Reemplazada por una versión más reciente

UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

CCITT X.292

COMITÉ CONSULTIVO
INTERNACIONAL
TELEGRÁFICO Y TELEFÓNICO

(09/92)

REDES DE COMUNICACIÓN DE DATOS

METODOLOGÍA Y MARCO DE LAS PRUEBAS DE CONFORMIDAD PARA INTERCONEXIÓN DE SISTEMAS ABIERTOS DE LAS RECOMENDACIONES SOBRE LOS PROTOCOLOS PARA APLICACIONES DEL CCITT

Notación combinada arborescente y tabular

Reemplazada por una versión más reciente

Recomendación X.292

PREFACIO

El CCITT (Comité Consultivo Internacional Telegráfico y Telefónico) es un órgano permanente de la Unión Internacional de Telecomunicaciones (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 Plenaria del CCITT, que se celebra cada cuatro años, establece los temas que han de estudiarse y aprueba las Recomendaciones preparadas por sus Comisiones de Estudio. La aprobación de Recomendaciones por los miembros del CCITT entre las Asambleas Plenarias de éste es el objeto del procedimiento establecido en la Resolución N.º 2 del CCITT (Melbourne, 1988).

La Recomendación X.292 ha sido preparada por la Comisión de Estudio VII y fue aprobada por el procedimiento de la Resolución N.º 2 el 10 de septiembre de 1992.

NOTA DEL CCITT

En esta Recomendación, la expresión «Administración» se utiliza para designar, en forma abreviada, tanto una Administración de telecomunicaciones como una empresa privada de explotación reconocida de telecomu-nicaciones.

Reemplazada por una versión más reciente

ÍNDICE

Página

Recomendación X.292

METODOLOGÍA Y MARCO DE LAS PRUEBAS DE CONFORMIDAD PARA INTERCONEXIÓN DE SISTEMAS ABIERTOS DE LAS RECOMENDACIONES SOBRE LOS PROTOCOLOS PARA APLICACIONES DEL CCITT

Notación combinada arborescente y tabular[1)]

(1992)

El CCITT,

considerando

(a) que la Recomendación X.200 define el modelo de referencia de interconexión de sistemas abiertos (OSI) para aplicaciones del CCITT;

(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 deben elaborarse sucesiones de pruebas normalizadas para cada Recomendación sobre protocolos de OSI como un medio de:

– 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 superan las pruebas de conformidad normalizadas;

(d) la necesidad de normalizar el proceso de prueba de conformidad para lograr un grado aceptable y útil de comparabilidad 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 sea acorde con esta Recomendación.

0 Introducción

En esta Recomendación se define una notación de prueba informal denominada notación combinada arborescente y tabular (TTCN, *tree and tabular combined notation*), para su utilización en la especificación de sucesiones de pruebas abstractas de conformidad de OSI.

En el diseño de una sucesión 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.

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

a) proporcionar una notación con la que puedan expresarse casos de pruebas abstractas en sucesiones 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.

En la metodología de comprobación abstracta, se contempla una sucesión de pruebas como una jerarquía que se extiende desde la sucesión 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

---

1) La Recomendación X.292 y la parte 3 de la norma ISO/CEI 9646 «Information Processing Systems – Open Systems Interconnection – Conformance testing methodology and framework – Part 3: Tree and Tabular Combined Notation (TTCN)» están técnicamente armonizadas.

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 abstracto (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 el hombre denominada TTCN.GR, para su utilización en las normas de sucesiones 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 computador. Ambas formas son semánticamente equivalentes.

Esta Recomendación se ha publicado también como norma ISO/CEI 9646 parte 3. El anexo F de la norma ISO/CEI 9646 parte 3, contiene un resumen de las diferencias entre las versiones DIS e IS de TTCN.

1 Ámbito

En esta Recomendación se define una notación de prueba informal denominada notación combinada arborescente y tabular (TTCN) para las sucesiones 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 X.290 y X.291.

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

Esta Recomendación se aplica a la especificación de casos de pruebas de conformidad que pueden formularse de forma abstracta en términos de control y la observación de unidades de datos de protocolo y primitivas de servicio abstracto. Sin embargo, para algunos protocolos, puede ser necesario utilizar casos de prueba que no pueden formularse 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 sobre sucesiones de pruebas de conformidad.

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

Esta Recomendación especifica los requisitos que podría establecer una Recomendación relativa a la sucesión de pruebas en lo que respecta a una realización conforme de la sucesión de pruebas, incluida la semántica operacional de las sucesiones de pruebas con TTCN.

Nota – La Recomendación X.293 especifica requisitos relativos a la realización de pruebas, incluida la derivación de sucesiones de pruebas ejecutables (ETS).

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

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

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

c) la especificación de casos de prueba en los cuales intervienen concurrentemente más de una descripción de comportamiento;

   Nota – La utilización de árboles paralelos y la sincronización entre ellos quedará cubierta por una futura enmienda a esta Recomendación;

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

2 Referencias

- Recomendación X.200 (1988), *Modelo de referencia de interconexión de sistemas abiertos para aplicaciones del CCITT*. (Véase también la norma ISO 7498.)
- Recomendación X.210 (1988), *Convenios relativos a la definición del servicio de capa en la interconexión de sistemas abiertos*. (Véase también la norma ISO/TR 8509.)
- Recomendación X.208 (1988), *Especificación de la notación de sintaxis abstracta uno (ASN.1)*. (Véase también la norma ISO/CEI 8824.)
- Recomendación X.209 (1988), *Especificación de las reglas básicas de codificación de la notación de sintaxis abstracta uno (ASN.1)*. (Véase también la norma ISO/CEI 8825.)
- Recomendación X.290 (1992), *Metodología y marco de las pruebas de conformidad de interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del CCITT – Conceptos generales*. (Véase también la parte 1 de la norma ISO/CEI 9646.)
- Recomendación X.291 (1992), *Metodología y marco de las pruebas de conformidad de interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del CCITT. Especificación de sucesiones de pruebas abstractas*. (Véase también la parte 2 de la norma ISO/CEI 9646).
- Recomendación X.293 (1992), *Metodología y marco de las pruebas de conformidad de interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del CCITT – Realización de pruebas*. (Véase también la parte 4 de la norma ISO/CEI 9646.)
- Recomendación X.294 (1992), *Metodología y marco de las pruebas de conformidad de interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del CCITT – Requisitos que deberán cumplir los laboratorios de pruebas y los clientes en el proceso de evaluación de la conformidad*. (Véase también la parte 5 de la norma ISO/CEI 9646.)

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

  ISO/CEI 10646-1: . . . [1], *Information technology – Multiple-Octet Coded Character Set – Part 1: Architecture and Basic Multilingual Plane*.

3 Definiciones

3.1 *Definiciones básicas extraídas de la Recomendación X.290*

Se aplican los siguientes términos definidos en la Recomendación X.290:

a) primitiva de servicio abstracta,
b) metodología de comprobación abstracta,
c) caso de prueba abstracta,
d) método de prueba abstracta,
e) sucesión de pruebas abstractas,
f) registro cronológico de conformidad,
g) sucesión 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) sucesión de pruebas ejecutables,
m) veredicto de fracaso,

---

[1] Actualmente en estado de proyecto.

n) estado de comprobación en reposo,

o) realización 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) enunciado de conformidad de realización de protocolo,

y) información suplementaria de realización de protocolo para pruebas,

z) punto de control y observación,

aa) método de prueba a distancia,

ab) estado de comprobación estable,

ac) sucesión 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 pruebas,

am) objetivo de grupo de pruebas,

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) sucesión de pruebas,

ax) sistema de prueba,

ay) probador superior,

az) veredicto (de una prueba),

ba) estado de comprobación.

3.2 *Términos de la Recomendación X.200*

Se aplican los siguientes términos definidos en la Recomendación X.200:

a) capa de aplicación,

b) unidad de datos de protocolo,

c) punto de acceso al servicio,

d) capa de sesión,

e) subred,

f) sintaxis de transferencia,

g) capa de transporte.

3.3 *Términos de la Recomendación X.210*

Se aplica el siguiente término definido en la Recomendación X.210:

Proveedor del servicio,

3.4 *Términos de la Recomendación X.208*

Se aplican los siguientes términos definidos en la Recomendación X.208:

a) tipo bitstring (cadena de bits),

b) tipo characterstring (cadena de caracteres),

c) tipo enumerated (enumerado),

d) tipo external (externo),

e) identificador de objeto,

f) tipo octectstring (cadena de octetos),

g) tipo real,

h) tipo selection (selección),

i) tipo sequence (secuencia),

j) tipo sequence-of (secuencia-de),

k) tipo set (conjunto),

l) tipo set-of (conjunto-de),

m) subtipo.

Nota – Cuando pueda existir ambigüedad dentro de los términos de la TTCN, tales términos irán precedidos por el término en ASN.1, como prefijo.

3.5 *Términos de la Recomendación X.209*

Se aplica el siguiente término definido en la Recomendación X.209:

Codificación.

3.6 *Términos específicos de la TTCN*

Para los fines de esta Recomendación, se aplicarán las siguientes definiciones:

3.6.1 **constructivo adjuntar**

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

3.6.2 **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.

3.6.3 **tipo de base**

El tipo del que se deriva un tipo definido en una sucesión de pruebas.

3.6.4 **línea de comportamiento**

Asiento en 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 de constricciones e información de comentario, según sea aplicable.

3.6.5 **árbol de comportamiento**

Especificación de un conjunto de secuencias de eventos de pruebas y otros enunciados en notación combinada arborescente y tabular.

3.6.6 **casilla en blanco**

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

3.6.7 **árbol llamante**

Árbol de comportamiento al que se adjunta un paso de prueba.

3.6.8 **cuadro de constricciones compacto**

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

3.6.9 **cuadro de casos de prueba compacto**

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

3.6.10 **parte constricciones**

Parte de una sucesión 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 realización 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 realización sometida a prueba.

3.6.11 **referencia de constricciones**

Referencia a una constricción contenida en una línea de comportamiento.

3.6.12 **parte declaraciones**

Parte de una sucesión de pruebas con notación combinada arborescente y tabular con la definición y/o declaración de todos los componentes no predefinidos que hayan de utilizarse en la sucesión de pruebas.

3.6.13 **comportamiento por defecto**

Eventos y otros enunciados en notación combinada arborescente y tabular que pueden producirse a cualquier nivel del árbol asociado y que aparecen indicados en el formulario de comportamiento por defecto.

3.6.14 **grupo de valores por defecto**

Un conjunto denominado de comportamientos por defecto.

3.6.15 **referencia de grupo de valores por defecto**

Trayecto que especifica la ubicación lógica de un elemento por defecto en la biblioteca de valores.

3.6.16 **identificador de valor por defecto**

Nombre exclusivo de un valor por defecto.

3.6.17 **biblioteca de valores por defecto**

Conjunto de comportamientos por defecto en una sucesión de pruebas.

3.6.18 **referencia a valores por defecto**

Referencia a un valor en la biblioteca de valores por defecto, desde un caso de prueba o un cuadro de pasos de prueba.

3.6.19 **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.

3.6.20 **encadenamiento dinámico**

Vinculación de declaraciones de constricciones de un parámetro de 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 mediante una parametrización. Las unidades de datos de protocolo que son encadenadas se especifican en la referencia de constricciones de una línea de comportamiento.

3.6.21 **parte dinámica**

Parte de una sucesión 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.

3.6.22 **evento enviar implícito**

Mecanismo utilizado en los métodos de prueba a distancia para especificar que la realización sometida a prueba debe iniciar una unidad de datos de protocolo o una primitiva de servicio abstracta concreta.

3.6.23 **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.

3.6.24 **árbol local**

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

3.6.25 **constricción modificado**

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 ese constricción de base.

3.6.26 **semántica operacional**

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

3.6.27 **evento en otro caso,** *otherwise*

Mecanismo de notación combinada arborescente y tabular previsto para manejar eventos de prueba imprevistos de una manera controlada.

3.6.28 **parte visión general (o visión de conjunto)**

Parte de una sucesión de pruebas en notación combinada arborescente y tabular relacionada con la presentación de una visión general de la estructura de la sucesión 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 a los casos de prueba, pasos de prueba y valores por defecto.

3.6.29 **resultado preliminar**

Resultado registrado antes de la finalización de un caso de prueba, que indica si la parte asociada del caso de prueba ha superado con éxito la prueba, ha fallado o no hay conclusión al respecto.

3.6.30 **seudo evento**

Un seudo evento 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.

3.6.31 **evento calificado**

Evento que tiene asociada una expresión booleana.

3.6.32 **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.

3.6.33 **árbol raíz**

Árbol de comportamiento principal de un caso de prueba que se presenta en el nivel de la entrada en el caso de prueba.

3.6.34 **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.

3.6.35 **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.

3.6.36 **cuadro de constricción único**

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.

3.6.37 **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 este entorno es congelado efectivamente durante un periodo prescrito.

3.6.38 **valor específico**

Valor en notación combinada arborescente y tabular que no contiene ningún mecanismo de concordancia o variable no vinculada.

3.6.39 **encadenamiento estático**

Vinculación de declaraciones de constricciones de un parámetro de 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.

3.6.40 **semántica estática**

Reglas semánticas que restringen la utilización de la sintaxis de notación combinada arborescente y tabular.

3.6.41 **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, pudiendo utilizarse para especificar una porción de estructura plana (*flat structure*) o subestructura dentro de la primitiva de servicio abstracta o la unidad de datos de protocolo.

3.6.42 **identificador de caso de prueba**

Nombre exclusivo de un caso de prueba.

3.6.43 **variable de caso de prueba**

Una de un conjunto de variables, declarada globalmente para la sucesión de pruebas, cuyo valor se retiene solamente para la ejecución de un solo caso de prueba.

3.6.44 **referencia de grupo de prueba**

Trayecto que especifica la ubicación lógica de un caso de prueba dentro de la estructura de la sucesión de pruebas abstractas.

3.6.45 **grupo de paso de prueba**

Un conjunto denominado de pasos de prueba.

3.6.46 **referencia de grupo de paso de prueba**

Trayecto que especifica la ubicación lógica de un paso de prueba en la biblioteca de pasos de prueba.

3.6.47 **identificador de paso de prueba**

Nombre exclusivo de un paso de prueba.

3.6.48 **biblioteca de pasos de prueba**

Conjunto de descripciones de comportamiento dinámico de pasos de prueba en la sucesión de pruebas, que no son pasos de prueba locales.

3.6.49 **objetivo de paso de prueba**

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

3.6.50 **constante de sucesión de pruebas**

Una de un conjunto de constantes *no* derivadas del enunciado de conformidad de realización de protocolo o información suplementaria de realización de protocolo para pruebas, que permanecerá constante en la sucesión de pruebas.

3.6.51 **parámetro de sucesión de pruebas**

Una de un conjunto de constantes derivadas del enunciado de conformidad de realización de protocolo o de la información suplementaria de realización de protocolo para pruebas, que parametriza globalmente una sucesión de pruebas.

3.6.52 **variable de sucesión de pruebas**

Una de un conjunto de variables declaradas globalmente para la sucesión de pruebas y que retienen sus valores en casos de prueba.

3.6.53 **evento de temporización**

Evento utilizado dentro de un árbol de comportamiento para verificar la expiración de un temporizador especificado.

3.6.54 **adjunción de árbol**

Método para indicar que 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) debe incluirse en el árbol de comportamiento vigente.

3.6.55 **encabezamiento de árbol**

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

3.6.56 **identificador de árbol**

Nombre que identifica un árbol local.

3.6.57 **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.

3.6.58 **nodo de árbol**

Enunciado en notación combinada arborescente y tabular única.

3.6.59 **notación de árbol**

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

3.6.60 **enunciado en notación combinada arborescente y tabular**

Evento, seudoevento o constructivo especificado en una descripción de comportamiento.

3.6.61 **evento de prueba imprevisto**

Evento de prueba no identificado como evento de prueba en el resultado de la prueba previsto en la sucesión de pruebas. Se maneja habitualmente utilizando el evento OTHERWISE (en otro caso).

3.6.62 **evento no calificado**

Evento que no tiene asociada una expresión booleana.

4 Abreviaturas

4.1 *Abreviaturas definidas en la Recomendación X.290*

A los efectos de esta Recomendación, se aplican las siguientes abreviaturas definidas en el § 4 de la Recomendación X.290:

ATS Sucesión de pruebas abstractas (*abstract test suite*)

ASP Primitiva de servicio abstracta (*abstract service primitive*)

ETS Sucesión de pruebas ejecutables (*executable test suite*)

IUT Realización sometida a prueba (*implementation under test*)

LT Probador inferior (*lower tester*)

MOT Medio de pruebas (*means of testing*)

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

PICS Enunciado de conformidad de realización de protocolo (*protocol implementation conformance statement*)

PIXIT Información suplementaria de realización 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*)

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

UT Probador superior (*upper tester*)

4.2 *Abreviaturas definidas en la Recomendación X.291*

A los efectos de esta Recomendación, se aplican las siguientes abreviaturas definidas en el § 4 de la Recomendación X.291:

CS Método de pruebas monocapa coordinado (*coordinated 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*)

4.3 *Otras abreviaturas*

A los efectos de esta Recomendación, se aplican asimismo las siguientes abreviaturas:

ACSE Elemento de servicio de control de asociación (*association control service element*)

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

BNF Forma Backus-Naur (forma ampliada de BNF utilizada en la TTCN) (*the extended BNF used in TTCN*)

DIS Proyecto de norma internacional (*draft international standard*)

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

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

FTAM Transferencia, acceso y gestión de ficheros (*file transfer, access and management*)

IS Norma internacional (*international standard*)

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

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

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 Formas de sintaxis de la TTCN

La TTCN se facilita bajo dos modalidades:

a) una forma gráfica (TTCN.GR) legible por el ser humano;

b) 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 empleando formularios tabulares. La TTCN.MP se define empleando 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). Las entradas en los cuadros de la TTCN.GR se definen mediante producciones de sintaxis que no incluyen ningún tipo de palabras clave de TTCN.MP. Estas producciones son comunes a la TTCN.GR y la TTCN.MP.

En el anexo A se especifican las producciones de sintaxis de TTCN.MP. Como ayuda para clarificar la descripción en TTCN.GR muchas de las producciones de sintaxis que son comunes a la TTCN.MP y la TTCN.GR están incluidas en el texto de esta Recomendación con la denominación: DEFINICIÓN DE SINTAXIS. Para facilitar la comprensión aparecerán algunas producciones en varios lugares del texto.

Se ha previsto que las producciones de sintaxis incluidas en el texto sean copias idénticas de las producciones correspondientes del anexo A, pero en caso de discordancia tendrá prioridad el anexo A.

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, exepto las diferencias indicadas en el § 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 y las semánticas estáticas descritas en el texto y descritas en el anexo A se observará lo siguiente:

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

c) las restricciones de semántica estática 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

6.1 Las ATS que cumplan esta Recomendación deberán satisfacer los requisitos de la TTCN.GR o la TTCN.MP.

Nota – Para la explicación del uso del término «cumplimiento» (*compliance*) en las Recomendaciones de la serie X.290, véase el § 10 de la Recomendación X.290.

6.2 Una ATS que cumpla los requisitos de la TTCN.GR satisfará los requisitos de sintaxis de la TTCN.GR establecidos en los § 8 a 15 y A.4.

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

6.4 Una ATS que cumpla esta Recomendación deberá satisfacer los requisitos de semántica estática establecidos en los § 7 a 15 y tener una semántica operacional conforme con la definición de semántica operacional del anexo B, de forma que sean semánticamente válidas.

6.5 Una ATS normalizada que cumpla esta Recomendación deberá exigir que cualquier realización de sucesión de pruebas que pretenda ser conforme con la ATS normalizada deberá:

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

b) cumplir con la Recomendación X.293.

Nota – 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 Recomendación X.294 deberá registrar un error de caso de prueba ejecutable o de caso de prueba abstracta según donde haya sido producido el error.

6.6 Con relación a toda ATS normalizada que haya alcanzado la categoría de proyecto de Recomendación antes de o durante 1991 o que se haya aprobado como Recomendación del CCITT en el periodo de estudios 1989-1992 se podrá enunciar que cumple con esta Recomendación, pero que utiliza alguna de las características de la TTCN descritas en el proyecto de Norma Internacional, pero que han cambiado entre el proyecto y la Norma Internacional. Tal sucesión de pruebas hará referencia a esta Recomendación X.292 y contendrá una descripción de las diferencias entre las características de la TTCN que utiliza y las especificadas en esta Recomendación.

7 Convenios

7.1 *Introducción*

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

7.2 *Metanotación sintáctica*

En el cuadro 1/X.292 se define la metanotación utilizada para especificar la forma ampliada de gramática BNF para la TTCN (denominada de aquí en adelante BNF):

CUADRO 1/X.292

Metanotación sintáctica en TTCN.MP

::=	Definición
|	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}")"

Para los textos utilizados en los formularios de cuadro se emplearán los siguientes convenios:

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

b) El texto en cursiva (*como éste*) no aparecerá exactamente palabra por palabra en una sucesión 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*

Se define la TTCN.GR utilizando dos tipos de cuadro (o tabla):

a) cuadros de objetos con TTCN únicos (véase el § 7.3.2), utilizados para definir, declarar o describir un solo objeto con TTCN tal como una declaración de PDU o un comportamiento dinámico de caso de prueba;

b) cuadros de objetos con TTCN múltiples (véase el § 7.3.3), utilizados para definir cierto número de objetos con TTCN del mismo tipo en un solo cuadro, tales como las definiciones de tipo simple o las variables de caso de prueba.

7.3.2 *Cuadros de objetos con TTCN únicos*

En la figura 1/X.292 se muestra la disposición general de un cuadro para un objeto con TTCN único:

El encabezamiento del cuadro contiene información general sobre el objeto definido en el cuadro. El primer elemento es el encabezamiento denominado *Nombre del objeto* y contiene un identificador del objeto. El último elemento, denominado *Comentarios* contiene una descripción informal del objeto. Este elemento puede omitirse.

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 puede omitirse esta columna.

El pie del cuadro contiene un elemento, denominado *Comentarios detallados*. Puede utilizarse este pie para los mismos fines que la columna comentarios del cuerpo del cuadro. El especificador de la sucesión 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 puede omitirse el pie del cuadro.

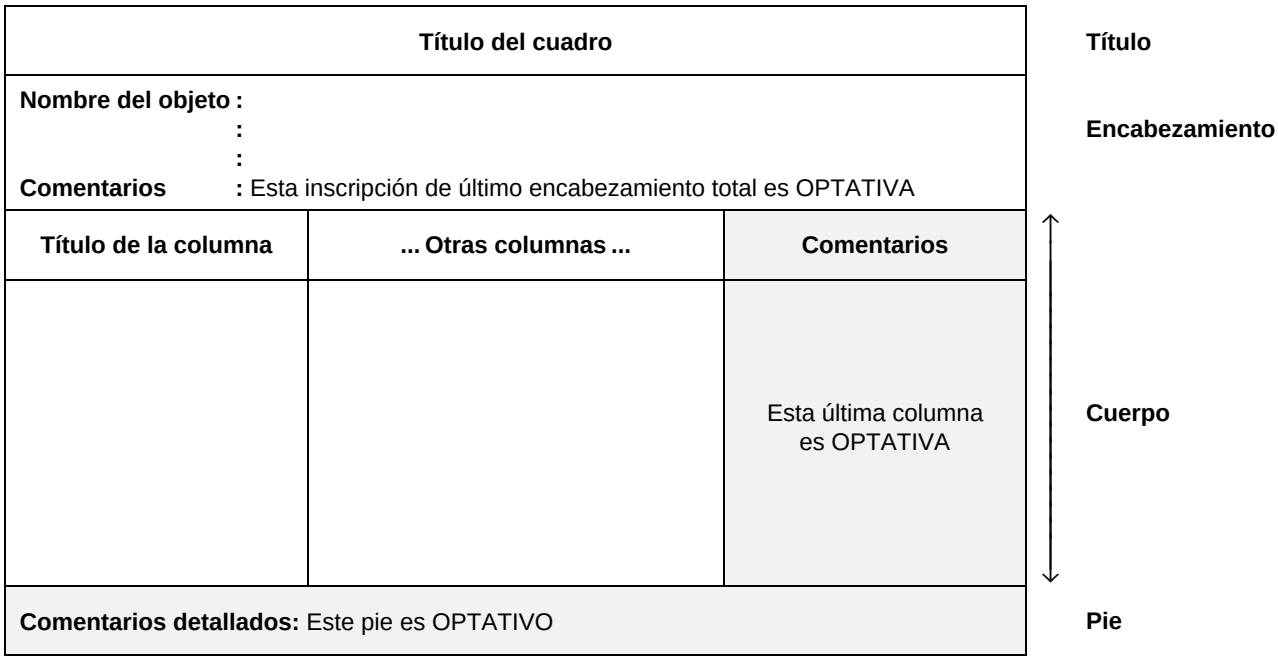


FIGURA 1/X.292

Disposición generalizada de un cuadro de declaración único

7.3.3 *Cuadros de objetos con TTCN múltiples*

En la figura 2/X.292 se muestra la disposición general de un cuadro para objetos TTCN múltiples:

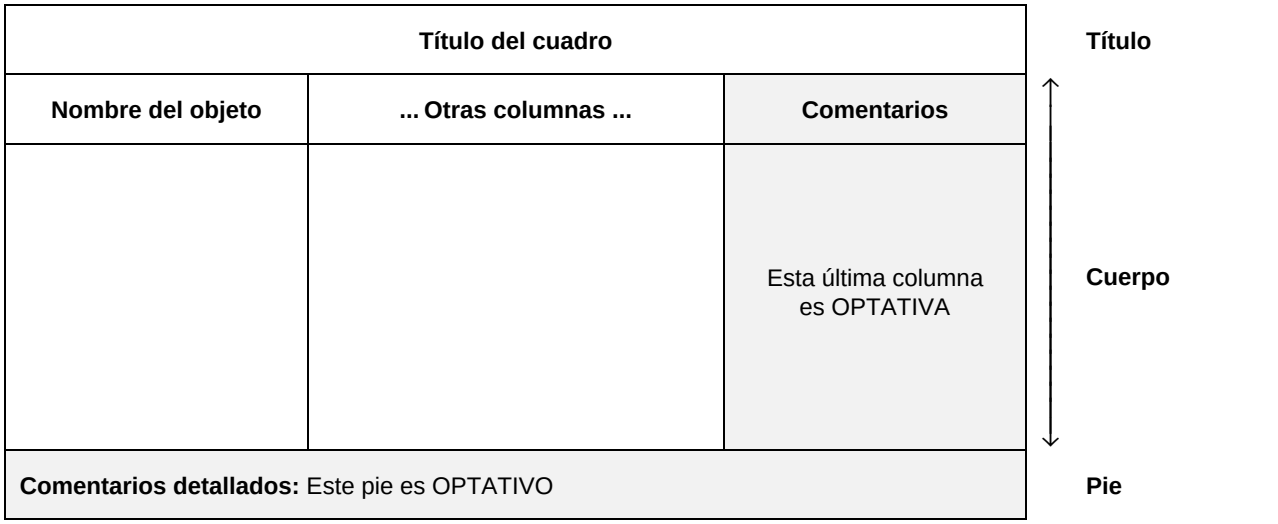


FIGURA 2/X.292

Disposición generalizada de un cuadro de declaración múltiple

Este tipo de cuadro carece de encabezamiento. 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 sucesión de pruebas. El pie del cuadro es idéntico al pie del cuadro de tipo único.

7.3.4 *Cuadros compactos alternativos*

En algunos casos, se permite la representación de varios cuadros de objetos con TTCN únicos en un formato compacto alternativo para ahorrar espacio. Es decir, pueden representarse varios cuadros de objetos con TTCN únicos en forma de un solo cuadro compacto. Los únicos cuadros que pueden presentarse bajo este formato son:

– constricciones de ASP (tabular y de 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 C se definen los formatos de estos formularios compactos alternativos.

7.3.5 *Especificación de formularios*

En esta Recomendación se especifican 32 tipos de cuadro (o tablas) con TTCN.GR y se proporciona una visión gráfica de los formularios correspondientes. Estos formularios son conformes con la disposición generalizada de los § 7.3.2 y 7.3.3. Cuando en un formulario una columna está sombreada, se señala con ello que tal columna es optativa.

7.4 *Texto libre y texto libre limitado*

Algunas inscripciones de los cuadros permiten la utilización de texto libre, es decir caracteres de cualquiera de los conjuntos de caracteres definidos en la norma ISO 10646. 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 aquéllos 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 «*/» que abren y cierran las cadenas de texto libre limitado (bounded free text) en la TTCN.MP, no aparecerán en la TTCN.GR, es decir, 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 Estructura de la sucesión de pruebas con TTCN

8.1 *Introducción*

La TTCN permite estructurar jerárquicamente una sucesión de pruebas, de conformidad con el § 8.1 de la Recomendación X.290. Los componentes de esta estructura son:

a) grupos de pruebas;

b) casos de prueba;

c) pasos de prueba.

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

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

8.2 *Referencias de grupos de pruebas*

La TTCN soporta una estructura de denominación que muestra una agrupación conceptual de los casos de prueba. Los grupos de pruebas pueden estar anidados (*nested*). Los casos de prueba pueden, asimismo, ser autónomos (*stand alone*) (véase el § 8, figura 9 de la Recomendación X.290). Las referencias de grupo de pruebas definen la estructura de la sucesión de pruebas. Las referencias de grupo de pruebas deberán tener la siguiente sintaxis:

Definición de sintaxis

235 TestGroupReference ::= [SuiteIdentifier "/"] {TestGroupIdentifier "/"}

Ejemplo 3 – Referencia de grupos de transporte: TRANSPORT/CLASS0/CONN_ESTAB/

8.3 *Referencias de grupo de pasos de prueba*

8.3.1 Los pasos de prueba pueden ser explícitamente identificados con TTCN y utilizados 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 pueden especificarse:

– lógicamente 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 dentro de una descripción de comportamiento del caso de prueba, en cuyo caso es implícito. De manera alternativa, un prólogo puede estar especificado explícitamente con su propia descripción de comportamiento. Si tal 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, deberá estar especificado en la biblioteca de pasos de prueba.

8.3.2 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 de 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 sucesión de pruebas. Las referencias de grupo de pasos de prueba tendrán la siguiente sintaxis:

Definición de sintaxis

248 TestStepGroupReference ::= [SuiteIdentifier "/"] {TestStepGroupIdentifier "/"}

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

8.4 *Referencias de grupo de valores por defecto*

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

Una referencia al 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 sucesión de pruebas. Las referencias a grupos de valores por defecto se atendrán a la siguiente sintaxis:

Definición de sintaxis

258 DefaultGroupReference ::= [SuiteIdentifier "/"] {DefaultGroupIdentifier "/"}

Ejemplo 5 – Referencia al grupo de valores por defecto de transporte:
TRANSPORT/DEFAULT_LIBRARY/CLASS0/

8.5 *Componentes de una sucesión de pruebas con TTCN*

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

a) Visión general de la sucesión (véase el § 9), que contiene la información necesaria para la presentación general y comprensión de la sucesión de pruebas, tal como las referencias de pruebas y una descripción de su finalidad global.

b) Parte declaraciones (véase el § 10), que contiene las definiciones o declaraciones de todos los componentes que constituyen la sucesión de pruebas (por ejemplo PCO, temporizadores, ASP, PDU, así como sus parámetros o campos).

c) Parte constricciones (véanse los § 11, 12, 13), que contiene las declaraciones de valores de las ASP, PDU y sus parámetros utilizados en la parte dinámica. Las constricciones deberán especificarse utilizando:

   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.

d) Parte dinámica (véase el § 14), 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 contenga las descripciones de comportamiento dinámico del paso de prueba (si existen);

   3) una biblioteca que contenga las descripciones de comportamiento dinámico de los valores por defecto (si existen).

9 Visión general de la sucesión de pruebas

9.1 *Introducción*

La finalidad de la sección de visión general de la sucesión de pruebas de la ATS es proporcionar la información necesaria para la presentación general y comprensión de la sucesión de pruebas. Dicha información comprende:

a) estructura de la sucesión de pruebas (véase el § 9.2);

b) índice de caso de prueba (véase el § 9.3);

c) índice de paso de prueba (véase el § 9.4);

d) índice de valores por defecto (véase el § 9.5).

9.2 *Estructura de la sucesión de pruebas*

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

Esta sección incluirá, al menos, la siguiente información:

a) Nombre de la sucesión de pruebas.

b) Referencias a las Recomendaciones de base pertinentes.

c) Referencia al formulario de PICS.

d) Referencia al formulario de PIXIT parcial (véase el § 15 de la Recomendación X.291).

e) Una indicación del método o métodos de prueba al que se aplica la sucesión 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 sucesión de pruebas, por ejemplo cómo se ha derivado. Tal información debería incluirse en forma de comentario.

g) Una lista de los grupos de pruebas de la sucesión de pruebas (si existen), en cuyo caso se facilitará para cada grupo la siguiente información:

1) referencia del grupo de pruebas, en la que el primer identificador deberá ser el nombre de la sucesión y cada identificador sucesivo representa un orden conceptual ulterior de la sucesión de pruebas. Los grupos de pruebas deberán enumerarse en el orden en que sus casos de prueba correspondientes aparecen en la ATS. Además, deberán ordenarse de forma que cada grupo de un solo grupo siga inmediatamente a ese grupo. Deberá hacerse una lista con todos los grupos de pruebas de la sucesión de pruebas.

2) Un identificador de expresión de selección optativo, que hace referencia a una inscripción 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 pruebas. Si, para un grupo, se proporciona un identificador de expresión de selección y la expresión de selección referenciada toma el valor FALSE (falso) no se seleccionará para su ejecución ningún caso de prueba de ese grupo. Si la expresión toma el valor TRUE (verdadero), se seleccionarán los casos de prueba de ese grupo para su ejecución, dependiendo de 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 de grupo de prueba, que es una declaración informal del objetivo del grupo de pruebas.

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 de grupo de prueba en el cuadro de estructura de la sucesión 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 indicado en el siguiente formulario:

Estructura de la sucesión de pruebas			
Nombre de la sucesión : *SuiteIdentifier* **Ref. de norma :** *Free Text* **Ref. de PICS :** *Free Text* **Ref. de PIXIT :** *FreeText* **Método(s) de prueba :** *FreeText* **Comentarios :** *[FreeText]*			
Referencia de grupo de pruebas	**Referencia de selección**	**Objetivo del grupo de pruebas**	**Número de página**
. . . . *TestGroupReference*	. . . . *[SelectExpr-Identifier]*	. . . . *FreeText*	. . . . *Number* . . . .
Comentarios detallados: *[FreeText] (texto libre)*			

Formulario 1 – Estructura de sucesión de pruebas

Definición de sintaxis

```
3    SuiteIdentifier ::= Identifier
235  TestGroupReference ::= [SuiteIdentifier "/"] {TestGroupIdentifier "/"}
83   SelectExprIdentifier ::= Identifier
```

9.3 *Índice de caso de prueba*

El índice de caso 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 pruebas), que define el lugar en que reside el caso de prueba dentro de la estructura de grupo de la sucesión de pruebas. Si falta la referencia al grupo para un caso de prueba, se supone que dicho caso de prueba reside en el mismo grupo de pruebas que el caso de prueba anterior en el índice. Los grupos de pruebas se pondrán en lista en el mismo orden en el que figuran en la ATS. Para el primer caso de prueba de cada grupo deberá proporcionarse una referencia explícita a un grupo de pruebas. Se proporcionará, asimismo, una referencia explícita al grupo de pruebas para cada caso de prueba que siga inmediatamente al último caso de prueba del grupo; esto es necesario si un grupo de pruebas contiene grupos de pruebas y casos de prueba.

b) Nombre del caso de prueba, que deberá ser 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 inscripción en el cuadro de definiciones de expresión de selección de casos de prueba utilizado para determinar si podría seleccionarse 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, dependiendo de la evaluación de la expresión de selección para los grupos de pruebas 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 es posiblemente una forma abreviada de la finalidad de la pruebas.

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 índices 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 indicado en el formulario siguiente:

Índice de caso de prueba				
Referencia de grupo de pruebas	**Id. de caso de prueba**	**Referencia de selección**	**Descripción**	**Número de página**
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
[TestGroupReference]	*TestCase Identifier*	*[SelectExpr-Identifier]*	*FreeText*	*Number*
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
Comentarios detallados: *[FreeText] (texto libre)*				

Formulario 2 – Índice de caso de prueba

Definición de sintaxis

```
235  TestGroupReference ::= [SuiteIdentifier "/"] {TestGroupIdentifier "/"}
233  TestCaseIdentifier ::= Identifier
83   SelectExprIdentifier ::= Identifier
```

9.4 *Índice de paso de prueba*

El índice de paso 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 deberá proporcionarse 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 es posiblemente una forma abreviada del objetivo de 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 indicado en el formulario siguiente:

Índice de paso de prueba			
Referencia de grupo de paso de prueba	**Id. paso de prueba**	**Descripción**	**Número de página**
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
[TestStepGroupReference]	*TestStep Identifier*	*FreeText*	*Number*
.		.	.
.	.	.	.
.	.	.	.
.	.	.	.
Comentarios detallados: *[FreeText] (texto libre)*			

Formulario 3 – Índice de paso de prueba

Definición de sintaxis

248 TestStepGroupReference ::= [SuiteIdentifier "/"] {TestStepGroupIdentifier "/"}
246 TestStepIdentifier ::= Identifier

9.5 *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, deberá proporcionarse la siguiente información:

a) Una referencia de 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 al grupo para un 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 deberá proporcionarse una referencia explícita al grupo de valores por defecto. Se proporcionará, asimismo, 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 será 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 es posiblemente 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 indicado en el formulario siguiente:

Índice de valores por defecto			
Referencia de grupo de valores por defecto	**Id. del valor por defecto**	**Descripción**	**Número de página**
. . . . *[DefaultGroupReference]*	. . . . *Default Identifier*	. . . . *FreeText*	. . . . *Number* . . . . .
Comentarios detallados: *[FreeText] (texto libre)*			

Formulario 4 – Índice de valores por defecto

Definición de sintaxis

258 DefaultGroupReference ::= [SuiteIdentifier "/"] {DefaultGroupIdentifier "/"}
257 DefaultIdentifier ::= Identifier

10 Parte declaraciones

10.1 *Introducción*

La finalidad de la parte declaraciones de la ATS es definir y declarar todos los componentes utilizados en la sucesión de pruebas. En la parte declaraciones deberán declararse los componentes de una ATS referenciada desde la parte general, desde la parte constricciones y desde la parte dinámica que a continuación se indican. Tales componentes son:

a) *Definiciones:*

   1) tipos de sucesiones de pruebas (véase 10.2.3);

   2) operaciones de sucesiones de pruebas (véase 10.3.4).

b) *Parametrización y selección de casos de prueba:*

   1) parámetros de sucesiones de pruebas (véase 10.4);

   2) expresiones de selección de caso de prueba (véase 10.5).

c) *Declaraciones/definiciones:*

   1) constantes de sucesión de pruebas (véase 10.6);

   2) variables de sucesión de pruebas (véase 10.7.1);

   3) variables de casos de prueba (véase 10.7.3);

   4) PCO (véase 10.8);

   5) temporizadores (véase 10.9);

   6) tipos de ASP (véase 10.10);

   7) tipos de PDU (véase 10.11);

   8) alias (véase 10.15).

10.2 *Tipos de TTCN*

10.2.1 *Introducción*

La TTCN soporta cierto número de tipos y mecanismos predefinidos que permiten la definición de tipos de sucesiones de pruebas específicas. Estos tipos pueden utilizarse en toda la sucesión de pruebas y referenciarse cuando se declaran parámetros de sucesión de pruebas, constantes de sucesión de pruebas, variables de sucesión de pruebas, parámetros de ASP, campos de PDU, etc.

10.2.2 *Tipos de TTCN predefinidos*

Se han predefinido diversos tipos utilizados habitualmente, para su empleo en TTCN. Todos los tipos definidos en ASN.1 y en este punto pueden ser referenciados aun cuando no aparezcan en una definición de tipo en una sucesión de pruebas. Los demás tipos utilizados en la sucesión de pruebas deberán declararse en las definiciones de tipo de sucesión de pruebas, definiciones de ASP o en definiciones de PDU y referenciarse por nombre.

Se considera que los tipos predefinidos de la TTCN que siguen son similares a sus contrapartidas en ASN.1:

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

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

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

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

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

   Los valores de tipo BITSTRING se designarán mediante un número arbitrario (que puede ser cero) de ceros y unos, precedidos por un ' simple y seguidos por el par de caracteres 'B;

Ejemplo 6 – ’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 (que puede ser cero) de dígitos hexadecimales:

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

precedidos por un ' simple y seguidos por el par de caracteres 'H. Se utiliza cada dígito HEX, para representar el valor de un semiocteto empleando la notación hexadecimal:

Ejemplo 7 – '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 se corresponde con una secuencia ordenada de 8 bits).

Los valores de tipo OCTETSTRING se indicarán mediante un número par arbitrario (que puede ser cero) de dígitos HEX:

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

precedidos de un ' simple y seguidos del par de caracteres ' O. Se utiliza cada dígito HEX para designar el valor de un semiocteto empleando la notación hexadecimal;

Ejemplo 8 – 'FF96'O

f) **Tipos predefinidos CharacterString** (cadena de caracteres): Tipos cuyos valores distinguidos son cero, uno o más caracteres del mismo conjunto de caracteres. Pueden utilizarse los tipos de cadena de caracteres del cuadro 2/X.292, definidos en el § 31 de la Recomendación X.208.

Los valores de los tipos CharacterString se designarán mediante un número arbitrario (que puede ser cero) de caracteres de un conjunto de caracteres referenciado por el tipo CharacterString precedidos y seguidos por el carácter doble comilla ("). Si el tipo CharacterString incluye el carácter doble comilla, este carácter se representará por un par de comillas dobles en la denotación de cualquier valor.

CUADRO 2/X.292

Tipos CharacterString predefinidos

NumericString
PrintableString
TeletexString (es decir T.61 String)
VideotexString
VisibleString (es decir ISO646 String)
IA5String
GraphicString
GeneralString

10.2.3 *Definiciones de tipos de sucesiones de pruebas*

10.2.3.1 *Introducción*

Las definiciones de tipos que hayan de utilizarse como tipos para objetos de datos y subtipos de ASP, PDU estructuradas, etc., podrán introducirse empleando un formato tabular y/o ASN.1. Cuando se haga referencia a los tipos en definiciones de tipos de sucesiones de pruebas, tales referencias no deberán ser recursivas (ni directa ni indirectamente).

10.2.3.2 *Definiciones de tipos simple utilizando 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 adoptará 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 de tipo INTEGER. El nuevo tipo comprende los valores con inclusión del límite inferior y del límite superior especificados en la gama. Para especificar una gama infinita, puede utilizarse 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 tipo cadena de sucesiones de pruebas. El valor o valores de la longitud se interpretarán de conformidad con el cuadro 4/X.292. Para el límite (cota) superior solo se utilizarán literales INTEGER no negativos o la palabra clave INFINITY.

Esta información se proporcionará con el formato indicado en el formulario siguiente:

Definiciones de tipos simples		
Nombre del tipo	**Definición del tipo**	**Comentarios**
. . *SimpleTypeIdentifier* . . .	. . *Type&Restriction* . .	. . *[FreeText]* . . .
Comentarios detallados: *[FreeText] (texto libre)*		

Formulario 5 – Definiciones de tipos simples

Definición de sintaxis

```
27  SimpleTypeIdentifier ::= Identifier
29  Type&Restriction ::= Type [Restriction]
331 Type ::= PredefinedType | Reference Type
332 PredefinedType ::= INTEGER | BOOLEAN | BITSTRING | HEXSTRING | OCTETSTRING |
        CharacterString
333 CharacterString ::= NumericString | PrintableString | TelexString | VideotexString |
        VisibleString | IA5String | GraphicString | GeneralString
334 ReferenceType ::= TS_TypeIdentifier | ASP_Identifier | PDU_Identifier
335 TS_TypeIdentifier ::= SimpleTypeIdentifier | StructIdentifier | ASN1_TypeIdentifier
30  Restriction ::= LengthRestriction | IntegerRange | SimpleValueList
31  LengthRestriction ::= SingleTypeLength | RangeTypeLength
32  SingleTypeLength ::= "["Number "]"
33  RangeTypeLength ::= "[" LowerTypeBound To UpperTypebound "]"
34  IntegerRange ::= "(" LowerTypeBound To UpperTypeBound ")"
35  LowerTypeBound ::= [Minus] Number | Minus INFINITY
36  UpperTypeBound ::= [Minus] Number | INFINITY
37  To ::= TO | ".."
38  SimpleValueList ::= "(" [Minus] LiteralValue {Comma [Minus] LiteralValue} ")"
```

Donde se utilice una gama en una definición de tipo, ya sea como gama de valores o como gama de longitudes (para cadenas), deberá indicarse con el menor de los dos valores a su 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 utilice una lista de valores, éstos deberán ser del tipo de base y constituir un subconjunto verdadero de los valores definidos por el tipo de base. Cuando se emplee una restricción de longitud, el conjunto de valores de tipo definidos por esta restricción deberá ser un subconjunto verdadero de los valores definidos por el tipo de base.

Ejemplo 9 – Definiciones de tipos de sucesiones de pruebas simples

Definiciones de tipos simples		
Nombre del tipo	**Definición del tipo**	**Comentarios**
Transport_classes	INTEGER(0, 1, 2, 3, 4)	Clases que pueden utilizarse 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
String10to20	IA5String [10 .. 20]	Cadena, long. min. 10 caracteres; long. máx. 20 caracteres

10.2.3.3 *Definiciones de tipos estructurados mediante cuadros*

Los tipos estructurados pueden definirse 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) Su nombre, que deberá ser el nombre completo apropiado como figura en la Recomendación sobre protocolos pertinentes; si se emplea una abreviatura, deberá figurar a continuación el nombre completo entre paréntesis.

b) Una lista de los elementos asociados al tipo estructurado, en la que para cada elemento se facilitará la siguiente información:

   1) Su nombre, que deberá ser el nombre completo, como figura en la Recomendación sobre protocolos apropiadas; si se emplea una abreviatura deberá figurar a continuación el nombre completo entre paréntesis.

   2) Su 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); puede utilizarse la restricción de longitud del elemento optativa a fin de proporcionar las longitudes máxima y mínima de un elemento de tipo cadena (véase el § 10.12).

Se considera que los elementos de definiciones de tipo estructurado son optativos, es decir, que en instancias de estos tipos pueden no estar presentes elementos completos.

Esta información se proporcionará con el formato indicado en el formulario siguiente:

Definiciones de tipos estructurados		
Nombre del tipo : *StructId&FullId* **Comentarios :** *[FreeText]*		
Nombre del elemento	**Definición del tipo**	**Comentarios**
. . *ElemId&FullId* . .	. . *Type&Attributes* . .	. . *[FreeText]* . .
Comentarios detallados: *[Free Text] (texto libre)*		

Formulario 6 – Definiciones de tipos estructurados

Definición de sintaxis

42 StructId&FullId ::= StructIdentifier [FullIdentifier]
44 StructIdentifier ::= Identifier
48 ElemId&FullId ::= ElemIdentifier [FullIdentifier]
49 ElemIdentifier ::= Identifier
43 FullIdentifier ::= "(" BoundedFreeText ")"
154 Type&Attributes ::= (Type [LengthAttribute]) | **PDU**
331 Type ::= PredefinedType | ReferenceType
332 PredefinedType ::= **INTEGER** | **BOOLEAN** | **BITSTRING** | **HEXSTRING** | **OCTETSTRING** | CharacterString
333 CharacterString ::= **NumericString** | **PrintableString** | **TeletexString** | **VideotexString** | **VisibleString** | **IA5String** | **GraphicString** | **GeneralString**
334 Reference Type ::= TS_TypeIdentifier | ASP_Identifier | PDU_Identifier
335 TS_TypeIdentifer ::= Simple TypeIdentifier | StructIdentifier | ASN1_TypeIdentifier
155 LengthAttribute ::= SingleLength | RangeLength
156 SingleLength ::= "[" Bound "]"
157 Bound ::= Number | TS_ParIdentifier | TS_ConstIdentifier
158 RangeLength ::= "[" LowerBound To UpperBound "]"
159 LowerBound ::= Bound
37 To ::= **TO** | ".."
160 UpperBound ::= Bound | **INFINITY**

10.2.3.4 *Definiciones de tipos de sucesiones de pruebas mediante la ASN.1*

Pueden especificarse los tipos de sucesiones de pruebas utilizando la ASN.1. Esto puede lograrse mediante una definición ASN.1 con la sintaxis ASN.1 definida en la Recomendación X.208. Para cada tipo ASN.1 se facilitará la siguiente información:

a) Su nombre, cuando proceda, deberá emplearse el nombre completo, como figura en la Recomendación sobre protocolos pertinente; si se emplea una abreviatura deberá figurar a continuación el nombre completo entre paréntesis.

b) Definición del tipo ASN.1, que se atendrá a la sintaxis definida en la Recomendación X.208. Dentro de esa definición no podrá utilizarse el símbolo de guión (-) para los identificadores. En su lugar deberá emplearse el símbolo de 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 otra partes de la sucesión de pruebas.

Las definiciones de tipo ASN.1 utilizadas dentro de la TTCN no harán uso de referencias de tipo externas, como las definidas en la Recomendación X.208, pudiendo hacerse comentarios dentro del cuerpo del cuadro. La columna de comentarios no estará presente en este cuadro.

Esta información se proporcionará como se indica en el formulario siguiente:

Definición de tipo ASN.1
Nombre del tipo : *ASN1_TypeId&FullId* **Comentarios :** *[FreeText]*
Definición del tipo
ASN1_Type&LocalTypes
Comentarios detallados: *[FreeText] (texto libre)*

Formulario 7 – Definición de tipo ASN.1

Definición de sintaxis

```
54  ASN1_TypeId&FullId ::= ASN1_TypeIdentifier [FullIdentifier]
55  ASN1_TypeIdentifier ::= Identifier
43  FullIdentifier ::= "(" BoundedFree Text ")"
57  ASN1_Type&LocalTypes ::= ASN1_Type {ASN1_LocalType}
58  ASN1_Type ::= Type
    /* REFERENCE – Donde Type es un no terminal definido en la Recomendación X.208 */
59  ASN1_LocalType ::= Typeassignment
    /* REFERENCE – Donde Typeassignment es un no terminal definido en la Recomendación X.208 */
```

Ejemplo 10 – Definición de tipo de sucesión de pruebas ASN.1

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

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

10.2.3.5 *Definiciones de tipo ASN.1 por referencia*

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

a) Su nombre, que podrá utilizarse en toda la sucesión de pruebas. El nombre deberá especificarse sin un FullIdentifier.

b) La referencia de tipo, que deberá atenerse a las normas de identificador establecidas en la Recomendación X.208.

c) Identificador de módulo, consistente en una referencia de módulo, que deberá atenerse a las reglas de identificador establecidas en la Recomendación X.208, y un ObjectIdentifier optativo; el módulo deberá ser exclusivo dentro del dominio de interés.

Esta información se proporcionará como se indica en el formulario siguiente:

Definiciones de tipo ASN.1 por referencia			
Nombre del tipo	**Referencia del tipo**	**Identificador del módulo**	**Comentarios**
. . *ASN1_TypeId&FullId* . .	. . *TypeReference* . .	. . *ModuleIdentifier* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*			

Formulario 8 – Definiciones de tipo ASN.1 por referencia

Definición de sintaxis

```
54  ASN1_TypeId&FullId ::= ASN1_TypeIdentifier [FullIdentifier]
55  ASN1_TypeIdentifier ::= Identifier
43  FullIdentifier ::= "(" BoundedFree Text ")"
63  TypeReference ::= typereference
    /* REFERENCE – Donde typereference se define en el § 8.2 de la Recomendación X.208 */
65  ModuleIdentifier ::= ModuleIdentifier
    /* REFERENCE – Donde ModuleIdentifier es un no terminal definido en la Recomendación X.208 */
```

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 Recomendación X.208, pueden contener guiones. Para poder utilizar las definiciones importadas en TTCN es necesario sustituir los guiones de los identificadores importados por caracteres de subrayado. Esto se efectúa en el proceso de importación.

Ejemplo 11 – 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
```

puede ser importada a TTCN con:

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

La anterior definición de referencia de Type_1 es equivalente a la siguiente definición:

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

10.3 *Operadores de TTCN y operaciones de TTCN*

10.3.1 *Introducción*

La TTCN soporta cierto número de operadores, operaciones y mecanismos predefinidos, que permiten la definición de operaciones de sucesiones de pruebas. Pueden utilizarse tales operadores y operaciones en cualesquiera descripciones de comportamiento dinámico y constricciones.

10.3.2 *Operadores de TTCN*

10.3.2.1 *Introducción*

Hay tres categorías de operadores predefinidos:

a) aritméticos;

b) relacionales;

c) booleanos.

En el cuadro 3/X.292 se indica la precedencia de estos operadores. Pueden utilizarse paréntesis para agrupar operandos en expresiones. Una expresión que figura entre paréntesis tiene la máxima precedencia para evaluación.

En cualquier fila del cuadro 3/X.292, 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 operadores

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

Definición de sintaxis

```
321  AddOp ::= "+" | "–" | OR
322  MultiplyOp ::= "*" | "/" | MOD | AND
323  UnaryOp ::= "+" | "–" | NOT
324  RelOp ::= "=" | "<" | ">" | "<>" | ">=" | "<="
```

10.3.2.2 *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 podrán utilizarse 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 entre 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 entre el segundo.

10.3.2.3 *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 deberán ser compatibles. Los demás operadores relacionales deberán tener 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 CharacterString podrán contener los caracteres comodín (*wildcard*) AnyOrNone (*) y AnyOne (?). En estos casos se efectúa la comparación según las reglas de concordancia de patrones (*patterns*) definidas en el § 11.6.5.

10.3.2.4 *Operadores booleanos predefinidos*

Los operadores booleanos predefinidos son:

NOT, **AND**, **OR**

Representan las operaciones de negación "Y" lógico y "O" lógico. Sus operandos deberán ser de tipo BOOLEAN (TTCN o ASN.1 o predefinidos). El tipo del 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; 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.

10.3.3 *Operaciones predefinidas*

10.3.3.1 *Introducción*

Las operaciones predefinidas son de dos categorías:

a) operaciones de conversión;

b) otras operaciones.

Las operaciones predefinidas pueden utilizarse en cualquier sucesión de pruebas. No requieren una definición explícita mediante una tabla de definición de operación de sucesión 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 deberán estar acotadas.

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

OPERATION_NAME (FORMAL_PARAMETER_LIST) ⇒ RESULT_TYPE

10.3.3.2 *Operaciones de conversión predefinidas*

10.3.3.2.1 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.

10.3.3.2.2 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.

10.3.3.2.3 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.

10.3.3.2.4 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 el *intvalue* (valor entero) es negativo o si el HEXSTRING resultante contiene más dígitos HEX que los especificados en el segundo parámetro.

10.3.3.2.5 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 el *intvalue* es negativo o si el BITSTRING resultante contiene más bits que los especificados en el segundo parámetro.

10.3.3.3 *Otras operaciones predefinidas*

La TTCN define también las siguientes operaciones predefinidas:

a) IS_PRESENT;

b) NUMBER_OF_ELEMENTS;

c) IS_CHOSEN;

d) LENGTH_OF.

10.3.3.3.1 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 10.3.4.

Ejemplo 12 – 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 operación llamará
IS_PRESENT(received_PDU.field_1)
tomar el valor TRUE si está presente field_1 de la instancia real de received_PDU.

10.3.3.3.2 NUMBER_OF_ELEMENTS(DataObjectReference) ⇒ INTEGER

La operación devuelve el número real de elementos de un objeto de datos que sea del tipo ASN.1 SEQUENCE OF o SET OF. La operación no podrá aplicarse a objetos de datos o campos de objetos de datos diferentes del tipo ASN.1 SEQUENCE OF o SET OF. El argumento de la operación tendrá el formato definido en el § 10.3.4.

Ejemplo 13 – Utilización de NUMBER_OF_ELEMENTS

si received_PDU es de tipo ASN.1

```
SEQUENCE    {    field_1 INTEGER OPTIONAL,
                 field_2 SEQUENCE OF INTEGER

                }
```

entonces la operación llamará

NUMBER_OF_ELEMENTS(received_PDU. field_2)
devuelve el número de elementos de SEQUENCE OF INTEGER dentro del objeto de datos real received_PDU.

10.3.3.3.3 IS_CHOSEN(DataObjectReference) ⇒ BOOLEAN

La operación devuelve el valor BOOLEAN TRUE si, y sólo si, la referencia del 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 podrá aplicarse 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 el § 10.3.4.

Ejemplo 14 – Utilización de IS_CHOSEN:

si received_PDU es del tipo ASN.1

```
CHOICE      {    p1 PDU_type1,
                 p2 PDU_type2,
                 p3 PDU_type3

                }
```

entonces la operación llamará
IS_CHOSEN(received_PDU.p2)

devuelve TRUE si la instancia real de received_PDU transporta una PDU del tipo PDU_type2.

10.3.3.3.4 LENGTH_OF(DataObjectReference) ⇒ INTEGER

La operación devuelve la longitud real de un objeto de datos que es del tipo BITSTRING, HEXSTRING, OCTETSTRING o CharacterString. En el cuadro 4/X.292 se definen las unidades de longitud de cada tipo cadena. El argumento de la operación tendrá el formato definido en el § 10.3.4.

La operación no podrá aplicarse a objetos de datos o campos de objetos de datos diferentes de BITSTRING, HEXSTRING, OCTETSTRING o CharacterString.

Ejemplo 15 – Utilización de LENGTH_OF

Si S = '010'B entonces LENGTH_OF (S) devuelve 3

Si S = 'F3'H entonces LENGTH_OF (S) devuelve 2

Si S = 'F2'O entonces LENGTH_OF (S) devuelve 1

Si S = "EXAMPLE" entonces LENGTH_OF (S) devuelve 7

10.3.4 *Definiciones de operaciones de sucesiones de pruebas*

El especificador de las ATS puede establecer operaciones específicas de una sucesión 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, esta es una lista de los nombres y tipos de los parámetros formales. Cada nombre de parámetro deberá ir seguido del carácter «dos puntos» (:), figurando a continuación el nombre del tipo de parámetro.

Cuando se utilice más de un parámetro del mismo tipo deberán especificarse los parámetros mediante una sublista de parámetros. Cuando se emplee la sublista de parámetros, los nombres de los parámetros deberán estar separados por comas. El parámetro final de la lista deberá ir seguido por «dos puntos» (:), figurando a continuación el nombre del tipo de parámetro.

Cuando se utilicen más de un par parámetro y tipo (o par lista de parámetros y tipo) los pares deberán ir separados entre sí mediante el signo de punto y coma.

Como tipos de parámetros formales solamente podrán utilizarse tipos predefinidos y tipos de datos tal y como están definidos en las definiciones de tipos de sucesión de pruebas, definiciones de tipo de ASP o definiciones de tipo de PDU. No podrán utilizarse los tipos de PCO como tipos de parámetros formales.

Ejemplo 16 – Listas de parámetros

Los métodos que siguen son formas equivalentes de especificación de una lista de parámetros utilizando 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) Descripción de la operación, que consistirá en una explicación de la operación, seguida al menos de un ejemplo que muestre una invocación y su resultado correspondiente. La operación deberá comenzar indicando el nombre de la operación, seguido de una lista entre paréntesis que contenga los nombres de los parámetros de la operación. Esto constituye una invocación «patrón» de la operación.

Esta información se proporcionará con el formato indicado en el formulario siguiente:

Definición de la operación de sucesión de pruebas
Nombre de la operación : *TS_OpId&ParList* **Tipo de resultado :** *Type* **Comentarios :** *[FreeText]*
Descripción
. . *FreeText* . .
Comentarios detallados: *[FreeText] (texto libre)*

Formulario 9 – Definición de la operación de sucesión de pruebas

Definición de sintaxis

```
69  TS_OpId&ParList ::= TS_OpIdentifier [FormalParList]
331 Type ::= PredefinedType | ReferenceType
332 PredefinedType ::= INTEGER | BOOLEAN | BITSTRING | HEXSTRING | OCTETSTRING |
    CharacterString
333 CharacterString ::= NumericString | PrintableString | TeletexString | VideotexString |
    VisibleString | IA5String | GraphicString | GeneralString
334 ReferenceType ::= TS_TypeIdentifier | ASP_Identifier | PDU_Identifier
335 TS_TypeIdentifier ::= SimpleTypeIdentifier | StructIdentifier | ASN1_TypeIdentifier
```

Una operación puede compararse con una función en un lenguaje de programación ordinario. Sin embargo, los parámetros de la operación no deberán alterarse como consecuencia de cualquier llamada de la operación y tampoco deberá haber efectos secundarios (es decir, no habrá modificaciones de ninguna sucesión de pruebas o variable de caso de prueba).

Cuando se invoque una operación de sucesión de pruebas

a) el número de parámetros reales será el mismo que el 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 aparezcan en la lista de parámetros deberán estar acotadas.

Ejemplo 17 – Definición de la operación SUBSTR:

Definición de la operación de sucesión de pruebas
Nombre de la operación : SUBSTR (source:IA5String; start_index, length:INTEGER) **Tipo de resultado :** IA5String **Comentarios :**
Descripción
SUBSTR(source, start_index, length) es la cadena de longitud *length* que comienza en el índice *start_index* de la cadena origen *source* Por ejemplo: SUBSTR("abcde",3,2) = "cd" SUBSTR("abcde",1,3) = "abc"

10.4 *Declaraciones de parámetros de sucesión 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 sucesión de pruebas. Se denomina a estas constantes parámetros de sucesión de pruebas y se utilizan como base para la selección de casos de prueba y parametrización de casos de prueba.

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

a) Su nombre.

b) Su tipo que deberá ser un tipo predefinido, un tipo ASN.1 un tipo de sucesión de pruebas o un tipo PDU.

c) Referencia de inscripción PICS/PIXIT, que es una referencia a una inscripción de formulario PICS/PIXIT individual que identificará con claridad dónde puede encontrarse el valor que ha de utilizarse para este parámetro de sucesión de pruebas.

Esta información se proporcionará con el formato indicado en el formulario siguiente:

Declaraciones de parámetro de sucesión de pruebas			
Nombre del parámetro	**Tipo**	**Ref. de PICS / PIXIT**	**Comentarios**
. . *TS_ParIdentifier* . .	. . *Type* . .	. . *FreeText* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*			

Formulario 10 – Declaraciones de parámetro de sucesión de pruebas

Definición de sintaxis

77 TS_ParIdentifier ::= Identifier
331 Type ::= PredefinedType | ReferenceType
332 PredefinedType ::= **INTEGER** | **BOOLEAN** | **BITSTRING** | **HEXSTRING** | **OCTETSTRING** | CharacterString
333 CharacterString ::= **NumericString** | **PrintableString** | **TeletexString** | **VideotexString** | **VisibleString** | **IA5String** | **GraphicString** | **GeneralString**
334 ReferenceType ::= TS_TypeIdentifier | ASP_Identifier | PDU_Identifier
335 TS_TypeIdentifier ::= SimpleTypeIdentifier | StructIdentifier | ASN1_TypeIdentifier

Ejemplo 18 – Declaración de parámetros de sucesión de pruebas:

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

10.5 *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 Recomendación X.291.

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

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

Deberá facilitarse la siguiente información relativa a cada expresión de selección de caso de prueba:

a) Su nombre.

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

Esta información se proporcionará con el formato indicado en el formulario siguiente:

Definiciones de expresión de selección de caso de prueba		
Nombre de la expresión	**Expresión de selección**	**Comentarios**
. . *SelectExprIdentifier* . .	. . *SelectionExpression* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*		

Formulario 11 – Definiciones de expresión de selección de caso de prueba

Definición de sintaxis

```
83  SelectExprIdentifier ::= Identifier
85  SelectionExpression ::= Expression
```

10.6 *Declaraciones de constantes de sucesiones 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 sucesión de pruebas.

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

a) Su nombre.

b) Su tipo, que deberá ser un tipo predefinido, un tipo ASN.1, un tipo de sucesión de pruebas o un tipo de PDU.

c) Su valor, no pudiendo contener los términos de la expresión de valor variables de sucesión 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 indicado en el formulario siguiente:

Declaraciones de constantes de sucesión de pruebas			
Nombre de la constante	**Tipo**	**Valor**	**Comentarios**
. . *TS_ConstIdentifier* . .	. . *Type* . .	. . *Declaration Value* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*			

Formulario 12 – Declaraciones de constantes de sucesión de pruebas

Definición de sintaxis

90 TS_ConstIdentifier ::= Identifier
331 Type ::= PredefinedType | ReferenceType
332 PredefinedType ::= **INTEGER** | **BOOLEAN** | **BITSTRING** | **HEXSTRING** | **OCTETSTRING** | CharacterString
333 CharacterString ::= **NumericString** | **PrintableString** | **TeletexString** | **VideotexString** | **VisibleString** | **IA5String** | **GraphicString** | **GeneralString**
334 ReferenceType ::= TS_TypeIdentifier | ASP_Identifier | PDU_Identifier
335 TS_TypeIdentifier ::= SimpleTypeIdentifier | StructIdentifier | ASN1_TypeIdentifier
93 DeclarationValue ::= Expression

Ejemplo 19 – Declaración de constantes de sucesión de pruebas

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

10.7 *Variables de TTCN*

10.7.1 *Declaraciones de variables de sucesiones de pruebas*

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

Se utiliza una variable de sucesión de pruebas siempre que sea necesario transferir información de un caso de pruebas a otro.

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

a) Su nombre.

b) Su tipo, que debará ser un tipo predefinido, un tipo ASN.1, un tipo de sucesión de pruebas o un tipo de PDU.

c) Su valor inicial (si existe), utilizándose la columna de valor inicial cuando se desee asignar un valor inicial a una variable de sucesión de pruebas en su punto de declaración; los términos de la expresión de valor no podrán contener variables de sucesión 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 indicado en el formulario siguiente:

Declaraciones de variables de sucesión de pruebas			
Nombre de la variable	**Tipo**	**Valor**	**Comentarios**
. . *TS_VarIdentifier* . .	. . *Type* . .	. . *[DeclarationValue]* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*			

Formulario 13 – Declaraciones de variables de sucesión de pruebas

Definición de sintaxis

97 TS_VarIdentifier ::= Identifier
331 Type ::= PredefinedType | ReferenceType
332 PredefinedType ::= **INTEGER** | **BOOLEAN** | **BITSTRING** | **HEXSTRING** | **OCTETSTRING** | CharacterString
333 CharacterString ::= **NumericString** | **PrintableString** | **TeletexString** | **VideotexString** | **VisibleString** | **IA5String** | **GraphicString** | **GeneralString**
334 ReferenceType ::= TS_TypeIdentifier | ASP_Identifier | PDU_Identifier
335 TS_TypeIdentifier ::= SimpleTypeIdentifier | StructIdentifier | ASN1_TypeIdentifier
93 DeclarationValue ::= Expression

Es posible que cada caso de prueba particular se ejecute con independencia de otros en la sucesión 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 20 – Declaración de las variables de sucesión de prueba:

Declaraciones de variables de sucesión 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.

10.7.2 *Vinculación de variables de casos de prueba*

Inicialmente, las variables de sucesiones de pruebas son no vinculadas (*unbound*). Dichas variables pueden devenir vinculadas (o ser revinculadas) en los siguientes contextos:

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

b) cuando la variable de sucesión de pruebas aparezca en el lado izquierdo de un enunciado de asignación (véase 14.10.4).

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

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

10.7.3 *Declaraciones de variables de casos de prueba*

Una sucesión de pruebas puede utilizar un conjunto de variables declaradas globalmente para esta sucesión de pruebas, pero cuyo ámbito (*scope*) se ha definido de manera que sea local al caso de prueba. Estas variables se denominan variables de caso de pruebas.

Para cada declaración de variable se facilitará la siguiente información:

a) Su nombre.

b) Su tipo, que deberá ser un tipo predefinido, un tipo ASN.1, un tipo de sucesión de pruebas o un tipo de PDU.

c) Su valor inicial (si existe), utilizándose 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 sucesión 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 indicado en el formulario siguiente:

Declaraciones de variables de casos de pruebas			
Nombre de la variable	**Tipo**	**Valor**	**Comentarios**
. . *TS_VarIdentifier* . .	. . *Type* . .	. . *[DeclarationValue]* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*			

Formulario 14 – Declaraciones de variables de casos de pruebas

Nota – Cuando se utilicen variables de casos de prueba como variables locales en un paso de prueba, deben tomarse precauciones, a fin de evitar la aparición de contradicciones en la utilización con otros pasos de prueba o variables de casos de prueba. Un especificador de sucesión de pruebas puede evitar tales problemas mediante la adopción de un convenio de denominación, que haga que todas esas variables resulten denominadas de forma unívoca dentro de una sucesión de pruebas.

Definición de sintaxis

103 TC_VarIdentifier ::= Identifier
331 Type ::= PredefinedType | ReferenceType
332 PredefinedType ::= **INTEGER** | **BOOLEAN** | **BITSTRING** | **HEXSTRING** | **OCTETSTRING** | CharacterString
333 CharacterString ::= **NumericString** | **PrintableString** | **TeletexString** | **VideotexString** | **VisibleString** | **IA5String** | **GraphicString** | **GeneralString**
334 ReferenceType ::= TS_TypeIdentifier | ASP_Identifier | PDU_Identifier
335 TS_TypeIdentifier ::= SimpleTypeIdentifier | StructIdentifier | ASN1_TypeIdentifier
93 DeclarationValue ::= Expression

10.7.4 *Vinculación de variables de casos de prueba*

Inicialmente las variables de casos de prueba son no-vinculadas. Podrán devenir vinculadas (o ser revinculadas), 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 de asignación (véase 14.10.4).

Una vez que una variable de caso de prueba haya sido vinculada a un valor, dicha variable mantendrá ese valor hasta que sea vinculada 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 revinculadas a su valor inicial, si se ha especificado; en caso contrario deviene no-vinculada.

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

10.8 *Declaraciones de PCO*

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

El número de PCO debe ser el definido en el § 7.5 de la Recomendación X.290 y el § 12.6 de la Recomendación X.291, para el método o métodos de prueba indicados en el cuadro de estructura de sucesiones de pruebas.

Los enunciados de comportamiento en TTCN especificados para ejecución en el PCO del UT no impondrán requisitos adicionales a los especificados por la Recomendación X.291.

En la TTCN, el modelo de PCO se basa en colas del tipo «primero en entrar, primero en salir» (FIFO):

– 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) 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 – El modelo de cola es tan solo un modelo abstracto y no se pretende que implique una realización específica.

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

a) Su nombre, utilizado en las descripciones de comportamiento para especificar dónde se producen los eventos particulares.

b) Su tipo, utilizado para identificar la frontera del servicio en la que está ubicado el PCO.

c) Su cometido, que es una explicación del tipo de probador situado en el PCO. El identificador predefinido **UT** indica que el PCO es un probador superior y **LT** especifica un PCO de probador inferior.

Esta información se proporcionará con el formato indicado en el formulario siguiente:

Declaraciones de PCO			
Nombre del PCO	**Tipo de PCO**	**Cometido**	**Comentarios**
. . *PCO_Identifier* . .	. . *PCO_TypeIdentifier* . .	. . *PCO_Role* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*			

Formulario 15 – Declaraciones de PCO

Definición de sintaxis

```
109  PCO_Identifier ::= Identifier
111  PCO_TypeIdentifier ::= Identifier
113  PCO_Role ::= UT | LT
```

Ejemplo 21 – 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 SAP (puntos de acceso al servicio) que constituyen ese PCO

– se hallan 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 – 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).

10.9 *Declaraciones de temporizador*

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

a) El nombre del temporizador.

b) La duración del temporizador optativo, siendo la duración por defecto del temporizador una expresión que podrá omitirse si no puede establecerse el valor antes de la ejecución de la sucesión de pruebas. Los términos de la expresión de valor no podrán contener variables de sucesiones de pruebas ni variables de casos de prueba; la duración del temporizador tomará a un valor INTEGER positivo sin signo.

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).

Las unidades de tiempo las determina el diseñador de la sucesión de pruebas y están fijadas en el momento de la especificación. Dentro de la misma sucesión de pruebas, temporizadores diferentes pueden utilizar unidades diferentes. Si existe una inscripción PICS o PIXIT, la declaración del temporizador especificará las mismas unidades que figuran en la inscripción PICS/PIXIT.

Esta información se proporcionará con el formato indicado en el formulario siguiente:

Declaraciones de temporizador			
Nombre del temporizador	**Duración**	**Unidad**	**Comentarios**
. . *TimerIdentifier* . .	. . *[DeclarationValue]* . .	. . *TimeUnit* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*			

Formulario 16 – Declaraciones de temporizador

Definición de sintaxis

```
117  TimerIdentifier ::= Identifier
93   DeclarationValue ::= Expression
120  TimeUnit ::= ps | ns | μs | ms | s | min
```

Ejemplo 22 – 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 sucesión de pruebas.

10.10 *Definiciones de tipos de ASP*

10.10.1 *Introducción*

La finalidad de esta parte de la sucesión 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.

10.10.2 *Definiciones de tipos de ASP utilizando cuadros*

Para cada ASP se facilitará la siguiente información:

a) Su nombre, deberá utilizarse el nombre completo como figura en las Recomendaciones sobre protocolo apropiadas. Si se emplea una abreviatura, deberá seguir el nombre completo entre paréntesis.

b) El tipo de PCO asociado con la ASP, que será uno de los tipos de PCO utilizados en el formulario de declaración de PCO. Si dentro de una sucesión 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) Su nombre, que podrá ser:

      – El nombre completo que figura en la Recomendación sobre protocolos apropiada. Si se emplea una abreviatura, deberá seguir el nombre completo entre paréntesis o

      – El símbolo macro (<–), para indicar que la inscripción 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 deberá utilizarse 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, incluyendo el especificado como tipo de sucesión 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 el § 10.12. Los valores de longitud se interpretarán de conformidad con el cuadro 4/X.292. Los límites se especificarán en términos de literales INTEGER no negativos, parámetros de sucesión de pruebas, constantes de sucesiones de pruebas o la palabra clave INFINITY.

No habrá contradicción entre las especificaciones de longitud definidas para el tipo de parámetro de ASP en las definiciones de tipo de sucesión 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 sucesión de pruebas.

Puede utilizarse la palabra clave INFINITY, como valor de límite superior, para indicar que la longitud no tiene límite superior.

Nota – Normalemente 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 indicado en el formulario siguiente:

Definición de tipo de ASP		
Nombre de la ASP : *ASP_ Id&FullId* **Tipo de PCO :** *[PCO_TypeIdentifier]* **Comentarios :** *[FreeText]*		
Nombre del parámetro	**Tipo de parámetro**	**Comentarios**
. . *ASP_ParIdOrMacro* . .	. . *Type&Attributes* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*		

Formulario 17 – Definición de tipo de ASP

Definición de sintaxis

```
127  ASP_Id&FullId ::= ASP_Identifier [FullIdentifier]
128  ASP_Identifier ::= Identifier
43   FullIdentifier ::= "(" BoundedFreeText ")"
111  PCO_TypeIdentifier ::= Identifier
132  ASP_ParIdOrMacro ::= ASP_ParId&FullId | MacroSymbol
133  ASP_ParId&FullId ::= ASP_ParIdentifier [FullIdentifier]
134  ASP_ParIdentifier ::= Identifier
150  MacroSymbol ::= "<–"
154  Type&Attributes ::= (Type [LengthAttribute]) | PDU
331  Type ::= PredefinedType | ReferenceType
332  PredefinedType ::= INTEGER | BOOLEAN | BITSTRING | HEXSTRING | OCTETSTRING |
     CharacterString
333  CharacterString ::= NumericString | PrintableString | TeletexString | VideotexString |
     VisibleString | IA5String | GraphicString | GeneralString
334  ReferenceType ::= TS_TypeIdentifier | ASP_Identifier | PDU_Identifier
335  TS_TypeIdentifier ::= SimpleTypeIdentifier | StructIdentifier | ASN1_TypeIdentifier
155  LengthAttribute ::= SingleLength | RangeLength
156  SingleLength ::= "[" Bound "]"
157  Bound ::= Number | TS_ParIdentifier | TS_ConstIdentifier
158  RangeLength ::= "[" LowerBound To UpperBound "]"
159  LowerBound ::= Bound
37   To ::= TO | ".."
160  UpperBound ::= Bound | INFINITY
```

Ejemplo 23 – Primitiva de servicio abstracta, petición T_CONEXIÓN

En la figura que sigue, se muestra un ejemplo del servicio de transporte (véase la Recomendación X.214). Podría ser parte del conjunto de ASP utilizadas para describir el comportamiento de un UT abstracto en una sucesión de pruebas DS para el transporte de clase 0. CDA, CGA y QOS son tipos de sucesiones de pruebas.

Definición de tipo 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		

10.10.3 *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 inscripción 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 deberá utilizarse 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.

10.10.4 *Definiciones de tipos de ASP utilizando ASN.1*

Donde sea más apropiado, pueden especificarse las ASP empleando la ASN.1. Esto se logrará mediante una definición ASN.1 que utilice la sintaxis ASN.1 definida en la Recomendación X.208. Para cada ASP en ASN.1 se proporcionará la siguiente información:

a) Su nombre, deberá utilizarse el nombre completo como figura en la Recomendación sobre protocolos apropiadas. Si se emplea una abreviatura deberá seguir el nombre completo entre paréntesis.

b) El tipo de PCO asociado con la ASP, que deberá ser uno de los tipos de PCO utilizados en el formulario de declaración de PCO. Si dentro de una sucesión 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 Recomendación X.208. En los identificadores internos a esa definición no deberá utilizarse 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 sucesión de pruebas.

Pueden utilizarse comentarios en ASN.1 en el cuerpo del cuadro. En el cuadro no estará presente la columna de comentarios.

Esta información se proporcionará con el formato indicado en el formulario siguiente:

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

Formulario 18 – Definición de tipo de ASP en ASN.1

Definición de sintaxis

```
127  ASP_Id&FullId ::= ASP_Identifier [FullIdentifier]
128  ASP_Identifier ::= Identifier
43   FullIdentifier ::= "(" BoundedFreeText ")"
111  PCO_TypeIdentifier ::= Identifier
57   ASN1_Type&LocalTypes ::= ASN1_Type {ASN1_LocalType}
58   ASN1_Type ::= Type
     /* REFERENCE – Donde Type es un no terminal definido en la Recomendación X.208 */
59   ASN1_LocalType ::= Typeassignment
     /* REFERENCE – Donde Typeassignment es un no terminal definido en la Recomendación X.208 */
```

10.10.5 *Definiciones de tipos de ASP en ASN.1 por referencia*

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

a) Su nombre, que deberá utilizarse a lo largo de toda la sucesión de pruebas.

b) El tipo de PCO asociado con la ASP, que deberá ser uno de los tipos de PCO utilizados en el formulario de declaración de PCO. Si dentro de una sucesión 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 de tipo, que seguirá las reglas de identificador establecidas en la Recomendación X.208.

d) El identificador del módulo, que consiste en una referencia de módulo que seguirá las reglas de identificador establecidas en la Recomendación X.208 y un ObjectIdentifier optativo.

La información se proporcionará con el formato indicado en el formulario siguiente:

Definiciones de tipos de ASP en ASN.1 por referencia				
Nombre de la ASP	**Tipo de PCO**	**Referencia de tipo**	**Identificador de módulo**	**Comentarios**
. . *ASP_Id&FullId* . .	. . *[PCO_TypeIdentifier]* . .	. . *TypeReference* . .	. . *ModuleIdentifier* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*				

Formulario 19 – Definiciones de tipos de ASP en ASN.1 por referencia

Definición de sintaxis

```
127  ASP_Id&FullId ::= ASP_Identifier [FullIdentifier]
128  ASP_Identifier ::= Identifier
43   FullIdentifier ::= "(" BoundedFreeText ")"
111  PCO_TypeIdentifier ::= Identifier
63   TypeReference ::= typereference
     /* REFERENCE – Donde typereference se define en el § 8.2 de la Recomendación X.208 */
65   ModuleIdentifier ::= ModuleIdentifier
     /* REFERENCE – Donde ModuleIdentifier es un no terminal definido en la Recomendación X.208 */
```

Las referencias de tipos de identificadores en ASN.1 y las referencias de valor pueden contener guiones. Para de poder utilizar definiciones importadas en la TTCN es preciso cambiar los guiones por caracteres de subrayado (véase el § A.4.2.1)

10.11 *Definiciones de tipos de PDU*

10.11.1 *Introducción*

La finalidad de esta parte de la sucesión 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 Recomendación pertinente sobre protocolo 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.

10.11.2 *Definición de tipos de PDU utilizando cuadros (o tablas)*

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

a) Su nombre, se utilizará el nombre completo como figura en las Recomendación sobre protocolos apropiada. Si se emplea una abreviatura deberá seguir el nombre completo entre paréntesis.

b) El tipo de PCO asociado con la PDU, que deberá ser uno de los tipos de PCO utilizados en las declaraciones de PCO. Si una PDU solamente se envía o se recibe insertada en las ASP, en la totalidad de la sucesión de pruebas, la especificación de las PCO es optativa. Si en una sucesión de pruebas se define solamente un PCO único, la especificación del tipo de PCO en la definición de tipo de PDU, es optativa.

c) Lista de los campos asociados con la PDU, debiendo proporcionarse la siguiente información para cada campo:

   1) Su nombre, que podrá ser:

      – el nombre completo que figura en la Recomendación sobre protocolos apropiada. Si se emplea una abreviatura, deberá seguir el nombre completo entre paréntesis o

      – el símbolo macro (<–), para indicar que la inscripción 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 deberá utilizarse 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 sucesión 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 el § 10.12. Los valores se interpretarán longitud de conformidad con el cuadro 4/X.292. Los límites se especificarán en términos de literales INTEGER no negativos, parámetros de sucesiones de pruebas, constantes de sucesiones 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 sucesiones 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 sucesión de pruebas.

Puede utilizarse la palabra clave INFINITY, como valor del límite superior, para indicar que la longitud no tiene límite superior.

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 indicado en el formulario siguiente:

Definición de tipo PDU		
Nombre de la PDU : *PDU_ Id&Fullld* **Tipo de PCO :** *[PCO_TypeIdentifier]* **Comentarios :** *[FreeText]*		
Nombre del campo	**Tipo de campo**	**Comentarios**
. . *PDU_FieldIdOrMacro* . .	. . *Type&Attributes* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*		

Formulario 20 – Definición de tipo PDU

Definición de sintaxis

144 PDU_Id&FullId ::= PDU_Identifier [FullIdentifier]
145 PDU_Identifier ::= Identifier
43 FullIdentifier ::= "(" BoundedFreeText ")"
111 PCO_TypeIdentifier ::= Identifier
149 PDU_FieldIdOrMacro ::= PDU_Fieldld&FullId | MacroSymbol
151 PDU_Fieldld&FullId ::= PDU_FieldIdentifier [FullIdentifier]
152 PDU_FieldIdentifier ::= Identifier
150 MacroSymbol ::= "<–"
154 Type&Attributes ::= (Type [LengthAttribute]) | **PDU**
331 Type ::= PredefinedType | ReferenceType
332 PredefinedType ::= **INTEGER** | **BOOLEAN** | **BITSTRING** | **HEXSTRING** | **OCTETSTRING** | CharacterString
333 CharacterString ::= **NumericString** | **PrintableString** | **TeletexString** | **VideotexString** | **VisibleString** | **IA5String** | **GraphicString** | **GeneralString**
334 ReferenceType ::= TS_TypeIdentifier | ASP_Identifier | PDU_Identifier
335 TS_TypeIdentifier ::= SimpleTypeIdentifier | StructIdentifier | ASN1_TypeIdentifier
155 LengthAttribute ::= SingleLength | RangeLength
156 SingleLength ::= "[" Bound "]"
157 Bound ::= Number | TS_ParIdentifier | TS_ConstIdentifier
158 RangeLength ::= "[" LowerBound To UpperBound "]"
159 LowerBound ::= Bound
37 To ::= **TO** | ".."
160 UpperBound ::= Bound | **INFINITY**

Ejemplo 24 – Definición de tipo de PDU típica

Definición de tipo PDU		
Nombre de la PDU : INTC (Interrupt Confirm) **Tipo de PCO :** NSAP **Comentarios :** NSAP		
Nombre del campo	**Tipo del 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

10.11.3 *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 inscripción 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 deberá utilizarse 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.

10.11.4 *Definiciones de tipo de PDU utilizando ASN.1*

Donde sea más apropiado, pueden especificarse PDU en ASN.1. Esto se consigue mediante una definición ASN.1 que emplee la sintaxis ASN.1 definida en la Recomendación X.208. Para cada PDU de tipo ASN.1 se facilitará la siguiente información:

a) Su nombre, que deberá utilizarse completo como figura en las Recomendación sobre protocolos pertinente. Si se emplea una abreviatura deberá seguir el nombre completo entre paréntesis.

b) Tipo de PCO asociado con la PDU, que deberá ser uno de los tipos de PCO utilizados en las declaraciones de PCO. 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 sucesión 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) Definición de tipo de PDU en ASN.1, que seguirá la sintaxis definida en la Recomendación X.208. En los identificadores internos a esa definición no deberá utilizarse 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 sucesión de pruebas.

Pueden utilizarse comentarios en ASN.1 en el cuerpo del cuadro. En el cuadro no estará presente la columna de comentarios.

La información se proporcionará con el formato indicado en el formulario siguiente:

Definición de tipo PDU en ASN.1
Nombre de la PDU : *PDU_Id&FullId* **Tipo de PCO :** *[PCO_TypeIdentifier]* **Comentarios :** *[FreeText]*
Definición de tipo
. . *ASN1_Type&LocalTypes* . .
Comentarios detallados: *[FreeText] (texto libre)*

Formulario 21 – Definición de tipo PDU en ASN.1

Definición de sintaxis

144 PDU_Id&FullId ::= PDU_Identifier [FullIdentifier]
145 PDU_Identifier ::= Identifier
43 FullIdenfifier ::= "(" BoundedFreeText ")"
111 PCO_TypeIdentifier ::= Identifier
57 ASN1_Type&LocalTypes ::= ASN1_Type {ASN1_LocalType}
58 ASN1_Type ::= *Type*

/* REFERENCE – Donde Type es un no terminal definido en la Recomendación X.208 */

59 ASN1_LocalType ::= *Typeassignment*

/* REFERENCE – Donde Typeassignment es un no terminal definido en la Recomendación X.208 */

Ejemplo 25 – 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

SECUENCIA {
state_result State_result DEFAULT success,
action_result Action_Result multiple success,
protocol_id Protocol_Version,
 	
-- etc.
}

10.11.5 *Definiciones de tipo de PDU en ASN.1 por referencia*

Las PDU pueden especificarse mediante una referencia precisa a una PDU en ASN.1 definida en una Recomendación sobre la OSI o referenciando un tipo ASN.1 definido en un módulo ASN.1 adjuntado a la sucesión de pruebas. Para cada PDU se facilitará la siguiente información:

a) Su nombre, que deberá utilizarse en toda la sucesión de pruebas.

b) El tipo de PCO asociado con la PDU, que deberá ser uno de los tipos PCO utilizados en las declaraciones de PCO. Si una PDU solamente se envía o se recibe insertada en las ASP dentro de la totalidad de la sucesión de pruebas, la especificación del tipo de PCO es optativa. Si, en una sucesión 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 de tipo, que deberá seguir las reglas de identificador establecidas en la Recomendación X.208.

d) El identificador de módulo, consistente en la referencia de módulo que deberá seguir las reglas establecidas en la Recomendación X.208 para el identificador y a un ObjectIdentifier optativo.

La información se proporcionará con el formato indicado en el formulario siguiente:

Definiciones de tipo de PDU en ASN.1 por referencia				
Nombre de la PDU	**Tipo de PCO**	**Referencia de tipo**	**Identificador de módulo**	**Comentarios**
. . *PDU_Id&FullId* . .	. . *[PCO_TypeIdentifier]* . .	. . *TypeReference* . .	. . *ModuleIdentifier* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*				

Formulario 22 – Definiciones de tipo de PDU en ASN.1 por referencia

Definición de sintaxis

144 PDU_Id&FullId ::= PDU_Identifier [FullIdentifier]
145 PDU_Identifier ::= Identifier
43 FullIdentifier ::= "(" BoundedFreeText ")"
111 PCO_TypeIdentifier ::= Identifier
63 TypeReference ::= *typereference*

/* REFERENCE – Donde typereference se define en el § 8.2 de la Recomendación X.208 */

65 ModuleIdentifier ::= *ModuleIdentifier*

/* REFERENCE – Donde ModuleIdentifier es un no terminal definido en la Recomendación X.208 */

Las referencias de tipo de identificadores en ASN.1 y las referencias de valor pueden contener guiones. Para poder utilizar definiciones importadas en la TTCN, es necesario sustituir los guiones por símbolos de subrayado (véase el § A.4.2.1).

10.12 *Especificaciones de longitud de cadena*

10.12.1 La TTCN permite la especificación de restricciones de longitud a tipos de cadena (esto es, BITSTRING, HEXSTRING, OCTETSTRING y todos los tipos de CharacterString) en las siguientes circunstancias:

a) cuando se declaren tipos de sucesiones 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.

10.12.2 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, puede utilizarse 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, pueden especificarse 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 4/X.292 se especifican las unidades de longitud de los diferentes tipos de cadenas:

CUADRO 4/X.292

Unidades de longitud utilizadas en especificaciones de longitud de campo

Tipo	Unidades de longitud
BITSTRING	Bits
HEXSTRING	Dígitos Hex
OCTETSTRING	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 26 – Especificación de longitud

Supóngase las siguientes definiciones de tipo en ASN.1:

```
type 1 ::= OCTETSTRING [0 .. 25]
type 2 ::= type 1 [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:

```
type 2 ::= type 1 [15 .. 30]
```

no es válido, porque contiene valores no incluidos en type1.

10.13 *Definiciones de ASP y PDU 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 11, 12 y 13) se corresponderán con la definición del parámetro o campo. Esto significa que:

a) el valor será del tipo especificado para ese parámetro de ASP o campo de PDU y,

b) los valores satisfarán cualquier tipo de restricciones de longitud pertinentes asociadas con el tipo.

10.14 *Definiciones de ASP y PDU 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 de 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.

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 utilizando tipos estructurados o ASN.1, las reglas anteriores son aplicables a los campos de la subestructura o subestructuras recursivamente.

10.15 *Definiciones de alias*

10.15.1 *Introducción*

Para acrecentar la legibilidad de las descripciones de comportamiento en TTCN, puede utilizarse un alias que facilite la redenominación («renaming») 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 deberá facilitarse la siguiente información:

a) un identificador de alias;

b) su expansión, que, en sí mismo, es un identificador.

La información se proporcionará con el formato indicado en el formulario siguiente:

Definiciones de alias		
Nombre de alias	**Expansión**	**Comentarios**
. . *AliasIdentifier* . .	. . *Expansion* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*		

Formulario 23 – Definiciones de alias

Definición de sintaxis

```
168  AliasIdentifier ::= Identifier
170  Expansion ::= ASP_Identifier | PDU_Identifier
```

10.15.2 *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. Esto significa que un alias queda delimitado por cualquier carácter (símbolo) no permitido en un identificador de TTCN.

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) Podrá utilizarse 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 podrá utilizarse 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 27 – Definición de alias desde una sucesión de pruebas de transporte

Definiciones de alias		
Nombre de alias	**Desarrollo**	**Comentarios**
CR (Connection Request, petición de conexión)	N_DATArequest	Alias de la ASP de petición N_DATA utilizado para transportar una CR_TPDU
DR (Disconnect Request, petición de desconexión)	N_DATArequest	Alias de la ASP de petición N_DATA utilizado para transportar una DR_TPDU
CC (Connection Confirm, confirmación de conexión)	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 el BNF para las líneas de evento TTCN.

11 Parte constricciones

11.1 *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 el § 14) 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 pueden especificarse en cualquiera de las dos formas siguientes:

a) constricciones tabulares (véase el § 12);

b) constricciones en ASN.1 (véase el § 13).

11.2 *Principios generales*

En este punto 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. En cada contexto puede utilizarse 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 utilizando 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.

En las constricciones pueden utilizarse 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 deberá ser 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 literales, Parámetros de sucesiones de pruebas, constantes de sucesiones de pruebas, parámetros formales y operaciones de sucesión de pruebas.

Para facilitar el encadenamiento estático, se admite también como valor de parámetro o valor de campo una referencia de constricciones (posiblemente parametrizada).

En las constricciones, no se utilizarán variables de sucesiones de pruebas ni variables de caso de prueba, a menos que se pasen como parámetros reales. En este último caso deberán estar vinculadas a un valor y no serán modificadas por la ocurrencia de un evento SEND o RECEIVE.

En el § 11.6.2 se definen los mecanismos de concordancia.

11.3 *Parametrización de constricciones*

Las constricciones pueden ser parametrizadas. En tales casos, el nombre de la constricción deberá ir seguido de una lista de parámetros formales encerrados entre paréntesis. Los parámetros se utilizarán formales para especificar valores de parámetros de ASP o de campos de PDU en la constricción.

Cada nombre de parámetro formal deberá ir seguido del símbolo : (dos puntos) y del nombre del tipo de parámetro. Si se utiliza más de un parámetro del mismo tipo, puede especificarse el parámetro en forma sublista de parámetros. Cuando se utilice una sublista de parámetros, los nombres de parámetro irán separados por comas. El parámetro final de la sublista deberá ir 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 de punto y coma.

En la referencia a restricciones efectuada desde una descripción de comportamiento, pueden transferirse como parámetros reales de la constricción valores literales, parámetros de sucesión de pruebas, constantes de sucesión de pruebas, variables de sucesión de pruebas, variables de caso de prueba y PDU o constricciones de tipo de sucesión de pruebas. Estos parámetros no podrán ser de tipo PCO ni de tipo ASP.

11.4 *Encadenamiento de constricciones*

Las constricciones pueden encadenarse 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 de 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 de 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 D se presentan ejemplos de encadenamiento estático y dinámico).

Cuando se haga referencia a las constricciones en declaraciones de constricción, tales referencias no deberán ser recursivas (ni directa ni indirectamente).

11.5 *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 específicos 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 utilizando el símbolo Omit, para indicar que el parámetro de 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 utilizando el símbolo Omit o, simplemente, dejando fuera el parámetro de ASP o campo de PDU pertinente.

Ninguno de los mecanismos de concordancia definidos en el § 11.6.2, excepto SpecificValue, proporciona un valor para un parámetro de 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.

11.6 *Constricciones para eventos RECEIVE*

11.6.1 *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 el § 11.6.3 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 los parámetros de la ASP y/o los campos de la PDU son del tipo especificado en las definiciones de ASP y/o PDU, si el valor, el alfabeto y la longitud satisfacen cualquier limitación asociada al tipo y si los valores del parámetro de ASP y/o del campo de PDU concuerdan correctamente con los de la constricción.

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.

11.6.2 *Mecanismos de concordancia*

En el cuadro 5/X.292, 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 pueden utilizarse *en vez de* valores;

c) símbolos especiales que pueden utilizarse *dentro de* valores;

d) símbolos especiales que describen *atributos* de valores.

Algunos de los símbolos pueden utilizarse combinados como se indica en las cláusulas que siguen.

La zona sombreada del cuadro 5/X.292 indica los mecanismos que se aplican a los tipos de ASN.1 y de TTCN predefinidos.

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, pueden especificarse en las constricciones numerosas restricciones de protocolo, evitando de este modo detalles de computación no deseables en la parte comportamiento.

CUADRO 5/X.292

Mecanismos de concordancia en TTCN

	VALUE	INSTEAD OF VALUE						INSIDE VALUE					ATTRIBUTES		
TYPE	Specific Value	Complement	Omit (–)	AnyValue (?)	AnyOrOmit (*)	ValueList		Range	AnyOne (?)	SuperSet	AnyOrNone (*)	SubSet	Permutation	Length	IfPresent
BOOLEAN	•	•	•	•	•										•
INTEGER	•	•	•	•	•	•									•
ENUMERATED	•	•	•	•	•										•
BITSTRING	•	•	•	•	•				•	•				•	•
OCTETSTRING	•	•	•	•	•				•	•				•	•
HEXSTRING	•	•	•	•	•				•	•				•	•
CHARSTRINGS	•	•	•	•	•				•	•				•	•
SEQUENCE	•	•	•	•	•										•
SEQUENCE OF	•	•	•	•	•				•	•	•			•	•
SET	•	•	•	•	•										•
SET OF	•	•	•	•	•	•	•		•	•				•	•
ANY	•	•	•	•	•										•
CHOICE	•	•	•	•	•										•
OBJECT ID	•	•	•	•	•										•

11.6.3 *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 de una ASP o un campo de una PDU de constricción, concuerda con el parámetro de ASP o el campo de PDU correspondiente si, y sólo si, el parámetro de 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.

11.6.4 *Instead of value (valor en vez de)*

11.6.4.1 *Complement (complemento)*

Complement es una operación de concordancia que puede utilizarse 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 de ASP o campo de PDU en el que se utilice el mecanismo de complemento.

Definición de sintaxis

201 Complement **::=** **COMPLEMENT** ValueList

Un parámetro de ASP o campo de PDU de constricción que utilice Complement concuerda con el parámetro de ASP o el campo de PDU correspondiente si, y sólo si, el parámetro de ASP o el campo de PDU entrante no concuerda con ninguno de los valores enumerados en ValueList.

Ejemplo 28 – Constricciones utilizando Complement en vez de un valor y con una lista de valores

Tipo	*Constricción*
INTEGER	COMPLEMENT (5)
INTEGER	COMPLEMENT(1, 3, 5)

11.6.4.2 *Omit (omitir)*

Omit es un símbolo especial para concordancia que puede utilizarse con valores de todos los tipos, siempre que el parámetro de ASP o campo de PDU sea opcional.

En las constricciones en ASN.1 también es posible dejar fuera un parámetro de ASP o campo de PDU OPTIONAL, en vez de utilizar explícitamente OMIT.

En las constricciones en forma tabular, Omit deberá denotarse por un guión (–). En constricciones en ASN.1, Omit se denota por **OMIT**.

Definición de sintaxis

202 Omit ::= Dash | **OMIT**

Se utiliza un símbolo Omit en una constricción para indicar la ausencia de un parámetro de ASP o un campo de PDU opcional.

Ejemplo 29 – Constricciones utilizando Omit en vez de un valor en el nivel superior

Tipo	*Constricción*
INTEGER OPTIONAL	OMIT

11.6.4.3 *AnyValue (cualquier valor)*

AnyValue es un símbolo especial para concordancia que puede utilizarse con valores de todos los tipos. Tanto para las constricciones en ASN.1 como en forma tabular, AnyValue se denota por «?».

Definición de sintaxis

203 AnyValue ::= "?"

Un parámetro de ASP o campo de PDU de constricción que utiliza AnyValue, concuerda con el parámetro de ASP o campo de PDU entrante correspondiente si, y sólo si, el parámetro de ASP o campo de PDU toma el valor de un único elemento del tipo especificado.

Ejemplo 30 – Constricciones utilizando Value en combinación con AnyValue

Tipo	*Constricción*
SEQUENCE OF SET OF INTEGER	{ {1, 2}, ?, {1, 2, ?} }

11.6.4.4 *AnyOrOmit (cualquiera u omitir)*

AnyOrOmit es un símbolo especial para concordancia que puede utilizarse con valores de todos los tipos, siempre que se hayan declarado como opcionales el parámetro de 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 los § 11.6.4.4 y 11.6.5.2.

Definición de sintaxis

204 AnyOrOmit ::= "*"

Un parámetro de ASP o campo de PDU de constricción que utiliza AnyOrOmit concuerda con el parámetro de ASP o campo de PDU entrante correspondiente únicamente si, o bien el parámetro de ASP, o el campo de PDU entrante toma el valor de cualquier elemento del tipo especificado, o bien está ausente el parámetro de ASP o campo de PDU entrante.

Ejemplo 31 – Constricciones utilizando Value en combinación con AnyOrOmit

Tipo	*Constricción*
SEQUENCE OF { id1 SET OF INTEGER id2 SET OF INTEGER	{ id1 {2, 5}, id2 * }

11.6.4.5 *ValueList (lista de valores)*

ValueList puede utilizarse 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.

Definición de sintaxis

```
205  ValueList ::= "(" ConstraintValue&Attributes { Comma ConstraintValue&Attributes }")"
```

Un parámetro de ASP o campo de PDU de constricción que utiliza una ValueList concuerda con el parámetro de ASP o campo de PDU entrante correspondiente si, y sólo si, el valor del parámetro de 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 de ASP o campo de PDU en el que se utiliza el mecanismo ValueList.

Ejemplo 32 – Constricciones utilizando ValueList en vez de un valor específico, para tipo INTEGER

Tipo	*Constricción*
INTEGER	(2, 4, 6)

Ejemplo 33 – Constricciones utilizando ValueList en vez de un valor específico, para el tipo CHOICE

Tipo	*Constricción*
CHOICE { a INTEGER, b BOOLEAN }	(a 2, b TRUE)

11.6.4.6 *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 de constricción 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 deberá ser menor que el límite superior.

Definición de sintaxis

```
206  ValueRange ::= "(" ValRange ")"
207  ValRange ::= (LowerRangeBound To UpperRangeBound)
208  LowerRangeBound ::= ConstraintExpression | Minus INFINITY
209  UpperRangeBound ::= ConstraintExpression | INFINITY
```

Un parámetro de ASP o campo de PDU de constricción que utiliza una gama, concuerda con el parámetro de ASP o campo de PDU entrante correspondiente si, y sólo si, el valor del parámetro de ASP o campo de PDU entrante es igual a uno de los valores de la gama.

Ejemplo 34 – Constricciones utilizando Range en vez de valor

Tipo	*Constricción*
INTEGER	(1 .. 6) (–INFINITY .. 8) (12 .. INFINITY)

11.6.4.7 *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**.

Definición de sintaxis

210 SuperSet ::= **SUPERSET**"(" ConstraintValue&Attributes ")"

Un parámetro de ASP o campo de PDU de constricción que utiliza SuperSet concuerda con el parámetro de ASP o campo de PDU entrante correspondiente si, y sólo si, el parámetro de 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 de ASP o el campo de PDU en el que se utiliza el mecanismo SuperSet.

Ejemplo 35 – Constricciones utilizando SuperSet en vez de un valor específico

Tipo	*Constricción*
SET OF INTEGER	SUPERSET({1, 2, 3})

11.6.4.8 *SubSet (subconjunto)*

SubSet es una operación de concordancia que puede utilizarse solamente con valores de tipo SET OF. Solamente se utilizará SubSet con constricciones en ASN.1. SubSet se denota por **SUBSET**.

Definición de sintaxis

211 SubSet ::= **SUBSET** "(" ConstraintValue&Attributes ")"

Un parámetro de ASP o campo de PDU de constricción que utiliza SubSet concuerda con el parámetro de ASP o campo de PDU entrante correspondiente si, y sólo si, el parámetro de 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 de ASP o campo de PDU en el que se utiliza el mecanismo SuperSet.

Ejemplo 36 – Constricciones utilizando SuperSet en vez de un valor específico

Tipo	*Constricción*
SET OF INTEGER	SUBSET({2, 4, 6, 8, 10})

11.6.5 *Inside Values (valores interiores)*

11.6.5.1 *AnyOne (cualquiera)*

AnyOne es un símbolo especial para concordancia que puede utilizarse con valores de tipos cadena, SEQUENCE OF y SET OF. En las constricciones en forma tabular y en ASN.1, AnyOne se denota por «?».

Definición de sintaxis

351 AnyOne ::= "?"

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 37 – Constricciones utilizando 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.

11.6.5.2 *AnyOrNone (cualquiera o ninguno)*

AnyOrNone es un símbolo especial para concordancia que puede utilizarse 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.

Definición de sintaxis

```
352  AnyOrNone ::= "*"
```

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 38 – Constricciones utilizando AnyOrOne

Tipo	*Constricción*
IA5String	"ab*z"
SEQUENCE OF INTEGER	{1, 2, *, 10}
SEQUENCE OF IA5String	{ "ab*z", *, "abc"}

11.6.5.3 *Permutation (permutación)*

Permutation es una operación para concordancia que puede utilizarse 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**.

Definición de sintaxis

```
212  Permutation ::= PERMUTATION ValueList
```

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 de ASP o campo de PDU.

Ejemplo 39 – Constricciones utilizando Permutation

Tipo	*Constricción*
SEQUENCE OF INTEGER	{PERMUTATION (1, 2, 3), 5}

Ejemplo 40 – Constricciones utilizando 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}.

11.6.6 *Atributes of values (atributos de valores)*

11.6.6.1 *Length (longitud)*

Length es una operación para concordancia que puede utilizarse solamente como atributo de los siguientes mecanismos: Specific Value, Complement, Omit, AnyValue, AnyOrOmit, AnyOne, AnyOrNone y Permutation.

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 el § 10.12. Las unidades de longitud se interpretarán según el cuadro 4/X.292. Los límites pueden designarse mediante valores INTEGER específicos no negativos. Alternativamente, puede utilizarse 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 de ASP o campo de PDU, en las definiciones de tipo de sucesiones 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.

Definición de sintaxis

```
214  ValueLength ::= SingleValueLength | RangeValueLength
215  SingleValueLength ::= "[" ValueBound "]"
216  ValueBound ::= Number | TS_ParIdentifier | TS_ConstIdentifier | FormalParIdentifier
217  RangeValueLength ::= "[" LowerValueBound To UpperValueBound "]"
218  LowerValueBound ::= ValueBound
37   To ::= TO | ".."
219  UpperValueBound ::= ValueBound | INFINITY
```

Un parámetro de ASP o campo de PDU de constricción que utiliza Length como atributo de un símbolo, concuerda con el parámetro de ASP o campo de PDU entrante correspondiente si, y sólo si, el parámetro de 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 de 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 de ASP o campo de PDU recibidos es exactamente el valor especificado.

Ejemplo 41 – Constricciones utilizando Value en combinación con Length

Tipo	*Constricción*
IA5String	"ab*ab" [13]

11.6.6.2 *IfPresent (si presente)*

IfPresent es un símbolo especial para concordancia que puede utilizarse 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 de ASP o campo de PDU de constricción que utiliza un símbolo IfPresent como atributo de otro símbolo, concuerda con el parámetro de ASP o campo de PDU entrante correspondiente si, y sólo si, el parámetro de ASP o campo de PDU entrante concuerda con el símbolo o si está ausente el parámetro de ASP o campo de PDU entrante.

Nota – El símbolo AnyOrOmit (*) tiene exactamente el mismo significado que ? IF_PRESENT.

Ejemplo 42 – Constricciones utilizando Value en combinación con IfPresent

Tipo	*Constricción*
IA5String OPTIONAL	"abcdef" IF_PRESENT

12 Especificación de constricciones mediante cuadros (o tablas)

12.1 *Introducción*

En esta cláusula se describe la especificación de constricciones en forma tabular impuestas a tipos estructurados, ASP y PDU. Se indica cómo pueden utilizarse los cuadros de constricciones simples para especificar constricciones impuestas a ASP o PDU planas (no estructuradas) y cómo pueden especificarse constricciones estructuradas declarando constricciones impuestas a tipos estructurados, definidos en los tipos de sucesiones de pruebas.

En el anexo C se definen cuadros adicionales que permiten efectuar muchas definiciones de constricciones simples en un solo cuadro.

12.2 *Declaraciones de constricciones de tipo estructurado*

Si se define una ASP o PDU utilizando tipos estructurados ya sea como expansiones de macro o subestructuras, deberán subestructurarse análogamente las constricciones para esas ASP o PDU. Los cuadros de constricciones de estructuras son muy similares a los cuadros de constricciones de PDU. Los valores de elementos de las constricciones de estructura deberán proporcionarse con el formato indicado en el formulario siguiente:

Declaración de constricción de tipo estructurado		
Nombre de la constricción : *ConsId&ParList* **Tipo estructurado :** *StructIdentifier* **Trayecto de derivación :** *[DerivationPath]* **Comentarios :** *[FreeText]*		
Nombre del elemento	**Valor del elemento**	**Comentarios**
. . *ElemIdentifier* . .	. . *ConstraintValue&Attributes* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*		

Formulario 24 – Declaración de constricción de tipo estructurado

Definición de sintaxis

```
190  ConsId&ParList ::= ConstraintIdentifier [FormalParList]
191  ConstraintIdentifier ::= Identifier
44   StructIdentifier ::= Identifier
193  DerivationPath ::= {ConstraintIdentifier Dot}+
49   ElemIdentifier ::= Identifier
197  ConstraintValue&Attributes ::= ConstraintValue ValueAttributes
198  ConstraintValue ::= ConstraintExpression | MatchingSymbol | ConsRef
199  ConstraintExpression ::= Expression
200  MatchingSymbol ::= Complement | Omit | AnyValue | AnyOrOmit | ValueList | ValueRange |
       SuperSet | SubSet | Permutation
277  ConsRef ::= ConstraintIdentifier [ActualCrefParList]
278  ActualCrefParList ::= "(" ActualCrefPar {Comma ActualCrefPar} ")"
279  ActualCrefPar ::= Value | DataObjectIdentifier | ConsRef
213  ValueAttributes ::= [ValueLength] [IF_PRESENT]
214  ValueLength ::= SingleValueLength | RangeValueLength
215  SingleValueLength ::= "[" ValueBound "]"
216  ValueBound ::= Number | TS_ParIdentifier | TS_ConstIdentifier | FormalParIdentifier
217  RangeValueLength ::= "[" LowerValueBound To UpperValueBound "]"
218  LowerValueBound ::= ValueBound
37   To ::= TO | ".."
219  UpperValueBound ::= ValueBound | INFINITY
```

Este formulario se utiliza del mismo modo que se utiliza para las PDU el formulario de declaración de constricción de PDU (véase 12.4).

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.

12.3 *Declaraciones de constricciones de ASP*

Los valores de parámetros para constricciones de ASP se proporcionarán con el formato que se indica en el formulario siguiente:

Declaración de constricción de ASP		
Nombre de la constricción : *ConsId&ParList* **Tipo de ASP :** *ASP_Identifier* **Trayecto de derivación :** *[DerivationPath]* **Comentarios :** *[FreeText]*		
Nombre del parámetro	**Valor del parámetro**	**Comentarios**
. . *ASP_ParldOrMacro* . .	. . *ConstraintValue&Attributes* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*		

Formulario 25 – Declaración de constricción de ASP

Definición de sintaxis

```
190  ConsId&ParList ::= ConstraintIdentifier [FormalParList]
191  ConstraintIdentifier ::= Identifier
128  ASP_Identifier ::= Identifier
193  DerivationPath ::= {ConstraintIdentifier Dot}+
132  ASP_ParldOrMacro ::= ASP_Parld&FullId | MacroSymbol
133  ASP_Parld&FullId ::= ASP_ParIdentifier [FullIdentifier]
134  ASP_ParIdentifier ::= Identifier
150  MacroSymbol ::= "<–"
197  ConstraintValue&Attributes ::= ConstraintValue ValueAttributes
198  ConstraintValue ::= ConstraintExpression | MatchingSymbol | ConsRef
199  ConstraintExpression ::= Expression
200  MatchingSymbol ::= Complement | Omit | AnyValue | AnyOrOmit | ValueList | ValueRange |
     SuperSet | SubSet | Permutation
277  ConsRef ::= ConstraintIdentifier [ActualCrefParList]
278  ActualCrefParList ::= "(" ActualCrefPar {Comma ActualCrefPar} ")"
279  ActualCrefPar ::= Value | DataObjectIdentifier | ConsRef
213  ValueAttributes ::= [ValueLength] [IF_PRESENT]
214  ValueLength ::= SingleValueLength | RangeValueLength
215  SingleValueLength ::= "[" ValueBound "]"
216  ValueBound ::= Number | TS_ParIdentifier | TS_ConstIdentifier | FormalParIdentifier
217  RangeValueLength ::= "[" LowerValueBound To UpperValueBound "]"
218  LowerValueBound ::= ValueBound
37   To ::= TO | ".."
219  UpperValueBound ::= ValueBound | INFINITY
```

Este formulario se utiliza para las ASP del mismo modo que se utiliza el formulario de declaración de constricción de PDU (véase 12.4).

12.4 *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 deberá facilitarse 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 el § 12.6).

d) Un valor de constricción para cada campo, debiendo suministrarse, para cada campo, la siguiente información:

1) Su nombre:

Cada una de las inscripciones del campo en la columna de nombre del campo deberá haberse declarado 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.

La información se proporcionará con el formato indicado en el formulario siguiente:

Declaración de constricción de PDU		
Nombre de la constricción : *ConsId&ParList* **Tipo de PDU :** *PDU_Identifier* **Trayecto de derivación :** *[DerivationPath]* **Comentarios :** *[FreeText]*		
Nombre del campo	**Valor del campo**	**Comentarios**
. . *PDU_FieldIdOrMacro* . .	. . *ConstraintValue&Attributes* . .	. . *[FreeText]* . .
Comentarios detallados: *[FreeText] (texto libre)*		

Formulario 26 – Declaración de constricción de PDU

Definición de sintaxis

```
190  ConsId&ParList ::= ConstraintIdentifier [FormalParList]
191  ConstraintIdentifier ::= Identifier
145  PDU_Identifier ::= Identifier
193  DerivationPath ::= {ConstraintIdentifier Dot}+
149  PDU_FieldIdOrMacro ::= PDU_FieldId&FullId | MacroSymbol
151  PDU_FieldId&FullId ::= PDU_FieldIdentifier [FullIdentifier]
152  PDU_FieldIdentifier ::= Identifier
150  MacroSymbol ::= "<–"
197  ConstraintValue&Attributes ::= ConstraintValue ValueAttributes
198  ConstraintValue ::= ConstraintExpression | MatchingSymbol | ConsRef
199  ConstraintExpression ::= Expression
200  MatchingSymbol ::= Complement | Omit | AnyValue | AnyOrOmit | ValueList | ValueRange |
     SuperSet | SubSet | Permutation
277  ConsRef ::= ConstraintIdentifier [ActualCrefParList]
278  ActualCrefParList ::= "(" ActualCrefPar {Comma ActualCrefPar} ")"
279  ActualCrefPar ::= Value | DataObjectIdentifier | ConsRef
213  ValueAttributes ::= [ValueLength] [IF_PRESENT]
214  ValueLength ::= SingleValueLength | RangeValueLength
215  SingleValueLength ::= "[" ValueBound "]"
216  ValueBound ::= Number | TS_ParIdentifier | TS_ConstIdentifier | FormalParIdentifier
217  RangeValueLength ::= "[" LowerValueBound To UpperValueBound "]"
218  LowerValueBound ::= ValueBound
37   To ::= TO | ".."
219  UpperValueBound ::= ValueBound | INFINITY
```

Ejemplo 43 – Constricción, denominada C.1, 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"	

12.5 *Parametrización de constricciones*

Las constricciones pueden parametrizarse utilizando una lista de parámetros formales. Los parámetros reales se pasan a una constricción desde una referencia a constricciones de una descripción de comportamiento.

Ejemplo 44 – 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"	
Comentarios detallados: Una posible referencia al C2 desde un caso de prueba o un paso de prueba puede ser: C2 (0, TRUE)		

12.6 *Constricciones de base y constricciones modificadas*

Para cada definición de tipo de PDU deberá especificarse, al menos, una constricción de base. Una constricción de base especifica un conjunto de valores de base o por defecto 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 D).

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 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 inscripción del trayecto de derivación en el encabezamiento de la constricción. Esta inscripción 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:

a) Si en la constricción no están especificados un parámetro o campo y su valor correspondiente, se utilizará el valor de la constricción progenitora (es decir, se hereda el valor).

b) Si en la constricción están especificados un parámetro o campo y su valor correspondiente, el valor especificado sustituye al valor contenido en la constricción progenitora.

12.7 *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 Especificación de constricciones mediante ASN.1

13.1 *Introducción*

En este punto se describe un método de especificación de constricciones en ASN.1 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 comodines y permutaciones. Se definen mecanismos con los que sustituir u omitir partes de constricciones en ASN.1 para utilizarlas en constricciones modificadas.

13.2 *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 utilizando referencias a constricciones de tipo de sucesiones de pruebas en ASN.1, para valores de campos complejos. Los tipos de sucesiones de pruebas en ASN.1 se definen en la parte declaraciones de las ATS.

Los cuadros de constricciones de tipo en ASN.1 son muy parecidos a los cuadros de constricciones de ASP. Las declaraciones de constricciones de tipo en ASN.1 se especificarán con el formato indicado en el formulario siguiente:

Declaración de constricción de tipo en ASN.1
Nombre de la restricción : *ConsId&ParList* **Tipo estructurado :** *ASN1_TypeIdentifier* **Trayecto de derivación :** *[DerivationPath]* **Comentarios :** *[FreeText]*
Valor de la constricción
. . C*onstraintValue&AttributesOrReplace* . .
Comentarios detallados: *[FreeText] (texto libre)*

Formulario 27 – Declaración de constricción de tipo en ASN.1

Definición de sintaxis

190 ConsId&ParList ::= ConstraintIdentifier [FormalParList]
191 ConstraintIdentifier ::= Identifier
55 ASN1_TypeIdentifier ::= Identifier
193 DerivationPath ::= {ConstraintIdentifier Dot}+
223 ConstraintValue&AttributesOrReplace ::= ConstraintValue&Attributes | Replacement {Comma Replacement}
197 ConstraintValue&Attributes ::= ConstraintValue ValueAttributes
198 ConstraintValue ::= ConstraintExpression | MatchingSymbol | ConsRef
199 ConstraintExpression ::= Expression
200 MatchingSymbol ::= Complement | Omit | AnyValue | AnyOrOmit | ValueList | ValueRange | SuperSet | SubSet | Permutation
277 ConsRef ::= ConstraintIdentifier [ActualCrefParList]
278 ActualCrefParList ::= "(" ActualCrefPar {Comma ActualCrefPar} ")"
279 ActualCrefPar ::= Value | DataObjectIdentifier | ConsRef
213 ValueAttributes ::= [ValueLength] [**IF_PRESENT**]
214 ValueLength ::= SingleValueLength | RangeValueLength
215 SingleValueLength ::= "[" ValueBound "]"
216 ValueBound ::= Number | TS_ParIdentifier | TS_ConstIdentifier | FormalParIdentifier
217 RangeValueLength ::= "[" LowerValueBound To UpperValueBound "]"
218 LowerValueBound ::= ValueBound
37 To ::= **TO** | ".."
219 UpperValueBound ::= ValueBound | **INFINITY**
224 Replacement ::= (**REPLACE** ReferenceList **BY** ConstraintValue&Attributes) | (**OMIT** ReferenceList)
225 ReferenceList ::= (ArrayRef | ComponentIdentifier | ComponentPosition) {ComponentReference}

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 el § 13.4).

13.3 *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 los § 12.6 y 13.6), 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 inscripción del trayecto de derivación.

d) El valor de la constricción, conteniendo 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 pueden utilizarse todos los valores y atributos de la constricción definidos en el § 11.6.

Las declaraciones de constricciones de ASP en ASN.1 se especificarán con el formato indicado en el formulario siguiente:

Declaración de constricción de ASP en ASN.1
Nombre de la constricción : *ConsId&ParList* **Tipo de ASP :** *ASP_Identifier* **Trayecto de derivación :** *[DerivationPath]* **Comentarios :** *[FreeText]*
Valor de la constricción
. . *ConstraintValue&AttributesOrReplace* . .
Comentarios detallados: *[FreeText] (texto libre)*

Formulario 28 – Declaración de constricción de ASP en ASN.1

Definición de sintaxis

```
190  ConsId&ParList ::= ConstraintIdentifier [FormalParList]
191  ConstraintIdentifier ::= Identifier
128  ASP_Identifier ::= Identifier
193  DerivationPath ::= {ConstraintIdentifier Dot}+
223  ConstraintValue&AttributesOrReplace ::= ConstraintValue&Attributes | Replacement
     {Comma Replacement}
197  ConstraintValue&Attributes ::= ConstraintValue ValueAttributes
198  ConstraintValue ::= ConstraintExpression | MatchingSymbol | ConsRef
199  ConstraintExpression ::= Expression
200  MatchingSymbol ::= Complement | Omit | AnyValue | AnyOrOmit | ValueList | ValueRange |
     SuperSet | SubSet | Permutation
277  ConsRef ::= ConstraintIdentifier [ActualCrefParList]
278  ActualCrefParList ::= "(" ActualCrefPar {Comma ActualCrefPar} ")"
279  ActualCrefPar ::= Value | DataObjectIdentifier | ConsRef
213  ValueAttributes ::= [ValueLength] [IF_PRESENT]
214  ValueLength ::= SingleValueLength | RangeValueLength
215  SingleValueLength ::= "[" ValueBound "]"
216  ValueBound ::= Number | TS_ParIdentifier | TS_ConstIdentifier | FormalParIdentifier
217  RangeValueLength ::= "[" LowerValueBound To UpperValueBound "]"
218  LowerValueBound ::= ValueBound
37   To ::= TO | ".."
219  UpperValueBound ::= ValueBound | INFINITY
224  Replacement ::= (REPLACE ReferenceList BY ConstraintValue&Attributes) | (OMIT ReferenceList)
225  ReferenceList ::= (ArrayRef | ComponentIdentifier | ComponentPosition) {ComponentReference}
```

Este formulario se utiliza para las ASP en ASN.1 del mismo modo que se utiliza el formulario de declaracion de constricción de PDU en ASN.1 (véase el § 13.4).

13.4 *Declaraciones de constricciones de PDU en ASN.1*

Los cuadros de constricciones de PDU son muy parecidos a los cuadros de constricciones de ASP. Las declaraciones de constricciones de PDU en ASN.1 se especificarán con el formato indicado en el formulario siguiente:

Declaración de constricción de PDU en ASN.1
Nombre de la constricción : *ConsId&ParList* **Tipo de PDU :** *PDU_Identifier* **Trayecto de derivación :** *[DerivationPath]* **Comentarios :** *[FreeText]*
Valor de la constricción
. . *ConstraintValue&AttributesOrReplace* . .
Comentarios detallados: *[FreeText] (texto libre)*

Formulario 29 – Declaración de constricción de PDU en ASN.1

Definición de sintaxis

190 ConsId&ParList ::= ConstraintIdentifier [FormalParList]
191 ConstraintIdentifier ::= Identifier
145 PDU_Identifier ::= Identifier
193 DerivationPath ::= {ConstraintIdentifier Dot}+
223 ConstraintValue&AttributesOrReplace ::= ConstraintValue&Attributes | Replacement {Comma Replacement}
197 ConstraintValue&Attributes ::= ConstraintValue ValueAttributes
198 ConstraintValue ::= ConstraintExpression | MatchingSymbol | ConsRef
199 ConstraintExpression ::= Expression
200 MatchingSymbol ::= Complement | Omit | AnyValue | AnyOrOmit | ValueList | ValueRange | SuperSet | SubSet | Permutation
277 ConsRef ::= ConstraintIdentifier [ActualCrefParList]
278 ActualCrefParList ::= "(" ActualCrefPar {Comma ActualCrefPar} ")"
279 ActualCrefPar ::= Value | DataObjectIdentifier | ConsRef
213 ValueAttributes ::= [ValueLength] [**IF_PRESENT**]
214 ValueLength ::= SingleValueLength | RangeValueLength
215 SingleValueLength ::= "[" ValueBound "]"
216 ValueBound ::= Number | TS_ParIdentifier | TS_ConstIdentifier | FormalParIdentifier
217 RangeValueLength ::= "[" LowerValueBound To UpperValueBound "]"
218 LowerValueBound ::= ValueBound
37 To ::= **TO** | ".."
219 UpperValueBound ::= ValueBound | **INFINITY**
224 Replacement ::= (**REPLACE** ReferenceList **BY** ConstraintValue&Attributes) | (**OMIT** ReferenceList)
225 ReferenceList ::= (ArrayRef | ComponentIdentifier | ComponentPosition) {ComponentReference}

13.5 *Constricciones en ASN.1 parametrizadas*

Las constricciones en ASN.1 pueden ser parametrizadas (véase el § 12.5).

13.6 *Constricciones en ASN.1 modificadas*

Pueden especificarse 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 utilizando 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.

Definición de sintaxis

224 Replacement ::= (**REPLACE** ReferenceList **BY** ConstraintValue&Attributes) | (**OMIT** ReferenceList)
225 ReferenceList ::= (ArrayRef | ComponentIdentifier | ComponentPosition) {ComponentReference}

Si un campo pertenece a una estructura SEQUENCE, SET o CHOICE, puede utilizarse la posición del campo entre paréntesis para reemplazar al identificador de campo. Este método deberá utilizarse cuando en la declaración del campo no se incluya el identificador.

13.7 *Listas de parámetros formales en las constricciones en ASN.1 modificadas*

Los requisitos del § 12.7 se aplican, también, a las constricciones en ASN.1 modificadas.

13.8 *Nombres de parámetro de 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 45 – Valores de campo en una constricción de PDU en ASN.1. Supóngase la definición de tipo

Definición de tipo PDU en ASN.1
Nombre de la PDU : XY_PDU **Typo de PCO :** **Comentarios :**
Definición de tipo
SET { field_1 INTEGER OPTIONAL, field_2 BOOLEAN, field_3 INTEGER OPTIONAL, field_4 INTEGER OPTIONAL }

Una posible constricción es entonces:

Declaración de 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 Parte dinámica

14.1 *Introducción*

La parte dinámica contiene el cuerpo principal de la sucesión de pruebas, es decir las descripciones de caso de prueba, de paso de prueba y de comportamiento por defecto.

14.2 *Comportamiento dinámico de caso de prueba*

14.2.1 *Especificación del cuadro de comportamiento de caso de prueba*

14.2.1.1 El título del cuadro será «comportamiento dinámico de caso de prueba».

14.2.1.2 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 del grupo de pruebas, que contendrá el nombre completo del nivel más bajo del grupo que contiene el caso de prueba. Ese nombre completo deberá ser conforme con los requisitos del § 8.2 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 Recomendación (si existe) sobre finalidades de las prueba y estructuras de las sucesiones de pruebas pertinentes o en la parte equivalente de la Recomendación sobre sucesiones de pruebas (si existe);

d) Referencia de comportamiento por defecto, identificador (incluyendo 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 el § 14.4).

14.2.1.3 El cuerpo del cuadro deberá contener las siguientes columnas con la información correspondiente:

a) Una columna (optativa) con el número de línea (véase el § 14.2.4), la cual, de estar presente, deberá estar situada en el lado izquierdo del cuadro.

b) Una columna de etiqueta, en la que deberán colocarse etiquetas para identificar los enunciados en TTCN que permitan efectuar bifurcaciones utilizando el constructivo GOTO (véase el § 14.14).

c) Una descripción de comportamiento, que describe el comportamiento del LT y/o el UT, en términos de enunciados en TTCN y sus parámetros, utilizando la notación arborescente (véase el § 14.6).

d) Una columna de referencias de constricciones, en la que se sitúan las referencias de 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 el § 11).

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 el § 14.7).

f) Una columna de comentarios (optativa), que se utiliza para insertar comentarios que faciliten la comprensión de los enunciados en TTCN, proporcionando notas breves o referencias al texto adicional 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.

14.2.1.4 Un pie (optativo) puede contener comentarios detallados.

14.2.2 *Formulario de comportamiento dinámico de caso de prueba*

El comportamiento dinámico de caso de prueba se proporcionará con el formato indicado en el formulario siguiente:

Comportamiento dinámico de caso de prueba						
Nombre del caso de prueba : *TestCaseIdentifier* **Grupo :** *TestGroupReference* **Finalidad :** *FreeText* **Valor por defecto :** *[DefaultReference]* **Comentarios :** *[FreeText]*						
N.°	**Etiqueta**	**Descripción de comportamiento**	**Etiqueta**	**Referencia de constricciones**	**Veredicto**	**Comentarios**
1 2 . . . . . . *n*	. . *[Label]*	. . *StatementLine* . . *TreeHeader* *StatementLine* . .	. . Posición alternativa para la columna de etiqueta .	. . *[ConstraintReference]*	. . *[Verdict]*	. . *[FreeText]* . . . . . .
Comentarios detallados: *[FreeText] (texto libre)*						

Formulario 30 – Comportamiento dinámico de caso de prueba

Definición de sintaxis

```
233  TestCaseIdentifier ::= Identifier
235  TestGroupReference ::= [SuiteIdentifier "/"] {TestGroupIdentifier "/"}
238  DefaultReference ::= DefaultIdentifier [ActualParList]
299  ActualParList ::= "(" ActualPar {Comma ActualPar} ")"
300  ActualPar ::= Value | PCO_Identifier
274  Label ::= Identifier
286  StatementLine ::= (Event [Qualifier] [AssignmentList] [TimerOps]) | (Qualifier [AssignmentList]
     [TimerOps]) | (AssignmentList [TimerOps]) | TimerOps | Construct | lmplicitSend
264  TreeHeader ::= TreeIdentifier [FormalParList]
265  TreeIdentifier ::= Identifier
266  FormalParList ::= "(" FormalPar&Type {SemiColon FormalPar&Type} ")"
267  FormalPar&Type ::= FormalParIdentifier {Comma FormalParIdentifier} Colon FormalParType
268  FormalParIdentifier ::= Identifier
269  FormalParType ::= Type | PCO_TypeIdentifier | PDU
331  Type ::= PredefinedType | ReferenceType
277  ConsRef ::= ConstraintIdentifier [ActualCrefParList]
278  ActualCrefParList ::= "(" ActualCrefPar {Comma ActualCrefPar} ")"
279  ActualCrefPar ::= Value | DataObjectIdentifier | ConsRef
281  Verdict ::= Pass | Fail | Inconclusive | Result
282  Pass ::= PASS | P | "(" PASS ")" | "(" P ")"
283  Fail ::= FAIL | F | "(" FAIL ")" | "(" F ")"
284  Inconclusive ::= INCONC | I | "(" INCONC ")" | "(" I ")"
285  Result ::= Identifier
```

Se representa en líneas de trazos la posición alternativa de la columna de etiquetas.

Los encabezamientos de columna de este formulario pueden abreviarse así: **L, Cref, 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.

14.2.3 *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:

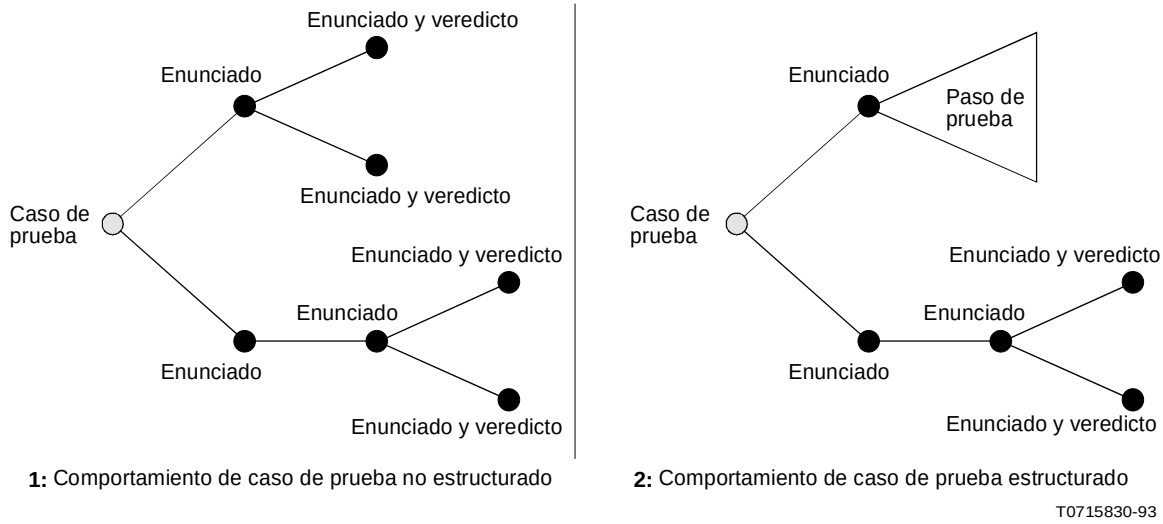


FIGURA 3/X.292

Estructura del comportamiento de caso de prueba

En TTCN se expresa esta modularización explícita utilizando pasos de prueba y el constructivo ATTACH.

14.2.4 *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 inscripción 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 – Podrán utilizarse números de línea a efectos de registro cronológico (*logging*), para registrar inequívocamente qué línea de comportamiento ha sido ejecutada.

Nota 2 – Los números de línea pueden utilizarse 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 46 – Impresión de líneas de comportamiento largas

46.1 Estilo recomendado:

N.°	Etiqueta	Descripción de comportamiento	Referencia 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 de constricción demasiado larga para su impresión en un renglón		
3		Línea de enunciado alternativa	Ref2		

46.2 Estilo alternativo:

Etiqueta	Descripción de comportamiento	Referencia 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 de constricción demasiado larga para su impresión en un renglón		
	Línea de enunciado alternativa	Ref2		

14.3 *Comportamiento dinámico de paso de prueba*

14.3.1 *Especificación del cuadro de comportamiento dinámico de los pasos de prueba*

El comportamiento dinámico de los pasos de prueba se define utilizando el mismo método que para los casos de prueba, con la excepción de que los pasos de prueba pueden ser parametrizados (véase el § 14.7). 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, el cual 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 pueden utilizarse 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 de 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 el § 8.3) 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.

14.3.2 *Formulario de comportamiento dinámico de paso de prueba*

El comportamiento dinámico del paso de prueba se proporcionará con el formato indicado en el formulario siguiente:

Comportamiento dinámico del paso de prueba						
Nombre del paso de prueba : *TestStepId&ParList* **Grupo :** *TestStepGroupReference* **Objetivo :** *FreeText* **Valor por defecto :** *[DefaultReference]* **Comentarios :** *[FreeText]*						
N.°	**Etiqueta**	**Descripción de comportamiento**	**Etiqueta**	**Referencia de constricciones**	**Veredicto**	**Comentarios**
1 2 . . . . . . *n*	. . *[Label]*	. . *StatementLine* . . *TreeHeader* *StatementLine* . .	. Posición de alternativa para la columna de Etiqueta .	. . *[ConstraintReference]*	. . *[Verdict]*	. . *[FreeText]* . . . . . .
Comentarios detallados: *[FreeText] (texto libre)*						

Formulario 31 – Comportamiento dinámico del paso de prueba

Definición de sintaxis

```
245  TestStepId&ParList ::= TestStepIdentifier [FormalParList]
246  TestStepIdentifier ::= Identifier
266  FormalParList ::= "(" FormalPar&Type {SemiColon FormalPar&Type} ")"
267  FormalPar&Type ::= FormalParIdentifier {Comma FormalParIdentifier} Colon FormalParType
268  FormalParIdentifier ::= Identifier
269  FormalParType ::= Type | PCO_TypeIdentifier | PDU
331  Type ::= PredefinedType | ReferenceType
248  TestStepGroupReference ::= [SuiteIdentifier "/"] {TestStepGroupIdentifier "/"}
238  DefaultReference ::= DefaultIdentifier [ActualParList]
299  ActualParList ::= "(" ActualPar {Comma ActualPar} ")"
300  ActualPar ::= Value | PCO_Identifier
274  Label ::= Identifier
286  StatementLine ::= (Event [Qualifier] [AssignmentList] [TimerOps]) | (Qualifier [AssignmentList] [TimerOps]) I
     (AssignmentList [TimerOps]) | TimerOps | Construct | ImplicitSend
264  TreeHeader ::= TreeIdentifier [FormalParList]
265  TreeIdentifier ::= Identifier
266  FormalParList ::= "(" FormalPar&Type {SemiColon FormalPar&Type} ")"
267  FormalPar&Type ::= FormalParIdentifier {Comma FormalParIdentifier} Colon FormalParType
268  FormalParIdentifier ::= Identifier
269  FormalParType ::= Type | PCO_TypeIdentifier | PDU
331  Type ::= PredefinedType | ReferenceType
277  ConsRef ::= ConstraintIdentifier [ActualCrefParList]
278  ActualCrefParList ::= "(" ActualCrefPar {Comma ActualCrefPar} ")"
279  ActualCrefPar ::= Value | DataObjectIdentifier | ConsRef
281  Verdict ::= Pass | Fail | Inconclusive | Result
282  Pass ::= PASS | P | "(" PASS ")" | "(" P ")"
283  Fail ::= FAIL | F | "(" FAIL ")" | "(" F ")"
284  Inconclusive ::= INCONC | I | "(" INCONC ")" | "(" I ")"
285  Result ::= Identifier
```

La posición alternativa de la columna de etiqueta se representa mediante líneas de puntos.

Los encabezamientos de columna de este formulario pueden abreviarse así: **L, Cref, V** y **C**.

14.4 *Comportamiento dinámico por defecto*

14.4.1 *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 utilizando 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 constará de un único árbol (es decir no habrá árboles locales).

c) El árbol de la descripción del comportamiento no empleará la adjunción (*attachement*) de árbol (es decir, los árboles de comportamiento por defecto no adjuntarán pasos de prueba).

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 de estos parámetros son las mismas que las correspondientes a la adjunción de árbol (véase el § 14.13).

14.4.2 *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, el cual 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 pueden utilizarse 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 de 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 deberá ajustarse a los requisitos de la referencia de grupo de valores por defecto (véase el § 8.4) 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.

14.4.3 *Formulario de comportamiento dinámico por defecto*

El comportamiento dinámico por defecto se proporcionará con el formato indicado en el formulario siguiente:

Comportamiento dinámico por defecto						
Nombre por defecto : *DefaultId&ParList* **Grupo :** *DefaultGroupReference* **Objetivo :** *FreeText* **Comentarios :** *[FreeText]*						
N.°	**Etiqueta**	**Descripción de comportamiento**	**Etiqueta**	**Referencia de constricciones**	**Veredicto**	**Comentarios**
1 2 . . . . *n*	. . *[Label]*	. . *StatementLine*	. . Posición alternativa para la columna de Etiqueta .	. . *[ConstraintReference]*	. . *[Verdict]*	. . *[FreeText]* . . . . . .
Comentarios detallados: *[FreeText] (texto libre)*						

Formulario 32 – Comportamiento dinámico por defecto

Definición de sintaxis

```
256  DefaultId&ParList ::= DefaultIdentifier [FormalParList]
257  DefaultIdentifier ::= Identifier
266  FormalParList ::= "(" FormalPar&Type {SemiColon FormalPar&Type} ")"
267  FormalPar&Type ::= FormalParIdentifier {Comma FormalParIdentifier} Colon FormalParType
268  FormalParIdentifier ::= Identifier
269  FormalParType ::= Type | PCO_TypeIdentifier | PDU
331  Type ::= PredefinedType | ReferenceType
238  DefaultReference ::= DefaultIdentifier [ActualParList]
299  ActualParList ::= "(" ActualPar {Comma ActualPar} ")"
300  ActualPar ::= Value | PCO_Identifier
274  Label ::= Identifier
286  StatementLine ::= (Event [Qualifier] [AssignmentList] [TimerOps]) | (Qualifier [AssignmentList] [TimerOps]) |
     (AssignmentList [TimerOps]) | TimerOps | Construct | ImplicitSend
277  ConsRef ::= ConstraintIdentifier [ActualCrefParList]
278  ActualCrefParList ::=  "(" ActualCrefPar {Comma ActualCrefPar} ")"
279  ActualCrefPar ::= Value | DataObjectIdentifier | ConsRef
281  Verdict ::= Pass | Fail | Inconclusive I Result
282  Pass ::= PASS | P | "(" PASS ")" | "(" P ")"
283  Fail ::= FAIL | F | "(" FAIL ")" | "(" F ")"
284  Inconclusive ::= INCONC | I | "(" INCONC ")" | "(" I ")"
285  Result ::= Identifier
```

La posición alternativa de la columna etiqueta se representa mediante líneas de puntos.

Los encabezamientos de columna de este formulario pueden abreviarse así: **L, Cref, V** y **C**.

14.5 *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 sucesión de pruebas. Se denomina árbol de comportamiento al conjunto de esas combinaciones. Cada enunciado en TTCN es un nodo del árbol de comportamiento.

14.6 *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 47 – 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 pueden producirse en ese momento. En consecuencia, este conjunto de enunciados en TTCN se definirá como *conjunto de alternativas* o, simplemente *alternativas*.

Ejemplo 48 – Enunciados en TTCN alternativos

```
CONSTRUCT_A1
STATEMENT_A2
EVENT_A3
```

Ejemplo 49 – 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 fructuosa 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 fructuosamente 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 fructuoso. Esto puede conducir a un comportamiento inalcanzable, en especial si los enunciados se codifican como alternativas siguiendo a enunciados cuya evaluación es siempre fructuosa.

REPEAT y GOTO son siempre fructuosos. Adicionalmente, SEND, IMPLICIT SEND, asignaciones y operaciones de temporizador son fructuosos 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 1.

Definición de sintaxis

271 Line ::= $Line Indentation StatementLine

Ejemplo 50 – $Línea [6] + R1_POSTAMBLE

14.7 *Nombres de árbol y lista de parámetros*

14.7.1 *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.

Definición de sintaxis

261 RootTree ::= {BehaviourLine}+
262 LocalTree ::= Header {BehaviourLine}+

14.7.2 *Árboles con parámetros*

Todos los árboles pueden ser parametrizados, 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 deberán ir 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 sucesiones 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 51 – Paso de prueba que utiliza parámetros formales:

EXAMPLE_TREE (L:TSAP; X:INTEGER; Y:INTEGER)

Ejemplo 52 – Paso de prueba que utiliza parámetros formales con una sublista:

EXAMPLE_TREE (L:TSAP; X, Y:INTEGER)

14.8 *Enunciados en TTCN*

La notación en forma de árbol permite la especificación de eventos de prueba iniciados por el LT o el UT (eventos SEND e IMPLICIT SEND), eventos de prueba recibidos por el LT o el UT (RECEIVE, OTHERWISE y TIMEOUT), constructivos (GOTO, ATTACH y REPEAT), y seudoevento 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.

14.9 *Eventos de prueba en TTCN*

14.9.1 *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 el § 10.8 y el § 14.9.5.2

Definición de sintaxis:

289 Send ::= [PCO_Identifier | FormalParIdentifier] "!" (ASP_Identifier | PDU_Identifier)

291 Receive ::= [PCO_Identifier | FormalParIdentifier] "?" (ASP_Identifier | PDU_Identifier)

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 sucesión de pruebas.

Ejemplo 53 – !CONreq o ?CONind

Si en una sucesión 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 54 – L! CONreq o L? CONind

14.9.2 *Eventos de recibir*

Una línea de evento RECEIVE evalúa fructuosamente 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 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.

b) La ASP o la PDU concuerda con la referencia de constricciones en la línea de evento.

c) 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.

14.9.3 *Eventos de enviar*

Una línea de evento SEND con un calificador es fructuosa si la expresión del calificador toma el valor TRUE. Los eventos SEND no calificados son siempre fructuosos. 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 los § 11, 12 y 13, 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.

La generación de un parámetro de 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.

14.9.4 *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 55 – !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 56 – ?A_PDU (VAR := A_PDU.FIELD),

pudiéndose utilizar VAR en líneas de evento subsiguientes a la recepción de A_PDU.

14.9.5 *Ejecución del árbol de comportamiento*

14.9.5.1 *Introducción*

El especificador de la sucesión 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 sucesión de pruebas exija que el LT o el UT intente iniciarlas o recibirlas, de ser necesario, repetidamente, hasta que una de ellas concuerde.

Ejemplo 57 – Ilustración de un árbol de comportamiento TTCN

Supóngase que puede producirse 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:

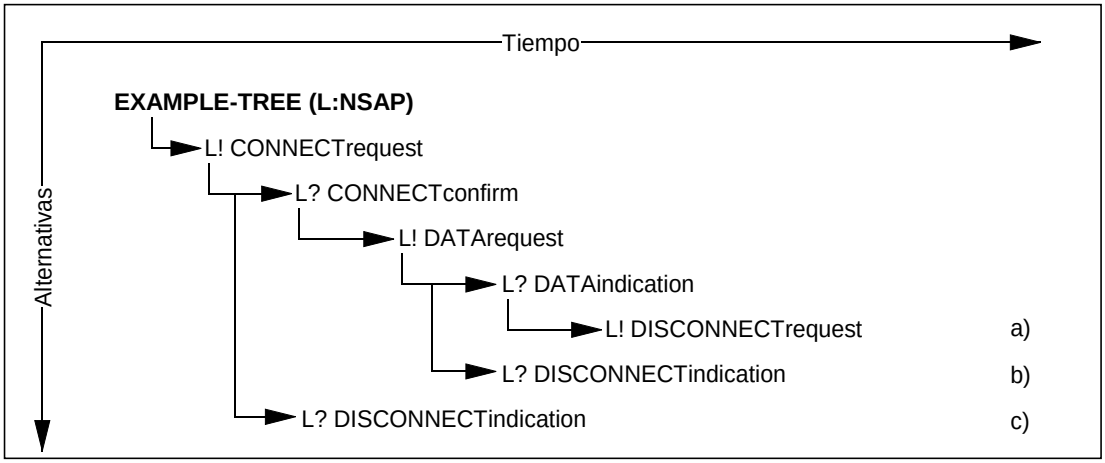
a) Petición CONEXIÓN, confirmación CONEXIÓN, petición DATOS, indicación DATOS, petición DESCONEXIÓN.

   La progresión puede ser atenuada 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-c) porque los eventos SEND L! son siempre fructuosos. 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 TCN:



En la TTCN no hay líneas, flechas ni nombres de hoja. El árbol de comportamiento del ejemplo anterior podría representarse como sigue:

Ejemplo 58 – 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 :** **Comentarios :** *Nota* – Puede simplificarse este ejemplo utilizando valores por defecto					
N.°	**Etiqueta**	**Descripción de comportamiento**	**Referencia 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

14.9.5.2 *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.

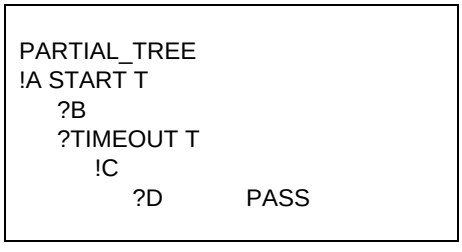
14.9.5.3 *Restricciones a la utilización de eventos*

Para evitar errores en casos de prueba se aplican las siguientes restricciones:

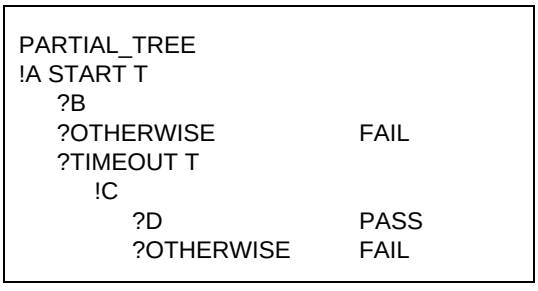
a) Un caso de prueba o paso de prueba no debe contener un comportamiento en el cual la velocidad de procesamiento relativa del MOT pudiera influir en los resultados. En evitación de tales problemas una línea de eventos RECEIVE, OTHERWISE o TIMEOUT sólo deberá ir seguida por otras líneas de eventos RECEIVE, 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 pertinentes.

Ejemplo 59 – Conjunto incompleto de eventos RECEIVE



a)



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 utilizando el enunciado OTHERWISE:

14.9.6 *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. Para ello, se define el evento enviar implícito con arreglo a la siguiente sintaxis:

Definición de sintaxis

290 ImplicitSend ::= "<" **IUT** "!" (ASP_Identifier | PDU_Identifier) ">"

La **IUT** sustituye en la sintaxis al identificador de PCO utilizado con un SEND o una RECEIVE normal, indicando que la IUT debe enviar la ASP o la PDU especificada. Los paréntesis angulares indican que se trata de un evento implícito, es decir, no se especifica la acción realizada por la IUT para desencadenar esta reacción, sino simplemente la propia reacción exigida.

Se considera que un evento IMPLICIT SEND siempre es fructuoso, 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(s) Recomendación(es) pertinente(s) sobre OSI permita(n) 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 sucesión de pruebas, el especificador de la sucesión 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 utilizando el método de prueba DS.

Nota – Por ejemplo, cuando se pruebe una realización 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 Recomendación pertinente sobre protocolo.

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 Recomendación sobre el 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, podría controlarse el SUT para provocar la iniciación de la ASP o PDU especificada. En la PIXIT (o en la documentación referenciada por la PIXIT), deberá especificarse 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 deberá haber un evento RECEIVE que concuerde con la ASP o la PDU, el cual debería haber sido enviado como resultado por la IUT.

Ejemplo 60 – Ejemplo de utilización de IMPLICIT SEND

Comportamiento dinámico de caso de prueba					
Nombre del caso de prueba : IMP1 **Grupo :** TTCN_EXAMPLES/IMPLICIT_SEND1/ **Finalidad :** Árbol parcial para ilustrar la utilización de IMPLICIT SEND **Valor por defecto :** **Comentarios :**					
N.°	**Etiqueta**	**Descripción de comportamiento**	**Referencia de constricciones**	**Veredicto**	**Comentarios**
. . 5 6 7 . . 12 . .		. . <IUT ! CR> L?CR L!CC . . L?OTHERWISE . .	 CR1 CR1 CC1		

14.9.7 *Evento OTHERWISE*

El evento predefinido OTHERWISE es el mecanismo de la TTCN utilizado para tratar eventos de prueba imprevistos de una forma controlada. OTHERWISE tiene la siguiente sintaxis:

Definición de sintaxis

292 Otherwise ::= [PCO_Identifier | FormalParIdentifier] "?" OTHERWISE

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.

Si en una sucesión de pruebas hay más de un PCO, el prefijo de OTHERWISE deberá ser un nombre del PCO que figure en la parte declaraciones, o en la lista de parámetros formales del árbol. 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 61 – 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 62 – 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.

14.9.8 *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 inscripción, 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 fructuoso.

Si no se indica ningún nombre de temporizador, cualquier evento TIMEOUT de la lista de temporizaciones es fructuoso. El evento TIMEOUT es fructuoso cuando la lista no está vacía. Cuando así ocurre, se vacía la lista completa de temporizaciones.

TIMEOUT tiene la siguiente sintaxis:

Definición de sintaxis

293 Timeout ::= "?" **TIMEOUT** [TimerIdentifier]

Ejemplo 63 – Utilización de TIMEOUT

	?TIMEOUT T			

Puesto que los eventos TIMEOUT no son eventos RECEIVE, las alternativas OTHERWISE antes indicadas no las hacen inalcanzables.

14.10 *Expresiones de TTCN*

14.10.1 *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 sucesiones de pruebas;

b) constantes de sucesiones de pruebas;

c) variables de caso de prueba y de sucesiones 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 vinculadas. Si se utiliza una variable no vinculada, se tratará de un error de caso de prueba.

Definición de sintaxis

```
303  Expression ::= SimpleExpression [RelOp SimpleExpression]
304  SimpleExpression ::= Term {AddOp Term}
305  Term ::= Factor {MultiplyOp Factor}
306  Factor ::= [UnaryOp] Primary
307  Primary ::= Value I DataObjectReference I OpCall I SelectExprIdentifier I "(" Expression ")"
336  Value ::= LiteralValue I ASN1_Value
337  LiteralValue ::= Number I BooleanValue I Bstring I Hstring I Ostring I Cstring
338  Number ::= (NonZeroNum {Num} ) I 0
339  NonZeroNum ::= 1 I 2 I 3 I 4 I 5 I 6 I 7 I 8 I 9
340  Num ::=  0 I NonZeroNum
341  BooleanValue ::= TRUE I FALSE
342  Bstring ::= " ' " {Bin I Wildcard} " ' " B
343  Bin ::= 0 I 1
344  Hstring ::=  " ' " {Hex I Wildcard} " ' " H
345  Hex ::= Num I A I B I C I D I E I F
346  Ostring ::= " ' " {Oct I Wildcard} " ' " O
347  Oct ::= Hex Hex
348  Cstring ::= " " " {Char I Wildcard I "\"} " " "
349  Char ::= /* REFERENCE – A character defined by the relevant character string type */
350  Wildcard ::= AnyOne I AnyOrNone
351  AnyOne ::=  "?"
352  AnyOrNone ::= "*"
308  DataObjectReference ::= DataObjectIdentifier {ComponentReference}
309  DataObjectIdentifier ::= TS_ParIdentifier I TS_ConstIdentifier I TS_VarIdentifier I TC_VarIdentifier I
       FormalParIdentifier I ASP_Identifier I PDU_Identifier
310  ComponentReference ::= RecordRef I ArrayRef I BitRef
311  RecordRef ::= Dot (ComponentIdentifier I PDU_Identifier I StructIdentifier I ComponentPosition)
312  ComponentIdentifier ::= ASP_ParIdentifier I PDU_FieldIdentifier I ElemIdentifier I ASN1_Identifier
314  ComponentPosition ::= "("Number")"
315  ArrayRef ::= Dot "[" ComponentNumber "]"
316  ComponentNumber ::= Expression
317  BitRef ::= Dot (BitIdentifier I "[" BitNumber "]")
318  BitIdentifier ::= Identifier
319  BitNumber ::= Expression
320  OpCall ::= TS_OpIdentifier ( ActualParList I "(" ")" )
321  AddOp ::= "+" | "–" | OR
322  MultiplyOp ::= "*" I "/" | MOD I AND
323  UnaryOp ::= "+" I "–" I NOT
324  RelOp ::= "=" | "<" | ">" | "< >" | ">=" | "<="
```

14.10.2 *Referencias para objetos de datos definidos en ASN.1*

Para hacer referencia a componentes u objetos de datos estructurados se han previsto los siguientes mecanismos de acceso:

a) Construcción de una referencia a un componente de uno de los tipos siguientes: SEQUENCE, SET y CHOICE, utilizando una notación de punto, es decir, añadiendo un punto y el nombre del componente deseado (identificador del componente) al identificador del objeto de datos. Se utilizará el identificador del componente si ha sido especificado.

b) Construcción de referencias a componentes no designados, indicando la posición del componente dentro de la definición de tipo entre paréntesis y numerando tales componentes de forma que el tipo comience con cero.

Definición de sintaxis

```
310  ComponentReference ::= RecordRef I ArrayRef | BitRef
311  RecordRef ::= Dot (ComponentIdentifier | PDU_Identifier | StructIdentifier | ComponentPosition)
312  ComponentIdentifier ::= ASP_ParIdentifier | PDU_FieldIdentifier | ElemIdentifier | ASN1_Identifier
314  ComponentPosition ::= "("Number")"
315  ArrayRef ::= Dot "[" ComponentNumber "]"
316  ComponentNumber ::= Expression
```

Para hacer referencia a un componente del tipo SEQUENCE OF o SET OF en ASN.1 se utiliza un índice encerrado entre corchetes.

El primer componente tiene el valor cero.

La expresión tomará como valor un INTEGER no negativo.

Ejemplo 64 – Referencias 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 65 – Referencias de campo de PDU

```
XY_PDUtype ::= SEQUENCE  {
                                    :
                                user_data        OCTET STRING,
                                    : }
```

En una línea de enunciado que contuviera XY_PDUtype, se podría escribir: L? XY_PDU (buffer := XY_PDUtype_user_data)

Se utiliza la notación índice para hacer referencia a elementos (bits) de tipo BITSTRING en ASN.1. Se supone que la BITSTRING está definida como SEQUENCE OF {BOOLEAN}. Si algunos bits de la BITSTRING están asociados con un identificador (bit denominado), puede utilizarse la notación de punto o el identificador para hacer referencia a ese bit.

Definición de sintaxis

```
317  BitRef ::= Dot (BitIdentifier | "[" BitNumber "]")
318  BitIdentifier ::= Identifier
319  BitNumber ::= Expression
```

El bit de más a la izquierda tiene el valor cero.

La expresión tomará como valor un INTEGER no negativo.

Ejemplo 66 – Referencias de componentes en tipos BITSTRING

```
B_type ::= BIT STRING { ack(0), poll(3) }
```

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».

En la Recomendación X.208, se definen tipos SET y 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 y SET OF de la misma forma que objetos de tipo SEQUENCE y SEQUENCE OF; es decir, que la referencia a componentes con número *i* significa siempre una referencia al campo i-ésimo declarado en el tipo. Los identificadores de componentes en ASN.1 comprenden el nombre de los parámetros de ASP en ASN.1, campos de PDU en ASN.1 o sub-componentes de parámetros de ASP en ASN.1 o de campos de PDU en ASN.1.

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 o SET OF, no se alteraría en cualquier operación realizada en la TTCN.

14.10.3 *Referencias para objetos de datos definidos utilizando cuadros*

Para construir referencias a parámetros de ASP, campos de PDU o elementos de tipos estructurados definidos en forma tabular, se utilizará la sintaxis definida en el § 14.10.2. Para especificar una referencia a un parámetro, campo o elemento, el identificador de objeto de datos deberá ir seguido de un punto (.) y de un identificador de parámetro, campo o elemento.

Cuando se defina un parámetro, campo o elemento como subestructura verdadera de un tipo definido, en un cuadro de tipos estructurados, la referencia a los elementos en la subestructura consistirá en la referencia al identificador del parámetro, campo o elemento seguida de un punto, con el identificador del elemento dentro de esa subestructura.

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.

14.10.4 *Asignaciones*

14.10.4.1 *Introducción*

Pueden asociarse eventos de prueba con una lista de asignaciones y/o un calificador. Las asignaciones se separan entre sí mediante comas y la lista se encierra entre paréntesis.

Definición de sintaxis

301 AssignmentList ::= "(" Assignment {Comma Assignment} ")"

302 Assignment ::= DataObjectReference ":=" Expression

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 sucesión de pruebas o del caso de prueba (o el parámetro de la ASP o el campo de la PDU) al valor de la expresión. La expresión no podrá contener variables no vinculadas.

Todas las asignaciones se producen según el orden en que aparecen, es decir, el procesamiento se efectúa de izquierda a derecha;

Ejemplo 67 – 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 fructuosamente, 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.

14.10.4.2 *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 vinculada (es decir, que puede tener una longitud infinita), deberá quedar vinculada al valor del lado derecho sin relleno. El relleno sólo será necesario cuando la variable sea de tipo cadena de longitud fija.

14.10.5 *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.

Definición de sintaxis

288 Qualifier ::= "[" Expression "]"

14.10.6 *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 de ASP, campo de PDU o elemento de estructura de una constricción en la parte comportamiento aplastará («sobre-escribirá») los valores de constricción en una línea de evento SEND.

Ejemplo 68 – 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.

Ejemplo 69 – Utilización de calificadores y asignaciones OTHERWISE

El evento OTHERWISE puede utilizarse 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 de asignación, esa asignación solamente tendrá lugar si se satisfacen todas las condiciones para que concuerde el OTHERWISE. Por ejemplo:

```
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 PCO 1 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 CharacterString. Motivo: si en PCO2 se recibe un evento que no satisface ninguno de estos escenarios, el OTHERWISE final concordará.

14.11 *Seudoeventos*

Se permite la utilización de asignaciones, calificadores y operaciones de temporización por sí mismos en una línea de enunciado de 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/o operaciones de temporizador precedidas de un calificador, éste se evaluará en primer lugar, procediéndose después a la evaluación de las asignaciones y/o operaciones de temporizador solamente en el caso en que el valor del calificador sea TRUE.

14.12 *Gestión de temporizador*

14.12.1 *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.

Podrán aplicarse 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 sucesión 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.

Definición de sintaxis

325 TimerOps ::= TimerOp {Comma TimerOp}
326 TimerOp ::= StartTimer | CancelTimer | ReadTimer

14.12.2 *Operación START*

La operación START se utiliza para indicar que debe arrancar un temporizador.

Definición de sintaxis

327 StartTimer ::= **START** TimerIdentifier ["(" TimerValue ")"]
117 TimerIdentifier ::= Identifier
329 TimerValue ::= Expression

Se deberá utilizar 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 vinculadas. Si se utiliza una variable no vinculada, 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 70 – Utilizaciones del temporizador START:

START T0

START T0 (V0)

START T1, START T2 (V2)

donde los T_i son identificadores de temporizador y los V_i son valores de temporizador.

La operación START puede aplicarse a un temporizador en marcha, en cuyo caso se cancela, repone y rearranca el temporizador. Cualquier inscripción en la lista de duraciones de temporización de este temporizador se eliminará de dicha lista.

14.12.3 *Operación CANCEL*

La operación CANCEL se utiliza para detener un temporizador en marcha.

Definición de sintaxis

328 CancelTimer ::= **CANCEL** [TimerIdentifier]
117 TimerIdentifier ::= Identifier
329 TimerValue ::= Expression

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 71 – Algunos usos del temporizador CANCEL:

CANCEL

CANCEL T0

CANCEL T1, CANCEL T2

CANCEL TI, START T3

donde los T_i son identificadores de temporizador.

14.12.4 *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 sucesión 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 convenio, la aplicación de la operación READTIMER a un temporizador inactivo devolverá el valor cero.

Definición de sintaxis

330 ReadTimer ::= **READTIMER** TimerIdentifier "(" DataObjectReference ")"
117 TimerIdentifier ::= Identifier

Ejemplo 72 – 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 sucesión 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 sucesión de pruebas.

Ejemplo 73 – READTIMER utilizado en combinación con otras operaciones de temporizador

READTIMER T1 (PASSED_TIME), CANCEL T1

READTIMER T1 (V1), START NEW_TIMER (V1)

14.13 *Constructivo ATTACH*

14.13.1 *Introducción*

Mediante el constructivo ATTACH, cuya sintaxis se indica a continuación, pueden adjuntarse unos árboles a otros:

Definición de sintaxis

296 Attach ::= "+" TreeReference [ActualParList]
298 TreeReference ::= TestStepIdentifier | TreeIdentifier
299 ActualParList ::= "(" ActualPar {Comma ActualPar} ")"
300 ActualPar ::= Value | PCO_Identifier

Las variables de caso de prueba o sucesión 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.

14.13.2 *Ambito 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.

Debe observarse 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 de árbol puede consistir en identificadores de paso de prueba o identificadores de árbol, donde:

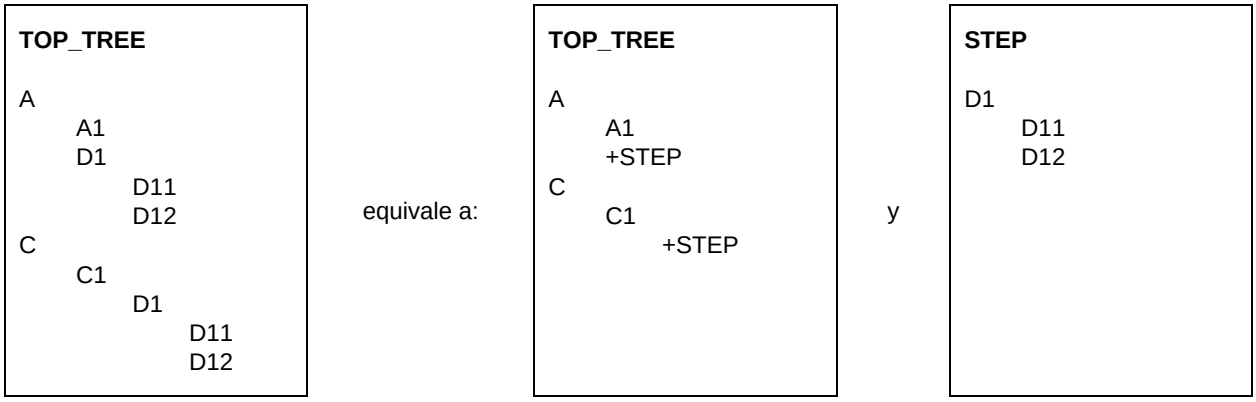
a) Un identificador de paso de prueba representa la adjunción de un paso de prueba que reside en la biblioteca de pasos de prueba. El paso de prueba se referencia mediante su identificador exclusivo.

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.

14.13.3 *Aspectos básicos de la adjunción de árbol*

Dado un árbol de comportamiento es posible 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 74 – Partición de un árbol grande en dos árboles más pequeños



Esta operación puede realizarse no solamente 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, pueden especificarse 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 pueden ser parametrizados.

14.13.4 *Significación de la adjunción de árbol*

14.13.4.1 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 A_i) de un conjunto ordenado de alternativas:

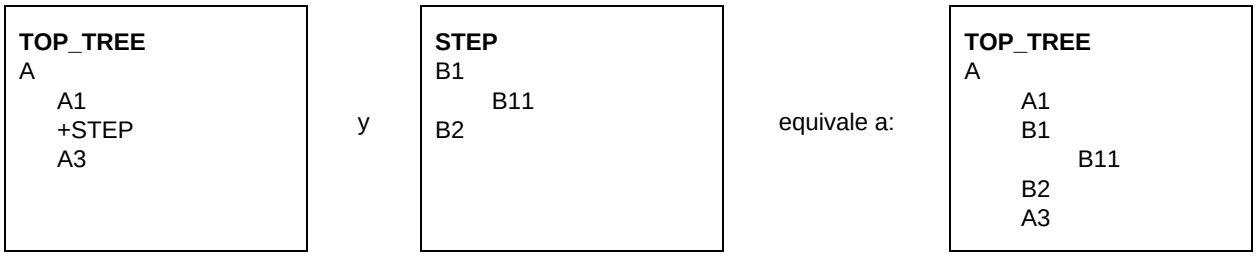
$$(A_1 , ... ,A_i , ... ,A_n)$$

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:

$$(A_1 , ... ,A_{(i-1)} , ... ,B_1 , ... ,B_m, A_{(i+1)} , ... ,A_n)$$

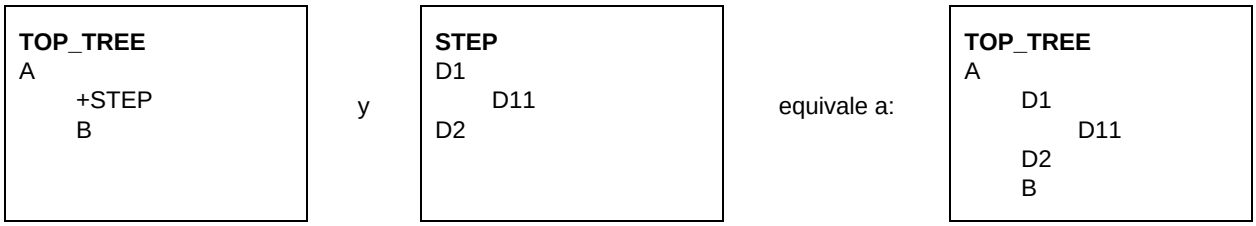
Cualquier comportamiento subsiguiente a las B se adjuntará a ellas.

Ejemplo 75 – 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 76 – 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 (*scoping*):

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:

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.

Ejemplo 77 – Sustitución de parámetros

```
TOP_TREE(L:NSAP; U:TSAP)
L!CONreq (M:=1)
    +STEP (U, M, 2)
```

y

```
STEP(PCO:TSAP; X,Y:INTEGER)
PCO?CONind (X:=Y)
```

equivale a:

```
TOP_TREE(L:NSAP; U:TSAP)
L!CONreq (M:=1)
    U?CONind (M:=2)
```

Ejemplo 78 – 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.°	**Etiqueta**	**Descripción de comportamiento**	**Referencia de constricciones**	**Veredicto**	**Comentarios**
1 2 3		?A +TEST_STEP_2(X) +LOCAL(5)	A1		
4 5		LOCAL(F:INTEGER) !B (TC_VAR:=F+Y)	B1	PASS	
Comentarios detallados: Cuando TEST_STEP1 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_STEP2. Sin embargo, en el constructivo ATTACH "+TEST_STEP2(X)" el valor del parámetro real sustituye al valor formal X. Esto hace que se utilice el valor X del parámetro real (en TEST_STEP1) en lugar de cualesquiera parámetros formales que aparezcan en la declaración TEST_STEP2. 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.					

14.13.5 *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 79 – 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 de constricciones en STEP no está parametrizada:

```
TOP_TREE
      :
      +STEP(C1(3))
      :
```

```
STEP(PAR:A_PDU)
      :
      !A_PDU      PAR
      :
```

14.13.6 *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 80 – Adjunción recursiva legal de un árbol

```
TOP_TREE
A
      +STEP
      B
```

y

```
STEP
C
      +TOP_TREE
D
```

Una expansión equivale a:

```
TOP_TREE
A
      C
            +TOP_TREE
                  B
      D
            B
```

Un árbol no podrá adjuntarse a sí mismo directa ni indirectamente, en su nivel máximo de samgrado.

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 81– Adjunción recursiva ilegal de un árbol

```
TOP_TREE
A
      +STEP
      B
```

y

```
STEP
C
      D
+TOP_TREE
```

Una expansión equivale a:

```
TOP_TREE
A
C
      D
            B
+TOP_TREE
      B
```

14.13.7 *Adjunción de árboles y valores por defecto*

Antes de adjuntar un árbol en cualquier lugar deberá haberse completado la expansión de los valores por defecto en el árbol (véase el § 14.18.2).

Nota – Cuando en una descripción de comportamiento se utilicen adjunciones de árboles y valores por defecto deberá actuarse con un cuidado especial.

14.14 *Etiquetas y constructivo GOTO*

Puede colocarse una etiqueta en la columna de etiquetas de cualquier línea de enunciado del árbol de comportamiento.

Nota – Cuando se ejecute una inscripción 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 inscripción.

Dentro de un árbol de comportamiento, puede especificarse un GOTO a una etiqueta siempre que ésta se asocie con la primera de un conjunto de alternativas, una de las cuales es un nodo predecesor del punto desde el cual se va a efectuar el GOTO. El constructivo GOTO sólo podrá utilizarse 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 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.

Definición de sintaxis

295 GoTo ::= ("–>" | **GOTO**) Label

Cualquier etiqueta deberá ser 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 – Puede colocarse una expresión booleana como antecedente inmediato de un GOTO, con lo cual se obtiene el efecto de un salto condicional.

Ejemplo 82 – Utilización de GOTO

Comportamiento dinámico de caso de prueba

Nombre del caso de prueba : GOTO_EX1
Grupo : TTCN_EXAMPLES/GOTO_EXAMPLE1/
Finalidad : Ilustrar la utilización de etiquetas y GOTO
Valor por defecto :
Comentarios :

N.°	Etiqueta	Descripción de comportamiento	Referencia 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		–>LA			
7	LE	[E=1]			
8	LF	!F	F1		

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).

14.15 *Constructivo REPEAT*

En este apartado se describe el mecanismo que ha de utilizarse en las descripciones de comportamiento para efectuar la iteración de un paso de prueba cierto número de veces, siendo la sintaxis del constructivo REPEAT como a continuación se indica.

Definición de sintaxis

297 Repeat ::= **REPEAT** TreeReference [ActualParList] UNTIL Qualifier

La referencia al árbol deberá ser 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 el § 14.13. El constructivo REPEAT tiene el siguiente significado: en primer lugar se ejecuta el árbol referenciado por la referencia de 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) del § 14.9.5.3.

Nota – Se recomienda utilizar el constructivo REPEAT, si es aplicable, en lugar de GOTO.

Ejemplo 83 – Utilizaciones de REPEAT (véase también el anexo D)

Comportamiento dinámico de caso de prueba					
Nombre del caso de prueba : RPT_EX1 **Grupo :** TTCN_EXAMPLES/REPEAT_EXAMPLE1/ **Finalidad :** Ilustrar la utilización de REPEAT **Valor por defecto :** **Comentarios :**					
N.°	**Etiqueta**	**Descripción de comportamiento**	**Referencia de constricciones**	**Veredicto**	**Comentarios**
1 2 3 4 5 6		(FLAG:=FALSE) !A REPEAT STEP1 (FLAG) UNTIL [FLAG] !D STEP1 (F:BOOLEAN) ?B (F:=TRUE) ?C (F:=FALSE)	 A1 D1 B1 C1	 PASS	
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.					

14.16 *Referencia a constricciones*

14.16.1 *Finalidad de la columna referencia a constricciones*

Esta columna permite efectuar referencias a una constricción específica situada en una ASP o en una PDU. Tales constricciones se definen en la parte constricciones (véanse los § 11, 12 y 13). La referencia a constricciones deberá estar presente junto con SEND, IMPLICIT SEND y RECEIVE. Si una ASP carece de parámetros, la referencia a constricciones es optativa y no estará presente con ningún otro tipo de sentencia en TTCN.

La sintaxis de referencia a constricciones es la siguiente:

Definición de sintaxis

277 ConsRef ::= ConstraintIdentifier [ActualCrefParList]
278 ActualCrefParList ::= "(" ActualCrefPar {Comma ActualCrefPar} ")"
279 ActualCrefPar ::= Value | DataObjectIdentifier | ConsRef

Ejemplo 84 – Referencia a constricciones sin lista de parámetros:

	N_SAP? CR_PDU	CR1		

14.16.2 *Paso de parámetros en referencias a constricciones*

Una referencia a constricciones puede poseer 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:

a) el número de parámetros reales deberá ser igual al número de parámetros formales, y

b) el valor de cada parámetro real será el de un elemento de su tipo formal correspondiente.

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 vinculadas. Si se utiliza una variable no vinculada, se produce un error de caso de prueba.

Ejemplo 85 – 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).

14.16.3 *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.

14.17 *Veredictos*

14.17.1 *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 inscripción 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.

Definición de sintaxis

```
281  Verdict ::= Pass | Fail | Inconclusive | Result
282  Pass ::= PASS | P | "(" PASS ")" | "(" P ")"
283  Fail ::= FAIL | F | "(" FAIL ")" | "(" F ")"
284  Inconclusive ::= INCONC | I | "(" INCONC ")" | "(" I ")"
285  Result ::= Identifier
```

Nota – Durante la ejecución del caso de prueba, cada vez que se produzca una inscripción en un árbol de comportamiento para el que exista una inscripción 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 inscripción en el árbol de comportamiento.

14.17.2 *Resultados preliminares*

En cada caso de prueba se dispone de una variable predefinida, llamada R, para el almacenamiento de todos los resultados intermedios. R puede adoptar los valores *éxito, fracaso, no concluyente* y *ninguno*. Estos valores son identificadores predefinidos y como tales son sensibles al tipo de letra, mayúscula o minúscula, con que se escriban.

R puede utilizarse 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 – 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 *pass* (éxito), *fail* (fracaso), *inconc* (no concluyente) y *none* (ninguno), 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 inscripción correspondiente en el árbol de comportamiento, se modificará el valor de la variable de caso de prueba predefinida R, según el cuadro 6/X.292.

CUADRO 6/X.292

Cálculo de la variable R

Valor vigente de R	Inscripción en la columna veredicto		
	(PASS)	**(INCONC)**	**(FAIL)**
ninguno	éxito	no concluyente	fracaso
éxito	éxito	no concluyente	fracaso
no concluyente	no concluyente	no concluyente	fracaso
fracaso	fracaso	fracaso	fracaso

Nota – 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.

14.17.3 *Veredicto final*

Cuando haya de especificarse un veredicto final explícito en la columna de veredicto, éste será uno de los siguientes:

a) **P** o **PASS** indicando que debe registrarse un veredicto de éxito.

b) **I** o **INCONC**, indicando que debe registrarse un veredicto no concluyente.

c) **F** o **FAIL**, indicando que debe registrarse un veredicto de fracaso.

d) la variable predefinida R, indicando que el valor de R ha de tomarse como veredicto final, a menos que dicho valor sea *none* (ninguno), en cuyo caso se registra un error de caso de prueba en lugar de un veredicto final.

CUADRO 7/X.292

Cálculo del veredicto final R

Valor vigente de R	Inscripción en la columna veredicto			
	PASS	**INCONC**	**FAIL**	R
ninguno	pass	inconc	fail	*error*
éxito	pass	inconc	fail	pass
no concluyente	*error*	inconc	fail	inconc
fracaso	*error*	*error*	fail	fail

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 Recomendación X.291, solamente podrá especificarse 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 – 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 – En tales casos, se inscribirá «error de caso de prueba» en el registro cronológico de conformidad.

14.17.4 *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.

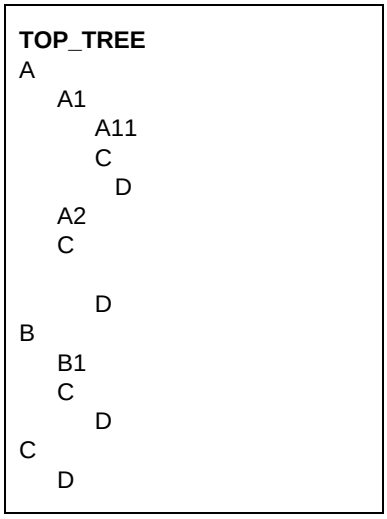
14.18 *Significado de los valores por defecto*

14.18.1 *Introducción*

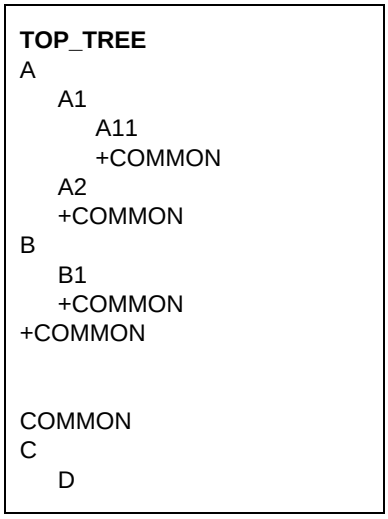
En muchos casos se utilizará el comportamiento por defecto para destacar un conjunto de trayectos interesantes a través de una prueba, mediante la declaración de las alternativas comunes menos interesantes (+ su comportamiento subsiguiente) como comportamiento por defecto.

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.

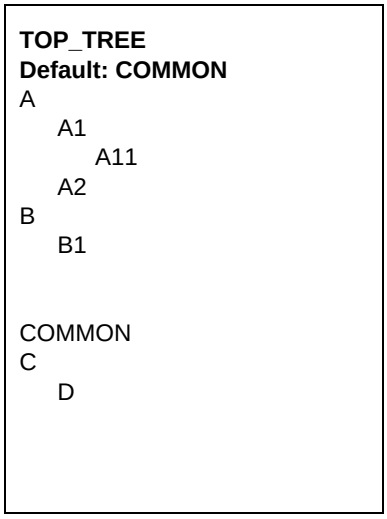
Ejemplo 86 – Identificación de un árbol por defecto



1: conjunto completo de alternativas



2: adjunción de árbol explícita



3: el valor por defecto efectúa la misma función que 2

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, se asignará un veredicto final a cada hoja de árbol por defecto. Si el veredicto final es consecuencia de un enunciado OTHERWISE del árbol por defecto, dicho veredicto final será FAIL.

14.18.2 *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:

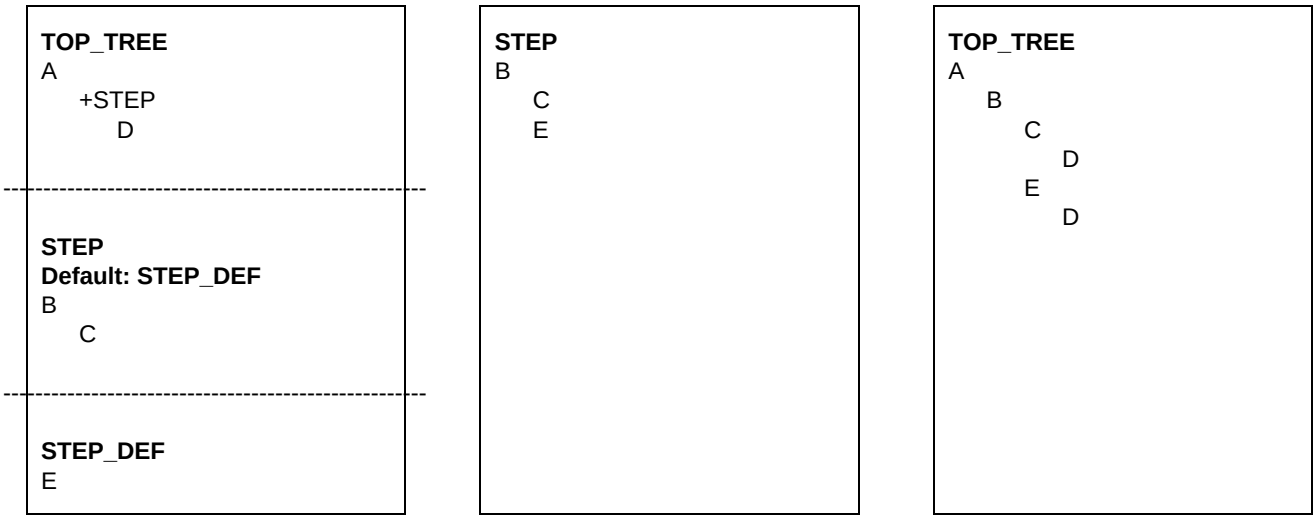
a) antes de que se expanda el árbol como árbol adjuntado; y

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 87 – Ubicación de un valor por defecto frente a un paso de prueba

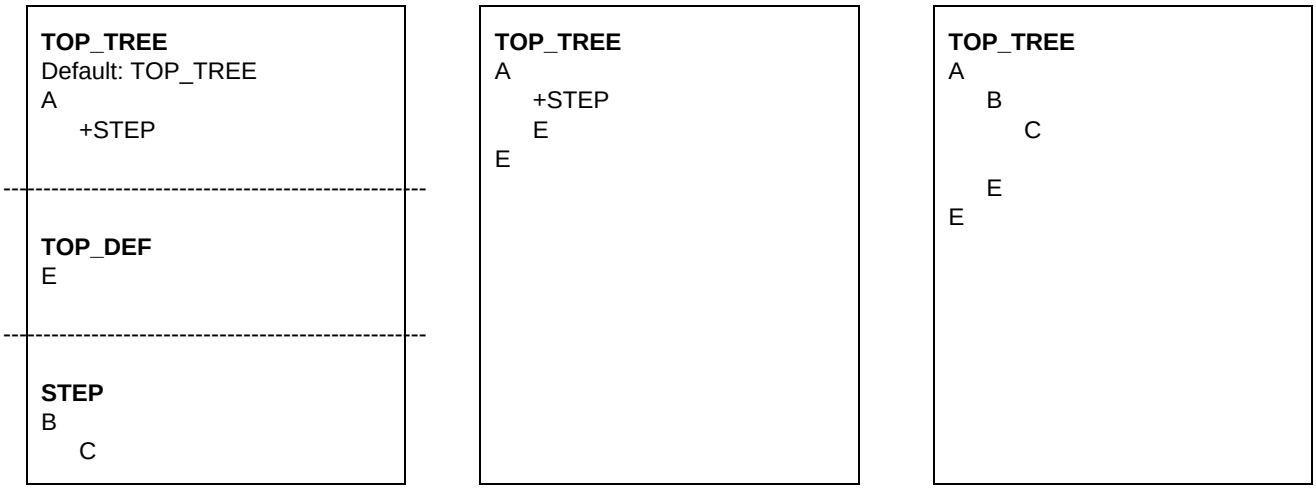


1: TOP_TREE adjunta STEP que tiene el valor por defecto STEP_DEF

2: STEP_DEF expandido a STEP

3: STEP expandido a TOP_TREE

Ejemplo 88 – Ubicación de un valor por defecto frente a un árbol llamante:

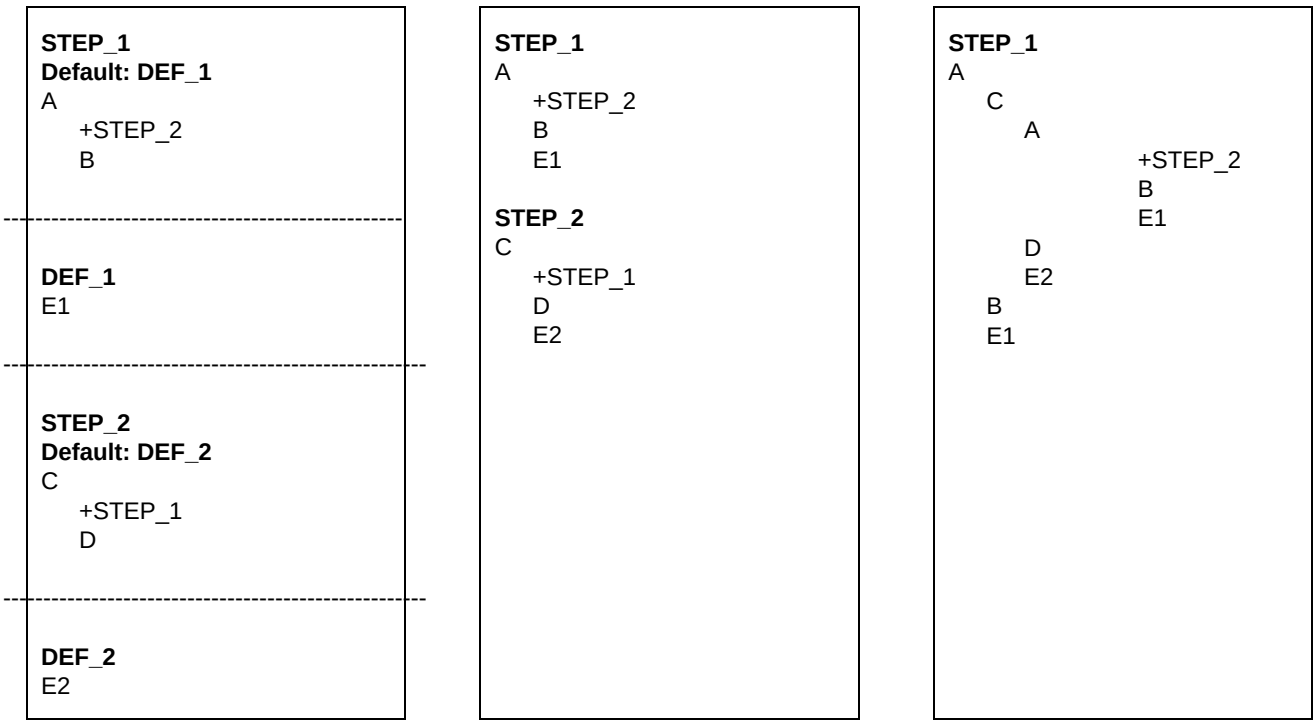


1: TOP_TREE adjunta STEP. TOP_TREE tiene el valor por defecto TOP_DEF

2: TOP_DEF expandido a TOP_TREE

3: STEP expandido a TOP_TREE

Ejemplo 89 – Caso de adjudicación cíclica de árbol



1: STEP_1 y STEP_2 se adjuntan entre sí. STEP_1 tiene como valor por defecto DEF_1. STEP_2 tiene como valor por defecto DEF_2

2: DEF_1 expandido a STEP_1 y DEF_2 expandido a STEP_2

3: Tras una expansión de STEP_2 sin valor por defecto y una expansión de STEP_1 sin valor por defecto

Nota – No se aconsejan tales adjunciones cíclicas.

14.19 *Referencias de valores por defecto*

Los comportamientos de paso de prueba y caso de prueba hacen referencia al comportamiento por defecto en la biblioteca de valores por defecto a través de la inscripción de valores por defecto en el encabezamiento del cuadro. Esta referencia localiza el valor por defecto mediante su identificador exclusivo. Los valores por defecto pueden ser parametrizados. La lista de parámetros reales deberá cumplir lo siguiente:

a) igualdad entre el número de parámetros reales y el número de parámetros formales, y

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 vinculadas cuando se invoque la constricción.

Definición de sintaxis

238 DefaultReference ::= DefaultIdentifier [ActualParList]

El identificador de valores por defecto será una referencia a un valor por defecto definido en la biblioteca de valores por defecto.

Ejemplo 90 – Referencia de valores por defecto

90.1

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** puede subdividirse en este caso de prueba con el comportamiento por defecto DEF1					
N.°	**Etiqueta**	**Descripción de comportamiento**	**Referencia 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	DSC1	PASS	Aceptar

90.2

Comportamiento dinámico por defecto					
Nombre del comportamiento por defecto : DEF1(X:XSAP) **Grupo :** TTCN_EXAMPLES/DEFAULTS_LIB/DEFAULT_1/ **Objetivo :** Ilustración de un comportamiento por defecto simple **Comentarios :**					
N.°	**Etiqueta**	**Descripción de comportamiento**	**Referencia 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 Continuación de página

15.1 *Continuación de página de 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 pueden dividirse 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 91 – Cuadro de parámetros de una sucesión de pruebas continuada

Declaraciones de parámetros de sucesión de pruebas			
Nombre del parámetro	**Tipo**	**Ref. de PICS/PIXIT**	**Comentarios**
PAR1 PAR2 PAR3	INTEGER BOOLEAN IA5String	Cuestión aa de la PICS Cuestión bb de la PICS Cuestión cc de la PIXIT	

Continúa en la página siguiente página *n*

Continuación de la página anterior página *n* + 1

Declaraciones de parámetros de sucesión de pruebas			
Nombre del parámetro	**Tipo**	**Ref. de PICS/PIXIT**	**Comentarios**
PAR4 PAR5	BOOLEAN HEXSTRING	Cuestión dd de la PICS Cuestión ee de la PICS	

15.2 *Continuación de página de cuadros de comportamiento dinámico*

Cuando sea necesario continuar un cuadro de comportamiento dinámico, podrá utilizarse 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 de 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 de 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.

Ejemplo 92 – División de páginas con realineación de sangrado utilizando indicadores entre corchetes

Comportamiento dinámico de caso de prueba					
Nombre del caso de prueba : SPLIT2 **Grupo :** TTCN_EXAMPLES/PAGE_SPLITTING2/ **Finalidad :** Demostrar la división de páginas de un cuadro de comportamiento dinámico **Valor por defecto :** **Comentarios :**					
N.°	**Etiqueta**	**Descripción de comportamiento**	**Referencia de constricciones**	**Veredicto**	**Comentarios**
1 2 3 4		?A ?H1 ?J ?K	CA CH1 CJ CK		

[1] [3]

Continúa en la página siguiente página *n*

Continuación de la página anterior página *n* + 1

[1] [3]

N.°	Etiqueta	Descripción de comportamiento	Referencia de constricciones	Veredicto	Comentarios
5 6 7		?L ?M ?H2	CL CM CH2	 PASS FAIL	

Los escritores de sucesiones de pruebas pueden utilizar, facultativamente, una rejilla en las partes superior e inferior del cuadro de comportamiento. Cuando una página continúe en la página siguiente, puede desplazarse la rejilla hacia la izquierda, permitiendo así el desplazamiento del sangrado.

Ejemplo 93 – Realineación de sangrado utilizando un indicador de rejilla:

Comportamiento dinámico de caso de prueba					
Nombre del caso de prueba : SPLIT2 **Grupo** : TTCN_EXAMPLES/PAGE_SPLITTING3/ **Finalidad** : Demostrar la división de páginas de un cuadro de comportamiento dinámico empleando una rejilla **Valor por defecto** : **Comentarios** :					
N.°	**Etiqueta**	**Descripción de comportamiento** 0 1 2 3	**Referencia de constricciones**	**Veredicto**	**Comentarios**
1 2 3 4		?A ?H1 ?J ?K 0 1 2 3	CA CH1 CJ CK		

Continúa en la página siguiente página *n*

Continuación de la página anterior página *n* + 1

N.°	**Etiqueta**	**Descripción de comportamiento** 1 2 3 4 ...	**Referencia de constricciones**	**Veredicto**	**Comentarios**
5 6 7		?L ?M ?H2 1 2 3 4 ...	CL CM CH2	PASS FAIL	

ANEXO A

(a la Recomendación X.292)

(Este anexo es parte integrante de la presente Recomendación)

Sintaxis y semántica estática de la TTCN

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 ATSs;

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 *Convenios para la descripción de sintaxis*

A.2.1 *Metanotación sintáctica*

En el cuadro A-1/X.292 se define la metanotación utilizada para especificar la forma ampliada de gramática BNF para TTCN (denominada en lo que sigue BNF).

CUADRO A-1/X.292

Metanotación sintáctica de TTCN.MP

::=	Se define como
|	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

A.2.2 *Definiciones de sintaxis de TTCN.MP*

A.2.2.1 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.

A.2.2.2 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**

A.2.2.3 Los campos individuales de una línea se representan por:

$KEYWORD ....

No hay palabra clave de cierre.

Ejemplo A.3 – TS_Parlid ::= **$TS_Parld** TS_Parlidentifier

Ejemplo A.4 – TS_Parlidentifier ::= Identifier

A.2.2.4 Los conjuntos de cuadros, hasta la sucesión 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**

A.2.2.5 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

A.3 *Producciones de sintaxis de TTCN.MP en BNF*

A.3.1 *Sucesión de pruebas*

1 Suite ::= **$Suite** SuiteId SuiteOverviewPart DeclarationsPart ConstraintsPart DynamicPart **$End_Suite**

2 SuiteId ::= **$SuiteId** SuiteIdentifier

3 SuiteIdentifier ::= Identifier

A.3.2 *Visión general de sucesión de pruebas*

A.3.2.1 *Generalidades*

4 SuiteOverviewPart ::= **$SuiteOverviewPart** SuiteStructure TestCaseIndex [TestStepIndex] [DefaultIndex] **$End_SuiteOverviewPart**

A.3.2.2 *Estructura de sucesión de pruebas*

5 SuiteStructure ::= **$Begin_SuiteStructure** SuiteId StandardsRef PICSref PIXITref TestMethods [Comment] Structure&Objectives [Comment] **$End_SuiteStructure**

6 StandardsRef ::= **$StandardsRef** BoundedFreeText

7 PICSref ::= **$PICSref** BoundedFreeText

8 PIXITref ::= **$PIXITref** BoundedFreeText

9 TestMethods ::= **$TestMethods** BoundedFreeText

10 Comment ::= **$Comment** [BoundedFreeText]

11 Structure&Objectives ::= **$Structure&Objectives** {Structure&Objective} **$End_Structure&Objectives**

12 Structure&Objective ::= **$Structure&Objective** TestGroupRef SelExprId Objective **$End_Structure&Objective**

13 SelExprId ::= **$SelectExprId** [SelectExprIdentifier]

A.3.2.3 *Índice de casos de prueba*

14 TestCaseIndex ::= **$Begin_TestCaseIndex** {CaseIndex}+ [Comment] **$End_TestCaseIndex**

15 CaseIndex ::= **$CaseIndex** TestGroupRef TestCaseId SelExprId Description **$End_CaseIndex**

/*SEMÁNTICA ESTÁTICA – La relación de los casos de prueba deberá figurar en el orden en que aparecen en la parte dinámica*/

/*SEMÁNTICA ESTÁTICA – Se proporcionará una referencia de grupo de pruebas explícita para el primer caso de prueba de cada grupo*/

/*SEMÁNTICA ESTÁTICA – Se proporcionará una referencia de grupo de pruebas explícita para cada caso de prueba que siga inmediatamente a un grupo de pruebas*/

16 Description ::= **$Description** BoundedFreeText

A.3.2.4 *Índice de pasos de prueba*

17 TestStepIndex ::= **$Begin_TestStepIndex** {StepIndex} [Comment] **$End_TestStepIndex**

18 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 de paso de prueba explícita para el primer paso de prueba de cada grupo*/

/*SEMÁNTICA ESTÁTICA – Se proporcionará una referencia de grupo de pasos de prueba explícita para cada paso de prueba que siga inmediatamente a un grupo de pasos de prueba*/

A.3.2.5 *Índice de valores por defecto*

19 DefaultIndex ::= **$Begin_DefaultIndex** {DefIndex} [Comment] **$End_DefaultIndex**

20 DefIndex ::= **$DefIndex** DefaultRef DefaultId Description **$DefIndex**

/*SEMÁNTICA ESTÁTICA – Defaultd 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 de grupo valor por defecto explícita para el primer valor por defecto de cada grupo*/

/*SEMÁNTICA ESTÁTICA – Se proporcionará una referencia de grupo valores por defecto explícita para cada valor por defecto que siga inmediatamente a un grupo valores por defecto*/

A.3.3 *Parte declaraciones*

A.3.3.1 *Generalidades*

21 DeclarationsPart ::= **$DeclarationsPart** Definitions Parameterization&Selection Declarations ComplexDefinitions **$End_DeclarationsPart**

A.3.3.2 *Definiciones*

A.3.3.2.1 *Generalidades*

22 Definitions ::= [TS_TypeDefs] [TS_OpDefs]

A.3.3.2.2 *Definiciones de tipo de sucesión de pruebas*

23 TS_TypeDefs ::= **$TS_TypeDefs** [SimpleTypeDefs] [StructTypeDefs] [ASN1_TypeDefs] [ASN1_TypeRefs] **$End_TS_TypeDefs**

A.3.3.2.3 *Definiciones de tipo simple*

24 SimpleTypeDefs ::= **$Begin_SimpleTypeDefs** {SimpleTypeDef}+ [Comment] **$End_SimpleTypeDefs**

25 SimpleTypeDef ::= **$SimpleTypeDef** SimpleTypeId SimpleTypeDefinition [Comment] **$End_SimpleTypeDef**

26 SimpleTypeId ::= **$SimpleTypeId** SimpleTypeIdentifier

27 SimpleTypeIdentifier ::= Identifier

28 SimpleTypeDefinition ::= **$SimpleTypeDefinition** Type&Restriction

29 Type&Restriction ::= Type [Restriction]

30 Restriction ::= LengthRestriction | IntegerRange | SimpleValueList

/*SEMÁNTICA ESTÁTICA – El conjunto de valores definidos por Restriction deberá ser un subconjunto verdadero de los valores del tipo base*/

31 LengthRestriction ::= SingleTypeLength | RangeTypeLength

/*SEMÁNTICA ESTÁTICA – LengthRestriction será proporcionada solamente cuando el tipo base sea un tipo cadena (es decir, BITSTRING, HEXSTRING, OCTETSTRING o CharacterString) o se derive de un tipo cadena*/

32 SingleTypeLength ::= "[" Number "]"

33 RangeTypeLength ::= "[" LowerTypeBound To UpperTypeBound "]

34 IntegerRange ::= "(" LowerTypeBound To UpperTypeBound ")"

/*SEMÁNTICA ESTÁTICA – LowerTypebound será menor que UpperTypebound*/

35 LowerTypeBound ::= [Minus] Number | Minus **INFINITY**

36 UpperTypeBound ::= [Minus] Number | **INFINITY**

37 To ::= TO | ".."

38 SimpleValueList ::= "(" [Minus] LiteralValue {Comma [Minus] LiteralValue} ")"

/*SEMÁNTICA ESTÁTICA – Si se utiliza Minus en SimpleValueList, LiteralValue deberá ser un número*/

/*SEMÁNTICA ESTÁTICA – Los LiteralValues deberán utilizarse como tipo base y deberán ser un subconjunto verdadero de los valores definidos por el tipo de base*/

A.3.3.2.4 *Definiciones de tipo estructurado*

39 StructTypeDefs ::= **$StructTypeDefs** {StructTypeDef}+ **$End_StructTypeDefs**

40 StructTypeDef ::= **$Begin_StructTypeDef** StructId [Comment] ElemDcls [Comment] **$End_StructTypeDef**

41 StructId ::= **$StructId** StructId&FullId

42 StructId&FullId ::= StructIdentifier [FullIdentifier]

43 FullIdentifier ::= "(" BoundedFreeText ")"

/*SEMÁNTICA ESTÁTICA – Algunos objetos de TTCN permiten utilizar nombres abreviados según se indica en la Recomendación de protocolo apropiada. Si se utiliza una abreviatura, FullIdentifier deberá aparecer en la declaración del objeto*/

44 StructIdentifier ::= Identifier

45 ElemDcls ::= **$ElemDcls** {ElemDcl}+ **$End_ElemDcls**

46 ElemDcl ::= **$ElemDcl** ElemId ElemType [Comment] **$End_ElemDcl**

47 ElemId ::= **$ElemId** ElemId&FullId

48 ElemId&FullId ::= ElemIdentifier [FullIdentifier]

49 ElemIdentifier ::= Identifier

50 ElemType ::= **$ElemType** Type&Attributes

/*SEMÁNTICA ESTÁTICA – En Type&Attributes no habrá referencias recursivas (ni directas ni indirectas)*/

/*SEMÁNTICA ESTÁTICA – Un tipo de elemento estructurado deberá ser un PredefinedType, un TS_TypeIdentifier, un PDU_Identifier o una PDU*/

A.3.3.2.5 *Definiciones de tipo ASN.1*

51 ASN1_TypeDefs ::= **$ASN1_TypeDefs** {ASN1_TypeDef}+ **$End_ASN1_TypeDefs**

52 ASN1_TypeDef ::= **$Begin_ASN1_TypeDef** ASN1_TypeId [Comment] ASN1_TypeDefinition [Comment] **$End_ASN1_TypeDef**

53 ASN1_TypeId ::= **$ASN1_TypeId** ASN1_TypeId&FullId

54 ASN1_TypeId&FullId ::= ASN1_TypeIdentifier [FullIdentifier]

55 ASN1_TypeIdentifier ::= Identifier

56 ASN1_TypeDefinition ::= **$ASN1_TypeDefinition** ASN1_Type&LocalTypes **$End_ASN1_TypeDefinition**

57 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 otras tablas de definiciones de tipo ASN.1, definidos por referencia en las tablas de referencia de tipo ASN.1, o definidos localmente (es decir, ASN1_LocalTypes) en la misma tabla, inmediatamente después de la primera definición de tipo*/

/*SEMÁNTICA ESTÁTICA – ASN1_LocalTypes no se utilizarán en otras partes de la sucesión de pruebas*/

58 ASN1_Type ::= Type

/*REFERENCE – Donde Type es un no terminal definido en la Recomendación X.208*/

/*SEMÁNTICA ESTÁTICA – Cada referencia de 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 referencias de tipo externo definidas en la Recomendación X.208*/

59 ASN1_LocalType ::= Typeassignment

/*REFERENCE – Donde Typeassignment es un no terminal definido en la Recomendación X.208*/

/*SEMÁNTICA ESTÁTICA – Las definiciones de tipo ASN.1 utilizadas en TTCN no utilizarán referencias de tipo externo definidas en la Recomendación X.208*/

A.3.3.2.6 *Definiciones de tipo ASN.1 por referencia*

60 ASN1_TypeRefs ::= **$Begin_ASN1_TypeRefs** {ASN1_TypeRef}+ [Comment] **$End_ASN1_TypeRefs**

61 ASN1_TypeRef ::= **$ASN1_TypeRef** ASN1_TypeId ASN1_TypeReference ASN1_ModuleId [Comment] **$End_ASN1_TypeRef**

/*SEMÁNTICA ESTÁTICA – No deberá especificarse ASN1_TypeId en un FullIdentifier*/

62 ASN1_TypeReference ::= **$ASN1_TypeReference** TypeReference

63 TypeReference ::= typereference

/*REFERENCE – Donde typereference está definida en el § 8.2 de la Recomendación X.208*/

64 ASN1_ModuleId ::= **$ASN1_ModuleId** ModuleIdentifier

65 ModuleIdentifier ::= ModuleIdentifier

/*REFERENCE – Donde ModuleIdentifier es un no terminal definido en la Recomendación X.208*/

/*SEMÁNTICA ESTÁTICA – ModuleIdentifier deberá ser exclusivo dentro del dominio de interés*/

A.3.3.2.7 *Definiciones de operaciones de sucesiones de pruebas*

66 TS_OpDefs ::= **$TS_OpDefs** {TS_OpDef}+ **$End_TS_OpDefs**

67 TS_OpDef ::= **$Begin_TS_OpDef** TS_OpId TS_OpResult [Comment] TS_OpDescription [Comment] **$End_TS_OpDef**

68 TS_OpId ::= **$TS_OpId** TS_OpId&ParList

69 TS_OpId&ParList ::= TS_OpIdentifier [FormalParList]

/*SEMÁNTICA ESTÁTICA – Un parámetro formal de operación de sucesión de pruebas Type deberá ser un PredefinedType, TS_TypeIdentifier o PDU_Identifier*/

70 TS_OpIdentifier ::= Identifier

71 TS_OpResult ::= **$TS_OpResult** Type

/*SEMÁNTICA ESTÁTICA – Type será un PredefinedType, un TS_TypeIdentifier o PDU_Identifier*/

72 TS_OpDescription ::= **$TS_OpDescription** BoundedFreeText

A.3.3.3 *Parametrización y selección*

A.3.3.3.1 *Generalidades*

73 Parameterization&Selection ::= [TS_ParDcls] [SelectExprDefs]

A.3.3.3.2 *Declaraciones de parámetros de sucesiones de pruebas*

74 TS_ParDcls ::= **$Begin_TS_ParDcls** {TS_ParDcl}+ [Comment] **$End_TS_ParDcls**

75 TS_ParDcl ::= **$TS_ParDcl** TS_ParId TS_ParType PICS_PIXITref [Comment] **$End_TS_ParDcl**

76 TS_ParId ::= **$TS_ParId** TS_ParIdentifier

77 TS_ParIdentifier ::= Identifier

78 TS_ParType ::= **$TS_ParType** Type

/*SEMÁNTICA ESTÁTICA – Type será un PredefinedType, TS_TypeIdentifier o un PDU_Identifier*/

79 PICS_PIXITref ::= **$PICS_PIXITref** BoundedFreeText

A.3.3.3.3 *Definiciones de expresiones de selección de caso de prueba*

80 SelectExprDefs ::= **$Begin_SelectExprDefs** {SelectExprDef}+ [Comment] **$End_SelectExprDefs**

81 SelectExprDef ::= **$SelectExprDef** SelectExprId SelectExpr [Comment] **$End_SelectExprDef**

82 SelectExprId ::= **$SelectExprId** SelectExprIdentifier

83 SelectExprIdentifier ::= Identifier

84 SelectExpr ::= **$SelectExpr** SelectionExpression

85 SelectionExpression ::= Expression

/*SEMÁNTICA ESTÁTICA – SelectionExpression sólo contendrá LiteralValues, TS_ParIdentifiers, TS_ConstIdentifiers y SelectExprIdentifiers*/

/*SEMÁNTICA ESTÁTICA – SelectionExpression tomará un valor BOOLEAN específico*/

/*SEMÁNTICA ESTÁTICA – Expression no referirá recursivamente (ni directa ni indirectamente) al SelExpreIdentifier que está siendo definido por esta expresión*/

A.3.3.4 *Declaraciones*

A.3.3.4.1 *Generalidades*

86 Declarations ::= [TS_ConstDcls] [TS_VarDcls] [TC_VarDcls] PCO_Dcls [TimerDcls]

A.3.3.4.2 *Declaraciones de constantes en sucesiones de pruebas*

87 TS_ConstDcls ::= **$Begin_TS_ConstDcls** {TS_ConstDcl}+ [Comment] **$End_TS_ConstDcls**

88 TS_ConstDcl ::= **$TS_ConstDcl** TS_ConstId TS_ConstType TS_ConstValue [Comment] **$End_TS_ConstDcl**

89 TS_ConstId ::= **$TS_ConstId** TS_ConstIdentifier

90 TS_ConstIdentifier ::= Identifier

91 TS_ConstType ::= **$TS_ConstType** Type

/*SEMÁNTICA ESTÁTICA – Type será un PredefinedType, un TS_TypeIdentifier o un PDU_Identifier*/

92 TS_ConstValue ::= **$TS_ConstValue** DeclarationValue

93 DeclarationValue ::= Expression

/*SEMÁNTICA ESTÁTICA – DeclarationValue sólo contendrá LiteralValues, TS_ParIdentifiers y TS_ConstIdentifiers*/

/*SEMÁNTICA ESTÁTICA – DeclarationValue tomará el valor de un elemento de su tipo declarado*/

A.3.3.4.3 *Declaraciones de variables de sucesiones de pruebas*

94 TS_VarDcls ::= **$Begin_TS_VarDcls** {TS_VarDcl}+ [Comment] **$End_TS_VarDcls**

95 TS_VarDcl ::= **$TS_VarDcl** TS_VarId TS_VarType TS_VarValue [Comment] **$End_TS_VarDcl**

96 TS_VarId ::= **$TS_VarId** TS_VarIdentifier

97 TS_VarIdentifier ::= Identifier

98 TS_VarType ::= **$TS_VarType** Type

/*SEMÁNTICA ESTÁTICA – Type será un PredefinedType, un TS_TypeIdentifier o un PDU_Identifier*/

99 TS_VarValue ::= **$TS_VarValue** [DeclarationValue]

A.3.3.4.4 *Declaraciones de variables de caso de prueba*

100 TC_VarDcls ::= **$Begin_TC_VarDcls** {TC_VarDcl}+ [Comment] **$End_TC_VarDcls**

101 TC_VarDcl ::= **$TC_VarDcl** TC_VarId TC_VarType TC_VarValue [Comment] **$End_TC_VarDcl**

102 TC_VarId ::= **$TC_VarId** TC_VarIdentifier

103 TC_VarIdentifier ::= Identifier

104 TC_VarType ::= **$TC_VarType** Type

/*SEMÁNTICA ESTÁTICA – Type será un PredefinedType, un TS_TypeIdentifier o un PDU_Identifier*/

105 TC_VarValue ::= **$TC_VarValue** [DeclarationValue]

A.3.3.4.5 *Declaraciones de PCO*

106 PCO_Dcls ::= **$Begin_PCO_Dcls** {PCO_Dcl}+ [Comment] **$End_PCO_Dcls**

/*SEMÁNTICA ESTÁTICA – De acuerdo con la Recomendación X.290, el número de puntos de control y observación (PCO) estará relacionado con el método de prueba utilizado*/

107 PCO_Dcl ::= **$PCO_Dcl** PCO_Id PCO_TypeId P_Role [Comment] **$End_PCO_Dcl**

108 PCO_Id ::= **$PCO_Id** PCO_Identifier

109 PCO_Identifier ::= Identifier

110 PCO_TypeId ::= **$PCO_TypeId** PCO_TypeIdentifier

111 PCO_TypeIdentifier ::= Identifier

112 P_Role ::= **$PCO_Role** PCO_Role

113 PCO_Role ::= **UT** | **LT**

A.3.3.4.6 *Declaraciones de temporizadores*

114 TimerDcls ::= **$Begin_TimerDcls** {TimerDcl}+ [Comment] **$End_TimerDcls**

115 TimerDcl ::= **$TimerDcl** TimerId Duration Unit [Comment] **$End_TimerDcl**

116 TimerId ::= **$TimerId** TimerIdentifier

117 TimerIdentifier ::= Identifier

118 Duration ::= **$Duration** [DeclarationValue]

/*SEMÁNTICA ESTÁTICA – Declarationvalue tomará un valor INTEGER positivo distinto de cero*/

119 Unit ::= **$Unit** TimeUnit

120 TimeUnit ::= **ps** | **na** | **μs** | **ms** | **s** | **min**

/*SEMÁNTICA ESTÁTICA – Si un temporizador está derivado del PICS/PIXIT, la declaración de temporizador especificará las mismas unidades que la inscripción PICS/PIXIT*/

A.3.3.5 *Definiciones de tipo ASP y PDU*

A.3.3.5.1 *Generalidades*

121 ComplexDefinitions ::= [ASP_TypeDefs] PDU_TypeDefs [AliasDefs]

A.3.3.5.2 *Definiciones de tipo*

122 ASP_TypeDefs ::= **$ASP_TypeDefs** [TTCN_ASP_TypeDefs] [ASN1_ASP_TypeDefs] [ASN1_ASP_TypeDefsByRef] **$End_ASP_TypeDefs**

A.3.3.5.3 *Definiciones de tipo ASP en forma de tabla*

123 TTCN_ASP_TypeDefs ::= **$TTCN_ASP_TypeDefs** {TTCN_ASP_TypeDe}+ **$End_TTCN_ASP_TypeDefs**

124 TTCN_ASP_TypeDef ::= **$Begin_TTCN_ASP_TypeDef** ASP_Id PCO_Type [Comment] ASP_ParDcls [Comment] **$End_TTCN_ASP_TypeDef**

125 PCO_Type ::= **$PCO_Type** [PCO_TypeIdentifier]

/*SEMÁNTICA ESTÁTICA – PCO_TypeIdentifier deberá ser uno de los tipos de PCO utilizados en el formulario de declaración de PCO*/

/*SEMÁNTICA ESTÁTICA – Si sólo está definido un PCO en una sucesión de pruebas, PCO_TypeIdentifier es optativo*/

126 ASP_Id ::= **$ASP_Id** ASP_Id&FullId

127 ASP_Id&FullId ::= ASP_Identifier [FullIdentifier]

128 ASP_Identifier ::= Identifier

129 ASP_ParDcls ::= **$ASP_ParDcls** {ASP_ParDcl} **$End_ASP_ParDcls**

130 ASP_ParDcl ::= **$ASP_ParDcl** ASP_ParId ASP_ParType [Comment] **$End_ASP_ParDcl**

131 ASP_ParId ::= **$ASP_ParId** ASP_ParIdOrMacro

132 ASP_ParIdOrMacro ::= ASP_ParId&FullId | MacroSymbol

/*SEMÁNTICA ESTÁTICA – El MacroSymbol deberá utilizarse solamente en combinación con una referencia a un tipo estructurado*/

133 ASP_ParId&FullId ::= ASP_ParIdentifier [FullIdentifier]

134 ASP_ParIdentifier ::= Identifier

135 ASP_ParType ::= **$ASP_ParType** Type&Attributes

/*SEMANTICÁ ESTÁTICA – Type será un PredefinedType o TS_TypeIdentifier, un PDU_Identifier o una **PDU***/

A.3.3.5.4 *Definiciones de tipo ASP en ASN.1*

136 ASN1_ASP_TypeDefs ::= **$ASN1_ASP_TypeDefs** {ASN1_ASP_TypeDef} **$End_ASN1_ASP_TypeDefs**

137 ASN1_ASP_TypeDef ::= **$Begin_ASN1_ASP_TypeDef** ASP_Id PCO_Type [Comment] ASN1_TypeDefinition [Comment] **$End_ASN1_ASP_TypeDef**

A.3.3.5.5 *Definiciones de tipo ASP en ASN.1 por referencia*

138 ASN1_ASP_TypeDefsByRef ::= **$Begin_ASN1_ASP_TypeDefsByRef** {ASN1_ASP_TypeDefByRef} + [Comment] **$End_ASN1_ASP_TypeDefsByRef**

139 ASN1_ASP_TypeDefByRef ::= **$ASN1_ASP_TypeDefByRef** ASP_Id PCO_Type ASN1_TypeReference ASN1_ModuleId [Comment] **$End_ASN1_ASP_TypeDefByRef**

/*SEMÁNTICA ESTÁTICA – ASP_Id no será especificado con un FullIdentifier*/

A.3.3.5.6 *Definiciones de tipo PDU*

140 PDU_TypeDefs ::= **$PDU_TypeDefs** [TTCN_PDU_TypeDefs] [ASN1_PDU_TypeDefs] [ASN1_PDU_TypeDefsByRef] **$End_PDU_TypeDefs**

A.3.3.5.7 *Definiciones de tipo PDU en forma de tabla*

141 TTCN_PDU_TypeDefs ::= **$TTCN_PDU_TypeDefs** {TTCN_PDU_TypeDef}+ **$End_TTCN_PDU_TypeDefs**

142 TTCN_PDU_TypeDef ::= **$Begin_TTCN_PDU_TypeDef** PDU_Id PCO_Type [Comment] PDU_FieldDcls [Comment] **$End_TTCN_PDU_TypeDef**

/*SEMÁNTICA ESTÁTICA – Si una PDU se envía o se recibe solamente incrustada en ASPs dentro de la totalidad de la sucesión de pruebas, PCO_TypeIdentifier (en PCO_Type) es optativo*/

143 PDU_Id ::= **$PDU_Id** PDU_Id&FullId

144 PDU_Id&FullId ::= PDU_Identifier [FullIdentifier]

145 PDU_Identifier ::= Identifier

146 PDU_FieldDcls ::= **$PDU_FieldDcls** {PDU_FieldDcl}+ **$End_PDU_FieldDcls**

147 PDU_FieldDcl ::= **$PDU_FieldDcl** PDU_FieldId PDU_FieldType [Comment] **$End_PDU_FieldDcl**

148 PDU_FieldId ::= **$PDU_FieldId** PDU_FieldIdOrMacro

149 PDU_FieldIdOrMacro ::= PDU_FieldId&FullId I MacroSymbol

/*SEMÁNTICA ESTÁTICA – El MacroSymbol deberá utilizarse solamente en combinación con una referencia a un tipo estructurado*/

150 MacroSymbol ::= "<–"

151 PDU_FieldId&FullId ::= PDU_FieldIdentifier [FullIdentifier]

152 PDU_FieldIdentifier ::= Identifier

153 PDU_FieldType ::= **$PDU_FieldType** Type&Attributes

/*SEMÁNTICA ESTÁTICA – Type será un PredefinedType o TS_TypeIdentifier, un PDU_Identifier o una **PDU**/

154 Type&Aftributes ::= (Type [LengthAttribute]) | **PDU**

/*SEMÁNTICA ESTÁTICA – El conjunto de valores definidos por LengthAttribute será un subconjunto verdadero de los valores del tipo base*/

/*SEMÁNTICA ESTÁTICA – LenghtAttribute será proporcionado solamente cuando el tipo base sea un tipo cadena (es decir BITSTRING, HEXSTRING, OCTETSTRING o CharacterString), o se derive de un tipo cadena*/

155 LengthAttribute ::= SingleLength | RangeLength

156 SingleLength ::= "[" Bound "]"

157 Bound ::= Number | TS_ParIdentifier | TS_ConstIdentifier

/*SEMÁNTICA ESTÁTICA – Bound tomará un valor INTEGER no negativo o INFINITY*/

158 RangeLength ::= "[" LowerBound To UpperBound "]"

/*SEMÁNTICA ESTÁTICA – LowerBound será menor que UpperBound*/

159 LowerBound ::= Bound

160 UpperBound ::= Bound I **INFINITY**

A.3.3.5.8 *Definiciones de tipo PDU en ASN.1*

161 ASN1_PDU_TypeDefs ::= **$ASN1_PDU_TypeDefs** {ASN1_PDU_TypeDef} **$End_ASN1_PDU_TypeDefs**

162 ASN1_PDU_TypeDef ::= **$Begin_ASN1_PDU_TypeDef** PDU_Id PCO_Type [Comment] ASN1_TypeDefinition [Comment] **$End_ASN1_PDU_TypeDef**

/*SEMÁNTICA ESTÁTICA – Si una PDU se envía o se recibe solamente incrustada en ASP dentro de la totalidad de la sucesión de pruebas, PCO_TypeIdentifier (en PCO_Type) es optativo*/

A.3.3.5.9 *Definiciones de tipo PDU en ASN.1 por referencia*

163 ASN1_PDU_TypeDefsByRef ::= **$Begin_ASN1_PDU_TypeDefsByRef** {ASN1_PDU_TypeDefByRef}+ [Comment] **$End_ASN1_PDU_TypeDefsByRef**

164 ASN1_PDU_TypeDefByRef ::= **$ASN1_PDU_TypeDefByRef** PDU_Id PCO_Type ASN1_TypeReference ASN1_ModuleId [Comment] **$End_ASN1_PDU_TypeDefByRef**

/*SEMÁNTICA ESTÁTICA – Si una PDU se envía o se recibe o solamente incrustada en ASP dentro de la totalidad de la sucesión de pruebas, PCO_TypeIdentifier (en PCO_Type) es optativo*/

/*SEMÁNTICA ESTÁTICA – PDU_Id no se especificará en un FullIdentifier*/

A.3.3.5.10 *Definiciones de alias*

165 AliasDefs ::= **$Begin_AliasDefs** {AliasDef}+ [Comment] **$End_AliasDefs**

166 AliasDef ::= **$AliasDef** AliasId ExpandedId [Comment] **$End_AliasDef**

167 AliasId ::= **$AliasId** AliasIdentifier

168 AliasIdentifier ::= Identifier

/*SEMÁNTICA ESTÁTICA – Un AliasIdentifier se utilizará solamente en una línea de enunciado de una descripción de comportamiento*/

/*SEMÁNTICA ESTÁTICA – Un AliasIdentifier se utilizará solamente donde un ASP_Identifier o PDU_Identifier sea válido*/

169 ExpandedId ::= **$ExpandedId** Expansion

170 Expansion ::= ASP_Identifier | PDU_Identifier

A.3.4 *Parte constricciones*

A.3.4.1 *Generalidades*

171 ConstraintsPart ::= **$ConstraintsPart** [TS_TypeConstraints] [ASP_Constraints] [PDU_Constraints] **$End_ConstraintsPart**

A.3.4.2 *Declaraciones de constricciones de tipo de sucesión de pruebas*

172 TS_TypeConstraints ::= **$TS_TypeConstraints** [StructTypeConstraints] [ASN1_TypeConstraints] **$End_TS_TypeConstraints**

A.3.4.3 *Declaraciones de constricciones de tipo estructurado*

173 StructTypeConstraints ::= **$StructTypeConstraints** {StructTypeConstraint}+ **$End_StructTypeConstraints**

174 StructTypeConstraint ::= **$Begin_StructTypeConstraint** ConsId StructId DerivPath [Comment] ElemValues [Comment] **$End_StructTypeconstraint**

/*SEMÁNTICA ESTÁTICA – No deberá utilizarse el FullIdentifier que forma parte de Struct_Id*/

/*SEMÁNTICA ESTÁTICA – Una constricción modificada deberá tener la misma lista de parámetros que su constricción de base. En particular, no habrá parámetro omitidos de la lista o añadidos a ella*/

175 Elem Values ::= **$ElemValues** {ElemValue}+ **$End_ElemValues**

176 ElemValue ::= **$EemValue** ElemId ConsValue [Comment] **$End_ElemValue**

/*SEMÁNTICA ESTÁTICA – Los valores de elementos parametrizados en una constricción de base no serán modificados ni omitidos explícitamente en una constricción modificada*/

A.3.4.4 *Declaraciones de constricciones de tipo ASN.1*

177 ASN1_TypeConstraints ::= **$ASN1_TypeConstraints** {ASN1_TypeConstraint}+ **$End_ASN1_TypeConstraints**

178 ASN1_TypeConstraint ::= **$Begin_ASN1_TypeConstraint** ConsId ASN1_TypeId DerivPath [Comment] ASN1_ConsValue [Comment] **$End_ASN1_TypeConstraint**

/*SEMÁNTICA ESTÁTICA – No deberá utilizarse el FullIdentifier que forma parte de Struct_Id*/

/*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*/

A.3.4.5 *Declaraciones de constricciones de ASP*

179 ASP_Constraints ::= **$ASP_Constraints** [TTCN_ASP_Constraints] [ASN1_ASP_Constraints] **$End_ASP_Constraints**

A.3.4.6 *Declaraciones de constricciones de ASP en forma de tabla*

180 TTCN_ASP_Constraints ::= **$TTCN_ASP_Constraints** {TTCN_ASP_Constraint}+ **$End_TTCN_ASP_Constraints**

181 TTCN_ASP_Constraint ::= **$Begin_TTCN_ASP_Constraint** ConsId ASP_Id DerivPath [Comment] ASP_ParValues [Comment] **$End_TTCN_ASP_Constraint**

/*SEMÁNTICA ESTÁTICA – No deberá utilizarse el FullIdentifier que forma parte de ASP_Id*/

/*SEMÁNTICA ESTÁTICA – Si un ASP está subestructurado, las constricciones para ASP de ese tipo deberán tener la misma estructura*/

/*SEMÁNTICA ESTÁTICA – Para cada definición de tipo ASP se especificará por lo menos una constricción de base*/

/*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*/

182 ASP_ParValues ::= **$ASP_ParValues** {ASP_ParValue}+ **$End_ASP_ParValues**

183 ASP_ParValue ::= **$ASP_ParValue** ASP_ParId ConsValue [Comment] **$End_ASP_ParValue**

/*SEMÁNTICA ESTÁTICA – No deberá utilizarse el FullIdentifier que forma parte de ASP_Id*/

/*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 de 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 de ASP parametrizados en una constricción de base no serán modificados ni omitidos explícitamente en una constricción modificada*/

A.3.4.7 *Declaraciones de constricciones de ASP en ASN.1*

184 ASN1_ASP_Constraints ::= **$ASN1_ASP_Constraints** {ASN1_ASP_Constraint}+ **$End_ASN1_ASP_Constraints**

185 ASN1_ASP_Constraint ::= **$Begin_ASN1_ASP_Constraint** ConsId ASP_Id DerivPath [Comment] ASN1_ConsValue [Comment] **$End_ASN1_ASP_Constraint**

/*SEMÁNTICA ESTÁTICA – No deberá utilizarse 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*/

A.3.4.8 *Declaraciones de constricciones de PDU*

186 PDU_Constraints ::= **$PDU_Constraints** [TTCN_PDU_Constraints] [ASN1_PDU_Constraints] **$End_PDU_Constraints**

A.3.4.9 *Declaraciones de constricciones de PDU en forma de tabla*

187 TTCN_PDU_Constraints ::= **$TTCN_PDU_Constraints** {TTCN_PDU_Constraint}+ **$End_TTCN_PDU_Constraints**

188 TTCN_PDU_Constraint ::= **$Begin_TTCN_PDU_Constraint** ConsId PDU_Id DerivPath [Comment] PDU_FieldValues [Comment] **$End_TTCN_PDU_Constraint**

/*SEMÁNTICA ESTÁTICA – No deberá utilizarse 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 – Para cada definición de tipo PDU se especificará por lo menos una constricción de base*/

/*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*/

189 ConsId ::= **$ConsId ConsId&ParList**

190 ConsId&ParList ::= ConstraintIdentifier [FormalParList]

191 ConstraintIdentifier ::= Identifier

192 DerivPath ::= **$DerivPath** [DerivationPath]

193 DerivationPath ::= {ConstraintIdentifier Dot}+

/*SEMÁNTICA ESTÁTICA – Sin una definición de constricción es una modificación de una constricción existente, en la entrada de trayecto de derivación de la tabla 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 – La relación de las constricciones deberá figurar en el orden en que se aplican sus modificaciones a la constricción de base*/

/*SEMÁNTICA ESTÁTICA – No deberá haber espacios en blanco entre ConstraintIdentifier y el punto (Dot)*/

194 PDU_FieldValues ::= **$PDU_FieldValues** {PDU_FieldValue}+ **$End_PDU_FieldValues**

195 PDU_FieldValue ::= **$PDU_FieldValue** PDU_FieldId ConsValue [Comment] **$End_PDU_FieldValue**

/*SEMÁNTICA ESTÁTICA – No deberá utilizarse el FullIdentifier que forma parte de PDU_FieldId*/

/*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 estructuradas 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*/

196 ConsValue ::= **$ConsValue** ConstraintValue&Aftributes

/*SEMÁNTICA ESTÁTICA – ConsValue tomará el valor de un elemento del tipo especificado para el parámetro de ASP, campo de PDU o elemento de estructura*/

197 ConstraintValue&Aftributes ::= ConstraintValue ValueAttributes

/*SEMÁNTICA ESTÁTICA – ConstraintValue deberá cumplir todas las constricciones definidas para el parámetro de ASP, campo de PDU o tipo de elemento de estructura, incluyendo gamas de valores, listas de valores, constricciones de alfabeto y/o constricciones de longitud*/

/*SEMÁNTICA ESTÁTICA – Ninguna de las especificaciones de longitud definidas para el tipo de parámetro de ASP o campo de PDU en las declaraciones de tipo de prueba estará en contradicción con las especificaciones de longitud en la definición de tipo ASP o PDU*/

/*SEMÁNTICA ESTÁTICA – Ni las variables de sucesiones de prueba, ni las variables de casos de prueba deberán utilizarse en constricciones, a menos que se pasen como parámetros efectivos. En este último caso, estarán vinculadas a un valor y no serán cambiadas*/

198 ConstraintValue ::= ConstraintExpression | MatchingSymbol I ConsRef

/*SEMÁNTICA ESTÁTICA – LiteralValue, TS_ParIdentifier, TS_ConstIdentifier, TS_VarIdentifier, TC_VarIdentifier, ConsRef (una diferente) y FormalParIdentifier pueden pasarse como parámetros efectivos en ConsRef*/

199 ConstraintExpression ::= Expression

/*SEMÁNTICA ESTÁTICA – Los términos en ConstraintExpression sólo contendrán IValue, TS_ParIdentifier, TS_ConstIdentifier, FormalParIdentifier y OpCall*/

/*SEMÁNTICA ESTÁTICA – ConstraintExpression sólo tomará el valor de un elemento del tipo especificado*/

200 MatchingSymbol ::= Complement | Omit I AnyValue | AnyOrOmit | ValueList | ValueRange | SuperSet | SubSet | Permutation

/**Nota* – Si no hay símbolo de concordancia (*matching symbol*) se considera como un valor específico*/

201 Complement ::= **COMPLEMENT** ValueList

202 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*/

203 AnyValue ::= "?"

204 AnyOrOmit :: = "*"

205 ValueList ::= "(" ConstraintValue&Attributes {Comma ConstraintValue&Attributes} ")"

/*SEMÁNTICA ESTÁTICA – Cada uno de los ConstraintValue&Attributes será del tipo declarado para el parámetro de ASP, campo de PDU, o elemento de estructura en que se utilizó la ValueList*/

206 ValueRange ::.= "(" ValRange ")"

/*SEMÁNTICA ESTÁTICA – Se utilizará ValueRange solamente en parámetro de ASP, campo de PDU, o elemento de estructura de tipo INTEGER*/

/*SEMÁNTICA ESTÁTICA – El conjunto de valores definidos por ValueRange será un subconjunto verdadero de los valores permitidos por el tipo declarado del parámetro de ASP, campo de PDU o elemento de estructura*/

207 ValRange ::= (LowerRangeBound To UpperRangeBound)

/*SEMÁNTICA ESTÁTICA – LowerRangeBound será menor que UpperRangeBound*/

208 LowerRangeBound ::= ConstraintExpression | Minus **INFINITY**

/*SEMÁNTICA ESTÁTICA – ConstraintExpression tomará a un valor INTEGER específico*/

209 UpperRangeBound ::= ConstraintExpression | **INFINITY**

/*SEMÁNTICA ESTÁTICA – ConstraintExpression tomará a un valor INTEGER específico*/

210 SuperSet ::= **SUPERSET** "(" ConstraintValue&Attributes ")"

/*SEMÁNTICA ESTÁTICA – El argumento de SuperSet, es decir, ConstraintValue&Attributes, será de tipo SET OF*/

211 SubSet ::= **SUBSET** "(" ConstraintValue&Attributes ")"

/*SEMÁNTICA ESTÁTICA – El argumento de SubSet, es decir, ConstraintValue&Attributes, será de tipo SET OF*/

212 Permutation ::= **PERMUTATION** ValueList

/*SEMÁNTICA ESTÁTICA – Permutation deberá utilizarse solamente dentro de un valor de tipo SEQUENCE OF*/

/*SEMÁNTICA ESTÁTICA – ValueList será del tipo especificado en SEQUENCE OF*/

213 ValueAttributes ::= [ValueLength] [**IF_PRESENT**]

/*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*/

214 ValueLength ::= SingleValueLength | RangeValueLength

/*SEMÁNTICA ESTÁTICA – ValueLength deberá utilizarse solamente para parámetros de ASP, campos de PDU o elementos de estructura que están declarados como BITSTRING, HEXSTRING, OCTETSTRING, CharacterString, SEQUENCE OF o SET OF*/

/*SEMÁNTICA ESTÁTICA – ValueLength deberá utilizarse solamente en combinación con los siguientes mecanismos: Specificvalue, Complement, Omit, AnyValue, AnyOrOmit, AnyOrNone y Permutation*/

/*SEMÁNTICA ESTÁTICA – El conjunto de valores definidos por ValueLength será un subconjunto verdadero de los valores permitidos por el tipo declarado de parámetro de ASP, campo de PDU o elemento de estructura*/

215 SingleValueLength ::= "[" ValueBound "]"

216 ValueBound ::= Number | TS_ParIdentifier | TS_ConstIdentifier | FormalParIdentifier

/*SEMÁNTICA ESTÁTICA – ValueBound será menor que UpperValueBound*/

217 RangeValueLength ::= "[" LowerValueBound To UpperValueBound "]"

/*SEMÁNTICA ESTÁTICA – LowerValueBound será menor que UpperValueBound*/

218 LowerValueBound ::= ValueBound

219 UpperValueBound ::= ValueBound | **INFINITY**

A.3.4.10 *Declaraciones de constricciones de PDU en ASN.1*

220 ASN1_PDU_Constraints ::= **$ASN1_PDU_Constraints** {ASN1_PDU_Constraint}+ **$End_ASN1_PDU_Constraints**

221 ASN1_PDU_Constraint ::= **$Begin_ASN1_PDU_Constraint** Consld PDU_Id DerivPath [Comment] ASN1_ConsValue [Comment] **$End_ASN1_PDU_Constraint**

/*SEMÁNTICA ESTÁTICA – No deberá utilizarse el FullIdentifier que forma parte de PDU_Id*/

/*SEMÁNTICA ESTÁTICA – Si una PDU está subestructurada, las constricciones para PDU de ese tipo tendrán una estructura ASN.1 compatible (es decir, posiblemente con algunas agrupaciones)*/

/*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*/

222 ASN1_ConsValue ::= **$ASN1_ConsValue** ConstraintValue&AttributesOrReplace **$End_ASN1_ConsValue**

223 ConstraintValue&AttributesOrReplace ::= ConstraintValue&Attributes | Replacement {Comma Replacement}

224 Replacement ::= (**REPLACE** ReferenceList **BY** ConstraintValue&Attributes) | (**OMIT** ReferenceList)

/*SEMÁNTICA ESTÁTICA – Replacement se utilizará solamente cuando se especifica DerivPath*/

/*SEMÁNTICA ESTÁTICA – Los valores reemplazados parametrizados en una constricción de base no serán modificados ni omitidos explícitamente en una constricción modificada*/

225 ReferenceList ::= (ArrayRef | ComponentIdentifier | ComponentPosition) {ComponentReference}

A.3.5 *Parte dinámica*

A.3.5.1 *Generalidades*

226 DynamicPart ::= **$DynamicPart** TestCases [TestStepLibrary] [DefaultsLibrary] **$End_DynamicPart**

A.3.5.2 *Casos de prueba*

227 TestCases ::= **$TestCases** ({TestGroup | TestCase}+) **$End_TestCases**

228 TestGroup ::= **$TestGroup** TestGroupId {TestGroup | TestCase}+ **$End_TestGroup**

229 TestGroupId ::= **$TestGroupId** TestGroupIdentifier

230 TestGroupIdentifier ::= Identifier

231 TestCase ::= **$Begin_TestCase** TestCaseId TestGroupRef TestPurpose DefaultRef [Comment] BehaviourDescription [Comment] **$End_TestCase**

232 TestCaseId ::= **$TestCaseId** TestCaseIdentifier

233 TestCaseIdentifier ::= Identifier

234 TestGroupRef ::= **$TestGroupRef** TestGroupReference

235 TestGroupReference ::= [SuiteIdentifier "/"] {TestGroupIdentifier "/"}

/*SEMÁNTICA ESTÁTICA – No habrá espacios en blanco en ninguno de los dos lados de los caracteres"/"*/

236 TestPurpose ::= **$TestPurpose** BoundedFreeText

237 DefaultsRef ::= **$DefaultRef** [DefaultReference]

238 DefaultReference ::= DefaultIdentifier [ActualParList]

A.3.5.3 *Biblioteca (library) de pasos de prueba*

239 TestStepLibrary ::= **$TestStepLibrary** ({TestStepGroup | TestStep}+) **$End_TestStepLibrary**

240 TestStepGroup ::= **$TestStepGroup** TestStepGroupId {TestStepGroup | TestStep}+ **$End_TestStepGroup**

241 TestStepGroupId ::= **$TestStepGroupId** TestStepGroupIdentifier

242 TestStepGroupIdentifier ::= Identifier

243 TestStep ::= **$Begin_TestStep** TestStepId TestStepRef Objective DefaultsRef [Comment] BehaviourDescription [Comment] **$End_TestStep**

244 TestStepId ::= **$TestStepId** TestStepId&ParList

245 TestStepId&ParList ::= TestStepIdentifier [FormalParList]

246 TestStepIdentifier ::= Identifier

247 TestStepRef ::= **$TestStepRef** TestStepGroupReference

248 TestStepGroupReference ::= [SuiteIdentifier "/"] {TestStepGroupIdentifier "/"}

/*SEMÁNTICA ESTÁTICA – No habrá espacios en blanco en ninguno de los dos lados de los caracteres "/"*/

249 Objective ::= **$Objective** BoundedFreeText

A.3.5.4 *Biblioteca de supletorios*

250 DefaultsLibrary ::= **$DefaultsLibrary** ({DefaultGroup I Default}+) **$End_DefaultsLibrary**

251 DefaultGroup ::= **$DefaultGroup** DefaultGroupId {DefaultGroup | Default}+ **$End_DefaultGroup**

252 DefaultGroupId ::= **$DefaultGroupId** DefaultGroupIdentifier

253 Default ::= **$Begin_Default** DefaultId DefaultRef Objective [Comment] BehaviourDescription [Comment] **$End_Default**

/*SEMÁNTICA ESTÁTICA – BehaviourDescription consistirá en un árbol único (es decir, no habrá árboles locales)*/

/*SEMÁNTICA ESTÁTICA – BehaviourDescription no utilizará adjunción (*attachment*) de árbol (es decir, los árboles de comportamiento supletorios no adjuntarán pasos de prueba)*/

/*SEMÁNTICA ESTÁTICA – Se asignará un veredicto final a cada hoja de un árbol supletorio. Si el veredicto final es el resultado de un enunciado OTHERWISE en el árbol supletorio, el veredicto deberá ser FAIL*/

254 DefaultRef ::= **$DefaultRef** DefaultGroupReference

255 DefaultId ::= **$DefaultId** DefaultId&ParList

256 DefaultId&ParList ::= DefaultIdentifier [FormalParList]

257 DefaultIdentifier ::= Identifier

258 DefaultGroupReference ::= [SuiteIdentifier "/"] {DefaultGroupIdentifier "/"}

/*SEMÁNTICA ESTÁTICA – No habrá espacios en blanco en ninguno de los dos lados de los caracteres "/"*/

259 DefaultGroupIdentifier ::= Identifier

A.3.5.5 *Descripciones de comportamiento*

260 BehaviourDescription ::= **$BehaviourDescription** RootTree {LocalTree} **$End_BehaviourDescription**

261 RootTree ::= {BehaviourLine}+

262 LocalTree ::= Header {BehaviourLine}+

263 Header ::= **$Header** TreeHeader

264 TreeHeader ::= TreeIdentifier [FormalParList]

265 TreeIdentffier ::= Identifier

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

267 FormalPar&Type ::= FormalParIdentifier {Comma FormalParIdentifier} Colon FormalParType

268 FormalParIdentifier ::= Identifier

269 FormalParType ::= Type | PCO_TypeIdentifier | **PDU**

/*SEMÁNTICA ESTÁTICA – Si un parámetro formal de un paso de prueba es de tipo **PDU**, en el árbol de comportamiento de paso de prueba no se hará referencia a campos específicos de la PDU*/

A.3.5.6 *Líneas de comportamiento*

270 BehaviourLine ::= **$BehaviourLine** LabelId Line Cref VerdictId [Comment] **$End_BehaviourLine**

271 Line ::= **$Line** Indentation StatementLine

272 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 sagrado*/

273 LabelId ::= **$LabelId** [Label]

274 Label ::= Identifier

275 Cref ::= **$Cref** [ConstraintReference]

276 ConstraintReference ::= ConsRef | FormalParIdentifier

/*SEMÁNTICA ESTÁTICA – ConsRef deberá estar presente conjuntamente con SEND, IMPLICIT SEND y RECEIVE. No se necesita una ConstraintReference para ASP que no tengan parámetros. No estará presente con ninguna otra categoría de enunciado en TTCN*/

/*SEMÁNTICA ESTÁTICA – FormalParIdentifier deberá resolver a una ConsRef*/

/*SEMÁNTICA ESTÁTICA – LiteralValue, TS_ParIdentifier, TS_ConstIdentifier, TS_VarIdentifier, TC_VarIdentifier, ConsRef y FormalParIdentifier pueden pasarse como parámetros efectivos a una constricción en una ConstraintReference formada a partir de una descripción de comportamiento*/

/*SEMÁNTICA ESTÁTICA – /* ConstraintReferences sobre eventos SEND no incluirán comodines (*wildcards*), a menos que se asignen a éstos, explícitamente, valores específicos en la línea de evento SEND*/

277 ConsRef ::= ConstraintIdentifier [ActualCrefParList]

278 ActualCrefParList ::= "(" ActualCrefPar {Comma ActualCrefPar} ")"

/*SEMÁNTICA ESTÁTICA – Véase semántica estática en producción 299*/

279 ActualCrefPar ::= Value | DataObjectIdentifier | ConsRef

280 VerdictId ::= **$VerdictId** [Verdict]

281 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*/

282 Pass ::= **PASS** | **P** | "(" **PASS** ")" | "(" **P** ")"

283 Fail ::= **FAIL** | **F** | "(" **FAIL** ")" | "(" **F** ")"

284 Inconclusive ::= **INCONC** | **I** | "(" **INCONC** ")" | "(" **I** ")"

285 Result ::= Identifier

/*SEMÁNTICA ESTÁTICA – Result sólo será el identificador predefinido R*/

/*SEMÁNTICA ESTÁTICA – No se utilizará R en el lado izquierdo de una asignación*/

A.3.5.7 *Enunciados en TTCN*

286 StatementLine ::= (Event [Qualifier] [AssignmentList] [TimerOps]) | (Qualifier [AssignmentList] [TimerOps]) | (AssignmentList [TimerOps]) | TimerOps | Construct | ImplicitSend

287 Event ::= Send | Receive | Otherwise | Timeout

/*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*/

288 Qualifier ::= "[" Expression "]"

/*SEMÁNTICA ESTÁTICA – Qualifier tomará a un valor BOOLEAN específico*/

289 Send ::= [PCO_Identifier | FormalParIdentifier] "|" (ASP_Identifier | PDU_ Identifier)

/*SEMÁNTICA ESTÁTICA – PCO_Identifier o FormalParIdentifier estarán presentes si la sucesión de pruebas utiliza más de un PCO*/

/*SEMÁNTICA ESTÁTICA – FormalParIdentifier resolverá a un PCO_Identifier. Este deberá estar presente si la sucesión de pruebas utiliza más de un PCO*/

290 ImplicitSend ::= "<" **IUT** "|" (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*/

291 Receive ::= [PCO_Identifier | FormalParIdentifier] "?" (ASP_Identifier | PDU_Identifier)

/*SEMÁNTICA ESTÁTICA – PCO_Identifier o FormalParIdentifier estará presente si la sucesión de pruebas utiliza más de un PCO*/

292 Otherwise ::= [PCO_Identifier | FormalParIdentifier] "?" **OTHERWISE**

/*SEMÁNTICA ESTÁTICA – PCO_Identifier o FormalParIdentifier estará presente si la sucesión de pruebas utiliza más de un PCO*/

293 Timeout ::= "?" **TIMEOUT** [TimerIdentifier]

294 Construct ::= GoTo | Attach | Repeat

295 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 es 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*/

/*SEMÁNTICA ESTÁTICA – Cada una de las etiquetas utilizadas en un constructivo GoTo deberán encontrarse dentro del paso de prueba en el que se utilizó el GoTo*/

/*SEMÁNTICA ESTÁTICA – No se efectuará un GoTo al primer nivel de alternativas de árboles locales, pasos de prueba o valores por defecto*/

296 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 – LiteralValue, TS_ParIdentifier, TS_ConstIdentifier,TS_VarIdentifier, TC_VarIdentifier, ConstraintIdentifier y PCO pueden pasarse como parámetros reales a un árbol adjuntado*/

297 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, ConstraintIdentifier y PCO pueden pasarse como parámetros reales al árbol en un enunciado REPEAT*/

298 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*/

299 ActualParList ::= "(" ActualPar {Comma ActualPar} ")"

/*SEMÁNTICA ESTÁTICA – El número de parámetros reales será igual al número de parámetros formales*/

/*SEMÁNTICA ESTÁTICA – Cada parámetro real resolverá a un valor específico compatible con el tipo de su parámetro formal correspondiente*/

/*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 estarán vinculados (*bound*)*/

/*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*/

300 ActualPar ::= Value | PCO_Identifier

A.3.5.8 *Expressions*

301 AssignmentList ::= "(" Assignment {Comma Assignment} ")"

302 Assignment ::= DataObjectReference ":=" Expression

/*SEMÁNTICA ESTÁTICA – El lado izquierdo de Assignment sólo resolverá a: TS_VarIdentifier, TC_VarIdentifier, referencia al campo de una variable o referencia a un parámetro de ASP o campo de PDU que vaya a enviarse*/

/*SEMÁNTICA ESTÁTICA – Una expresión no contendrá variables no vinculadas*/

/*SEMÁNTICA ESTÁTICA – La expresión en el lado derecho de Assignment tomará un valor explícito del tipo del lado izquierdo */

303 Expression ::= SimpleExpression [RelOp SimpleExpression]

/*SEMÁNTICA ESTÁTICA – Si existen SimpleExpressions y RelOp, las SimpleExpressions tomarán valores específicos de tipos compatibles*/

/*SEMÁNTICA ESTÁTICA – 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*/

304 SimpleExpression ::= Term {AddOp Term}

/*SEMÁNTICA ESTÁTICA – 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*/

305 Term ::= Factor {MultiplyOp Factor}

/*SEMÁNTICA ESTÁTICA – 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*/

306 Factor ::= [UnaryOp] Primary

/*SEMÁNTICA ESTÁTICA – 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*/

307 Primary ::= Value | DataObjectReference | OpCall | SelectExprIdentifier | "(" Expression ")"

/*SEMÁNTICA ESTÁTICA – SelectExprIdentifier sólo se utilizará dentro de expresiones de selección*/

308 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 de ASP y de campo PDU en la propia línea de enunciado*/

/*SEMÁNTICA ESTÁTICA – Cada ComponentReference sólo hará referencia a un parámetro de ASP, campo de PDU, elemento de estructura o valor ASN.1 declarado explícitamente en el objeto que precede inmediatamente en la DataObjectReference*/

309 DataObjectIdentifier ::= TS_ParIdentifier | TS_ConstIdentifier | TS_VarIdentifier | TC_VarIdentifier | FormalParIdentifier | ASP_Identifier | PDU_Identifier

310 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*/

311 RecordRef ::= Dot (ComponentIdentifier | PDU_Identifier | StructIdentifier | 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. StructIdentfier y PDU-Identifiers se incluirán explícitamente en una RecordRef cada vez que 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 al 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*/

312 ComponentIdentifier ::= ASP_ParIdentifier | PDU_FieldIdentifier | ElemIdentifier | ASN1_Identifier

313 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*/

314 ComponentPosition ::= "(" Number ")"

315 ArrayRef ::= Dot "[" ComponentNumber "]"

316 ComponentNumber ::= Expression

/*SEMÁNTICA ESTÁTICA – ComponentNumber tomará a un valor INTEGER específico no negativo*/

317 BitRef ::= Dot (BitIdentifier | "[" BitNumber "]")

318 BitIdentifier ::= Identifier

/**Nota* – BitIdentifier identifica un bit particular dentro de una BIT STRING de la ASN.1*/

319 BitNumber ::= Expression

/*SEMÁNTICA ESTÁTICA – BitNumber tomará a un valor INTEGER específico no negativo*/

320 OpCall ::= TS_OpIdentifier (ActualParList | "(" ")")

/*SEMÁNTICA ESTÁTICA – Véase semántica estática en producción 299*/

321 AddOp ::= "+" | "–" | **OR**

/*SEMÁNTICA ESTÁTICA – 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*/

322 MultiplyOp ::= "*" | "/" | **MOD** | **AND**

/*SEMÁNTICA ESTÁTICA – Los operandos de los operadores "*", "/" y MOD eran del tipo INTEGER (es decir, predefinidos en TTCN o ASN.1) o derivaciones de INTEGER (es decir, subgrama). Los operandos del operador AND serán del tipo BOOLEAN (predefinidos en TTCN o ASN.1) o derivaciones de BOOLEAN*/

323 UnaryOp ::= "+" | "–" | **NOT**

/*SEMÁNTICA ESTÁTICA – 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*/

324 RelOp ::= "=" | "<" | ">" | "<>" | ">=" | "<="

A.3.5.9 *Operaciones de temporizadores*

325 TimerOps ::= TimerOp {Comma TimerOp}

326 TimerOp ::= StartTimer I CancelTimer I ReadTimer

327 StartTimer ::= **START** TimerIdentifier ["("TimerValue ")"]

328 CancelTimer ::= **CANCEL** [TimerIdentifier]

329 TimerValue ::= Expression

/*SEMÁNTICA ESTÁTICA – Timervalue tomará un valor INTEGER positivo no nulo*/

330 ReadTimer ::= **READTIMER** TimerIdentifier "(" DataObjectReference ")"

/*SEMÁNTICA ESTÁTICA – La DataObjectReference resolverá solamente a TS_VarIdentifier, TC_VarIdentifier, referencia al campo de una variable o referencia a un parámetro de ASP o campo de PDU que vaya a enviarse*/

/*SEMÁNTICA ESTÁTICA – La DataObjectReferece resolverá a tipo INTEGER*/

A.3.6 *Tipos*

A.3.6.1 *Generalidades*

331 Type ::= PredefinedType | ReferenceType

A.3.6.2 *Tipos predefinidos*

332 PredefinedType ::= **INTEGER** I **BOOLEAN** I **BITSTRING** I **HEXSTRING** I **OCTETSTRING** I CharacterString

333 CharacterString ::= **NumericString** I **PrintableString** I **TeletexString** I **VideotexString** I **VisibleString** I **IA5String** I **GraphicString** I **GeneralString**

A.3.6.3 *Tipos referenciados*

334 ReferenceType ::= TS_TypeIdentifier | ASP_Identifier | PDU_Identifier

/*SEMÁNTICA ESTÁTICA – Todos los tipos, con excepción de los tipos predefinidos, utilizados en una sucesión de pruebas, deberán ser declarados en las definiciones de tipo de caso de prueba, en las definiciones de tipo ASP, o en las definiciones de tipo PDU, y ser referenciados por nombre*/

335 TS_TypeIdentifier ::= SimpleTypeIdentifier | StructIdentifier | ASN1_TypeIdentifier

A.3.7 *Valores*

336 Value ::= LiteralValue | ASN1_Value

/*REFERENCE – Donde ASN1_Value es Value, tal como está definido en la Recomendación X.208*/

/*En la Recomendación X.208, la producción DefinedValue se define como: DefinedValue ::= Externalvaluereference|valuereference. A los efectos de la TTCN, esta producción se redefine de la manera siguiente: DefinedValue ::= ConstraintValue&Attributes. Obsérvese que, de esta forma, no se permiten referencias externas en TTCN*/

/*SEMÁNTICA ESTÁTICA – En expresiones aritméticas, no se utilizarán valores denominados de ASN.1 como operandos de operaciones*/

337 LiteralValue ::= Number | BooleanValue | Bstring | Hstring | Ostring | Cstring

338 Number ::= (NonZeroNum {Num}) | **0**

339 NonZeroNum ::= **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

340 Num ::= **0** | NonZeroNum

341 BooleanValue ::= **TRUE** | **FALSE**

342 Bstring ::= "'" {Bin I Wildcard} "'" **B**

343 Bin ::= **0** | **1**

344 Hstring ::= "'" {Hex I Wildcard} "'" **H**

345 Hex ::= Num | **A** | **B** | **C** | **D** | **E** | **F**

346 Ostring ::= "'" {Oct I Wildcard} "'" **O**

347 Oct ::= Hex Hex

348 Cstring ::= "'" {Char I Wildcard I "\"} "'"

349 Char ::= /* REFERENCIA – Un carácter definido por el tipo cadena de caracteres pertinente*/

/*SEMÁNTICA ESTÁTICA – Si el tipo CharacterString incluye el carácter " (comillas), este carácter será representado por un par de " (comillas) en la denotación de cualquier valor*/

350 Wildcard ::= AnyOne | AnyOrNone

351 AnyOne ::= "?"

/*SEMÁNTICA ESTÁTICA – AnyOne se utilizará solamente dentro de valores de tipo cadena, SEQUENCE OF y SET OF*/

352 AnyOrNone ::= "*"

/*SEMÁNTICA ESTÁTICA – AnyOrNone se utilizará solamente dentro de valores de tipo cadena, SEQUENCE OF y SET OF*/

353 Identifier ::= Alpha{AlphaNum | Underscore}

/*SEMÁNTICA ESTÁTICA –Todos los identificadores a los que se hace referencia en una sucesión de pruebas deberán ser declarados explícitamente en la sucesión de pruebas, declarados explícitamente en una definición de tipo ASN.1 referenciada por la sucesión de pruebas o ser un identificador predefinido en TTCN*/

354 Alpha ::= UpperAlpha | LowerAlpha

355 AlphaNum ::= Alpha | Num

356 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**

357 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**

358 ExtendedAlphaNum ::=/* REFERENCIA – Un carácter perteneciente a cualquier conjunto de caracteres definido en ISO/IEC 10646*/

359 BoundedFreeText ::= "/*" FreeText "*/"

360 FreeText ::= {ExtendedAlphaNum}

/*SEMÁNTICA ESTÁTICA – FreeText no contendrá la cadena "*/" a menos que vaya precedida por barra inclinada inversa ("\")*/

A.3.8 *Producciones diversas*

```
361  Comma ::= ","

362  Dot ::= "."

363  Dash ::= "–"

364  Minus ::= "–"

365  SemiColon ::= ";"

366  Colon ::= ":"

367  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 del § A.4 se especifican los demás requisitos de semántica estática comunes a la TTCN.GR y la TTCN.MP. En el § A.5.2 se especifican las semánticas estáticas adicionales de la TTCN.MP.

A.4.2 *Exclusividad de los identificadores*

A.4.2.1 En algunos casos, las sucesiones de pruebas pueden hacer referencia a elementos definidos en otras Recomendaciones sobre OSI. En especial, en las definiciones de tipo pueden hacerse referencias a módulos de definición de tipo en ASN.1, acordes con la Recomendación X.208. En todas las sucesiones de pruebas pueden utilizarse 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:

a) Las referencias de tipo y los identificadores de módulo de los diversos cuadros de definiciones de tipo en ASN.1 cumplirán los requisitos establecidos para los identificadores en la Recomendación X.208.

b) En los identificadores utilizados en otras partes de una sucesión de pruebas se sustituirán los caracteres guión (-) por caracteres de subrayado (_).

En algunos de los cuadros de TTCN puede utilizarse parte de la sintaxis ASN.1 para definir tipos. En esos casos, las reglas de ASN.1 deberán ir 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 sucesiones de pruebas de TTCN los demás requisitos definidos por la Recomendación X.208 (por ejemplo, los identificadores de tipo comenzarán con una letra mayúscula y los identificadores de campo dentro de las definiciones estructuradas en ASN.1 comenzarán con una letra minúscula).

A.4.2.2 Los identificadores de objetos de TTCN que se indican a continuación serán exclusivos a lo largo de la sucesión de pruebas:

a) tipos de sucesión de pruebas;

b) operaciones de sucesiones de pruebas;

c) parámetros de sucesiones de pruebas;

d) expresiones de selección de casos de prueba;

e) constantes de sucesiones de pruebas;

f) variables de sucesiones de pruebas;

g) variables de caso de prueba;

h) tipos de PCO;

i) puntos de control y observación (PCO);

j) temporizadores;

k) tipos ASP;

l) tipos PDU;

m) tipos estructurados;

n) alias;

o) constricciones de ASP;

p) constricciones de PDU;

q) constricciones de estructura;

r) casos de prueba;

s) pasos de prueba;

t) valores por defecto.

A.4.2.3 Las referencias de objetos de TTCN que se indican a continuación, serán exclusivas a lo largo de la sucesión de pruebas:

a) referencias de grupo de pruebas;

b) referencias de grupo de pasos de prueba;

c) referencias de grupo de valores por defecto.

A.4.2.4 En el cuadro A-2 se indican las palabras clave de la TTCN se indican y en el cuadro A-3, los identificadores predefinidos. Estas palabras clave y estos identificadores predefinidos son palabras reservadas y no podrán utilizarse como identificadores en una sucesión de pruebas de TTCN. Las palabras clave de la TTCN, los identificadores predefinidos (tipos predefinidos, operaciones, variables, valores) y los identificadores de la TTCN son sensibles al tipo de letras mayúscula o minúscula.

CUADRO A-2/X.292

Palabra clave de la TTCN

AND	ms	REPLACE
BY	NOT	s
CANCEL	ns	START
F	OMIT	SUPERSET
FAIL	OR	SUBSET
GOTO	OTHERWISE	TIMEOUT
I	P	TO
IF_PRESENT	PASS	UNTIL
INCONC	PDU	µs
IUT	PERMUTATION	UT
LT	ps	
min	READTIMER	
MOD	REPEAT	

CUADRO A-3/X.292

Identificadores predefinidos de la TTCN

BITSTRING	inconc	NumericString
BOOLEAN	INFINITY	OCTETSTRING
BIT_TO_INT	INTEGER	pass
fail	INT_TO_HEX	PrintableString
FALSE	INT_TO_BIT	R
GeneralString	IS_CHOSEN	TRUE
GraphicString	IS_PRESENT	TeletexString
HEXSTRING	LENGTH_OF	VideotexString
HEX_TO_INT	none	VisibleString
IA5String	NUMBER_OF_ELEMENTS	

A.4.2.5 En el cuadro A-4, se indican las palabras reservadas de la ASN.1. Estas palabras reservadas no podrán utilizarse como identificadores en una sucesión de pruebas en TTCN.

CUADRO A-4/X.292

Palabras reservadas en la ASN.1

ABSENT	FROM	PRESENT
ANY	GeneralStringGeneralizedTime	PRIVATE
APPLICATION	GraphicString	PrintableString
BEGIN	IA5String	REAL
BIT	IDENTIFIER	SEQUENCE
BOOLEAN	IMPLICIT	SET
CHOICE	IMPORT	SIZE
COMPONENT	INCLUDES	STRING
COMPONENTS	INTEGER	T61String
DEFAULT	ISO646String	TRUE
DEFINED	MAX	TeletexString
DEFINITIONS	MIN	UNIVERSAL
END	NULL	UTCTime
ENUMERATED	NumericString	VideotexString
EXPLICIT	OBJECT	VisibleString
EXPORT	OCTET	WITH
EXTERNAL	OF	
FALSE	OPTIONAL	

A.4.2.6 Cuando se utilice ASN.1 en una sucesión de pruebas en TTCN, los identificadores de ASN.1 de la lista anterior serán exclusivos en toda la sucesión de pruebas, con independencia de si la definición en la ASN.1 es explícita o implícita por referencia a:

a) *TypeIdentifiers* de una definición de tipo ASN.1;

b) identificadores que aparecen en un tipo ENUMERATED ASN.1 como valores distinguidos;

c) identificadores que aparecen en una *NamedNumberList* de un tipo INTEGER ASN.1

A.4.2.7 Los nombres de los parámetros de una ASP serán exclusivos dentro de la ASP en la que se hayan declarado. Los nombres de los campos de una PDU serán exclusivos dentro de la PDU en la que se hayan declarado.

A.4.2.8 Si se utiliza un tipo estructurado como expansión de macro, los nombres de los elementos dentro del tipo estructurado serán exclusivos dentro de cada ASP o PDU en la que se hayan expandido.

A.4.2.9 Las etiquetas utilizadas dentro de un árbol serán exclusivas dentro de ese árbol (a saber, un árbol raíz de caso de prueba, un árbol de paso de prueba, un árbol por defecto, o un árbol local).

A.4.2.10 El identificador de encabezamiento de árbol utilizado en los árboles locales será exclusivo dentro de la descripción de comportamiento dinámico donde éstos aparecen y no será el mismo que cualquier otro identificador que tenga un significado exclusivo en la sucesión de pruebas.

Nota – Esto significa que un identificador de árbol local puede tener el mismo nombre que un identificador de árbol local en otra descripción de comportamiento, pero no el mismo en otro paso de prueba de la biblioteca de pasos de prueba.

A.4.2.11 Los nombres de parámetros formales que pueden aparecer facultativamente como parte de lo que sigue, serán exclusivos dentro de la lista de parámetros formales y no podrán ser iguales a cualquier otro identificador que tenga un significado exclusivo en la sucesión de pruebas:

a) definición de operaciones de sucesión de pruebas;

b) encabezamiento de árbol de un árbol local;

c) identificador de paso de prueba;

d) identificador de valor por defecto;

e) declaración de constricción parametrizada.

A.4.2.12 Un nombre de parámetro formal contenido en la lista de parámetros formales de un encabezamiento de árbol local tendrá precedencia sobre un nombre de parámetro formal contenido en la lista de parámetros formales del paso de prueba en el que se haya definido, dentro del ámbito de esa lista de parámetros formales.

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 sucesión, 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 pruebas 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 caso de pruebas y en las referencias de grupo de prueba de las descripciones de comportamiento de caso de prueba.

e) La estructura de sucesiones de prueba, el índice de caso de prueba, el índice de paso de prueba y los cuadros de índices de valores por defecto requieren un número de página para cada inscripción. 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.MP 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.

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 sucesión de pruebas aparece en forma de identificadores de grupos de pruebas 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 sucesión de pruebas aplicable a los grupos de pruebas y definida por el índice de caso de prueba.

ANEXO B

(a la Recomendación X.292)

Semántica operacional de la TTCN

(Este anexo es parte integrante de la presente Recomendación)

B.1 *Introducción*

En este anexo se describe la definición de la semántica de la TTCN. Dicha definición se efectúa en dos fases. En la primera fase se establece la correspondencia entre textos en TTCN concretos y una estructura simplificada, que puede ser manipulada en el curso de la segunda fase. La segunda fase puede contemplarse como la descripción de una máquina de TTCN que interpreta esos textos en TTCN simplificados.

La primera fase puede descomponerse en tres pasos.

a) *Definición de sintaxis*

a un lenguaje que carezca de gramática fija (independiente del contexto) no puede asignársele ninguna semántica. En el anexo A se describe la sintaxis de la TTCN mediante reglas de producción BNF.

b) *Definición de semántica estática*

la semántica estática define los textos generados por la gramática independiente del contexto que tienen sentido (por ejemplo, deberán declararse los PCO utilizados, las constricciones a las que se hace referencia desde la parte comportamiento deberán estar presentes en la parte constricción de la serie de pruebas, etc). En el anexo A se especifican los requisitos de semántica estática de la TTCN;

c) *Definición de transformación de árbol*

en este tercer paso, se define la construcción de un árbol de evaluación abstracto a partir de un texto en TTCN concreto, que es correcto tanto desde el punto de vista de la sintaxis como de la semántica estática. En el § B.4 se describen los algoritmos de transformación.

En la segunda fase se define la máquina de TTCN abstracta. Es poco probable que puedan construirse alguna vez tal tipo de máquinas con la arquitectura descrita. En la definición de la semántica dinámica u operacional, no se han contemplado los problemas de realización. En el § B.5 se describe la máquina de TTCN abstracta. La parte principal de la máquina de TTCN es un proceso denominado EVALUATE_TEST_CASE que interpreta el árbol de evaluación abstracto. El árbol de evaluación abstracto es un parámetro de este proceso.

B.2 *Precedencia*

En los puntos que siguen se describe la semántica operacional de la TTCN con dos notaciones alternativas, que permiten al lector la elección de un método basado en su preferencia personal: 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. No obstante, en tal caso, el seudocódigo tendrá precedencia sobre el lenguaje natural, a reserva de la corrección que efectúe la organización de normalización.

B.3 *Procesamiento de errores de caso de prueba*

En el texto principal de esta Recomendación y en el presente anexo se describen condiciones que dan por resultado la detección de errores de caso de prueba. Cuando se detecte un error de caso de prueba, su ocurrencia deberá inscribirse en el registro cronológico de conformidad (*conformance log*).

B.4 *Algoritmos de transformación*

B.4.1 *Introducción*

En esta subcláusula se describe la forma de construir un árbol de evaluación abstracto a partir de un caso de prueba en TTCN que es correcto desde el punto de vista de la sintaxis y de semántica estática. El estudio se efectúa con arreglo a los siguientes pasos:

a) adición (*appending*) de comportamientos por defecto;

b) supresión (*removal*) de constructivos REPEAT;

c) expansión de árboles adjuntados.

Las transformaciones se describen mediante algoritmos en un seudolenguaje de programación. Además, se da una descripción en lenguaje natural para explicar el funcionamiento de los algoritmos.

B.4.2 *Adición de comportamiento por defecto*

La adición real de valores por defecto se efectúa añadiendo el constructivo «+DefaultReference» al final de cada conjunto de alternativas del caso de prueba. En el seudocódigo que sigue, Level (nivel) representa las alternativas del árbol de comportamiento vigente. Si A_i es una alternativa de un conjunto de M alternativas del árbol de comportamiento de que se trate, Level es el conjunto ordenado (A_1, A_2 ..., A_m) en el que cada una de las A_i es un evento, un seudoevento o un constructivo. Con cada A_i hay asociado un nivel inferior de cero o más alternativas, y en consecuencia, cada A_i representa un subárbol del árbol original.

```
• function APPEND_DEFAULT (TestCaseOrStep,Tree,Level): BOOLEAN

  begin
  if (TestCaseOrStep.DefaultReference <> empty) then
      begin

    (*Añadir Default al primer nivel del primer árbol de caso de prueba*)


     if ((TestCaseOrStep is a TestCase)) and
        (Tree=ROOT_TREE(TestCaseOrStep) and
        (Level=FIRST_LEVEL(Tree))) then
        APPEND('+' TestCaseOrStep.DefaultReference, Level);

    (*Añadir Default a todos los niveles por debajo del primer nivel*)

          if (Level <>  FIRST_LEVEL(Tree)) then
                APPEND('+' TestCaseOrStep.DefaultReference, Level)
       end;
     RETURN(TRUE);
  end
```

Se puede realizar la adición de todos los valores por defecto de un caso de prueba o de un paso de prueba recorriendo el árbol de comportamiento en su totalidad y utilizando APPEND_DEFAULT en cada nivel de alternativas. Para ello hay que seguir el procedimiento recursivo que a continuación se indica, invocado con TestCaseOrStep:= TestCase, Tree:=ROOT_TREE(TestCase), y Level:=FIRST_LEVEL(Tree):

- **procedure APPEND_TRAV** (TestCaseOrStep, Tree, Level)

```
begin
if (Am in Level is not '+' TestCaseOrStep.DefaultReference) then
    begin (*Hacerlo si aún no se han añadido valores por defecto Defaults*)
        for (every alternative Ai in Level) do

        begin
            if (Ai is not a leaf) then

            begin
                    NextLevel:=NEXT_LEVEL(Ai);
                    APPEND_TRAV(TestCaseOrStep, Tree, NextLevel)

            end
            if (Ai is an ATTACH construct) then
                if (Ai.AttachedTree is TestStepIdentifier) then

            begin
                    NextTestStep:=Ai.AttachedTree;
                    NextTree:=ROOT_TREE(NextTestStep);
                    NextLevel:=FIRST_LEVEL(NextTree);
                    APPEND_TRAV(NextStep, NextTree, NextLevel);

            end
            else (*El árbol adjuntado es un Treeldentifier*)

            begin
                    NextTestTree:=Ai.AttachedTree;
                    NextLevel:=FIRST_LEVEL(NextTree);
                    APPEND_TRAV(TestCaseOrStep, NextTree, NextLevel)

            end

        end
    APPEND_DEFAULT(TestCaseOrStep, NextTree, NextLevel)

end
end
```

B.4.3 *Supresión de constructivos REPEAT*

Si *TreeAndParameters* indica un *TreeIdentifier* particular seguido de *ActualPARList*, y la condición señala una expresión booleana particular, REPEAT *TreeAndParameters* UNTIL [*Condición*], puede sustituirse por:

```
                [TRUE]
label_A           [TRUE]
                    + TreeAndParameters
                      [NOT (condition)]
                            –> label_A
                      [condition]
                            :
```

Tras esta expresión siguen unas líneas que describen el comportamiento subsiguiente del constructivo REPEAT con un sangrado adicional de dos niveles.

B.4.4 *Expansión de árboles adjuntados*

Los árboles adjuntados se expanden sustituyendo el constructivo de adjunción +*TestStep* por el árbol *TestStep* y, si existe un comportamiento especificado que le continúa y está sangrado desde él, se utiliza a continuación el constructivo ATTACH para insertar este comportamiento después de, y sangrado desde, cada hoja del árbol adjuntado.

Se recomienda que cuando se evalúe la semántica de un árbol de comportamiento al que se hayan añadido valores por defecto, con el constructivo REPEAT suprimido, se expandan los árboles adjuntados en cada nivel (conjunto de alternativas), ya que ese nivel se alcanza al ejecutar el caso de prueba. Los árboles adjuntados en Level se expanden utilizando el siguiente procedimiento:

- **procedure EXPAND_LEVEL** (Level)

```
begin
    for (every alternative Ai in Level) do
    begin
        if (Ai is an ATTACH construct) then
        begin
                Subsequent:=SUBSEQUENT_BEHAVIOUR_TO(Ai);
                EXPAND(Ai.AttachedTree, Subsequent, Ai, Level)
        end
    end
end
```

- **procedure EXPAND** (SubTree, Subsequent, Alternative, Level)

```
begin
REPLACE_PARAMETERS(Alternative, SubTree);
```

(*Esto sustituye los parámetros formales de Subtree por los parámetros formados especificados en la lista de parámetros reales de Alternative, mediante sustitución textual*)

```
for (every leaf Li in SubTree) do APPEND_SUBSEQUENT_BEHAVIOUR_TO(Li, Subsequent);

New_Level:=FIRST_LEVEL(SubTree);
Level:=REPLACE(Level, New_Level, Alternative)
```

(*Esto sustituye Alternative de Level por la matriz New_Level; por ejemplo, REPLACE((A,B,C), (X,Y,Z), B) da: (A,X,Y,Z,C)*)

```
end
```

La expansión de árboles adjuntados se explica también en el § 14.13.

B.5 *Semántica operacional de la TTCN*

B.5.1 *Introducción*

Se supone que se han expandido previamente todos los subárboles (con adjunciones directas e indirectas) y árboles REPEAT utilizando las normas definidas en el texto principal de esta Recomendación. Se supone, asimismo, que se han añadido todos los valores por defecto según las normas definidas en el § B.4.2.

Se considera que un evento, seudoevento o constructivo en TTCN *concuerda* cuando puede evaluarse fructuosamente. 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 *Introducción a la notación de seudocódigo*

La semántica de la TTCN se define utilizando un procedimiento funcional simple, que explica la ejecución de un árbol de comportamiento en TTCN y los componentes que forman los nodos de ese árbol. Tales funciones están previstas como ayuda a la comprensión de la semántica de la TTCN y no se pretende que se asocien 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 funciones suponen la existencia de dos variables (que no tienen nada que ver con las variables de TTCN), que son:

SendObject	estructura de datos global temporal cuyo valor es una ASP o una PDU que ha de enviarse. Este valor se construye de conformidad con las especificaciones sobre constricciones pertinentes.
ReceivedObject	estructura de datos global temporal cuyo valor es la copia de una ASP o una PDU que se ha recibido.

La ejecución de un caso de prueba en TTCN comienza con la invocación de EVALUATE_TEST_CASE.

B.5.3 *Ejecución de un caso de prueba*

B.5.3.1 *Ejecución de un caso de prueba – seudocódigo*

- **function EVALUATE_TEST_CASE** (BehaviourTree): **BOOLEAN**

(*Level es una variable local a EVALUATE_TEST_CASE). Si A_i es una alternativa de un conjunto de *m* alternativas del árbol de comportamiento vigente, Level es el conjunto ordenado (A_1, A_2 , … A_m) en el que cada una de las A_i es un evento, un seudoevento o un constructivo. Obsérvese que, como el árbol está completamente expandido, el único enunciado en TTCN que puede aparecer es GOTO*)

begin
Level:=FIRST_LEVEL;
EVALUATE_LEVEL (Level)

end

- **function EVALUATE_LEVEL** (Level): **BOOLEAN**

(*Las alternativas A_1 , … A_m) contenidas en Level (nivel) se procesan en el orden en el que aparecen. La semántica operacional de la TTCN supone que el procesamiento de un conjunto de alternativas es instantáneo, es decir, el estado de cualquiera de los eventos no puede cambiar en el proceso de concordancia. Obsérvese que el nivel se actualiza al próximo nivel mediante la función de declaración apropiada*)

begin
repeat
TAKE_SNAPSHOT;
if EVALUATE_EVENT_LINE(A_1, Level) **then** EVALUATE_LEVEL(Level);
if EVALUATE_EVENT_LINE(A_2, Level) **then** EVALUATE_LEVEL(Level);
if ...
...
...
if EVALUATE_EVENT_LINE(A_m , Level) **then** EVALUATE_LEVEL(Level)
until SNAPSHOT_FIXED(Level)
RETURN(TestCaseError)

(*donde SNAPSHOT_FIXED (Level) devuelve TRUE si toda(s) la(s) cola(s) de los PCO pertinentes tienen algún(os) elementos(s) y todos los temporizadores pertinentes han expirado. En cualquier otro caso devolverá FALSE*)

end

- **function EVALUATE_EVENT_LINE** (A_i, Level): **BOOLEAN**

(*Esta función llama a EVALUATE_EVENT, EVALUATE_PSEUDO_EVENT o EVALUATE_CONSTRUCT según el tipo de evento que sea alternativa (A_i) vigente*)

case A_i of
EVENT: **if** EVALUATE_EVENT (A_i, Level) **then** RETURN(TRUE) **else** RETURN(FALSE);
PSEUDO_EVENT: **if** EVALUATE_PSEUDO_EVENT (A_i, Level) **then** RETURN(TRUE) **else** RETURN(FALSE);
TTCN_CONSTRUCT: **if** EVALUATE_CONSTRUCT (A_i, Level) **then** RETURN(TRUE) **else** RETURN(FALSE)

end

B.5.3.2 *Ejecución de un caso de prueba – Descripción en lenguaje natural*

Paso 1 La evaluación de un caso de prueba comienza al nivel de sangrado más a la izquierda (esto es, las líneas que aún no están sangradas en el árbol de TTCN.GR) del árbol.

Paso 2 Se toma una instantánea de la cola o colas del PCO de entrada y de la lista de temporizaciones.

Nota 1 – El acto de toma de la instantánea no elimina ningún evento del PCO.

Dentro de la línea de comportamiento del nivel vigente de alternativas, se considera la que esté especificada la primera.

Paso 3 El enunciado en TTCN de la línea de comportamiento vigente se evalúa en base a lo especificado en el elemento situado más a la izquierda, excepto si el enunciado en TTCN comienza con un identificador de PCO, en cuyo caso se evalúan en base a lo que siga al identificador de PCO.

La evaluación de cada tipo de enunciado en TTCN se especifica en la semántica operacional correspondiente a ese tipo de enunciado en TTCN. Se considera que las asignaciones, operaciones de temporizador y eventos SEND toman un valor correspondiente a una concordancia correcta, salvo en el caso en que se califiquen mediante una expresión booleana cuyo valor sea FALSE. Se considera que los enunciados IMPLICIT SEND y GOTO concuerdan siempre correctamente. El evento RECEIVE puede o no concordar, dependiendo de la expresión booleana, si existe, y del primer evento de la cola del PCO pertinente.

Paso 4 Si el enunciado en TTCN produce como evaluación una concordancia correcta, se va al paso 5.

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 3.

Si no hay más alternativas, pero todas las colas de PCO 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, es que existe un error de caso de prueba, por lo que ha detenerse el paso de prueba con la indicación de *error de caso de prueba*.

Nota – Se trata de un error de caso de prueba porque, en estas condiciones, no puede concordar ninguna alternativa del conjunto. Para los demás casos deberá tomarse una nueva instantánea de la cola o colas del PCO y de la lista de temporizaciones y considerar nuevamente el primer enunciado en TTCN del conjunto de alternativas. A continuación se va al paso 3.

Paso 5 Si se ha alcanzado un nodo hoja del árbol, se va al paso 6.

En cualquier otro caso, se considera para su evaluación un nuevo conjunto de alternativas (conjunto constituido por los enunciados en TTCN que siguen inmediatamente al enunciado en TTCN que acaba de concordar y están sangrados a un solo nivel del mismo). Ir al paso 2.

Paso 6 Si se ha alcanzado un nodo hoja del árbol, se utiliza el valor actual de la variable R de resultado preliminar como el veredicto final del caso de prueba, al igual que en el § B.5.16.2.

B.5.4 *Funciones para eventos de TTCN*

B.5.4.1 *Funciones para eventos de TTCN – Seudocódigo*

- **function EVALUATE_EVENT** (A_i, Level): **BOOLEAN**

(*Esta función llama a SEND, IMPLICIT SEND, RECEIVE, OTHERWISE o TIMEOUT dependiendo del tipo de evento que sea la alternativa (A_i) vigente*)

```
case Ai of
  SEND:          if  SEND (Ai, Level) then RETURN(TRUE) else RETURN(FALSE);
  RECEIVE:       if RECEIVE (Ai, Level) then RETURN(TRUE) else RETURN(FALSE);
  OTHERWISE:     if OTHERWISE (Ai, Level) then RETURN(TRUE) else RETURN(FALSE);
  TIMEOUT:       if TIMEOUT (Ai, Level) then RETURN(TRUE) else RETURN(FALSE);
  IMPLICIT_SEND: if IMPLICIT_SEND (Level) then RETURN(TRUE)
end
```

B.5.4.2 *Funciones para eventos de TTCN – Descripción en lenguaje natural*

Si el enunciado en TTCN es un evento, se evaluará como se indica en el § B.5.5.2 para un evento SEND; en el § B.5.6.2 para un evento RECEIVE; en el § B.5.7.2 para un evento OTHERWISE; en el § B.5.8.2 para un evento TIMEOUT; o en el § B.5.9.2 para un evento IMPLICIT SEND.

B.5.5 *Ejecución del evento SEND*

B.5.5.1 *Ejecución del evento SEND – Seudocódigo*

- **function SEND** (PCOidentifier,
  ASPidentifier or PDUidentifier,
  Qualifier,
  Assignment,
  TimerOperation,
  ConstraintsReference,
  Verdict,
  Level): **BOOLEAN**

  (*Todos los parámetros, excepto Level, se toman de A_i*)

  begin
  if EVALUATE_BOOLEAN (Qualifier) **then**
  begin
  BUILD_SEND_OBJECT (ASPidentifierOrPDUidentifier, ConstraintsReference);
  EXECUTE_ASSIGNMENT (Assignment);
  SEND_EVENT (PCOidentifier);
  TIMER_OP (TimerOperation);
  VERDICT (Verdict);
  Level:=NEXT_LEVEL;
  LOG (PCOidentifier, SendObject);
  RETURN(TRUE)
  end
  else RETURN(FALSE)

  end

- **function BUILD_SEND_OBJECT** (ASPidentifierOrPDUidentifier,
  ConstraintsReference): **BOOLEAN**

  begin
  SendObject := (**an instance of** ASPidentifierOrPDUidentifier, **whose parameters/fields have the values specified by** ConstraintsReference);
  RETURN(TRUE)

  end

B.5.5.2 *Ejecución del evento SEND – Descripción en lenguaje natural*

Ha de enviarse el contenido de la ASP o la PDU tal y como se especifica en la inscripción 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 Se asignan los valores especificados en la referencia a constricciones denominada, a la declaración de la ASP o la PDU

Paso 3 Si se ha hecho uso de la característica (*feature*) de encadenamiento dinámico, se asigna el valor especificado en la inscripción de referencia a constricciones al parámetro o campo apropiado de la ASP o la PDU 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 o la PDU 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 4 Si existe un enunciado de asignación, se efectúa esa asignación tal como se indica en el § B.5.12.2.

Paso 5 En este momento, la ASP o PDU está completamente llena de acuerdo con las especificaciones dadas. El LT o el UT envía la ASP o la PDU. (Si se especificó un PCO, se envía a dicho PCO la ASP o la PDU. 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.)

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 el § B.5.13.

Paso 7 Si hay un veredicto codificado, se procesa como se indica en el § B.5.16.2.

Paso 8 En el registro cronológico de conformidad se registra la información que sigue, así como la información especificada en el § B. 5.17.2:

– el PCO en el que se produjo el SEND;

– la ASP, la PDU o el TCP totalmente definido que se envió.

B.5.6 *Ejecución del evento RECEIVE*

B.5.6.1 *Ejecución del evento RECEIVE – Seudocódigo*

- **function RECEIVE** (PCOidentifier,
  ASPidentifier or PDUidentifier,
  Qualifier,
  Assignment,
  TimerOperation,
  ConstraintsReference,
  Verdict,
  Level): **BOOLEAN**

(*Todos los parámetros, excepto Level, se tomand de A_i*)

```
begin
    if  RECEIVE_EVENT (PCOidentifier) then
      begin
        if (RECEIVED_OBJECT(ASPidentifierOrPDUidentifier, ConstraintsReference)
            AND EVALUATE_BOOLEAN (Qualifier))
            then
              begin
                EXECUTE_ASSIGNMENT (Assignment);
                TIMER_OP (TimerOperation);
                REMOVE_OBJECT (PCOidentifier);
                VERDICT (Verdict);
                Level:=NEXT_LEVEL;
                LOG (PCOidentifier, ReceivedObject);
                RETURN(TRUE)
              end
        else RETURN(FALSE)
      end
    else RETURN(FALSE)
end
```

- **function RECEIVE_EVENT** (PCOidentifier): **BOOLEAN**

(*El objeto real NO se elimina de la cola del PCOidentifier*)

```
begin
    if INPUT_Q (PCOidentifier) NOT empty then
        begin
            ReceivedObject := copy of object at head of INPUT_Q (PCOidentifier);
            RETURN(TRUE)
        end
    else RETURN(FALSE)
end
```

- **function RECEIVED_OBJECT** (ASPidentifierOrPDUidentifier,
  ConstraintsReference): **BOOLEAN**

```
begin
    if ( (ReceivedObject is ASPidentifierOrPDUidentifier)
        AND
        (parameters/fields of ReceivedObject have the values specified by the
        ConstraintsReference) )
    then RETURN(TRUE)
  else  RETURN(FALSE)
end
```

B.5.6.2 *Ejecución del evento RECEIVE – Descripción en lenguaje natural*

El LT o el UT verifica si se ha recibido el contenido de la ASP o la PDU tal y como se especifica en la inscripción de referencia a constricciones denominada. Obsérvese que si existe un calificador, solamente podrá concordar el RECEIVE si dicho calificador toma el valor TRUE.

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 o PDU entrante, este RECEIVE no podrá concordar.

En cualquier otro caso, se continúa con el paso 2.

Paso 2 Se ensambla una copia de la ASP o la PDU utilizando la estructura definida en la declaración de la ASP o la PDU más los valores especificados en la referencia a constricciones denominada. Esta copia se utiliza para efectuar una comparación con la ASP o la PDU entrante (si existe), para determinar si RECEIVE puede concordar como se había especificado.

Paso 3 Si se ha hecho uso de la característica de encadenamiento dinámico, se asigna el valor especificado en la inscripción de referencia a constricciones al parámetro o campo apropiado de la ASP o la PDU que vaya a utilizarse en la comparación.

La utilización de la característica de encadenamiento dinámico tiene como efecto el almacenamiento de una copia de la constricción denominada en el parámetro o campo denominado de la ASP o la PDU que se está construyendo, a efectos de comparación. Para este parámetro o campo denominados se utiliza la estructura definida para la Referencia a Constricciones asociada.

Paso 4 En este momento, la ASP o la PDU está completamente llena, de acuerdo con las especificaciones dadas. El LT o el UT compara la copia creada a partir de los datos especificados con la ASP o PDU recibida en la instantánea, si existe.

Si se especificó un PCO, la ASP o la PDU se habrá recibido en ese PCO. 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.

Paso 5 Si existe un calificador, se evalúa ese calificador.

– Si el calificador toma el valor FALSE, RECEIVE no puede concordar.

– Si el calificador toma el valor TRUE, se va la paso 6.

Paso 6 Si el LT o la UT es incapaz de concordar la ASP o la PDU como está especificado (por ejemplo, no ha llegado ninguna ASP o PDU, ha llegado una ASP o una PDU, pero no concuerda con los datos como se había especificado, o bien la ASP o la PDU sí concuerda, pero no se cumple el calificador), se considera que el evento RECEIVE no concuerda, es decir, que se intenta la alternativa siguiente a este RECEIVE.

Si el RECEIVE concuerda satisfactoriamente, se continúa con el paso 7 (la ASP o la PDU entrante que acaba de concordadar, se elimina de la cola del PCO entrante y, en consecuencia, no estará disponible a efectos de concordancia con cualesquiera eventos subsiguientes del caso de prueba).

Paso 7 Si existe un enunciado de asignación, se efectúa esa asignación tal como se indica en el § B.5.12.2.

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 el § B.5.13.

Paso 9 Si hay un veredicto codificado, se procesa como se indica en el § B.5.16.2.

Paso 10 En el registro cronológico de conformidad se registra la información que sigue, así como la información especificada en el § B.5.17.2:

– el PCO en el que se produjo el RECEIVE;

– la ASP, la PDU o el TCP totalmente definido que se recibió.

B.5.7 *Ejecución del evento OTHERWISE*

B.5.7.1 *Ejecución del evento OTHERWISE – Seudocódigo*

```
• function OTHERWISE (PCOidentifier,
                      Qualifier,
                      Assignment,
                      TimerOperation,
                      Verdict,
                      Level): BOOLEAN

    (*Todos los parámetros, excepto Level, se toman de Ai*)

begin
    if ( (RECEIVE_EVENT (PCOidentifier))
      AND EVALUATE_BOOLEAN (Qualifier) )
    then
      begin
        EXECUTE_ASSIGNMENT (Assignment);
        TIMER_OP (TimerOperation);
        REMOVE_OBJECT (PCOidentifier);
        VERDICT (Verdict);
        Level:=NEXT_LEVEL;
        LOG (PCOidentifier, ReceivedObject);
        RETURN(TRUE)
      end
    else RETURN(FALSE)
end
```

B.5.7.2 *Ejecución del evento OTHERWISE – Descripción en lenguaje natural*

El probador debe aceptar cualquier dato de entrada 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 se especificó un PCO, la ASP o la PDU se habrá recibido en ese PCO. 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.

Paso 2 Si existe un calificador, se evalúa con antelación a cualquier otro tipo de procesamiento.

– Si el calificador toma el valor FALSE, el OTHERWISE no puede concordar.

– Si el calificador toma el valor TRUE, se continúa con el paso 3.

Paso 3 Si el probador es incapaz de recibir una ASP o una PDU (es decir, que no ha llegado ninguna ASP o PDU) se considerará que el evento OTHERWISE no concuerda y se intenta la alternativa siguiente a este OTHERWISE.

Si el OTHERWISE se verificó fructuosamente, se va al paso 4 (la ASP o la PDU entrante que acaba de concordar no estará disponible a efectos de concordancia con cualesquiera otros eventos subsiguientes del caso de prueba).

Paso 4 Si existe un enunciado de asignación, se efectúa esa asignación tal como se indica en el § B.5.12.2.

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 el § B.5.13.

Paso 6 Si hay un veredicto codificado, se procesa como se indica en el § B.5.16.2.

Paso 7 En el registro cronológico de conformidad se registra la información que sigue, así como la información especificada en el § B.5.17.2:

– el PCO en el que se produjo el OTHERWISE;

– la ASP o la PDU totalmente definida que se recibió.

B.5.8 *Ejecución del evento TIMEOUT*

B.5.8.1 *Ejecución del evento TIMEOUT – Seudocódigo*

```
function TIMEOUT (TimerIdentifier,
                  Qualifier,
                  Assignment,
                  TimerOperation,
                  Verdict,
                  Level): BOOLEAN

(*Todos los parámetros, excepto Level, se toman de Ai*)

begin
if EVALUATE_BOOLEAN (Qualifier) then
    begin
        if (TIMER_EXPIRED (TimerIdentifier)) then
            begin
                EXECUTE_ASSIGNMENT (Assignment);
                TIMER_OP (TimerOperation);
                VERDICT (Verdict);
                Level:=NEXT_LEVEL;
                LOG (TimerIdentifier);
                RETURN(TRUE)
            end
        else RETURN(FALSE)
    end
else RETURN(FALSE)

end

function TIMER_EXPIRED (TimerIdentifier): BOOLEAN

begin
if (timer has expired) then

begin
reset expired timer; (*véase el § B.5.8.2*)
    RETURN(TRUE)
end

else
RETURN(FALSE)

end
```

B.5.8.2 *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 inscripción 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 inscripción 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 de asignación, se efectúa esa asignación tal como se indica en el § B.5.12.2.

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 el § B.5.13.

Paso 5 Si hay un veredicto codificado, se procesa como se indica en el § B.5.16.2.

Paso 6 En el registro cronológico de conformidad se registra la información que sigue, así como la información especificada en el § B.5.17.2:

– nombre del temporizador que ha expirado.

B.5.9 *Ejecución del evento IMPLICIT SEND*

B.5.9.1 *Ejecución del evento IMPLICIT SEND – Seudocódigo*

- **function IMPLICIT_SEND** (Level): **BOOLEAN**

  begin

  (*Evaluar IMPLICIT_SEND de conformidad con el § 14.9.6**)
  Level:=NEXT_LEVEL;
  RETURN(TRUE)

  end

B.5.9.2 *Ejecución de IMPLICIT SEND – Descripción en lenguaje natural*

La IUT hace todo lo que sea necesario para el envío del contenido de la ASP o la PDU como se especifica en la inscripción de referencia de constricciones denominada.

Si se ha hecho uso de la característica de encadenamiento dinámico, se asigna el valor especificado en la inscripción de referencia de constricciones al parámetro o campo apropiado de la ASP o la PDU que haya de enviarse.

B.5.10 *Ejecución de PSEUDO_EVENT*

B.5.10.1 *Ejecución de PSEUDO_EVENTS – Seudocódigo*

- **function EVALUATE_PSEUDO_EVENT** (Qualifier,
  Assignment,
  TimerOperation,
  Verdict,
  Level): **BOOLEAN**

(*Todos los parámetros, excepto Level, se toman de A_i*)

```
begin
if EVALUATE_BOOLEAN (Qualifier)
    then
        begin
            ASSIGNMENT (Assignment);
            TIMER_OP (TimerOperation);
            VERDICT (Verdict);
            Level:=NEXT_LEVEL;
            LOG ( );
            RETURN(TRUE)
        end
    else RETURN(FALSE)
end
```

B.5.10.2 *Ejecución de PSEUDO_EVENTS – Descripción en lenguaje natural*

Si el enunciado TTCN es un seudoevento, se evaluará como se indica en el § B.5.11.2 para una expresión booleana; el § B.5.12.2 para un enunciado seudoevento asignación; en el § B.5.13.2 para un seudoevento de temporizador START; en el § B.5.13.3 para un seudoevento de temporizador CANCEL; o el § B.5.13.4 para un seudoevento de temporizador READ.

Paso n Tras la compleción del seudoevento, se registra en el registro cronológico de conformidad la información especificada en el § B.5.17.2.

B.5.11 *Ejecución de expresiones BOOLEAN*

B.5.11.1 *Ejecución de expresiones BOOLEAN – Seudocódigo*

- **function EVALUATE_BOOLEAN** (Qualifier): **BOOLEAN**

(*para más información, véase el § B.5.11.2*)

```
begin
if no argument then RETURN(TRUE)
else   begin
           if Qualifier then RETURN(TRUE)
           else RETURN(FALSE)
       end

end
```

B.5.11.2 *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 verdadera (TRUE) o falsa (FALSE). La expresión booleana puede enunciarse 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 el § 11.4.2.1.

Paso 2 Si la condición expresada mediante la expresión booleana toma el valor FALSE, la expresión booleana no es un evento concordante. Se omiten los pasos restantes de esta cláusula.

Si según la evaluación de la expresión booleana hay concordancia, el procesamiento subsiguiente depende de si la expresión booleana fue codificada como parte de un evento (esto es, SEND, RECEIVE, TIMEOUT u OTHERWISE) o si fue codificada como un seudoevento.

Si la expresión booleana se codificó como un evento, se considera que este segmento de la línea de enunciado concuerda, prosiguiéndose la evaluación del resto de la línea de enunciado, de conformidad con la semántica de ese tipo de evento. Deberán omitirse los pasos siguientes, puesto que se aplican solamente a expresiones booleanas codificadas como seudo eventos.

Si la expresión booleana se codificó como un seudoevento, se considera que la misma tiene un comportamiento concordante. Se continúa con el paso siguiente para procesar el resto de la línea de comportamiento de la expresión booleana.

Paso 3 Si hay un enunciado de asignación, se efectúa esa asignación tal como se indica en el § B.5.12.2.

Paso 4 Si en la línea de comportamiento se ha codificado una o más operaciones de temporizador, se efectúan la operación u operaciones de temporizador apropiadas, tal como se indica en el § B.5.13.

Paso 5 Si hay un veredicto codificado, se procesa como se indica en el § B.5.16.2.

B.5.12 *Ejecución de ASSIGNEMENT*

B.5.12.1 *Ejecución de EXECUTE_ASSIGNEMENT – Seudocódigo*

- **function EXECUTE_ASSIGNMENT** (Assignment): **BOOLEAN**

(*Obsérvese que asignaciones del tipo <*PDUidentifier*>, <*FIELDidentifier*> etc. asignan (reasignan) valores a campos/parámetros de SendObject. Este tipo de asignación no deberá emplearse con un evento RECEIVE*)

```
begin
    if no argument then RETURN(TRUE)
    else begin
        execute the clauses in Assignment in a left-to-right order;
        RETURN(TRUE)
    end
end
```

B.5.12.2 *Ejecución de ASSIGNEMENT – Descripción en lenguaje natural*

Un enunciado de asignación especifica que la variable del lado izquierdo de ese enunciado debe tomar el valor del lado derecho del enunciado. Un enunciado de asignación puede formularse como parte de una línea de enunciado (esto es, en la misma línea que SEND, RECEIVE, TIMEOUT u OTHERWISE), en combinación con una expresión booleana o como una línea de comportamiento propia (esto es, como un seudoevento).

Paso 1 La manera de ejecutar la asignación depende del formato utilizado en la especificación del lado derecho de la asignación. Las cláusulas se evalúan de izquierda a derecha, respetando la precedencia indicada en el cuadro 3.

Paso 2 Una vez ejecutada la asignación, el procesamiento subsiguiente depende de si la asignación fue codificada como parte de un evento (esto es, SEND, RECEIVE, TIMEOUT u OTHERWISE) o como un seudoevento.

Si el enunciado de asignación se codificó como un seudoevento, se continúa con el paso siguiente para procesar el resto de la línea de comportamiento del enunciado de asignación.

Paso 3 Si en la línea de comportamiento se han codificado una o más operaciones de temporizador, se efectúan las operaciones de temporizador apropiadas, tal como se indica en el § B.5.13.

Paso 4 Si hay un veredicto codificado, se procesa como se indica en el § B.5.16.2.

B.5.13 *Ejecución de operaciones TIMER*

B.5.13.1 *Ejecución de operaciones TIMER – Seudocódigo*

- **function TIMER_OP** (TimerOperation): **BOOLEAN**

```
begin
if no argument then RETURN(TRUE)
else  case TimerOperation of
            START_TIMER:     if   START_TIMER(TimerOperation) then RETURN(TRUE)
                             else RETURN(FALSE); (*véase el § B.5.13.2 para START_TIMER*)
            CANCEL_TIMER:    if   CANCEL_TIMER(TimerOperation) then RETURN(TRUE)
                             else RETURN(FALSE); (*véase el § B.5.13.3 para CANCEL_TIMER*)
            READ_TIMER:      if   READ_TIMER(TimerOperation) then RETURN(TRUE)
                             else RETURN(FALSE); (*véase el § B.5.13.4 para READ_TIMER*)
      end
end
```

B.5.13.2 *Ejecución de temporizador START – Descripción en lenguaje natural*

La operación de temporizador START especifica que un temporizador va a comenzar a funcionar. Esta operación puede codificarse como parte de una línea de enunciado (esto es, en la misma línea que SEND, RECEIVE, TIMEOUT u OTHERWISE), como una línea de comportamiento propia o en combinación con un calificador y/o un enunciado de asignación.

Paso 1 Se determina la duración que ha de utilizarse en esta instancia del temporizador. Si para esta operación no se ha especificado ninguna duración (en forma de valor entero o expresión) se utiliza la duración por defecto indicado en las declaraciones de temporizador.

Si, por el contrario, se ha especificado una duración, se utilizará esa duración en vez del valor por defecto indicado en la declaración del temporizador. La contraordenación de una duración de temporizador por defecto mediante la codificación de una duración en la operación de temporizador START se aplica solamente a esta instancia del temporizador. Si el temporizador se arranca en cualquier otro punto de la prueba no resulta afectado por esta contraordenación (*override*) de la duración.

Paso 2 Si este temporizador ya está en marcha, deberá cancelarse y rearrancarse, por lo que no transcurrirá ningún tiempo. Si el temporizador no está todavía en marcha, deberá arrancarse con un valor inicial que indique que no ha transcurrido ningún tiempo. Cualquier inscripción para este temporizador que figure en la lista de temporizaciones deberá eliminarse de la lista.

Paso 3 Si a continuación de esta operación START aparece codificada otra operación de temporizador, se procesa esa operación de temporizador (excluido el procesamiento de veredicto).

Paso 4 Si hay un veredicto codificado, se procesa como se indica en el § B.5.16.2.

B.5.13.3 *Temporizador CANCEL – Descripción en lenguaje natural*

La operación de temporizador CANCEL especifica que un temporizador o unos temporizadores van a dejar de funcionar. Esta operación puede codificarse como parte de una línea de enunciado (esto es, en la misma línea que SEND, RECEIVE, OTHERWISE o TIMEOUT), como una línea de comportamiento propia o en combinación con un calificador y/o un enunciado de asignación.

Paso 1 Se determina el nombre del temporizador o temporizadores que se han de cancelar:

– si no se especifica ningún identificador de temporizador, el LT o UT cancela *todos* los temporizadores que habían estado funcionando;

– si se especifica un identificador de temporizador, el LT o el UT cancela este temporizador (que había estado funcionando).

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 inscripción para el temporizador o temporizadores, se eliminará esa inscripción o inscripciones de la lista.

Paso 3 Si a continuación de esta operación CANCEL aparece codificada otra operación de temporizador, se procesa esa operación de temporizador (excluido el procesamiento de veredicto).

Paso 4 Si hay un veredicto codificado, se procesa como se indica en el § B.5.16.2.

B.5.13.4 *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. Esta operación puede codificarse como parte de una línea de enunciado (esto es, en la misma línea que SEND, RECEIVE, OTHERWISE o TIMEOUT), como una línea de comportamiento propia o en combinación con un calificador y/o un enunciado de asignación.

Paso 1 Se pregunta el valor del temporizador que tiene el nombre especificado. El valor del tiempo transcurrido se almacena en la variable denominada. Las unidades devueltas, son las mismas que las unidades declaradas para ese tipo de temporizador.

Si el temporizador no está funcionando en ese momento, la variable denominada se pone a cero.

Paso 2 Si a continuación de esta operación READTIMER aparece codificada otra operación de temporizador, se procesa esa operación de temporizador (excluido el procesamiento de veredicto).

Paso 3 Si hay un veredicto codificado, se procesa como se indica en el § B.5.16.2.

B.5.14 *Funciones para constructivos de TTCN*

B.5.14.1 *Funciones para constructivos de TTCN – Seudocódigo*

- **function EVALUATE_CONSTRUCT** (Construct, Level): **BOOLEAN**

(*Todos los parámetros, excepto Level, se toman de A_i*)

(*El árbol de caso de prueba está totalmente expandido antes del procesamiento, por lo que nunca se encuentran los constructivos REPEAT y ATTACH; véanse los § B.4.3 y B.4.4*)

case Construct **of**

GOTO: GOTO(Label, Level);

RETURN(TRUE)

end

B.5.14.2 *Funciones para constructivos de TTCN – Descripción en lenguaje natural*

Si la sentencia en TTCN es un constructivo de TTCN, se evalúa como se indica en el § B.5.15.2 para un constructivo GOTO; véase el § B.4 para la descripción de los constructivos REPEAT y ATTACH.

B.5.15 *Ejecución de constructivo GOTO*

B.5.15.1 *Ejecución de constructivo GOTO – Seudocódigo*

- **function GOTO** (Label, Level): **BOOLEAN**

begin

Level:=LABELED_LEVEL(Label);

RETURN(TRUE)

end

B.5.15.2 *Ejecución del constructivo GOTO – Descripción en lenguaje natural*

El LT o el UT controla la transferencia desde el evento en curso (esto es, el constructivo GOTO) 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.

B.5.16 *VERDICT (veredicto)*

B.5.16.1 *Veredicto– Seudocódigo*

- **function VERDICT** (VerdictColumnEntry): **BOOLEAN**

```
begin
if VerdictColumnEntry is blank then RETURN(TRUE)
else begin
        if VerdictColumnEntry = R then LOG(Verdict implied by R);
        else LOG(VerdictColumn);
        if VerdictColumnEntry is preliminary result then (*Contiene veredicto*)
            begin
              if ( (R = none) or (R = pass and VerdictColumnEntry <> (PASS))
                or (R = inconc and VerdictColumnEntry = (FAIL) ) then
                begin
                  if VerdictColumnEntry = (PASS) then R := pass;
                  if VerdictColumnEntry  = (INCONC) then R := inconc;
                  if VerdictColumnEntry = (FAIL) then R := fail;
                end
      end

   else  (*Veredicto final*)
   begin
      reset test case variables all timers;
      if ( (VerdictColumnEntry = R and R = none) or
         (VerdictColumnEntry = PASS and R = inconc) or
         (VerdictColumnEntry = PASS and R = fail) or
         (VerdictColumnEntry = INCONC and R = fail) )
       then raise test case error;
       STOP (*El caso de prueba termina aquí*)
   end
  end
end
```

B.5.16.2 *Veredicto – Descripción en lenguaje natural*

Si hay un veredicto codificado, se procesa el veredicto.

– Si el veredicto figura entre paréntesis, se actualiza la variable R de resultado temporal según el algoritmo de veredicto del § 15.17.2. El veredicto establecido se registra en el registro cronológico de conformidad.

– Si el veredicto es R, se utiliza el valor vigente de la variable R de resultado temporal 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 el veredicto establecido como veredicto final del caso de prueba. Si el veredicto final no es coherente con el veredicto preliminar se señala un *TestCaseError*.

B.5.17 *Registro cronológico de conformidad (conformance log)*

B.5.17.1 LOG- seudocódigo

- **procedure LOG (número variable de argumentos)**

```
begin
log the sequence 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 the verdict or preliminary result associated with the event line (if any);
log current time; (*el tiempo vigente puede ser real o relativo*)
end
```

B.5.17.2 *Registro cronológico de conformidad – Descripción en lenguaje natural*

En el registro cronológico de conformidad se registra la siguiente información:

– número secuencial de la línea de eventos (si existe);

– etiqueta asociada con la línea de eventos (si existe);

– asignación o asignaciones efectuadas (si existen)

– operación u operaciones de temporizador ejecutadas (si existen);

– veredicto o resultado preliminar asociado con la línea de eventos (si existen);

– indicación de tiempo (*time stamp*).

B.5.18 *Otras diversas funciones utilizadas por el seudocódigo*

- **function FIRST_LEVEL: LEVEL**

begin

RETURN **(conjunto de alternativas en el primer nivel de sangrado)**

end

- **function NEXT_LEVEL: LEVEL**

begin

if (existe un conjunto de alternativas en el siguiente nivel de sangrado) **entonces**

RETURN (conjunto de alternativas en el siguiente nivel de sangrado)

else

VERDICT(R)

STOP (*El caso de prueba termina aquí*)

end

- **function LABELED_LEVEL** (Label): **LEVEL**

begin

RETURN (**conjunto de alternativas en el nivel de sangrado indicado por** Label (etiqueta))

end

- **function RETURN** (argument): **BOOLEAN or LEVEL or QUEUE**

salir inmediatamente de la función vigente y devolver el valor de argument (argumento)

end

- **function OUTPUT_Q** (PCOidentifier): **QUEUE**

(*en la TTCN, cada PCO se modela como dos colas FIFO no limitadas: una es la cola INPUT (al LT o el UT) y la otra, es la cola OUTPUT (del LT o del UT). En las sucesiones de pruebas que utilizan un único PCO y cuando este PCO no esté asociado explícitamente con un evento, se tomará el único PCO (por defecto) de las declaraciones de PCO*)

begin

if no argument then RETURN(**default PCO output queue**)

else RETURN(**output queue identified by** PCOidentifier)

end

- **function INPUT_Q** (PCOidentifier): **QUEUE**

(*en la TTCN, cada PCO se modela como dos colas FIFO no limitadas: una es la cola INPUT (al LT o al UT) y la otra, es la cola OUTPUT (del LT o del UT). En las sucesiones de pruebas que utilizan un único PCO y cuando este PCO no esté asociado explícitamente con un evento, se tomará el único PCO (por defecto) de las declaraciones de PCO*)

begin
if no argument then RETURN(**default PCO input queue**)
else RETURN(**input queue identified by** PCOidentifier)
end

- **procedure TAKE_SNAPSHOT**

(*se emplea semántica instantánea (*snapshot semantics*) para eventos RECEIVE y temporizaciones, es decir, que en cada ciclo se toma una instantánea de un conjunto de alternativas de las que se han recibido eventos, y de las temporizaciones que han arrancado. Solo las temporizaciones identificadas en la instantánea pueden concordar en el ciclo siguiente a través de las alternativas*)

begin
update PCO input queues;
update TIMER queue;
end

- **procedure STOP**

begin
reset test case variables;
reset PCO queue;
reset TIMER queue;
reset timers;
terminate execution immediately
end

ANEXO C

(a la Recomendación X.292)

Formularios compactos

(Este anexo es parte integrante de la presente Recomendación)

C.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 el § 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.

C.2 *Formularios compactos para constricciones*

C.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; y

b) 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.

C.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 C-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 el § C.2.3).

C.2.3 *Formularios compactos para constricciones de PDU*

C.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 C-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, deberá haber *n* columnas de campo en el formulario de constricciones compactas.

La columna de trayecto de derivación es optativa; sin embargo, deberá utilizarse para especificar el trayecto de derivación de las constricciones modificadas (véase el § 12.6). 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 C.1 – Constricciones que utilizan el formulario de constricciones compacto

C.1.1 Dada la declaración de PDU_B en la forma:

Definición de tipo de PDU		
Nombre de la PDU : PDU_B **Tipo de PCO :** XSAP **Comentarios :**		
Nombre del campo	**Tipo del campo**	**Comentarios**
FIELD1	INTEGER	
FIELD2	BOOLEAN	
FIELD3	IA5String	

C.1.2 Las constricciones impuestas a PDU_B utilizando 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	3	TRUE	«A string»	
CN2	(4, 5, 6)	FALSE	«A string»	
CN3	0	?	–	

La referencia a constricciones en la parte dinámica podrá contener entonces inscripciones tales como PDU_B[CN1] y PDU_B[CN2].

Ejemplo C.2 – 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		0	'FF'H	'00'B	TRUE	
CN1	CN0.	1				
CN2	CN0.CN		–	?		

C.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 C.3 – Constricción compacta parametrizada

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) ó S5(Var) siendo Var un caso de prueba o una variable de caso de prueba.

Declaraciones de constricciones de PDU			
Tipo de PDU: PDU_X			
Nombre de la constricción	**Nombre de campo**		**Comentarios**
	P1	P2	
S1	0	0	
S2	0	1	
S3	1	0	
S4	1	1	
S5(A:INTEGER)	1	A	

C.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 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 C-3 – Declaraciones de constricciones compactas de tipo estructurado

Ejemplo C.4 – Utilización de constricciones compactas estructuradas

La PDU_Y consta de cinco campos, denominados Y1, Y2, Y3, Y4 e 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.

C.4.1 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	A1	0	1	
YY2	A2	1	0	
YY3	A2	0	1	

C.4.2 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	

C.4.3 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	

C.4.4 Contricciones 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		0	0	0	
A2	A1.		3		

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.

C.2.5 *Formularios compactos para constricciones en ASN.1*

Para definiciones de constricciones de ASP en ASN.1, PDU en ASN.1 y Tipo ASN.1 en forma compacta se utilizarán respectivamente los siguientes formularios:

Declaraciones de constricciones de ASP en ASN.1	
Tipo – ASP: *ASP_Identifier*	
Nombre de la constricción	**Valor en ASN.1**
ConsId&ParList$_1$	*ConstraintValue&Attribute*$_1$
⋮	⋮
ConsId&ParList$_m$	*ConstraintValue&Attribute*$_m$

Formulario C-4 – Declaraciones de constricciones de ASP en ASN.1 (compactas)

Declaraciones de constricciones de PDU en ASN.1	
Tipo – PDU: *PDU_Identifier*	
Nombre de la constricción	**Valor en ASN.1**
ConsId&ParList$_1$	*ConstraintValue&Attributes*$_1$
⋮	⋮
ConsId&ParList$_m$	*ConstraintValue&Attributes*$_m$

Formulario C-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 C-6 – Declaraciones de constricciones de tipo en ASN.1 (compactas)

C.3 *Formulario compacto para casos de prueba*

C.3.1 *Requisitos*

Solo 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 pruebas;

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.

C.3.2 *Formulario compacto para comportamiento dinámico de casos de prueba*

Cuando una sucesión 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* **Supletorio :** *DefaultReference*			
Nombre del caso de prueba	**Finalidad**	**Adjunción de caso de prueba**	**Comentarios**
. . *TestCaseIdentifier* . .	. . *FreeText* . .	. . *Attach* . .	. . *[FreeText]* . .

Formulario C-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 sucesión 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 Indice de caso de prueba.

Ejemplo C.5 – Cuadro de caso de prueba compacto que define una sucesión 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 (*override*) cuando existe el fichero	+OVERRIDE (FCRERQ_001,FCRERP_001)
OVERIDE2	Omitir el parámetro contraordenación (*override*) cuando no existe el fichero	+OVERRIDE (FCRERQ_002,FCRERP_002)

ANEXO D

(a la Recomendación X.292)

Ejemplos

(Este anexo no es parte integrante de la presente Recomendación)

D.1 *Ejemplos de constricciones en forma de tabla*

D.1.1 *Definiciones de ASP y PDU*

D.1.1.1 *Definición de tipo plano (flat)*

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	BITSTRING [4]	Longitud de 4 bits
Destination	BITSTRING [4]	Longitud de 4 bits
T_Class	INTEGER0to4	Definido como tipo simple
UserData	IA5String	

D.1.1.2 *Definición de tipo estructurado*

Definición de tipo de PDU		
Nombre de PDU : T_CONNECT2 **Tipo de PCO :** **Comentarios :** Illustración de mecanismos de TTCN		
Nombre del campo	**Tipo de campo**	**Comentarios**
T_Addresses	T_AddressInfo	
T_Class	INTEGER0to4	Definido como tipo simple
UserData	IA5String	

Definición de tipo estructurado		
Nombre de tipo : T_AddressInfo **Comentarios :** Puede utilizarse en todos los ejemplos de PDU de transporte		
Nombre del elemento	**Definición de tipo**	**Comentarios**
Source	BITSTRING [4]	Longitud de 4 bits
Destination	BITSTRING [4]	Longitud de 4 bits

D.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 :** A efectos de ilustración solamente		
Nombre del parámetro	**Tipo de parámetro**	**Comentarios**
CallingNetworkAddress	HEXSTRING	
CalledNetworkAddress	HEXSTRING	
ConnectionIdentifier	HEXSTRING	
Data	PDU	Para permitir el encadenamiento de constricciones

D.1.2 *Constricciones de ASP/PDU*

D.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	TS_Par1	
Destination	TS_Par2	
T_Class	4	
UserData	"testing, testing"	

D.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	WrongAddress	WrongAddress es una referencia a una constricción de tipo estructurado
T_Class	4	
UserData	"one, two, three"	

Declaración de constricción de tipo estructurado		
Nombre de la constricción : WrongAddress **Tipo de estructurado :** T_AddressInfo **Trayecto de derivación :** **Comentarios :**		
Nombre del elemento	**Valor del elemento**	**Comentarios**
Source	TS_Par1	
Destination	’0000’B	

D.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 (est es, un encadenamiento)		
Nombre del parámetro	**Valor del parámetro**	**Comentarios**
CallingNetworkAddress	TS_Par3	
CalledNetworkAddress	TS_Par4	
ConnectionIdentifier	'ABCDEF'H	
Data	TCON_Class4-1	

D.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 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	?1000'B	
Destination	?	
T_Class	class	Clase (*class*) es un parámetro formal
UserData	?	

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 puede invocarse, por ejemplo, de la siguiente manera:

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 de las PDU.

D.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 PDU :** T_CONNECT1 **Trayecto de derivación :** TCON_Class4_1 **Comentarios :** Clase 0 es aceptable		
Nombre del campo	**Valor del campo**	**Comentarios**
T_Class	0	

Pueden utilizarse 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 :** Es aceptable cualquier clase (0 .. 4)		
Nombre del campo	**Valor del campo**	**Comentarios**
T_Class	?	

Se considera, sin embargo, que este es un mal estilo. 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

D.2 *Ejemplos de constricciones en ASN.1*

D.2.1 *Definiciones de ASP y PDU*

D.2.1.1 *Plana*

Definición de tipo de PDU en ASN.1
Nombre de la 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 }

D.2.1.2 *Estructurada*

Definición de tipo de PDU en ASN.1
Nombre de la 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 del tipo : T_AddressInfo **Comentarios :**
Definición de tipo
SEQUENCE { source BITSTRING [SIZE(4..4)], destination BITSTRING [SIZE(4..4)], }

D.2.1.3 *Definición de ASP*

Definición de tipo de ASP en ASN.1
Nombre de la 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 del tipo : T_PDUS **Comentarios :**
Definición de tipo
CHOICE { t1 T_CONNECT1, t2 T_CONNECT2 }

D.2.2 *Constricciones de ASP/PDU en ASN.1*

D.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 puede omitirse, si se desea t_Class 4, userData "testing, testing" }

D.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 tipo en ASN.1
Nombre de la constricción : WrongAddress **Tipo estructurado :** T_AddressInfo **Trayecto de derivación :** **Comentarios :**
Valor de la constricción
{ source TS_PAR1, destination '0000'B }

D.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 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_Class_4_1} -- encadenamiento a una constricción de PDU }

D.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 (*wildcard*) 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 :** T_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.

D.2.2.5 *Constricciones modificadas*

Pueden construirse 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 constricción :** T_CON_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 :** T_CON_Class4_1. **Comentarios :**
Valor de la constricción
REPLACE t_Class BY ?

Para especificar campos que deberán omitirse 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 debe tenerse en cuenta que los propios campos parametrizados no pueden ser reemplazados:

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"

D.2.3 *Otros ejemplos de constricciones en ASN.1*

D.2.3.1 *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 la PDU : F_INITIALIZEresponse **Tipo de PCO :** **Comentarios :**
Definición del 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 del 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_0
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 -- Escribir solamente

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 de PDU :** 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.

D.3 *Constricciones de base y modificadas*

Supóngase que se tiene la siguiente definición de tipo de PDU:

Definición de tipo de PDU		
Nombre de la PDU : PDU_B **Tipo de PCO :** **Comentarios :** Esta 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 FIELD2 FIELD3 FIELD4	0 'FF'H '00'B 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 FIELD3	– ?	Se omite este campo Se acepta cualquier valor legal

La referencia a una constricción modificada en un árbol de comportamiento se hace mediante su nombre.

D.4 *Definición de tipo utilizando macros*

D.4.1 *Definición de tipo de PDU con el símbolo macro:*

Definición de tipo de PDU		
Nombre de la 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 del tipo : T_AddressGroup **Comentarios :**		
Nombre del elemento	**Definición del 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 de PDU :** T_AddressGroup **Trayecto de derivación :** **Comentarios :**		
Nombre del elemento	**Valor del elemento**	**Comentarios**
Source Destination	’0101’B ’1111’B	

D.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 **Valor por defecto :** **Comentarios :**					
N.°	**Etiqueta**	**Descripción y comportamiento**	**Ref. de constricciones**	**Veredicto**	**Comentarios**
1 2 3 4 5 6 7 8 9		(FLAG:=FALSE, COUNTER:=0) !A REPEAT STEP2 (FLAG, COUNTER) UNTIL [FLAG OR COUNTER=3] [FLAG] !D [COUNTER=3] !E STEP2 (F:BOOLEAN; NUMBER:INTEGER) ?B (F:=TRUE) ?C (F:=FALSE, NUMBER:=NUMBER+1)	 A1 D1 E1 B1 C1	 PASS FAIL	
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 E, y que si se reciben los otros tres mensajes se deberá enviar F.					

D.6 *Operaciones de sucesiones de pruebas*

Utilización de una sucesión de pruebas para establecer una suma de control (*cheksum*):

Definición de operación de sucesión 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.		

D.7 *Ejemplo de visión general de una sucesión de pruebas*

En el cuadro de estructura de sucesión de pruebas que figura más adelante se define una jerarquía de los grupos y casos de prueba de la sucesión. En dicha estructura se identifican expresiones de selección de prueba que rigen la selección de los grupos de pruebas 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 sucesión que pertenecen al grupo característica X.

Estructura de sucesión de pruebas			
Nombre de la sucesión : 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 de grupo de pruebas	**Ref. de selección**	**Objetivo de grupo de pruebas**	**N.° de pág.**
FEATURE_X	SELEXP_100	Prueba optativa de la característica X	50
FEATURE_X/ATTR_A		Prueba obligatoria del atributo A	50
FEATURE_X/ATTR_A/NEGOTIATION	SELEXP_101	Prueba optativa de negociación del atributo A	50
FEATURE_X/ATTR_A/USAGE		Prueba de la utilización del atributo A	60
FEATURE_X/ATTR_B		Prueba obligatoria de la característica Y	80

Para determinar si se soporta o no la característica X (*feature X*), ha de evaluarse SELEXP_100, lo que se efectúa determinando si el parámetro de sucesión 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 solo se seleccionarán las pruebas de la característica de negociación optativa del atributo A si SELEXP_101 es TRUE.

Índice de caso de prueba				
Referencia de grupo de pruebas	**Id. de caso de prueba**	**Ref. de selección**	**Descripción**	**N.° 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 sucesión de pruebas.

Declaraciones de parámetros de sucesión 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_NINIT	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 el § 10.5.

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

D.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 **Referencia :** 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.°	**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
3	L1	?CLEAR CANCEL TD	CLRC_0(LC)	PASS	liberación=5
4		+R1_POSTAMBLE			
5		?CLEAR 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		+D1C_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/PROPERT/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.

ANEXO E

(a la Recomendación X.292)

Guía de estilo

(Este anexo no es parte integrante de la presente Recomendación)

E.1 *Introducción*

En este anexo informativo 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 sucesiones de pruebas.

E.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 sucesión de pruebas deberá identificar con claridad los subárboles de prólogo y epílogo.

b) El epílogo y el prólogo deberán especificarse 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).

Utilizando este mecanismo, pueden identificarse con facilidad las fronteras entre prólogo, cuerpo de prueba y epílogo, dentro de un caso de prueba. Pueden utilizarse etiquetas (*labels*) para indicar el comienzo y el final del cuerpo de prueba en el registro cronológico de conformidad.

Comportamiento dinámico del caso de prueba

Nombre del caso de prueba : TTCN_EXAMPLES/STYLE1
Referencia : ST_EX1
Finalidad : Ilustrar la identificación del prólogo y los epílogos
Valor por defecto :
Comentarios :

N.°	Etiqueta	Descripción y comportamiento	Ref. de constricciones	Veredicto	Comentarios
1		+Preamble			Relacionado con la finalidad
2		!A	A1		Relacionado con la finalidad
3	Body	?B	B1		Relacionado con la finalidad
4	CinBody	?C	C1	(PASS)	
5		+postamble_1			Relacionado con la finalidad
6	DinBody	?D	D1	(PASS)	
7		+postamble_2			Relacionado con la finalidad
8		?E	E1	INCONC	Relacionado con la finalidad
9		?OTHERWISE		FAIL	

Figura E-1/X.292 – Identificación del prólogo y los epílogos

Puesto que los veredictos finales causan la terminación de la ejecución de un caso de prueba, un especificador de sucesión 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 sucesión 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.

E.3 *Utilización de la TTCN con diferentes métodos de prueba abstracta*

E.3.1 *Introducción*

En este punto se relaciona la TTCN con los métodos de prueba abstracta definidos en la Recomendación 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.

E.3.2 *La TTCN y el método de prueba LS*

Eventos TTCN posibles:

Descripción de comportamiento	*Referencia de constricciones*
LT!N_ASP	N_ASPconstraint(N_PDUconstraint)
LT?N_ASP	N_ASPconstraint(N_PDUconstraint)
UT!T_ASP	T_ASPconstraint
UT?T_ASP	T_ASPconstraint

E.3.3 *La TTCN y el método de prueba DS*

Eventos TTCN posibles:

Descripción de comportamiento	*Referencia de constricciones*
LT!N_ASP	N_ASPconstraint(T_PDUconstraint)
LT?N_ASP	N_ASPconstraint(T_PDUconstraint)
UT!T_ASP	T_ASPconstraint
UT?T_ASP	T_ASPconstraint

E.3.4 *La TTCN y el método de prueba CS*

Eventos TTCN posibles:

Descripción de comportamiento	*Referencia de constricciones*
LT!N_ASP	N_ASPconstraint(T_PDUconstraint)
LT?N_ASP	N_ASPconstraint(T_PDUconstraint)

Intercambio de TM_PDU entre las realizaciones 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 deberá haber 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))

E.3.5 *La TTCN y el método de prueba RS*

Eventos TTCN posibles:

Descripción de comportamiento	*Referencia de constricciones*
LT!N_ASP	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

E.4 *Utilización de valores por defecto*

Como cuestión de estilo, un especificador de sucesiones 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. Este 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 (*pass*).

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.

E.5 *Limitación del tiempo de ejecución de un caso de prueba*

En versiones anteriores de la TTCN se definió un enunciado (*statement*) 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 sucesión de pruebas.

Para limitar el tiempo de ejecución del caso de prueba se recomienda ahora que los realizadores 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.

E.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 (*external*)] 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 el § 10.2.3.3. 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. Pueden utilizarse 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.

e) 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.

f) Los cuadros de constricción de tipo estructurado sustituyen a los cuadros de campos genéricos de las versiones anteriores de la TTCN.

g) Se suprime el concepto de valores genéricos.

h) En el anexo D se presentan ejemplos.

E.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 pueden abreviarse los nombres de las ASP y PDU cuando se utilicen en líneas de evento. Esta facilidad se denomina ahora alias.

E.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.

E.9 *Asignación en caso de eventos SEND*

La TTCN permite sobreescribir valores de constricción antes de un evento SEND en un enunciado de 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 sucesión 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.

E.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 E.1 – 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 ASP 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 F

(a la Recomendación X.292)

Lista de números de producción en BNF

F.1 *Introducción*

En este anexo se presenta un índice alfabético de las producciones en BNF que aparecen en el anexo A. el índice da, para cada una de las producciones, una referencia mediante el número de producción (no el número de página).

A

B

C

D

O

P

Q

R

S

T

U

V

W

ANEXO G

(a la Recomendación X.292)

Índice

(Este anexo no es parte integrante de esta Recomendación)

G.1 *Introducción*

En este anexo se presenta un índice alfabético de términos y abreviaturas en inglés utilizados en la norma ISO/CEI 9646 (1991), parte 3. Para cada término o abreviatura se da un conjunto de referencias en términos de números de cláusula, de figura y cuadro, ya sea del cuerpo principal o de los anexos de la Parte 3. El significado de cada referencia se indica como sigue:

a) las definiciones de términos y abreviaturas aparecen en **negrita**;

b) la utilización principal del término o abreviatura figura en *cursiva*;

c) las demás utilizaciones se indican en escritura normal.

G.2 *Índice*

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

Reemplazada por una versión más reciente