxis y semántica estática de la TTCN

(Este anexo es parte integrante de la Recomendación)

A.1 Introducción

En este anexo se definen la sintaxis y la semántica estática de la TTCN. Hay dos formas de TTCN, una forma gráfica (TTCN.GR) y una forma procesable por máquina (TTCN.MP). Para un usuario humano, la forma gráfica de la TTCN, TTCN.GR, tiene la ventaja de una interpretación visual fácilmente comprensible. Sin embargo, la TTCN.GR no se presta a un procesamiento por máquina. La TTCN.MP aborda este problema y cumple los siguientes fines:

a) proporcionar una sintaxis formal para la TTCN en BNF;

b) actuar como una sintaxis de transferencia;

c) facilitar una derivación automatizada de ETS a partir de ATS;

d) otros procesamientos por máquina.

NOTA – La derivación automatizada de ETS está fuera del ámbito de esta Recomendación.

En este anexo se definen asimismo las semánticas estáticas de TTCN.GR y TTCN.MP.

A.2 Convenciones de la descripción de sintaxis

A.2.1 Metanotación sintáctica

En el cuadro A.1 se define la metanotación utilizada para especificar la forma ampliada de gramática BNF para TTCN (en adelante BNF).

Cuadro A.1/X.292 – Metanotación sintáctica de TTCN.MP

::=	se define como
abc xyz	abc seguido de xyz
|	alternativa
[abc]	0 ó 1 instancia de abc
{abc}	0 o más instancias de abc
{abc}+	1 o más instancias de abc
(...)	agrupación textual
abc	el símbolo no terminal abc
abc	un símbolo terminal abc
"abc"	un símbolo terminal abc

En la metanotación, la concatenación vincula más fuertemente que el operador alternativo. Por lo tanto "abc def | ghi jkl" es equivalente a "(abc def) | (ghi jkl)".

A.2.2 Definiciones de sintaxis de TTCN.MP

Los cuadros completos definidos en TTCN.GR se representan en TTCN.MP mediante producciones del siguiente tipo:

$Begin_KEYWORD .... $End_KEYWORD

EJEMPLO A.1 – TS_PARdcls ::= **$Begin_TS_PARdcls** {TS_PARdcl}+ **$End_TS_PARdcls**

Normalmente, estas producciones contienen al menos un campo obligatorio.

Los conjuntos de líneas de un cuadro así como las líneas individuales (esto es, conjuntos de campos en un cuadro) se representan mediante producciones del tipo:

$KEYWORD .... $End_KEYWORD

Begin no aparece en la palabra clave de apertura.

EJEMPLO A.2 – TS_PARdcl ::= **$TS_PARdcl** TS_PARid TS_PARtype PICS_PIXIT [Comment]
$End_TS_PARdcl

Los campos individuales de una línea se representan por:

$KEYWORD ....

No hay palabra clave de cierre.

EJEMPLO A.3 – TS_ParIid ::= **$TS_ParId** TS_ParIdentifier

EJEMPLO A.4 – TS_ParIidentifier ::= Identifier

Los conjuntos de cuadros, hasta la serie de pruebas inclusive, se representan mediante producciones del tipo:

$KEYWORD .... $End_KEYWORD

EJEMPLO A.5 – ASP_TypeDefs ::= **$ASP_TypeDefs** [TTCN_ASP_TypeDefs] [ASN1_ASP_TypeDefs] **$End_ASP_TypeDefs**

Las demás producciones que definen símbolos no terminales no tienen palabras clave al principio ni al final de la expresión del lado derecho.

EJEMPLO A.6 – TimerIdentifier ::= Identifier

Cuando se realiza el análisis gramatical de TTCN.MP, cualquier símbolo no permitido dentro de un identificador puede indicar el fin de un identificador. En aquellos casos en que sea necesario insertar uno o más separadores al final de un identificador para separarlo de otro identificador o palabra clave (por ejemplo, cuando un identificador va seguido de una palabra clave como **BY** u **OR**); los separadores son caracteres Space (espacio), Tab (tabulador) y Carriage Return (cambio de línea).

A.3 Las producciones de sintaxis TTCN.MP en BNF

A.3.1 Especificación de TTCN

1 TTCN_Specification ::= TTCN_Module | Suite

A.3.2 Módulo de TTCN

2 TTCN_Module ::= **$TTCN_Module** TTCN_ModuleId TTCN_ModuleOverviewPart
[TTCN_ModuleImportPart] [DeclarationsPart] [ConstraintsPart] [DynamicPart] **$End_TTCN_Module**
3 TTCN_ModuleId ::=**$TTCN_ModuleId** TTCN_ModuleIdentifier
4 TTCN_ModuleIdentifier ::= Identifier

A.3.2.1 Parte visión general del módulo de TTCN

5 TTCN_ModuleOverviewPart ::= **$TTCN_ModuleOverviewPart** TTCN_ModuleExports
[TTCN_ModuleStructure] [TestCaseIndex] [TestStepIndex] [DefaultIndex]
$End_TTCN_ModuleOverviewPart

A.3.2.1.1 Exportaciones del módulo de TTCN

6 TTCN_ModuleExports ::= **$Begin_TTCN_ModuleExports** TTCN_ModuleId [TTCN_ModuleRef]
[TTCN_ModuleObjective] [StandardsRef] [PICSref] [PIXITref] [TestMethods] [Comment] ExportedObjects [Comment]
$End_TTCN_ModuleExports
7 TTCN_ModuleRef ::= **$TTCN_ModuleRef** BoundedFreeText
8 TTCN_ModuleObjective ::= **$TTCN_ModuleObjective** BoundedFreeText
9 ExportedObjects ::= **$ExportedObjects** {ExportedObject} **$End_ExportedObjects**
10 ExportedObject ::= **$ExportedObject** ObjectId ObjectType [SourceInfo] [Comment] **$End_ExportedObject**
11 ObjectId ::= **$ObjectId** ObjectIdentifier
12 ObjectIdentifier ::= Identifier | ObjectTypeReference
13 ObjectTypeReference ::= Identifier "[" Identifier "]"
/* SEMÁNTICA ESTÁTICA – El primer identificador es un NamedNumber o una Enumeration y el identificador contenido entre corchetes nombre del tipo correspondiente. */
14 ObjectType ::= **$ObjectType** TTCN_ObjectType
15 TTCN_ObjectType ::= **SimpleType_Object** | **StructType_Object** | **ASN1_Type_Object** | **TS_Op_Object** | **TS_Proc_Object** | **TS_Par_Object** | **SelectExpr_Object** | **TS_Const_Object** | **TS_Var_Object** | **TC_Var_Object** | **PCO_Type_Object** | **PCO_Object** | **CP_Object** | **Timer_Object** | **TComp_Object** | **TCompConfig_Object** | **TTCN_ASP_Type_Object** | **ASN1_ASP_Type_Object** | **TTCN_PDU_Type_Object** | **ASN1_PDU_Type_Object** | **TTCN_CM_Type_Object** | **ASN1_CM_Type_Object** | **EncodingRule_Object** | **EncodingVariation_Object** | **InvalidFieldEncoding_Object** | **Alias_Object** | **StructTypeConstraint_Object** | **ASN1_TypeConstraint_Object** | **TTCN_ASP_Constraint_Object** | **ASN1_ASP_Constraint_Object** | **TTCN_PDU_Constraint_Object** | **ASN1_PDU_Constraint_Object** | **TTCN_CM_Constraint_Object** | **ASN1_CM_Constraint_Object** | **TestCase_Object** | **TestStep_Object** | **Default_Object** | **NamedNumber_Object** | **Enumeration_Object**
16 SourceInfo ::= **$SourceInfo** (SourceIdentifier | ObjectDirective)
/* SEMÁNTICA ESTÁTICA – SourceIdentifier es el nombre del objeto fuente original. */
17 SourceIdentifier ::= SuiteIdentifier | TTCN_ModuleIdentifier
18 ObjectDirective ::= Omit | **EXTERNAL**

A.3.2.1.2 Estructura del módulo de TTCN

19 TTCN_ModuleStructure ::= **$Begin_TTCN_ModuleStructure** Structure&Objectives [Comment] **$End_TTCN_ModuleStructure**

A.3.2.2 Parte importación del módulo de TTCN

20 TTCN_ModuleImportPart ::= **$TTCN_ModuleImportPart** [ExternalObjects] [ImportDeclarations] **$End_TTCN_ModuleImportPart**

A.3.2.2.1 Objetos externos (external)

21 ExternalObjects ::= **$Begin_ExternalObjects** {ExternalObject}+ [Comment] **$End_ExternalObjects**
22 ExternalObject ::= **$ExternalObject** ExternalObjectId ObjectType [Comment] **$End_ExternalObject**
23 ExternalObjectId ::= **$ExternalObjectId** ExternalObjectIdentifier
24 ExternalObjectIdentifier ::= ObjectIdentifier | TS_OpId&ParList | ConsId&ParList | TestStepId&ParList

A.3.2.2.2 Declaraciones de importación

25 ImportDeclarations ::= **$ImportDeclarations** {ImportsOrGroup}+ **$End_ImportDeclarations**
26 ImportsOrGroup ::= Imports | ImportsGroup
27 ImportsGroup ::= **$ImportsGroup** ImportsGroupId {ImportsOrGroup}+ **$End_ImportsGroup**
28 ImportsGroupId ::= **$ImportsGroupId** ImportsGroupIdentifier
29 Imports ::= **$Begin_Imports** SourceId [ImportsGroupRef] [SourceRef] [StandardsRef] [Comment] ImportedObjects [Comment] **$End_Imports**
30 SourceId ::= **$SourceId** SourceIdentifier
31 ImportsGroupRef ::= **$ImportsGroupRef** ImportsGroupReference
32 ImportsGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {ImportsGroupIdentifier "/"}
33 ImportsGroupIdentifier ::= Identifier
34 SourceRef ::= **$SourceRef** BoundedFreeText
35 ImportedObjects ::= **$ImportedObjects** {ImportedObject}+ **$End_ImportedObjects**
36 ImportedObject ::= **$ImportedObject** ObjectId ObjectType [SourceInfo] [Comment] **$End_ImportedObject**

A.3.3 Serie de pruebas

37 Suite ::= **$Suite** SuiteId SuiteOverviewPart [ImportPart] DeclarationsPart ConstraintsPart DynamicPart **$End_Suite**
/* SEMÁNTICA ESTÁTICA – SuiteId será el mismo que el SuiteId declarado en el cuadro de TestSuiteStructure (Estructura de serie). */
38 SuiteId ::= **$SuiteId** SuiteIdentifier
39 SuiteIdentifier ::= Identifier

A.3.3.1 La visión general de la serie de pruebas

40 SuiteOverviewPart::= **$SuiteOverviewPart** [TestSuiteIndex] SuiteStructure TestCaseIndex [TestStepIndex] [DefaultIndex] [TestSuiteExports] **$End_SuiteOverviewPart**

A.3.3.2 Índice de serie de pruebas

41 TestSuiteIndex ::= **$Begin_TestSuiteIndex** {ObjectInfo} [Comment] **$End_TestSuiteIndex**

A.3.3.2.1 The Imported Object Info

42 ObjectInfo ::= **$ObjectInfo** ObjectId ObjectType SourceId OrigObjectId [PageNum] [Comment] **$End_ObjectInfo**
43 PageNum ::= **$PageNum** PageNumber
44 PageNumber ::= Number
45 OrigObjectId ::= **$OrigObjectId** ObjectIdentifier

A.3.3.3 Estructura de la serie de pruebas

46 SuiteStructure ::= **$Begin_SuiteStructure** SuiteId StandardsRef PICSref PIXITref TestMethods [Comment] Structure&Objectives [Comment] **$End_SuiteStructure**
47 StandardsRef ::= **$StandardsRef** BoundedFreeText
48 PICSref ::= **$PICSref** BoundedFreeText
49 PIXITref ::= **$PIXITref** BoundedFreeText
50 TestMethods ::= **$TestMethods** BoundedFreeText
51 Comment ::= **$Comment** [BoundedFreeText]
52 Structure&Objectives ::= **$Structure&Objectives** {Structure&Objective} **$End_Structure&Objectives**
53 Structure&Objective ::= **$Structure&Objective** TestGroupRef SelExprId Objective **$End_Structure&Objective**
54 SelExprId ::= **$SelectExprId** [SelectExprIdentifier]

A.3.3.4 Índice de casos de prueba

55 TestCaseIndex ::= **$Begin_TestCaseIndex** {[CollComment] CaseIndex}+ [Comment] **$End_TestCaseIndex**
56 CollComment ::= **$CollComment** [BoundedFreeText]
57 CaseIndex ::= **$CaseIndex** TestGroupRef TestCaseId SelExprId Description **$End_CaseIndex**
/* SEMÁNTICA ESTÁTICA – La lista de los casos de prueba seguirá el orden de la parte dinámica. */
/* SEMÁNTICA ESTÁTICA – Se proporcionará una referencia a grupo de prueba explícita para cada caso de prueba que pertenezca a un grupo de prueba. */
58 Description ::= **$Description** BoundedFreeText

A.3.3.5 Índice de pasos de prueba

59 TestStepIndex ::= **$Begin_TestStepIndex** {[CollComment] StepIndex} [Comment] **$End_TestStepIndex**
60 StepIndex ::= **$StepIndex** TestStepRef TestStepId Description **$End_StepIndex**
/* SEMÁNTICA ESTÁTICA – TestStepId no incluirá una lista de parámetros formales. */
/* SEMÁNTICA ESTÁTICA – La relación de los pasos de prueba deberá figurar en el orden en que aparecen en la parte dinámica. */
/* SEMÁNTICA ESTÁTICA – Se proporcionará una referencia a paso de prueba explícita para cada paso de prueba que pertenece a un grupo de pasos. */

A.3.3.6 Índice de valores por defecto

61 DefaultIndex ::= **$Begin_DefaultIndex** {[CollComment] DefIndex} [Comment] **$End_DefaultIndex**
62 DefIndex ::= **$DefIndex** DefaultRef DefaultId Description **$End_DefIndex**
/* SEMÁNTICA ESTÁTICA – DefaultId no incluirá una lista de parámetros formales. */
/* SEMÁNTICA ESTÁTICA – La relación de los valores por defecto deberá figurar en el orden en que aparecen en la parte dinámica. */
/* SEMÁNTICA ESTÁTICA – Se proporcionará una referencia a grupo valores por defecto explícita para cada valor por defecto que pertenezca a un grupo valores por defecto. */

A.3.3.7 Exportaciones de series de pruebas

63 TestSuiteExports::= **$Begin_TestSuiteExports** ExportedObjects [Comment] **$End_TestSuiteExports**

A.3.3.8 La parte importación

64 ImportPart ::= **$ImportPart** ImportDeclarations **$End_ImportPart**

A.3.3.9 La parte declaraciones

65 DeclarationsPart ::= **$DeclarationsPart** Definitions Parameterization&Selection Declarations ComplexDefinitions **$End_DeclarationsPart**

A.3.3.10 Definiciones

A.3.3.10.1 Generalidades

66 Definitions ::= [TS_TypeDefs] [EncodingDefs] [TS_OpDefs] [TS_ProcDefs]

A.3.3.10.2 Definiciones de tipo de serie de pruebas

67 TS_TypeDefs ::= **$TS_TypeDefs** {SimpleTypeDefsOrGroup} [StructTypeDefs] [ASN1_TypeDefs] {ASN1_TypeRefsOrGroup} **$End_TS_TypeDefs**

A.3.3.10.3 Definiciones de tipo simple

68 SimpleTypeDefsOrGroup ::= SimpleTypeDefs | SimpleTypeGroup
69 SimpleTypeGroup ::= **$SimpleTypeGroup** SimpleTypeGroupId {SimpleTypeDefsOrGroup}+ **$End_SimpleTypeGroup**
70 SimpleTypeGroupId ::= **$SimpleTypeGroupId** SimpleTypeGroupIdentifier
71 SimpleTypeDefs ::= **$Begin_SimpleTypeDefs** [SimpleTypeGroupRef] {[CollComment] SimpleTypeDef}+ [Comment] **$End_SimpleTypeDefs**
72 SimpleTypeGroupRef ::= **$SimpleTypeGroupRef** SimpleTypeGroupReference
73 SimpleTypeGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {SimpleTypeGroupIdentifier "/"}
74 SimpleTypeGroupIdentifier ::= Identifier
75 SimpleTypeDef ::= **$SimpleTypeDef** SimpleTypeId SimpleTypeDefinition [PDU_FieldEncoding] [Comment] **$End_SimpleTypeDef**
76 SimpleTypeId ::= **$SimpleTypeId** SimpleTypeIdentifier
77 SimpleTypeIdentifier ::= Identifier
78 SimpleTypeDefinition ::= **$SimpleTypeDefinition** Type&Restriction
/* SEMÁNTICA ESTÁTICA – No habrá referencias recursivas (ni directa ni indirectamente) en Type&Restriction. */
79 Type&Restriction ::= Type [Restriction]
/* SEMÁNTICA ESTÁTICA – Type será un PredefinedType o un SimpleType. */
80 Restriction ::= LengthRestriction | IntegerRange | SimpleValueList

/* SEMÁNTICA ESTÁTICA – El conjunto de valores definidos por Restriction será un subconjunto verdadero de los valores del tipo de base. */

81 LengthRestriction ::= SingleLength | RangeLength

/* SEMÁNTICA ESTÁTICA – LengthRestriction será proporcionada solamente cuando el tipo de base sea un tipo cadena (es decir, BITSTRING, HEXSTRING, OCTETSTRING o CharacterString) o se derive de un tipo cadena. */

82 SingleLength ::= "[" ConstantExpression "]"

83 RangeLength ::= "[" LowerBound To UpperBound "]"

/* SEMÁNTICA ESTÁTICA – LowerBound será un número no negativo. */

/* SEMÁNTICA ESTÁTICA – LowerBound será menor que UpperBound. */

84 IntegerRange ::= "(" LowerBound To UpperBound ")"

/* SEMÁNTICA ESTÁTICA – LowerBound será menor que UpperBound. */

85 LowerBound ::= ConstantExpression | Minus **INFINITY**

86 UpperBound ::= ConstantExpression | **INFINITY**

87 To ::= **TO** | "**..**"

88 SimpleValueList ::= "(" ConstantExpression {Comma ConstantExpression } ")"

/* SEMÁNTICA ESTÁTICA – Los ConstantExpression deberán utilizarse como tipo de base y deberán ser un subconjunto verdadero de los valores definidos por el tipo de base. */

A.3.3.10.4 Definiciones de tipo estructurado

89 StructTypeDefs ::= **$StructTypeDefs** {StructTypeDefOrGroup}+ **$End_StructTypeDefs**

90 StructTypeDefOrGroup ::= StructTypeDef | StructTypeGroup

91 StructTypeGroup ::= **$StructTypeGroup** StructTypeGroupId {StructTypeDefOrGroup}+ **$End_StructTypeGroup**

92 StructTypeGroupId ::= **$StructTypeGroupId** StructTypeGroupIdentifier

93 StructTypeDef ::= **$Begin_StructTypeDef** StructId [StructTypeGroupRef] [EncVariationId] [Comment] ElemDcls [Comment] **$End_StructTypeDef**

94 StructId ::= **$StructId** StructId&FullId

95 StructId&FullId ::= StructIdentifier [FullIdentifier]

96 FullIdentifier ::= "(" BoundedFreeText ")"

/* SEMÁNTICA ESTÁTICA – Algunos objetos de TTCN permiten utilizar nombres abreviados según se indica en la norma de protocolo apropiada. Si se utiliza una abreviatura, FullIdentifier deberá aparecer en la declaración del objeto. */

97 StructIdentifier ::= Identifier

98 StructTypeGroupRef ::= **$StructTypeGroupRef** StructTypeGroupReference

99 StructTypeGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {StructTypeGroupIdentifier "/"}

100 StructTypeGroupIdentifier ::= Identifier

101 ElemDcls ::= **$ElemDcls** {ElemDcl}+ **$End_ElemDcls**

102 ElemDcl ::= **$ElemDcl** ElemId ElemType [PDU_FieldEncoding] [Comment] **$End_ElemDcl**

103 ElemId ::= $ElemId ElemId&FullId

104 ElemId&FullId ::= ElemIdentifier [FullIdentifier]

105 ElemIdentifier ::= Identifier

106 ElemType ::= **$ElemType** Type&Attributes

/* SEMÁNTICA ESTÁTICA – En Type&Attributes no habrá referencias recursivas (ni directas ni indirectas). */

/* SEMÁNTICA ESTÁTICA – Un tipo de elemento estructurado será un PredefinedType, un TS_TypeIdentifier, un PDU_Identifier o una PDU. */

A.3.3.10.5 Definiciones de tipo ASN.1

107 ASN1_TypeDefs ::= **$ASN1_TypeDefs** {ASN1_TypeDefOrGroup}+ **$End_ASN1_TypeDefs**

108 ASN1_TypeDefOrGroup ::= ASN1_TypeDef | ASN1_TypeGroup

109 ASN1_TypeGroup ::= **$ASN1_TypeGroup** ASN1_TypeGroupId {ASN1_TypeDefOrGroup}+ **$End_ASN1_TypeGroup**

110 ASN1_TypeGroupId ::= **$ASN1_TypeGroupId** ASN1_TypeGroupIdentifier

111 ASN1_TypeDef ::= **$Begin_ASN1_TypeDef** ASN1_TypeId [ASN1_TypeGroupRef] [EncVariationId] [Comment] ASN1_TypeDefinition [Comment] **$End_ASN1_TypeDef**

112 ASN1_TypeId ::= **$ASN1_TypeId** ASN1_TypeId&FullId

113 ASN1_TypeId&FullId ::= ASN1_TypeIdentifier [FullIdentifier]

114 ASN1_TypeIdentifier ::= Identifier

115 ASN1_TypeGroupRef ::= **$ASN1_TypeGroupRef** ASN1_TypeGroupReference

116 ASN1_TypeGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {ASN1_TypeGroupIdentifier "/"}

117 ASN1_TypeGroupIdentifier ::= Identifier

118 ASN1_TypeDefinition ::= **$ASN1_TypeDefinition** ASN1_Type&LocalTypes **$End_ASN1_TypeDefinition**

119 ASN1_Type&LocalTypes ::= ASN1_Type {ASN1_LocalType}

/* SEMÁNTICA ESTÁTICA – Los tipos a los que se hace referencia a partir de la definición de ASN1_Type serán definidos en otros cuadros de definiciones de tipo en ASN.1, definidos por referencia en los cuadros de referencia a tipo en ASN.1, o definidos localmente (es decir, ASN1_LocalTypes) en el mismo cuadro, inmediatamente después de la primera definición de tipo. */

/* SEMÁNTICA ESTÁTICA – Los tipos ASN.1_Local no se utilizarán en otras partes de la serie de pruebas. */

120 ASN1_Type ::= Type

/* REFERENCIA – Donde Type es un no terminal definido en la Rec. UIT-T X.680:
Type ::= BuiltinType | ReferencedType | ConstrainedType
A los fines de la TTCN, la producción de la Rec. UIT-T X.680 que establece:
SubtypeElements ::= SingleValue | ConstrainedSubtype | ValueRange | PermittedAlphabet | SizeConstraint |

TypeConstraint | InnerTypeConstraint

se redefine así:

SubtypeElements ::= SingleValue | ConstrainedSubtype | ValueRange | PermittedAlphabet | SizeConstraint | TypeConstraint | InnerTypeConstraint | ASN1_Encoding

Esto significa que puede aplicarse la codificación ASN1 en cualquier situación en la que se puede aplicar una constricción de tipo (TypeConstraint): al conjunto de un ASN1_Type o a cualquier tipo ASN.1 dentro del ASN1_Type o a un tipo SET OF o SEQUENCE OF [colocando la codificación ASN1 entre paréntesis inmediatamente después de la palabra clave SET o SEQUENCE – a diferencia de una constricción de tamaño (SizeConstraint) en tal posición, los paréntesis son necesarios porque no hay argumento de compatibilidad hacia atrás para permitir su omisión].

A los fines de la TTCN, las producciones siguientes en la Rec. UIT-T X.680:

```
BuiltinType ::=
        BitStringType |
        BooleanType |
        CharacterStringType |
        ChoiceType |
        EmbeddedPDUType |
        EnumeratedType |
        ExternalType |
        InstanceOfType |
        IntegerType |
        NullType |
        ObjectClassFieldType |
        ObjectIdentifierType |
        OctetStringType |
        RealType |
        SequenceType |
        SequenceOfType |
        SetType |
        SetOfType |
        TaggedType

ReferencedType ::=
        DefinedType |
        UsefulType |
        SelectionType |
        TypeFromObject |
        ValueSetFromObjects

DefinedType ::=
        Externaltypereference |
        typereference |
        ParameterizedType |
        ParameterizedValueSetType

Elements ::=
        SubtypeElements |
        ObjectSetElements |
        "(" ElementSetSpec ")"
```

se redefinen así:

```
BuiltinType ::=
        BitStringType |
        BooleanType |
        CharacterStringType |
        ChoiceType |
        EmbeddedPDUType |
        EnumeratedType |
        ExternalType |
        IntegerType |
        NullType |
        ObjectIdentifierType |
        OctetStringType |
        RealType |
        SequenceType |
        SequenceOfType |
        SetType |
        SetOfType |
        TaggedType
```

ReferencedType ::=
DefinedType |
UsefulType |
SelectionType

DefinedType ::=
Externaltypereference |
typereference

Elements ::=
SubtypeElements |
"(" ElementSetSpec ")" */

/* SEMÁNTICA ESTÁTICA – Cada referencia a tipo terminal utilizada dentro de la producción Type, será una de las siguientes: ASN1_LocalType typereference, TS_TypeIdentifier o PDU_Identifier. */
/* SEMÁNTICA ESTÁTICA – Las definiciones de tipo ASN.1 utilizadas en TTCN no utilizarán las referencias de tipo externo definidas en la Rec. UIT-T X.680. */

121 ASN1_LocalType ::= Typeassignment
/* REFERENCIA – Donde Typeassignment es un no terminal definido en la Rec. UIT-T X.680. */
/* SEMÁNTICA ESTÁTICA – Las definiciones de tipo ASN.1 utilizadas en TTCN no utilizarán las referencias de tipo externo definidas en la Rec. UIT-T X.680. */

A.3.3.10.6 Definiciones de tipo ASN.1 por referencia

122 ASN1_TypeRefsOrGroup ::= ASN1_TypeRefs | ASN1_TypeRefsGroup
123 ASN1_TypeRefsGroup ::= **$ASN1_TypeRefsGroup** ASN1_TypeRefsGroupId {ASN1_TypeRefsOrGroup}+ **$End_ASN1_TypeRefsGroup**
124 ASN1_TypeRefsGroupId ::= **$ASN1_TypeRefsGroupId** ASN1_TypeGroupIdentifier
125 ASN1_TypeRefs ::= **$Begin_ASN1_TypeRefs** [ASN1_TypeRefsGroupRef] {[CollComment] ASN1_TypeRef}+ [Comment] **$End_ASN1_TypeRefs**
126 ASN1_TypeRefsGroupRef ::= **$ASN1_TypeRefsGroupRef** ASN1_TypeGroupReference
127 ASN1_TypeRef ::= **$ASN1_TypeRef** ASN1_TypeId ASN1_TypeReference ASN1_ModuleId [EncVariationId] [Comment] **$End_ASN1_TypeRef**
/* SEMÁNTICA ESTÁTICA – No se especificará ASN1_TypeId en un FullIdentifier. */
128 ASN1_TypeReference ::= **$ASN1_TypeReference** TypeReference
129 TypeReference ::= typereference
/* REFERENCIA – Donde typereference es un no terminal definido en la Rec. UIT-T X.680. */
/* SEMÁNTICA ESTÁTICA – Si la definición de tipo ASN.1 tiene una referencia a otro tipo en el mismo módulo ASN.1, el tipo referenciado es implícitamente importado (del mismo modo que para el módulo TTCN). */
130 ASN1_ModuleId ::= **$ASN1_ModuleId** ASN1_ModuleIdentifier
131 ASN1_ModuleIdentifier ::= ModuleIdentifier
/* REFERENCIA – Donde ModuleIdentifier es un no terminal definido en la Rec. UIT-T X.680. */
/* SEMÁNTICA ESTÁTICA – ModuleIdentifier será único dentro del dominio de interés. */

A.3.3.10.7 Definiciones de operaciones series de pruebas

132 TS_OpDefs ::= **$TS_OpDefs** {TS_OpDefOrGroup}+ **$End_TS_OpDefs**
133 TS_OpDefOrGroup ::= TS_OpDef | TS_OpDefGroup
134 TS_OpDefGroup ::= **$TS_OpDefGroup** TS_OpDefGroupId {TS_OpDefOrGroup}+ **$End_TS_OpDefGroup**
135 TS_OpDefGroupId ::= **$TS_OpDefGroupId** TS_OpDefGroupIdentifier
136 TS_OpDefGroupIdentifier ::= Identifier
137 TS_OpDef ::= **$Begin_TS_OpDef** TS_OpId [TS_OpGroupRef] TS_OpResult [Comment] TS_OpDescription [Comment] **$End_TS_OpDef**
138 TS_OpId ::= **$TS_OpId** TS_OpId&ParList
139 TS_OpId&ParList ::= TS_OpIdentifier [FormalParList]
/* SEMÁNTICA ESTÁTICA – Un parámetro formal de operación serie de pruebas Type será un PredefinedType, un TS_TypeIdentifier, un PDU_Identifier o un ASP_Identifier, o el metatipo **PDU***/
140 TS_OpIdentifier ::= Identifier
141 TS_OpGroupRef ::= **$TS_OpGroupRef** TS_OpGroupReference
142 TS_OpGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {TS_OpGroupIdentifier "/"}
143 TS_OpGroupIdentifier ::= Identifier
144 TS_OpResult ::= **$TS_OpResult** TypeOrPDU
/* SEMÁNTICA ESTÁTICA – TypeOrPDU será un PredefinedType, un TS_TypeIdentifier, un PDU_Identifier o un ASP_Identifier, o el metatipo **PDU**. */
145 TS_OpDescription ::= **$TS_OpDescription** BoundedFreeText

A.3.3.10.8 Definiciones de procedimientoes de operaciones serie de pruebas

146 TS_ProcDefs ::= **$TS_ProcDefs** {TS_ProcDefOrGroup}+ **$End_TS_ProcDefs**
147 TS_ProcDefOrGroup ::= TS_ProcDef | TS_ProcDefGroup
148 TS_ProcDefGroup ::= **$TS_ProcDefGroup** TS_ProcDefGroupId {TS_ProcDefOrGroup}+ **$End_TS_ProcDefGroup**
149 TS_ProcDefGroupId ::= **$TS_ProcDefGroupId** TS_ProcDefGroupIdentifier

150 TS_ProcDefGroupIdentifier ::= Identifier
150 TS_ProcDefGroupIdentifier ::= Identifier
151 TS_ProcDef ::= **$Begin_TS_ProcDef** TS_ProcId [TS_ProcGroupRef] TS_ProcResult [Comment] TS_ProcDescription [Comment] **$End_TS_ProcDef**
152 TS_ProcId ::= **$TS_ProcId** TS_ProcId&ParList
153 TS_ProcId&ParList ::= TS_ProcIdentifier [FormalParList]
/* SEMÁNTICA ESTÁTICA – Un parámetro formal de operación serie de pruebas de procedimiento Type será un PredefinedType, un TS_TypeIdentifier, un PDU_Identifier o un ASP_Identifier, o el metatipo **PDU**.*/
154 TS_ProcIdentifier ::= Identifier
155 TS_ProcGroupRef ::= **$TS_ProcGroupRef** TS_ProcGroupReference
156 TS_ProcGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {TS_ProcGroupIdentifier "/"}
157 TS_ProcGroupIdentifier ::= Identifier
158 TS_ProcResult ::= **$TS_ProcResult** TypeOrPDU
/* SEMÁNTICA ESTÁTICA – TypeOrPDU será un PredefinedType, un TS_TypeIdentifier, un PDU_Identifier o un ASP_Identifier, o el metatipo **PDU**. */
159 TS_ProcDescription ::= **$TS_ProcDescription** TS_OpProcDef **$End_TS_ProcDescription**
160 TS_OpProcDef ::= [VarBlock] ProcStatement
/* NOTA – Se admiten comentarios dentro de TS_OpProcDef, que empiecen con "/*" y terminen con "*/", pero se supone que estos comentarios son suprimidos antes del análisis gramatical de la sintaxis. Por tanto la BNF no incluye la sintaxis de tales comentrios insertados. */
161 VarBlock ::= **VAR** VarDcls **ENDVAR**
162 VarDcls ::= {VarDcl SemiColon}
163 VarDcl ::= [**STATIC**] VarIdentifiers Colon TypeOrPDU [Colon Value]
164 VarIdentifiers ::= VarIdentifier {Comma VarIdentifier}
165 VarIdentifier ::= Identifier
166 ProcStatement ::= ReturnValueStatement | Assignment | IfStatement | WhileLoop | CaseStatement | ProcBlock
167 ReturnValueStatement ::= **RETURNVALUE** Expression
168 IfStatement ::= **IF** Expression **THEN** {ProcStatement SemiColon}+ [**ELSE** {ProcStatement SemiColon}+] **ENDIF**
169 WhileLoop ::= **WHILE** Expression **DO** {ProcStatement SemiColon}+ **ENDWHILE**
170 CaseStatement ::= **CASE** Expression **OF** {CaseClause SemiColon}+ [**ELSE** {ProcStatement SemiColon}+] **ENDCASE**
171 CaseClause ::= IntegerLabel Colon ProcStatement
172 IntegerLabel ::= Number | TS_ParIdentifier | TS_ConstIdentifier
173 ProcBlock ::= **BEGIN** {ProcStatement SemiColon}+ **END**

A.3.3.11 Parametrización y selección

A.3.3.11.1 Generalidades

174 Parameterization&Selection ::= {TS_ParDclsOrGroup} {SelectExprDefsOrGroup}

A.3.3.11.2 Declaraciones de parámetros de series de pruebas

175 TS_ParDclsOrGroup ::= TS_ParDcls | TS_ParDclsGroup
176 TS_ParDclsGroup ::= **$TS_ParDclsGroup** TS_ParDclsGroupId {TS_ParDclsOrGroup}+ **$End_TS_ParDclsGroup**
177 TS_ParDclsGroupId ::= **$TS_ParDclsGroupId** TS_ParDclsGroupIdentifier
178 TS_ParDclsGroupIdentifier ::= Identifier
179 TS_ParDcls ::= **$Begin_TS_ParDcls** [TS_ParGroupRef] {[CollComment] TS_ParDcl}+ [Comment] **$End_TS_ParDcls**
180 TS_ParGroupRef ::= **$TS_ParGroupRef** TS_ParGroupReference
181 TS_ParGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {TS_ParGroupIdentifier "/"}
182 TS_ParGroupIdentifier ::= Identifier
183 TS_ParDcl ::= **$TS_ParDcl** TS_ParId TS_ParType [TS_ParDefault] PICS_PIXITref [Comment] **$End_TS_ParDcl**
184 TS_ParId ::= **$TS_ParId** TS_ParIdentifier
185 TS_ParIdentifier ::= Identifier
186 TS_ParType ::= **$TS_ParType** TypeOrPDU
/* SEMÁNTICA ESTÁTICA – TypeOrPDU será un PredefinedType, un TS_TypeIdentifier, un PDU_Identifier o un ASP_Identifier, o el metatipo **PDU**. */
187 TS_ParDefault ::= **$TS_ParDefault** [ConstantExpression]
/* SEMÁNTICA OPERACIONAL – ConstantExpression tomará el valor de una elemento de su tipo declarado. */
188 PICS_PIXITref ::= **$PICS_PIXITref** BoundedFreeText

A.3.3.11.3 Definiciones de expresiones de selección de caso de prueba

189 SelectExprDefsOrGroup ::= SelectExprDefs | SelectExprDefsGroup
190 SelectExprDefsGroup ::= **$SelectExprDefsGroup** SelectExprDefsGroupId {SelectExprDefsOrGroup}+ **$End_SelectExprDefsGroup**
191 SelectExprDefsGroupId ::= **$SelectExprDefsGroupId** SelectExprDefsGroupIdentifier
192 SelectExprDefsGroupIdentifier ::= Identifier
193 SelectExprDefs ::= **$Begin_SelectExprDefs** [SelectExprGroupRef] {[CollComment] SelectExprDef}+ [Comment] **$End_SelectExprDefs**
194 SelectExprGroupRef ::= **$SelectExprGroupRef** SelectExprGroupReference
195 SelectExprGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {SelectExprGroupIdentifier "/"}

196 SelectExprGroupIdentifier ::= Identifier
197 SelectExprDef ::= **$SelectExprDef** SelectExprId SelectExpr [Comment] **$End_SelectExprDef**
198 SelectExprId ::= **$SelectExprId** SelectExprIdentifier
199 SelectExprIdentifier ::= Identifier
200 SelectExpr ::= **$SelectExpr** SelectionExpression
201 SelectionExpression ::= ConstantExpression
/* SEMÁNTICA OPERACIONAL – SelectionExpression tomará un valor BOOLEAN específico. */
/* SEMÁNTICA ESTÁTICA – ConstantExpression no se referirá recursivamente (ni directa ni indirectamente) al SelExprIdentifier que está siendo definido por esta Expression. */

A.3.3.12 Declaraciones

A.3.3.12.1 Generalidades

202 Declarations ::= {TS_ConstDclsOrGroup} {TS_ConstRefsOrGroup} {TS_VarDclsOrGroup} {TC_VarDclsOrGroup} {PCO_TypeDclsOrGroup} {PCO_DclsOrGroup} {CP_DclsOrGroup} {TimerDclsOrGroup} {TcompDclsOrGroup} [TCompConfigDcls]

A.3.3.12.2 Declaraciones de constantes en series de pruebas

203 TS_ConstDclsOrGroup ::= TS_ConstDcls | TS_ConstDclsGroup
204 TS_ConstDclsGroup ::= **$TS_ConstDclsGroup** TS_ConstDclsGroupId {TS_ConstDclsOrGroup}+ **$End_TS_ConstDclsGroup**
205 TS_ConstDclsGroupId ::= **$TS_ConstDclsGroupId** TS_ConstDclsGroupIdentifier
206 TS_ConstDclsGroupIdentifier ::= Identifier
207 TS_ConstDcls ::= **$Begin_TS_ConstDcls** [TS_ConstGroupRef] {[CollComment] TS_ConstDcl}+ [Comment] **$End_TS_ConstDcls**
208 TS_ConstGroupRef ::= **$TS_ConstGroupRef** TS_ConstGroupReference
209 TS_ConstGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {TS_ConstGroupIdentifier "/"}
210 TS_ConstGroupIdentifier ::= Identifier
211 TS_ConstDcl ::= **$TS_ConstDcl** TS_ConstId TS_ConstType TS_ConstValue [Comment] **$End_TS_ConstDcl**
212 TS_ConstId ::= **$TS_ConstId** TS_ConstIdentifier
213 TS_ConstIdentifier ::= Identifier
214 TS_ConstType ::= **$TS_ConstType** Type
/* SEMÁNTICA ESTÁTICA – Type no será un tipo estructurado, un tipo de PDU, un tipo de ASP o un tipo de CM expresado en forma tabular. */
215 TS_ConstValue ::= **$TS_ConstValue** ConstantExpression
/* SEMÁNTICA OPERACIONAL – ConstantExpression tomará el valor de un elemento de su tipo declarado. */

A.3.3.12.3 Declaraciones de constantes en series de pruebas por referencia

216 TS_ConstRefsOrGroup ::= TS_ConstRefs | TS_ConstRefsGroup
217 TS_ConstRefsGroup ::= **$TS_ConstRefsGroup** TS_ConstRefsGroupId {TS_ConstRefsOrGroup}+ **$End_TS_ConstRefsGroup**
218 TS_ConstRefsGroupId ::= **$TS_ConstRefsGroupId** TS_ConstRefsGroupIdentifier
219 TS_ConstRefsGroupIdentifier ::= Identifier
220 TS_ConstRefs ::= **$Begin_TS_ConstRefs** [TS_ConstRefsGroupRef] {[CollComment] TS_ConstRef}+ [Comment] **$End_TS_ConstRefs**
221 TS_ConstRefsGroupRef ::= **$TS_ConstRefsGroupRef** TS_ConstGroupReference
222 TS_ConstRef ::= **$TS_ConstRef** TS_ConstId TS_ConstType ASN1_ValueReference ASN1_ModuleId [Comment] **$End_TS_ConstRef**
/* SEMÁNTICA ESTÁTICA – Type en TS_ConstType será un PredefinedType o un ASN1_Type importado por una definición de tipo en ASN.1.1 por referencia al módulo referenciado por ASN1_ModuleId. */
223 ASN1_ValueReference ::= **$ASN1_ValueReference** ValueReference
224 ValueReference ::= valuereference
/* REFERENCIA – valuereference es un no terminal definido en la Rec. UIT-T X.680. */
/* SEMÁNTICA ESTÁTICA – El valor corresponderá a un elemento del tipo de TS_ConstType. */

A.3.3.12.4 Declaraciones de variables de series de pruebas

225 TS_VarDclsOrGroup ::= TS_VarDcls | TS_VarDclsGroup
226 TS_VarDclsGroup ::= **$TS_VarDclsGroup** TS_VarDclsGroupId {TS_VarDclsOrGroup}+ **$End_TS_VarDclsGroup**
227 TS_VarDclsGroupId ::= **$TS_VarDclsGroupId** TS_VarDclsGroupIdentifier
228 TS_VarDclsGroupIdentifier ::= Identifier
229 TS_VarDcls ::= **$Begin_TS_VarDcls** [TS_VarGroupRef] {[CollComment] TS_VarDcl}+ [Comment] **$End_TS_VarDcls**
230 TS_VarGroupRef ::= **$TS_VarGroupRef** TS_VarGroupReference
231 TS_VarGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {TS_VarGroupIdentifier "/"}
232 TS_VarGroupIdentifier ::= Identifier
233 TS_VarDcl ::= **$TS_VarDcl** TS_VarId TS_VarType TS_VarValue [Comment] **$End_TS_VarDcl**
234 TS_VarId ::= **$TS_VarId** TS_VarIdentifier
235 TS_VarIdentifier ::= Identifier
236 TS_VarType ::= **$TS_VarType** TypeOrPDU

/* SEMÁNTICA ESTÁTICA – TypeOrPDU será un PredefinedType, un TS_TypeIdentifier, un PDU_Identifier o ASP_Identifier, o el metatipo **PDU**. */

237 TS_VarValue ::= **$TS_VarValue** [ConstantExpression]

A.3.3.12.5 Declaraciones de variables de caso de prueba

238 TC_VarDclsOrGroup ::= TC_VarDcls | TC_VarDclsGroup
239 TC_VarDclsGroup ::= **$TC_VarDclsGroup** TC_VarDclsGroupId {TC_VarDclsOrGroup}+ **$End_TC_VarDclsGroup**
240 TC_VarDclsGroupId ::= **$TC_VarDclsGroupId** TC_VarDclsGroupIdentifier
241 TC_VarDclsGroupIdentifier ::= Identifier
242 TC_VarDcls ::= **$Begin_TC_VarDcls** [TC_VarGroupRef] {[CollComment] TC_VarDcl}+ [Comment] **$End_TC_VarDcls**
243 TC_VarGroupRef ::= **$TC_VarGroupRef** TC_VarGroupReference
244 TC_VarGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {TC_VarGroupIdentifier "/"}
245 TC_VarGroupIdentifier ::= Identifier
246 TC_VarDcl ::= **$TC_VarDcl** TC_VarId TC_VarType TC_VarValue [Comment] **$End_TC_VarDcl**
247 TC_VarId ::= $**TC_VarId** TC_VarIdentifier
248 TC_VarIdentifier ::= Identifier
249 TC_VarType ::= **$TC_VarType** TypeOrPDU
/* SEMÁNTICA ESTÁTICA – TypeOrPDU será un PredefinedType, un TS_TypeIdentifier, un PDU_Identifier o ASP_Identifier, o el metatipo **PDU**. */
250 TC_VarValue ::= **$TC_VarValue** [ConstantExpression]

A.3.3.12.6 Declaración de tipo de PCO

251 PCO_TypeDclsOrGroup ::= PCO_TypeDcls | PCO_TypeDclsGroup
252 PCO_TypeDclsGroup ::= **$PCO_TypeDclsGroup** PCO_TypeDclsGroupId {PCO_TypeDclsOrGroup}+ **$End_PCO_TypeDclsGroup**
253 PCO_TypeDclsGroupId ::= **$PCO_TypeDclsGroupId** PCO_TypeDclsGroupIdentifier
254 PCO_TypeDclsGroupIdentifier ::= Identifier
255 PCO_TypeDcls ::= **$Begin_PCO_TypeDcls** [PCO_TypeGroupRef] {[CollComment] PCO_TypeDcl}+ [Comment] **$End_PCO_TypeDcls**
256 PCO_TypeGroupRef ::= **$PCO_TypeGroupRef** PCO_GroupReference
257 PCO_TypeDcl ::= **$PCO_TypeDcl** PCO_TypeId RoleOrComment **$End_PCO_TypeDcl**
258 PCO_TypeId ::= **$PCO_TypeId** PCO_TypeIdentifier
259 PCO_TypeIdentifier ::= Identifier
260 RoleOrComment ::= P_Role [Comment] | Comment
/* NOTA – Puesto que cada PCO_Type en un Cuadro de declaraciones de tipo de PCO ha de tener un cometido especificado en la columna de Cometido o en la columna de Comentarios, por lo menos uno de los Comentarios de P_Role ha de estar presente. */

A.3.3.12.7 Declaraciones de PCO

261 PCO_DclsOrGroup ::= PCO_Dcls | PCO_DclsGroup
262 PCO_DclsGroup ::= **$PCO_DclsGroup** PCO_DclsGroupId {PCO_DclsOrGroup}+ **$End_PCO_DclsGroup**
263 PCO_DclsGroupId ::= **$PCO_DclsGroupId** PCO_DclsGroupIdentifier
264 PCO_DclsGroupIdentifier ::= Identifier
265 PCO_Dcls ::= **$Begin_PCO_Dcls** [PCO_GroupRef] {[CollComment] PCO_Dcl}+ [Comment] **$End_PCO_Dcls**
/* SEMÁNTICA ESTÁTICA – Para estar de acuerdo con la Rec. UIT-T X.290, el número de puntos de control y observación (PCO) estará relacionado con el método de prueba utilizado. */
266 PCO_GroupRef ::= **$PCO_GroupRef** PCO_GroupReference
267 PCO_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {PCO_GroupIdentifier "/"}
268 PCO_GroupIdentifier ::= Identifier
269 PCO_Dcl ::= **$PCO_Dcl** PCO_Id PCO_TypeId&MuxValue [P_Role] [Comment] **$End_PCO_Dcl**
270 PCO_Id ::= **$PCO_Id** PCO_Identifier
271 PCO_Identifier ::= Identifier
272 PCO_TypeId&MuxValue ::= **$PCO_TypeId** PCO_TypeIdentifier ["(" MuxValue ")"]
273 MuxValue ::= TS_ParIdentifier
274 P_Role ::= **$PCO_Role** [PCO_Role]
275 PCO_Role ::= **UT** | **LT**

A.3.3.12.8 Declaraciones de CP

276 CP_DclsOrGroup ::= CP_Dcls | CP_DclsGroup
277 CP_DclsGroup ::= **$CP_DclsGroup** CP_DclsGroupId {CP_DclsOrGroup}+ **$End_CP_DclsGroup**
278 CP_DclsGroupId ::= **$CP_DclsGroupId** CP_DclsGroupIdentifier
279 CP_DclsGroupIdentifier ::= Identifier
280 CP_Dcls ::= **$Begin_CP_Dcls** [CP_GroupRef] {[CollComment] CP_Dcl}+ [Comment] **$End_CP_Dcls**
281 CP_GroupRef ::= **$CP_GroupRef** CP_GroupReference
282 CP_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {CP_GroupIdentifier "/"}
283 CP_GroupIdentifier ::= Identifier
284 CP_Dcl ::= **$CP_Dcl** CP_Id [Comment] **$End_CP_Dcl**

285 CP_Id ::= **$CP_Id** CP_Identifier
286 CP_Identifier ::= Identifier

A.3.3.12.9 Declaraciones de temporizadores

287 TimerDclsOrGroup ::= TimerDcls | TimerDclsGroup
288 TimerDclsGroup ::= **$TimerDclsGroup** TimerDclsGroupId {TimerDclsOrGroup}+ **$End_TimerDclsGroup**
289 TimerDclsGroupId ::= **$TimerDclsGroupId** TimerDclsGroupIdentifier
290 TimerDclsGroupIdentifier ::= Identifier
291 TimerDcls ::= **$Begin_TimerDcls** [TimerGroupRef] {[CollComment] TimerDcl}+ [Comment] **$End_TimerDcls**
292 TimerGroupRef ::= **$TimerGroupRef** TimerGroupReference
293 TimerGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {TimerGroupIdentifier "/"}
294 TimerGroupIdentifier ::= Identifier
295 TimerDcl ::= **$TimerDcl** TimerId Duration Unit [Comment] **$End_TimerDcl**
296 TimerId ::= **$TimerId** TimerIdentifier
297 TimerIdentifier ::= Identifier
298 Duration ::= **$Duration** [ConstantExpression]
/* SEMÁNTICA OPERACIONAL – ConstantExpression tomará un valor INTEGER positivo distinto de cero. */
299 Unit ::= **$Unit** TimeUnit
300 TimeUnit ::= **ps** | **ns** | **us** | **ms** | **s** | **min**
/* SEMÁNTICA ESTÁTICA – Si un temporizador está derivado del PICS/PIXIT la declaración de temporizador especificará las mismas unidades que la entrada PICS/PIXIT. */

A.3.3.12.10 Declaraciones de componentes de prueba

301 TCompDclsOrGroup ::= TCompDcls | TCompDclsGroup
302 TCompDclsGroup ::= **$TCompDclsGroup** TCompDclsGroupId {TCompDclsOrGroup}+ **$End_TCompDclsGroup**
303 TCompDclsGroupId ::= **$TCompDclsGroupId** TCompDclsGroupIdentifier
304 TCompDclsGroupIdentifier ::= Identifier
305 TCompDcls ::= **$Begin_TCompDcls** [TCompGroupRef] {[CollComment] TCompDcl}+ [Comment] **$End_TCompDcls**
306 TCompGroupRef ::= **$TCompGroupRef** TCompGroupReference
307 TCompGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {TCompGroupIdentifier "/"}
308 TCompGroupIdentifier ::= Identifier
309 TCompDcl ::= **$TCompDcl** TCompId C_Role NumOf_PCOs NumOf_CPs [Comment] **$End_TCompDcl**
310 TCompId ::= **$TCompId** TCompIdentifier
311 TCompIdentifier::= Identifier
312 C_Role ::= **$TCompRole** TCompRole
313 TCompRole ::= **MTC** | **PTC**
314 NumOf_PCOs ::= **$NumOf_PCOs** Num_PCOs
315 Num_PCOs ::= Number
316 NumOf_CPs ::= **$NumOf_CPs** Num_CPs
317 Num_CPs ::= Number

A.3.3.12.11 Declaraciones de configuraciones de componentes de prueba

318 TCompConfigDcls ::= **$TCompConfigDcls** {TCompConfigDclOrGroup}+ **$End_TCompConfigDcls**
319 TCompConfigDclOrGroup ::= TCompConfigDcl | TCompConfigDclGroup
320 TCompConfigDclGroup ::= **$TCompConfigDclGroup** TCompConfigDclGroupId {TCompConfigDclOrGroup}+ **$End_TCompConfigDclGroup**
321 TCompConfigDclGroupId ::= **$TCompConfigDclGroupId** TCompConfigDclGroupIdentifier
322 TCompConfigDclGroupIdentifier ::= Identifier
323 TCompConfigDcl ::= **$Begin_TCompConfigDcl** TCompConfigId [TCompConfigGroupRef] [Comment] TCompConfigInfos [Comment] **$End_TCompConfigDcl**
324 TCompConfigId ::= **$TCompConfigId** TCompConfigIdentifier
325 TCompConfigIdentifier ::= Identifier
326 TCompConfigGroupRef ::= **$TCompConfigGroupRef** TCompConfigGroupReference
327 TCompConfigGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {TCompConfigGroupIdentifier "/"}
328 TCompConfigGroupIdentifier ::= Identifier
329 TCompConfigInfos ::= **$TCompConfigInfos** {TCompConfigInfo}+ **$End_TCompConfigInfos**
/* SEMÁNTICA ESTÁTICA – Exactamente una de las TCompConfigInfos será para un Test Component que tenga un TCompRole que es **MTC**. */
330 TCompConfigInfo ::= **$TCompConfigInfo** TCompUsed PCOs_Used CPs_Used [Comment] **$End_TCompConfigInfo**
331 TCompUsed ::= **$TCompUsed** TCompIdentifier
332 PCOs_Used ::= **$PCOs_Used** [PCO_List]
333 PCO_List ::= PCO_Identifier {Comma PCO_Identifier}
/* SEMÁNTICA ESTÁTICA – El número de puntos PCO en la PCO_List será igual al de la declaración de componentes de prueba. */
/* SEMÁNTICA ESTÁTICA – Un PCO_Identifier determinado no se utilizará más de una vez en la misma configuración de componentes de prueba. */
334 CPs_Used ::= **$CPs_Used** [CP_List]
335 CP_List ::= CP_Identifier {Comma CP_Identifier}

/* SEMÁNTICA ESTÁTICA – Para un PTC, el número de CP en la CP_List será igual al de la declaración de componentes de prueba. */
/* SEMÁNTICA ESTÁTICA – Para un MTC, el número de CP en la CP_List no será superior al de la declaración de componentes de prueba. */
/* SEMÁNTICA ESTÁTICA – Un CP_Identifier dado no deberá aparecer más de una vez en una CP_List determinada. */
/* SEMÁNTICA ESTÁTICA – Cada CP_Identifier que se utilice en una configuración de componentes de prueba deberá aparecer en la CP_List de exactamente dos componentes de prueba de esa configuración. */

A.3.3.13 Definiciones de tipo ASP, PDU y CM

A.3.3.13.1 Generalidades

ComplexDefinitions ::= [ASP_TypeDefs] [PDU_TypeDefs] [CM_TypeDefs] {AliasDefsOrGroup}
/* SEMÁNTICA ESTÁTICA – Las PDU serán optativas*/

A.3.3.13.2 Definiciones de tipo ASP

ASP_TypeDefs ::= **$ASP_TypeDefs** [TTCN_ASP_TypeDefs] [ASN1_ASP_TypeDefs] {ASN1_ASP_TypeDefsByRefOrGroup} **$End_ASP_TypeDefs**

A.3.3.13.3 Definiciones de tipo ASP en forma de tabla

TTCN_ASP_TypeDefs ::= **$TTCN_ASP_TypeDefs** {TTCN_ASP_TypeDefOrGroup}+ **$End_TTCN_ASP_TypeDefs**
TTCN_ASP_TypeDefOrGroup ::= TTCN_ASP_TypeDef | TTCN_ASP_TypeDefGroup
TTCN_ASP_TypeDefGroup ::= **$TTCN_ASP_TypeDefGroup** TTCN_ASP_TypeDefGroupId {TTCN_ASP_TypeDefOrGroup}+ **$End_TTCN_ASP_TypeDefGroup**
TTCN_ASP_TypeDefGroupId ::= **$TTCN_ASP_TypeDefGroupId** ASP_GroupIdentifier
TTCN_ASP_TypeDef ::= **$Begin_TTCN_ASP_TypeDef** ASP_Id [ASP_GroupRef] PCO_Type [Comment] ASP_ParDcls [Comment] **$End_TTCN_ASP_TypeDef**
ASP_Id ::= **$ASP_Id** ASP_Id&FullId
ASP_Id&FullId ::= ASP_Identifier [FullIdentifier]
ASP_Identifier ::= Identifier
/* SEMÁNTICA ESTÁTICA – El identificador puede ser AliasIdentifier siempre que se esté con en la columna de comportamiento de un cuadro de comportamiento (es decir, en una descripción de comportamiento). */
ASP_GroupRef ::= **$ASP_GroupRef** ASP_GroupReference
ASP_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {ASP_GroupIdentifier "/"}
ASP_GroupIdentifier ::= Identifier
PCO_Type ::= **$PCO_Type** [PCO_TypeIdentifier]
/* SEMÁNTICA ESTÁTICA – Si no existe cuadro de declaración de PCO_Type, PCO_TypeIdentifier será uno de los tipos de PCO utilizados en el cuadro de declaración de PCO. */
/* SEMÁNTICA ESTÁTICA – Si sólo está definido un PCO simple en una serie de pruebas, PCO_TypeIdentifier es optativo. */
ASP_ParDcls ::= **$ASP_ParDcls** {ASP_ParDcl} **$End_ASP_ParDcls**
ASP_ParDcl ::= **$ASP_ParDcl** ASP_ParId ASP_ParType [Comment] **$End_ASP_ParDcl**
ASP_ParId ::= **$ASP_ParId** ASP_ParIdOrMacro
ASP_ParIdOrMacro ::= ASP_ParId&FullId | MacroSymbol
/* SEMÁNTICA ESTÁTICA – El MacroSymbol se utilizará solamente en combinación con una referencia a un tipo estructurado. */
ASP_ParId&FullId ::= ASP_ParIdentifier [FullIdentifier]
ASP_ParIdentifier ::= Identifier
ASP_ParType ::= **$ASP_ParType** Type&Attributes
/* SEMÁNTICA ESTÁTICA – Type será un PredefinedType o TS_TypeIdentifier, un PDU_Identifier o una **PDU**. */

A.3.3.13.4 Definiciones de tipo ASP en ASN.1

ASN1_ASP_TypeDefs ::= **$ASN1_ASP_TypeDefs** {ASN1_ASP_TypeDefOrGroup} **$End_ASN1_ASP_TypeDefs**
ASN1_ASP_TypeDefOrGroup ::= ASN1_ASP_TypeDef | ASN1_ASP_TypeDefGroup
ASN1_ASP_TypeDefGroup ::= **$ASN1_ASP_TypeDefGroup** ASN1_ASP_TypeDefGroupId {ASN1_ASP_TypeDefOrGroup}+ **$End_ASN1_ASP_TypeDefGroup**
ASN1_ASP_TypeDefGroupId ::= **$ASN1_ASP_TypeDefGroupId** ASN1_ASP_GroupIdentifier
ASN1_ASP_TypeDef ::= **$Begin_ASN1_ASP_TypeDef** ASP_Id [ASN1_ASP_GroupDef] PCO_Type [Comment] ASN1_TypeDefinition [Comment] **$End_ASN1_ASP_TypeDef**
ASN1_ASP_GroupRef ::= **$ASN1_ASP_GroupRef** ASN1_ASP_GroupReference
ASN1_ASP_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {ASN1_ASP_GroupIdentifier "/"}
ASN1_ASP_GroupIdentifier ::= Identifier

A.3.3.13.5 Definiciones de tipo ASP en ASN.1 por referencia

ASN1_ASP_TypeDefsByRefOrGroup ::= ASN1_ASP_TypeDefsByRef | ASN1_ASP_TypeDefsByRefGroup
ASN1_ASP_TypeDefsByRefGroup ::= **$ASN1_ASP_TypeDefsByRefGroup** ASN1_ASP_TypeDefsByRefGroupId {ASN1_ASP_TypeDefsByRefOrGroup}+ **$End_ASN1_ASP_TypeDefsByRefGroup**
ASN1_ASP_TypeDefsByRefGroupId ::= **$ASN1_ASP_TypeDefsByRefGroupId** ASN1_ASP_GroupIdentifier

368 ASN1_ASP_TypeDefsByRef ::= **$Begin_ASN1_ASP_TypeDefsByRef** [ASN1_ASP_DefsByRefGroupRef] {[CollComment] ASN1_ASP_TypeDefByRef}+ [Comment] **$End_ASN1_ASP_TypeDefsByRef**
369 ASN_ASP_DefsByRefGroupRef ::= **$ASN1_ASP_DefsByRefGroupRef** ASN1_ASP_GroupReference
370 ASN1_ASP_TypeDefByRef ::= **$ASN1_ASP_TypeDefByRef** ASP_Id PCO_Type ASN1_TypeReference ASN1_ModuleId [Comment] **$End_ASN1_ASP_TypeDefByRef**
/* SEMÁNTICA ESTÁTICA – ASP_Id no se especificará en un FullIdentifier. */

A.3.3.13.6 Definiciones de tipo PDU

371 PDU_TypeDefs ::= **$PDU_TypeDefs** [TTCN_PDU_TypeDefs] [ASN1_PDU_TypeDefs] {ASN1_PDU_TypeDefsByRefOrGroup} **$End_PDU_TypeDefs**

A.3.3.13.7 Definiciones de tabulares de tipo PDU

372 TTCN_PDU_TypeDefs ::= **$TTCN_PDU_TypeDefs** {TTCN_PDU_TypeDefOrGroup}+ **$End_TTCN_PDU_TypeDefs**
373 TTCN_PDU_TypeDefOrGroup ::= TTCN_PDU_TypeDef | TTCN_PDU_TypeDefGroup
374 TTCN_PDU_TypeDefGroup ::= **$TTCN_PDU_TypeDefGroup** TTCN_PDU_TypeDefGroupId {TTCN_PDU_TypeDefOrGroup}+ **$End_TTCN_PDU_TypeDefGroup**
375 TTCN_PDU_TypeDefGroupId ::= **$TTCN_PDU_TypeDefGroupId** PDU_GroupIdentifier
376 TTCN_PDU_TypeDef ::= **$Begin_TTCN_PDU_TypeDef** PDU_Id [PDU_GroupRef] PCO_Type [PDU_EncodingId] [EncVariationId] [Comment] PDU_FieldDcls [Comment] **$End_TTCN_PDU_TypeDef**
/* SEMÁNTICA ESTÁTICA – Si una PDU se envía o se recibe solamente incrustada en ASPs dentro de la totalidad de la serie de pruebas, PCO_TypeIdentifier (en PCO_Type) es optativo. */
377 PDU_Id ::= **$PDU**_Id PDU_Id&FullId
378 PDU_Id&FullId ::= PDU_Identifier [FullIdentifier]
379 PDU_Identifier ::= Identifier
/* SEMÁNTICA ESTÁTICA – El identificador puede ser AliasIdentifier siempre que se esté con en la columna comportamiento de un cuadro de comportamiento (es decir, en una descripción de comportamiento). */
380 PDU_GroupRef ::= **$PDU_GroupRef** PDU_GroupReference
381 PDU_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {PDU_GroupIdentifier "/"}
382 PDU_GroupIdentifier ::= Identifier
383 PDU_EncodingId ::= **$PDU_EncodingId** [EncodingRuleIdentifier]
384 PDU_FieldDcls ::= **$PDU_FieldDcls** {PDU_FieldDcl} **$End_PDU_FieldDcls**
385 PDU_FieldDcl ::= **$PDU_FieldDcl** PDU_FieldId PDU_FieldType [PDU_FieldEncoding] [Comment] **$End_PDU_FieldDcl**
386 PDU_FieldId ::= **$PDU_FieldId** PDU_FieldIdOrMacro
387 PDU_FieldIdOrMacro ::= PDU_FieldId&FullId | MacroSymbol
/* SEMÁNTICA ESTÁTICA – El MacroSymbol se utilizará solamente en combinación con una referencia a un tipo estructurado. */
388 MacroSymbol ::= "<-"
389 PDU_FieldId&FullId ::= PDU_FieldIdentifier [FullIdentifier]
390 PDU_FieldIdentifier ::= Identifier
391 PDU_FieldType ::= **$PDU_FieldType** Type&Attributes
/* SEMÁNTICA ESTÁTICA – Type será un PredefinedType o TS_TypeIdentifier, un PDU_Identifier o una **PDU**. */
392 Type&Attributes ::= (Type [LengthRestriction]) | PDU
/* SEMÁNTICA OPERACIONAL – El conjunto de valores definidos por LengthRestriction será un subconjunto verdadero de los valores del tipo de base. */
/* SEMÁNTICA ESTÁTICA – LenghtRestriction será proporcionado solamente cuando el tipo de base sea un tipo cadena (es decir BITSTRING, HEXSTRING, OCTETSTRING o CharacterString), o se derive de un tipo cadena. */

A.3.3.13.8 Definiciones de tipo PDU en ASN.1

393 ASN1_PDU_TypeDefs ::= **$ASN1_PDU_TypeDefs** {ASN1_PDU_TypeDefOrGroup} $End_ASN1_PDU_TypeDefs
394 ASN1_PDU_TypeDefOrGroup ::= ASN1_PDU_TypeDef | ASN1_PDU_TypeDefGroup
395 ASN1_PDU_TypeDefGroup ::= **$ASN1_PDU_TypeDefGroup** ASN1_PDU_TypeDefGroupId {ASN1_PDU_TypeDefOrGroup}+ **$End_ASN1_PDU_TypeDefGroup**
396 ASN1_PDU_TypeDefGroupId ::= **$ASN1_PDU_TypeDefGroupId** ASN1_PDU_GroupIdentifier
397 ASN1_PDU_TypeDef ::= **$Begin_ASN1_PDU_TypeDef** PDU_Id [ASN1_PDU_GroupRef] PCO_Type [PDU_EncodingId] [EncVariationId] [Comment] ASN1_TypeDefinition [Comment] **$End_ASN1_PDU_TypeDef**
/* SEMÁNTICA ESTÁTICA – Si una PDU se envía o se recibe solamente incrustada en ASP dentro de la totalidad de la serie de pruebas, PCO_TypeIdentifier (en PCO_Type) es optativo. */
398 ASN1_PDU_GroupRef ::= **$ASN1_PDU_GroupRef** ASN1_PDU_GroupReference
399 ASN1_PDU_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {ASN1_PDU_GroupIdentifier "/"}
400 ASN1_PDU_GroupIdentifier ::= Identifier

A.3.3.13.9 Definiciones de tipo PDU en ASN.1 por referencia

401 ASN1_PDU_TypeDefsByRefOrGroup ::= ASN1_PDU_TypeDefsByRef | ASN1_PDU_TypeDefsByRefGroup
402 ASN1_PDU_TypeDefsByRefGroup ::= **$ASN1_PDU_TypeDefsByRefGroup** ASN1_PDU_TypeDefsByRefGroupId {ASN1_PDU_TypeDefsByRefOrGroup}+ **$End_ASN1_PDU_TypeDefsByRefGroup**
403 ASN1_PDU_TypeDefsByRefGroupId ::= **$ASN1_PDU_TypeDefsByRefGroupId** ASN1_PDU_GroupIdentifier

ASN1_PDU_TypeDefsByRef ::= **$Begin_ASN1_PDU_TypeDefsByRef** [ASN1_PDU_DefsByRefGroupRef] {[CollComment] ASN1_PDU_TypeDefByRef}+ [Comment] **$End_ASN1_PDU_TypeDefsByRef**
ASN1_PDU_DefsByRefGroupRef ::= **$ASN1_PDU_DefsByRefGroupRef** ASN1_PDU_GroupReference
ASN1_PDU_TypeDefByRef ::= **$ASN1_PDU_TypeDefByRef** PDU_Id PCO_Type ASN1_TypeReference ASN1_ModuleId [PDU_EncodingId] [EncVariationId] [Comment] **$End_ASN1_PDU_TypeDefByRef**
/* SEMÁNTICA ESTÁTICA – Si una PDU se envía o se recibe solamente incrustada en ASP dentro de la totalidad de la serie de pruebas, PCO_TypeIdentifier (en PCO_Type) es optativo. */
/* SEMÁNTICA ESTÁTICA – PDU_Id no se especificará en un FullIdentifier. */

A.3.3.13.10 Definiciones de tipo CM

CM_TypeDefs ::= **$CM_TypeDefs** [TTCN_CM_TypeDefs] [ASN1_CM_TypeDefs] **$End_CM_TypeDefs**

A.3.3.13.11 Definición de tipo CM en forma de tabla

TTCN_CM_TypeDefs ::= **$TTCN_CM_TypeDefs** {TTCN_CM_TypeDefOrGroup}+ **$End_TTCN_CM_TypeDefs**
TTCN_CM_TypeDefOrGroup ::= TTCN_CM_TypeDef | TTCN_CM_TypeDefGroup
TTCN_CM_TypeDefGroup ::= **$TTCN_CM_TypeDefGroup** TTCN_CM_TypeDefGroupId {TTCN_CM_TypeDefOrGroup}+ **$End_TTCN_CM_TypeDefGroup**
TTCN_CM_TypeDefGroupId ::= **$TTCN_CM_TypeDefGroupId** CM_GroupIdentifier
TTCN_CM_TypeDef ::= **$Begin_TTCN_CM_TypeDef** CM_Id [CM_GroupRef] [Comment] CM_ParDcls [Comment] **$End_TTCN_CM_TypeDef**
CM_Id ::= **$CM_Id** CM_Identifier
CM_Identifier ::= Identifier
CM_GroupRef ::= **$CM_GroupRef** CM_GroupReference
CM_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {CM_GroupIdentifier "/"}
CM_GroupIdentifier ::= Identifier
CM_ParDcls ::= **$CM_ParDcls** {CM_ParDcl} **$End_CM_ParDcls**
CM_ParDcl ::= **$CM_ParDcl** CM_ParId CM_ParType [Comment] **$End_CM_ParDcl**
CM_ParId ::= **$CM_ParId** CM_ParIdOrMacro
CM_ParIdOrMacro ::= CM_ParIdentifier | MacroSymbol
/* SEMÁNTICA ESTÁTICA – El MacroSymbol se utilizará solamente en combinación con una referencia a un tipo estructurado. */
CM_ParIdentifier ::= Identifier
CM_ParType ::= **$CM_ParType** Type&Attributes

A.3.3.13.12 Definiciones de tipo CM en ASN.1

ASN1_CM_TypeDefs ::= **$ASN1_CM_TypeDefs** {ASN1_CM_TypeDefOrGroup}+ **$End_ASN1_CM_TypeDefs**
ASN1_CM_TypeDefOrGroup ::= ASN1_CM_TypeDef | ASN1_CM_TypeDefGroup
ASN1_CM_TypeDefGroup ::= **$ASN1_CM_TypeDefGroup** ASN1_CM_TypeDefGroupId {ASN1_CM_TypeDefOrGroup}+ **$End_ASN1_CM_TypeDefGroup**
ASN1_CM_TypeDefGroupId ::= **$ASN1_CM_TypeDefGroupId** ASN1_CM_GroupIdentifier
ASN1_CM_TypeDef ::= **$Begin_ASN1_CM_TypeDef** CM_Id [ASN1_CM_GroupRef] [Comment] ASN1_TypeDefinition [Comment] **$End_ASN1_CM_TypeDef**
ASN1_CM_GroupRef ::= **$ASN1_CM_GroupRef** ASN1_CM_GroupReference
ASN1_CM_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {ASN1_CM_GroupIdentifier "/"}
ASN1_CM_GroupIdentifier ::= Identifier

A.3.3.13.13 Diversas clases de definiciones de codificación

EncodingDefs ::= **$EncodingDefs** {EncodingDefinitionsOrGroup} [EncodingVariations] [InvalidFieldEncodingDefs] **$End_EncodingDefs**

A.3.3.13.13.1 Definiciones de codificación

EncodingDefinitionsOrGroup ::= EncodingDefinitions | EncodingDefinitionsGroup
EncodingDefinitionsGroup ::= **$EncodingDefinitionsGroup** EncodingDefinitionsGroupId {EncodingDefinitionsOrGroup}+ **$End_EncodingDefinitionsGroup**
EncodingDefinitionsGroupId ::= **$EncodingDefinitionsGroupId** EncodingGroupIdentifier
EncodingDefinitions ::= **$Begin_EncodingDefinitions** [EncodingGroupRef] {[CollComment] EncodingDefinition}+ [Comment] **$End_EncodingDefinitions**
EncodingGroupRef ::= **$EncodingGroupRef** EncodingGroupReference
EncodingGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {EncodingGroupIdentifier "/"}
EncodingGroupIdentifier ::= Identifier
EncodingDefinition ::= **$EncodingDefinition** EncodingRuleId EncodingRef EncodingDefault [Comment] **$End_EncodingDefinition**
/* SEMÁNTICA OPERACIONAL – Un EncodingRuleIdentifier como máximo tendrá un EncodingDefault que tome el valor TRUE.*/
EncodingRuleId ::= **$EncodingRuleId** EncodingRuleIdentifier
EncodingRuleIdentifier ::= Identifier
EncodingRef ::= **$EncodingRef** EncodingReference

444 EncodingReference ::= BoundedFreeText
445 EncodingDefault ::= **$EncodingDefault** [ConstantExpression]
/* SEMÁNTICA ESTÁTICA – ConstantExpression tendrá un valor booleano asignado */

A.3.3.13.13.2 Variaciones de codificación

446 EncodingVariations ::= **$EncodingVariations** {EncodingVariationSetOrGroup}+ **$End_EncodingVariations**
447 EncodingVariationSetOrGroup ::= EncodingVariationSet | EncodingVariationSetGroup
448 EncodingVariationSetGroup ::= **$EncodingVariationSetGroup** EncodingVariationSetGroupId {EncodingVariationSetOrGroup}+ **$End_EncodingVariationSetGroup**
449 EncodingVariationSetGroupId ::= **$EncodingVariationSetGroupId** EncVariationGroupIdentifier
450 EncodingVariationSet ::= **$Begin_EncodingVariationSet** EncodingRuleId [EncVariationGroupRef] Encoding_TypeList [Comment] EncodingVariationList [Comment] **$End_EncodingVariationSet**
451 EncVariationGroupRef ::= **$EncVariationGroupRef** EncVariationGroupReference
452 EncVariationGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {EncVariationGroupIdentifier "/"}
453 EncVariationGroupIdentifier ::= Identifier
454 EncodingVariationList ::= **$EncodingVariationList** {EncodingVariation}+ **$End_EncodingVariationList**
455 Encoding_TypeList ::= **$Encoding_TypeList** [TypeList]
456 TypeList ::=Type {Comma Type}
/* SEMÁNTICA ESTÁTICA – Type no será un ASP_Identifier, un PDU_Identifier o un StructIdentifier, puesto que tales tipos pueden codificarse mediante reglas de codificación pero no mediante codificaciones de campo. */
457 EncodingVariation ::= **$EncodingVariation** EncodingVariationId VariationRef VariationDefault [Comment] **$End_EncodingVariation**
/* SEMÁNTICA OPERACIONAL – Un EncodingIdentifier como máximo tendrá un VariationDefault que tome el valor TRUE. */
458 EncodingVariationId ::= **$EncodingVariationId** EncVariationId&ParList
459 EncVariationId&ParList ::= EncVariationIdentifier [FormalParList]
460 EncVariationIdentifier ::= Identifier
461 VariationRef ::= **$VariationRef** VariationReference
462 VariationReference ::= BoundedFreeText
463 VariationDefault ::= **$VariationDefault** [ConstantExpression]

A.3.3.13.13.3 Definiciones de codificaciones no válidas

464 InvalidFieldEncodingDefs ::= **$InvalidFieldEncodingDefs** {InvalidFieldEncodingDefOrGroup}+ **$End_InvalidFieldEncodingDefs**
465 InvalidFieldEncodingDefOrGroup ::= InvalidFieldEncodingDef | InvalidFieldEncodingGroup
466 InvalidFieldEncodingGroup ::= **$InvalidFieldEncodingGroup** InvalidFieldEncodingGroupId {InvalidFieldEncodingOrGroup}+ **$End_InvalidFieldEncodingGroup**
467 InvalidFieldEncodingGroupId ::= **$InvalidFieldEncodingGroupId** InvalidFieldEncodingGroupIdentifier
468 InvalidFieldEncodingDef ::= **$Begin_InvalidFieldEncodingDef** InvalidFieldEncodingId [InvalidFieldEncodingGroupRef] Encoding_TypeList [Comment] InvalidFieldEncodingDefinition [Comment] **$End_InvalidFieldEncodingDef**
469 InvalidFieldEncodingId ::= **$InvalidFieldEncodingId** InvalidFieldEncodingId&ParList
470 InvalidFieldEncodingId&ParList ::= InvalidFieldEncodingIdentifier [FormalParList]
471 InvalidFieldEncodingIdentifier ::= Identifier
472 InvalidFieldEncodingGroupRef ::= **$InvalidFieldEncodingGroupRef** InvalidFieldEncodingGroupReference
473 InvalidFieldEncodingGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"]{InvalidFieldEncodingGroupIdentifier"/"}
474 InvalidFieldEncodingGroupIdentifier ::= Identifier
475 InvalidFieldEncodingDefinition ::= **$InvalidFieldEncodingDefinition** TS_OpProcDef **$End_InvalidFieldEncodingDefinition**
/* SEMÁNTICA OPERACIONAL – TS_OpProcDef producirá un resultado BitString, que será interpretado como la codificación que ha de transmitirse siendo el primer bit el de orden más elevado. */

A.3.3.13.14 Definiciones de alias

476 AliasDefsOrGroup ::= AliasDefs | AliasDefsGroup
477 AliasDefsGroup ::= **$AliasDefsGroup** AliasDefsGroupId {AliasDefsOrGroup}+ **$End_AliasDefsGroup**
478 AliasDefsGroupId ::= **$AliasDefsGroupId** AliasDefsGroupIdentifier
479 AliasDefsGroupIdentifier ::= Identifier
480 AliasDefs ::= **$Begin_AliasDefs** [AliasGroupRef] {[CollComment] AliasDef}+ [Comment] **$End_AliasDefs**
481 AliasGroupRef ::= **$AliasGroupRef** AliasGroupReference
482 AliasGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {AliasGroupIdentifier "/"}
483 AliasGroupIdentifier ::= Identifier
484 AliasDef ::= **$AliasDef** AliasId ExpandedId [Comment] **$End_AliasDef**
485 AliasId ::= **$AliasId** AliasIdentifier
486 AliasIdentifier ::= Identifier
/* SEMÁNTICA ESTÁTICA – Un AliasIdentifier se utilizará solamente en una línea de enunciado una descripción de comportamiento. */
/* SEMÁNTICA ESTÁTICA – Un AliasIdentifier se utilizará solamente donde un ASP_Identifier o PDU_Identifier sea válido. */

487 ExpandedId ::= **$ExpandedId** Expansion
488 Expansion ::= ASP_Identifier | PDU_Identifier

A.3.3.14 Parte constricciones

489 ConstraintsPart ::= **$ConstraintsPart** [TS_TypeConstraints] [ASP_Constraints] [PDU_Constraints] [CM_Constraints] **$End_ConstraintsPart**

A.3.3.15 Declaraciones de constricciones de tipo de serie de pruebas

490 TS_TypeConstraints ::= **$TS_TypeConstraints** [StructTypeConstraints] [ASN1_TypeConstraints] **$End_TS_TypeConstraints**

A.3.3.16 Declaraciones de constricciones de estructurado

491 StructTypeConstraints ::= **$StructTypeConstraints** {StructTypeConstraintOrGroup}+ **$End_StructTypeConstraints**
492 StructTypeConstraintOrGroup ::= StructTypeConstraint | StructTypeConstraintGroup
493 StructTypeConstraintGroup ::= **$StructTypeConstraintGroup** StructTypeConstraintGroupId {StructTypeConstraintOrGroup}+ **$End_StructTypeConstraintGroup**
494 StructTypeConstraintGroupId ::= **$StructTypeConstraintGroupId** StructTypeConstraintGroupIdentifier
495 StructTypeConstraint ::= **$Begin_StructTypeConstraint** ConsId [StructTypeConstraintGroupRef] StructId DerivPath [EncVariationId] [Comment] ElemValues [Comment] **$End_StructTypeConstraint**
/* SEMÁNTICA ESTÁTICA – No se utilizará el FullIdentifier que forma parte del Struct_Id. */
/* SEMÁNTICA ESTÁTICA – Una constricción modificada tendrá la misma lista de parámetros que su constricción de base. En particular, no habrá parámetros omitidos de la lista o añadidos a ella. */
496 StructTypeConstraintGroupRef ::= **$StructTypeConstraintGroupRef** StructTypeConstraintGroupReference
497 StructTypeConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"]{StructTypeConstraintGroupIdentifier"/"}
498 StructTypeConstraintGroupIdentifier ::= Identifier
499 EncVariationId ::= **$EncVariationId** [EncVariationCall]
500 EncVariationCall ::= EncVariationIdentifier [ActualParList]
501 ElemValues ::= **$ElemValues** {ElemValue}+ **$End_ElemValues**
502 ElemValue ::= **$ElemValue** ElemId ConsValue [PDU_FieldEncoding] [Comment] **$End_ElemValue**
/* SEMÁNTICA ESTÁTICA – No se utilizará el FullIdentifier que forma parte de ElemId. */
/* SEMÁNTICA ESTÁTICA – Hay que declarar ElemId en el tipo relacionado con la constricción. */
/* SEMÁNTICA ESTÁTICA – Los valores de elementos parametrizados en una constricción de base no serán modificados ni omitidos explícitamente en una constricción modificada. */
503 PDU_FieldEncoding ::= **$PDU_FieldEncoding** [PDU_FieldEncodingCall]
504 PDU_FieldEncodingCall ::= EncVariationCall | InvalidFieldEncodingCall
505 InvalidFieldEncodingCall ::= InvalidFieldEncodingIdentifier (ActualParList | "(" ")")

A.3.3.17 Declaraciones de constricciones de tipo ASN.1

506 ASN1_TypeConstraints ::= **$ASN1_TypeConstraints** {ASN1_TypeConstraintOrGroup}+ **$End_ASN1_TypeConstraints**
507 ASN1_TypeConstraintOrGroup ::= ASN1_TypeConstraint | ASN1_TypeConstraintGroup
508 ASN1_TypeConstraintGroup ::= **$ASN1_TypeConstraintGroup** ASN1_TypeConstraintGroupId {ASN1_TypeConstraintOrGroup}+ **$End_ASN1_TypeConstraintGroup**
509 ASN1_TypeConstraintGroupId ::= **$ASN1_TypeConstraintGroupId** ASN1_TypeConstraintGroupIdentifier
510 ASN1_TypeConstraint ::= **$Begin_ASN1_TypeConstraint** ConsId [ASN1_TypeConstraintGroupRef] ASN1_TypeId DerivPath [EncVariationId] [Comment] ASN1_ConsValue [Comment] **$End_ASN1_TypeConstraint**
/* SEMÁNTICA ESTÁTICA – No se utilizará el FullIdentifier que forma parte de ASN1_TypeId. */
/* SEMÁNTICA ESTÁTICA – Una constricción modificada deberá tener la misma lista de parámetros que su constricción de base. En particular, no habrá parámetros omitidos de la lista o añadidos a ella. */
511 ASN1_TypeConstraintGroupRef ::= **$ASN1_TypeConstraintGroupRef** ASN1_TypeConstraintGroupReference
512 ASN1_TypeConstraintGroupReference::=[(SuiteIdentifier | TTCN_ModuleIdentifier)"/"]{ASN1_TypeConstraintGroupIdentifier"/"}
513 ASN1_TypeConstraintGroupIdentifier ::= Identifier

A.3.3.18 Declaraciones de constricciones de ASP

514 ASP_Constraints ::= **$ASP_Constraints** [TTCN_ASP_Constraints] [ASN1_ASP_Constraints] **$End_ASP_Constraints**

A.3.3.19 Declaraciones de constricciones ASP en forma de tablas

515 TTCN_ASP_Constraints ::= **$TTCN_ASP_Constraints** {TTCN_ASP_ConstraintOrGroup}+ **$End_TTCN_ASP_Constraints**
516 TTCN_ASP_ConstraintOrGroup ::= TTCN_ASP_Constraint | TTCN_ASP_ConstraintGroup
517 TTCN_ASP_ConstraintGroup ::= **$TTCN_ASP_ConstraintGroup** TTCN_ASP_ConstraintGroupId {TTCN_ASP_ConstraintOrGroup}+ **$End_TTCN_ASP_ConstraintGroup**
518 TTCN_ASP_ConstraintGroupId ::= **$TTCN_ASP_ConstraintGroupId** ASP_ConstraintGroupIdentifier

519 TTCN_ASP_Constraint ::= **$Begin_TTCN_ASP_Constraint** ConsId [ASP_ConstraintGroupRef] ASP_Id DerivPath [Comment] ASP_ParValues [Comment] **$End_TTCN_ASP_Constraint**
/* SEMÁNTICA ESTÁTICA – No se utilizará el FullIdentifier que forma parte de ASP_Id. */
/* SEMÁNTICA ESTÁTICA – Si una ASP está subestructurada, las constricciones para ASP de ese tipo deberán tener la misma estructura.*/
/* SEMÁNTICA ESTÁTICA – Una constricción modificada deberá tener la misma lista de parámetros que su constricción de base. En particular, no habrá parámetros omitidos de la lista o añadidos a ella. */
520 ASP_ConstraintGroupRef ::= **$ASP_ConstraintGroupRef** ASP_ConstraintGroupReference
521 ASP_ConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {ASP_ConstraintGroupIdentifier "/"}
522 ASP_ConstraintGroupIdentifier ::= Identifier
523 ASP_ParValues ::= **$ASP_ParValues** {ASP_ParValue} **$End_ASP_ParValues**
524 ASP_ParValue ::= **$ASP_ParValue** ASP_ParId ConsValue [Comment] **$End_ASP_ParValue**
/* SEMÁNTICA ESTÁTICA – No se utilizará el FullIdentifier que forma parte de ASP_ParId. */
/* SEMÁNTICA ESTÁTICA – Hay que declarar ASP_ParId en el tipo relacionado con la constricción. */
/* SEMÁNTICA ESTÁTICA – Si una definición de ASP se refiere a un tipo estructurado como una subestructura de un parámetro (es decir, con un nombre de parámetro), la constricción correspondiente tendrá el mismo nombre de parámetro en la posición correspondiente de la columna de nombre de parámetro la constricción, y el valor será una referencia a una constricción para ese parámetro (es decir, para esa subestructura de acuerdo con la definición del tipo estructurado). */
/* SEMÁNTICA ESTÁTICA – Si una definición de ASP se refiere a un parámetro especificado como del metatipo PDU, en la constricción correspondiente se especificará el valor de ese parámetro como el nombre de una constricción PDU, o un parámetro formal. */
/* SEMÁNTICA ESTÁTICA – Las constricciones estructuradas por expansión de macro en una constricción no deberán utilizarse a menos que la correspondiente definición de ASP haga referencia al mismo tipo estructurado por expansión de macro. */
/* SEMÁNTICA ESTÁTICA – Los valores de parámetro ASP parametrizados en una constricción de base no serán modificados ni omitidos explícitamente en una constricción modificada. */

A.3.3.20 Declaraciones de constricciones de ASP en ASN.1

525 ASN1_ASP_Constraints ::= **$ASN1_ASP_Constraints** {ASN1_ASP_ConstraintOrGroup}+ **$End_ASN1_ASP_Constraints**
526 ASN1_ASP_ConstraintOrGroup ::= ASN1_ASP_Constraint | ASN1_ASP_ConstraintGroup
527 ASN1_ASP_ConstraintGroup ::= **$ASN1_ASP_ConstraintGroup** ASN1_ASP_ConstraintGroupId {ASN1_ASP_ConstraintOrGroup}+ **$End_ASN1_ASP_ConstraintGroup**
528 ASN1_ASP_ConstraintGroupId ::= **$ASN1_ASP_ConstraintGroupId** ASN1_ASP_ConstraintGroupIdentifier
529 ASN1_ASP_Constraint ::= **$Begin_ASN1_ASP_Constraint** ConsId [ASN1_ASP_ConstraintGroupRef] ASP_Id DerivPath [Comment] ASN1_ConsValue [Comment] **$End_ASN1_ASP_Constraint**
/* SEMÁNTICA ESTÁTICA – No se utilizará el FullIdentifier que forma parte de ASP_Id. */
/* SEMÁNTICA ESTÁTICA – Si una ASP está subestructurada, las constricciones para ASP de ese tipo tendrán una estructura ASN.1 compatible (es decir, posiblemente con algunas agrupaciones). */
/* SEMÁNTICA ESTÁTICA – Una constricción modificada deberá tener la misma lista de parámetros que su constricción de base. En particular, no habrá parámetros omitidos de la lista o añadidos a ella. */
530 ASN1_ASP_ConstraintGroupRef ::= **$ASN1_ASP_ConstraintGroupRef** ASN1_ASP_ConstraintGroupReference
531 ASN1_ASP_ConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {ASN1_ASP_ConstraintGroupIdentifier "/"}
532 ASN1_ASP_ConstraintGroupIdentifier ::= Identifier

A.3.3.21 Declaraciones de constricciones de PDU

533 PDU_Constraints ::= **$PDU_Constraints** [TTCN_PDU_Constraints] [ASN1_PDU_Constraints] **$End_PDU_Constraints**

A.3.3.22 Declaraciones de constricciones de PDU en forma de tabla

534 TTCN_PDU_Constraints ::= **$TTCN_PDU_Constraints** {TTCN_PDU_ConstraintOrGroup}+ **$End_TTCN_PDU_Constraints**
535 TTCN_PDU_ConstraintOrGroup ::= TTCN_PDU_Constraint | TTCN_PDU_ConstraintGroup
536 TTCN_PDU_ConstraintGroup ::= **$TTCN_PDU_ConstraintGroup** TTCN_PDU_ConstraintGroupId {TTCN_PDU_ConstraintOrGroup}+ **$End_TTCN_PDU_ConstraintGroup**
537 TTCN_PDU_ConstraintGroupId ::= **$TTCN_PDU_ConstraintGroupId** PDU_ConstraintGroupIdentifier
538 TTCN_PDU_Constraint ::= **$Begin_TTCN_PDU_Constraint** ConsId [PDU_ConstraintGroupRef] PDU_Id DerivPath [EncRuleId] [EncVariationId] [Comment] PDU_FieldValues [Comment] **$End_TTCN_PDU_Constraint**
/* SEMÁNTICA ESTÁTICA – No se utilizará el FullIdentifier que forma parte de PDU_Id. */
/* SEMÁNTICA ESTÁTICA – Si una PDU está subestructurada, las constricciones para PDU de ese tipo tendrán la misma estructura.*/
/* SEMÁNTICA ESTÁTICA – Una constricción modificada deberá tener la misma lista de parámetros que su constricción de base. En particular, no habrá parámetros omitidos de la lista o añadidos a ella. */
539 PDU_ConstraintGroupRef ::= **$PDU_ConstraintGroupRef** PDU_ConstraintGroupReference
540 PDU_ConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {PDU_ConstraintGroupIdentifier "/"}
541 PDU_ConstraintGroupIdentifier ::= Identifier
542 EncRuleId ::= **$EncRuleId** [EncodingRuleIdentifier]
543 ConsId ::= **$ConsId** ConsId&ParList

544 ConsId&ParList ::= ConstraintIdentifier [FormalParList]
545 ConstraintIdentifier ::= Identifier
546 DerivPath ::= **$DerivPath** [DerivationPath]
547 DerivationPath ::= {ConstraintIdentifier Dot}+
/* SEMÁNTICA ESTÁTICA – Si una definición de constricción es una modificación de una constricción existente, en la entrada de trayecto de derivación del cuadro se hará referencia al nombre de la constricción que se ha tomado como base de esta modificación. */
/* SEMÁNTICA ESTÁTICA – El primer ConstraintIdentifier en DerivationPath será un identificador de constricción de base. */
/* SEMÁNTICA ESTÁTICA – El DerivationPath será la lista completa de constricciones en el orden en que se aplican sus modificaciones a la constricción de base. */
/* SEMÁNTICA ESTÁTICA – No habrá espacios en blanco entre ConstraintIdentifier y el punto (Dot). */
548 PDU_FieldValues ::= **$PDU_FieldValues** {PDU_FieldValue} **$End_PDU_FieldValues**
549 PDU_FieldValue ::= **$PDU_FieldValue** PDU_FieldId ConsValue [PDU_FieldEncoding] [Comment] **$End_PDU_FieldValue**
/* SEMÁNTICA ESTÁTICA – No se utilizará el FullIdentifier que forma parte de PDU_FieldId. */
/* SEMÁNTICA ESTÁTICA – Hay que declarar PDU_FieldId en el tipo relacionado con la constricción. */
/* SEMÁNTICA ESTÁTICA – Si una definición de PDU se refiere a un tipo estructurado como una subestructura de un campo (es decir, con un nombre de campo), la constricción correspondiente tendrá el mismo nombre de campo en la posición correspondiente de la columna de nombre de campo de la constricción, y el valor será una referencia a una constricción para ese campo (es decir, para esa subestructura de acuerdo con la definición del tipo estructurado). */
/* SEMÁNTICA ESTÁTICA – Si una definición de PDU se refiere a un campo especificado como del metatipo PDU, en la constricción correspondiente se especificará el valor de ese campo como el nombre de una constricción de PDU, o un parámetro formal. */
/* SEMÁNTICA ESTÁTICA – Las constricciones tructuradas por expansión de macro, en una constricción no deberán utilizarse a menos que la correspondiente definición de PDU haga referencia al mismo tipo estructurado por expansión de macro. */
/* SEMÁNTICA ESTÁTICA – Los valores de campo de PDU parametrizados en una constricción de base no serán modificados ni explícitamente omitidos en una constricción modificada. */
550 ConsValue ::= **$ConsValue** ConstraintValue&Attributes
/* SEMÁNTICA OPERACIONAL – ConsValue tomará el valor de un elemento del tipo especificado para el parámetro ASP, campo de PDU o elemento de estructura. Éste puede incluir símbolos de concordancia compatibles con el tipo especificado. */
551 ConstraintValue&Attributes ::= ConstraintValue ValueAttributes
/* NOTA – ConstraintValue&Attributes pueden conseguirse vía DefinedValue en la sintaxis ASN.1. Véase la referencia en la producción 739 para Value. */
/* SEMÁNTICA ESTÁTICA – ConstraintValue deberá cumplir todas las restricciones definidas para el parámetro ASP, campo de PDU o tipo de elemento de estructura, incluyendo gamas de valores, listas de valores, restricciones de alfabeto y/o restricciones de longitud, deberá cumplir las restricciones definidas por ValueAttributes. */
/* SEMÁNTICA OPERACIONAL – Ninguna de las especificaciones de longitud definidas para el tipo de parámetro ASP o campo de PDU en las declaraciones de tipo de serie de pruebas estará en contradicción con las especificaciones de longitud en la definición de tipo ASP o PDU. */
/* SEMÁNTICA ESTÁTICA – Ni las variables de series de pruebas, ni las variables de casos de prueba deberán utilizarse en constricciones, a menos que se pasen como parámetros efectivos. En este último caso, estarán acotadas a un valor y no serán cambiadas. */
552 ConstraintValue ::= ConstantExpression | MatchingSymbol | ConsRef
/* SEMÁNTICA OPERACIONAL – ConstantExpression tomará el valor de un elemento del tipo especificado. */
553 MatchingSymbol ::= Complement | Omit | AnyValue | AnyOrOmit | ValueList | IntegerRange | SuperSet | SubSet | Permutation
/* NOTA – Si no hay símbolo de concordancia se considera como un valor específico. */
554 Complement ::= COMPLEMENT ValueList
555 Omit ::= Dash | OMIT
/* SEMÁNTICA ESTÁTICA – En constricciones en ASN.1, se utilizará Omit solamente para parámetros de ASP o campos de PDU que están declarados OPTIONAL o DEFAULT. */
556 AnyValue ::= "?"
557 AnyOrOmit ::= "*"
558 ValueList ::= "(" ConstraintValue&Attributes {Comma ConstraintValue&Attributes} ")"
/* SEMÁNTICA ESTÁTICA – Cada uno de los ConstraintValue&Attributes será del tipo declarado para el parámetro ASP, campo de PDU o elemento de estructura en que se utilizó la ValueList. */
559 SuperSet ::= **SUPERSET** "(" ConstraintValue&Attributes ")"
/* SEMÁNTICA ESTÁTICA – El argumento de SuperSet, es decir, ConstraintValue&Attributes, será de tipo SET OF. */
560 SubSet ::= **SUBSET** "(" ConstraintValue&Attributes ")"
/* SEMÁNTICA ESTÁTICA – El argumento de SubSet, es decir, ConstraintValue&Attributes, será de tipo SET OF. */
561 Permutation ::= **PERMUTATION** ValueList
/* SEMÁNTICA ESTÁTICA -- La Permutation se usará sólo dentro de un valor de tipo SEQUENCE OF. */
/* SEMÁNTICA ESTÁTICA – ValueList será del tipo especificado en SEQUENCE OF. */
562 ValueAttributes ::= [LengthRestriction] [**IF_PRESENT**] [ASN1_Encoding]
/* SEMÁNTICA ESTÁTICA – Las constricciones de ASN.1 IF_PRESENT deberán utilizarse solamente para parámetros de ASP o campos de PDU que están declarados OPTIONAL o DEFAULT. */

/* SEMÁNTICA ESTÁTICA – ASN1_Encoding solamente se utilizará para ValueAttributes en constricciones de tipo ASN.1 y constricciones de PDU en ASN.1. */
/* SEMÁNTICA ESTÁTICA -- LengthRestriction se usará sólo en parámetros ASP, campos PDU o elementos estructurados declarados como BITSTRING, HEXSTRING, OCTETSTRING, CharacterString, SEQUENCE OF o SET OF.*/
*/ SEMÁNTICA ESTÁTICA -- LengthRestriction se usará sólo en combinación con los mecnaismos siguientes: SpecificValue, Complement, Omit, AnyValue, AnyOrOmit, AnyOrNone y Permutation. */
*/ SEMÁNTICA ESTÁTICA -- El conjunto de valores definido por LengthRestriction será un subconjunto verdadero de los valores permitidos por los tipos declarados de los parámetros ASP, los campos PDU y los elementos estructurados. */

563 ASN1_Encoding ::= **ENC** PDU_FieldEncodingCall

A.3.3.23 Declaraciones de constricciones de PDU en ASN.1

564 ASN1_PDU_Constraints ::= **$ASN1_PDU_Constraints** {ASN1_PDU_ConstraintOrGroup}+ **$End_ASN1_PDU_Constraints**
565 ASN1_PDU_ConstraintOrGroup ::= ASN1_PDU_Constraint | ASN1_PDU_ConstraintGroup
566 ASN1_PDU_ConstraintGroup ::= **$ASN1_PDU_ConstraintGroup** ASN1_PDU_ConstraintGroupId {ASN1_PDU_ConstraintOrGroup}+ **$End_ASN1_PDU_ConstraintGroup**
567 ASN1_PDU_ConstraintGroupId ::= **$ASN1_PDU_ConstraintGroupId** ASN1_PDU_ConstraintGroupIdentifier
568 ASN1_PDU_Constraint ::= **$Begin_ASN1_PDU**_Constraint ConsId [ASN1_PDU_ConstraintGroupRef] PDU_Id DerivPath [EncRuleId] [EncVariationId] [Comment] ASN1_ConsValue [Comment] **$End_ASN1_PDU_Constraint**
/* SEMÁNTICA ESTÁTICA – No se utilizará el FullIdentifier que forma parte de PDU_Id. */
/* SEMÁNTICA ESTÁTICA – Si un PDU está subestructurado, las constricciones para PDU de ese tipo tendrán una estructura ASN.1 compatible (es decir, posiblemente con algunas agrupaciones). */
/* SEMÁNTICA ESTÁTICA – Una constricción modificada deberá tener la misma lista de parámetros que su constricción de base. En particular, no habrá parámetros omitidos de la lista o añadidos a ella. */
569 ASN1_PDU_ConstraintGroupRef ::= **$ASN1_PDU_ConstraintGroupRef** ASN1_PDU_ConstraintGroupReference
570 ASN1_PDU_ConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {ASN1_PDU_ConstraintGroupIdentifier "/"}
571 ASN1_PDU_ConstraintGroupIdentifier ::= Identifier
572 ASN1_ConsValue ::= **$ASN1_ConsValue** ConstraintValue&AttributesOrReplace **$End_ASN1_ConsValue**
573 ConstraintValue&AttributesOrReplace ::= ConstraintValue&Attributes | Replacement {Comma Replacement}
574 Replacement ::= **REPLACE** ReferenceList **BY** ConstraintValue&Attributes | **OMIT** ReferenceList
/* SEMÁNTICA ESTÁTICA – Replacement sólo se utilizará cuando el DerivPath está especificado. */
/* SEMÁNTICA ESTÁTICA – Los valores reemplazados parametrizados en una constricción de base no será modificados u omitidos explícitamente en una constricción modificada. */
575 ReferenceList ::= (ArrayRef | ComponentIdentifier | ComponentPosition) {ComponentReference}

A.3.3.24 Declaraciones de constricciones de CM

576 CM_Constraints ::= **$CM_Constraints** [TTCN_CM_Constraint**s**] [ASN1_CM_Constraint**s**] **$End_CM_Constraints**

A.3.3.25 Declaraciones de constricciones de CM en forma de tabla

577 TTCN_CM_Constraints ::= **$TTCN_CM_Constraints** {TTCN_CM_ConstraintOrGroup}+ **$End_TTCN_CM_Constraints**
578 TTCN_CM_ConstraintOrGroup ::= TTCN_CM_Constraint | TTCN_CM_ConstraintGroup
579 TTCN_CM_ConstraintGroup ::= **$TTCN_CM_ConstraintGroup** TTCN_CM_ConstraintGroupId {TTCN_CM_ConstraintOrGroup}+ **$End_TTCN_CM_ConstraintGroup**
580 TTCN_CM_ConstraintGroupId ::= **$TTCN_CM_ConstraintGroupId** CM_ConstraintGroupIdentifier
581 TTCN_CM_Constraint ::= **$Begin_TTCN_CM_Constraint** ConsId [CM_ConstraintGroupRef] CM_Id DerivPath [Comment] CM_ParValues [Comment] **$End_TTCN_CM_Constraint**
582 CM_ConstraintGroupRef ::= **$CM_ConstraintGroupRef** CM_ConstraintGroupReference
583 CM_ConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {CM_ConstraintGroupIdentifier "/"}
584 CM_ConstraintGroupIdentifier ::= Identifier
585 CM_ParValues ::= **$CM_ParValues** {CM_ParValue} **$End_CM_ParValues**
586 CM_ParValue ::= **$CM_ParValue** CM_ParId ConsValue [Comment] **$End_CM_ParValue**
/* SEMÁNTICA ESTÁTICA – Hay que declarar CM_ParId en el tipo relacionado con la constricción. */

A.3.3.26 Declaración de constricciones de CM en ASN.1

587 ASN1_CM_Constraints ::= **$ASN1_CM_Constraints** {ASN1_CM_ConstraintOrGroup}+ **$End_ASN1_CM_Constraints**
588 ASN1_CM_ConstraintOrGroup ::= ASN1_CM_Constraint | ASN1_CM_ConstraintGroup
589 ASN1_CM_ConstraintGroup ::= **$ASN1_CM_ConstraintGroup** ASN1_CM_ConstraintGroupId {ASN1_CM_ConstraintOrGroup}+ **$End_ASN1_CM_ConstraintGroup**
590 ASN1_CM_ConstraintGroupId ::= **$ASN1_CM_ConstraintGroupId** ASN1_CM_ConstraintGroupIdentifier
591 ASN1_CM_Constraint ::= **$Begin_ASN1_CM_Constraint** ConsId [ASN1_CM_ConstraintGroupRef] CM_Id DerivPath [Comment] ASN1_ConsValue [Comment] **$End_ASN1_CM_Constraint**
592 ASN1_CM_ConstraintGroupRef ::= **$ASN1_CM_ConstraintGroupRef** ASN1_CM_ConstraintGroupReference
593 ASN1_CM_ConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {ASN1_CM_ConstraintGroupIdentifier "/"}

594 ASN1_CM_ConstraintGroupIdentifier ::= Identifier

A.3.3.27 Parte dinámica

595 DynamicPart ::= **$DynamicPart** [TestCases] [TestStepLibrary] [DefaultsLibrary] **$End_DynamicPart**

A.3.3.28 Casos de prueba

596 TestCases ::= **$TestCases** {TestGroup | TestCase}+ **$End_TestCases**
597 TestGroup ::= **$TestGroup** TestGroupId {TestGroup | TestCase}+ **$End_TestGroup**
598 TestGroupId ::= **$TestGroupId** TestGroupIdentifier
599 TestGroupIdentifier ::= Identifier
600 TestCase ::= **$Begin_TestCase** TestCaseId TestGroupRef TestPurpose [Configuration] DefaultsRef [Comment] BehaviourDescription [Comment] **$End_TestCase**
601 TestCaseId ::= **$TestCaseId** TestCaseIdentifier
602 TestCaseIdentifier ::= Identifier
603 TestGroupRef ::= **$TestGroupRef** TestGroupReference
604 TestGroupReference ::= [SuiteIdentifier "/"] {TestGroupIdentifier "/"}
/* SEMÁNTICA ESTÁTICA – No habrá espacios en blanco en ninguno de los dos lados de los caracteres "/". */
605 TestPurpose ::= **$TestPurpose** BoundedFreeText
606 Configuration ::= **$Configuration** TCompConfigIdentifier
607 DefaultsRef::= **$DefaultsRef** [DefaultRefList]
608 DefaultRefList ::= DefaultReference {Comma DefaultReference}
609 DefaultReference ::= DefaultIdentifier [ActualParList]

A.3.3.29 Biblioteca de pasos de prueba

610 TestStepLibrary ::= **$TestStepLibrary** {TestStepGroup | TestStep}+ **$End_TestStepLibrary**
611 TestStepGroup ::= **$TestStepGroup** TestStepGroupId {TestStepGroup | TestStep}+ **$End_TestStepGroup**
612 TestStepGroupId ::= **$TestStepGroupId** TestStepGroupIdentifier
613 TestStepGroupIdentifier ::= Identifier
614 TestStep ::= **$Begin_TestStep** TestStepId TestStepRef Objective DefaultsRef [Comment] BehaviourDescription [Comment] **$End_TestStep**
615 TestStepId ::= **$TestStepId** TestStepId&ParList
616 TestStepId&ParList ::= TestStepIdentifier [FormalParList]
617 TestStepIdentifier ::= Identifier
618 TestStepRef ::= **$TestStepRef** TestStepGroupReference
619 TestStepGroupReference ::= [SuiteIdentifier "/"] {TestStepGroupIdentifier "/"}
/* SEMÁNTICA ESTÁTICA – No habrá espacios en blanco en ninguno de los dos lados de los caracteres "/". */
620 Objective ::= **$Objective** BoundedFreeText

A.3.3.30 Biblioteca de supletorios

621 DefaultsLibrary ::= **$DefaultsLibrary** {DefaultGroup | Default}+ **$End_DefaultsLibrary**
622 DefaultGroup ::= **$DefaultGroup** DefaultGroupId {DefaultGroup | Default}+ **$End_DefaultGroup**
623 DefaultGroupId ::= **$DefaultGroupId** DefaultGroupIdentifier
624 Default ::= **$Begin_Default** DefaultId DefaultRef Objective [Comment] BehaviourDescription [Comment] **$End_Default**
/* SEMÁNTICA ESTÁTICA – BehaviourDescription no utilizará adjunción de árbol excepto para adjuntar árboles locales (es decir, los árboles de comportamiento por defecto no adjuntarán pasos de prueba). */
625 DefaultRef ::= **$DefaultRef** DefaultGroupReference
626 DefaultId ::= **$DefaultId** DefaultId&ParList
627 DefaultId&ParList ::= DefaultIdentifier [FormalParList]
628 DefaultIdentifier ::= Identifier
629 DefaultGroupReference ::= [SuiteIdentifier "/"] {DefaultGroupIdentifier "/"}
/* SEMÁNTICA ESTÁTICA – No habrá espacios en blanco en ninguno de los dos lados de los caracteres "/". */
630 DefaultGroupIdentifier ::= Identifier

A.3.3.31 Descripciones de comportamiento

631 BehaviourDescription ::= **$BehaviourDescription** RootTree {LocalTree} **$End_BehaviourDescription**
632 RootTree ::= {BehaviourLine}+
633 LocalTree ::= Header {BehaviourLine}+
634 Header ::= **$Header** TreeHeader
635 TreeHeader ::= TreeIdentifier [FormalParList]
636 TreeIdentifier ::= Identifier
637 FormalParList ::= "(" FormalPar&Type {SemiColon FormalPar&Type} ")"
638 FormalPar&Type ::= FormalParIdentifier {Comma FormalParIdentifier} Colon FormalParType
639 FormalParIdentifier ::= Identifier
640 FormalParType ::= Type | PCO_TypeIdentifier | **PDU** | **CP** | **TIMER**
/* SEMÁNTICA ESTÁTICA – En una operación serie de pruebas o una operación de codificación FormalParType no será un tipo de PCO o la palabra clave CP.*/

/* SEMÁNTICA ESTÁTICA – Si un parámetro formal es del tipo de **PDU**, no se utilizará con una referencia a componente (es decir, los campos específicos de la PDU no pueden referenciarse). */

A.3.3.32 Líneas de comportamiento

641 BehaviourLine ::= **$BehaviourLine** LabelId Line Cref VerdictId [Comment] **$End_BehaviourLine**
642 Line ::= **$Line** Indentation StatementLine
643 Indentation ::= "[" Number "]"
/* SEMÁNTICA ESTÁTICA – Los enunciados del primer nivel de alternativas de una descripción de comportamiento tendrán un sangrado de valor cero. */
/* SEMÁNTICA ESTÁTICA – Los enunciados que tengan un predecesor tendrán el valor de sangrado del predecesor más uno, como su valor de sangrado. */
644 LabelId ::= **$LabelId** [Label]
645 Label ::= Identifier
646 Cref ::= **$Cref** [ConstraintReference]
647 ConstraintReference ::= ConsRef | FormalParIdentifier | AnyValue
/* SEMÁNTICA ESTÁTICA – ConsRef deberá estar presente conjuntamente con SEND, IMPLICIT SEND y RECEIVE, y tendrá un tipo coherente con (es decir, el mismo o un subconjunto de) el tipo de ASP, PDU o CM especificado en el enunciado SEND, IMPLICIT_SEND o RECEIVE. No se necesita una ConstraintReference para ASP y CM que no tengan parámetros o PDU que no tengan campos. No estará presente con ninguna otra categoría de enunciado en TTCN. */
/* SEMÁNTICA ESTÁTICA – FormalParIdentifier deberá resolver a una ConsRef. */
/* SEMÁNTICA ESTÁTICA – ConstraintReferences sobre eventos SEND no incluirán ningún MatchingSymbol excepto Omit, a menos que se asigne a MatchingSymbol, explícitamente, valores pecíficos en la línea de evento SEND. */
648 ConsRef ::= ConstraintIdentifier [ActualCrefParList]
649 ActualCrefParList ::= "(" ActualCrefPar {Comma ActualCrefPar} ")"
/* SEMÁNTICA ESTÁTICA – Véase la semántica estática en la producción 699. */
650 ActualCrefPar ::= Value
/* NOTA – Mediante Value, se puede alcanzar MatchingSymbol, TS_ParIdentifier, TS_ConstIdentifier, TS_VarIdentifier, TC_VarIdentifier, FormalParIdentifier o ConsRef. */
651 VerdictId ::= **$VerdictId** [Verdict]
652 Verdict ::= Pass | Fail | Inconclusive | Result
/* SEMÁNTICA ESTÁTICA – No podrán producirse veredictos que correspondan a cualquiera de las siguientes inscripciones en el árbol de comportamiento: vacío, un constructivo ATTACH, un constructivo REPEAT, un constructivo GOTO, un IMPLICIT SEND o un RETURN.*/
653 Pass ::= **PASS** | **P** | "(" **PASS** ")" | "(" **P** ")"
654 Fail ::= **FAIL** | **F** | "(" **FAIL** ")" | "(" **F** ")"
655 Inconclusive ::= **INCONC** | **I** | "(" **INCONC** ")" | "(" **I** ")"
656 Result ::= **R** | **MTC_R**
/* SEMÁNTICA ESTÁTICA – No se utilizará R en el lado izquierdo de una asignación. */
/* SEMÁNTICA ESTÁTICA – MTC_R se utilizará R solamente en el MTC. */

A.3.3.33 Declaraciones TTCN

657 StatementLine ::= (Event [Qualifier] [AssignmentList] [TimerOps]) | (Qualifier [AssignmentList] [TimerOps]) | (AssignmentList [TimerOps]) | TimerOps | Construct | ImplicitSend
658 Event ::= Send | Receive | Otherwise | Timeout | Done
/* SEMÁNTICA ESTÁTICA – Un evento Receive, Otherwise o Timeout sólo será seguido por otros eventos Receive, Otherwise y Timeout a través del resto del conjunto de alternativas en un árbol totalmente expandido. En consecuencia, los árboles por defecto sólo contendrán eventos Receive, Otherwise y Timeout en el primer nivel de alternativas. */
659 Qualifier ::= "[" Expression "]"
/* SEMÁNTICA OPERACIONAL – Qualifier tomará un valor BOOLEAN específico. */
660 Send ::= [PCO_Identifier | CP_Identifier | FormalParIdentifier] "!" (ASP_Identifier | PDU_Identifier | CM_Identifier)
/* SEMÁNTICA ESTÁTICA – PCO_Identifier, CP_Identifier o FormalParIdentifier estarán presentes a menos que la serie de pruebas utilice solamente un PCO y ningún CP. */
/* SEMÁNTICA ESTÁTICA – FormalParIdentifier resolverá a un PCO_Identifier o CP_Identifier.*/
/* SEMÁNTICA ESTÁTICA – Solamente se pueden intercambiar CM en CP y solamente se pueden intercambiar ASP y PDU en PCO. */
661 ImplicitSend ::= "<" (**IUT** | PCO_Identifier | FormalParIdentifier) "!" (ASP_Identifier | PDU_Identifier) ">"
/* SEMÁNTICA ESTÁTICA – ImplicitSend no se utilizará a menos que el método de prueba que se emplee sea uno de los métodos de prueba a distancia. */
/* SEMÁNTICA ESTÁTICA – FormalParIdentifier se convertirá en un PCO_Identifier.*/
662 Receive ::= [PCO_Identifier | CP_Identifier | FormalParIdentifier] "?" (ASP_Identifier | PDU_Identifier | CM_Identifier)
/* SEMÁNTICA ESTÁTICA – PCO_Identifier, CP_Identifier o FormalParIdentifier estarán presentes a menos que la serie de pruebas utilice solamente un PCO y ningún CP. */
/* SEMÁNTICA ESTÁTICA – Solamente pueden intercambiarse CM en CP y solamente pueden intercambiarse ASP y PDU en PCO. */
/* SEMÁNTICA ESTÁTICA – FormalParIdentifier se convertirá en un PCO_Identifier o CP_Identifier.*/
663 Otherwise ::= [PCO_Identifier | CP_Identifier | FormalParIdentifier] "?" **OTHERWISE**
/* SEMÁNTICA ESTÁTICA – PCO_Identifier, CP_Identifier o FormalParIdentifier estarán presentes a menos que la serie de pruebas utilice solamente un PCO y ningún CP. */
/* SEMÁNTICA ESTÁTICA – FormalParIdentifier se convertirá en un PCO_Identifier o CP_Identifier.*/

664 Timeout ::= "?" **TIMEOUT** [TimerIdentifier | FormalParIdentifier]
/* STATIC SEMANTICS – FormalParIdentifier no será del tipo TIMER type. */
665 Done ::= "?" **DONE** "(" [TCompIdList] ")"
666 TCompIdList ::= TCompIdentifier {Comma TCompIdentifier}
667 Construct ::= GoTo | Attach | Repeat | Return | Activate | Create
668 Activate ::= **ACTIVATE** "(" [DefaultRefList] ")"
/* SEMÁNTICA ESTÁTICA – El constructivo ACTIVATE no se utilizará en cuadro de comportamiento por defecto. */
669 Return ::= **RETURN**
/* SEMÁNTICA ESTÁTICA – El constructivo RETURN no se utilizará salvo en árboles de comportamiento por defecto (incluyéndose algunos árboles locales dentro de cuadros de comportamiento por defecto). */
670 Create ::= **CREATE** "(" CreateList ")"
671 CreateList ::= CreateTComp {Comma CreateTComp}
672 CreateTComp ::= TCompIdentifier Colon TreeReference [ActualParList]
/* STATIC SEMANTICS – TCompIdentifier no será del Role MTC */
673 GoTo ::= ("**->**" | **GOTO**) Label
/* SEMÁNTICA ESTÁTICA – La columna de etiquetas contendrá las etiquetas a las que se hace referencia a partir del GoTo. */
/* SEMÁNTICA ESTÁTICA – Label deberá estar asociada con la primera de un conjunto de alternativas, una de las cuales un nodo predecesor del punto a partir del cual ha de efectuarse el GoTo. */
/* SEMÁNTICA ESTÁTICA – GoTo solamente se utilizará para saltos dentro de un árbol, a saber, un árbol raíz de caso de prueba, un árbol de paso de prueba, un árbol por defecto o un árbol local. Y de este modo, cada una de las etiquetas utilizadas en un constructivo GoTo deberán encontrarse dentro del árbol en el que se utilizó el GoTo. */
/* SEMÁNTICA ESTÁTICA – No habrá ninguna operación ACTIVATE como nodo predecesor del constructivo GoTo en la rama del árbol entre la Label y el GoTo. */
/* SEMÁNTICA ESTÁTICA – No se efectuará un GoTo al primer nivel de alternativas de árboles locales, pasos de prueba o valores por defecto. */
674 Attach ::= "+" TreeReference [ActualParList]
/* SEMÁNTICA ESTÁTICA – TreeReference no se adjuntará a sí misma, ni directa ni indirectamente, en su nivel máximo de sangrado. */
/* SEMÁNTICA ESTÁTICA – El número de parámetros reales será igual al número de parámetros formales. */
/* SEMÁNTICA ESTÁTICA – Los parámetros formales y reales de los pasos de prueba se utilizará de forma que sólo se cree una TTCN válida por substitución de texto. */
/* SEMÁNTICA ESTÁTICA – LiteralValue, TS_ParIdentifier, TS_ConstIdentifier, TS_VarIdentifier, TC_VarIdentifier, ConsRef, MatchingSymbol, FormalParIdentifier, PCO_Identifier y CP_Identifier pueden pasarse como parámetros reales a un árbol adjuntado. */
675 Repeat ::= **REPEAT** TreeReference [ActualParList] **UNTIL** Qualifier
/* SEMÁNTICA ESTÁTICA – TreeReference no se adjuntará a sí misma, ni directa ni indirectamente, en su nivel máximo de sangrado. */
/* SEMÁNTICA ESTÁTICA – El número de parámetros reales será igual al número de parámetros formales. */
/* SEMÁNTICA ESTÁTICA – LiteralValue, TS_ParIdentifier, TS_ConstIdentifier, TS_VarIdentifier, TC_VarIdentifier, ConsRef, MatchingSymbol, FormalParIdentifier, PCO_Identifier y CP_Identifier pueden pasarse como parámetros reales al árbol en un enunciado REPEAT. */
676 TreeReference ::= TestStepIdentifier | TreeIdentifier
/* SEMÁNTICA ESTÁTICA – TreeIdentifier será el nombre de uno de los árboles en la descripción de comportamiento vigente, es decir, los árboles locales no son accesibles fuera de la descripción de comportamiento en la que están especificados. */
677 ActualParList ::= "(" ActualPar {Comma ActualPar} ")"
/* SEMÁNTICA ESTÁTICA – El número de parámetros reales será igual al número de parámetros formales. */
/* SEMÁNTICA OPERACIONAL – Cada uno de los parámetros reales resolverán a un valor específico compatible con el tipo de su parámetro formal correspondiente o, en el caso de operaciones predefinidas, compatible con los tipos para los cuales se define la operación. */
/* SEMÁNTICA ESTÁTICA – Si un parámetro es una constricción parametrizada, la constricción se pasará junto con su lista de parámetros reales. */
/* SEMÁNTICA ESTÁTICA – Los parámetros reales tarán vinculados. */
/* SEMÁNTICA ESTÁTICA – Si el tipo del parámetro formal es PDU, el tipo del parámetro real será declarado como PDU o como un tipo **PDU** específico. */
678 ActualPar ::= Value | PCO_Identifier | CP_Identifier | TimerIdentifier
/* NOTA – Mediante Value, se puede alcanzar MatchingSymbol, TS_ParIdentifier, TS_ConstIdentifier, TS_VarIdentifier, TC_VarIdentifier, FormalParIdentifier o ConsRef. */

A.3.3.34 Expresiones

679 ConstantExpression ::= Expression
/* SEMÁNTICA ESTÁTICA – ConstantExpression no contendrá TS_Variables o TC_Variables y se convertirá en un valor constante. */
680 AssignmentList ::= "(" Assignment {Comma Assignment} ")"
681 Assignment ::= DataObjectReference ":=" Expression
/* SEMÁNTICA ESTÁTICA – Excepto en una definición de procedimiento o en una definición de codificación, el lado izquierdo de Assignment sólo resolverá a: TS_VarIdentifier, TC_VarIdentifier, referencia al campo de una variable o referencia a un parámetro ASP o campo de PDU que vaya a enviarse. */

/* SEMÁNTICA ESTÁTICA – Dentro de una definición de procedimiento de una TSOp o EncodingOp, el DataObject Identifier en el lado izquierdo de una asignación será un VarIdentifier. */
/* SEMÁNTICA ESTÁTICA – Una expresión no contendrá variables no acotadas. */
/* SEMÁNTICA OPERACIONAL – La expresión en el lado derecho de Assignment tomará un valor explícito del tipo del lado izquierdo. */

682 Expression ::= SimpleExpression [RelOp SimpleExpression]
/* SEMÁNTICA OPERACIONAL – Si existen SimpleExpressions y RelOp, las SimpleExpressions tomarán valores pecíficos de tipos compatibles. */
/* SEMÁNTICA OPERACIONAL – Si RelOp es "<" | ">" | >=" | "<=", cada una de las SimpleExpressions tomará un valor INTEGER específico. */
/* SEMÁNTICA ESTÁTICA – En las expresiones aritméticas no se utilizarán valores denominados de ASN.1 como operandos de operaciones. */

683 SimpleExpression ::= Term {AddOp Term}
/* SEMÁNTICA OPERACIONAL – Cada Term (término) resolverá a un valor específico. Si existe más de un Term, y si AddOp es "OR" los términos resolverán a tipo BOOLEAN; si AddOp es "+" o "-", los términos resolverán a tipo INTEGER. */

684 Term ::= Factor {MultiplyOp Factor}
/* SEMÁNTICA OPERACIONAL – Cada Factor resolverá a un valor específico. Si existe más de un Factor y MultiplyOp es "AND", los Factores resolverán a tipo BOOLEAN; si MultiplyOp es "*" o "/", los factores resolverán a tipo INTEGER. */

685 Factor ::= [UnaryOp] Primary
/* SEMÁNTICA OPERACIONAL – Primary resolverá a un valor específico. Si UnaryOp existe y es "NOT", Primary resolverá a tipo BOOLEAN; si UnaryOP es "+" o "-", Primary resolverá a tipo INTEGER. */

686 Primary ::= Value | DataObjectReference | OpCall | SelectExprIdentifier | "(" Expression ")"
/* SEMÁNTICA ESTÁTICA – SelectExprIdentifier sólo se utilizará dentro de expresiones de selección. */
/* NOTA – Mediante Value, se puede alcanzar MatchingSymbol, TS_ParIdentifier, TS_ConstIdentifier, TS_VarIdentifier, TC_VarIdentifier, FormalParIdentifier o ConsRef. */

687 DataObjectReference ::= DataObjectIdentifier {ComponentReference}
/* SEMÁNTICA ESTÁTICA – Se utilizarán identificadores de parámetros de ASP y campos de PDU asociados con SEND y RECEIVE solamente para hacer referenciar a valores de parámetro ASP y de campo PDU en la propia línea de enunciado. */
/* SEMÁNTICA ESTÁTICA – Cada ComponentReference sólo hará referencia a un parámetro ASP, campo de PDU, elemento de estructura o valor ASN.1 declarado explícitamente en el objeto que precede inmediatamente en la DataObjectReference. */
/* SEMÁNTICA ESTÁTICA – DataObjectIdentifier no será un VarIdentifier excepto en una definición de procedimiento o una TestSuiteOperation o EncodingOperation. */

688 DataObjectIdentifier ::= TS_ParIdentifier | TS_ConstIdentifier | TS_VarIdentifier | TC_VarIdentifier | FormalParIdentifier | ASP_Identifier | PDU_Identifier | CM_Identifier | VarIdentifier

689 ComponentReference ::= RecordRef | ArrayRef | BitRef
/* SEMÁNTICA ESTÁTICA – RecordRef se utilizará para hacer referencia a componentes SEQUENCE, SET y CHOICE en ASN.1. No se utilizará para hacer referencia a componentes de ningún otro tipo ASN.1. */
/* SEMÁNTICA ESTÁTICA – RecordRef se utilizará para referenciar parámetros ASP, campos PDU y elementos de estructura en forma tabular. */
/* SEMÁNTICA ESTÁTICA – ArrayRef se utilizará para referenciar componentes SEQUENCE OF y SET OF en ASN.1. No se utilizará para hacer referencia a componentes de ningún otro tipo ASN.1. */

690 RecordRef ::= Dot (ComponentIdentifier | ComponentPosition)
/* SEMÁNTICA ESTÁTICA – La forma ComponentIdentifier de RecordRef se utilizará siempre para hacer referencia a componentes SEQUENCE, SET y CHOICE de la ASN.1, cuando se haya declarado un identificador para el componente. */
/* SEMÁNTICA ESTÁTICA – La forma ComponentIdentifier de RecordRef se utilizará siempre para hacer referencia a parámetros de ASP, campos de PDU y elementos de estructura declarados en la forma tabular. */
/* SEMÁNTICA ESTÁTICA – La forma ComponentPosition de RecordRef se utilizará siempre para hacer referencia a componentes SEQUENCE, SET y CHOICE de la ASN.1, cuando no se haya declarado un identificador para el componente. */
/* SEMÁNTICA ESTÁTICA – StructIdentifier no se utilizará si la estructura pertinente se utiliza como un macro. StructIdentifiers y PDU_Identifiers se incluirán en una RecordRef cuando un parámetro, campo o elemento esté encadenado a una PDU o estructura y la RecordRef tenga que identificar un componente de esa PDU o estructura. */
/* SEMÁNTICA ESTÁTICA – Cuando una estructura se utilice como una expansión de macro, se hará referencia a los elementos de la estructura como si ésta hubiese sido expandida en la ASP o PDU referente a la misma. */
/* SEMÁNTICA ESTÁTICA – Si un parámetro, campo o elemento está definido de modo que sea de metatipo PDU, no se hará referencia a campos de esa subestructura. */

691 ComponentIdentifier ::= ASP_ParIdentifier | PDU_FieldIdentifier | CM_ParIdentifier | ElemIdentifier | ASN1_Identifier

692 ASN1_Identifier ::= Identifier
/* NOTA– ASN1_ Identifier identifica un campo dentro de un tipo SEQUENCE, SET o CHOICE de la ASN.1. */
/* SEMÁNTICA ESTÁTICA – No se utilizará un ASN1_Identifier asociado con un NamedValue (valor denominado) a menos que el valor se halle dentro de un tipo SEQUENCE, SET o CHOICE. */
/* SEMÁNTICA ESTÁTICA – Se proporcionará un ASN1_Identifier para identificar la variante en un tipo CHOICE. */
/* SEMÁNTICA ESTÁTICA – Se proporcionará un ASN1_Identifier cada vez que la definición de valor resulte ambigua por haberse omitido valores OPTIONAL en un tipo SEQUENCE. */

693 ComponentPosition ::= "(" Number ")"

694 ArrayRef ::= Dot "[" ComponentNumber "]"

695 ComponentNumber ::= Expression
/* SEMÁNTICA OPERACIONAL – ComponentNumber tomará a un valor INTEGER específico no negativo. */
696 BitRef ::= Dot (BitIdentifier | "[" BitNumber "]")
697 BitIdentifier ::= Identifier
/* NOTA – BitIdentifier identifica un bit particular dentro de una BIT STRING de la ASN.1. */
698 BitNumber ::= Expression
/* SEMÁNTICA OPERACIONAL – BitNumber tomará un valor INTEGER específico no negativo. */
699 OpCall ::= OpIdentifier (ActualParList | "(" ")")
/* SEMÁNTICA ESTÁTICA – Véase semántica estática en producción 699. */
700 OpIdentifier ::= TS_OpIdentifier | TS_ProcIdentifier | PredefinedOpIdentifier
701 PredefinedOpIdentifier ::= **BIT_TO_INT** | **HEX_TO_INT** | **INT_TO_BIT** | **INT_TO_HEX** | **IS_CHOSEN** | **IS_PRESENT** | **LENGTH_OF** | **NUMBER_OF_ELEMENTS**
702 AddOp ::= "+" | "-" | **OR**
/* SEMÁNTICA OPERACIONAL – Los operandos de los operadores "+" y "-" serán de tipo INTEGER (es decir, predefinidos en TTCN o ASN.1) o derivaciones de INTEGER (es decir, subgama). Los operandos del operador OR serán del tipo BOOLEAN (predefinidos en TTCN o ASN.1) o derivaciones de BOOLEAN. */
703 MultiplyOp ::= "*" | "/" | **MOD** | **AND**
/* SEMÁNTICA OPERACIONAL – Los operandos de los operadores "*", "/" y MOD serán del tipo INTEGER (es *decir*, predefinidos en TTCN o ASN.1) o derivaciones de INTEGER (es decir, subgama). Los operandos del operador AND serán del tipo BOOLEAN (predefinidos en TTCN o ASN.1) o derivaciones de BOOLEAN. */
704 UnaryOp ::= "+" | "-" | **NOT**
/* SEMÁNTICA OPERACIONAL – Los operandos de los operadores "+" y "-" serán del tipo INTEGER (es decir, predefinidos en TTCN o ASN.1) o derivaciones de INTEGER (es decir, subgama). Los operandos del operador NOT serán del tipo BOOLEAN (predefinidos en TTCN o ASN.1) o derivaciones de BOOLEAN. */
705 RelOp ::= "=" | "<" | ">" | "<>" | ">=" | "<="

A.3.3.35 Operaciones de temporizadores

706 TimerOps ::= TimerOp {Comma TimerOp}
707 TimerOp ::= StartTimer | CancelTimer | ReadTimer
708 StartTimer ::= **START** (TimerIdentifier | FormalParIdentifier) ["(" TimerValue ")"]
/* SEMÁNTICA ESTÁTICA – FormalParIdentifier sólo será del tipo TIMER. */
709 CancelTimer ::= **CANCEL** [TimerIdentifier | FormalParIdentifier]
/* SEMÁNTICA ESTÁTICA – FormalParIdentifier sólo será del tipo TIMER. */
710 TimerValue ::= Expression
/* SEMÁNTICA OPERACIONAL – Timervalue tomará un valor INTEGER positivo distinto de cero. */
711 ReadTimer ::= **READTIMER** (TimerIdentifier | FormalParIdentifier) "(" DataObjectReference ")"
/* SEMÁNTICA ESTÁTICA – FormalParIdentifier será solamente del tipo TIMER. */
/* SEMÁNTICA ESTÁTICA – La DataObjectReference sólo resolverá a TS_VarIdentifier, TC_VarIdentifier, o referencia al campo de una variable. */
/* SEMÁNTICA OPERACIONAL – La DataObjectReference sólo resolverá a tipo INTEGER. */

A.3.3.36 Tipos

712 TypeOrPDU ::= Type | **PDU**
713 Type ::= PredefinedType | ReferenceType

A.3.3.36.1 Tipos predefinidos

714 PredefinedType ::= INTEGER | BOOLEAN | BITSTRING | HEXSTRING | OCTETSTRING | OBJECTIDENTIFIER | R_Type | CharacterString
715 CharacterString ::= NumericString | PrintableString | TeletexString | VideotexString | VisibleString | IA5String | GraphicString | GeneralString | T61String | ISO646String | BMPString | UniversalString

A.3.3.36.2 Tipos referenciados

716 ReferenceType ::= TS_TypeIdentifier | ASP_Identifier | PDU_Identifier | CM_Identifier
/* SEMÁNTICA ESTÁTICA – Todos los tipos, con excepción de los tipos predefinidos, utilizados en una serie de pruebas, deberán ser declarados en las definiciones de tipo de caso de prueba, en las definiciones de tipo ASP, en las definiciones de tipo PDU o en las definiciones de tipo CM, y ser referenciados por nombre. */
717 TS_TypeIdentifier ::= SimpleTypeIdentifier | StructIdentifier | ASN1_TypeIdentifier

A.3.3.37 Valores

718 Value ::= LiteralValue | ASN1_Value [ASN1_Encoding]
/* REFERENCIA – Donde ASN1_Value es el Value no terminal, como está definido en la Rec. UIT-T X.680. A los efectos de la TTCN, en la Rec. UIT-T X.680 se define la siguiente producción:
DefinedValue ::= Externalvaluereference | valuereference | ParameterizedValue se redefine para que sea:
DefinedValue ::= ConstraintValue&Attributes | valuereference
Esto significa que en la TTCN no se admiten las referencias externas ASN.1, pero que sí están permitidas todas las posibilidades de ConstraintValue&Attributes que se definen en la producción 562 dentro de valores ASN.1 en la TTCN.

Ello quiere decir que se incluyen todas las expresiones, símbolos de concordancia, referencias de constricción, longitudes de valor, IF_PRESENT y operaciones de codificación de campo ASN.1.
A los efectos de la TTCN, las siguientes producciones en la Rec. UIT-T X.680:

BuiltinValue ::=
    BitStringValue |
    BooleanValue |
    CharacterStringValue |
    ChoiceValue |
    EmbeddedPDUValue |
    EnumeratedValue |
    ExternalValue |
    InstanceOfValue |
    IntegerValue |
    NullValue |
    ObjectClassFieldValue |
    ObjectIdentifierValue |
    OctetStringValue |
    RealValue |
    SequenceValue |
    SequenceOfValue |
    SetValue |
    SetOfValue |
    TaggedValue

ReferencedValue ::=
    DefinedValue |
    ValueFromObject

se redefinen para que sean:
BuiltinValue ::=
    BitStringValue |
    BooleanValue |
    CharacterStringValue |
    ChoiceValue |
    EmbeddedPDUValue |
    EnumeratedValue |
    ExternalValue |
    IntegerValue |
    NullValueValue |
    ObjectIdentifierValue |
    OctetStringValue |
    RealValue |
    SequenceValue |
    SequenceOfValue |
    SetValue |
    SetOfValue |
    TaggedValue

ReferencedValue ::=
    DefinedValue */
/* SEMÁNTICA ESTÁTICA – En expresiones aritméticas, no se utilizarán valores denominados de ASN.1 como operandos de operaciones. */
719 LiteralValue ::= Number | BooleanValue | Bstring | Hstring | Ostring | Cstring | R_Value
720 Number ::= (NonZeroNum {Num}) | **0**
721 NonZeroNum ::= **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**
722 Num ::= **0** | NonZeroNum
723 BooleanValue ::= **TRUE** | **FALSE**
724 Bstring ::= """ {Bin | Wildcard} """ **B**
725 Bin ::= **0** | **1**
726 Hstring ::= """ {Hex | Wildcard} """ **H**
727 Hex ::= Num | **A** | **B** | **C** | **D** | **E** | **F**
728 Ostring ::= """ {Oct | Wildcard} """ **O**
729 Oct ::= Hex Hex
730 Cstring ::= """ {Char | Wildcard | "\"} """
731 Char ::= /* REFERENCE – A character defined by the relevant CharacterString type. */
/* REQUISITO DE LÉXICO – Si el tipo CharacterString incluye el carácter " (comillas), este carácter será representado por un par de " (comillas) en la denotación de cualquier valor. */
732 Wildcard ::= AnyOne | AnyOrNone
733 AnyOne ::= "?"
/* SEMÁNTICA ESTÁTICA – AnyOne se utilizará solamente dentro de valores de tipo cadena, SEQUENCE OF y SET OF. */

734 AnyOrNone ::= "*"
/* SEMÁNTICA ESTÁTICA – AnyOrNone se utilizará solamente dentro de valores de tipo cadena, SEQUENCE OF y SET OF. */
735 R_Value ::= **pass** | **fail** | **inconc** | **none**
736 Identifier ::= Alpha{AlphaNum | Underscore | DoubleColon}
/* SEMÁNTICA ESTÁTICA – Todos los identificadores a los que se hace referencia en una serie de pruebas deberán ser declarados explícitamente en la serie de pruebas en TTCN, declarados explícitamente en una definición de tipo ASN.1 referenciada por la serie de pruebas o ser un identificador predefinido en TTCN. */
/* SEMÁNTICA ESTÁTICA – Los dos puntos dobles solamente se utilizarán en identificadores que están declarados en un cuadro de importaciones. Los identificadores que contengan los dos puntos dobles no deberán aparecer en un cuadro de exportaciones. Los dos puntos dobles se utilizan para separar el nombre de un módulo TTCN de un identificador especificado originariamente en ese módulo TTCN. */
737 Alpha ::= UpperAlpha | LowerAlpha
738 AlphaNum ::= Alpha | Num
739 UpperAlpha ::= **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** | **M** | **N** | **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V** | **W** | **X** | **Y** | **Z**
740 LowerAlpha ::= **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **j** | **k** | **l** | **m** | **n** | **o** | **p** | **q** | **r** | **s** | **t** | **u** | **v** | **w** | **x** | **y** | **z**
741 ExtendedAlphaNum ::= /* REFERENCE – A character from any character set defined in ISO/IEC 10646-1 */
742 BoundedFreeText ::= ″/*″ FreeText ″*/″
743 FreeText ::= {ExtendedAlphaNum}
/* REQUISITO DE LÉXICO – FreeText no contendrá la cadena "*/" a menos que vaya precedida por barra inclinada inversa ("\"). */

A.3.3.38 Producciones diversas

744 Comma ::= ","
745 Dot ::= "."
746 Dash ::= "-"
747 Minus ::= "-"
748 SemiColon ::= ";"
749 DoubleColon ::= Colon Colon
750 Colon ::= ":"
751 Underscore ::= "_"

A.4 Requisitos generales de semántica estática

A.4.1 Introducción

Los requisitos de semántica estática relacionados con producciones BNF concretas se especifican como comentarios sobre las producciones pertinentes, con el siguiente formato:

/* STATIC SEMANTICS – ... */

En el resto de A.4 se especifican los demás requisitos de semántica estática comunes a la TTCN.GR y la TTCN.MP. En A.5.2 se especifican las semánticas estáticas adicionales de la TTCN.MP.

A.4.2 Exclusividad de los identificadores

En algunos casos, las series de pruebas pueden hacer referencia a elementos definidos en otras normas OSI. En especial, en las definiciones de tipo pueden hacerse referencias a módulos de definición de tipo en ASN.1, acordes con la Rec. UIT-T X.680. En todas las series de pruebas Se pueden usar nombres derivados de estos módulos (tales como identificadores o subcampos dentro de las definiciones de tipos estructurados en ASN.1).

Las reglas para los identificadores en ASN.1 y en TTCN son contradictorias, por lo que se aplicarán los siguientes convenios:

- las referencias de tipo, los identificadores de módulo y las referencias de valor de los diversos cuadros de definiciones de tipo en ASN.1 cumplirán los requisitos establecidos para los identificadores en la Rec. UIT-T X.680;
- en los identificadores utilizados en otras partes de una serie de pruebas se sustituirán los caracteres guión (-) por caracteres de subrayado (_).

En algunos de los cuadros de TTCN se puede utilizar parte de la sintaxis ASN.1 para definir tipos. En esos casos, las reglas de ASN.1 irán seguidas de identificadores, con la salvedad de que no podrán utilizarse caracteres guión (-), debiendo emplearse en su lugar caracteres de subrayado (_). Cuando se utilice la ASN.1, se aplicarán a las series de pruebas de TTCN los demás requisitos definidos por la Rec. UIT-T X.680 (por ejemplo, los identificadores de tipo comenzarán con una letra mayúscula y los identificadores de campo dentro de las definiciones tructuradas en ASN.1 comenzarán con una letra minúscula).

Los identificadores de objetos de TTCN que se indican a continuación serán exclusivos a lo largo de la serie de pruebas:

a) tipos de serie de pruebas;
b) operaciones series de pruebas;
c) parámetros de series de pruebas;
d) expresiones de selección de casos de prueba;
e) constantes de series de pruebas;
f) variables de series de pruebas;
g) variables de caso de prueba;
h) tipos de PCO;

NOTA 1 – Si no existe cuadro de declaraciones de tipos de PCO, los tipos de PCO se declaran implícitamente en el cuadro de declaraciones de PCO, en cuyo caso la unidad se refiere al significado del tipo de PCO – el mismo tipo de PCO puede aparecer varias veces en el cuadro de declaraciones de PCO con el mismo significado.

i) puntos de control y observación (PCO);
j) puntos de coordinación (CP);
k) temporizadores;
l) componentes de prueba;
m) configuraciones de componentes de prueba;
n) tipos de ASP;
o) tipos de PDU;
p) tipos de CM
q) tipos estructurados;
r) reglas de codificación;
s) variaciones de codificación;
t) codificaciones de campo no válidas;
u) alias;
v) constricciones de ASP;
w) constricciones de PDU;
x) constricciones de CM;
y) constricciones de estructura;
z) casos de prueba;
aa) pasos de prueba;
ab) valores por defecto.
ac) nombres de reglas de codificación;
ad) nombres de variaciones codificación;
ae) nombres de codificaciones de campo no válidas.

Las referencias de objetos de TTCN que se indican a continuación, serán exclusivas a lo largo de la serie de pruebas:

a) referencias de grupo de prueba;
b) referencias de grupo de pasos de prueba;
c) referencias de grupo de valores por defecto.

En el cuadro A.2 se indican las palabras reservadas de la TTCN. Estas palabras reservadas no podrán utilizarse como identificadores en una serie de pruebas de TTCN. Las palabras reservadas de la TTCN y los identificadores de la TTCN son sensibles al tipo de letras, mayúscula o minúscula.

Cuadro A.2/X.292 – Palabras reservadas de la TTCN

ACTIVATE	IF	PDU
AND	IF_PRESENT	PERMUTATION
BEGIN	INCONC	PrintableString
BITSTRING	inconc	ps
BIT_TO_INT	INFINITY	PTC
BOOLEAN	INTEGER	R
BY	INT_TO_BIT	READTIMER
CANCEL	INT_TO_HEX	REPEAT
CASE	IS_CHOSEN	REPLACE
COMPLEMENT	IS_PRESENT	RETURN
CP	IUT	RETURNVALUE
CREATE	LT	R_Type
DO	min	s
DONE	MOD	START
ELSE	ms	STATIC
ENC	MTC	SUPERSET
END	MTC_R	SUBSET
ENDCASE	NOT	TeletexString
ENDIF	ns	THEN
ENDVAR	OF	TIMEOUT
ENDWHILE	OMIT	TIMER
F	OR	TO
FAIL	OTHERWISE	TRUE
fail	P	UNTIL
FALSE	LENGTH_OF	us
GeneralString	none	UT
GOTO	NUMBER_OF_ELEMENTS	VAR
GraphicString	NumericString	VideotexString
HEXSTRING	OCTETSTRING	VisibleString
HEX_TO_INT	OBJECTIDENTIFIER	WHILE
I	PASS	
IA5String	pass	

En el cuadro A.3, se indican las palabras reservadas de la ASN.1. Estas palabras reservadas no podrán utilizarse como identificadores en una serie de pruebas en TTCN.

Cuadro A.3/X.292 – Palabras reservadas en la ASN.1

ABSENT	EXTERNAL	OPTIONAL
ABSTRACT-SYNTAX	FALSE	PDV
ALL	FROM	PRESENT
APPLICATION	GeneralString	PRIVATE
AUTOMATIC	GeneralizedTime	PrintableString
BEGIN	GraphicString	REAL
BIT	IA5String	SEQUENCE
BMPString	IDENTIFIER	SET
BOOLEAN	IMPLICIT	SIZE
CHARACTER	IMPORTS	STRING
CHOICE	INCLUDES	SYNTAX
CLASS	INSTANCE	T61String
COMPONENT	INTEGER	TRUE
COMPONENTS	INTERSECTION	TeletexString
CONSTRAINED	ISO646String	TYPE-IDENTIFIER
DEFAULT	MAX	UNION
DEFINITIONS	MIN	UNIQUE
EMBEDDED	NULL	UNIVERSAL
END	NumericString	UniversalString
ENUMERATED	OBJECT	UTCTime
EXCEPT	ObjectDescriptor	VideotexString
EXPLICIT	OCTET	VisibleString
EXPORT	OF	WITH

NOTA 2 – En el cuadro A.3 se indica un cierto número de palabras clave que en el momento actual no se recogen en esta norma. Estas palabras se han reservado para facilitar la integración ulterior a la notación TTCN de funciones en ASN.1 1994.

Cuando en una serie de pruebas TTCN se usa ASN.1, los identificadores de la lista siguiente serán únicos en toda la serie de pruebas, independientemente de si la definición ASN.1 es explícita o implícita por referencia:

a) *TypeIdentifiers* de una definición de tipo ASN.1.

b) Identificadores que aparecen en un tipo ASN.1 ENUMERATED como valores distinguidos.

c) Identificadores que aparecen en *NamedNumberList* de un tipo ASN.1 INTEGER.

Los nombres de los parámetros ASP serán únicos dentro de la ASP en la que se declaran. Los nombres de los campos PDU serán únicos dentro de la PDU en la que se declaran. Los nombres de los parámetros CM serán únicos dentro del CM en el que se declaran.

Cuando se usa Structured Type como una extensión de macro, los nombres de los elementos dentro del tipo estructurado serán únicos en cada ASP, PDU o CM donde se expandirán.

Las etiquetas usadas dentro de un árbol serán únicas dentro del árbol (por ejemplo, el árbol raíz Caso de prueba, el árbol Paso de prueba, el árbol Valores por defecto, el árbol local).

El identificador de encabezamiento del árbol que se usa en los árboles locales será único dentro de la descripción de comportamiento dinámico en la que aparecen, y no será el mismo que cualquier identificador que tiene un significado único en la serie de pruebas.

NOTA 3 – Esto significa que un identificador de árbol local puede tener el mismo nombre que un identificador de árbol local de otra descripción de comportamiento, pero no el mismo nombre que el de otro Paso de prueba de la Biblioteca de pasos de prueba.

Los nombres de los parámetros formales que pueden aparecer optativamente como parte de lo siguiente serán únicos dentro de la lista de parámetros formales, y no serán los mismos que cualquier identificador que tiene un significado único en la serie de pruebas:

a) Definición de operaciones de serie de pruebas.

b) Encabezamiento de árbol de un árbol local.

c) Identificador de Paso de prueba.

d) Identificador de Valores por Defecto.

e) Declaración de constricción parametrizada.

Un nombre de parámetro local que figura en la lista de parámetros locales de un encabezamiento de árbol local tendrá precedencia con respecto a un nombre de parámetro local que figura en la lista de parámetros formales de Paso de prueba en el que se define, dentro del alcance de esa lista de parámetros formales locales.

En la TTCN concurrente, los PCO y CP que se usan en Caso de prueba serán sólo los determinados por la configuración de componente de prueba para ese caso de prueba.

Cada identificador usado en la definición de procedimiento de una operación de serie de pruebas será uno de los siguientes:

a) nombre de variable declarada localmente;

b) nombre de tipo usado en una declaración de variable;

c) nombre de parámetro formal declarado en la lista de parámetros formales de la operación;

d) nombre de la operación serie de pruebas.

El alcance de los nombres de parámetro formal y de los nombres de variable declarada localmente es la definición de procedimiento de la operación serie de pruebas. Por consiguiente, los valores de todos los demás tipos de identificador no están directamente accesibles dentro de la definición de procedimiento de una operación serie de pruebas. Para acceder a esos valores, se los transferirá directamente a la operación serie de pruebas como parámetros reales.

Las constricciones para tipos estructurados en TTCN, ASP en TTCN, PDU en TTCN y CM en TTCN no se especificarán mediante cuadros ASN.1 (es decir, constricciones de tipo en ASN.1, constricciones de ASP en ASN.1, constricciones de PDU en ASN.1 o constricciones de CM en ASN.1). Inversamente, las constricciones para tipos en ASN.1, ASP en ASN.1, PDU en ASN.1 y CM en ASN.1 no se especificarán con cuadros TTCN (es decir, constricciones de tipo estructurado, constricciones de ASP en TTCN, constricciones de PDU en TTCN o constricciones de CM en TTCN).

NOTA 4 – Sin embargo, cuando se encadenan ASP o PDU a otras PDU, las ASP o PDU adjuntantes pueden, por ejemplo, especificarse en TTCN tabular, mientras que las PDU adjuntadas se pueden especificar en ASN.1.

A.5 Diferencias entre TTCN.GR y TTCN.MP

A.5.1 Diferencias de sintaxis

A continuación se presenta una lista de diferencias de sintaxis entre TTCN.MP y TTCN.GR:

a) TTCN.MP utiliza palabras clave como delimitadores entre inscripciones, en tanto que TTCN.GR utiliza casillas.

b) TTCN.MP utiliza una denotación explícita de niveles de sangrado para eventos de prueba, mientras que el sangrado se indica de forma visual en TTCN.GR.

c) TTCN.MP contiene una ocurrencia adicional del identificador de la serie, que se utiliza para facilitar la identificación de la ATS en un método automatizado.

d) En TTCN.MP, las descripciones de comportamiento de caso de prueba se agrupan de forma explícita mediante la inclusión de identificadores de grupo de prueba apropiados de forma secuencial, antes de las descripciones de comportamiento de caso de prueba pertenecientes a cada grupo. Esta información duplica la información contenida en el Índice de casos de pruebas y en las referencias de grupo de prueba de las descripciones de comportamiento de caso de prueba.

e) La estructura de series de pruebas, el Índice de casos de prueba, el Índice de pasos de prueba y los cuadros de índices de valores por defecto requieren un número de página para cada entrada. Tales números de página son irrelevantes en la forma procesable por máquina, por lo que no se reflejan en la TTCN.MP.

f) TTCN.GR admite formularios simples y formularios compactos para constricciones de ASP y de PDU y casos de prueba. TTCN solamente admite BNF para el formato de cuadro único y la presentación de cierto número de cuadros únicos en formato compacto de TTCN.GR es una cuestión de visualización. Cuando se establezca la correspondencia entre un cuadro de constricciones compacto y la TTCN.MP (es decir, formato único) deberán omitirse los campos en blanco originados por la modificación.

g) Los símbolos "/*" y "*/" que abren y cierran cadenas BoundedFreeText en TTCN.MP no aparecen en TTCN.GR.

h) En TTCN.GR hay dos posiciones alternativas para la columna de etiquetas en los cuadros de descripción de comportamiento, en tanto que en TTCN.MP hay una sola posición fija para las etiquetas.

i) La continuación de página y de línea son peculiaridades de TTCN.GR que no son representadas en TTCN.MP.

j) La numeración de página y línea son peculiaridades de TTCN.GR que no son representados en TTCN.MP.

k) Si se utilizan referencias de grupo TTCN.GR con definiciones, declaraciones o constricciones para indicar una agrupación de objetos jerárquica, cada identificador de grupo pertinente en la TTCN.MP se inserta antes de la sintaxis para el grupo de cuadros que comparten ese identificador de grupo, y la sintaxis para el identificador de grupo y el siguiente grupo de cuadros se encierran entre las palabras clave TTCN.MP apropiadas, correspondientes al tipo de objeto.

A.5.2 Semántica estática adicional en la TTCN.MP

A continuación se indica la semántica estática adicional en la TTCN.MP:

a) En la TTCN.MP, los enunciados del primer nivel de alternativas que no tengan predecesor en el árbol raíz o local al que pertenecen tendrán el valor de sangrado de cero. Los enunciados que tengan un predecesor, tendrán un valor de sangrado igual al valor de sangrado del predecesor aumentado en uno.

b) En la TTCN.MP, la información de estructura de serie de pruebas aparece en forma de identificadores de grupos de prueba que preceden a las descripciones de comportamiento de casos de prueba y tendrán la misma estructura que la definida por la parte de la estructura de serie de pruebas aplicable a los grupos de prueba y definida por el Índice de casos de prueba.

A.6 Lista de números de producción en BNF

Vacía.

Anexo B

Semántica operacional de la TTCN

(Este anexo es parte integrante de la Recomendación)

B.1 Introducción

En el anexo A se describe la sintaxis de TTCN mediante las reglas de producción BNF y restricciones impuestas a estas producciones, cuyo cumplimiento se puede verificar estática y dinámicamente.

En este anexo se define la semántica de TTCN y se describre un procedimiento abstracto que ejecuta series de pruebas TTCN válidas sintácticamente. El procedimiento arranca, para cada caso de prueba, una "máquina de TTCN" abstracta que evalúa estos casos de prueba por medio de la creación, expansión e interpretación de un "árbol de evaluación", ocupándose al mismo tiempo de un nivel (conjunto ordenado de alternativas en una determinada posición en el árbol) a la vez. En la ejecución de la TTCN concurrente, se arrancan máquinas de TTCN adicionales, una para cada PTC creado. Estas máquinas trabajan de igual modo que la máquina TTCN principal, que está entonces ejecutando el componente de prueba principal. Se supone que los PCO y CP necesarios, que conectan las máquinas de TTCN con sus entornos y con cada uno de los demás, existen ya y están inicialmente vacíos.

El procedimiento abstracto (EVALUATE_TEST_SUITE) y las máquinas de TTCN (EVALUATE_TEST_CASE, EVALUATE_TEST_COMPONENT) se describen en la cláusula B.5. El árbol de evaluación tiene la forma de un árbol de comportamiento de TTCN, pero enriquecido por componentes adicionales. En una máquina de TTCN se fija inicialmente el caso de prueba indicado o árbol raíz de caso de prueba, o árbol local. Durante la ejecución de casos de prueba, el árbol de evaluación se amplía, y el "control" generalmente degrada el árbol de evaluación, salvo en la ejecución de GOTO y RETURN, en cuyo caso el control produce una mejora.

Los componentes adicionales del árbol, introducidos por motivos técnicos, son los siguientes: cada nodo (alternativa) tiene, además de la StatementLine denotada, un valor booleano IsDefault, que dice si el nodo se deriva de un cuadro de comportamiento por defecto. Cada nivel tiene, además de la lista denotada de StatementLines, un valor booleano IsExpanded, que indica si el nivel ya ha sido expandido.

No es necesario construir una máquina de TTCN real de manera que trabaje internamente del mismo modo exactamente que una máquina abstracta. La semántica operacional de la TTCN define solamente cómo debe comportarse externamente una máquina de TTCN real, es decir, en relación con las colas de PCO y CP, temporizadores y lista de temporizadores, e información de terminación de componentes de prueba. Los detalles de la implementación no son importantes.

B.2 Precedencia

En las cláusulas que siguen se describe la semántica operacional de la TTCN en una combinación de seudocódigo y lenguaje natural. Cuando ambas notaciones se superpongan, sus significados deben ser idénticos. Si el seudocódigo y el lenguaje natural son contradictorios, se trata de un error que debe notificarse a la organización de normalización mediante un informe de defectos. En tal caso, el seudocódigo tendrá precedencia sobre el lenguaje natural, a reserva de la corrección del defecto que efectúe la organización de normalización.

B.3 Procesamiento de errores de caso de prueba

En el texto principal de esta Recomendación, así como en el anexo A y en el presente anexo, se describen condiciones que dan por resultado la detección de errores de caso de prueba. La observación de un error de caso de prueba deberá inscribirse en el registro cronológico de conformidad y producir el aborto del caso de prueba.

Sin que se señale explícitamente en lo que sigue, un error de caso de prueba se detecta siempre dinámicamente si ninguna de las partes de una expresión toma un valor definido. Las expresiones son evaluadas, entre otras ocasiones, en la aplicación de asignaciones, calificadores y constricciones.

B.4 Conversión de una serie de pruebas modularizada a una serie de pruebas expandida equivalente

Este algoritmo no trata casos de error. Requiere que los objetos sean únicos en el ámbito en el que son definidos y utilizados.

En la conversión de una serie de pruebas modularizada a una serie de pruebas expandida, es necesario redenominar algunos objetos TTCN importados (con el fin de evitar conflictos de nombre). En este proceso de redenominación se admiten dos opciones:

a) se retiene el nombre original como se define en la declaración/definición del objeto;

b) se construye el nuevo nombre por concatenación del identificador de módulo y el nombre original del objeto. Ambos estarán separados por dos signos de subrayado, por ejemplo, ModuleA_ _ConnectionRequest.

El principio de este algoritmo consiste, para cada objeto fuente, en la realización de una copia temporal del mismo, la expansión de esta copia, la marcación a continuación de cada uno de los objetos que han de ser importados y, finalmente, la fusión de cada objeto marcado dentro de la serie de importación.

En la expansión de las fuentes importadas, todos los objetos importados explícita e implícitamente se redenominan como Module::Identifier, si no han sido antes redenominados en importación. Cada módulo tendrá un identificador exclusivo. En la serie de pruebas expandida todos los objetos importados explícita e implícitamente son reconocibles con facilidad, y como cada módulo ha de tener un nombre exclusivo no existe posibilidad de que se produzcan conflictos de nombre.

B.5 Semántica operacional de la TTCN

B.5.1 Introducción

Los árboles de comportamiento en la TTCN son evaluados en un nivel de alternativas al mismo tiempo. En cada nivel, se añaden valores por defecto, se expanden constructivos de adjunción, y se sustituyen constructivos REPEAT. Esto produce un conjunto de alternativas que pueden evaluarse para averiguar cual de ellas concuerda exitosamente y determina por tanto el conjunto de alternativas para pasar al siguiente. Los requisitos para la concordancia de un enunciado en TTCN dependen de lo que está codificado en esa línea de comportamiento y se describen en este texto de semántica.

B.5.2 Notación de seudocódigo

B.5.2.1 Introducción

La semántica de la TTCN se define con un procedimiento funcional simple, que explica la ejecución de una descripción de comportamiento de caso de prueba en TTCN, incluida la expansión por pasos de un árbol de evaluación, y la ejecución de los nodos de ese árbol. Tales funciones se prevén para ayudar a comprender la semántica de la TTCN, y no se pretende asociarlas con ningún modelo particular de ejecución o lenguaje de programación de alto nivel. No tienen por objeto ser métodos directos de ejecución de la TTCN.

Las palabras clave de seudocódigo se imprimen en negritas, por ejemplo, **procedure**, **function**, **begin**, **end**, **if**, **then**, **else**. En el encabezamiento de sus definiciones, procedimientos, procesos y funciones, los nombres son resaltados en negritas para facilitar su búsqueda. Por la misma razón se resalta el tipo de los datos de un función. Por otra parte, los tipos de datos no se tratan de manera explícita.

B.5.2.2 Procedimientos y funciones

Muchos enunciados son llamadas a **procedimiento** (**procedure**). Se pueden usar las expresiones función (function) siempre que se necesite un valor del tipo asociado. Ellas obtienen su valor (y son inmediatamente terminadas) mediante **return**, seguido de una expresión del valor.

Los parámetros de procedure y de function son en general "parámetros globales", es decir, parámetros formales que pueden ser tanto de "lectura" como de "escritura". En especial, las funciones pueden producir "efectos secundarios" y son esencialmente "procedimientos con un valor". Las variables de un cuerpo de procedimiento o de función que no son parámetros formales ni alguno de los parámetros globales mencionados anteriormente, son variables locales de este cuerpo, sin declaración explícita.

Hay que adoptar precauciones para que:

- los parámetros sean de lectura solamente cuando tienen un valor definido;
- los términos se utilicen como parámetros reales sólo en el caso de que el procedimiento o función no asigne un valor al parámetro formal respectivo, es decir que el parámetro sea puramente un parámetro entrada.

B.5.2.3 Procesos

Los **procesos** tienen un comportamiento similar a los procedimientos, excepto en que cada uno de ellos se ejecuta en una máquina de TTCN independiente. Los procesos no se ejecutan en forma anidada. En un proceso se pueden declarar los objetos de datos globales, de modo que se encuentren disponibles en todos los procedimientos y funciones llamados en el proceso sin que sean explícitamente transferidos como parámetros. Al evitar listas de parámetros largas, el seudocódigo

es más fácil de leer. Por supuesto, existen instancias de objetos globales independientes en cada proceso (máquina de TTCN). No existe ninguna relación entre los objetos globales de procesos diferentes.

En este anexo se tratan los objetos mencionados a continuación como objetos globales en cada proceso:

- EvaluationTree (árbol de evaluación), del caso de prueba (o componente de prueba principal) o componente de prueba paralelo.
- CurrentLevel (nivel vigente), que será expandido o puesto en concordancia.
- Defaults (valores por defecto), contexto por defecto vigente, utilizado en la expansión por defecto.
- Instantánea, visión fija temporal del entorno.
- ReturnLevel (nivel de retorno), que se considerará después de la ejecución del enunciado RETURN.
- ReturnDefaults (valores por defecto de retorno), contexto por defecto del ReturnLevel.
- SendObject, la ASP, la PDU o el CM que han de enviarse seguidamente.
- ReceiveObject, la ASP, la PDU o el CM último recibido.

De este modo, cada máquina de TTCN tendrá su propio árbol de evaluación, etc.

Otros objetos son, sin embargo, accesibles desde todos los procesos. El estado pertinente del "entorno de EVALUATE_TEST_SUITE", es decir, se supone que el contenido de los PCO y CP pertinentes, así como las listas de temporizadores que han expirado, los valores de los temporizadores y la lista de los componentes de prueba paralelos terminados, son globalmente accesibles desde todos los componentes de prueba y no es necesario transferirlos explícitamente como parámetros. Análogamente, se supone que los parámetros de serie de pruebas, las constantes de serie de pruebas y las variables de serie de pruebas son accesibles desde todos los procesos de componentes de prueba o casos de prueba.

B.5.2.4 Lenguaje natural dentro de seudocódigo

Para hacer menos complejo este anexo, algunas partes de seudocódigo se escriben en lenguaje natural. Estas partes se encierran entre los signos /# y #/. Tales partes representan enunciados, detalles "for-loop" o expresiones de seudocódigo, y se supone que serán ejecutadas o evaluadas cuando se presenten.

Los comentarios puros, destinados a la persona que lee, y no para ser ejecutados o evaluados por una máquina de TTCN, se encierran entre (* y *).

B.5.2.5 Niveles y alternativas

Un nivel visitado en un árbol denota tanto una posición en el árbol como el conjunto ordenado de alternativas en este nivel.

Una alternativa visitada en un árbol determina una posición de nivel en el árbol, véase LEVEL_OF en B.5.25. La alternativa denota simultáneamente un posición en ese nivel, una línea de comportamiento, una línea de enunciado, etc.

Así, los niveles y las alternativas en un árbol son punteros, pero el desempaquetado de los datos a los que señalan se realiza implícitamente.

B.5.3 Ejecución de un caso de prueba

B.5.3.1 Introducción

La serie de pruebas se ejecuta en el procedimiento principal, EVALUATE_TEST_SUITE. Cada componente de prueba principal (caso de prueba en el caso no concurrente) se ejecuta en una máquina de TTCN abstracta mediante la ejecución de EVALUATE_TEST_CASE. Cada componente de prueba paralelo se ejecuta en una máquina de TTCN independiente, ejecutando EVALUATE_TEST_COMPONENT.

```
procedure EVALUATE_TEST_SUITE(TestSuiteId)
        (* Este procedimiento introduce nombres únicos para todos los árboles de la TTCN, incluidos los subárboles locales.
           Establece objetos de datos específicos de la serie de pruebas y evalúa cada caso de prueba cuyas expresiones de
           selección toman el valor TRUE. *)
begin
        for /# every Test Case, Test Step or Default behaviour table Table in TestSuiteId #/ do
        begin
          /#    Rename all local trees of Table such that they become unique throughout the test suite and different from any Test
                Case,
                Test Step or Default behaviour table name in the Test Suite. #/;
          /#    Rename accordingly in Table all references to local trees in attachments. #/;
          /#    Every node in every behaviour tree gets a new Boolean component "IsDefault".
                This component is set to TRUE for all nodes in Default Dynamic Behaviour Tables
                and FALSE for all nodes in all other tables. #/;
        end;
        for /# every Default behaviour table Table in TestSuiteId #/ do
```

```
        begin
          /#      For each leaf of the behaviour tree which does not have an entry in the verdict column assign the verdict R. #/
          /#      or each leaf of the behaviour table which has a preliminary result assigned, change the preliminary result to a
                  verdict by removing the parentheses around it. #/
        end;
        Evaluated := /# empty list of Test Case Identifiers #/;
        /#  Set values of Test Suite Parameters, Test Suite Constants, and, where to be initialized, of Test Suite Variables #/;
        for /# every Test Case Identifier TCId of TestSuiteId that is not yet in Evaluated #/ do (* in any order *)
        begin
          SelEx := /# conjunction of the selection expressions of all test groups containing Test Case TCId (directly or via lower
          groups) #/;
          if EVALUATE_BOOLEAN(SelEx) then
                start process EVALUATE_TEST_CASE (TCId);
          /# add TCId to the list Evaluated #/;
        end
end
```

B.5.4 Ejecución de un caso de prueba

B.5.4.1 Ejecución de un caso de prueba – Seudocódigo

```
process EVALUATE_TEST_CASE(TestCaseId)
        (*  Este proceso inicializa el árbol de evaluación por el árbol raíz de caso de prueba y el contexto por defecto por las
            referencias por defecto relacionadas con la descripción de comportamiento de caso de prueba. Traslada el control al nivel
            de alternativas superior y pide su evaluación. *)
global EvaluationTree, CurrentLevel, Defaults, Snapshot, ReturnLevel, ReturnDefaults, SendObject, ReceiveObject;
begin
        /#  Initialize Test Case Variables, global R and MTC_R, PCOs, CPs, Timers, and the Timeout List of TestCaseId. #/;
        EvaluationTree := ROOT_TREE(TestCaseId);
          (*  El árbol de evaluación es un árbol finito en crecimiento construido por copias, expandidas y pegadas juntas, de árboles
              procedentes de la descripción de comportamiento de caso de prueba y de las bibliotecas de casos de prueba y valores
              por defecto. A cada nivel se añade un componente IsExpanded. *)
          CurrentLevel := FIRST_LEVEL(EvaluationTree) ;
          (*  Un nivel denota tanto una posición en un árbol como el conjunto ordenado de alternativas en esta posición. *)
          ReturnLevel := CurrentLevel;
          Defaults := DEF_REF_LIST(TestCaseId);
          ReturnDefaults := Defaults;
          EVALUATE_LEVELS ();
          (*  Incluye, mediante llamadas anidadas, la evaluación de todos los niveles subsiguientes pertinentes en el árbol de
              evaluación en crecimiento. *)
end

procedure EVALUATE_LEVELS ()
        (*  Este procedimiento, en primer lugar expande y evalúa el CurrentLevel, que es el nivel de alternativas activo en ese
            momento del árbol de evaluación. Los valores por defecto dan en contexto por defecto activo en ese momento. Las
            alternativas contenidas en CurrentLevel se procesan según su orden de aparición, y si es necesario de manera cíclica.
            CurrentAlternative es la variable de bucle del "for-loop", que denota la alternativa considerada en ese momento en
            CurrentLevel. Mediante el mecanismo de instantánea, en cada ciclo de tentativas de concordancia a través de
            CurrentLevel, el estado del entorno considerado no cambia, lo que da a cada uno de tales ciclos un carácter instantáneo.
            Excepto para los errores de casos de prueba detectados dinámicamente, la evaluación de CurrentLevel incluye la
            evaluación con éxito de una alternativa. Se continúa con la asignación de un veredicto y la evaluación del siguiente
            nivel, y, en consecuencia, por inducción, con la evaluación de todos los niveles a los que el control cambia
            subsiguientemente. *)
begin
          if NOT  IS_EXPANDED() then
            (*  Mediante esta condición evitamos la expansión repetida de niveles que son objetivos de los GOTO. *)
            EXPAND_CURRENT_LEVEL ();
          (*  El nivel vigente está ahora libre de REPEAT y adjunciones, e incluye los valores por defecto necesarios. *)
          repeat
            (*  ... la realización de ciclos durante el nivel vigente, intentando concordar una alternativa.*)
            TAKE_SNAPSHOT();
            (*  ... de la cola o las colas de PCO y CP entrantes, la lista de temporizaciones pertinente y el estado de terminación de
                cualquier otro de los componentes de pruebas. *)
            for /# every CurrentAlternative in CurrentLevel, in the given order #/          do
            (*  intentar concordar la alternativa vigente. Obsérvese que una alternativa visitada en un árbol determina una posición
                de nivel en el árbol y denota, segúnl contexto en el cual es utilizado, una posición en ese nivel, una línea de
                comportamiento, una línea de enunciado, etc. *)
          begin
            if EVALUATE_EVENT_LINE (CurrentAlternative) then
            (*  En ausencia de errores de caso de prueba el componente de prueba o caso de prueba determinará dentro de la
                llamada EVAL_VERDICT_ENTRY o GOTO_NEXT_LEVEL_OR_STOP_WITH_VERDICT de la instancia
```

```
            recursiva más interna de EVALUATE_LEVELS, por ejemplo, si hay un veredicto final o no hay ningún nivel
            siguiente. Entonces, el "for-loop" será abortado, también. *)
        begin
          if /# Alternative has a verdict column entry VerdictEntry #/ then
            EVAL_VERDICT_ENTRY(VerdictEntry);
          GOTO_NEXT_LEVEL_OR_STOP_WITH_VERDICT(CurrentAlternative);
          EVALUATE_LEVELS();
        end
      end
      until SNAPSHOT_FIXED();
      (*   SNAPSHOT_FIXED devuelve TRUE si la instantánea no puede ya cambiar. *)
      LOG(TEST_CASE_ERROR);
      STOP_TEST_CASE();
end
```

B.5.4.2 Ejecución de un caso de prueba o componente de prueba – Descripción en lenguaje natural

Paso 1 La evaluación comienza al nivel de sangrado numéricamente más bajo (en la TTCN.MP), esto es, el situado más a la izquierda (en la TTCN.GR), del árbol raíz.

Paso 2 Expansión del nivel vigente para incluir todos los valores por defecto explícitamente, y reemplazar todas las adjunciones de árbol, si es necesario, así como todos los REPEAT, por sus expansiones.

Paso 3 Tomar una instantánea de la cola o colas del PCO y CP de entrada y la lista de temporizaciones.

NOTA 1 – El acto de toma de la instantánea no elimina ningún evento del PCO o el CP.

Considerar la primera línea de comportamiento en el nivel vigente de alternativas.

Paso 4 Se evalúa el enunciado en TTCN de la línea de comportamiento vigente.

La evaluación de cada tipo de enunciado en TTCN se especifica en la semántica operacional correspondiente a ese tipo de enunciado en TTCN.

Paso 5 Si el enunciado en TTCN produce como evaluación una concordancia correcta, se va al paso 6.

Si no es así y hay más de una alternativa en el conjunto vigente de alternativas, se considera la línea siguiente de comportamiento del conjunto de alternativas y se va al paso 4.

Si no hay más alternativas, pero todas las colas de PCO y CP pertinentes a este conjunto de alternativas contienen todavía al menos un evento y todos los temporizadores asociados a enunciados de temporización del conjunto de alternativas están en la lista de temporizaciones, se detiene el paso de prueba y se indica *error de caso de prueba*.

NOTA 2 – En estas condiciones, no puede concordar ninguna alternativa del conjunto.

Para los demás casos – es decir, no hay más alternativas y la instantánea siguiente puede mostrar una imagen diferente – se va al paso 3.

Paso 6 Si se codifica un veredicto preliminar, se procede como en B.5.23.2.

Paso 7 Si se ha alcanzado un nodo hoja del árbol, se va al paso 8.

En cualquier otro caso, se determina y considera para su evaluación el siguiente nivel de alternativas y se va al paso 2.

Paso 8 Se utiliza el veredicto final, o, si no se especifica, el valor vigente de la variable de resultado preliminar R, como veredicto final del caso de prueba, al igual que en B.5.23.2 y B.5.25.

B.5.5 Expansión de un conjunto de alternativas

B.5.5.1 Introducción

En esta subcláusula se define el modo de expandir un conjunto de alternativas en preparación para evaluar qué alternativa concuerda.

Se lleva a cabo en cuatro pasos:

a) almacenamiento del contexto por defecto si el nivel está etiquetado;

b) adjunción del conjunto vigente de árboles de comportamiento por defecto;

c) adjunción de los árboles adjuntados, si es necesario, de manera recursiva, hasta que no haya más alternativas de adjunción en el conjunto;

d) expansión de constructivos REPEAT, reemplazándolos por un subárbol en el cual las adjunciones de árbol y los constructivos GOTO ocurran solamente en niveles inferiores.

```
procedure EXPAND_CURRENT_LEVEL ()
begin
    if /# CurrentLevel has a label #/ then
      SAVE_DEFAULTS ();
    APPEND_DEFAULTS ();
    EXPAND_ATTACHMENTS (EvaluationTree, CurrentLevel, Defaults);
    (*   CurrentLevel está ahora libre de adjunciones de árbol. *)
    EXPAND_REPEATS ();
    /# Component IsExpanded of CurrentLevel #/ := TRUE;
end
```

B.5.5.2 Almacenamiento de valores por defecto

```
procedure SAVE_DEFAULTS ()
begin
        /# Replace CurrentLevel and its subsequent behaviour in the EvaluationTree by ACTIVATE (Defaults), followed by
        CurrentLevel and its subsequent behaviour, with the label of the former CurrentLevel moved to the ACTIVATE line. #/;
        /# Consider new ACTIVATE line as the CurrentLevel #/;
end
```

B.5.5.3 Expansión de los constructivos REPEAT

Si *RepeatedTree* denota una TreeReference particular junto con su ActualParList, y *Condition* denota una expresión booleana particular y *label* denota una etiqueta que no se utiliza en ningún otro sitio, "REPEAT *RepeatedTree* UNTIL [*Condition*]" puede sustituirse por:

```
            [TRUE]
label         +RepeatedTree
                    [NOT (Condition)]
                        -> label
                    [Condition]
                        :
```

Líneas que describan el comportamiento subsiguiente del constructivo REPEAT seguido de [*Condition*] en esta expansión, con un sangrado adicional de nivel uno.

```
procedure EXPAND_REPEATS ()
begin
        for /# every alternative A in CurrentLevel, in the given order #/ do
        begin
          if /# A is of the form REPEAT RepeatedTree UNTIL [Condition] #/ then
          begin
            Subsequent := SUBSEQUENT_BEHAVIOUR_TO (EvaluationTree,A);
            Label := NEW_LABEL ();
            (*   Crear una etiqueta que no se ha utilizado ni en la serie de pruebas (reetiquetada) ni en el árbol de evaluación. *)
            Expansion :=    MAKE_TREE ( "[TRUE]",
                      MAKE_TREE ( Label: "+" RepeatedTree,
                        MAKE_TREE ( "[NOT(" Condition ")]",
                              "->" Label,
                        MAKE_TREE ( "[" Condition "]",
                                  Subsequent,
                                  )),
                              ),
                            );
            REPLACE_ALT_TREE (EvaluationTree, CurrentLevel, A, Expansion);
          end
        end
end
```

B.5.5.4 Adición de comportamientos por defecto

Durante la evaluación de un caso de prueba, en cada nivel de alternativas hay una lista vigente de referencias de árbol por defecto. Esta lista procede de la lista del cuadro de comportamiento dinámico apropiada o del constructivo ACTIVATE evaluado más recientemente. La adición de los valores por defecto se realiza añadiendo, para cada entrada de la lista de valores por defecto vigente, el constructivo "+ DefaultReference" al final del conjunto de alternativas.

```
procedure APPEND_DEFAULTS ()
begin
        for /# every D in Defaults, in the given order #/ do
        begin
          APPEND_TO_LEVEL (EvaluationTree, CurrentLevel, "+" D);
          (*  EvaluationTree and CurrentLevel son actualizados añadiendo la adjunción de D a CurrentLevel. *)
        end
end
```

B.5.5.5 Expansión de árboles adjuntados

Los árboles adjuntados se expanden sustituyendo el constructivo de adjunción + *TestStep* por el árbol, o, cuando sea aplicable, por el árbol raíz de *TestStep* y consiguientemente, si existe un comportamiento especificado que le continúa y está sangrado desde el constructivo Attach, para insertar este comportamiento después del sangrado de cada hoja del árbol adjuntado. Puesto que los árboles adjuntados pueden tener su propia lista de referencias de árbol por defecto en el encabezamiento del cuadro de comportamiento dinámico de caso de prueba, la expansión de adjunción de árbol debe garantizar que no concuerda ningún evento en el primer nivel de alternativas del árbol adjuntado el contexto por defecto se cambia, y si se alcanza un nodo hoja de ese árbol adjuntado sin que se asigne un veredicto, se restablece el contexto por defecto del árbol llamante antes de que se evalúe el comportamiento siguiente. Estos cambios en el contexto por defecto se describen casi siempre en términos de la inserción de constructivos ACTIVATE apropiados en los lugares pertinentes. Si el árbol adjuntado es de hecho un árbol por defecto, entonces no habrá referencias por defecto en su encabezamiento, de modo que los constructivos ACTIVATE que se injertan en ese árbol no tienen parámetros, y por tanto desactivarán todos los valores por defecto dentro del ámbito del árbol por defecto.

Los árboles adjuntados en Level se expanden con el siguiente procedimiento:

```
procedure EXPAND_ATTACHMENTS (Tree, Level, OuterDefaults)
begin
        for /# every alternative A in Level in Tree, in the given order #/ do
        begin
          if /# A is an ATTACH construct, i.e. of the form "+" AttachedTreeId ActualParList #/ then
          begin
            Subsequent := SUBSEQUENT_BEHAVIOUR_TO (Tree,A);
            AttachedTree := ROOT_TREE (AttachedTreeId);
            REPLACE_PARAMETERS (AttachedTreeId, AttachedTree, ActualParList);
            (*  Sustituye a los parámetros formales en AttachedTree por los parámetros reales especificados en ActualParList,
                mediante sustitución textual *)
            RELABEL(AttachedTree);
            NewDefaults := DEF_REF_LIST(AttachedTreeId);
            NewLevel := FIRST_LEVEL(AttachedTree);
            EXPAND_ATTACHMENTS (AttachedTree, NewLevel, NewDefaults);
            EXPAND_SUBTREE (AttachedTree, Subsequent, NewDefaults, OuterDefaults);
            (*  Es decir: Insertar ACTIVATE(NewDefaults) debajo del primer nivel de AttachedTree & Adjuntar
                ACTIVATE(OuterDefaults) y Subsequent a cada nodo hoja de AttachedTree *)
            REPLACE_ALT_TREE(Tree, Level, A, AttachedTree);
          end
        end
end

procedure EXPAND_SUBTREE (SubTree, Subsequent, InnerDefaults, OuterDefaults)
        (*  Este procedimiento inserta en primer lugar ACTIVATE(InnerDefaults) debajo del primer nivel de SubTree y a
            continuación adjunta ACTIVATE(OuterDefaults) y Subsequent a cada nodo hoja de SubTree. *)
begin
        Level := FIRST_LEVEL(SubTree);
        for /# every alternative A of Level in SubTree #/ do
        begin
          SubOfA := SUBSEQUENT_BEHAVIOUR_TO (SubTree, A);
          ActTree := MAKE_TREE(A,
                MAKE_TREE("ACTIVATE(" InnerDefaults ")",
                    SubOfA, ), );
          REPLACE_ALT_TREE(SubTree, Level, A, ActTree);
        end
        for /# every leaf A in SubTree #/ do
        begin
          LeafTree :=          MAKE_TREE (A,
              MAKE_TREE ( "ACTIVATE(" OuterDefaults ")",
                    Subsequent, ), );
          REPLACE_ALT_TREE(SubTree, LEVEL_OF(SubTree, A), A, LeafTree);
        end
end
```

La expansión de árboles adjuntados se expone también en 15.13.

B.5.6 Evaluación de una línea de evento

B.5.6.1 Seudocódigo

```
function EVALUATE_EVENT_LINE(Alternative): BOOLEAN
        (* Esta función llama a EVALUATE_EVENT, EVALUATE_PSEUDO_EVENT o EVALUATE_CONSTRUCT, segúnl
           tipo de StatementLine que es la alternativa vigente *)
begin
        case    STATEMENT_LINE_TYPE_OF(Alternative) of
        begin
          EVENT:          if  EVALUATE_EVENT (Alternative)          then return TRUE; else return FALSE;
          PSEUDO_EVENT:   if  EVALUATE_PSEUDO_EVENT (Alternative)   then return TRUE; else return FALSE;
            CONSTRUCT:    (*  El constructivo ahora sólo puede ser GoTo, Return, Activate, Create. *)
                          if  EVALUATE_CONSTRUCT (Alternative)       then return TRUE; else return FALSE;
        end
end
```

B.5.6.2 Descripción en lenguaje natural

Se evalúa el enunciado en TTCN en la línea de comportamiento actual, basándose en el tipo de enunciado, es decir, según se trate de un evento, un seudoevento o un constructivo. La evaluación de cada tipo de enunciado en TTCN se especifica en la semántica operacional para ese tipo de enunciado en TTCN en las cláusulas siguientes.

B.5.7 Funciones para eventos de TTCN

B.5.7.1 Funciones para eventos TTCN – Seudocódigo

```
function EVALUATE_EVENT(Alternative): BOOLEAN
        (* Esta función llama a SEND, RECEIVE, OTHERWISE, TIMEOUT , DONE, o IMPLICIT SEND, según el tipo de
           evento que sea la alternativa vigente *)
begin
        case EVENT_TYPE_OF(Alternative) of
        begin
          SEND :          if  SEND (Alternative)          then return TRUE; else return FALSE;
          RECEIVE:        if  RECEIVE (Alternative)       then return TRUE; else return FALSE;
          OTHERWISE:      if  OTHERWISE (Alternative)     then return TRUE; else return FALSE;
          TIMEOUT:        if  TIMEOUT (Alternative)       then return TRUE; else return FALSE;
          DONE:           if  DONE (Alternative)          then return TRUE; else return FALSE;
          IMPLICIT_SEND:  if  IMPLICIT_SEND (Alternative) then return TRUE; else return FALSE;
        end
end
```

B.5.7.2 Funciones para eventos de TTCN – Descripción en lenguaje natural

Si el enunciado en TTCN es un evento, se evaluará como se indica en B.5.8 para un evento SEND; en B.5.9 para un evento RECEIVE; en B.5.10 para un evento OTHERWISE; en B.5.11 para un evento TIMEOUT; en B.5.12 para un evento DONE; o en B.5.13 para un evento IMPLICIT SEND.

B.5.8 Ejecución del evento SEND

B.5.8.1 Ejecución del evento SEND – Seudocódigo

```
function SEND ( SendLine ): BOOLEAN
begin
        /# ReadPCOorCPidentifier,
                ASPorPDUorCMidentifier,
                Qualifier,
                Assignments,
                TimerOperations,
                ConstraintsReference            from SendLine #/;
        if EVALUATE_BOOLEAN (Qualifier) then
        begin
          BUILD_SEND_OBJECT (ASPorPDUorCMidentifier, ConstraintsReference );
          EXECUTE_ASSIGNMENTS (Assignment);
          SEND_EVENT (PCOorCPidentifier, ConstraintReference);
          TIMER_OPS (TimerOperations);
          LOG(PCOorCPidentifier, SendObject);
          return TRUE;
        end
```

```
        else return FALSE;
end

procedure BUILD_SEND_OBJECT (ASPorPDUorCMidentifier, ConstraintsReference)
begin
        SendObject :=   /#an instance of ASPorPDUorCMidentifier whose parameters/fields have the values specified by
                        ConstraintsReference #/;
end

procedure SEND_EVENT (PCOorCPidentifier, ConstraintsReference)
begin
        /#  Encode SendObject according to applicable encoding rules and variations, see ConstraintsReference and associated type
            definitions #/;
        /#  Put encoded SendObject at the end of OUTPUT_Q(PCOorCPidentifier) #/;
end
```

B.5.8.2 Ejecución del evento SEND – Descripción en lenguaje natural

Ha de enviarse el contenido de la ASP, la PDU o el CM tal y como se especifica en la entrada de referencia a constricciones denominada. Obsérvese que si existe un calificador solamente podrá ejecutarse el SEND si dicho calificador toma el valor TRUE.

Paso 1 Si existe un calificador, se evalúa con antelación a cualquier otro tipo de procesamiento:

- si el calificador toma el valor FALSE, no puede ejecutarse el SEND;
- si el calificador toma el valor TRUE, se continúa con el paso 2.

Paso 2 Crear una ASP, una PDU o un CM como se especifica en la referencia a constricciones denominada.

Si se ha hecho uso de la característica de encadenamiento dinámico, se asigna el valor especificado en la entrada de referencia a constricciones al parámetro o campo apropiado de la ASP, la PDU o el CM que haya de enviarse.

La utilización de la característica de encadenamiento dinámico tiene como efecto el almacenamiento de una copia de la constricción denominada en el parámetro o en el campo denominado de la ASP, la PDU o el CM que se está construyendo, a efectos de comparación. Para este parámetro o campo denominado se utiliza la estructura definida para la referencia a constricciones asociada.

Paso 3 Si existe un enunciado asignación, se efectúa esa asignación tal como se indica en B.5.16, en especial posiblemente cambiando la ASP, la PDU o el CM que ha de enviarse.

Paso 4 En este momento, la ASP, la PDU o el CM están completamente llenos de acuerdo con las especificaciones dadas. El LT o el UT codificará las PDU (pero no codificará las ASP o los CM, aparte de las PDU incrustadas en los mismos) de acuerdo con las reglas de codificación aplicables. El LT o el UT envía la ASP, con sus PDU codificadas incrustadas, o la PDU codificada. Si se especificó un PCO o un CP, se envía a dicho PCO o CP la ASP, la PDU o el CM. Si no se especificó ningún PCO, es decir, que la prueba utiliza un PCO único, se envía la ASP o la PDU desde el PCO inferior, pues no un CP no puede estar implicado.

Paso 5 Si en la línea de comportamiento se han codificado una o más operaciones de temporizador, se efectúan la operación u operaciones de temporizador apropiadas tal como se indica en B.5.17.

Paso 6 En el registro cronológico de conformidad se registra la información que sigue, así como la información especificada en B.5.24.2:

- el PCO o CP en el que se produjo el SEND;
- la ASP, la PDU o el CM totalmente definido que se envió.

B.5.9 Ejecución del evento RECEIVE

B.5.9.1 Ejecución del evento RECEIVE – Seudocódigo

```
function RECEIVE( ReceiveLine ): BOOLEAN
begin
        /# Read   PCOorCPidentifier,
                  ASPorPDUorCMidentifier,
                  Qualifier,
                  Assignments,
                  TimerOperations,
                  ConstraintsReference        from ReceiveLine #/;
        if /# INPUT_Q (PCOorCPidentifier) is not empty #/ then
        begin
```

```
            if ( OBJECT_MATCHES(PCOorCPidentifier, ASPorPDUorCMidentifier, ConstraintsReference)
                AND EVALUATE_BOOLEAN (Qualifier) ) then
            begin
                EXECUTE_ASSIGNMENTS (Assignments);
                TIMER_OPS (TimerOperations);
                REMOVE_OBJECT (PCOorCPidentifier);
                LOG(PCOorCPidentifier, ReceiveObject);
                return TRUE;
            end
            else return FALSE;
        end
        else return FALSE;
end

function OBJECT_MATCHES (PCOorCPidentifier, ASPorPDUorCMidentifier, ConstraintsReference): BOOLEAN
begin
        ReceiveObject := /# copy of encoded object at head of INPUT_Q(PCOorCPidentifier) #/;
        if /#  ReceiveObject can be decoded according to applicable encoding rules and variations, as given by
               ConstraintsReference and associated type definitions #/ then
        begin
          /# decode it, to yield new version of ReceiveObject #/;
          if  ( /# ReceiveObject is of type ASPorPDUorCMidentifier #/
                AND
                /# parameters/fields of ReceiveObject have values matching the ConstraintsReference #/ ) then
              return TRUE;
          else return FALSE;
        end
        else return FALSE;
end

procedure REMOVE_OBJECT (PCOorCPidentifier),
begin
        /# remove object at head of INPUT_Q(PCOorCPidentifier) #/;
end
```

B.5.9.2 Ejecución del evento RECEIVE – Descripción en lenguaje natural

Paso 1 Si la instantánea tomada al comienzo de la iteración en curso de la verificación de este nivel de alternativas a efectos de concordancia indica que *no* hay ninguna ASP, PDU o CM entrante, este RECEIVE no podrá concordar.

En cualquier otro caso, se continúa con el paso 2.

Paso 2 Si se especificó un PCO o CP, la ASP, la PDU o el CM se habrá recibido en ese PCO o CP. Si no se especificó ningún PCO, es decir, la serie de pruebas utiliza un PCO único, se habrá recibido la ASP o la PDU en el PCO inferior. Obsérvese que no puede estar implicado un CP.

Paso 3 Las PDU entrantes se decodifican de conformidad con las reglas de codificación aplicables. Se hace una copia de la PDU entrante decodificada o de la ASP o el CM entrante con las PDU anidadas decodificadas.

Paso 4 Si el calificador, que puede utilizar valores del objeto de datos entrante, toma el valor FALSE, RECEIVE no puede concordar. En los demás casos, se continúa con el paso 5.

Paso 5 Se ensambla una copia del modelo de la ASP, la PDU o el CM esperado con la estructura definida en la declaración de la ASP, la PDU o el CM más los valores, mecanismos de concordancia y referencias a constricciones encadenadas, especificados en la referencia a constricciones denominada.

Esta copia se compara con la ASP, la PDU o el CM entrante, y sus PDU decodificadas o la PDU decodificada, para determinar si el RECEIVE puede concordar como se había especificado. Sólo si el RECEIVE no concuerda exitosamente, se continúa en el paso 6.

Paso 6 La ASP, la PDU o el CM entrante que ha concordado exactamente se suprimirá de la cola de PCO o CP de entrada y será descartado.

Paso 7 Si existen enunciados de asignación, se realizarán tal como se indica en B.5.16.2.

Paso 8 Si en la línea de comportamiento se han codificado una o más operaciones de temporizador, se efectúan la operación u operaciones apropiadas, tal como se indica en B.5.17.

Paso 9 En el registro cronológico de conformidad se registra la información que sigue, así como la información especificada en B.5.24.2:

- el PCO o el CP en el que se produjo el RECEIVE;
- la ASP, la PDU o el CM totalmente definido que se recibió.

B.5.10 Ejecución del evento OTHERWISE

B.5.10.1 Ejecución del evento OTHERWISE – Seudocódigo

```
function OTHERWISE ( OtherwiseLine): BOOLEAN
begin
        /# Read   PCOorCPidentifier,
                  Qualifier,
                  Assignments,
                 TimerOperations          from OtherwiseLine #/;
        if   (   /# INPUT_Q (PCOorCPidentifier) is not empty #/
               AND EVALUATE_BOOLEAN (Qualifier) ) then
        begin
          EXECUTE_ASSIGNMENTS (Assignments);
          TIMER_OPS (TimerOperations);
          REMOVE_OBJECT (PCOorCPidentifier);
          LOG(PCOidentifier, ReceivedObject);
          return TRUE;
        end
        else return FALSE;
end
```

B.5.10.2 Ejecución del evento OTHERWISE – Descripción en lenguaje natural

El probador debe aceptar cualquier dato de entrada que no se haya podido decodificar o que no haya concordado con una alternativa anterior a este evento OTHERWISE. Obsérvese que si existe un calificador, solamente podrá concordar el OTHERWISE si dicho calificador toma el valor TRUE.

Paso 1 Si el calificador toma el valor FALSE, el OTHERWISE no puede concordar. En los demás casos se continúa en el paso 2.

Paso 2 Si la instantánea tomada al comienzo de la iteración en curso de la verificación de este nivel de alternativas a efectos de concordancia indica que no hay ninguna ASP, PDU o CM entrante, este OTHERWISE no podrá concordar.

En cualquier otro caso, se continúa con el paso 3.

Paso 3 Si se especificó un PCO, la ASP o la PDU se habrá recibido en ese PCO. Si se especificó el CP, se habrá recibido el CM en ese CP. Si no se especificó ningún PCO, es decir, que la prueba utiliza un PCO único, se habrá recibido la ASP o la PDU en el PCO inferior, ya que un CP no puede estar involucrado.

Paso 4 La ASP, la PDU o el CM entrante serán eliminados de la cola del PCO o CP a la entrada y descartados.

Paso 5 Si existen enunciados de asignación, se efectúan tal como se indica en B.5.16.2

Paso 6 Si en la línea de comportamiento se han codificado una o más operaciones de temporizador, se efectúan la operación u operaciones de temporizador apropiadas, tal como se indica en B.5.17.

Paso 7 En el registro cronológico de conformidad se registra la información que sigue, así como la información especificada en B.5.24.2:

- el PCO o el CP en el que se produjo el OTHERWISE;
- la ASP, la PDU o el CM que se recibió.

B.5.11 Ejecución del evento TIMEOUT

B.5.11.1 Ejecución del evento TIMEOUT – Seudocódigo

```
function TIMEOUT ( TimeoutLine ): BOOLEAN
begin
        /# Read   TimerIdentifier,
                  Qualifier,
                  Assignments,
                  TimerOperations          from TimeoutLine #/;
        if EVALUATE_BOOLEAN (Qualifier) then
```

```
        begin
          if TIMER_EXPIRED (TimerIdentifier) then
          begin
            EXECUTE_ASSIGNMENTS (Assignments);
            TIMER_OPS (TimerOperations);
            LOG(TimerIdentifier);
            return TRUE;
          end
          else return FALSE;
        end
        else return FALSE;
end

function TIMER_EXPIRED (TimerIdentifier): BOOLEAN
begin
        if /# TimerIdentifier is not empty #/ then
        begin
          if /# timeout notification from TimerIdentifier is in copy of timeout list in Snapshot #/ then
          begin
            /# delete timeout notification from TimerIdentifier in actual timeout list #/;
            /# stop and reset the timer TimerIdentifier #/;
          return TRUE;
          end
          else return FALSE;
        end
        else (* TimerIdentifier not specified *)
        begin
          if /# any timeout notification is in copy of timeout list in Snapshot #/ then
          begin
            /# stop and reset all timers mentioned in actual timeout list#/;
            /# delete all timeout notifications in actual timeout list #/;
            return TRUE;
          end
          else return FALSE;
        end
end
```

B.5.11.2 Ejecución del evento TIMEOUT – Descripción en lenguaje natural

El probador verifica si ha expirado el temporizador denominado. (Si no se facilita ningún nombre de temporizador, el probador verificará si ha expirado *cualquier* temporizador.) Obsérvese que si existe un calificador, se considera que el TIMEOUT concuerda si dicho calificador toma el valor TRUE.

Paso 1 Si existe un calificador, se evalúa con antelación a cualquier otro tipo de procesamiento.

- Si el calificador toma el valor FALSE, el TIMEOUT no puede concordar.
- Si el calificador toma el valor TRUE, se continúa con el paso 2.

Paso 2 Se comprueba si alguno de los temporizadores denominados explícita o implícitamente en el evento TIMEOUT ha estado en marcha y ha expirado.

- Si no se especifica ningún identificador de temporizador, el probador debe verificar si ha estado en marcha *algún* temporizador y ha expirado ya. Si esto es así, se reponen todos los temporizadores que hayan expirado (y se dejan parados). Se eliminan de la lista de temporizaciones la entrada o inscripciones de la temporización.
- Si se especifica un identificador de temporizador, el probador verifica si este temporizador ha estado en marcha pero ya ha expirado. De ser así, se repone el temporizador que ha expirado (y se deja parado). Se elimina de la lista de temporizaciones la entrada de la temporización.
- Si no han expirado los temporizadores, el evento TIMEOUT no puede concordar, es decir, que se intenta la alternativa siguiente.

Paso 3 Si existe un enunciado asignación, se efectúa esa asignación tal como se indica en B.5.16.2.

Paso 4 Si en la línea de comportamiento se han codificado una o más operaciones de temporizador, se efectúan la operación u operaciones de temporizador apropiadas tal como se indica en B.5.17.

Paso 5 En el registro cronológico de conformidad se registra la información especificada en B.5.24, así como el nombre del temporizador que ha expirado.

B.5.12 Ejecución del evento DONE

B.5.12.1 Ejecución del evento DONE – Seudocódigo

```
function DONE ( DoneLine): BOOLEAN
begin
        /# Read   TCompList,
                  Qualifier,
                  Assignments,
                  TimerOperations          from DoneLine #/;
        if EVALUATE_BOOLEAN (Qualifier)          AND ALL_TERMINATED(TCompList)  then
        begin
          EXECUTE_ASSIGNMENTS (Assignments);
          TIMER_OPS (TimerOperations);
          LOG(TCompList);
          return TRUE;
        end
        else return FALSE;
end

function ALL_TERMINATED(TCompList): BOOLEAN
begin
        if   TCompList =/# EmptyList #/ then
             TCompList := /# list of all created Parallel Test Components #/;
        for /# every TComp in TCompList #/ do
        begin
          if /# TComp has not terminated in the Snapshot #/ then
            return FALSE;
        end
        return TRUE;
end
```

B.5.12.2 Ejecución del evento DONE – Descripción en lenguaje natural

Ha de comprobarse el estado terminación de la lista de componentes de prueba dada. Si todos los componentes dados han terminado (en el momento de la última SNAPSHOT) el evento concuerda, siempre que el calificador también tome el valor TRUE.

Paso 1 Si existe un calificador, se evalúa con antelación a cualquier otro tipo de procesamiento:

- si el calificador toma el valor FALSE, el DONE no puede tener éxito;
- si el calificador toma el valor TRUE, se continúa con el paso 2.

Paso 2 Si todos los componentes de prueba relacionados en la TCompList hubieran terminado al mismo tiempo que la última SNAPSHOT, se continúa con el paso 3, y en los demás casos este DONE no puede concordar.

Paso 3 Si existe un enunciado asignación, se efectuará esa asignación como se indica en B.5.16.

Paso 4 Si en la línea de comportamiento se han codificado una o más operaciones de temporizador, se efectúan la operación u operaciones de temporizador apropiadas tal como se indica en B.5.17.

Paso 5 En el registro cronológico de conformidad se registra la información especificada en B.5.24, así como la TCompList.

B.5.13 Ejecución del evento IMPLICIT SEND

B.5.13.1 Ejecución del evento IMPLICIT SEND – Seudocódigo

```
function IMPLICIT_SEND (Alternative): BOOLEAN
begin
        /# Execute IMPLICIT_SEND according to natural language description #/;
        return TRUE;
end
```

B.5.13.2 Ejecución de IMPLICIT SEND – Descripción en lenguaje natural

La IUT es inducida a realizar todo lo que sea necesario para el envío del contenido de la ASP o la PDU como se especifica en la entrada de referencia a constricción de la alternativa.

Si se ha hecho uso de la característica de encadenamiento dinámico, se asigna el valor especificado en la entrada de referencia a constricción al parámetro o campo apropiado de la ASP o la PDU que haya de enviarse.

IMPLICIT SENDing siempre tiene éxito.

B.5.14 Ejecución de un pseudo-event

B.5.14.1 Ejecución de un pseudo-event – Seudocódigo

```
function EVALUATE_PSEUDO_EVENT ( PseudoEventLine ): BOOLEAN
begin
        /# Read   Qualifier,
        Assignments,
        TimerOperations                 from PseudoEventLine #/;
        if EVALUATE_BOOLEAN (Qualifier) then
        begin
          EXECUTE_ASSIGNMENTS (Assignments);
          TIMER_OPS (TimerOperations);
          LOG( );
          return TRUE;
        end
        else return FALSE;
end
```

B.5.14.2 Ejecución de PSEUDO-EVENTS – Descripción en lenguaje natural

Si el enunciado TTCN es un seudoevento, se evaluará como se indica en B.5.15 para una expresión booleana, en B.5.16 para un enunciado asignación y en B.5.17 para una operación de temporizador (START, CANCEL o READTIMER).

Tras la compleción del seudoevento, se registra en el registro cronológico de conformidad la información especificada en B.5.24

B.5.15 Ejecución de expresiones BOOLEAN

B.5.15.1 Ejecución de expresiones BOOLEAN – Seudocódigo

```
function EVALUATE_BOOLEAN(Qualifier): BOOLEAN
begin
        if /# Qualifier is empty #/ then
          return TRUE;
        else
        begin
          if /# Qualifier evaluates to TRUE #/ then
            return TRUE;
          else return FALSE;
        end
end
```

B.5.15.2 Ejecución de expresiones BOOLEAN – Descripción en lenguaje natural

Una expresión booleana (es decir, un calificador) especifica una condición que ha de comprobarse. Esta condición podrá ser TRUE o FALSE. La expresión booleana se puede enunciar como parte de una línea de enunciado (esto es, en la misma línea que SEND, RECEIVE, TIMEOUT u OTHERWISE) o como una línea de enunciado propia (esto es, como un seudoevento).

Paso 1 Se evalúa la expresión booleana para determinar si la condición especificada es TRUE o FALSE. Se aplican las reglas normales de la lógica booleana, junto con las reglas de prioridad especificadas en 11.3.2.1.

B.5.16 Ejecución de asignaciones

B.5.16.1 Ejecución de assignments – Seudocódigo

```
procedure EXECUTE_ASSIGNMENTS (AssignmentList)
begin
        for /# every assignment CurrentAssignment in AssignmentList, in the given order #/ do
        begin
          /# Execute CurrentAssignment #/;
        end
end
```

B.5.16.2 Ejecución de ASSIGNEMENT – Descripción en lenguaje natural

La lista de asignaciones se evalúa de izquierda a derecha. En cada asignación, la variable del lado izquierdo de ese enunciado tomará el valor de la expresión del lado derecho del enunciado. La expresión se evalúa observando la precedencia indicada en el cuadro 3.

Si la asignación se ejecuta en una línea Send, el lado izquierdo puede denotar un componente de ASP-, PDU- o CM-, referente al objeto que se ha de enviar. Si la asignación se ejecuta en la línea Receive, la expresión se puede referir a componentes de la ASP-, PDU- o CM que ha de recibirse.

B.5.17 Ejecución de operaciones TIMER

B.5.17.1 Ejecución de operaciones TIMER – Seudocódigo

```
procedure TIMER_OPS (TimerOperations)
begin
        for /# every TimerOperation in TimerOperations #/ do
        case TIMER_OP_TYPE_OF(TimerOperation) of
        begin
          START_TIMER:              START_TIMER(TimerOperation);
          CANCEL_TIMER:             CANCEL_TIMER(TimerOperation);
          READ_TIMER:               READ_TIMER(TimerOperation);
        end
end

procedure START_TIMER (TimerOperation)
begin
        /# perform as in B.5.17.2 #/;
end

procedure CANCEL_TIMER (TimerOperation)
begin
        /# perform as in B.5.17.3 #/;
end

procedure READ_TIMER (TimerOperation)
begin
        /# perform as in B.5.17.4 #/;
end
```

B.5.17.2 Ejecución de temporizador START– Descripción en lenguaje natural

Paso 1 Si el temporizador ya está en marcha, deberá cancelarse e ir al paso 2. En los demás caso se va directamente al paso 2.

Paso 2 El temporizador deberá arrancarse con un valor inicial que indique que no ha transcurrido ningún tiempo. Cualquier entrada para este temporizador que figure en la lista de temporizaciones deberá eliminarse de la lista.

B.5.17.3 Ejecución del temporizador CANCEL – Descripción en lenguaje natural

La operación de temporizador CANCEL especifica que un temporizador o unos temporizadores van a dejar de funcionar.

Paso 1 Se determina el nombre del temporizador o temporizadores que se han de cancelar:

- si no se especifica ningún identificador de temporizador, se cancelan *todos* los temporizadores;
- si se especifica un identificador de temporizador, se cancela el temporizador con este identificador de temporizador.

Paso 2 El estado del temporizador o temporizadores denominados o implícitos pasará a "detenido". El tiempo transcurrido para el temporizador (o temporizadores) se pone a cero. Si la lista de temporizaciones contiene una entrada para el temporizador o temporizadores, se eliminará esa entrada o inscripciones de la lista.

B.5.17.4 Ejecución de READTIMER – Descripción en lenguaje natural

La operación READTIMER especifica el almacenamiento en una variable del tiempo transcurrido del funcionamiento de un temporizador que, en ese momento, está funcionando. El temporizador continúa funcionando sin interrupción.

Paso 1 Se pregunta el valor del temporizador que tiene el nombre especificado. Si el tiempo transcurrido es *n* de las unidades declaradas para este tipo de temporizador, se almacena *n* en la variable denominada.

Si el temporizador no está funcionando en ese momento, la variable denominada se pone a cero.

B.5.18 Funciones para constructivos de TTCN

B.5.18.1 Funciones para constructivos de TTCN – Seudocódigo

```
function EVALUATE_CONSTRUCT (Construct): BOOLEAN
        (* Ya que el árbol de evaluación es expandido en el CurrentLevel, aquí no se encuentran los constructivos REPEAT y
           ATTACH. *)
```

```
begin
        case CONSTRUCT_TYPE_OF(Construct) of
        begin
          ACTIVATE:     ACTIVATE(Construct);
          CREATE:       CREATE (Construct);
          GOTO:         (*  aquí ninguna acción, véase GOTO_NEXT_LEVEL_OR_STOP_WITH_VERDICT *);
          RETURN:       (*  aquí ninguna acción, véase GOTO_NEXT_LEVEL_OR_STOP_WITH_VERDICT *);
        end
        return TRUE;
end
```

B.5.18.2 Funciones para constructivos de TTCN – Descripción en lenguaje natural

Si el enunciado en TTCN es un constructivo de TTCN, se evalúa como se indica en B.5.19 para un constructivo ACTIVATE, como se indica en B.5.20 para un constructivo CREATE, como se indica en B.5.21 para un constructivo GOTO o como se indica en B.5.22 para un constructivo RETURN. Como han sido sustituidos en el CurrentLevel, no es necesario tratar con constructivos REPEAT.

Los constructivos de TTCN serán siempre exitosos.

B.5.19 Ejecución del constructivo ACTIVATE

B.5.19.1 Ejecución del constructivo ACTIVATE – Seudocódigo

```
procedure ACTIVATE ( ActivateLine )
begin
        /# Read DefRefList      from ActivateLine #/;
        Defaults:=DefRefList;
        LOG(DefRefList);
end
```

B.5.19.2 Ejecución del ACTIVATE – Descripción en lenguaje natural

Se cambia el contexto por defecto vigente a la DefaultRefList que aparece como parámetro del constructivo ACTIVATE.

Paso 1 Se cambia el contexto por defecto a DefaultRefList.

Paso 2 En el registro cronológico de conformidad se registra la siguiente información, así como la información especificada en B.5.24:

- la DefaultRefList.

B.5.20 Ejecución del constructivo CREATE

B.5.20.1 Ejecución del evento CREATE – Seudocódigo

```
procedure CREATE ( CreateLine ): BOOLEAN
begin
        /#  Read    CreateList     from CreateLine #/;
        for /# every (TCompIdentifier,    TreeReference,    ActualParList) drawn from CreateList #/ do
        begin
          start process EVALUATE_TEST_COMPONENT(TCompIdentifier, TreeReference, ActualParList);
          (*  Arranca la evaluación concurrente de TreeReference. *)
          LOG(TCompIdentifier,TreeReference, ActualParList);
        end
end

process EVALUATE_TEST_COMPONENT(TCompId, TreeReference, ActualParList)
        (*  Este proceso inicializa el árbol de evaluación por el árbol raíz o árbol local de serie de pruebas apropiado y el contexto
            por defecto por las referencias por defecto listadas del cuadro de comportamiento correspondiente. Con ello se desplaza
            el control al nivel superior de alternativas y pide su evaluación. *)
global EvaluationTree, CurrentLevel, Defaults, Snapshot, ReturnLevel, ReturnDefaults, SendObject, ReceiveObject;
begin
    /#  Initialize the local instances of Test Case Variables, local R, Timers, and the Timeout List of TCompId. #/;
    EvaluationTree := ROOT_TREE(TreeReference);
      (*    El árbol de evaluación es un árbol finito en crecimiento construido por copias expandidas y pegadas juntas de árboles
            procedentes de la descripción de comportamiento de caso de prueba y de las bibliotecas de casos de prueba y valores por
            defecto. A cada nivel se añade un componente IsExpanded. *)
    REPLACE_PARAMETERS (TreeReference, EvaluationTree, ActualParList);
      CurrentLevel := FIRST_LEVEL(EvaluationTree) ;
      (*  Un nivel denota tanto una posición en un árbol como el conjunto ordenado de alternativas en esta posición. *)
    ReturnLevel := CurrentLevel;
```

```
    Defaults := DEF_REF_LIST(TreeReference);
    ReturnDefaults := Defaults;
    EVALUATE_LEVELS ();
    (* Incluye, mediante llamadas anidadas, la evaluación de todos los niveles subsiguientes pertinentes en el árbol de evaluación en
       crecimiento. *)
end
```

B.5.20.2 Ejecución del evento CREATE – Descripción en lenguaje natural

Se arrancará la evaluación del componente de prueba dado.

Paso 1 Se arranca la evaluación de TCompIdentifier, vinculado a TreeReference, con los parámetros de la ActualParList sustituyendo a los parámetros formales por sustitución textual en TreeReference. Todas las variables de caso de prueba, la variable de resultados locales R, los temporizadores y la lista de temporizaciones local se facilitan de nuevo para el uso exclusivo por este componente de prueba.

Paso 2 En el registro cronológico de conformidad se registra la información que sigue, así como la información especificada en B.5.24:

- el TCompIdentifier;
- la TreeReference;
- la ActualParList.

B.5.21 Ejecución del constructivo GOTO

Se transfiere el control al conjunto de alternativas que tengan el objetivo de etiqueta especificado en la columna de etiqueta. La ejecución prosigue a continuación en este nuevo nivel.

En seudocódigo, el constructivo GOTO se ejecuta como parte de GOTO_NEXT_LEVEL_OR_STOP_WITH_ VERDICT.

B.5.22 Ejecución del constructivo RETURN

Se transfiere el control al conjunto de alternativas desde el cual se incluyeron los valores por defecto la última vez. La ejecución prosigue a continuación en este mismo nivel.

En seudocódigo, el constructivo RETURN se ejecuta como parte de GOTO_NEXT_LEVEL_OR_STOP_WITH_ VERDICT.

B.5.23 Veredicto

B.5.23.1 Veredicto – Seudocódigo

```
procedure EVAL_VERDICT_ENTRY (VerdictEntry)
begin
        /# Expand VerdictEntry to full word, e.g. (P) becomes (PASS) #/;
        if /# VerdictEntry is a preliminary verdict "("PrelimVerdict")" #/ then
        begin
          UPDATE_PRELIM ( PrelimVerdict, /# local R, or MTC_R in case of Main Test Component #/);
          UPDATE_PRELIM ( PrelimVerdict, /# global R #/);
        end
        else (*VerdictEntry es un veredicto final. *)
        begin
          if /# Current process is EVALUATE_TEST_CASE #/ then
          begin
            EXCLUDE_INCOMPATIBLE_ENTRY ( VerdictEntry, /# global R #/);
            LOG(VerdictEntry);
            /# assign final verdict in main test component or test case #/;
            TERMINATE_TEST_CASE();
          end
          else (*El proceso es EVALUATE_TEST_COMPONENT *)
          begin
            EXCLUDE_INCOMPATIBLE_ENTRY ( VerdictEntry, /# local R #/);
            UPDATE_PRELIM          ( VerdictEntry, /# global R #/);
            stop process;
          end
        end
end
```

```
process EXCLUDE_INCOMPATIBLE_ENTRY (Entry, RVal)
begin
        if ( ( Entry = "R" AND /# RVal = none #/ ) OR
             (Entry = "PASS" AND /# Rval = inconc #/ ) OR
             (Entry = "PASS" AND /# Rval = fail #/ ) OR
             (Entry = "INCONC" AND /# Rval = fail #/ ) ) then
        begin
          LOG(TestCaseError);
          STOP_TEST_CASE();
          return FALSE;
        end
        else return TRUE;
end

procedure UPDATE_PRELIM (PrelimVerdict, ResultVar)
begin
        if (  ResultVar = none OR
             (ResultVar = pass AND PrelimVerdict <> PASS) OR
             (ResultVar = inconc AND PrelimVerdict = FAIL) ) then
        begin
          /# replace value of ResultVar by PrelimVerdict in lower case letters #/;
          LOG("("PrelimVerdict")");
        end
end
```

B.5.23.2 Veredicto – Descripción en lenguaje natural

Si hay un veredicto codificado, se procesa el veredicto:

- Si el veredicto es preliminar, es decir, figura entre paréntesis, se actualizan las variables de resultado local y global según el algoritmo de veredicto de 15.17.2. Obsérvese que en el componente de prueba principal la R local se señala mediante MTC_R. El veredicto establecido se registra en el registro cronológico de conformidad.
- Si el veredicto es R, se utilizará, en la TTCN no concurrente o en el componente de prueba principal, el valor vigente de R (sólo o la R global) como veredicto del caso de prueba. Si R está fijada a ninguno, se señala un error de caso de prueba.
- Si el veredicto es PASS, INCONC o FAIL, se utiliza, en la TTCN no concurrente o en el componente de prueba principal, el veredicto establecido como veredicto final del caso de prueba. Si el veredicto final no es coherente con la R local o global se señala un TestCaseError.
- En los componentes de prueba paralelos, se utiliza un veredicto final R, PASS, INCONC o FAIL para actualizar la R global como un veredicto preliminar. El veredicto establecido se registra en el registro cronológico de conformidad. Un veredicto final termina la evaluación del componente de prueba.

B.5.24 Registro cronológico de conformidad

B.5.24.1 LOG – Seudocódigo

```
procedure LOG( /# any number of arguments #/ )
begin
        /# log the line number of the event line (if any) #/;
        /# log the label associated with the event line (if any) #/;
        /# log the arguments passed to LOG #/;
        /# log the assignment(s) made (if any) #/;
        /# log the timer operation(s) performed (if any) #/;
        /# log current time #/; (* current time may be actual or relative *)
end
```

B.5.24.2 Registro cronológico de conformidad – Descripción en lenguaje natural

En el registro cronológico de conformidad se registra la siguiente información:

- número de línea de la línea de eventos (si existe);
- etiqueta asociada con la línea de eventos (si existe);
- otros argumentos definidos en otras partes de este anexo que estén asociados con la línea de eventos (si existe), por ejemplo, el veredicto final o preliminar, o el objeto de datos enviado o recibido;
- asignación o asignaciones efectuadas (si existen);
- operación u operaciones de temporizador ejecutadas (si existen);

- indicación de tiempo.

B.5.25 Funciones y procedimientos de tratamiento del árbol

Para facilitar su búsqueda, los procedimientos y funciones se definen por orden alfabético.

```
procedure APPEND_TO_LEVEL (Tree,Level,Alternative)
begin
        /# Update Level and Tree by appending Alternative as new last alternative in Level in Tree #/;
end

function FIRST_LEVEL (Tree): LEVEL
begin
        return /# the set of alternatives at the first level of indentation of Tree, i.e. the numerically lowest (in TTCN.MP),
                i.e. the leftmost (in TTCN.GR), level of indentation of the root tree #/;
end

procedure GOTO_NEXT_LEVEL_OR_STOP_WITH_VERDICT(Alternative)
begin
        (*   búsqueda del siguiente nivel que hay que evaluar, si existe *)
        if /# Alternative is of the type "GOTO Label" or "-> Label" #/ then
          CurrentLevel := /# the unique level labelled with Label #/;
        else if /# Alternative is of the type "RETURN" #/ then
        begin
          CurrentLevel := ReturnLevel;
          Defaults := ReturnDefaults;
        end
        else if /# Alternative is a leaf of EvaluationTree #/; (*pero no un RETURN o un GOTO *) then
          EVAL_VERDICT_ENTRY("R"); (*Parará la ejecución del proceso. *)
        else
          CurrentLevel := /# set of alternatives at next level of indentation below Alternative #/;
        (*   almacenamiento de información para enunciados RETURN entrantes *)
        if /# Component IsDefault of CurrentLevel #/ = FALSE then
        begin
          ReturnLevel := CurrentLevel;
          ReturnDefault := Default;
        end
end

function IS_EXPANDED (): BOOLEAN
begin
        return /# Component IsExpanded of CurrentLevel #/;
end

function LEVEL_OF (Tree, Alternative): LEVEL
begin
        return /# the level in Tree of which this Alternative is a member #/;
end

function MAKE_TREE (Statement, Tree1, Tree2): TREE
begin
        return /# the following tree:
            Statement
              Tree1
            Tree2                                   #/     ;
        (*   Tree1 y/o Tree2 puede estar vacío, lo que se indica mediante una posición de parámetro vacía en la llamada de
            MAKE_TREE. *)
end

function NEW_LABEL ( ): LABEL
begin
        return /# a label which has not yet been used in the execution of this Test Component, nor in the (relabelled) Test Suite #/ ;
                (*   Puede alcanzarse mediante contadores y nombres de componentes de prueba. *)
end

procedure RELABEL (Tree)
begin
        for /# each label L originally occurring in Tree #/ do
          begin
            NewLabel := NEW_LABEL();
            for /# each occurrence of L in Tree, in the label column or as the target of a GOTO #/ do
              begin
                /# replace L by NewLabel #/;
```

```
            end
        end
end

procedure REPLACE_ALT_TREE (Tree, Level, A, ReplacementTree)
begin
        (*  A es una alternativa de Level, que es un nivel en Tree *)
        /#  In Tree, replace the subtree of Tree consisting of
            A and SUBSEQUENT_BEHAVIOUR_TO (Tree, A) by ReplacementTree,
            with all values of IsDefault in ReplacementTree set to the IsDefault-value of A,
            and all values of IsExpanded of levels in ReplacementTree set to FALSE. #/;
end

procedure REPLACE_PARAMETERS (TreeId, Tree, ActualParList)
begin
        /#  Replace the formal parameters in Tree by the actual parameters specified in ActualParList,
            doing so by textual substitution in Tree, using the formal parameter list accessible via TreeId. #/;
end

function ROOT_TREE (TreeId): TREE
begin
        return    /#  its root tree if TreeId denotes a Test Case or Test Step or Default Behaviour Table –
                      otherwise the local tree with this name. Each level gets a new Boolean component
                      "IsExpanded", initialized with value FALSE, indicating that this level has not yet been expanded. #/;
end

function SUBSEQUENT_BEHAVIOUR_TO (Tree, Alternative): TREE
begin
        return /# the subtree below Alternative in Tree #/;
          (*  Éste sería el Tree3 si Tree tiene la forma:
              Tree1
                Tree2
                Alternative
                  Tree3
                Tree4
              Tree5                    *)
end
```

B.5.26 Funciones diversas utilizadas por el seudocódigo

```
function CONSTRUCT_TYPE_OF(Construct): CONSTRUCT_TYPE
begin
        return /# ACTIVATE, CREATE, GOTO, or RETURN, as appropriate #/;
end

function DEF_REF_LIST(TreeReference): DEFAULT_REF_LIST
begin
          return /# the default reference list in the header of the corresponding table in the case of a test step in the test step library,
        or the empty list in the case of default behaviour, or in the case of a local tree attachment the current value of Defaults
        (i.e., the currently active defaults in the calling tree)#/;
end

function EVENT_TYPE_OF(Alternative): EVENT_TYPE
begin
        return /# SEND, RECEIVE, OTHERWISE, TIMEOUT, DONE, or IMPLICIT_SEND, as appropriate #/;
end

function INPUT_Q(PCOorCPidentifier): QUEUE
begin
        if /# PCOorCPidentifier is empty #/ then
          return /# default PCO input queue #/;
        else return /# input queue identified by PCOorCPidentifier #/;
end

function OUTPUT_Q(PCOorCPidentifier): QUEUE
begin
        if /# PCOorCPidentifier is empty #/ then
          return /# default PCO output queue #/;
        else return /# output queue identified by PCOorCPidentifier #/;
end
```

```
function SNAPSHOT_FIXED (): BOOLEAN
begin
        if /# all relevant PCO and CP queue(s) have some event(s) on them and all relevant timers have expired #/ then
          return TRUE;
        else return FALSE;
end

function STATEMENT_LINE_TYPE_OF(Alternative): STATEMENT_LINE_TYPE
begin
        return /# EVENT, PSEUDO_EVENT, or CONSTRUCT, as appropriate #/;
end

procedure STOP_TEST_CASE()
begin
        /# stop all running processes #/;
end

procedure procedure TAKE_SNAPSHOT()
        (*  Se toma una instantánea de la cola o colas de PCO y CP a la entrada, la lista de temporizaciones pertinente, y el estado
            de terminación de cualquier otro componente de prueba. El hecho de tomar una instantánea no elimina un evento de
            ningún PCO, CP o lista de temporizaciones.*)
begin
        /# save current PCO and CP input queues in Snapshot #/;
        /# save current timeout list in Snapshot #/;
        /# save current list of terminated Test Components in Snapshot #/;
end

procedure TERMINATE_TEST_CASE()
begin
        if /# any Parallel Test Component processes are still running #/ then
          LOG(TEST_CASE_ERROR);
        STOP_TEST_CASE();
end

function TIMER_OP_TYPE_OF(Alternative): TIMER_OP_TYPE
begin
        return /# START_TIMER, CANCEL_TIMER, or READ_TIMER, as appropriate #/;
end
```

Anexo C

Módulos TTCN

(Este anexo es parte integrante de la Recomendación)

C.1 Introducción

Un módulo TTCN contendrá las siguientes secciones en el orden indicado:

a) parte visión general del módulo TTCN;
b) parte importación;
c) parte declaraciones;
d) parte constricciones;
e) parte dinámica.

C.2 Parte visión general del módulo TTCN

C.2.1 Introducción

La finalidad de la parte visión general del módulo TTCN es proporcionar información precisa para la utilización del módulo por otros módulos o series de pruebas. Incluye:

a) exportaciones del módulo TTCN;
b) estructura del módulo TTCN;
c) Índice de casos de prueba;
d) Índice de pasos de prueba;
e) índice de valores por defecto.

C.2.2 Exportaciones del módulo TTCN

El cuadro Exportaciones del módulo TTCN identifica el módulo y proporciona información sobre el objetivo global del módulo TTCN (por ejemplo, la biblioteca de constricciones para un protocolo concreto).

Si el objeto es importado se da el nombre del objeto fuente original.

Si el objeto se declara como un objeto externo (externo explícitamente) o si se trata de un objeto que se omite en el objeto fuente importado (externo implícitamente), en lugar del nombre del objeto fuente se da la palabra clave EXTERNAL.

La exportación de un objeto del tipo Enumeration o Named Number requiere que esté dado el tipo correspondiente. Los otros objetos que se definen en el tipo correspondiente no se exportan también. Sin embargo, éstos son exportados implícitamente y pueden ser referenciados en otros objetos exportados. El nombre de tipo se da como un sufijo del nombre del objeto incrustado entre corchetes.

Se proporcionará la siguiente información de las exportaciones del módulo TTCN:

a) nombre del módulo TTCN;
b) una descripción del objetivo del módulo;
c) una referencia completa del módulo TTCN;
d) referencias a las normas de base pertinentes si existen;
e) una referencia al formulario de PICS si existe;
f) una referencia al formulario de PIXIT si existe;
g) una indicación del método o métodos de prueba si existen;
h) la inclusión como comentario de otras informaciones que puedan ayudar en la comprensión del módulo TTCN;
i) una lista de los objetos exportados,

   donde se facilitará la siguiente información para cada objeto exportado:

   1) El nombre del objeto.

      Si el objeto es del tipo NamedNumber o Enumeration, el tipo correspondiente se da como un sufijo del nombre del objeto insertado entre corchetes.

2) El tipo de objeto.

3) El nombre del objeto fuente original si el objeto es importado, o la instrucción de objeto EXTERNAL.

4) Un número de página,

que proporcione la ubicación del objeto en el módulo (no se dará ningún número de página para los objetos importados).

Esta información se proporcionará con el formato del formulario siguiente.

Exportaciones del módulo TTCN				
Nombre del módulo TTCN : *TTCN_ModuleIdentifier*				
Objetivo : *[FreeText]*				
Ref. del módulo TTCN : *[FreeText]*				
Ref. de normas : *[FreeText]*				
Ref. de PICS : *[FreeText]*				
Ref. de PIXIT : *[FreeText]*				
Método(s) de prueba : *[FreeText]*				
Comentario : *[FreeText]*				
Nombre del objeto	**Tipo de objeto**	**Nombre de la fuente**	**N.º de página**	**Comentarios**
⋮ *ObjectIdentifier* ⋮	⋮ *TTCN_ObjectType* ⋮	⋮ *[SourceIdentifier|ObjectDirective]* ⋮	⋮ *Number* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*				

Formulario C.1 – Exportaciones del módulo TTCN

EJEMPLO C.1 – Exportaciones del módulo TTCN:

Exportaciones del módulo TTCN				
Nombre del módulo TTCN : *TTCN_Module_A*				
Objetivo : Ilustrar el uso del cuadro de exportaciones del módulo TTCN				
Ref. del módulo TTCN :				
Ref. del normas :				
Ref. de PICS :				
Ref. de PIXIT :				
Método(s) de prueba :				
Comentarios :				
Nombre del objeto	**Tipo de objeto**	**Nombre de la fuente**	**N.º de página**	**Comentarios**
String5	SimpleType_Object		3	
wait	Timer_Object	Module_B		
INTC	TTCN_PDU_Type_Object		13	
DEF1	Default_Object	TestSuite_1		
TC_2	TestCase_Object	TestSuite_2		
TC_3	TestCase_Object		33	
Preamble	TestStep_Object	EXTERNAL		

C.2.3 Estructura del módulo TTCN

La estructura del módulo TTCN contiene una lista de los grupos de prueba del módulo (si existen). Para cada grupo de prueba se suministrará la siguiente información:

a) la referencia al grupo de prueba,

   donde el primer identificador puede ser el nombre del módulo, y cada identificador sucesivo representa una ordenación conceptual ulterior del módulo;

b) un identificador de expresión de selección optativo;

c) el objetivo del grupo de prueba;

d) un número de página (no se suministrará el número de página para los grupos importados).

Esta información se proporcionará con el formato del formulario siguiente.

Estructura del módulo TTCN			
Referencia al grupo de pruebas	**Referencia al Grupo de prueba**	**Objetivo del grupo de pruebas**	**N.º de página**
⋮ *TestGroupReference* ⋮	⋮ *[SelectExprIdentifier]* ⋮	⋮ *FreeText* ⋮	⋮ *[Number]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario C.2 – Estructura del módulo TTCN

La semántica estática que se describe en "10.3 de Estructura de caso de prueba" es aplicable a la estructura del módulo TTCN.

C.2.4 Índice de casos de prueba

La definición del Índice de casos de prueba para módulos es igual a la del Índice de casos de prueba para series de pruebas.

C.2.5 Índice de pasos de prueba

La definición del Índice de pasos de prueba para módulos es igual a la del Índice de pasos de prueba para series de pruebas.

C.2.6 Índice de valores por defecto

La definición del índice de valores por defecto para módulos es igual a la del índice de valores por defecto para series de pruebas.

C.3 Parte importación

C.3.1 Introducción

La finalidad de la parte importación de un módulo es declarar los objetos que no están definidos explícitamente pero que se han utilizado. Estos objetos son declarados como objetos externos o son importados de otros objetos fuente. Esta parte incluye:

a) externos;

b) importación.

C.3.2 Externos

El cuadro Objetos externos presenta una relación de los objetos referenciados por su identificador en el módulo TTCN, pero que no son importados ni están definidos explícitamente. Un objeto externo permite al importador conocer lo que tiene que definir cuando se importa el módulo TTCN.

Para cada objeto externo se proporcionará la siguiente información:

a) el identificador del objeto y los parámetros,

   los parámetros están incluidos cuando el objeto es una operación serie de pruebas, una constricción o un paso de prueba;

b) el tipo de objeto;

c) un comentario optativo.

Esta información se proporcionará con el formato del formulario siguiente.

Objetos externos		
Nombre del objeto	**Tipo de objeto**	**Comentarios**
⋮ *Identifier | TS_OpId&ParList | ConsId&ParList | TestStepId&ParList* ⋮	⋮ *TTCN_ObjectType* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*		

Formulario C.3 – Objetos externos

EJEMPLO C.2 – Objetos externos:

Objetos externos (External)		
Nombre del objeto	**Tipo de objeto**	**Comentarios**
CRC(P:A_PDU) CONSTRAINT_A(acstr:t_CONNECT) TESTSTEP_A(I:INTEGER) DEF3	TS_Op_Object TTCN_PDU_Constraint_Object TestStep_Object Default_Object	

C.3.3 Importación

La definición de la importación para módulos es igual a la definición de la importación para series de pruebas (véase 10.8).

Anexo D

Índice de serie de pruebas

(Este anexo es parte integrante de la Recomendación)

Vacío.

Anexo E

Formularios compactos

(Este anexo es parte integrante de la Recomendación)

E.1 Introducción

Es posible, facultativamente, imprimir, en un cuadro único muchas constricciones y/o casos de prueba. Esto puede ser útil para destacar las relaciones entre constricciones individuales y/o casos de prueba individuales. En este anexo se establecen los requisitos de utilización de formularios compactos de constricciones y de casos de prueba y se facilitan algunos ejemplos. Dichos formularios son específicos y difieren de las presentaciones generalizadas descritas en 7.3. Los nuevos formularios constituyen simplemente una forma alternativa de presentación de la misma información, por lo que no tienen asociada ninguna TTCN.MP. La información contenida en un cuadro de constricciones compactas y/o de casos de prueba compactos puede trasladarse a la TTCN.MP asociada con los numerosos cuadros de constricciones y/o casos de prueba que tengan el mismo contenido de información.

E.2 Formularios compactos para constricciones

E.2.1 Requisitos

Solo se permitirá la impresión de numerosos cuadros de constricciones individuales en forma de cuadro compacto de constricciones único si:

a) las constricciones tienen el mismo tipo de ASP, tipo de PDU, tipo estructurado o tipo ASN.1;

b) no hay información de codificación especificada en ninguno de los encabezamientos de los cuadros de constricción únicos ni en la columna de codificación de ninguno de esos cuadros (las codificaciones ASN.1 especificadas en ASN.1 Value pueden, sin embargo, especificarse en formularios compactos); y

c) no hay inscripciones en la columna de comentarios de ninguno de los cuadros de constricciones individuales.

NOTA – Si los cuadros de constricciones individuales solamente tienen comentarios en el pie de comentarios detallados (es decir, que la columna de comentarios está en blanco), es posible imprimir estas constricciones en formato compacto. En tales casos, los comentarios detallados de los formularios individuales deberán reunirse e imprimirse como un solo comentario en el pie de comentarios detallados del formulario compacto.

E.2.2 Formularios compactos para las constricciones de ASP

Cuando una constricción contenga tan sólo unos pocos parámetros o cuando el número de constricciones sea reducido, las constricciones podrán presentarse en la versión compacta del formulario de constricciones de ASP:

Declaraciones de constricciones de ASP					
Tipo de ASP : *ASP_Identifier*					
Nombre de la constricción	**Trayecto de derivación**	**Nombre del parámetro**			**Comentarios**
		ASP_ParIdentifier$_1$		*ASP_ParIdentifier*$_n$	
ConsId-&ParList$_1$	*Derivation-Path*$_1$	*ConstraintValue-&Attributes*$_{1, 1}$		*ConstraintValue-&Attributes*$_{1, n}$	*[FreeText]*$_1$
ConsId-&ParList$_2$	*Derivation-Path*$_2$	*ConstraintValue-&Attributes*$_{2, 1}$		*ConstraintValue-&Attributes*$_{2, n}$	*[FreeText]*$_2$
⋮	⋮	⋮		⋮	⋮
ConsId-&ParList$_m$	*Derivation-Path*$_m$	*ConstraintValue-&Attributes*$_{m, 1}$		*ConstraintValue-&Attributes*$_{m, n}$	*[FreeText]*$_m$

Formulario E.1: Declaraciones de constricciones de ASP (compactas)

Este formulario se utiliza para las ASP y sus parámetros de la misma forma en que se utiliza el formulario de declaraciones de constricciones de PDU para las PDU y sus campos (véase E.2.3).

E.2.3 Formularios compactos para constricciones de PDU

E.2.3.1 Introducción

Cuando una constricción contenga tan solo unos pocos campos o cuando el número de constricciones sea reducido, las constricciones podrán presentarse en la versión compacta del formulario de constricciones de PDU:

Declaraciones de constricciones de PDU					
Tipo de PDU : *PDU_Identifier*					
Nombre de la constricción	**Trayecto de derivación**	**Nombre del campo**			**Comentarios**
		ASP_ParIdentifier$_1$		*ASP_ParIdentifier*$_n$	
ConsId-&ParList$_1$	*Derivation-Path*$_1$	*ConstraintValue-&Attributes*$_{1, 1}$		*ConstraintValue-&Attributes*$_{1, n}$	*[FreeText]*$_1$
ConsId-&ParList$_2$	*Derivation-Path*$_2$	*ConstraintValue-&Attributes*$_{2, 1}$		*ConstraintValue-&Attributes*$_{2, n}$	*[FreeText]*$_2$
⋮	⋮	⋮		⋮	⋮
ConsId-&ParList$_m$	*Derivation-Path*$_m$	*ConstraintValue-&Attributes*$_{m, 1}$		*ConstraintValue-&Attributes*$_{m ,n}$	*[FreeText]*$_m$

Formulario E.2: Declaraciones de constricciones de PDU (compactas)

El formulario de constricciones compacto tiene los nombres de los campos a todo lo largo de su parte superior, mientras que las diferentes instancias de constricciones figuran en las filas del mismo. Si en la definición de tipo de la PDU hay *n* campos, habrá *n* columnas de campo en el formulario de constricciones compactas.

La columna de trayecto de derivación es optativa; sin embargo, se utilizará para especificar el trayecto de derivación de las constricciones modificadas (véase 13.6). Un cuadro compacto puede reunir varias constricciones de base (como se ilustra en el ejemplo C.1) o puede reunir una constricción de base y sus constricciones modificadas, como se indica en el ejemplo C.2. Cuando en un cuadro compacto se declaren constricciones modificadas, los campos no modificados aparecerán en las constricciones modificadas como casillas en blanco en la intersección de la fila de constricciones modificadas con la columna de campo. Cuando se establezca la correspondencia entre un cuadro compacto y la TTCN.MP (es decir, formato único), se omitirán los campos en blanco heredados. En las constricciones modificadas se dejarán en blanco los campos no especificados en dichas constricciones.

EJEMPLO E.1 – Constricciones que utilizan el formulario de constricciones compacto:

Definición de tipo PDU		
Nombre de la PDU : PDU_B **Tipo de PCO** : XSAP **Comentario** :		
Nombre del campo	**Tipo del campo**	**Comentarios**
FIELD1 FIELD2 FIELD3	INTEGER BOOLEAN IA5String	

EJEMPLO E.1.1 – Dada la declaración de PDU_B en la forma:

Definición de tipo PDU		
Nombre de la PDU : PDU_B **Tipo de PCO** : XSAP **Comentario** :		
Nombre del campo	**Tipo del campo**	**Comentarios**
FIELD1 FIELD2 FIELD3	INTEGER BOOLEAN IA5String	

EJEMPLO E.1.2 – Las constricciones impuestas a PDU_B con el formulario de constricciones compacto, podrían ser:

Declaraciones de constricciones de PDU				
Tipo de PDU : PDU_B				
Nombre de la constricción	**Nombre del campo**			**Comentarios**
	FIELD1	FIELD2	FIELD3	
CN1 CN2 CN3	3 (4,5,6) 0	TRUE FALSE ?	"A string" "A string" –	

La referencia a constricciones en la parte dinámica podrá contener entonces inscripciones tales como PDU_B[CN1] y PDU_B[CN2].

EJEMPLO E.1.3 – Mecanismo de herencia que utiliza el formulario de constricciones compacto:

Declaraciones de constricciones de PDU						
Tipo de PDU : PDU_A						
Nombre de la constricción	**Trayecto de derivación**	**Nombre del campo**				**Comentarios**
		FIELD1	FIELD2	FIELD3	FIELD4	
CN0 CN1 CN2	 CN0. CN0. CN.	0 1	'FF'H –	'00'B ?	TRUE	

E.2.3.2 Constricciones compactas parametrizadas

Las constricciones compactas también pueden estar parametrizadas. En tales casos, las listas de parámetros deberán añadirse al nombre de la constricción y aparecer en la columna de nombre de la constricción de los formularios de constricciones compactas.

EJEMPLO E.2 – Constricción compacta parametrizada:

Declaraciones de constricciones de PDU			
Tipo de PDU : PDU_X			
Nombre de la constricción	**Nombre del campo**		**Comentarios**
	P1	P2	
S1	0	0	
S2	0	1	
S3	1	0	
S4	1	1	
S5(A:INTEGER)	1	A	

La invocación de las constricciones impuestas a PDU_X en un paso de prueba puede efectuarse como sigue: S1, S2, S3, S4, S5(0), S5(1) o S5(Var) siendo Var un caso de prueba o una variable de caso de prueba.

E.2.4 Formularios compactos para constricciones de tipo estructurado

Las constricciones compactas de tipo estructurado se proporcionarán en el formulario siguiente:

Declaraciones de constricciones de tipo estructurado					
Tipo de estructurado : *StructIdentifier*					
Nombre de la constricción	**Trayecto de derivación**	**Nombre del campo**			**Comentarios**
		ASP_ParIdentifier$_1$		*ASP_ParIdentifier*$_n$	
ConsId-&ParList$_1$	*Derivation-Path*$_1$	*ConstraintValue-&Attributes*$_{1, 1}$		*ConstraintValue-&Attributes*$_{1, n}$	*[FreeText]*$_1$
ConsId-&ParList$_2$	*Derivation-Path*$_2$	*ConstraintValue-&Attributes*$_{2, 1}$		*ConstraintValue-&Attributes*$_{2, n}$	*[FreeText]*$_2$
⋮	⋮	⋮		⋮	⋮
ConsId-&ParList$_m$	*Derivation-Path*$_m$	*ConstraintValue-&Attributes*$_{m, 1}$		*ConstraintValue-&Attributes*$_{m, n}$	*[FreeText]*$_m$

Formulario E.3: Declaraciones de constricciones de tipo estructurado compactas

EJEMPLO E.3 – Utilización de constricciones compactas estructuradas

La PDU_Y consta de cinco campos, denominados Y1, a Y5. Los campos Y1, Y2 e Y3 se han combinado en el tipo estructurado denominado A. El primer cuadro de los que figuran a continuación representa las constricciones definidas sobre PDU_Y. Los cuadros segundo y tercero contienen la misma información que el último cuadro.

Los cuadros segundo y tercero muestran la especificación de las constricciones en las A de tipo estructurado, empleando los formularios de constricción individual, en tanto que el último cuadro representa la constricción de A empleando el formulario de constricción compacta. En ambos, se utiliza también el mecanismo de modificación.

De los cuadros que siguen se desprende que si se utilizara la constricción YY1, los valores de los campos Y1 a Y5 serían 0, 0, 0, 0, 1 respectivamente, habiéndose derivado los valores de los campos Y1 e Y3 del tipo estructurado A, mediante la constricción A1. Si se hubiera utilizado la constricción YY2, los valores de Y1 a Y5 serían 0, 3, 0, 1, 0 respectivamente, habiéndose derivado los valores de los campos Y1 a Y3 del tipo estructurado A, mediante la constricción A2.

EJEMPLO E.3.1 – Cuadro de constricciones de PDU que utiliza un tipo estructurado (denominado A):

Declaraciones de constricciones de PDU				
Tipo de PDU : PDU_Y				
Nombre de la constricción	**Nombre del campo**			**Comentarios**
	A	Y4	Y5	
YY1 YY2 YY3	A1 A2 A2	0 1 0	1 0 1	

EJEMPLO E.3.2 – A1 es una constricción de base de tipo estructurado A:

Declaración de constricciones de tipo estructurado		
Nombre de la constricción : A1 **Tipo estructurado** : A **Trayecto de derivación** : **Comentarios** :		
Nombre del elemento	**Valor del elemento**	**Comentarios**
Y1 Y2 Y3	0 0 0	

EJEMPLO E.3.3 – La constricción de tipo estructurado A2 es una constricción modificada derivada de A1:

Declaración de constricción de tipo estructurado		
Nombre de la constricción : A2 **Tipo estructurado** : A **Trayecto de derivación** : A1. **Comentarios** :		
Nombre del elemento	**Valor del elemento**	**Comentarios**
Y2	3	

EJEMPLO E.3.4 – Constricciones A1 y A2 de tipo estructurado A en forma compacta:

Declaraciones de constricciones de tipo estructurado					
Nombre del tipo estructurado : A					
Nombre de la constricción	**Trayecto de derivación**	**Nombre del elemento**			**Comentarios**
		Y1	Y2	Y3	
A1 A2	 A1.	0	0 3	0	

Cuando se utilizan tipos estructurados dentro de las declaraciones de constricciones de PDU cada uno de los nombres de campo empleados en la definición de tipo estructurado concordará exactamente con el nombre (o nombre abreviado, si se han definido tanto el nombre abreviado como el nombre completo) del campo de la PDU al que representa, a partir de la definición de tipo PDU original.

E.2.5 Formularios compactos para constricciones en ASN.1

Para las definiciones de constricciones de ASP en ASN.1, PDU en ASN.1 y Tipo ASN.1 en forma compacta se utilizarán respectivamente los formularios siguientes.

Declaraciones de constricciones de ASP en ASN.1	
Tipo de ASP : *ASP_Identifier*	
Nombre de la constricción	**Valor en ASN.1**
ConsId&ParList$_1$	*ConstraintValue&Attributes*$_1$
.	.
.	.
.	.
ConsId&ParList$_m$	*ConstraintValue&Attributes*$_m$

Formulario E.4: Declaraciones de constricciones de ASP en ASN.1 (compactas)

Declaraciones de constricciones de PDU en ASN.1	
Tipo de PDU : *PDU_Identifier*	
Nombre de la constricción	**Valor en ASN.1**
ConsId&ParList$_1$	*ConstraintValue&Attributes*$_1$
.	.
.	.
.	.
ConsId&ParList$_m$	*ConstraintValue&Attributes*$_m$

Formulario E.5: Declaraciones de constricciones de PDU en ASN.1 (compactas)

Declaraciones de constricciones de tipo en ASN.1	
Nombre de tipo : *ASN1_TypeIdentifier*	
Nombre de la constricción	**Valor en ASN.1**
ConsId&ParList$_1$	*ConstraintValue&Attributes*$_1$
.	.
.	.
.	.
ConsId&ParList$_m$	*ConstraintValue&Attributes*$_m$

Formulario E.6: Declaraciones de constricciones de tipo en ASN.1 (compactas)

E.3 Formulario compacto para casos de prueba

E.3.1 Requisitos

Sólo se permite la impresión de numerosos cuadros de comportamiento dinámico de casos de prueba individuales en un solo cuadro de comportamiento dinámico de caso de prueba compacto cuando se apliquen las siguientes normas:

a) todos los cuadros de comportamiento dinámico de casos de prueba individuales deberán pertenecer al mismo grupo de prueba;

b) todos los cuadros de comportamiento dinámico de casos de prueba individuales deberán tener el mismo árbol por defecto o no tener ninguno. Se recomienda que no haya árbol por defecto;

c) la descripción de comportamiento indicada en cada cuadro de comportamiento dinámico de caso de prueba individual deberá contener un constructivo ATTACH único.

E.3.2 Formulario compacto para comportamiento dinámico de casos de prueba

Cuando una serie de casos de prueba tengan esencialmente el mismo comportamiento dinámico y solamente haya diferencias en las constricciones referenciadas (por ejemplo, pruebas de variaciones de los parámetros de las ASP y/o PDU), podrán presentarse los casos de prueba en la versión compacta del formulario de comportamiento dinámico del caso de prueba.

Comportamientos dinámicos de casos de prueba			
Grupo : *TestGroupReference* **Valor por defecto** : *DefaultReference*			
Nombre del caso de prueba	**Finalidad**	**Adjunción de caso de prueba**	**Comentarios**
⋮ *TestCaseIdentifier* ⋮	⋮ *FreeText* ⋮	⋮ *Attach* ⋮	⋮ *[FreeText]* ⋮
Comentarios detallados: *[FreeText]*			

Formulario E.7: Comportamientos dinámicos de casos de prueba (compactos)

Cada fila del cuerpo de este formulario describe un solo caso de prueba. Si se utiliza el formulario del caso de prueba compacto, el cuadro único sustituye a una serie de cuadros de comportamiento dinámico de caso de prueba en la parte de comportamiento de la serie de pruebas.

La columna de comentarios contiene comentarios pertenecientes a los casos de prueba individuales frente a cada adjunción.

Los casos de prueba contenidos en un formulario de caso de prueba compacta pueden formar un subconjunto de su grupo y aparecerán en el orden indicado en el Índice de casos de prueba.

EJEMPLO E.4 – Cuadro de caso de prueba compacto que define una serie de pruebas para FTAM:

Comportamientos dinámicos de casos de prueba		
Grupo : R/BV/PV/LM/CR/OV **Valor por defecto** :		
Nombre del caso de prueba	**Finalidad**	**Adjunción del paso de prueba**
OVERIDE1	Omitir el parámetro contraordenación, cuando existe el fichero.	+OVERRIDE (FCRERQ_001, FCRERP_001)
OVERIDE2	Omitir el parámetro contraordenación, cuando no existe el fichero	+OVERRIDE (FCRERQ_002, FCRERP_002)

Anexo F

Ejemplos

(Este anexo no es parte integrante de la Recomendación)

F.1 Ejemplos de constricciones en forma de cuadro

F.1.1 Definiciones de ASP y PDU

F.1.1.1 Definición de tipo plano

Definición de tipo de PDU		
Nombre de PDU : T_CONNECT1 **Tipo de PCO** : **Comentarios** : Ilustración de mecanismos de TTCN		
Nombre del campo	**Tipo de campo**	**Comentarios**
Source Destination T_Class UserData	BITSTRING [4] BITSTRING [4] INTEGER0to4 IA5String	Longitud de 4 bits Longitud de 4 bits Definido como tipo simple

F.1.1.2 Definición de tipo estructurado

Definición de tipo de PDU		
Nombre de PDU : T_CONNECT2 **Tipo de PCO** : **Comentarios** : Ilustración de mecanismo de TTCN		
Nombre del campo	**Tipo de campo**	**Comentarios**
T_Addresses T_Class UserData	T_AddressInfo INTEGER0to4 IA5String	Definido como tipo simple

Definiciones de tipo estructurado		
Nombre del tipo : T_AddressInfo **Comentarios** : Se puede utilizar en todos los ejemplos de PDU de transporte		
Nombre del campo	**Definición del campo**	**Comentarios**
Source Destination	BITSTRING[4] BITSTRING[4]	Longitud de 4 bits Longitud de 4 bits

F.1.1.3 PDU de tipo especial para permitir la utilización de encadenamiento (estático) de constricciones

Definición de tipo de ASP		
Nombre de ASP : N_DATArequest **Tipo de PCO** : N_SAP **Comentarios** : Para permitir el encadenamiento de constricciones		
Nombre del parámetro	**Tipo de parámetro**	**Comentarios**
CallingNetworkAddress CalledNetworkAddress ConnectionIdentifier Data	HEXSTRING HEXSTRING HEXSTRING PDU	 Para permitir el encadenamiento de constricciones

F.1.2 Constricciones de ASP/PDU

F.1.2.1 Plano

Declaración de constricción de PDU		
Nombre de la constricción : TCON_Class4_1 **Tipo de PDU** : T_CONNECT1 **Trayecto de derivación** : **Comentarios** :		
Nombre del campo	**Valor del campo**	**Comentarios**
Source Destination T_Class UserData	TS_Par1 TS_Par2 4 "testing, testing"	

F.1.2.2 Estructurado, referente a grupos de campo

Declaración de constricción de PDU		
Nombre de la constricción : TCON_Class4_2 **Tipo de PDU** : T_CONNECT2 **Trayecto de derivación** : **Comentarios** :		
Nombre del campo	**Valor del campo**	**Comentarios**
T_Addresses T_Class UserData	WrongAddress 4 "one, two, three"	WrongAddress es una referencia a una constricción de tipo estructurado

Declaración de constricción de tipo estructurado		
Nombre de la constricción : WrongAddress		
Tipo de ASP : T_AddressInfo		
Trayecto de derivación :		
Comentarios :		
Nombre del parámetro	**Valor del parámetro**	**Comentarios**
Source Destination	TS_Par1 '0000'B	

F.1.2.3 Encadenamiento, de utilidad para PDU (anidadas) en las ASP

Declaración de constricción de ASP		
Nombre de la constricción : N_DATAreq_With_T_CON_Class4_1		
Tipo de ASP : N_DATArequest		
Trayecto de derivación :		
Comentarios : TCON_Class4_1 es una constricción de PDU (esto es, un encadenamiento)		
Nombre del parámetro	**Valor del parámetro**	**Comentarios**
CallingNetworkAddress CalledNetworkAddress ConnectionIdentifier Data	TS_Par3 TS_Par4 'ABCDEF'H TCON_Class4_1	

F.1.2.4 Constricciones parametrizadas

Es posible parametrizar constricciones planas, estructuradas y encadenadas. En los siguientes ejemplos se muestra la parametrización utilizada para pasar un valor:

Declaración de constricción de PDU		
Nombre de la constricción : TCON_1(class:INTEGER)		
Tipo de PDU : T_CONNECT1		
Trayecto de derivación :		
Comentarios :		
Nombre del campo	**Valor del campo**	**Comentarios**
Source Destination T_Class UserData	'1000'B ? class ?	 Clase es un parámetro formal

A esto puede hacerse referencia desde el caso de prueba, desde el paso de prueba o desde cuadros de comportamiento por defecto, como por ejemplo:

TCON_1(4) o TCON_1(TCvariable)

Los valores de campo pueden ser PDU (encadenadas) completas:

Declaración de constricción de ASP		
Nombre de la constricción	**:**	N_DATAreq_With_T_CON(A_Constraint:T_CONNECT2)
Tipo de ASP	**:**	N_DATArequest
Trayecto de derivación	**:**	
Comentarios	**:**	TCON_Class4_1 es una constricción de PDU (esto es, un encadenamiento)

Nombre del parámetro	**Valor del parámetro**	**Comentarios**
CallingNetworkAddress	TS_Par3	
CalledNetworkAddress	TS_Par4	
ConnectionIdentifier	'1234567'H	
Data	A_Constraint	A_Constraint es un parámetro formal

Esta constricción se puede invocar, por ejemplo, de la siguiente manera:

N_DATAreq_With_TCON(TCON_Class4_2)

Como el parámetro real es un nombre de constricción que a su vez, se puede parametrizar, es posible expresar una profundidad cualquiera de anidamiento de las PDU.

F.1.2.5 Constricciones modificadas

Es posible utilizar las constricciones existentes y modificarlas para definir nuevas constricciones. Esto puede efectuarse con constricciones planas, estructuradas y parametrizadas:

Declaración de constricción de PDU		
Nombre de la constricción	**:**	TCON_CLASS0_1
Tipo de PDU	**:**	T_CONNECT1
Trayecto de derivación	**:**	TCON_Class4_1.
Comentarios	**:**	Class 0 es aceptable

Nombre del campo	**Valor del campo**	**Comentarios**
T_Class	0	

Se pueden usar comodines para valores:

Declaración de constricción de PDU		
Nombre de la constricción	**:**	TCON_AnyClass
Tipo de PDU	**:**	T_CONNECT1
Trayecto de derivación	**:**	TCON_Class4_1.
Comentarios	**:**	Any class (0..4) is acceptable.

Nombre del campo	**Valor del campo**	**Comentarios**
T_Class	?	

Se considera, sin embargo, que este estilo no es bueno. Es mejor utilizar como base la constricción más general.

También es posible suprimir campos completos:

Declaración de constricción de PDU		
Nombre de la constricción : TCON_Erroneous_NoClass		
Tipo de PDU : T_CONNECT1		
Trayecto de derivación : TCON_Class4_1.		
Comentarios : No hay presente ninguna clase		
Nombre del campo	**Valor del campo**	**Comentarios**
T_Class	–	T_Class omitted

F.2 Ejemplos de constricciones en ASN.1

F.2.1 Definiciones de ASP y PDU

F.2.1.1 Plana

Definición de tipo de PDU en ASN.1
Nombre de PDU : T_CONNECT1 **Tipo de PCO** : **Comentarios** :
Definición de tipo
-- sólo para ilustrar la utilización de ASN.1 en TTCN SEQUENCE { source BITSTRING (SIZE (4..4)), Destination BITSTRING (SIZE (4..4)), t_Class INTEGER (0..4), userData IA5String OPTIONAL. }

F.2.1.2 Estructurada

Definición de tipo de PDU en ASN.1
Nombre de PDU : T_CONNECT2 **Tipo de PCO** : **Comentarios** :
Definición de tipo
-- sólo para ilustrar la utilización de ASN.1 en TTCN SEQUENCE { t_Addresses T_AddressInfo, t_Class INTEGER (0..4), userData IA5String } *-- la expansión de T_AddressInfo figura en su propio cuadro*

Las producciones en ASN.1 conexas, que figuran normalmente en un solo módulo ASN.1, pueden estar distribuidas en más cuadros en TTCN:

Definición de tipo en ASN.1			
Nombre de tipo	**:** T_AddressInfo		
Comentarios	**:**		
Definición de tipo			
SEQUENCE	{	source	BITSTRING (SIZE (4..4)),
		destination	BITSTRING (SIZE (4..4)),
	}		

F.2.1.3 Definición de ASP

Definición de tipo de ASP en ASN.1				
Nombre de ASP	**:** N_DATArequest			
Tipo de PCO	**:** N_SAP			
Comentarios	**:**			
Definición de tipo				
SEQUENCE	{	callingNetworkAddress	OCTETSTRING,	-- *número par de octetos*
		calledNetworkAddress	OCTETSTRING,	-- *número par de octetos*
		connectionIdentifier	OCTETSTRING,	-- *número par de octetos*
		Data	T_PDUS	
	}			

Definición de tipo en ASN.1			
Nombre de tipo	**:** T_PDUS		
Comentarios	**:**		
Definición de tipo			
CHOICE	{	t1 T_CONNECT1.	
		t2 T_CONNECT2	
		}	

F.2.2 Constricciones de ASP/PDU en ASN.1

F.2.2.1 Plana

Declaraciones de constricción de PDU en ASN.1		
Nombre de la constricción	**:**	TCON_Class4_1
Tipo de PDU	**:**	T_CONNECT1
Trayecto de derivación	**:**	
Comentarios	**:**	

Valor de la constricción	
{ source	TS_PAR1,
	TS_PAR2, -- *el identificador de campo se puede omitir, si se desea*
t_Class	4,
userData	"testing, testing"
}	

F.2.2.2 Estructurada

Declaraciones de constricción de PDU en ASN.1		
Nombre de la constricción	**:**	TCON_Class4_2
Tipo de PDU	**:**	T_CONNECT2
Trayecto de derivación	**:**	
Comentarios	**:**	

Valor de la constricción	
{ t_Addresses	WrongAddress, -- *referencia a una constricción de campo de PDU*
t_Class	4,
userData	"one, two, three"
}	

Declaración de constricción de PDU en ASN.1		
Nombre de la constricción	**:**	WrongAddress
Tipo de PDU	**:**	T_AddressInfo
Trayecto de derivación	**:**	
Comentarios	**:**	

Valor de la constricción	
{ source	TS_PAR1,
destination	'0000'B
}	

F.2.2.3 Encadenamiento de una constricción de PDU

Declaraciones de constricción de ASP en ASN.1		
Nombre de la constricción	**:**	N_DATAreq_With_TCON_Class4_1
Tipo de la ASP	**:**	N_DATArequest
Trayecto de derivación	**:**	
Comentarios	**:**	

Valor de la constricción

```
{  callingNetworkAddress      TS_PAR_3,
   calledNetworkAddress       TS_PAR_4,
   connectionIdentifier       'ABCDEF'H,
   data                       t1  TCON_Class4_1  -- encadenamiento a una constricción de PDU
}
```

F.2.2.4 Constricciones parametrizadas

Las constricciones en ASN.1 pueden ser parametrizadas como constricciones en TTCN en forma de tabla, por ejemplo:

Declaración de constricción de PDU en ASN.1		
Nombre de la constricción	**:**	TCON_1(class:INTEGER)
Tipo de PDU	**:**	T_CONNECT1
Trayecto de derivación	**:**	
Comentarios	**:**	

Valor de la constricción

```
{  source          '0000'B,
   destination                ?,        -- comodín
   t_Class         class,                -- parámetro formal
   userData        ?
}
```

A esto puede hacerse referencia desde el caso de prueba, desde el paso de prueba o desde los cuadros de comportamiento por defecto, como por ejemplo:

TCON_1(4) o TCON_1(TCvariable)

Un parámetro puede también representar una PDU encadenada completa:

Declaración de constricción de ASP en ASN.1		
Nombre de la constricción	**:**	N_DATAreq_With_TCON(a_constraint:T_CONNECT2)
Tipo de ASP	**:**	N_DATArequest
Trayecto de derivación	**:**	
Comentarios	**:**	

Valor de la constricción

```
{  callingNetworkAddress      TS_PAR_3,
   calledNetworkAddress       TS_PAR_4,
   connectionIdentifier       '1234567'H,
   data                       t2 a_constraint
                    -- a_constraint es un parámetro formal que contiene una PDU completa.
}
```

A esto puede hacerse referencia desde el caso de prueba, el paso de prueba o desde los cuadros de comportamiento por defecto, como por ejemplo:

N_DATAreq_With_TCON(TCON_Class4_2)

Como el parámetro real es un nombre de constricción que puede, a su vez, ser parametrizado, es posible expresar una profundidad cualquiera de anidamiento.

F.2.2.5 Constricciones modificadas

Se puede construir nuevas constricciones modificando constricciones ya definidas mediante el mecanismo REPLACE:

Declaración de constricción de PDU en ASN.1		
Nombre de la constricción	**:**	TCON_Class0_1
Tipo de PDU	**:**	T_CONNECT1
Trayecto de derivación	**:**	TCON_Class4_1.
Comentarios	**:**	
Valor de la constricción		
REPLACE t_Class BY 0		

También es posible utilizar comodines como sustitutos:

Declaración de constricción de PDU en ASN.1		
Nombre de la constricción	**:**	TCON_AnyClass
Tipo de PDU	**:**	T_CONNECT1
Trayecto de derivación	**:**	TCON_Class4_1.
Comentarios	**:**	
Valor de la constricción		
REPLACE t_Class BY ?		

Para especificar campos que hay que omitir se emplea el mecanismo OMIT; esto sólo se permite si el campo se declara como OPTIONAL:

Declaración de constricción de PDU en ASN.1		
Nombre de la constricción	**:**	TCON_NoUserData
Tipo de PDU	**:**	T_CONNECT1
Trayecto de derivación	**:**	TCON_Class4_1.TCON_AnyClass.
Comentarios	**:**	
Valor de la constricción		
OMIT userData		

Es posible modificar constricciones en ASN.1 parametrizadas, pero hay que tener en cuenta que los propios campos parametrizados no se pueden reemplazar:

Declaración de constricción de PDU en ASN.1		
Nombre de la constricción	**:**	TCON_2(class:INTEGER)
Tipo de PDU	**:**	T_CONNECT1
Trayecto de derivación	**:**	TCON_1.
Comentarios	**:**	
Valor de la constricción		
REPLACE userData BY "CPS"		

F.2.3 Otros ejemplos de constricciones en ASN.1

Definición de una PDU FTAM F_INITIALIZEresponse, efectuada en un cuadro de definición de tipo de PDU en ASN.1:

Definición de tipo de PDU en ASN.1		
Nombre de PDU	**:**	F_INITIALIZEresponse
Tipo de PCO	**:**	
Comentarios	**:**	

Definición de tipo

```
SEQUENCE    {
            state_result                        State_Result DEFAULT success,
            action_result                               Action_Result DEFAULT success,
            protocol_version                            Protocol_Version DEFAULT { version_1},
            implementation_information                  Implementation_Information OPTIONAL,
            presentation_context_management     [2] IMPLICIT BOOLEAN DEFAULT FALSE,
            service_class                               Service_Class DEFAULT { transfer_class },
            functional_units                            Functional_Units,
            attribute_groups                            Attribute_Groups DEFAULT { },
            shared_ASE_information                      Shared_ASE_Information OPTIONAL,
            ftam_quality_of_service                     FTAM_Quality_Of_Service,
            contents_type_list                          Contents_Type_List OPTIONAL,
            diagnostic                          Diagnostic OPTIONAL,
            checkpoint_window                   [8] IMPLICIT INTEGER DEFAULT 1
}
```

Los campos de la PDU (State_Result, Action_Result, etc.) se declaran en definiciones de tipo ASN.1.

Por ejemplo, Functional_Units:

Definición de tipo ASN.1	
Nombre de tipo	**:** Functional_Units
Comentarios	**:**
Definición de tipo	
[4] IMPLICIT BITSTRING { read(2), write(3), file_access(4), limited_file_management(5), enhanced_file_management(6), grouping(7), fadu_locking(8), recovery(9), restart_data_transfer(10) }	

Una constricción de base F_INITrsp_001, en la F-INITIALIZEresponse, se declara en la parte constricciones:

Declaración de constricción de PDU en ASN.1		
Nombre de la constricción	**:**	F_INITrsp_001
Tipo de PDU	**:**	F_INITIALIZEresponse
Trayecto de derivación	**:**	
Comentarios	**:**	
Valor de la constricción		
{		
state_result		State_Result_001,
action_result		Action_Result_001,
protocol_version		Protocol_Version_001,
implementation_information		Implementation_Information_001,
presentation_context_management		FALSE,
service_class		Service_Class_001,
functional_units		Functional_Units_001,
attribute_groups		Attribute_Groups_001,
shared_ASE_information		Shared_ASE_Information_001,
ftam_quality_of_service		FTAM_Quality_Of_Service_001,
contents_type_list		Contents_Type_List_001,
diagnostic		Diagnostic_001,
checkpoint_window		1
}		

Una constricción impuesta a Functional_Units, Functional_Units_001, se declara en una declaración de constricción de campo de PDU en ASN.1:

Declaración de constricción de tipo en ASN.1		
Nombre de la constricción	**:**	Functional_Units_001
Tipo estructurado	**:**	Functional_Units
Trayecto de derivación	**:**	
Comentarios	**:**	
Valor de la constricción		
'001'B – Write only		

Una segunda constricción, F_INITrsp_002, puede construirse modificando la constricción de base F_INIT_rsp001:

Declaración de constricción de PDU en ASN.1		
Nombre de la constricción	**:**	F_INITrsp_002
Tipo estructurado	**:**	F_INITIALIZEresponse
Trayecto de derivación	**:**	F_INITrsp_001.
Comentarios	**:**	

Valor de la constricción		
OMIT	implementation_information,	
REPLACE	presentation_context_management	BY TRUE,
REPLACE	functional_units	BY Functional_Units_002,
REPLACE	checkpoint_window	BY ?

siendo Functional_Units_002 una declaración de constricción de PDU en ASN.1.

F.3 Constricciones de base y modificadas

Supóngase que se tiene la siguiente definición de tipo de PDU:

Definición de tipo de PDU		
Nombre de PDU	**:**	PDU_B
Tipo de PCO	**:**	
Comentarios	**:**	Ésta es la declaración de la unidad de datos de protocolo PDU_B

Nombre del campo	**Tipo de campo**	**Comentarios**
FIELD1	INTEGER	
FIELD2	HEXSTRING	
FIELD3	BITSTRING	
FIELD4	BOOLEAN	

Una constricción de base para PDU_B podría ser:

Declaración de constricción de PDU		
Nombre de la constricción : C0		
Tipo de PDU : PDU_B		
Trayecto de derivación :		
Comentarios :		
Nombre del campo	**Valor del campo**	**Comentarios**
FIELD1	0	
FIELD2	'FF'H	
FIELD3	'00'B	
FIELD4	TRUE	

Una constricción modificada, C1, de la constricción de base C0, podría ser:

Declaración de constricción de PDU		
Nombre de la constricción : C1		
Tipo de PDU : PDU_B		
Trayecto de derivación : C0		
Comentarios :		
Nombre del campo	**Valor del campo**	**Comentarios**
FIELD1	1	En la base C0 este valor de campo es 0

Se puede seguir construyendo sobre C1:

Declaración de constricción de PDU		
Nombre de la constricción : C2		
Tipo de PDU : PDU_B		
Trayecto de derivación : C0.C1.		
Comentarios :		
Nombre del campo	**Valor del campo**	**Comentarios**
FIELD2	–	Se omite este campo
FIELD3	?	Se acepta cualquier valor legal

La referencia a una constricción modificada en un árbol de comportamiento se hace mediante Nombre.

F.4 Definición de tipo con macros

Definición de tipo de PDU con el símbolo macro:

Definición de tipo de PDU		
Nombre de PDU : T_CONNECT3 **Tipo de PCO** : **Comentarios** : Ilustración del mecanismo de macro TTCN		
Nombre del campo	**Tipo de campo**	**Comentarios**
<- T_Class UserData	T_AddressGroup INTEGER0to4 IA5String	 Definido como un tipo simple

Definición de tipo estructurado		
Nombre de tipo : T_AddressGroup **Comentarios** :		
Nombre del elemento	**Definición de tipo**	**Comentarios**
Source Destination	BITSTRING [4] BITSTRING [4]	La longitud es 4 bits La longitud es 4 bits

Declaración de constricción de PDU		
Nombre de la constricción : TCON_Class4_3 **Tipo de PDU** : T_CONNECT3 **Trayecto de derivación** : **Comentarios** :		
Nombre del campo	**Valor del campo**	**Comentarios**
<- T_Class UserData	GoodAddress 4 "one, two, three"	Referencia a la declaración de constricción de tipo estructurado

Declaración de constricción de tipo estructurado		
Nombre de la constricción : GoodAddress		
Tipo estructurado : T_AddressGroup		
Trayecto de derivación :		
Comentarios :		
Nombre del elemento	**Valor del elemento**	**Comentarios**
Source	'0101'B	
Destination	'1111'B	

F.5 Utilización de REPEAT

Comportamiento dinámico de caso de prueba					
Nombre del caso de prueba : *RPT_EX2*					
Grupo : *TTCN_EXAMPLES/REPEAT_EXAMPLE2/*					
Finalidad : *Para ilustrar la utilización de REPEAT y el paso de parámetros por sustitución textual*					
Valor por defecto :					
Comentarios :					
N.º	**Etiqueta**	**Descripción y comportamiento**	**Ref. de constricciones**	**Veredicto**	**Comentarios**
1		(FLAG:=FALSE, COUNTER:=0)			
2		!A	A1		
3		REPEAT STEP2 (FLAG, COUNTER)			
		UNTIL [FLAG OR COUNTER=3]			
4		[FLAG]			
5		!D	D1	PASS	
6		[COUNTER=3]			
7		!E	E1	FAIL	
		STEP2 (F:BOOLEAN; NUMBER: INTEGER)			
8		?B (F:=TRUE)	B1		
9		?C (F:=FALSE, NUMBER:=NUMBER+1)	C1		
Comentarios detallados: Este ejemplo muestra cómo la ejecución repetida de STEP2 puede ser terminada ya sea por la recepción del mensaje B o por la recepción de otros tres mensajes. En las líneas que siguen al constructivo REPEAT, se utilizan expresiones booleanas para indicar que, si se recibe B, se habrá de enviar un mensaje D, y que si se reciben los otros tres mensajes se deberá enviar E. Este ejemplo ilustra también el efecto de paso de parámetros por sustitución de textual. Ello significa que FL se sustituye por FLAG, y NUMBER se sustituye por COUNTER, haciendo posible de este modo que FLAG y COUNTER obtengan los resultados de las asignaciones en STEP2.					

F.6 Operaciones series de pruebas

Utilización de una serie de pruebas para establecer una suma de control:

Definición de operación serie de pruebas		
Nombre de la operación	**:**	CRC(P:A_PDU)
Tipo de resultado	**:**	INTEGER
Comentarios	**:**	
Descripción		
Calcular y retornar la suma de control de la PDU P, según el algoritmo CRC. NOTA – En una ATS real esta operación se describirá con mayor detalle.		

Declaración de constricción de PDU		
Nombre de la constricción	**:**	CONS1
Tipo de PDU	**:**	A_PDU
Trayecto de derivación	**:**	
Comentarios	**:**	

Nombre del campo	**Valor del campo**	**Comentarios**
:	:	
Checksum	?	
:	:	
A_PDU.Checksum := CRC(CONS1) en el evento SEND apropiado de una descripción de comportamiento establecerá la suma de control en la constricción CONS1.		

F.7 Ejemplo de visión general de una serie de pruebas

En el cuadro de estructura de serie de pruebas que figura más adelante se define una jerarquía de los grupos y casos de prueba de la serie. En dicha estructura se identifican expresiones de selección de prueba que rigen la selección de los grupos de prueba y de los casos de prueba para ejecución. Por ejemplo, SELEXP_100 es referenciada como la expresión que controla la característica X del protocolo. Si no se soporta la característica X, no se selecciona ninguno de los casos de prueba de la serie que pertenecen al grupo característica X.

Estructura de serie de pruebas		
Nombre de la serie	**:**	TEST_SUITE_A
Ref. de normas	**:**	ISO/CEI xxxx
Ref. de PICS	**:**	ISO/CEI aaaa
Ref. de PIXIT	**:**	ISO/CEI bbbb
Notación(es) de la prueba	**:**	Método de prueba DS
Comentarios	**:**	Esto es sólo un ejemplo

Referencia a grupo de prueba	**Ref. de selección**	**Objetivo de grupo de prueba**	**N.º de pág.**
FEATURE_X	SELEXP_100	Prueba optativa de la característica X	50
FEATURE_A/ATTR_A		Prueba obligatoria del atributo A	50
FEATURE_A/ATTR_A/NEGOTIATION	SELEXP_101	Prueba optativa de negociación del atributo A	50
FEATURE_A/ATTR_A/USAGE		Prueba de la utilización del atributo A	60
FEATURE_A/ATTR_B		Prueba obligatoria de la característica Y	80

Para determinar si se soporta o no la característica X, ha de evaluarse SELEXP_100, lo que se efectúa determinando si el parámetro serie de pruebas en SELEXP_100, es decir, TST_FX, es TRUE. Si lo es, continúa el procesamiento dentro del grupo. Obsérvese que se seleccionarán las pruebas del atributo A (sin expresión), pero que sólo se seleccionarán las pruebas de la característica de negociación optativa del atributo A si SELEXP_101 es TRUE.

Índice de casos de prueba

Referencia a grupo de prueba	Id. de caso de prueba	Ref. de selección	Descripción	N.º de pág.
FEATURE_X/ATTR_A/NEGOTIATION	FX_ANEG_1	SELEXP_102	Pet. atr. A, neg. válida	50
	FX_ANEG_2	SELEXP_102	Pet. atr. A, neg. no válida	52
	FX_ANEG_3		Recep. atr. A, neg. no válida	54
	FX_ANEG_4		Recep. atr. A, neg. no válida	56
FEATURE_X/ATTR_A/USAGE	FX_AUSE_1	SELEXP_103	Utiliz. atr. A (VAL = 0)	60
	FX_AUSE_2		Recep. atr. A	62
	FX_AUSE_3		Recep. atr. A	64

Si se soporta la negociación del atributo A, los casos de prueba comprendidos entre FX_ANEG_01 y FX_ANEG_04 son candidatos a la selección. Sin embargo, solamente se elegirán los casos de prueba "01" y "02" si la expresión de selección adicional SELEXP_102 es TRUE. Solamente se seleccionará el caso de prueba FX_ANEG_01 si el PICS indica que se admite un valor de cero del atributo A.

Las preguntas de PICS y PIXIT utilizadas en las expresiones de selección de prueba se, declaran como parámetros de serie de pruebas.

Declaraciones de parámetros de serie de pruebas

Nombre del parámetro	Tipo	Ref. de PICS/PIXIT	Comentarios
TSP_FX	BOOLEAN	PICS question FX1	¿Característica X soportada?
TSP_FXA_N	BOOLEAN	PICS question FX2	¿Característica X no soportada?
TSP_FXA_NINT	BOOLEAN	PICS question FX3	¿Necesita negociación la IUT?
TSP_FXA_MINVAL	INTEGER	PIXIT question FXVAL	¿Utilizará la IUT VAL = 0?

Las expresiones de selección se declaran como expresiones booleanas, según lo expuesto en 11.5.

Definiciones de expresiones de selección de caso de prueba

Nombre de la expresión	Expresión de selección	Comentarios
SELEXP_100	TSP_FX	Característica X soportada
SELEXP_101	TSP_FXA_N	Negociación de la característica X
SELEXP_102	TSP_FXA_NINIT	Pet. negociación de la característica X
SELEXP_103	TSP_FXA_VAL=0	Aceptación de la característica X VAL = 0

F.8 Ejemplo de un caso de prueba en forma TTCN.MP

Para el caso de prueba simple que figura a continuación:

Comportamiento dinámico del caso de prueba					
Nombre del caso de prueba	**:**	PACKET/P4/PROPER/T_02			
Grupo	**:**	T_7_02			
Finalidad	**:**	Verificar que la IUT acusa recibo de un código 05 de causa de liberación cuando se encuentra en el estado p4			
Valor por defecto	**:**				
Comentarios	**:**				
N.º	**Etiqueta**	**Descripción de comportamiento**	**Ref. de constricciones**	**Veredicto**	**Comentarios**
0		+R1_PREAMBLE(SVC)			
1		+P4D1_PREAMBLE			
2		!CLEAR START TD	CLR_0(LC)		causa de liberación = 5
3	L1	?CLEARC CANCEL TD	CLRC_0(LC)	(PASS)	
4		+R1_POSTAMBLE			
5		?CLEARC CANCEL TD	CLR_L0(LC)	(PASS)	
6		+R1_POSTAMBLE			
7		?RESTART [RST_ON_ERR] CANCEL TD	STRT_DTEA	(PASS)	
8		!RESTARTC	STRTC		
9		+R1_POSTAMBLE			
10		+DIC_UNEXPECTED			
11		->L1			
12		+RSRT_UNEXPECTED			
13		?TIMEOUT TD		FAIL	
14		?OTHERWISE CANCEL TD		FAIL	

La TTCN.MP que corresponde a este cuadro es la siguiente:

$BeginTestCase
$TestCaseId T_7_02
$TestGroupRef PACKET/P4/PROPER/T_02
$TestPurpose /* Verificar que la IUT acusa recibo de un código 05 de causa de liberación cuando está en el estado p4 */
$DefaultsRef
$BehaviourDescription
$BehaviourLine
$Label
$Line **[0] +R1_PREAMBLE(SVC)**
$Cref
$Verdict
$End_BehaviourLine
$BehaviourLine
$Label
$Line **[1] +P4D1_PREAMBLE**
$Cref
$Verdict
$End_BehaviourLine
$BehaviourLine
$Label
$Line **[2] !CLEAR START TD**
$Cref **CLR_0(LC)**
$Verdict
$Comment /* causa de liberación = 5 */
$End_BehaviourLine

$BehaviourLine
$Label **L1**
$Line **[3] ?CLEARC CANCEL TD**
$Cref **CLRC_0(LC)**
$Verdict **(PASS)**
$End_BehaviourLine
$BehaviourLine
$Label
$Line **[4] +R1_POSTAMBLE**
$Cref
$Verdict
$End_BehaviourLine
$BehaviourLine
$Label
$Line **[3] ?CLEAR CANCEL TD**
$Cref **CLR_L0(LC)**
$Verdict **(PASS)**
$End_BehaviourLine
$BehaviourLine
$Label
$Line **[4] +R1_POSTAMBLE**
$Cref
$Verdict
$End_BehaviourLine
$BehaviourLine
$Label
$Line **[3] ?RESTART [RST_ON_ERR] CANCEL TD**
$Cref **STRT_DTEA**
$Verdict **(PASS)**
$End_BehaviourLine
$BehaviourLine
$Label
$Line **[4] !RESTARTC**
$Cref **STRTC**
$Verdict
$End_BehaviourLine
$BehaviourLine
$Label
$Line **[5] +R1_POSTAMBLE**
$Cref
$Verdict
$End_BehaviourLine
$BehaviourLine
$Label
$Line **[3] +D1C_UNEXPECTED**
$Cref
$Verdict
$End_BehaviourLine
$BehaviourLine
$Label
$Line **[4] -> L1**
$Cref
$Verdict
$End_BehaviourLine
$BehaviourLine
$Label
$Line **[3] +RSRT_UNEXPECTED**
$Cref
$Verdict
$End_BehaviourLine
$BehaviourLine
$Label
$Line **[3] ?TIMEOUT TD**
$Cref
$Verdict **FAIL**
$End_BehaviourLine
$BehaviourLine
$Label
$Line **[3] ?OTHERWISE CANCEL TD**
$Cref

$Verdict **FAIL**
$End_BehaviourLine
$End_BehaviourDescription
$End_TestCase

El formato aquí mostrado sólo tiene por objeto facilitar la lectura.

F.9 Utilización de referencia a componentes para la asignación de valores de campo en constricciones

Cuando cierto número de valores de campo en una PDU recibida deben asignarse a los campos de varias PDU que se envían seguidamente, el cuadro de comportamiento dinámico se hace confuso con enunciados de asignación de gran longitud que utilizan la notación de puntos.

La TTCN admite asignaciones de valores de campo de PDU en el cuadro de constricciones con referencias a componentes asociadas con un parámetro formal. Las ASP o PDU recibidas en el cuadro de comportamiento pueden asignarse a una variable y transferirse seguidamente como un parámetro real en la referencia a constricciones a un parámetro formal en el cuadro de constricciones. El cuadro de constricciones especifica entonces las asignaciones de campo requeridas con los parámetros formales y sus componentes. Los cuadros mostrados a continuación ilustran estos principios.

En la figura F.1 se ilustran posibles asignaciones de campo en la especificación de comportamiento sin utilizar referencia a componentes.

Comportamiento dinámico de caso de prueba					
Nombre del caso de prueba : STYLE1					
Grupo : TTCN_EXAMPLES/					
Finalidad : Ilustrar el uso de referencias a componentes en la descripción de comportamiento					
Valor por defecto :					
Comentarios :					
N.º	**Etiqueta**	**Descripción de comportamiento**	**Ref. de constricciones**	**Veredicto**	**Comentarios**
1 2		?InASP(v:=InASP.userdata) !OutASP (OutASP.userdata.OutPDU.FieldA:=v.Field2, OutASP.userdata.OutPDU.FieldC:=v.Field3)	Cin1 Cout1		
Comentarios detallados:					

Figura F.1/X.292 – Enunciados de asignación muy largos que crean confusión en la descripción del comportamiento

En la figura F.2 se ilustra la simplificación de la especificación del comportamiento resultante de la utilización de referencia a componentes en las constricciones.

Comportamiento dinámico de caso de prueba					
Nombre del caso de prueba : TTCN_EXAMPLES/STYLE1					
Referencia : ST_EX1					
Finalidad : Ilustrar la utilización de referencias a componentes en la descripción del comportamiento					
Valor por defecto :					
N.º	**Etiqueta**	**Descripción de comportamiento**	**Ref. de constricciones**	**Veredicto**	**Comentarios**
1 2		?InASP(v:=InASP.userdata) !OutASP	Cin1 Cout2(v)		

Figura F.2/X.292 – Los enunciados de asignación muy largos se han suprimido de la descripción de comportamiento

Para simplificar, se han omitido las definiciones de todos los tipos de ASP y PDU requeridos.

Los tipos de ASP InASP y OutASP se componen del campo de parámetro único datos de usuario, que es del tipo InPDU y OutPDU, respectivamente. InPDU contiene los tres campos Field1, Field2 y Field3, todos ellos del tipo IA5String.

OutPDU contiene los tres campos FieldA, FieldB y FieldC, también todos ellos del tipo IA5String.

v debe declararse como una variable de caso de prueba de un tipo de PDU.

En los cuadros a continuación se presentan las declaraciones de constricción de PDU y ASP requeridas:

Declaración de constricción de ASP		
Nombre de la constricción : Cout1 **Tipo de la ASP** : OutASP **Trayecto de derivación** : **Comentarios** :		
Nombre del parámetro	**Valor del parámetro**	**Comentarios**
userdata	CoutPDU1	

Declaración de constricción de ASP		
Nombre de la constricción : Cout2(p:**PDU**) **Tipo de la ASP** : OutASP **Trayecto de derivación** : **Comentarios** :		
Nombre del parámetro	**Valor del parámetro**	**Comentarios**
userdata	CoutPDU2(p)	

Declaración de constricción de ASP		
Nombre de la constricción : Cin1 **Tipo de la ASP** : InASP **Trayecto de derivación** : **Comentarios** :		
Nombre del parámetro	**Valor del parámetro**	**Comentarios**
userdata	CinPDU	

Declaración de constricción de PDU		
Nombre de la constricción : CoutPDU1		
Tipo de PDU : OutPDU		
Trayecto de derivación :		
Comentarios :		
Nombre del campo	**Valor del campo**	**Comentarios**
FieldA	'A'	
FieldB	'B'	
FieldC	'C'	

Declaración de constricción de PDU		
Nombre de la constricción : CoutPDU2(p:InPDU)		
Tipo de PDU : OutPDU		
Trayecto de derivación :		
Comentarios :		
Nombre del campo	**Valor del campo**	**Comentarios**
FieldA	p.Field2	
FieldB	'B'	
FieldC	p.Field3	

Declaración de constricción de PDU		
Nombre de la constricción : CinPDU		
Tipo de PDU : InPDU		
Trayecto de derivación :		
Comentarios :		
Nombre del campo	**Valor del campo**	**Comentarios**
Field1	*	
Field2	*	
Field3	*	

F.10 Pruebas multiparte

En la figura F.3 se presenta una configuración de componente de prueba para un contexto de prueba multiparte típico. Se muestra solamente un probador superior único, ya que la comunicación entre múltiples probadores de prueba superiores y/o la función de control de probador superior (UTCF, *upper tester control function*) solamente es aplicable en los contextos que utilicen exclusivamente el método de prueba local.

En el ejemplo de la figura F.3, por sencillez, cada probador inferior es especificado por un PTC y la LTCF por el MTC. Se utiliza otro PTC para especificar el probador superior. Se utilizan puntos de coordinación entre los PTC del probador inferior y el MTC.

Un uso claro de la concurrencia es satisfacer los requisitos de las pruebas multiparte, pero no debe aplicarse para significar que habrá una relación exacta individualizada (de uno a uno) entre los probadores inferiores y los PTC, o entre la LTCF y el MTC, o entre el probador superior y un PTC.

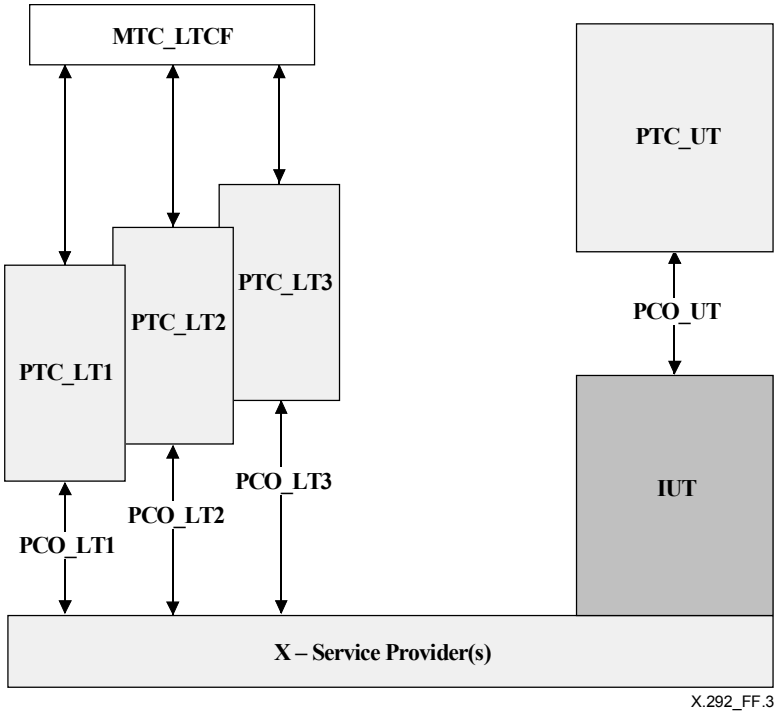


Figura F.3/X.292 – Ejemplo de configuración de componente de prueba para el contexto de pruebas multiparte con un solo probador superior

F.11 Multiplexación/demultiplexación

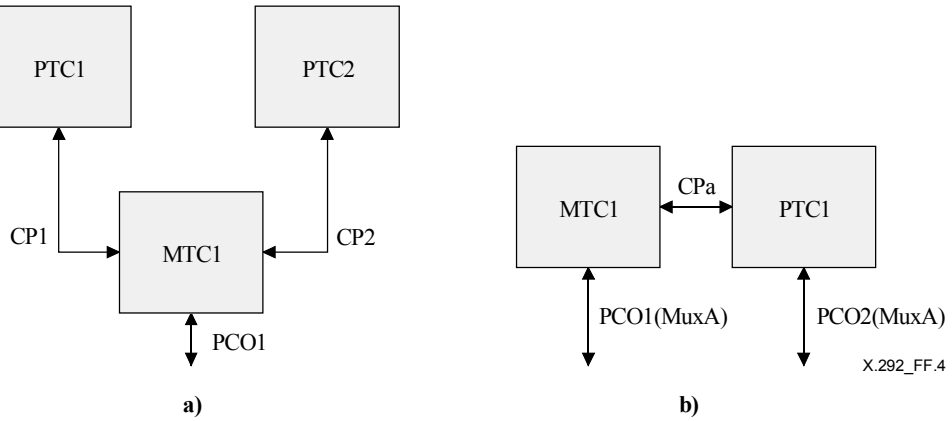


Figura F.4/X.292 – Configuraciones posibles para casos de prueba de multiplexación/demultiplexación

Existen dos modos de utilizar la TTCN concurrente en los casos de prueba que emplean la multiplexación/ demultiplexación. Ambos se presentan en la figura F.4. El primero, en la figura F.4 a), especifica la multiplexación y la demultiplexación de manera explícita dentro del componente de prueba MTC1, con PTC1 y PTC2 gestionando cada uno de ellos el comportamiento de una de las dos conexiones multiplexadas. Esto proporciona la máxima flexibilidad en el modo de especificar el comportamiento de la multiplexación y la demultiplexación, incluida la posibilidad de que se produzcan comportamientos no válidos. Sin embargo, este procedimiento tiene el inconveniente de que ha de especificarse el multiplexor/demultiplexor, que es relativamente complejo, incluso si la finalidad de la prueba se refiere solamente al comportamiento de cada una de las dos conexiones. El procedimiento alternativo consiste en utilizar un PCO independiente para cada tren de eventos separado y un parámetro de sucesión de pruebas (MuxValue) asociado con cada uno de estos PCO que han de multiplexarse y demultiplexarse en el proveedor del servicio subyacente, en lugar de hacerlo en el probador inferior. Esto permite el uso de la configuración de la figura F.4 b). Puesto que la multiplexación/demultiplexación se efectúa en el proveedor del servicio, hay dos PCO en esta configuración, que corresponden con los dos CP en la otra configuración, pero a ambos se les da un MuxValue común, MuxA, para indicar que deben multiplexarse en el proveedor del servicio. En aras de la simplificación, se hace que uno de los componentes de prueba sea el MTC, si bien se podría utilizar en su lugar, si se prefiere, un MTC independiente no conectado a un PCO.

F.12 Partición y recombinación

Para especificar los casos de prueba que incluyen la partición y recombinación, no existe ninguna alternativa a la especificación explícita del comportamiento de la partición y recombinación en el caso de prueba. Se puede utilizar la concurrencia para separar el comportamiento de la partición y recombinación en un componentes de prueba, el MTC1 en la figura F.5, del comportamiento del protocolo situado encima de esta función mediante el uso de un segundo componente de prueba, PTC1 en la figura F.5.

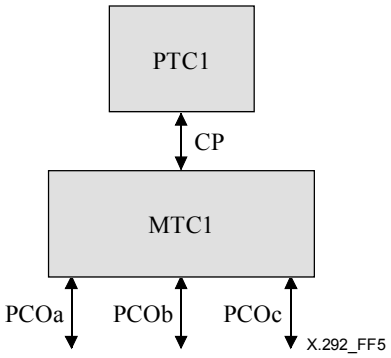


Figura F.5/X.292 – Posible configuración para casos de prueba con partición/recombinación

F.13 Casos de prueba multiprotocolo

Los casos de prueba multiprotocolo, incluidos aquellos que utilizan las variantes insertadas de los métodos de prueba, pueden emplear la TTCN concurrente para separar el comportamiento asociado con cada protocolo en un componente de prueba diferente, tal como se ilustra en la figura F.6, que presenta un ejemplo de configuración de sesión de prueba insertada bajo FTAM.

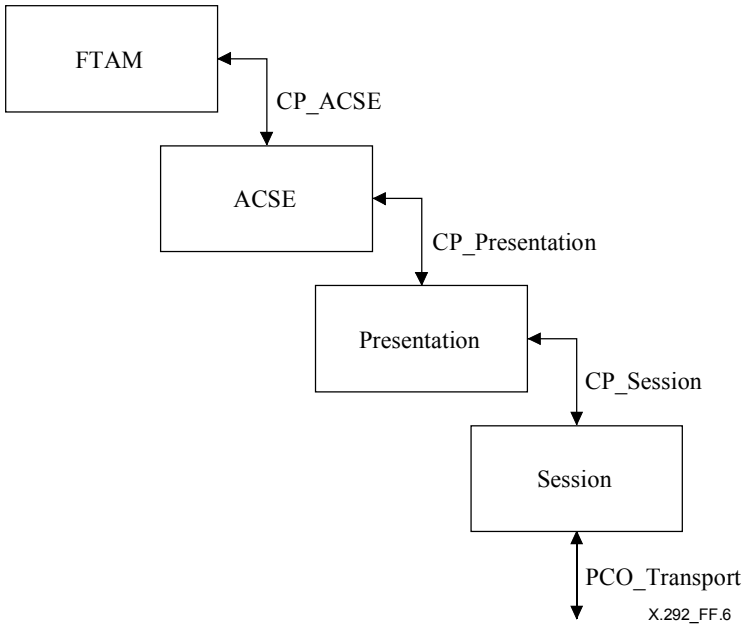


Figura F.6/X.292 – Posible configuración de prueba multiprotocolo – Sesión insertada bajo FTAM

F.14 Ejemplo de TTCN modular

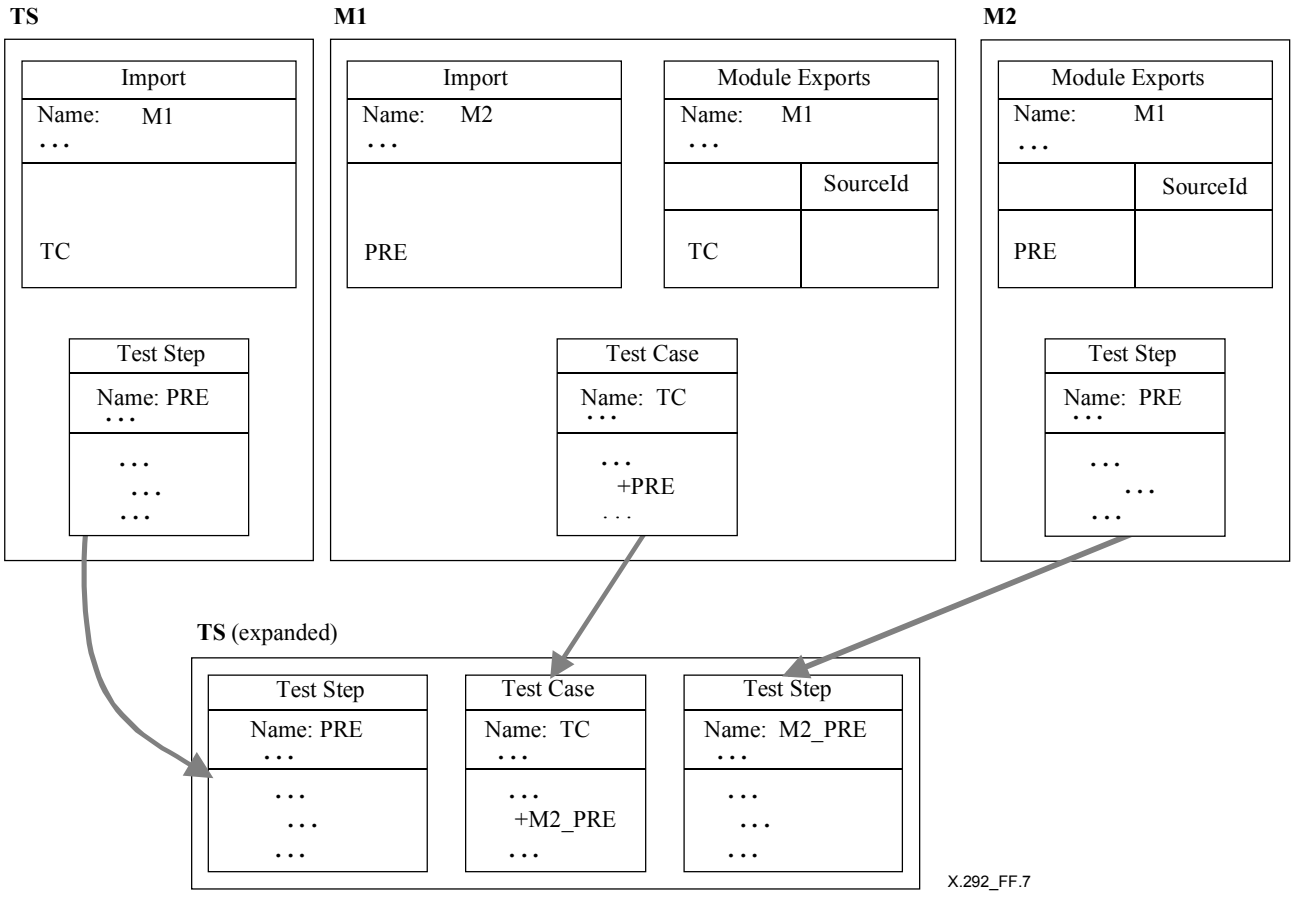


Figura F.7/X.292 – Ejemplo de TTCN modular

El paso de prueba PRE (que se define en el módulo M2) es importado implícitamente de M1 en TS.

Anexo G

Guía de estilo

(Este anexo no es parte integrante de la Recomendación)

G.1 Introducción

En este anexo se presentan algunas reglas de estilo recomendadas, que pueden emplearse cuando se utiliza la TTCN. El objetivo es proporcionar una coherencia básica entre los estilos de la TTCN utilizados por diferentes especificadores de series de pruebas.

G.2 Estructura de caso de prueba

A fin de lograr un mejor análisis de los resultados de una prueba e identificar con facilidad si se ha logrado o no la finalidad de la prueba se sugiere tener en cuenta el texto que sigue cuando se estructuren casos de prueba:

a) El especificador de la serie de pruebas deberá identificar con claridad los subárboles de prólogo y epílogo.

b) El epílogo y el prólogo se especificarán mediante una sola adjunción de árbol de prueba (local al caso de prueba o tomado de la biblioteca de pasos de prueba) en el árbol de comportamiento principal del caso de prueba. Tales árboles de prueba pueden adjuntar subárboles subsiguientes.

c) Una vez identificados los subárboles del prólogo y del(de los) epílogo(s) dentro de un árbol de comportamiento principal del caso de prueba, se podrá considerar que los eventos restantes del árbol principal de comportamiento del caso de prueba están relacionados con el cuerpo de prueba (es decir, que son eventos relacionados con la finalidad de la prueba).

Con este mecanismo, pueden identificarse con facilidad las fronteras entre prólogo, cuerpo de prueba y epílogo, dentro de un caso de prueba. Se pueden usar etiquetas para indicar el comienzo y el final del cuerpo de prueba en el registro cronológico de conformidad.

EJEMPLO G.1 – Identificación de prólogos y epílogos

Comportamiento dinámico del caso de prueba

Nombre del caso de prueba : ST_EX1
Grupo : TTCN_EXAMPLES/STYLE1
Finalidad : Ilustrar la identificación de los prólogos y los epílogos
Configuración
Valor por defecto :
Comentarios :

N.º	Etiqueta	Descripción y comportamiento	Ref. de constricciones	Veredicto	Comentarios
1		+ Preamble			Relacionados con la finalidad
2		! A	A1		
3	Body	? B	B1		
4	CinBody	? C	C1	(PASS)	
5		+ postamble_1			Relacionados con la finalidad
6	DinBody	? D	D1	(PASS)	
7		+ postamble_2			Relacionados con la finalidad
8		? E	E1	INCONC	
9		? OTHERWISE		FAIL	

Comentarios detallados:

Puesto que los veredictos finales causan la terminación de la ejecución de un caso de prueba, un especificador de serie de pruebas no podrá asignar un veredicto final en el cuerpo de prueba si es necesario introducir el epílogo. Incluso es conveniente dar un veredicto en el punto del caso de prueba en el que se logra la finalidad de la prueba y no ocultar veredictos en epílogos. Se recomienda, por tanto, indicar resultados preliminares en la columna de veredictos cuando se haya logrado la finalidad de la prueba pero se tenga que ejecutar todavía un epílogo. En la definición del epílogo, el especificador de la serie de pruebas puede utilizar la variable R como veredicto asignado en las hojas del árbol de

comportamiento, a fin de indicar que, si no se han encontrado errores en el epílogo, el veredicto está determinado en el cuerpo de prueba.

G.3 Utilización de la TTCN con diferentes métodos de prueba abstracta

G.3.1 Introducción

En esta subcláusula se relaciona la TTCN con los métodos de prueba abstracta definidos en la Rec. UIT-T X.291. Se da la sintaxis de TTCN utilizada para expresar la aparición de eventos en los PCO, así como referencias de constricciones para diversos métodos de prueba abstracta.

Se supone que las definiciones de tipos de ASP definen el tipo de parámetro UserData como PDU. Es posible, por tanto, utilizar el encadenamiento de constricciones (es decir, hacer referencia a una constricción para una ASP que contiene una PDU en el parámetro UserData), como referencia a una constricción de ASP que tiene una constricción de PDU como parámetro real.

G.3.2 La TTCN y el método de prueba LS

Eventos TTCN posibles:

Descripción de comportamiento	*Referencia a constricción*
LT! N_ASP	N_ASPconstraint (N_PDUconstraint)
LT? N_ASP	N_ASPconstraint (N_PDUconstraint)
UT! T_ASP	T_ASPconstraint
UT? T_ASP	T_ASPconstraint

G.3.3 La TTCN y el método de prueba DS

Eventos TTCN posibles:

Descripción de comportamiento	*Referencia a constricción*
LT! N_ASP	N_ASPconstraint (T_PDUconstraint)
LT? N_ASP	N_ASPconstraint (T_PDUconstraint)
UT! T_ASP	T_ASPconstraint
UT? T_ASP	T_ASPconstraint

G.3.4 La TTCN y el método de prueba CS

Eventos TTCN posibles:

Descripción de comportamiento	*Referencia a constricción*
LT! N_ASP	N_ASPconstraint (T_PDUconstraint)
LT? N_ASP	N_ASPconstraint (T_PDUconstraint)

Intercambio de TM_PDU entre las implementaciones de protocolo LT y TM en la IUT, a través de la conexión utilizada para pruebas. Obsérvese que, en este caso, la definición de la PDU habrá declarado su campo UserData como de tipo de PDU.

LT! N_ASP	N_ASPconstraint (T_PDUconstraint (TM_PDUconstraint))
LT? N_ASP	N_ASPconstraint (T_PDUconstraint (TM_PDUconstraint))

G.3.5 La TTCN y el método de prueba RS

Eventos TTCN posibles:

Descripción de comportamiento	*Referencia a constricción*
LT! N_ASP	N_ASPconstraint (T_PDUconstraint)
LT? N_ASP	N_ASPconstraint (T_PDUconstraint)

Como no hay UT o TMP, se utiliza el IMPLICIT SEND para describir eventos de envío en el lado de la conexión de la IUT.

<IUT! N_ASP>	N_ASPconstraint (T_PDUconstraint)
<IUT! T_PDU>	T_PDUconstraint

G.4 Utilización de valores por defecto

Como cuestión de estilo, un especificador de series de pruebas deberá eludir situaciones en las que la tentativa de una alternativa de un comportamiento por defecto sea la especificación normal del comportamiento *esperado* de la IUT. Éste sería el caso, por ejemplo, si un paso de prueba representara el comportamiento del LT o UT y de la IUT, cuando se envían eventos de prueba válidos, y si las respuestas de la IUT a eventos de prueba inoportunos o no válidos enviados por el LT o UT se especificasen en valores por defecto adjuntados implícitamente a ese paso de prueba cuando es llamado por otros casos de prueba. Tales valores por defecto deberían llevar veredictos de éxito.

Esta práctica no es recomendable cuando la adjunción de un árbol por defecto queda sin especificar e introduce cierto grado de incertidumbre. En su lugar, deberán utilizarse árboles adjuntados explícitamente o bien el árbol principal.

G.5 Limitación del tiempo de ejecución de un caso de prueba

En versiones anteriores de la TTCN se definió un enunciado ELAPSE que permitía al especificador del caso de prueba limitar la duración anormal de un caso de prueba si, por ejemplo, un procesamiento instantáneo no terminaba nunca o si se producía una repetición incontrolada de la adjunción de árbol.

El enunciado ELAPSE ya no forma parte de la TTCN, porque se considera que el problema que trata de resolver queda fuera del ámbito de la especificación de la serie de pruebas.

Para limitar el tiempo de ejecución del caso de prueba se recomienda ahora que los implementadores de la prueba introduzcan mecanismos locales en los medios de comprobación. Pueden emplearse temporizadores explícitos, junto con el evento TIMEOUT, cuando se necesite fijar un límite al tiempo de espera de que ocurra un evento.

G.6 Tipos estructurados

a) En versiones de DIS anteriores de la TTCN se definieron campos genéricos y valores genéricos como características que permitían agrupar varios valores o campos en un cuadro de constricciones y/o reutilizar ese grupo en varios cuadros de constricciones de contenido similar.

b) En esta versión se introduce primero la agrupación de los parámetros de ASP y de los campos de PDU (tipos de datos externos) en la parte declaraciones, para que esa parte quede más completa y sea coherente con la utilización de ASN.1 en TTCN. Para la definición de los cuadros de definiciones de tipos estructurados véase 11.2.3.3. Una vez declarado un tipo estructurado, puede ser utilizado por una o más definiciones de tipo de ASP o tipo de PDU. En consecuencia, el cuadro de definición de ASP o de PDU puede ser "plano" (es decir, sin grupos propiamente dichos, ni tampoco grupos introducidos por una llamada de macro), o estructurados (mediante especificaciones de estructura de parámetros de ASP o campos de PDU denominados).

c) En la parte constricciones deben asignarse valores a los elementos de estructura en los cuadros de constricciones de tipo estructurado. Se pueden usar los nombres de estas constricciones como valores en los cuadros de constricciones de ASP o PDU de base.

   Por consiguiente, los cuadros de constricciones de ASP o PDU pueden ser también:

   – planos, es decir, cuadros que asignan valores a todos los parámetros o campos individualmente y sólo se refieren a los cuadros de constricción de estructura mediante una llamada de macro; o

   – estructurados, es decir, cuadros que reemplazan valores de grupos declarados de parámetros o campos por nombres de constricciones de grupo.

d) Si la ASP o la PDU declarada se estructura mediante la utilización de algunos parámetros de ASP o campos de PDU especificados por referencia a elementos de estructura, las constricciones deberán tener la misma estructura.

   Cualquiera que sea la forma utilizada, las constricciones de ASP/PDU pueden ser también:

   – modificadas; y

   – parametrizadas mediante un parámetro que deberá estar vinculado a un valor de campo/parámetro o a una constricción de tipo estructurado.

e) Los cuadros de constricción de tipo estructurado sustituyen a los cuadros de campos genéricos de las versiones anteriores de la TTCN.

f) Se suprime el concepto de valores genéricos.

g) En el anexo F se presentan ejemplos.

G.7 Abreviaturas

En las versiones anteriores de la TTCN se permitía declarar, en un cuadro específico, las abreviaturas previstas para su utilización en las columnas de comportamiento de los casos de prueba y pasos de prueba. Se observó que esta facilidad conducía a confusiones, por lo que se ha restringido, de forma que sólo se pueden abreviar los nombres de las ASP y PDU cuando se utilicen en líneas de evento. Esta facilidad se denomina ahora alias.

G.8 Descripciones de prueba

Si se desea, podrán incluirse, en una ATS normalizada, descripciones de comportamiento informales que proporcionan más detalles que la finalidad de la prueba, pero menos que la especificación en TTCN de los casos de prueba.

Tales descripciones de prueba pueden utilizar texto, cronogramas o cualquier otra notación y podrán situarse en el campo de comentarios de los cuadros, en un anexo informativo, o en ambos lugares.

Las especificaciones en TTCN de los casos de prueba tienen siempre precedencia sobre esas descripciones de prueba informales.

G.9 Asignación en caso de eventos SEND

La TTCN permite sobreescribir valores de constricción antes de un evento SEND en un enunciado asignación de una línea de evento. Esto significa que el primer dato a enviar se construye a partir de la definición de la constricción, ejecutándose después las asignaciones.

Esta característica debe utilizarse con precaución, porque puede producir confusión al lector de la serie de pruebas respecto al valor real que debe enviarse. Se considera de mal estilo, en particular, utilizar la misma constricción para enviar y para recibir.

G.10 PCO multiservicio

Cuando un PCO abarque más de un SAP, la especificación precisa de ese PCO viene dada por el conjunto de las ASP y PDU que pueden ocurrir.

EJEMPLO G.2 – Un PCO de FTAM:

Declaraciones de PCO			
Nombre del PCO	**Tipo de PCO**	**Cometido**	**Comentarios**
L	A_P_SAPs	LT	PCO a través del cual podemos observar todas las ASP de ACSE y todas las ASP de presentación, excepto P-CONNECT, P-RELEASE y P-ABORT.

El PCO "L" es del tipo A_P_SAP, que es capaz de observar todas las ASP de ACSE y de presentación, excepto P-CONNECT, P-RELEASE y P-ABORT. La columna de tipo muestra las SAP que pertenecen al conjunto observado por el PCO como "A" y "P", estando cada SAP separado por el signo de subrayado ("_"). En la columna de comentarios se describe exactamente lo que puede ser visto por el PCO.

Este método es extensible a varios SAP, cada uno de los cuales estaría separado por un signo de subrayado.

Anexo H

Índice

(Este anexo no es parte integrante de la Recomendación)

H.1 Introducción

En este anexo se presenta un índice alfabético de términos y acrónimos utilizados en esta Recomendación. El índice da, para cada término o acrónimo (en inglés), un conjunto de referencias mediante los números de la cláusula, figura y cuadro, tanto en el cuerpo principal como en los anexos. El grado de importancia de cada una de las referencias se indica como sigue:

a) la definición de los términos y los acrónimos se señala en **negritas**;

b) los usos principales del término o acrónimo se señala en *cursiva*;

c) otros usos se presentan en tipo normal.

H.2 Índice

A
ABSENT: *A.4.2.5*
Abstract Service Primitive: *1*, *4.1*
Abstract Syntax Notation One: *4.3*
ABSTRACT SYNTAX: *A.4.2.5*
Abstract test case: *6*, *8.2*, *11.13.2*, *15.17.1*
Abstract test suite: *1*, *2*, *4.1*, *8.2*
Abstract testing methodology: *1*
Access to behaviour description: 15.13.2
ACTIVATE procedure: **B.5.19.1**
ACTIVATE: *15.4.1*, *15.14*, **15.18.4**, *15.18.4*, 15.18.6, *15.18.6*, *A.4.2.4*, *B.5.5.4*, *B.5.5.5*, *B.5.18.2*, *B.5.19.2*
Actual parameters: *15.13.5*, *15.16.2*
ActualParList: *B.5.5.3*
Alias definition: *11.1*, *A.3.3.13.14*
ALL: *A.4.2.5*
Ancestor node: *15.14*
AND: *A.4.2.4*
AnyOne: **12.6.5.1**, *12.6.6.1*
AnyOrNone: **12.6.5.2**, *12.6.5.3*, *12.6.6.1*
AnyOrOmit: **12.6.4.4**, *12.6.6.1*
AnyValue: **12.6.4.3**, *12.6.5.1*, *12.6.6.1*
APPEND_DEFAULTS: **B.5.5.4**
APPEND_TO_LEVEL: **B.5.25**
Applicable encoding rules: **3.6.1**
APPLICATION: *A.4.2.5*
Arithmetic operators: 11.3.2.2
Array references: 15.10.2.3
ASN.1 ASP constraints: *14.2*, 14.3, *A.4.2.15*
ASN.1 ASP type definition: 11.14
ASN.1 CM constraints: 14.9, *A.4.2.15*
ASN.1 CM type: *11.17.3*
ASN.1 comments: 11.2.3.4, *11.15.4*, *14.1*
ASN.1 compact constraints: E.2.5
ASN.1 constraint declaration: *12.6.6.1*, 14, *A.3.3.22*, *E.2.5*, F.2
ASN.1 constraint: *12.6.6.2*
ASN.1 constraints: *12.1*, *12.6.5.2*, *14.1*
ASN.1 dash symbol: *14.1*
ASN.1 defined data objects: 15.10.2
ASN.1 encoding rules: *11.15.1*
ASN.1 identifier: *3.6.48*
ASN.1 module: 11.2.3.5, *11.14.5*
ASN.1 PDU constraint declaration: 14.4
ASN.1 PDU constraints: *14.2*, *A.4.2.15*
ASN.1 PDU type definition: 11.15
ASN.1 type constraints: 7.3.4, *11.16.4*, 14.2, *A.4.2.15*
ASN.1 type definition: *11.2.3.4*, 11.2.3.5, *11.18.2*, *14.5*, *14.8*, *A.4.2.1*, *A.4.2.6*
ASN.1 type: *11.2.3.4*, *11.2.3.5*, *11.8.1*, *11.8.3*, *11.14.2*, *11.14.5*, *11.15.4*, *12.6.2*

ASN.1: *1*, *2*, **4.3**, *8.1*, *9.5*, *11.2.2*, *11.2.3.4*, *11.6*, *11.7*, *11.14.3*, *11.14.4*, *11.14.5*, *11.15.4*, *11.15.5*, *11.17.3*, *12.2*, *12.6.1*, *12.6.4.2*, *15.10.2*, *A.4.2.1*, *A.4.2.5*, *E.2.1*, *G.6*
ASP constraint compact proforma: E.2.2
ASP constraint declaration: *3.6.62*, 13.3, d), *A.5.1*, *E.2.5*
ASP constraints: 7.3.4
ASP identifier: *11.21*
ASP parameter: *3.6.66*, *11.2.1*, *11.14.2*, *12.5*, *12.6.2*, 12.6.3, *12.6.4.1*, *12.6.4.2*, *12.6.4.3*, *12.6.4.4*, *12.6.4.5*, *12.6.4.7*, *12.6.4.8*, *12.6.5.1*, *12.6.5.3*, *12.6.6.2*
ASP specified by reference: 11.14.5
ASP type definition: *3.6.3*, *3.6.68*, *11.1*, *11.2.2*, 11.14, 11.19, 11.20, *A.3.3.19*, *A.3.3.22*, *G.3.1*
ASP type: *11.3.4.2*, *14.3*, *15.7.2*
ASP: *3.6.9*, *3.6.13*, *3.6.25*, *3.6.38*, *3.6.44*, *3.6.57*, *3.6.60*, *3.6.68*, **4.1**, *8.1*, *9.5*, *11.2.1*, *11.2.2*, *11.2.3.3*, *11.3.4.1*, *11.3.4.2*, *11.6*, *11.7*, *11.10*, 11.14, *11.14.2*, *11.14.3*, *11.14.4*, *11.14.5*, *11.15*, *11.15.1*, *11.15.5*, *11.16.4*, *11.19*, *11.20*, *11.21*, *12.1*, *12.4*, *12.6.1*, 12.6.3, *13.2*, *13.6*, *14.5*, *14.6*, *14.8*, *15.2.1.3*, *15.9*, *15.9.5.3*, *15.9.6*, *15.10.1*, *15.10.2.2*, *15.10.2.3*, *15.10.3*, *15.10.6*, *15.16.1*, *A.4.2.7*, *A.4.2.8*, *B.5.2.3*, *B.5.8.2*, *B.5.9.2*, *B.5.10.2*, *B.5.11.2*, *B.5.12.2*, *B.5.13.2*, *B.5.16.2*, *E.2.1*, *G.6*, *G.7*, *G.10*
ASPs specified in ASN.1: 11.14.4
Assignment rules: 15.10.4.2
Assignment: *11.3.4.3*, *11.3.4.6*, *11.8.2*, *11.8.4*, *15.6*, *15.8*, *15.9.3*, *15.9.4*, *15.10.1*, 15.10.4, *15.10.5*, 15.10.6, *15.11*, 15.16.3, *15.17.2*, B.5.16, G.9
ATS: *3.6.74*, **4.1**, *6*, *10.1*, *10.2*, *10.3*, *10.4*, *10.5*, *11.1*, *11.3.4.1*, *b)*, *11.9*, *11.14.4*, *11.16.1*, *12.1*, *A.1*, *A.5.1*
Attach construct: **3.6.2**, *15.2.3*, *15.8*, *15.17.1*, *B.5.5.4*, *B.5.5.5*, *E.3.1*
ATTACH: *15.9.10.1*, *15.13.1*, *15.13.4.1*, *E.3.1*
Attached tree: *15.13.3*
Attachment construct: *B.5.1*
Attribute: *11.15.2*, *11.18.1*, *13.4*
Attributes of values: 12.6.6
AUTOMATIC: *A.4.2.5*

B

Backus-Naur Form: *4.3*
Base constraint: **3.6.3**, *3.6.24*, *3.6.44*, 13.6, *13.7*, *A.3.3.19*, *A.3.3.22*, *E.2.3*, F.3
Base type: **3.6.4**, *11.18.2*
BEGIN: *11.3.4.4*, *A.4.2.5*
Behaviour description: *3.6.40*, *3.6.55*, *3.6.78*, *3.6.90*, *11.10*, *11.21*, *12.1*, *12.3*, *15.2.1*, *15.2.1.3*, *15.2.5*, 15.5, *15.13.2*, *15.15*, *A.4.2.10*, *A.5.1*, *A.5.2*, *E.3.1*, *G.3*, *G.8*
Behaviour line: **3.6.5**, *3.6.14*, *3.6.25*, *15.2.5*, *B.5.1*
Behaviour tree: **3.6.6**, *3.6.8*, *3.6.42*, *3.6.49*, *3.6.59*, *3.6.83*, *3.6.84*, *3.6.85*, *3.6.87*, *15.2.1.3*, *15.2.2*, *15.4.1*, *15.5*, 15.9.5, *15.11*, *15.13.3*, *15.13.4.1*, *15.14*, *15.16.2*, *15.17*, *15.18.1*, *B.5.1*, *B.5.5.4*, *B.5.5.5*, *G.2*
BehaviourLine: *B.5.2.5*
BER: *11.15.2*, *11.15.4*, *11.15.5*, *11.16.4*, *13.4*, *14.4*
Binding of variables: 11.8.4
Bit reference: 15.10.2.4
BIT: *A.4.2.5*
BIT_TO_INT: *11.3.3.2.1*, 11.3.3.2.3, *A.4.2.4*
BITSTRING: **11.2.2**, *11.18*, *15.10.2.4*, *15.10.4.2*, *A.4.2.4*
Blank entry: **3.6.7**
BMPString: *A.4.2.5*
BNF grammar for TTCN: 7.2
BNF: **4.3**, *7.2*, A.3
Boolean expression: *15.10.1*
Boolean operators: 11.3.2.4
BOOLEAN: **11.2.2**, *11.3.3.3.1*, *11.3.3.3.2*, *b)*, *15.10.2.4*, *A.4.2.4*, *A.4.2.5*, B.5.15
Bound variable: 11.8.2, *15.16.2*, *15.18.2*
Bounded free text: 7.4
BUILD_SEND_OBJECT: **B.5.8.1**
BY: *A.4.2.4*

C

Calling tree: *3.6.2*, **3.6.8**, *3.6.42*, *15.13.3*, *15.17.3*, *15.18.5*
CANCEL operation: **15.12.3**
CANCEL: *15.12.1*, *15.12.3*, *A.4.2.4*, *B.5.14.2*, *B.5.17*
CANCEL_TIMER: **B.5.17.1**
CASE OF ELSE: *11.3.4.9*
CASE: *11.3.4.9*, *A.4.2.4*

Chaining of constraints: *12.4, 15.10.2.2, 15.10.3, F.1.1.3, F.1.2.3, F.2.2.3, G.3.1*
CHARACTER: *A.4.2.5*
Characterstring type: *11.3.3.3.4*, *11.18.1*
CharacterString: **11.2.2**, *12.6.5.1*, *12.6.5.2*, *15.10.4.2*
CHOICE: *11.3.3.3.2*, *12.4*, *14.5*, *14.8*, *15.10.2.2*, *15.10.2.3*, *A.4.2.5*
CLASS: *A.4.2.5*
CM constraint declarations: 13.8
CM parameters: *11.17.1*
CM type: 11.17.2, 11.17.3
CM: *3.6.16*, **4.3**, *8.1*, *11.3.4.2*, *11.6*, *11.7*, *11.11*, *11.17.1*, *11.17.2*, *12.1*, *13.6*, *13.8*, *14.9*, *15.9.2*, *15.9.3*, *15.9.4*, *15.9.5.3*, *15.9.5.4*, *15.9.8*, *15.10.2.2*, *15.10.2.3*, *15.10.3*, *15.10.6*, *15.16.1*, *15.17.5*, *15.18.8*, *A.4.2.7*, *A.4.2.8*, *B.5.2.3*, *B.5.8.2*, *B.5.9.2*, *B.5.10.2*, *B.5.11.2*, *B.5.12.2*, *B.5.16.2*
CMs and defaults: 15.18.8
Collective comment: 7.3.3, *11.2.3.2*
Compact constraint table: *3.6.7*, **3.6.9**, *13.1*, *E.1*, E.2
Compact proformas: Annex E
Compact test case table: **3.6.10**, *E.1*, E.3
Complement matching operation: 12.6.4.1
COMPLEMENT: *A.4.2.4*
Complement: **12.6.4.1**, *12.6.6.1*
Complex CMs: 11.17.1
Compliance: 6, *15.17.3*
Component of data object: 15.10.2.2, 15.10.2.3
COMPONENT: *A.4.2.5*
Component: *3.6.72*
Concurrent test case behaviour: 15.2.4
Concurrent test case: *3.6.11*, *3.6.72*
Concurrent TTCN: *3.6.11*, **3.6.12**, *3.6.47*
Conflict between TTCN forms: 5
Conformance log: *15.17*, *B.3*, *B.5.20.2*, *B.5.23.2*, B.5.24, *B.5.24.2*, *G.2*
Conformance test suite: *1*
CONSTRAINED: *A.4.2.5*
Constraint declarations: *3.6.25*
Constraints for RECEIVE: 12.6
Constraints part: **3.6.13**, 9.5, 12, *15.2.1.3*, *15.16.1*, *A.3.3.36.2*
Constraints reference: *3.6.5*, **3.6.14**, *3.6.25*, *12.2*, *12.3*, *15.2.1.3*, 15.16, 15.16.1, *B.1*, *B.5.8.2*, *B.5.9.2*, *B.5.10.2*, *B.5.11.2*, *B.5.12.2*, *B.5.13.2*, *G.3*
Construct: *3.6.61*, *3.6.90*, *15.2.1.3*, *15.8*, *15.9.5*, *15.17.1*, *15.18.1*, B.5.18
CONSTRUCT_TYPE_OF: **B.5.26**
Co-ordinated test method: G.3.4
Co-ordination message declarations: 8.1
Co-ordination message: **3.6.15**, *4.3*, *8.1*
Co-ordination point declarations: 8.1
Co-ordination point model: 11.11
Co-ordination point: *3.6.15*, **3.6.16**, *4.3*
CP: *3.6.72*, *3.6.73*, **4.3**, *8.2*, *11.11*, *11.13.1.1*, *11.13.1.3*, *11.13.2*, *15.2.4*, *15.9.5.3*, *15.9.8*, *15.9.10.1*, *15.10.6*, *A.4.2.4*, *A.4.2.13*, *B.1*, *B.5.4.2*, *F.11*
CREATE and defaults: 15.18.7
Create construct: **15.9.10.1**
CREATE procedure: **B.5.20.1**
CREATE: *8.2*, *11.13.1.1*, *11.13.1.2*, *15.9.10*, *15.9.10.1*, *15.9.10.2*, *15.18.7*, *A.4.2.4*, *B.5.18.2*, F.15
CurrentLevel: *B.5.2.3*

D

Data object: *12.2*, *15.10.1*
Declaration by reference: 11.7
Declarations part: **3.6.17**, 9.5, *11.1*, *15.9.1*, *A.3.3.36.2*, *G.6*
DEF_REF_LIST: **B.5.26**
Default behaviour proforma: *3.6.18*, *B.1*
Default behaviour: **3.6.18**, *3.6.19*, *3.6.22*, *15.1*, *15.2.1*, 15.4, *15.18.1*, *15.18.2*, *15.18.4*, *B.5.5*, B.5.5.4, *G.4*
Default duration: *11.12*
Default dynamic behaviour: *3.6.26*, 9.5
Default expansion: *15.18.3*
Default expression: *11.16.1*, *11.16.2*
Default group reference: **3.6.20**, *9.4*, *10.6*

Default group: **3.6.19**, *9.1*
Default identifier: **3.6.21**, *10.6*, *15.18.2*, *A.4.2.11*
Default index: 10.1, 10.6, *A.5.1*
Default library: *3.6.20*, **3.6.22**, *3.6.23*, *3.6.52*, *9.4*, *10.6*, *15.4.1*, *15.18.2*
Default objective: *10.6*
Default reference: **3.6.23**, *15.2.1*, 15.18.2, *B.5.5.4*
Default tree: *15.10.1*, *15.14*, *15.18.1*, *15.18.3*, *A.3.3.33*, *A.4.2.9*, *E.3.1*, *G.4*
Default value: *13.6*
DEFAULT: *11.3.3.3.1*, *15.18.1*, *A.4.2.5*
Default: *15.4.1*, 15.18, *15.18.2*, 15.18.6, *B.5.1*, *B.5.2.3*, *B.5.5.1*, *B.5.5.4*, *B.5.5.5*, G.4
Defect report: *B.2*
Definition by reference: 11.7
DEFINITIONS: *A.4.2.5*
DER: *11.15.2*, *11.15.4*, *11.15.5*, *13.4*, *14.4*
Derivation path: **3.6.24**, *13.4*, *13.6*, *14.3*, *d)*, *14.4*, *A.3.3.22*, *E.2.3*
Derivation: *A.1*
Detailed comments: *11.3.4.1*
Distinguished value: *A.4.2.6*
Distributed test method: *4.2*, G.3.3
DO: *11.3.4.8*, *A.4.2.4*
Done event: **15.9.10.2**
DONE: *15.9.10*, *15.9.10.2*, *A.4.2.4*, *B.5.7.2*, *B.5.12.2*, F.15
Dot notation: 15.10.2.2, 15.10.2.3
DS: **4.2**
Dynamic behaviour: *3.6.12*, *11.13.2*
Dynamic chaining: **3.6.25**, *12.4*
Dynamic part: **3.6.26**, 9.5, *11.1*, 15, *A.3.3.36.2*

E

EBDIF: *A.4.2.4*
Element: *15.10.3*
ELSE: *A.4.2.4*
EMBEDDED: *A.4.2.5*
Encoding definition: *11.3.3.2.1*, *11.15.2*, *11.15.4*, *11.15.5*, *11.16.1*, *11.16.2*, *11.16.4*
Encoding operation: *11.16.3*
Encoding rules precedence: 11.16.4
Encoding rules: *11.16.1*, *11.16.2*, *11.16.4*
Encoding variations: *11.2.3.2*, *11.2.3.3*, *11.2.3.4*, *11.2.3.5*, *11.15.2*, *11.15.4*, *11.15.5*, *11.16.2*, *13.2*, *13.4*, *14.2*, *14.4*
END: *11.3.4.4*, *A.4.2.4*, *A.4.2.5*
ENDCASE: *11.3.4.9*, *A.4.2.4*
ENDIF: *11.3.4.7*
ENDVAR: *11.3.4.3*, *A.4.2.4*
ENDWHILE: *11.3.4.8*, *A.4.2.4*
Enumerated type: *A.4.2.6*
ENUMERATED: *A.4.2.5*, *A.4.2.6*
Equivalence of TTCN forms: 5
ETS: **4.1**
EVAL_VERDICT_ENTRY: **B.5.23.1**
EVALUATE_BOOLEAN: **B.5.15.1**
EVALUATE_CONSTRUCT: **B.5.18.1**
EVALUATE_EVENT: **B.5.7.1**
EVALUATE_EVENT_LINE: **B.5.6.1**
EVALUATE_LEVELS: **B.5.4.1**
EVALUATE_PSEUDO_EVENT: **B.5.14.1**
EVALUATE_TEST_CASE: *B.1*, *B.5.3.1*, **B.5.4.1**
EVALUATE_TEST_COMPONENT: *B.5.3.1*
EVALUATE_TEST_COMPONENT: *B.1*, **B.5.20.1**
EVALUATE_TEST_SUITE: *B.1*, *B.5.2.3*, **B.5.3.1**
Evaluation tree: *B.1*, *B.5.2.1*, *B.5.2.3*
Event line: *15.9*, *15.10.1*, *15.10.4.1*, 15.10.6, *G.7*, *G.9*
Event matching: 15.10.6
EVENT_TYPE_OF: **B.5.26**
Examples of tabular constraints: F.1

Examples: Annex F
EXCEPT: *A.4.2.5*
EXCLUDE_INCOMPATIBLE_ENTRY: **B.5.23.1**
Executable test case error: *6.5*
Executable test case: *6.5*
Executable test suite: *1*, *4.1*
EXECUTE_ASSIGNMENT: **B.5.16.1**
Execution of a test suite: B.5.3
EXPAND_ATTACHMENTS: **B.5.5.5**
EXPAND_CURRENT_LEVEL: **B.5.5.1**
EXPAND_REPEATS: **B.5.5.3**
EXPAND_SUBTREE: **B.5.5.5**
Expanded test suite: **3.6.27**
Expanding a set of alternatives: B.5.5.1
Expanding modularized test suite: B.4
Expansion of aliases: 11.21
Expansion of default trees: 15.13.7
Explicit external: **3.6.28**
EXPLICIT: *A.4.2.5*
Explicitly defined object: **3.6.29**, *3.6.32*, *3.6.33*, *3.6.36*
Explicitly exported object: **3.6.30**, *3.6.32*, *3.6.36*
Explicitly external object: *3.6.33*
Explicitly imported object: **3.6.31**, *3.6.32*, *3.6.36*, *3.6.37*, *3.6.39*, *10.8.1*
Explicitly imported: *B.1*
EXPORT: *A.4.2.5*
Export: *11.9*
Exported object: **3.6.32**
Exporting object type: 10.7
External object: *3.6.28*, **3.6.33**
External objects: *C.3.1*, **C.3.2**
EXTERNAL: *10.7*, *10.8.2*, *A.4.2.5*, *C.2.2*
Externally declared object: *3.6.35*
Externally defined object: *3.6.46*

F

F: *15.17.2*, *15.17.3*, *A.4.2.4*
FAIL: *15.17.1*, *15.17.2*, *15.17.3*, *15.17.4*, *15.18.1*, *A.4.2.4*, *B.5.23.2*
Fail: *3.6.54*
FALSE: *10.3*, *10.4*, *11.2.2*, *11.3.3.3.1*, *11.3.3.3.2*, *11.3.4.7*, *11.3.4.8*, *11.16.1*, *11.16.2*, *15.11*, *A.4.2.4*, *A.4.2.5*, *B.5.8.2*, *B.5.9.2*, *B.5.10.2*, *B.5.11.2*, *B.5.12.2*, *B.5.15.2*
FDT: **4.3**
Field encoding definition: *11.2.3.2*, *11.2.3.4*, *11.15.2*, *11.15.4*, *11.16.3*, *13.4*
Field: *15.10.3*
FIFO: **4.3**, *11.10*
Final verdict: *15.9.10.2*, *15.17.1*, *15.17.3*, *15.18.1*, *G.2*
FIRST_LEVEL: **B.5.25**
Formal Description Technique: *4.3*
Formal description technique: *1*
Formal parameter list: *12.3*, *13.4*, *13.7*, *d)*, 14.7, *14.7*, *15.9.1*, *15.16.2*, *A.4.2.11*, *A.4.2.12*
Formal parameter name precedence: *A.4.2.12*
Formal parameter: *A.4.2.14*
Formal parameters: *3.6.85*, *15.7.2*, *15.13.5*
Free text: 7.4
FROM: *A.4.2.5*

G

GeneralizedTime: *A.4.2.5*
GeneralString: *A.4.2.4*, *A.4.2.5*
Global result variable: **3.6.34**
Global test steps: 9.3.2
GOTO construct: **15.14**
GOTO: *15.2.1.3*, *15.6*, *15.8*, *15.9.5.1*, *15.14*, *15.17.1*, *A.4.2.4*, *B.1*, *B.5.5.1*, *B.5.18.2*, B.5.21, *B.5.21*, B.5.22
GOTO_NEXT_LEVEL_OR_STOP_WITH_VERDICT: *B.5.21*, *B.5.22*, **B.5.25**
GraphicString: *A.4.2.4*, *A.4.2.5*

H
HEX_TO_INT: *11.3.3.2.1*, 11.3.3.2.2, *A.4.2.4*
HEXSTRING: **11.2.2**, *11.18.1*, *11.18.2*, *15.10.4.2*, *A.4.2.4*
Hyphen symbol: *11.15.4*

I
I: *15.17.2*, *15.17.3*, *A.4.2.4*
IA5String: *A.4.2.4*, *A.4.2.5*
IDENTIFIER: *A.4.2.5*
Idle testing state: *15.17.3*
IF THEN ELSE: *11.3.4.7*
IF THEN: *11.3.4.7*
IF: *A.4.2.4*
IF_PRESENT: *A.4.2.4*
IfPresent: *12.6.6.1*, **12.6.6.2**
Illegal variations of encoding: 11.16.3
Implementation Under Test: *4.1*
Implicit external: **3.6.35**
Implicit send event: **3.6.38**, **15.9.6**
IMPLICIT SEND: *15.6*, *15.8*, *15.9.5.3*, *15.9.6*, *15.16.1*, *15.17.1*, *B.5.7.2*, *B.5.13.2*, *G.3.5*
IMPLICIT: *A.4.2.5*
IMPLICIT_SEND: **B.5.13.1**
Implicitly exported object: *3.6.32*, **3.6.36**
Implicitly external object: *3.6.33*
Implicitly imported object: **3.6.37**, *3.6.37*, *3.6.39*, *10.8.1*, *B.1*
Import part: 9.5, 10.8.1, *C.1*
IMPORT: *A.4.2.5*
Import: 10.8.2, *C.3.1*, **C.3.3**
Imported object: *3.6.27*, *3.6.30*, *3.6.33*, **3.6.39**, *10.8.1*, *B.4*
INCLUDES: *A.4.2.5*
INCONC: *15.17.1*, *15.17.2*, *15.17.3*, *A.4.2.4*, *B.5.23.2*
Inconclusive verdict: *15.17.3*
Indentation: *3.6.61*, *15.2.5*, *15.6*, *15.9.5*, *15.15*, *A.5.1*, *A.5.2*, *B.5.5.3*
Index notation: 15.10.2.4
INFINITY: *11.2.3.2*, *11.14.2*, *11.15.2*, *11.17.2*, *11.18.2*, *12.6.4.6*, *12.6.6.1*, *A.4.2.4*
INPUT_Q: **B.5.26**
Inside values: 12.6.5
INSTANCE: *A.4.2.5*
INT_TO_BIT: *11.3.3.2.1*, 11.3.3.2.5, *A.4.2.4*
INT_TO_HEX: *11.3.3.2.1*, 11.3.3.2.4, *A.4.2.4*
INTEGER: **11.2.2**, *11.2.3.2*, *11.3.3.3.3*, *11.12*, *11.14.2*, *11.17.2*, *11.18.2*, *12.6.4.6*, *12.6.5.1*, *12.6.5.3*, *12.6.6.1*, *15.10.2.3*, *15.10.2.4*, *15.12.2*, *15.12.4*, *A.4.2.4*, *A.4.2.5*, *A.4.2.6*
INTERSECTION: *A.4.2.5*
Invalid field encoding definition: *14.2*, *14.4*
Invalid field encoding: *11.2.3.3*, *11.16.3*, *12.6.4.2*
Invalid test event: *15.17.4*
IS_CHOSEN: 11.3.3.3.2, *A.4.2.4*
IS_EXPANDED: **B.5.25**
IS_PRESENT: 11.3.3.3.1, *A.4.2.4*
IsDefault: *B.1*
IsExpanded: *B.1*
ISO646String: *A.4.2.5*
IUT: *3.6.13*, *3.6.38*, **4.1**, *10.3*, *11.10*, *11.11*, *11.15.1*, *15.9.6*, *A.4.2.4*, *G.3.4*, *G.4*

L
Label: *3.6.5*, *15.2.1.3*, **15.14**
Length: *11.18.2*, **12.6.6.1**
LENGTH_OF: 11.3.3.3.4, *A.4.2.4*
Level of indentation: **3.6.40**
LEVEL_OF: *B.5.2.5*, **B.5.25**
Levels of alternatives: B.5.2.5
Lifetime of events: 15.9.4
Line continuation: 15.2.5, *A.5.1*
Literal values: *11.16.1*, *11.16.2*

Local result variable: **3.6.41**, *B.5.20.2*
Local test method: *4.2*, G.3.2
Local test steps: 9.3.2
Local tree: **3.6.42**, *3.6.85*, *3.6.86*, *15.2.5*, *15.4.1*, *15.6*, *15.10.1*, *15.13.2*, *15.13.3*, *15.13.4.1*, *15.15*, *A.4.2.9*, *A.4.2.10*, *A.4.2.11*, *A.5.2*
Local variables: *11.3.4.3*, *11.3.4.4*, *11.3.4.6*
Location of object: 10.7
LOG procedure: **B.5.24.1**
Lower Tester Control Function: *4.1*
Lower tester: *4.1*, *11.13.1.2*
LS: **4.2**
LT: **4.1**, *11.9*, *11.10*, *15.2.1.3*, *15.8*, *15.9.1*, *15.9.5.1*, *15.9.6*, *15.9.7*, *A.4.2.4*, *B.5.8.2*, *B.5.9.2*, *B.5.10.2*, *B.5.11.2*, *B.5.12.2*, *G.4*
LTCF: **4.1**

M
Macro expansion: *12.2*, *13.2*, *13.4*, 15.10.3, *15.10.3*, *A.3.3.34*, *A.4.2.8*
Macro symbol: *11.14.3*, *11.15.3*
Main test component: *3.6.34*, **3.6.43**, *3.6.53*, *4.3*, *8.1*, 11.13.1.1, *11.13.1.3*, *15.9.10.1*, *B.5.2.3*, *B.5.3.1*, *B.5.23.2*
MAKE_TREE: **B.5.25**
Matching ASP: *12.6.1*
Matching attributes of values: 12.6.2
Matching inside values: 12.6.2
Matching instead of values: 12.6.2
Matching mechanism: *3.6.65*, *12.2*, *12.5*, 12.6.3, *14.1*, *15.9.9*
Matching mechanisms: 12.6.2
Matching PDU: *12.6.1*
Matching values in constraints: 12.6.1
MAX: *A.4.2.5*
Means of Testing: *4.1*, *G.5*
MIN: *A.4.2.5*
min: *A.4.2.4*
MOD: *A.4.2.4*
Modified ASN.1 constraints: 14.6, *14.7*, 14.7
Modified constraints: *3.6.7*, *3.6.24*, **3.6.44**, 13.6, *13.7*, *A.3.3.19*, *A.3.3.22*, *E.2.3*, *E.2.4*, F.1.2.5, *F.2.2.5*, F.3, *G.6*
Modular TTCN: F.14
Modularized test suite: **3.6.45**
Module constraints part: *C.1*
Module declarations part: *C.1*
Module default index: *C.2.1*, **C.2.6**
Module dynamic part: *C.1*
Module exports: C.2.2
Module import part: C.3
Module structure: C.2.3
Module test case index: *C.2.1*, **C.2.4**
Module test step index: *C.2.1*, **C.2.5**
Module: *3.6.28*, *3.6.29*, *3.6.32*, *3.6.33*, **3.6.46**, *3.6.50*, *3.6.69*, *10.8.1*, *B.1*, *B.4*
MOT: **4.1**, *15.9.5.3*
MPyT: *3.6.34*
ms: *A.4.2.4*
MTC: *3.6.58*, **4.3**, *8.1*, *8.2*, *11.8.1*, *11.8.3*, *11.13.1.2*, *11.13.2*, *15.2.4*, *15.9.10.2*, *15.17.5*, *15.18.7*
MTC_R: *3.6.58*, *15.17.5*
Multi-party testing: F.10
Multiplexing/demultiplexing: F.11
Multi-protocol test cases: F.13
MuxValue: 11.10, *F.11*

N
NEW_LABEL: **B.5.25**
Non-concurrent test case: **3.6.47**
none: *A.4.2.4*
NOT: *A.4.2.4*
ns: *A.4.2.4*
NULL: *A.4.2.5*

NUMBER_OF_ELEMENTS: 11.3.3.3.3, *A.4.2.4*
NumericString: *A.4.2.4*, *A.4.2.5*

O
Object group: 7.3.2
Object name: 7.3.2, 7.3.3
OBJECT: *A.4.2.5*
Object: **3.6.48**, *3.6.50*, *10.8.2*
OBJECT_MATCHES: **B.5.9.1**
ObjectDescriptor: *A.4.2.5*
OBJECTIDENTIFIER: **11.2.2**, *A.4.2.4*
OCTET: *A.4.2.5*
OCTETSTRING: **11.2.2**, *11.3.3.3.4*, *11.18.1*, *11.18.2*, *15.10.4.2*
OF: *A.4.2.4*
Omit symbol: *12.5*
OMIT: *10.8.2*, *14.6*, *A.4.2.4*
Omit: **12.6.4.2**
Open Systems Interconnection: *4.3*
Operational semantics: *1*, **3.6.49**, *5*, *6*, *15.9.5.2*, Annex B, B.5
OPTIONAL: *11.3.3.3.1*, *11.3.3.3.3*, *12.5*, *12.6.4.2*, *12.6.6.2*, *14.5*, *14.8*, *A.4.2.5*
OR: *A.4.2.4*
Order of receipt of events: *15.9.5.4*
Original source object: **3.6.50**
OSI: *1*, *2*, **4.3**, *A.4.2.1*
Otherwise event: **3.6.51**, *15.9.7*
OTHERWISE function: **B.5.10.1**
OTHERWISE: *3.6.91*, *15.8*, *15.9.5.3*, *15.9.7*, *15.9.8*, *15.10.6*, *15.17.4*, *15.18.5*, *A.3.3.33*, *A.4.2.4*, *B.5.7.2*, *B.5.10.2*, *B.5.15.2*
OUTPUT_Q: **B.5.26**
Overview part: **3.6.52**

P
P: *15.17.2*, *15.17.3*, *A.4.2.4*
Page continuation: 16, 16.1, 16.2, *A.5.1*
Parallel test component: *3.6.43*, **3.6.53**, *4.3*, *8.1*, 11.13.1.2, *11.13.1.3*, *15.9.10.1*, *B.5.2.3*, *B.5.3.1*, *B.5.23.2*
Parameter list: *12.3*, 13.5, 13.7, 14.7, *15.2.1*, 15.7, *15.9.1*, *15.13.4.1*, *15.16.2*, *15.18.2*, *A.3.3.19*, *A.3.3.22*, *E.2.3.2*
Parameter: *3.6.13*, *3.6.66*, *3.6.68*, d), *11.15.2*, *11.19*, 13.5, *14.5*, *15.9.4*, *15.10.3*, *A.3.3.19*, *A.3.3.22*, *A.3.3.34*, *A.4.2.7*, *G.6*
Parameterization: *3.6.25*, *11.1*, *11.4*, *15.18.2*, */* STATIC SEMANTICS -*, *A.3.3.19*, *A.3.3.22*, *A.3.3.23*
Parameterized compact constraints: *E.2.3.2*
Parameterized constraint: *3.6.7*, 12.3, 13.5, *A.4.2.11*, F.1.2.4, F.1.2.5, F.2.2.4
PASS: *15.17.1*, *15.17.2*, *15.17.3*, *15.17.4*, *A.4.2.4*, *B.5.23.2*
Pass: *3.6.54*
Passing of constraints: 15.13.5
Passing parameters: 15.16.2
PCO declaration: 11.10, *11.15.2*
PCO model: *15.9.1*
PCO queue: *15.9.2*
PCO type: 11.9, *11.15.2*, *12.3*, *15.7.2*
PCO: *3.6.57*, *3.6.60*, *3.6.72*, *3.6.73*, **4.1**, *8.1*, *8.2*, *9.5*, *10.7*, *11.3.4.1*, *11.9*, *11.10*, *11.11*, *11.13.1.1*, *11.13.1.3*, *11.13.2*, *11.14.2*, *11.14.4*, *11.14.5*, *11.15.1*, *11.15.2*, *11.15.4*, *11.15.5*, *15.2.4*, *15.3.1*, *15.4.1*, *15.9*, *15.9.1*, *15.9.5.3*, *15.9.5.4*, *15.9.6*, *15.9.7*, *15.9.8*, *15.9.10.1*, *15.18.1*, *15.18.8*, *A.4.2.13*, *B.1*, *B.5.4.2*, *B.5.8.2*, *B.5.9.2*, *B.5.10.2*, *B.5.11.2*, *B.5.12.2*, *F.11*, G.10
PDU constraint compact proforma: *E.2.3*
PDU constraint declaration: *3.6.62*, *13.2*, 13.4, *A.5.1*
PDU constraints: 7.3.4, *11.16.3*, *12.6.6.1*, *13.4*, *14.1*
PDU field value: *11.20*, *12.2*, *12.4*, *12.6.4.5*, *12.6.4.6*, *15.9.3*, *15.9.4*
PDU field: *3.6.66*, *11.2.1*, *11.16.3*, *11.17.1*, *12.1*, *12.5*, *12.6.2*, 12.6.3, *12.6.4.1*, *12.6.4.2*, *12.6.4.3*, *12.6.4.4*, *12.6.4.5*, *12.6.4.7*, *12.6.4.8*, *12.6.5.1*, *12.6.5.3*, *12.6.6.2*
PDU identifier: *11.15.2*, *11.21*, *15.9.1*
PDU specification in ASN.1: 11.15.5
PDU type definition: *3.6.3*, *3.6.68*, 11.15, 11.19, 11.20, *13.4*, *E.2.3*, F.4, *G.6*
PDU type: *11.3.4.2*, *11.8.1*, *11.8.3*, *13.4*, *14.4*, *15.7.2*

PDU: *3.6.1*, *3.6.9*, *3.6.13*, *3.6.25*, *3.6.38*, *3.6.44*, *3.6.57*, *3.6.60*, *3.6.66*, *3.6.68*, **4.3**, *7.3.1*, *9.5*, *11.2.1*, *11.2.2*, *11.2.3.2*, *11.2.3.3*, *11.2.3.4*, *11.2.3.5*, *11.3.4.1*, *11.3.4.2*, *11.6*, *11.7*, *11.10*, *11.14.2*, *11.15.1*, *11.15.2*, *11.15.3*, *11.15.4*, *11.15.5*, *11.16.2*, *11.16.4*, *11.17.1*, *11.17.2*, *11.17.3*, 12.6.3, *13.2*, *13.6*, *14.5*, *14.6*, *14.8*, *15.9*, *15.9.5.3*, *15.9.5.4*, *15.9.6*, *15.10.1*, *15.10.2.2*, *15.10.2.3*, *15.10.3*, *15.10.4.1*, *15.10.6*, *15.16.1*, *15.18.8*, *A.3.3.19*, *A.3.3.22*, *A.3.3.34*, *A.4.2.4*, *A.4.2.5*, *A.4.2.7*, *A.4.2.8*, *B.5.2.3*, *B.5.8.2*, *B.5.9.2*, *B.5.10.2*, *B.5.11.2*, *B.5.12.2*, *B.5.13.2*, *B.5.16.2*, *E.2.1*, *G.3.1*
PERMUTATION: *A.4.2.4*
Permutation: **12.6.5.3**, *12.6.6.1*
PICS proforma: *11.4*
PICS: *3.6.80*, *3.6.81*, **4.1**, *10.3*, *11.4*, *11.6*, *11.7*, *11.12*, *C.2.2*
PIXIT proforma: *11.4*
PIXIT: *3.6.80*, *3.6.81*, **4.1**, *10.3*, *11.4*, *11.6*, *11.7*, *11.10*, *11.12*, *15.9.6*, *C.2.2*
Point of attachment: *15.13.5*
Point of control and observation: *4.1*, *8.1*
Postamble: *G.2*
Preamble: *G.2*
Precautions for concurrent TTCN: 15.9.5.4
Precedence of assignments and qualifiers: 15.10.6
Precedence of operators: Table 3 -
Precedence of pseudo-events: 15.11
Precedence: *15.17.2*, *A.4.2.11*, B.2, *G.8*
Predefined type: *11.3.4.2*, *d)*, *11.6*, *11.7*, *11.8.1*, *11.15.2*, *11.16.3*
Predefined variable: *3.6.41*
Preliminary result variable: *B.5.4.2*
Preliminary result: *3.6.34*, *3.6.41*, **3.6.54**, *3.6.58*, *11.13.1.1*, *11.13.1.2*, *15.9.10.2*, *15.17.1*, 15.17.2
PRESENT: *A.4.2.5*
PrintableString: *A.4.2.5*
PRIVATE: *A.4.2.5*
Procedural definition of test suite operation: *A.4.2.14*
Procedural definition: *11.3.4.3*
Procedure statement: *11.3.4.4*
Protocol Data Unit: *1*, *4.3*
Protocol error: *15.17.2*
Protocol Implementation Conformance Statement: *4.1*
Protocol Implementation eXtra Information for Testing: *4.1*
ps: *A.4.2.4*
Pseudo-code keywords: B.5.2.1
Pseudo-code notation: B.5.2
Pseudo-code precedence: *B.2*
Pseudo-code procedures and functions: B.5.2.2
Pseudo-code process: B.5.2.3
Pseudo-code with natural language: B.5.2.4
Pseudo-code: *B.5.2.3*, *B.5.2.4*, *B.5.5.4*
Pseudo-event: **3.6.55**, *3.6.61*, *3.6.90*, *15.8*, 15.11, *B.5.1*, *B.5.5.4*, B.5.14, *B.5.14.2*
PTC: *3.6.58*, **4.3**, *8.1*, *8.2*, *11.13.1.1*, *11.13.1.2*, *11.13.2*, *15.2.4*, *15.9.10.1*, *15.9.10.2*, *15.17.5*, *15.18.7*

Q

Qualified event: **3.6.56**
Qualifier evaluation: 15.10.6
Qualifier: *15.6*, *15.8*, *15.9.2*, *15.10.4.1*, 15.10.5, *15.11*, *15.15*, 15.16.3
Queue: *15.9.2*

R

R: *3.6.58*, *15.17.2*, *15.17.3*, *15.17.5*, *15.18.1*, *B.5.23.2*, *G.2*
R_TYPE: **11.2.2**, *15.17.2*, *15.17.5*
R_Type: *A.4.2.4*
Range: *11.18.2*, **12.6.4.6**, *12.6.6.1*
READ_TIMER: **B.5.17.1**
READTIMER operation: **15.12.4**
READTIMER: *15.12.1*, *15.12.4*, *A.4.2.4*, *B.5.14.2*, *B.5.17*
REAL: *A.4.2.5*
Receive event: **3.6.57**, 11.20, *12.1*, *12.2*, 15.9.2, *15.10.4.1*, *A.3.3.33*
RECEIVE function: **B.5.9.1**
RECEIVE: *8.1*, *8.2*, *11.16.4*, *15.9.4*, *15.9.5.3*, *15.9.6*, *15.10.6*, *15.16.1*, *B.5.7.2*, *B.5.9.2*, *B.5.15.2*

ReceiveObject: *B.5.2.3*
Record references: 15.10.2.2
Recursive tree attachment: 15.13.6
References in chaining of constraints: 15.10.2.2
References using tables: 15.10.3
RELABEL: **B.5.25**
Relational operators: 11.3.2.3
Remote test method: *3.6.38*, *4.2*, *15.9.6*, G.3.5
REMOVE_OBJECT: **B.5.9.1**
REPEAT construct: **15.15**
REPEAT: *15.6*, *15.8*, *15.15*, *15.17.1*, *A.4.2.4*, *B.5.1*, *B.5.5*, *B.5.5.1*, B.5.5.3, *B.5.5.5*, *B.5.18.2*
RepeatTree: *B.5.5.3*
REPLACE: *14.6*, *A.4.2.4*
REPLACE_ALT_TREE: **B.5.25**
REPLACE_PARAMETERS: **B.5.25**
Restrictions on using events: 15.9.5.3
Result type: *11.3.4.5*
Result variable: **3.6.58**, *3.6.58*
RETURN statement: **15.18.3**
RETURN: *15.18.1*, *15.18.3*, 15.18.6, *15.18.6*, *B.1*, *B.5.2.3*, *B.5.18.2*, *B.5.22*
ReturnDefaults: *B.5.2.3*
ReturnLevel: *B.5.2.3*
RETURNVALUE: *11.3.4.1*, *11.3.4.5*, *A.4.2.4*
Root tree: **3.6.59**, *15.6*, *15.13.3*, *15.13.4.1*, *15.14*, *15.18.5*, *A.4.2.9*, *A.5.2*
ROOT_TREE: **B.5.25**
RS: **4.2**

S

SAP: **4.3**, *11.10*, *G.10*
SAVE_DEFAULTS: **B.5.5.2**
Scope of tree attachment: 15.13.2
Scoping rules: *15.13.4.1*
sec: *A.4.2.4*
Selection expression: *3.6.52*, 11.5, *F.7*
Selection: *11.1*, *b)*, *F.7*
Semantics of TTCN: B.1
Send event: **3.6.60**, 11.19, *12.1*, *12.2*, 15.9.3, *15.10.4.1*, B.5.8, G.9
SEND function: **B.5.8.1**
SEND: *8.1*, *8.2*, *11.10*, *12.5*, *15.9.4*, *15.10.6*, *B.5.7.2*, *B.5.15.2*
SEND_EVENT: **B.5.8.1**
SendObject: *B.5.2.3*
SEQUENCE OF INTEGER: *12.6.5.1*, *12.6.5.3*
SEQUENCE OF: *11.3.3.3.3*, *11.18.2*, 12.6.3, *12.6.5.1*, *12.6.5.2*, *12.6.5.3*
SEQUENCE: 12.6.3, *14.5*, *14.8*, *15.10.2.2*, *15.10.2.3*, *15.10.2.4*, *A.4.2.5*
Service Access Point: *4.3*
Set of alternatives: **3.6.61**, *15.6*, *15.9.5.2*, *15.9.9*, *15.13.4.1*, *15.18.5*, *A.3.3.33*, *B.5.5.4*, *B.5.5.5*
SET OF: *11.3.3.3.3*, *11.18.2*, *12.5*, 12.6.3, *12.6.4.7*, *12.6.4.8*, *12.6.5.1*, *12.6.5.2*, *12.6.6.1*
SET: *12.5*, 12.6.3, *14.5*, *14.8*, *15.10.2.2*, *15.10.2.3*, *A.4.2.5*
Simple CMs: 11.17.1
Simple type: 11.2.3.2, *11.6*, *11.7*, *11.14.2*, *11.14.3*, *11.15.2*, *11.15.3*
Single constraint table: **3.6.62**, *13.1*, *E.1*, *E.2.1*, *E.2.4*
SIZE: *A.4.2.5*
Snapshot semantics: **3.6.63**, 15.9.5.2
SNAPSHOT: *B.5.12.2*
SNAPSHOT_FIXED: **B.5.26**
Specific value: **3.6.65**, *12.2*, 12.6.3, *12.6.4.5*, *12.6.6.1*, *15.9.3*
Splitting and Recombining: F.12
SPyT: *3.6.34*
Stable testing state: *15.17.3*
Standardized ATS: *6.5*, *G.8*
START operation: **15.12.2**
START: *15.12.1*, *15.12.2*, *A.4.2.4*, *B.5.14.2*
START_TIMER: **B.5.17.1**
STATEMEMT_LINE_TYPE_OF: **B.5.26**
Statement line: *B.1*

StatementLine: *B.5.2.5*
Static chaining: **3.6.66**, *12.4*
Static conformance requirements: *1*
STATIC SEMANTICS: *A.4.1*
Static semantics: **3.6.67**, *5*, Annex A, 390, *B.1*
STATIC: *11.3.4.3*, *A.4.2.4*
Step-wise expansion: *B.5.2.1*
STOP_TEST_CASE: **B.5.26**
STRING: *A.4.2.5*
Structure: *15.10.3*
Structured type constraint declaration: 13.2
Structured type constraints: 7.3.4, *A.4.2.15*, E.2.4
Structured type: *3.6.9*, **3.6.68**, 11.2.3.3, *11.2.3.3*, *11.14.2*, *11.14.3*, *11.15.2*, *11.15.3*, *11.18.1*, *11.20*, *12.6.1*, 12.6.3, *13.1*, *13.2*, *13.4*, *15.10.3*, *A.3.3.19*, *A.3.3.22*, *A.4.2.8*, *E.2.1*, *E.2.4*, G.6
Structured types within ASP type: 11.14.3
Style guide: Annex G
Submodule: **3.6.69**
Subsequent behaviour: *15.13.3*
SUBSEQUENT_BEHAVIOUR_TO: **B.5.25**
SUBSET: *A.4.2.4*
SubSet: **12.6.4.8**, *12.6.6.1*
Substructure: *3.6.68*, *11.20*, *13.3*, *15.10.3*, *A.3.3.19*, *A.3.3.22*, *A.3.3.34*
Subtree: *B.5.5.4*, *G.4*
Suite overview part: 9.5
Suite overview: *10.1*
SUPERSET: *A.4.2.4*
SuperSet: **12.6.4.7**, *12.6.6.1*
SUT: **4.1**
Syntactic metanotation: A.2.1
Syntax definition: 5
Syntax forms of TTCN: 5
Syntax production: *5*, A.3
SYNTAX: *A.4.2.5*
System Under Test: *4.1*

T
T61String: *A.4.2.5*
Tabular ASP type definition: *13.1*
Tabular PDU type definition: *13.1*
TAKE_SNAPSHOT: **B.5.26**
TCP: **4.3**
TeletexString: *A.4.2.5*
TERMINATE_TEST_CASE: **B.5.26**
Test body: *G.2*
Test case dynamic behaviour: *7.3.1*, 7.3.4, 9.5, 15.2, *15.18.2*, *A.5.1*, *A.5.2*, E.3, E.3.2
Test case error processing: B.3
Test case error: *11.3.3.2.4*, *11.3.3.2.5*, *11.16.1*, *11.16.2*, *15.9.3*, *15.9.10.1*, *15.12.2*, *15.17.3*, *B.5.4.2*
Test case execution pseudo-code: B.5.4.1
Test case execution, natural language: B.5.4.2
Test case identifier: **3.6.70**
Test case index: 10.1, 10.4, *b)*, *A.5.2*
Test case root tree: *15.7.2*
Test case selection expression: 10.3
Test case selection: *11.1*, *d)*, b)
Test case termination: 11.8.4
Test case variable: *3.6.34*, *3.6.58*, **3.6.71**, *7.3.1*, *11.6*, *11.7*, *11.8.1*, 11.8.3, *11.8.4*, *11.12*, *12.3*, *15.10.1*, *15.10.4.1*, *15.13.1*, *15.17.2*, *B.5.20.2*
Test case writer: G.5
Test case: *1*, *3.6.10*, *3.6.11*, *3.6.12*, *3.6.23*, *3.6.26*, *3.6.34*, *3.6.47*, *3.6.52*, *3.6.54*, *3.6.59*, *3.6.61*, *3.6.63*, *3.6.70*, *3.6.71*, *3.6.73*, *3.6.74*, *3.6.82*, *3.6.89*, *9.1*, *9.2*, *9.3.1*, *9.5*, *10.3*, *10.4*, *d)*, *b)*, *11.8.3*, *11.8.4*, *15.1*, *15.2.1*, *15.3.1*, *15.4.1*, *15.9.5.1*, *15.9.10.1*, *15.12.1*, *15.12.4*, *15.13.2*, *15.14*, *15.17.2*, *15.18.1*, *15.18.4*, *A.4.2.13*, *B.5.2.1*, *B.5.2.3*, *B.5.3.1*, B.5.4, *E.3.2*, G.2, *G.5*, *G.8*
Test component configuration declaration: 8.1
Test component configuration: *3.6.12*, *3.6.16*, *3.6.43*, **3.6.73**, 8.2, *11.13.1.3*, 11.13.2, *15.2.4*, *A.4.2.13*
Test component declaration: 8.1, 11.13.1.3

Test component: *3.6.12*, *3.6.15*, *3.6.16*, *3.6.41*, *3.6.43*, *3.6.53*, **3.6.72**, *3.6.73*, *11.12*, *15.9.10.2*
Test coordination procedures: *4.3*
Test event: *3.6.5*, *3.6.6*, *3.6.91*, *15.8*, 15.9, *15.10.4.1*, *A.5.1*
Test group identifier: *A.5.1*, *A.5.2*
Test group objective: 10.3, *C.2.3*
Test group reference: **3.6.74**, 9.2, *10.3*, *10.4*, *15.2.1*, *A.5.1*, *C.2.3*
Test group: *3.6.10*, *3.6.52*, *9.1*, *9.2*, *10.3*, *10.4*, *A.5.2*, *C.2.3*, *E.3.1*
Test laboratory: *6.5*
Test management protocol: *4.1*
Test method: G.3
Test outcome: *3.6.91*
Test purpose: *15.2.1*, *G.2*, *G.8*
Test realizer: *G.5*
Test result: *3.6.54*
Test step dynamic behaviour: *3.6.78*, 9.5, 15.3, *15.18.2*
Test step group reference: **3.6.76**, *9.3.2*, *10.5*
Test step group: **3.6.75**, *9.1*, *9.3.1*, *10.6*
Test step identifier: **3.6.77**, *10.5*, *A.4.2.11*
Test step index: 10.1, 10.5, *A.5.1*
Test step library: *3.6.52*, *3.6.76*, **3.6.78**, *3.6.84*, *9.3.1*, *9.3.2*, *10.5*, *15.3.1*, *15.13.2*, *15.13.3*, *15.15*, *15.18.5*, *A.4.2.10*, *G.2*
Test step objective: **3.6.79**, *10.5*, *15.3.1*
Test step root tree: *15.7.2*
Test step: *3.6.2*, *3.6.8*, *3.6.23*, *3.6.26*, *3.6.75*, *3.6.76*, *3.6.77*, *3.6.79*, *3.6.84*, *3.6.87*, *9.1*, *9.3.1*, *9.3.2*, *10.5*, *15.1*, *15.2.3*, *15.3.1*, *15.4.1*, *15.9.5.1*, *15.9.10.1*, *15.13.2*, *15.13.3*, *15.13.4.1*, *15.13.5*, *15.15*, *15.18.1*, *15.18.5*, *A.4.2.12*, *B.5.5.5*
Test suite constant: **3.6.80**, *11.2.1*, *b)*, 11.6, *11.6*, *11.7*, *11.14.2*, *11.15.2*, *12.3*, *B.5.2.3*
Test suite constants: *11.16.1*, *11.16.2*, *11.17.2*, *15.10.1*
Test suite exports: 10.1
Test suite index: *10.1*, **10.2**
Test suite operation description: 11.3.4
Test suite operation procedural definition: 11.3.4
Test suite operation, assignment: 11.3.4.6
Test suite operation, CASE: 11.3.4.9
Test suite operation, IF: 11.3.4.7
Test suite operation, parameter passing: 11.3.4.2
Test suite operation, RETURNVALUE: 11.3.4.5
Test suite operation, variables: 11.3.4.3
Test suite operation, WHILE: 11.3.4.8
Test suite operation: *11.3.4.2*, *11.3.4.3*, *11.16.3*, *A.4.2.14*
Test suite operations: F.6
Test suite parameter: **3.6.81**, *11.2.1*, 11.4, *b)*, *11.15.2*, *11.16.1*, *11.16.2*, *11.17.2*, *12.3*, *15.10.1*, *B.5.2.3*, *F.7*
Test suite parameters: *11.14.2*
Test suite specifier: *15.9.5.1*, *G.1*, *G.2*, *G.4*
Test suite structure: 9, 10.1, 10.3, *15.2.1*, *A.5.1*, *A.5.2*, *F.7*
Test suite type definition: 11.2, *11.15.2*, *12.6.6.1*
Test suite type: *11.2.3.4*, *11.3.4.1*, *11.3.4.2*, *11.8.1*, *11.8.3*, *11.14.2*, *11.16.3*, *11.17.2*, *14.2*
Test suite variable: **3.6.82**, *11.2.1*, *11.6*, *11.7*, 11.8.1, *11.8.1*, *11.8.2*, *11.8.3*, *11.12*, *11.13.1.1*, *11.13.1.2*, *12.3*, *15.10.4.1*, *15.13.1*, *B.5.2.3*
Test suite: *3.6.4*, *3.6.13*, *3.6.17*, *3.6.22*, *3.6.26*, *3.6.27*, *3.6.29*, *3.6.32*, *3.6.45*, *3.6.48*, *3.6.50*, *3.6.52*, *3.6.71*, *3.6.78*, *3.6.80*, *3.6.81*, *3.6.82*, *3.6.91*, *9.1*, *9.2*, *10.1*, *10.8.1*, *11.2.1*, *11.2.3.2*, *11.4*, *11.12*, *11.15.2*, *15.12.4*, *A.4.2.6*, *A.4.2.10*
Test system: *12.1*
Test verdict: *3.6.34*, *3.6.43*
Textual substitution: *15.13.4.1*, *B.5.20.2*
THEN: *A.4.2.4*
Timeout event: **3.6.83**, **15.9.9**
TIMEOUT function: **B.5.11.1**
TIMEOUT: *15.8*, *15.9.5.2*, *15.9.5.3*, *15.9.9*, *15.12.3*, *A.3.3.33*, *A.4.2.4*, *B.5.7.2*, *B.5.11.2*, *B.5.15.2*, *G.5*
Timer declaration: 11.12
Timer management: 15.12
Timer name: *15.9.9*
Timer operation: *3.6.55*, *15.8*, *15.11*, 15.12.1, B.5.17
Timer value: *15.12.2*
Timer: *3.6.83*, *15.9.9*, *G.5*
TIMER_EXPIRED: **B.5.11.1**
TIMER_OP_TYPE_OF: **B.5.26**

TIMER_OPS: **B.5.17.1**
TMP: **4.1**, *10.3*
TO: *11.18.2*, *12.6.4.6*, *A.4.2.4*
Transfer syntax: *A.1*
Transformation algorithm: *B.1*
Tree and Tabular Combined Notation: *4.2*
Tree attach symbol: *15.13.3*
Tree attachment: **3.6.84**, *15.4.1*, *c)*, 15.13, *15.13.1*, 15.13.2, 15.13.3, *NOTE* -, 15.18.5, 15.18.6, B.5.5.5, *G.2*, *G.5*
Tree header: **3.6.85**, *A.4.2.10*, *A.4.2.11*
Tree identifier: *3.6.85*, **3.6.86**, *A.4.2.10*
Tree leaf: **3.6.87**
Tree name: 15.7
Tree node: **3.6.88**
Tree notation: **3.6.89**, *15.2.1.3*, 15.6
TreeReference: *B.5.5.3*
Trees with parameters: 15.7.2
TRUE: *10.3*, *10.4*, *11.2.2*, *11.3.3.3.1*, *11.3.3.3.2*, *11.3.4.7*, *11.3.4.8*, *b)*, *11.16.1*, *11.16.2*, *15.6*, *15.10.5*, *15.10.6*, *15.11*, *15.12.1*, *15.15*, *A.4.2.5*, *B.5.8.2*, *B.5.9.2*, *B.5.10.2*, *B.5.11.2*, *B.5.12.2*, *B.5.15.2*
TTCN ASP constraints: *A.4.2.15*
TTCN CM constraints: *A.4.2.15*
TTCN expression: *3.6.55*, 15.10
TTCN graphical form: *4.3*
TTCN machine: *B.1*, *B.5.2.3*, *B.5.3.1*
TTCN machine-processable form: *4.3*
TTCN module exports: *C.2.1*
TTCN module overview part: *C.1*, C.2
TTCN module structure: *C.2.1*
TTCN object: *7.3.1*, *7.3.2*, *7.3.3*, *7.3.4*
TTCN operations: 11.3
TTCN operators: 11.3
TTCN PDU constraints: *A.4.2.15*
TTCN semantics: *B.5.2.1*
TTCN statement: *3.6.2*, *3.6.6*, *3.6.18*, *3.6.61*, *3.6.87*, *3.6.88*, **3.6.90**, *15.2.1.3*, *15.2.3*, *15.5*, *15.6*, 15.8, *15.16.1*, *B.5.1*
TTCN type: 11.2
TTCN.GR: **4.3**, *5*, *6*, *7.1*, *7.3.5*, *7.4*, *15.6*, *A.1*, *A.4.1*, A.5
TTCN.MP: **4.3**, *5*, *6*, *7.1*, *7.4*, *11.2.3.4*, *11.14.4*, *11.15.4*, *14.1*, *15.6*, *A.1*, *A.4.1*, A.5, *E.1*, F.8
TTCN: **4.2**
Type definition using macros: F.4
Type definitions using ASN.1: 11.2.3.4
Type list: *11.16.3*
Type: *11.16.3*
TYPEIDENTIFIER: *A.4.2.5*

U

Unbound variable: *3.6.65*, *15.10.4.1*
Unbound variables: *11.3.4.3*
Underscore symbol: *11.14.4*, *11.15.4*
Unforeseen test event: *3.6.51*, **3.6.91**
Unforseen test events: 15.9.7
UNION: *A.4.2.5*
UNIQUE: *A.4.2.5*
Units of length: 11.18.2
UNIVERSAL: *A.4.2.5*
UniversalString: *A.4.2.5*
Unqualified event: **3.6.92**
UNTIL: *A.4.2.4*
UPDATE_PRELIM: **B.5.23.1**
Upper tester: *4.1*, *11.13.1.2*
us: *A.4.2.4*
Use of REPEAT: F.5
UT: **4.1**, *11.9*, *11.10*, *15.2.1.3*, *15.9.1*, *15.9.5.1*, *15.9.7*, *A.4.2.4*, *B.5.8.2*, *B.5.9.2*, *B.5.10.2*, *B.5.11.2*, *B.5.12.2*, *G.4*
UTCTime: *A.4.2.5*

V
Value: *11.3.4.2*
ValueList: **12.6.4.5**
VAR: *11.3.4.3*
Variable declaration: *A.4.2.14*
Variable name: *A.4.2.14*
Variables: *11.3.4.3*
Verdict assignment: 15.17.5
Verdict: *3.6.5*, *11.13.1.1*, *15.2.1.3*, *15.2.3*, 15.17, B.5.22, *B.5.23.2*, *G.2*
VideotexString: VisibleString: *A.4.2.5*
VisibleString: *A.4.2.5*

W
WHILE DO: *A.4.2.4*
WHILE: *A.4.2.4*
Wildcards: *12.5*
WITH: A.4.2.5

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie B	Medios de expresión: definiciones, símbolos, clasificación
Serie C	Estadísticas generales de telecomunicaciones
Serie D	Principios generales de tarificación
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedios
Serie I	Red digital de servicios integrados
Serie J	Redes de cable y transmisión de programas radiofónicos y televisivos, y de otras señales multimedios
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	RGT y mantenimiento de redes: sistemas de transmisión, circuitos telefónicos, telegrafía, facsímil y circuitos arrendados internacionales
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de transmisión telefónica, instalaciones telefónicas y redes locales
Serie Q	Conmutación y señalización
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	**Redes de datos y comunicación entre sistemas abiertos**
Serie Y	Infraestructura mundial de la información y aspectos del protocolo Internet
Serie Z	Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación

22905

Impreso en Suiza

Ginebra, 2003