Unión Internacional de Telecomunicaciones

UIT-T X.680

(07/2002)

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

SERIE X: REDES DE DATOS, COMUNICACIONES DE SISTEMAS ABIERTOS Y SEGURIDAD

Gestión de redes de interconexión de sistemas abiertos y aspectos de sistemas – Notación de sintaxis abstracta uno

---

Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de la notación básica

Recomendación UIT-T X.680

UIT-T

Unión Internacional de Telecomunicaciones

RECOMENDACIONES UIT-T DE LA SERIE X

REDES DE DATOS, COMUNICACIONES DE SISTEMAS ABIERTOS Y SEGURIDAD

REDES PÚBLICAS DE DATOS	
Servicios y facilidades	X.1–X.19
Interfaces	X.20–X.49
Transmisión, señalización y conmutación	X.50–X.89
Aspectos de redes	X.90–X.149
Mantenimiento	X.150–X.179
Disposiciones administrativas	X.180–X.199
INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Modelo y notación	X.200–X.209
Definiciones de los servicios	X.210–X.219
Especificaciones de los protocolos en modo conexión	X.220–X.229
Especificaciones de los protocolos en modo sin conexión	X.230–X.239
Formularios para declaraciones de conformidad de implementación de protocolo	X.240–X.259
Identificación de protocolos	X.260–X.269
Protocolos de seguridad	X.270–X.279
Objetos gestionados de capa	X.280–X.289
Pruebas de conformidad	X.290–X.299
INTERFUNCIONAMIENTO ENTRE REDES	
Generalidades	X.300–X.349
Sistemas de transmisión de datos por satélite	X.350–X.369
Redes basadas en el protocolo Internet	X.370–X.379
SISTEMAS DE TRATAMIENTO DE MENSAJES	X.400–X.499
DIRECTORIO	X.500–X.599
GESTIÓN DE REDES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS Y ASPECTOS DE SISTEMAS	
Gestión de redes	X.600–X.629
Eficacia	X.630–X.639
Calidad de servicio	X.640–X.649
Denominación, direccionamiento y registro	X.650–X.679
Notación de sintaxis abstracta uno	**X.680–X.699**
GESTIÓN DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Marco y arquitectura de la gestión de sistemas	X.700–X.709
Servicio y protocolo de comunicación de gestión	X.710–X.719
Estructura de la información de gestión	X.720–X.729
Funciones de gestión y funciones de arquitectura de gestión distribuida abierta	X.730–X.799
SEGURIDAD	X.800–X.849
APLICACIONES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Compromiso, concurrencia y recuperación	X.850–X.859
Procesamiento de transacciones	X.860–X.879
Operaciones a distancia	X.880–X.889
Aplicaciones genéricas de la notación de sintaxis abstracta uno	X.890–X.899
PROCESAMIENTO DISTRIBUIDO ABIERTO	X.900–X.999
SEGURIDAD DE LAS TELECOMUNICACIONES	X.1000–

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

NORMA INTERNACIONAL ISO/CEI 8824-1
RECOMENDACIÓN UIT-T X.680

Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de la notación básica

Resumen

La presente Recomendación | Norma Internacional proporciona una notación llamada notación de sintaxis abstracta uno (ASN.1) para la definición de la sintaxis de datos de información. Se definen en ella varios tipos de datos sencillos y se especifica una notación para hacer referencia a esos tipos y especificar valores de los mismos.

La notación ASN.1 puede aplicarse cuando sea necesario definir la sintaxis abstracta de la información sin constreñir de ningún modo la manera de codificar la información para la transmisión.

Orígenes

La Recomendación UIT-T X.680 fue aprobada el 14 de julio de 2002 por la Comisión de Estudio 17 (2001-2004) del UIT-T por el procedimiento de la Recomendación UIT-T A.8. Se publica también un texto idéntico como Norma Internacional ISO/CEI 8824-1.

PREFACIO

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

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

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

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

NOTA

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

La observancia de esta Recomendación es voluntaria. Ahora bien, la Recomendación puede contener ciertas disposiciones obligatorias (para asegurar, por ejemplo, la aplicabilidad o la interoperabilidad), por lo que la observancia se consigue con el cumplimiento exacto y puntual de todas las disposiciones obligatorias. La obligatoriedad de un elemento preceptivo o requisito se expresa mediante las frases "tener que, haber de, hay que + infinitivo" o el verbo principal en tiempo futuro simple de mandato, en modo afirmativo o negativo. El hecho de que se utilice esta formulación no entraña que la observancia se imponga a ninguna de las partes.

PROPIEDAD INTELECTUAL

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

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

ÍNDICE

Introducción

Esta Recomendación | Norma Internacional presenta una notación normalizada para la definición de tipos y valores de datos. Un *tipo de dato* (o simplemente *tipo*) es una categoría de información (por ejemplo, numérica, textual, de imagen fija o de vídeo). Un *valor de datos* (o simplemente *valor*) es un ejemplar de un tipo. En la presente Recomendación | Norma Internacional se definen varios tipos básicos y sus valores correspondientes, así como reglas para combinarlos en tipos y valores más complejos.

En algunas arquitecturas de protocolo, cada mensaje se especifica como el valor binario de una secuencia de octetos. Sin embargo, quienes preparan las normas necesitan definir tipos de datos muy complejos para llevar sus mensajes, con independencia de su representación binaria. Para especificar estos tipos de datos precisan una notación que no necesariamente determine la representación de cada valor. La ASN.1 es esa notación. Esta notación se complementa mediante la especificación de uno o más algoritmos llamados *reglas de codificación* que determinan el valor de los octetos que llevan la semántica de la aplicación (llamada *sintaxis de transferencia*). La Rec. UIT-T X.690 | ISO/CEI 8825-1, la Rec. UIT-T X.691 | ISO/CEI 8825-2 y la Rec. UIT-T X.693 | ISO/CEI 8825-4 especifican tres familias de reglas de codificación normalizadas, llamadas *reglas de codificación básica (BER)*, *reglas de codificación paquetizada (PER)* y *reglas de codificación XML (XER)*.

Algunos usuarios desean redefinir sus protocolos heredados utilizando la ASN.1, pero no pueden utilizar reglas de codificación normalizadas porque necesitan retener sus representaciones binarias existentes. Otros usuarios desean tener un control más completo sobre la disposición exacta de los bits en la línea (la sintaxis de transferencia). De estos requisitos se ocupa la Rec. UIT-T X.692 | ISO/CEI 8825-3 que especifica una *notación de control de codificación* (*ECN*) para ASN.1. La ECN permite a los diseñadores especificar formalmente la sintaxis abstracta de un protocolo utilizando ASN.1, pero para tener a continuación (si así lo desean) control completo o parcial de los bits de la línea estableciendo una especificación de ECN de acompañamiento (que puede hacer referencia a reglas de codificación normalizadas para algunas partes de la codificación).

Una técnica muy general para definir un tipo complicado a nivel abstracto consiste en definir un pequeño número de *tipos simples* definiendo todos los valores posibles de tipos simples, para combinarlos después de diversas maneras. A continuación se indican algunas maneras de definir nuevos tipos:

a) dada una lista (ordenada) de tipos existentes, se puede formar un valor como una secuencia (ordenada) de valores, formada por uno de cada uno de los tipos existentes; la colección de todos los valores posibles obtenidos de esta manera es un nuevo tipo (si todos los tipos existentes en la lista son distintos, este mecanismo puede ampliarse para permitir que algunos valores no sean incluidos en aquélla);

b) dado un conjunto desordenado de tipos existentes (distintos), se puede formar un valor como conjunto (desordenado) de valores, formado por uno de cada uno de los tipos existentes; la colección de todos los conjuntos posibles de valores desordenados obtenidos de esta manera es un nuevo tipo (también en este caso puede ampliarse el mecanismo para permitir la omisión de algunos valores);

c) dado un tipo existente único, puede formarse un valor como una lista (ordenada) o un conjunto (desordenado) de cero, uno o más valores del tipo existente; la colección de todas las listas o conjuntos de valores posibles obtenidos de esta manera es un nuevo tipo;

d) dada una lista de tipos (distintos), se puede escoger un valor cualquiera de esos tipos distintos; el conjunto de todos los posibles valores obtenidos de esta manera es un nuevo tipo;

e) dado un tipo, se puede formar un nuevo tipo como un subconjunto de dicho tipo utilizando alguna relación de estructura o de orden entre los valores.

Un aspecto importante de esta forma de combinar tipos es que las reglas de codificación deben reconocer las construcciones combinantes, proporcionando codificaciones inequívocas de la colección de valores de los tipos básicos. Así pues, a todo tipo básico definido utilizando la notación especificada en esta Recomendación | Norma Internacional se le asigna un *rótulo* (*tag*) para ayudar a la codificación inequívoca de los valores.

Los rótulos están destinados, sobre todo, a ser utilizados por la máquina, y no son esenciales para la notación destinada al ser humano, definida en esta Recomendación | Norma Internacional. No obstante, cuando es necesario exigir que ciertos tipos sean distintos, esto se expresa especificando que tengan rótulos distintos. Por esta razón, la asignación de rótulos es una parte importante de la utilización de esta notación, aunque (desde 1994) es posible especificar la asignación automática de rótulos.

NOTA 1 – En esta Recomendación | Norma Internacional, se asignan valores de rótulos a todos los tipos simples y mecanismos de construcción. Las restricciones impuestas a la utilización de la notación garantizan que los rótulos pueden utilizarse en la transferencia para identificar de manera inequívoca los valores.

Una especificación ASN.1 se producirá inicialmente con un conjunto de tipos ASN.1 totalmente definidos. En una fase posterior, sin embargo, es posible que sea necesario cambiar esos tipos (normalmente añadiendo componentes suplementarios a una secuencia o tipo conjunto). Para que sea posible de modo que las implementaciones que utilizan las definiciones de tipos antiguos puedan interfuncionar con implementaciones que utilizan las definiciones de tipos

nuevos, es necesario que las reglas de codificación den el soporte apropiado. La notación ASN.1 soporta la inclusión de un *marcador de extensión* en varios tipos. Ello señala a las reglas de codificación la intención del diseñador de que ese tipo forme parte de una serie de tipos relacionados (es decir, versiones del mismo tipo inicial) llamada *serie de extensión*, y que se necesitan las reglas de codificación para permitir la transferencia de información entre implementaciones que utilizan tipos diferentes que están relacionados porque forman parte de la misma serie de extensión.

Las cláusulas 10 a 31 (inclusive) definen los tipos simples soportados por ASN.1 y especifican la notación que ha de emplearse para referenciar tipos simples y para definir nuevos tipos utilizando aquéllos. Las cláusulas 10 a 31 especifican también la notación que ha de emplearse para especificar valores de tipos definidos mediante ASN.1. Se proporcionan dos notaciones de valor. La primera se denomina la notación de valor ASN.1 básica, y ha formado parte de la notación ASN.1 desde que se introdujo por primera vez. La segunda se denomina la notación de valor ASN.1 XML, y proporciona una notación de valor que utiliza el lenguaje de marcaje extensible (XML, *extensible markup language*).

NOTA 2 – La notación de valor XML proporciona un medio de representación de valores ASN.1 utilizando XML. Por consiguiente, una definición de tipo ASN.1 también especifica la estructura y contenido de un elemento XML. Esto permite que ASN.1 sea un lenguaje de esquema simple para XML.

Las cláusulas 33 a 34 (inclusive) definen los tipos soportados por la ASN.1 para llevar dentro de ellos la codificación completa de los tipos ASN.1.

Las cláusulas 35 a 40 (inclusive) definen los tipos cadenas de caracteres.

Las cláusulas 41 a 44 (inclusive) definen ciertos tipos que se consideran de utilidad general, pero que no requieren reglas de codificación adicionales.

Las cláusulas 45 a 47 (inclusive) definen una notación que permite definir subtipos a partir de los valores de un tipo parent (progenitor).

La cláusula 48 define una notación que permite que los tipos ASN.1 especificados en una "versión 1" se identifiquen con probabilidad de extenderse a la "versión 2", y que las adiciones que se efectúen en versiones subsiguientes se enumeren por separado y se identifiquen por su número de versión.

La cláusula 49 define una notación que permite que las definiciones de tipo ASN.1 incluyan una indicación del tratamiento de errores que se pretende si se reciben codificaciones para valores que caen fuera de los especificados en la definición normalizada actual.

El anexo A es parte integrante de la presente Recomendación | Norma Internacional y especifica expresiones ASN.1 normales.

El anexo B es parte integrante de la presente Recomendación | Norma Internacional y especifica reglas de compatibilidad de tipos y valores.

El anexo C es parte integrante de la presente Recomendación | Norma Internacional y registra valores de identificador de objeto y de descriptor de objeto asignados en la serie de Recomendaciones | Normas Internacionales sobre ASN.1.

El anexo D no es parte integrante de la presente Recomendación | Norma Internacional y describe los arcos de nivel superior del árbol de registro de los identificadores de objetos.

El anexo E no es parte integrante de la presente Recomendación | Norma Internacional e incluye ejemplos y consejos acerca de la utilización de la notación ASN.1.

El anexo F no es parte integrante de la presente Recomendación | Norma Internacional sino un texto didáctico sobre las cadenas de caracteres ASN.1.

El anexo G no es parte integrante de la presente Recomendación | Norma Internacional sino un texto didáctico sobre el modelo ASN.1 de extensión de tipo.

El anexo H no es parte integrante de la presente Recomendación | Norma Internacional y proporciona un resumen de la ASN.1 con la notación de la cláusula 5.

NORMA INTERNACIONAL
RECOMENDACIÓN UIT-T

Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de la notación básica

1 Alcance

Esta Recomendación | Norma Internacional proporciona una notación normalizada, llamada notación de sintaxis abstracta uno (ASN.1), que se utiliza para la definición de tipos de datos, valores y constricciones a los tipos de datos.

La presente Recomendación | Norma Internacional:

- define varios tipos simples con sus rótulos, y especifica una notación para referenciar estos tipos y para especificar los valores de estos tipos;
- define mecanismos para construir nuevos tipos a partir de tipos más básicos, y especifica una notación para definir tales tipos y asignarles rótulos y para especificar valores de estos tipos;
- define juegos de caracteres (por medio de referencias a otras Recomendaciones y/o Normas Internacionales) para uso dentro de la ASN.1.

La notación ASN.1 puede aplicarse cuando sea necesaria para definir la sintaxis abstracta de información.

La notación ASN.1 también es referenciada por otras normas que definen reglas de codificación para los tipos ASN.1.

2 Referencias normativas

Las siguientes Recomendaciones y Normas Internacionales contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación | Norma Internacional. Al efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y Normas son objeto de revisiones, por lo que se preconiza que los participantes en acuerdos basados enla presente Recomendación | Norma Internacional investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y las normas citadas a continuación. Los miembros de la CEI y de la ISO mantienen registros de las Normas Internacionales actualmente vigentes. La Oficina de Normalización de las Telecomunicaciones de la UIT mantiene una lista de las Recomendaciones UIT-T actualmente vigentes.

2.1 Recomendaciones | Normas Internacionales idénticas

- Recomendación CCITT X.660 (1992) | ISO/CEI 9834-1:1993*, Tecnología de la información – Interconexión de sistemas abiertos – Procedimientos para la operación de autoridades de registro para interconexión de sistemas abiertos – Procedimientos generales: (más enmiendas)*.
- Recomendación UIT-T X.681 (2002) | ISO/CEI 8824-2:2002, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de objetos de información*.
- Recomendación UIT-T X.682 (2002) | ISO/CEI 8824-3:2002, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de constricciones*.
- Recomendación UIT-T X.683 (2002) | ISO/CEI 8824-4:2002, *Tecnología de la información – Notación de sintaxis abstracta uno: Parametrización de especificaciones de notación de sintaxis abstracta uno*.
- Recomendación UIT-T X.690 (2002) | ISO/CEI 8825-1:2002, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno: Especificación de las reglas de codificación básica, de las reglas de codificación canónica y de las reglas de codificación distinguida*.
- Recomendación UIT-T X.691 (2002) | ISO/CEI 8825-2:2002, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno: Especificación de las reglas de codificación compactada*.
- Recomendación UIT-T X.692 (2002) | ISO/CEI 8825-3:2002, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno: Especificación de la notación de control de codificación*.

– Recomendación UIT-T X.693 (2001) | ISO/CEI 8825-4:2002, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno: Reglas de codificación del lenguaje de marcaje extensible*.

2.2 Referencias adicionales

– Recomendación UIT-R TF.460-5 (1997), *Emisiones de frecuencias patrón y señales horarias*.

– Recomendación CCITT T.100 (1988), *Intercambio de información internacional para el Videotex interactivo*.

– Recomendación UIT-T T.101 (1994), *Interfuncionamiento internacional de servicios Videotex*.

– ISO *International Register of Coded Character Sets to be used with Escape Sequences*.

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

– ISO/CEI 2022:1994, *Information technology – Character code structure and extension techniques*.

– ISO/CEI 6523:1998, *Data interchange – Structures for the identification of organizations*.

– ISO/CEI 7350:1991, *Information technology – Registration of repertoires of graphic characters from ISO/CEI 10367*.

– ISO 8601:2000, *Data elements and interchange formats – Information interchange – Representation of dates and times*.

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

– The Unicode Standard, Version 3.2.0:2002. The Unicode Consortium. (Reading, MA, Addison-Wesley)

NOTA 1 – La referencia anterior se incluye debido a que contiene los nombres de los caracteres de control.

– W3C XML 1.0:2000, *Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation, Copyright © [6 de octubre de 2000] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/2000/REC-xml-20001006.*

NOTA 2 – La referencia a un documento en el marco de esta Recomendación | Norma Internacional no le confiere al mismo carácter de Recomendación o Norma Internacional.

3 Definiciones

A los efectos de la presente Recomendación | Norma Internacional, se aplican las siguientes definiciones.

3.1 Especificación de los objetos de información

Esta Recomendación | Norma Internacional utiliza los siguientes términos definidos en la Rec. UIT-T X.681 | ISO/CEI 8824-2:

a) objeto de información;

b) clase de objeto de información;

c) conjunto de objetos de información;

d) tipo ejemplar de;

e) tipo campo de clase de objeto.

3.2 Especificación de constricción

Esta Recomendación | Norma Internacional utiliza los siguientes términos definidos en la Rec. UIT-T X.682 | ISO/CEI 8824-3:

a) constricción de relación de componentes;

b) constricción de tabla.

3.3 Parametrización de la especificación de ASN.1

Esta Recomendación | Norma Internacional utiliza los siguientes términos definidos en la Rec. UIT-T X.683 | ISO/CEI 8824-4:

a) tipo parametrizado;

b) valor parametrizado.

3.4 Estructura para la identificación de organizaciones

Esta Recomendación | Norma Internacional utiliza los siguientes términos definidos en ISO/CEI 6523:

a) organización emisora;

b) código de organización;

c) designador de indicativo internacional.

3.5 Conjunto universal de caracteres codificados de octeto múltiple (UCS)

Esta Recomendación | Norma Internacional utiliza los siguientes términos definidos en ISO/CEI 10646-1:

a) plano plurilingüe básico (BMP, *basic multilingual plane*);

b) célula;

c) carácter combinante;

d) símbolo gráfico;

e) grupo;

f) subconjunto limitado;

g) plano;

h) fila;

i) subconjunto seleccionado.

3.6 Definiciones adicionales

3.6.1 carácter abstracto: Valor abstracto que se emplea para la organización, control y representación de datos textuales.

NOTA – En el anexo F se presenta una descripción más completa del término carácter abstracto.

3.6.2 valor abstracto: Valor cuya definición se basa sólo en el tipo que se utiliza para llevar alguna semántica, con independencia de cómo se representa en cualquier codificación.

NOTA – Los valores de tipo entero, tipo booleano, tipo cadena de caracteres y tipo que sea una secuencia (o una alternativa) de un entero y un booleano, son ejemplos de valores abstractos.

3.6.3 conjunto (o juego) de caracteres ASN.1: El juego de caracteres, especificado en la cláusula 10, que se utiliza en la notación ASN.1.

3.6.4 especificación en notación de sintaxis abstracta uno: Colección de uno o más módulos ASN.1.

3.6.5 tipo asociado: Un tipo que se utiliza solamente para definir la notación de valor y subtipo de un tipo.

NOTA – En la presente Recomendación | Norma Internacional se definen tipos asociados cuando es necesario establecer claramente que puede haber una diferencia significativa entre el modo de definir el tipo en ASN.1 y el modo de codificarlo. Los tipos asociados no aparecen en las especificaciones de usuario.

3.6.6 tipo cadena de bits: Un tipo simple cuyos valores distinguidos son una secuencia ordenada de cero, uno o más bits.

NOTA – Cuando es necesario llevar codificaciones incrustadas de un valor abstracto, se desaconseja la utilización de un tipo cadena de bits (o cadena de octetos) sin una limitación de contenido (véase la Rec. UIT-T X.682 | ISO/CEI 8824-3, cláusula 11). De lo contrario, el uso del tipo pdv incrustado (véase la cláusula 33) proporcionará un mecanismo más flexible, que posibilita el anuncio de la sintaxis abstracta y de la codificación del valor abstracto incrustado.

3.6.7 tipo booleano: Un tipo simple con dos valores distinguidos.

3.6.8 propiedad de carácter: Conjunto de información asociada a una célula de una tabla que define un repertorio de caracteres.

NOTA – La información incluirá normalmente algunos o la totalidad de los siguientes elementos:

a) un símbolo gráfico;

b) un nombre de carácter;

c) la definición de funciones asociadas al carácter cuando se utiliza en entornos particulares;

d) si representa un dígito;

e) un carácter asociado que difiere sólo en el tipo de letra (mayúscula/minúscula).

3.6.9 sintaxis abstracta de caracteres: Cualquier sintaxis abstracta cuyos valores se especifican como el conjunto de cadenas de caracteres de cero, uno o más caracteres tomados de una colección de caracteres especificada.

3.6.10 repertorio de caracteres: Caracteres de un conjunto de caracteres sin implicación alguna sobre cómo se codifican.

3.6.11 tipos cadenas de caracteres: Tipos simples cuyos valores son cadenas de caracteres tomados de algún juego de caracteres definido.

3.6.12 sintaxis de transferencia de caracteres: Cualquier sintaxis de transferencia para una sintaxis abstracta de caracteres.

NOTA – La ASN.1 no soporta sintaxis de transferencias de caracteres que no codifiquen todas las cadenas de caracteres como un múltiplo entero de 8 bits.

3.6.13 tipos elección: Tipos definidos por referencia a una lista de tipos diferentes; cada valor del tipo elección deriva del valor de uno de los tipos componentes.

3.6.14 tipo componente: Uno de los tipos referenciados cuando se define **`CHOICE`** (elección), **`SET`** (conjunto), **`SEQUENCE`** (secuencia), **`SET OF`** (conjunto de) o **`SEQUENCE OF`** (secuencia de).

3.6.15 constricción: Una notación que puede utilizarse asociada a un tipo, para definir un subtipo del mismo.

3.6.16 constricción de contenido: Constricción sobre un tipo de cadena de bits o de cadena de octetos que especifica que el contenido debe ser una codificación de un tipo ASN.1 especificado, o que deben utilizarse procedimientos específicos para producir y procesar los contenidos.

3.6.17 caracteres de control: Los que aparecen en algunos repertorios de caracteres con un nombre (y quizá una función definida en relación con determinados entornos) pero a los que no se ha asignado un símbolo gráfico ni son caracteres de espaciado.

NOTA – HORIZONTAL TABULATION (tabulación horizontal) (9) y LINE FEED (cambio de renglón) (10) son ejemplos de caracteres de control a los que se les ha asignado una función de formateo en un entorno de impresión. DATA LINK ESCAPE (escape de transmisión) (16) es un ejemplo de carácter de control al que se ha asignado una función en un entorno de comunicaciones.

3.6.18 Tiempo Universal Coordinado (UTC, *coordinated universal time*): La escala de tiempo (u horaria) mantenida por el Bureau International de l'Heure (Oficina Internacional de la Hora) que constituye la base de una diseminación coordinada de frecuencias patrón y señales horarias.

NOTA 1 – La fuente de esta definición es la Rec. UIT-R TF.460-5. El UIT-R también ha definido UTC como el acrónimo para Tiempo Universal Coordinado.

NOTA 2 – UTC y tiempo medio (u hora media) de Greenwich (GMT, *greenwich mean time*) son dos patrones horarios alternativos que, para la mayoría de los fines prácticos, determinan la misma hora.

3.6.19 elemento: Valor de un tipo gobernante o de un objeto información de una clase de objeto de información gobernante, que se distingue de los demás valores del mismo tipo o de objetos de información de la misma clase, respectivamente.

3.6.20 conjunto de elementos: Conjunto de elementos que representan valores de un tipo gobernante u objetos de información de una clase gobernante.

NOTA – La clase gobernante se define en la Rec. UIT-T X.681 | ISO/CEI 8824-2, 3.4.7.

3.6.21 tipo pdv inscrustado: Un tipo cuyo conjunto de valores es formalmente la unión de los conjuntos de valores en todas las sintaxis abstractas posibles. Este tipo puede utilizarse en una especificación ASN.1 que desee llevar en su protocolo un valor abstracto cuyo tipo puede ser definido externamente a esa especificación ASN.1. Lleva una identificación de la sintaxis abstracta (el tipo) del valor abstracto que se transporta, así como una identificación de las reglas de codificación utilizadas para codificar ese valor abstracto.

3.6.22 codificación: El esquema de bits resultante de la aplicación de un conjunto de reglas de codificación a un valor abstracto.

3.6.23 reglas de codificación (en notación de sintaxis abstracta uno): Reglas que especifican la representación, durante una transferencia, de los valores de tipos ASN.1. Permiten también recuperar los valores a partir de la representación, si se conoce el tipo.

NOTA – A los fines de la especificación de reglas de codificación, las diversas notaciones referenciadas de tipo (y de valor) que pueden proporcionar notaciones alternativas para tipos (y valores) incorporados no son relevantes.

3.6.24 tipos enumerados: Tipos simples a cuyos valores se les da identificadores distintos como parte de la notación de tipo.

3.6.25 adición de extensión: Una de las notaciones añadidas en una serie de extensión. Para los tipos conjunto, secuencia y elección, cada adición de extensión es la adición de un grupo adición de extensión único o de un tipo componente único. Para tipos enumerados es la adición de una enumeración posterior única. Para una constricción es la adición de un (solo) elemento subtipo.

NOTA – Las adiciones de extensión están ordenadas tanto textualmente (siguiendo al marcador de extensión) como lógicamente (con valores de enumeración crecientes, y, en el caso de alternativas de **`CHOICE`** (elección), con rótulos crecientes).

3.6.26 grupo de adición de extensión: Uno o más componentes de un tipo de conjunto, secuencia o elección agrupados dentro de corchetes de versión. El grupo de adición de extensión se utiliza para identificar claramente los componentes de un tipo conjunto, secuencia o elección que fueron añadidos en una versión particular de un módulo ASN.1, y para poder identificar la versión con un número entero simple.

3.6.27 tipo adición de extensión: Tipo contenido dentro de un grupo adición de extensión o tipo componente único que es en sí mismo una adición de extensión (en cuyo caso no está contenido dentro de un grupo de adición de extensión).

3.6.28 constricción extensible: Una constricción de subtipo con un marcador de extensión en el nivel más exterior, o que es extensible mediante la utilización de aritmética de conjuntos con conjuntos de valores extensibles.

3.6.29 punto de inserción de extensión (o punto de inserción): Ubicación dentro de una definición de tipo donde se insertan las adiciones de extensión. Esta ubicación es el final de la notación de tipo inmediatamente anterior en la serie de extensión si hay una sola elipsis en la definición de tipo, o inmediatamente antes de la segunda elipsis si hay un par de marcadores de extensión en la definición del tipo.

NOTA – Puede haber como máximo un punto de inserción dentro de los componentes de cualquier tipo de elección, secuencia, o conjunto.

3.6.30 marcador de extensión: Una bandera sintáctica (puntos suspensivos) que está incluida en todos los tipos que forman parte de una serie de extensión.

3.6.31 par de marcadores de extensión: Par de marcadores de extensión entre los que se insertan las adiciones de extensión.

3.6.32 relacionados en extensión: Dos tipos con la misma raíz de extensión, uno de los cuales se ha creado añadiendo cero o más adiciones de extensión al otro.

3.6.33 raíz de extensión: Tipo extensible que es el primer tipo de una serie de extensión. Incluye, bien el marcador de extensión sin notación adicional que no sea comentarios y un espacio en blanco entre el marcador de extensión y la correspondiente "}" o ")", o bien un par de marcadores de extensión sin notación adicional que no sea exclusivamente una coma, comentarios y espacios en blanco entre los marcadores de extensión.

NOTA – Sólo una raíz de extensión puede ser el primer tipo en una serie de extensión.

3.6.34 serie de extensión: Serie de tipos de notación de sintaxis abstracta uno que puede ordenarse de manera que cada tipo sucesivo de la serie esté formado por adición de texto en el punto de inserción de la extensión.

3.6.35 tipo extensible: Tipo con un marcador de extensión o al que se haya aplicado una constricción extensible.

3.6.36 referencia externa: Referencia de tipo, referencia de valor, objeto de información, referencia de clase, referencia de objeto de información o referencia de conjunto de objetos de información (que puedan parametrizarse), que se define en un módulo distinto de aquel al que se hace referencia y que es referido prefijando el nombre del módulo al elemento referido.

EJEMPLO – **`ModuleName.TypeReference`**

3.6.37 tipo externo: Tipo que es parte de una especificación ASN.1 y que lleva un valor cuyo tipo puede definirse externamente a esa especificación ASN.1. También lleva una identificación del tipo del valor que se transporta.

3.6.38 falso: Uno de los valores distinguidos de tipo booleano (véase también "verdadero").

3.6.39 gobernante (tipo); gobernador: Definición de, o referencia a, un tipo que afecta a la interpretación de una parte de la sintaxis ASN.1, lo que exige que haga referencia a valores del tipo gobernante.

3.6.40 definiciones de tipo idénticas: Dos ejemplares de producción "Type" ASN.1 (véase la cláusula 16) se consideran definiciones de tipo idénticas si, tras realizar las transformaciones especificadas en el anexo B, se convierten en listas ordenadas idénticas de elementos léxicos idénticos (véase la cláusula 11).

3.6.41 tipo entero: Un tipo simple cuyos valores distinguidos son los números enteros positivos y negativos, incluido el cero (como un valor único).

NOTA – Cuando el rango de valores de un entero se vea limitado por reglas de codificación particulares, tales limitaciones se establecen de tal modo que no afecten a ningún usuario de ASN.1.

3.6.42 elemento léxico: Secuencia, con nombre, de caracteres del juego de caracteres ASN.1, especificado en la cláusula 11, que se utiliza para formar la notación ASN.1.

3.6.43 módulo: Uno o más ejemplares de la utilización de la notación ASN.1 para definición de tipo, valor, conjunto de valores, clase de objeto información, objeto información y conjunto de objetos de información (así como sus variantes parametrizadas), encapsulados utilizando la notación de módulo ASN.1 (véase la cláusula 12).

NOTA – Los términos clase de objeto información (etc.) se especifican en la Rec. UIT-T X.681 | ISO/CEI 8824-2, y su parametrización se especifica en la Rec. UIT-T X.683 | ISO/CEI 8824-4.

3.6.44 tipo nulo: Tipo simple que consta de un solo valor, que también se llama nulo.

3.6.45 objeto: Elemento de información, definición o especificación, bien definido, que requiere un nombre para identificar su utilización en un ejemplar de comunicación.

NOTA – Puede tratarse de un objeto información como el que se define en la Rec. UIT-T X.681 | ISO/CEI 8824-2.

3.6.46 tipo descriptor de objeto: Tipo cuyos valores distinguidos son texto legible por el ser humano y que proporcionan una breve descripción de un objeto (véase 3.6.45).

NOTA – Un valor de descriptor de objeto está usualmente asociado a un solo objeto. Únicamente un valor de identificador de objeto identifica inequívocamente un objeto.

3.6.47 identificador de objeto: Valor universal único asociado a un objeto para identificarlo inequívocamente.

3.6.48 tipo identificador de objeto: Tipo simple cuyos valores son el conjunto de todos los identificadores de objeto atribuidos de conformidad con las reglas de las Recomendaciones de la serie UIT-T X.660 | ISO/CEI 9834.

NOTA – Las reglas de la Rec. UIT-T X.660 | ISO/CEI 9834-1 permiten a una amplia gama de autoridades asociar independientemente identificadores de objetos con objetos.

3.6.49 tipo cadena de octetos: Tipo simple cuyos valores distinguidos son una secuencia ordenada de cero, uno o más octetos, estando formado cada octeto por una secuencia ordenada de 8 bits.

3.6.50 interconexión de sistemas abiertos: Arquitectura de comunicación de ordenadores que aporta varios términos empleados en esta Recomendación | Norma Internacional precedidos de la abreviatura "OSI".

NOTA – El significado de esos términos puede obtenerse de las Recomendaciones de la serie UIT-T X.200 y de las Normas ISO/CEI equivalentes si resulta necesario. Los términos sólo pueden aplicarse si se emplea ASN.1 en un entorno OSI.

3.6.51 notación de tipo abierto: Notación ASN.1 utilizada para designar un conjunto de valores procedentes de más de un tipo ASN.1.

NOTA 1 – En el cuerpo de esta Recomendación | Norma Internacional se utiliza el término "tipo abierto" con el mismo significado que "notación de tipo abierto".

NOTA 2 – Todas las reglas de codificación ASN.1 proporcionan codificaciones inequívocas de los valores de un solo tipo ASN.1. No necesariamente son inequívocas, las codificaciones que proporcionan para "notación de tipo abierto" que lleva valores de tipos ASN.1 que normalmente no se determinan en el momento de la especificación. Es preciso conocer el tipo de valor que se codifica en la "notación de tipo abierto" para que se pueda determinar inequívocamente el valor abstracto de ese campo.

NOTA 3 – En esta Recomendación | Norma Internacional, la única notación que es una notación de tipo abierto es el "ObjectClassFieldType" especificado en la Rec. UIT-T X.681 | ISO/CEI 8824-2, cláusula 14, donde el "FieldName" denota un campo de tipo o un campo de valor de tipo variable.

3.6.52 tipo progenitor (de un subtipo): El tipo que se constriñe cuando se define un subtipo, y que gobierna la notación de subtipo.

NOTA – El tipo progenitor puede ser, a su vez, subtipo de algún otro tipo.

3.6.53 producción: Parte de la notación formal (denominada asimismo gramática o forma Backus-Naur, BNF) utilizada para especificar ASN.1.

3.6.54 tipo real: Tipo simple cuyos valores distinguidos (especificados en la cláusula 20) forman parte del conjunto de números reales.

3.6.55 definición recurrente (de un tipo): Conjunto de definiciones ASN.1 que no pueden ser reordenadas de modo tal que todos los tipos utilizados en una construcción se definan antes de la definición de la construcción.

NOTA – En ASN.1 se permiten definiciones recursivas: el usuario de la notación asume la responsabilidad de asegurar que los valores (de los tipos resultantes) que se utilizan tienen una representación finita y que el conjunto de valores asociado al tipo contiene al menos un valor.

3.6.56 identificador de objeto relativo: Valor que identifica un objeto por su posición relativa con respecto a algún identificador de objeto conocido (véase 3.6.47).

3.6.57 tipo identificador de objeto relativo: Tipo simple cuyos valores son el conjunto de todos los identificadores de objetos relativos posibles.

3.6.58 tipo cadena de caracteres restringida: Tipo cadena cuyos caracteres se toman de un repertorio fijo, identificado en la especificación del tipo.

3.6.59 tipos selección: Tipos definidos por referencia a un tipo componente de un tipo choice (elección) y cuyos valores son precisamente los valores de ese tipo componente.

3.6.60 tipos secuencia: Tipos definidos por referencia a una lista ordenada fija de tipos (algunos de los cuales pueden ser declarados como opcionales); cada valor de tipo secuencia es una lista ordenada de valores, uno de cada tipo componente.

NOTA – Cuando un tipo componente está declarado como opcional, no es necesario que un valor del tipo secuencia contenga un valor de ese tipo componente.

3.6.61 tipos secuencia de: Tipos definidos por referencia a un solo tipo componente; cada valor del tipo secuencia de es una lista ordenada de cero, uno o más valores del tipo componente.

3.6.62 aplicación en serie (de constricciones): Aplicación de una constricción a un tipo progenitor que ya está constreñido.

3.6.63 conjunto aritmético: Formación de nuevos conjuntos de valores o de objetos información por medio de las operaciones de unión, intersección y diferencia de conjuntos (uso de **EXCEPT**) como se especifica en 46.2.

NOTA – El término "aritmética de conjuntos" es ajeno al resultado de la aplicación de constricciones en serie.

3.6.64 tipos conjunto: Tipos definidos por referencia a una lista fija, no ordenada, de tipos (algunos de los cuales pueden ser declarados como opcionales); cada valor del tipo conjunto es una lista no ordenada de valores, uno de cada tipo componente.

NOTA – Cuando un tipo componente esté declarado como opcional, no es necesario que un valor del tipo conjunto contenga un valor de ese tipo componente.

3.6.65 tipos conjunto de: Tipos definidos por referencia a un solo tipo componente; cada valor del tipo conjunto de es una lista no ordenada de cero, uno o más valores del tipo componente.

3.6.66 tipos simples: Tipos definidos especificando directamente el conjunto de sus valores.

3.6.67 carácter de espaciado: Carácter de un repertorio destinado a su inclusión entre los caracteres gráficos de impresión pero cuya representación física es un espacio en blanco, no suele considerarse carácter de control (véase 3.6.17).

NOTA – En un repertorio de caracteres puede haber un solo carácter de espaciado o bien varios de distintas anchuras.

3.6.68 subtipo (de un tipo progenitor): Tipo cuyos valores son un subconjunto (o el conjunto completo) de los valores de algún otro tipo (el tipo progenitor).

3.6.69 rótulo: Designación de tipo que se asocia a todo tipo ASN.1.

3.6.70 tipos rotulados: Tipos definidos por referencia a un solo tipo existente y a un rótulo; el nuevo tipo es isomorfo con el tipo existente pero distinto de éste.

3.6.71 rotulación: Sustitución del rótulo existente (que puede ser el rótulo por defecto) de un tipo por un rótulo especificado.

3.6.72 sintaxis de transferencia: Conjunto de cadenas de bits que se utiliza para intercambiar valores abstractos en una sintaxis abstracta, obtenido normalmente por aplicación de reglas de codificación a una sintaxis abstracta.

NOTA – El término "sintaxis de transferencia" es sinónimo de "codificación".

3.6.73 verdadero (true): Uno de los valores distinguidos del tipo booleano (véase también "falso").

3.6.74 tipo: Conjunto de valores con nombre.

3.6.75 nombre de referencia de tipo: Nombre asociado de modo único con un tipo dentro de algún contexto.

NOTA – Se asignan nombres de referencia a los tipos definidos en esta Recomendación | Norma Internacional; estos nombres están universalmente disponibles en ASN.1. Otros nombres de referencia están definidos en otras Recomendaciones | Normas Internacionales y son aplicables solamente en los respectivos contextos de esa Recomendación | Norma Internacional.

3.6.76 tipo cadena de caracteres no restringida: Tipo cuyos valores abstractos son valores de una sintaxis abstracta de caracteres, junto con una identificación de la sintaxis abstracta de caracteres y de la sintaxis de transferencia de caracteres que se utilizará para su codificación.

3.6.77 usuario (de la notación de sintaxis abstracta uno): Individuo u organización que define la sintaxis abstracta de un elemento particular de información utilizando ASN.1.

3.6.78 correspondencia de valores: Relación biunívoca entre valores de dos tipos que permite utilizar la referencia a uno de esos valores como referencia al otro valor. Puede utilizarse, por ejemplo, para especificar subtipos y valores por defecto (véase el anexo B).

3.6.79 nombre de referencia de valor: El asociado de manera única a un valor dentro de algún contexto.

3.6.80 conjunto de valores: Colección de valores de un tipo. Semánticamente equivalente a un subtipo.

3.6.81 corchetes de versión: Par de corchetes adyacentes de apertura y cierre ("`[[`" o "`]]`") utilizados para delimitar el comienzo y el final de un grupo de adición de extensión. El par de corchetes de apertura puede ir seguido del número de versión del grupo de adición de extensión.

3.6.82 número de versión: Número que puede asociarse a un corchete de versión (véase G.1.8).

NOTA – El número de versión no puede añadirse a una adición de extensión que no sea parte de un grupo de adición de extensión, ni a adiciones de extensión de ningún tipo distinto de elección, secuencia, o conjunto.

3.6.83 espacio en blanco: Cualquier acción de formateo que produzca un espacio en una página impresa, tal como espacios o tabuladores.

4 Abreviaturas

A los efectos de esta Recomendación | Norma Internacional se utilizan las siguientes abreviaturas:

ASN.1	Notación de sintaxis abstracta uno (*abstract syntax notation one*)
BER	Reglas de codificación básica de ASN.1 (*basic encoding rules of ASN.1*)
BMP	Plano plurilingüe básico (*basic multilingual plane)*
CEI	Comisión Electrotécnica Internacional
DCC	Indicativo de país para datos (*data country code*)
DNIC	Código de identificación de red de datos (*data network identification code*)
ECN	Notación de control de codificación de ASN.1 (*encoding control notation ASN.1)*
EER	Empresa de explotación reconocida
ICD	Designador de indicativo internacional (*international code designator*)
ISO	Organización Internacional de Normalización (*international organization for standardization*)
OID	Identificador de objeto (*object identifier)*
OSI	Interconexión de sistemas abiertos (*open systems interconnection)*
PER	Reglas de codificación paquetizada de ASN.1 (*packed encoding rules of ASN.1*)
UCS	Conjunto universal de caracteres codificados de octeto múltiple (*universal multiple-octet coded character set*)
UIT-T	Unión Internacional de Telecomunicaciones – Sector de Normalización de las Telecomunicaciones
UTC	Tiempo universal coordinado (*coordinated universal time)*
XML	Lenguaje de marcaje extensible (*extensible markup language)*

5 Notación

5.1 Generalidades

5.1.1 La notación ASN.1 consiste en una secuencia de caracteres del juego de caracteres ASN.1 especificado en la cláusula 10.

5.1.2 Cada vez que se utiliza la notación ASN.1, se agrupan caracteres del conjunto ASN.1 para formar elementos léxicos. La cláusula 11 especifica todas las secuencias de caracteres que forman elementos léxicos y el nombre de cada uno de ellos.

5.1.3 La notación ASN.1 se define en la cláusula 12 (y siguientes), especificando y asignando un nombre a las secuencias de elementos léxicos que forman ejemplares válidos de la notación ASN.1, y la semántica ASN.1 de cada secuencia.

5.1.4 Para especificar las secuencias permitidas de los elementos léxicos, esta Recomendación | Norma Internacional utiliza una notación formal definida en las siguientes subcláusulas.

5.2 Producciones

5.2.1 Todos los elementos léxicos tienen nombre (véase la cláusula 11) y también lo tienen las secuencias permitidas de los elementos léxicos.

5.2.2 Se define una nueva secuencia permitida (más compleja) de elementos léxicos mediante una producción. Ésta utiliza los nombres de los elementos léxicos y de las secuencias permitidas de elementos léxicos y forma una nueva secuencia permitida con nombre de elementos léxicos.

5.2.3 Cada producción consta de las siguientes partes, que ocupan una o varias líneas, y siguen este orden:

a) un nombre para la nueva secuencia permitida de elementos léxicos;

b) los caracteres

::=

c) una o más secuencias alternativas de elementos léxicos, definidas en 5.3, separadas por el carácter

|

5.2.4 Una secuencia de elementos léxicos está presente en la nueva secuencia permitida de elementos léxicos si lo está en una o más de las alternativas. La nueva secuencia permitida de elementos léxicos es referenciada en esta Recomendación | Norma Internacional por el nombre a que se refiere 5.2.3 a).

NOTA – Si la misma secuencia de elementos léxicos aparece en más de una alternativa, cualquier ambigüedad semántica en la notación resultante se resuelve mediante el texto asociado.

5.3 Las colecciones alternativas

5.3.1 Cada una de las alternativas de una producción (véase 5.2.3.c) se especifica mediante una lista de nombres. Cada nombre es el nombre de un elemento léxico o el nombre de una secuencia permitida de elementos léxicos definidos y denominados por alguna otra producción.

5.3.2 La secuencia permitida de elementos léxicos definida por cada alternativa consiste en todas las secuencias obtenidas seleccionando cualquiera de las secuencias (o el elemento léxico) asociada al primer nombre, en combinación con (y seguida de) cualquiera de las secuencias (o el elemento léxico) asociada al segundo nombre, en combinación con (y seguida de) cualquiera de las secuencias (o el elemento léxico) asociada al tercer nombre, y así sucesivamente hasta el último nombre, inclusive, (o elemento léxico) de la alternativa.

5.4 Indicador de no espaciado

Si se inserta entre estos elementos el indicador "**&**" (AMPERSAND) de no espaciado en las secuencias de producción, ni el elemento léxico anterior al indicador ni el siguiente deben estar separados por un espacio en blanco.

NOTA – Este indicador sólo se emplea en las producciones que describen la notación de valor XML. Por ejemplo, se utiliza para especificar que después del elemento léxico "<" debe seguir inmediatamente un nombre de rótulo XML.

5.5 Ejemplo de producción

5.5.1 La producción:

```
ExampleProduction    ::=
        bstring
    |   hstring
    |   "{" IdentifierList "}"
```

asocia el nombre "ExampleProduction" a las secuencias de elementos léxicos siguientes:

a) cualquier "bstring" (un elemento léxico); o

b) cualquier "hstring" (un elemento léxico); o

c) cualquier secuencia de elementos léxicos asociada con "IdentifierList", precedida por un "`{`" y seguida por un "`}`".

NOTA – "`{`" y "`}`" son los nombres de elementos léxicos que contienen los caracteres únicos `{` y `}` (véase 11.26).

5.5.2 En este ejemplo, "IdentifierList" se definiría por una producción posterior, que iría antes o después de la que define "ExampleProduction".

5.6 Disposición

Cada producción utilizada en esta Recomendación | Norma Internacional va precedida y seguida por una línea vacía. Las producciones no contienen líneas vacías. La producción puede ocupar una sola línea o varias. La disposición (layout) no es significativa.

5.7 Recurrencia

Las producciones especificadas en esta Recomendación | Norma Internacional son frecuentemente recurrentes. En tal caso, las producciones deberán aplicarse continuamente hasta que no se generen más secuencias nuevas.

NOTA – En muchos casos, tal aplicación iterativa da lugar a un conjunto infinito de secuencias permitidas de elementos léxicos. Algunas de las secuencias del conjunto, o todas ellas, pueden contener a su vez varios elementos léxicos no acotados. Esto no es un error.

5.8 Referencia a secuencias permitidas de elementos léxicos

Esta Recomendación | Norma Internacional hace referencia a una secuencia permitida de elementos léxicos (parte de la notación ASN.1) mediante el nombre que precede a "::=" en una producción; el nombre va encerrado entre caracteres QUOTATION MARK (34) (") (comillas) para distinguirlo del texto en lenguaje natural, a menos que aparezca como parte de una producción.

5.9 Referencia a un elemento léxico

Esta Recomendación | Norma Internacional hace referencia a un elemento léxico mediante el nombre del elemento léxico; cuando el nombre aparece en texto en lenguaje natural, pudiendo confundirse con dicho texto, el nombre se encierra entre caracteres QUOTATION MARK (34) (") (comillas).

5.10 Notación abreviada

Para obtener producciones más concisas y legibles se utiliza la siguiente notación abreviada en la definición de las secuencias permitidas de elementos léxicos en esta Recomendación | Norma Internacional y asimismo en la Rec. UIT-T X.681 | ISO/CEI 8824-2, la Rec. UIT-T X.682 | ISO/CEI 8824-3 y la Rec. UIT-T X.683 | ISO/CEI 8824-4:

a) un asterisco (*) que sigue a dos nombres, "A" y "B", denota el elemento léxico "vacío" (empty) (véase 11.7) o una de las secuencias permitidas de elementos léxicos asociada con "A", o una serie alternante de una de las secuencias de elementos léxicos asociada con "A" y una de las secuencias de elementos léxicos asociada con "B" que empiezan y terminan con una asociada con "A". Así:

   C ::= A B *

   es equivalente a:

   C ::= D | empty
   D ::= A | A B D

   siendo "D" un nombre auxiliar que no aparece en otra parte de las producciones.

   EJEMPLO – "C ::= A B *" es la notación abreviada de las siguientes alternativas de C:

   empty
   A
   A B A
   A B A B A
   A B A B A B A
   ...

b) un signo más (+) produce los mismos efectos que el asterisco en a), salvo que queda excluido el elemento léxico "vacío". Así:

   E ::= A B +

   es equivalente a:

   E ::= A | A B E

   EJEMPLO – "E ::= A B +" es la notación abreviada de las siguientes alternativas de E:

A
A B A
A B A B A
A B A B A B A
...

c) un signo de interrogación (?) que sigue a un nombre denota, o bien el elemento léxico "vacío" (véase 11.7) o una secuencia permitida de elementos léxicos asociada con "A". Así:

F ::= A ?

es equivalente a:

F ::= empty | A

NOTA – Estas notaciones abreviadas tienen prioridad sobre la yuxtaposición de los elementos léxicos en las secuencias de producción (véase 5.2.2).

5.11 Referencia por valor y tipificación de valores

5.11.1 La notación de asignación de valor ASN.1 permite dar un nombre a un valor de un tipo especificado. Este nombre puede utilizarse cuando se necesite una referencia a dicho valor. En el anexo B se describe y especifica el mecanismo de correspondencia de valores que permite que un nombre de referencia de valor asignado a un tipo de valor determinado pueda identificar un valor de otro tipo (similar). De esta forma puede utilizarse la referencia al primer valor donde se requiera una referencia a un valor del segundo tipo.

5.11.2 En las normas ASN.1 se utiliza texto en inglés corriente para especificar la legalidad (o ilegalidad) de las construcciones en las que interviene más de un tipo. Por lo general, estas especificaciones de legalidad exigen que dos o varios tipos sean "compatibles". Por ejemplo, el tipo utilizado para definir una referencia de valor debe ser "compatible con" el tipo gobernante cuando se usa la referencia de valor. En el anexo B, normativo, se utiliza el concepto de correspondencia de valores para hacer declaraciones precisas sobre la legalidad o ilegalidad de una determinada construcción ASN.1.

6 Modelo de extensión de tipo ASN.1

Cuando se decodifica un tipo de extensión, el decodificador puede detectar:

a) la ausencia de adiciones de extensión esperadas en un tipo secuencia o conjunto; o

b) la presencia de adiciones de extensión arbitrarias no esperadas sobre las definidas (si existen) en un tipo secuencia o conjunto, o de una alternativa desconocida en un tipo elección, o una enumeración desconocida en un tipo enumerado, o de una longitud o valor no esperados de un tipo cuya constricción es extensible.

En términos formales, una sintaxis abstracta definida por un tipo extensible **X** contiene no sólo el valor del tipo **X**, sino también los valores de todos los tipos que están relacionados en extensión con **X**. De esta manera, el proceso de decodificación nunca indica un error cuando se detecta alguna de las situaciones anteriores (a o b). Las medidas a tomar en cada situación son determinadas por el especificador de ASN.1.

NOTA – A menudo estas medidas consistirán en ignorar la presencia de extensiones adicionales no esperadas y en utilizar un valor por defecto o un indicador "falta" para adiciones de extensión esperadas que estén ausentes.

Las adiciones de extensión imprevistas detectadas por un decodificador en un tipo extensible pueden incluirse más tarde en una codificación subsiguiente de dicho tipo (para su transmisión en retorno al emisor o a un tercero), siempre que se utilice la misma sintaxis de transferencia en la transmisión subsiguiente.

7 Requisitos de extensibilidad en reglas de codificación

NOTA – Estos requisitos se aplican a reglas de codificación normalizadas. No se aplican a reglas de codificación definidas utilizando ECN (véase la Rec. UIT-T X.692 | ISO/CEI 8825-3).

7.1 Todas las reglas de codificación ASN.1 permitirán la codificación de valores de un tipo extensible **X** de manera que puedan decodificarse utilizando un tipo extensible **Y** que esté relacionado en extensión con **X**. Además, las reglas de codificación permitirán que los valores decodificados mediante **Y** puedan recodificarse (utilizando **Y**) y decodificarse mediante un tercer tipo extensible **Z** que esté relacionado en extensión con **Y** (y por lo tanto también con **X**).

NOTA – Los tipos **X, Y** y **Z** pueden aparecer en la serie de extensión en cualquier orden.

Si un valor de un tipo extensible **X** se codifica y luego se retransmite (directamente o a través de una aplicación de retransmisión que utiliza un tipo relacionado en extensión **Z**) a otra aplicación que decodifica el valor utilizando el tipo extensible **Y** que está relacionado en extensión con **X**, el decodificador que utiliza el tipo **Y** obtiene un valor abstracto compuesto por:

a) un valor abstracto del tipo raíz de extensión;

b) un valor abstracto de cada adición de extensión que está presente en **X** y en **Y**;

c) una codificación delimitada para cada adición de extensión (si existe) que está en **X** pero no en **Y**.

Las codificaciones en c) podrán ser incluidas en una codificación posterior de un valor **Y**, si la aplicación así lo exige. Dicha codificación debe ser una codificación válida de un valor **X**.

Ejemplo aclaratorio: Si el sistema A utiliza un tipo raíz extensible (tipo **X**) que es un tipo secuencia o un tipo conjunto con una adición de extensión de un tipo entero opcional, mientras que el sistema B utiliza un tipo relacionado en extensión (tipo **Y**) que tiene dos adiciones de extensión cada una de ellas de tipo entero opcional, la transmisión por B de un valor de **Y** que omite el valor entero de la primera adición de extensión y que incluye el de la segunda no debe ser confundido por A con la presencia de (sólo) la primera adición de extensión de **X** que conoce. Además, A tiene que poder recodificar el valor de **X** con un valor presente para el primer tipo entero, seguido del segundo valor entero recibido de B, si así lo requiere el protocolo de aplicación.

7.2 Todas las reglas de codificación ASN.1 especificarán la codificación y decodificación del valor de un tipo enumerado y de un tipo elección de manera que, si un valor transmitido se encuentra en el conjunto de adiciones de extensión común al codificador y al decodificador, se decodifique con éxito; en otro caso el decodificador deberá poder delimitar su codificación e identificarlo como un valor de una adición de extensión (desconocida).

7.3 Todas las reglas de codificación ASN.1 especificarán los tipos de codificación y de decodificación con constricciones extensibles de manera que, si un valor transmitido está en el conjunto de adiciones de extensión común al codificador y al decodificador, se decodifique con éxito, en otro caso el decodificador deberá poder delimitar su codificación e identificarlo como un valor de una adición de extensión (desconocida).

La presencia de adiciones de extensión no afectará en ningún caso a la aptitud para reconocer información posterior cuando un tipo con un marcador de extensión esté anidado dentro de otro tipo.

NOTA 1 – Todas las variantes de las reglas de codificación básica de ASN.1 y de las reglas de codificación empaquetada de ASN.1 cumplen todos estos requisitos. Las reglas de codificación definidas utilizando ECN no satisfacen necesariamente todos estos requisitos, pero pueden hacerlo.

NOTA 2 – PER y BER no identifican el número de versión en la codificación de una adición de extensión. Las codificaciones especificadas utilizando ECN pueden o no proporcionar tal identificación.

8 Rótulos

8.1 Un rótulo se especifica mediante una clase y un número dentro de la clase. La clase es una de las siguientes:

– universal;

– aplicación;

– privada;

– específica del contexto.

8.2 El número es un entero no negativo, especificado en notación decimal.

8.3 Las restricciones a los rótulos asignados por el usuario de ASN.1 se especifican en la cláusula 30.

NOTA – La cláusula 30 incluye la restricción de que los usuarios de esta notación tienen prohibido especificar explícitamente rótulos de clase universal en sus especificaciones ASN.1. No existe ninguna diferencia formal entre la utilización de rótulos de las otras tres clases. Cuando se empleen rótulos de clase aplicación, podrán emplearse también, por lo general, rótulos de clase privada o específica al contexto, dependiendo de la elección y el estilo del usuario. La presencia de las tres clases se debe en buena medida a razones históricas, pero en E.2.12 se dan directrices respecto al modo de emplear normalmente las clases.

8.4 El cuadro 1 resume la asignación de los rótulos de la clase universal que se especifican en esta Recomendación | Norma Internacional.

Cuadro 1 – Asignaciones de rótulos de clase universal

UNIVERSAL 0	Reservado para uso en las reglas de codificación
UNIVERSAL 1	Tipo booleano
UNIVERSAL 2	Tipo entero
UNIVERSAL 3	Tipo cadena de bits
UNIVERSAL 4	Tipo cadena de octetos
UNIVERSAL 5	Tipo nulo
UNIVERSAL 6	Tipo identificador de objeto
UNIVERSAL 7	Tipo descriptor de objeto
UNIVERSAL 8	Tipos externo y ejemplar de
UNIVERSAL 9	Tipo real
UNIVERSAL 10	Tipo enumerado
UNIVERSAL 11	Tipo pdv inscrustado
UNIVERSAL 12	Tipo UTF8String
UNIVERSAL 13	Tipo identificador de objeto relativo
UNIVERSAL 14-15	Reservado para futuras ediciones de esta Recomendación | Norma Internacional
UNIVERSAL 16	Tipos secuencia y secuencia de
UNIVERSAL 17	Tipos conjunto y conjunto de
UNIVERSAL 18-22, 25-30	Tipos cadenas de caracteres
UNIVERSAL 23-24	Tipos tiempo
UNIVERSAL 31-...	Reservado para adiciones a esta Recomendación | Norma Internacional

8.5 Algunas reglas de codificación exigen un orden canónico de los rótulos. En 8.6 se define dicho orden, a efectos de uniformidad.

8.6 El orden canónico de los rótulos se basa en el rótulo exterior de cada tipo y se define de la siguiente manera:

a) los elementos o alternativas con rótulos de la clase universal aparecerán en primer lugar, seguidos de aquellos que tengan rótulos de clase aplicación, seguidos a su vez de aquellos que tengan rótulos específicos del contexto, seguidos por último de los de rótulo de clase privada;

b) dentro de cada clase de rótulos, los elementos o alternativas aparecerán en orden ascendente de sus números de rótulo.

9 Utilización de la notación ASN.1

9.1 La notación ASN.1 para una definición de tipo será "Type" (véase 16.1).

9.2 La notación ASN.1 para el valor de un tipo será "Value" (véase 16.7).

NOTA – Por lo general, no es posible interpretar la notación de valor sin conocer el tipo.

9.3 La notación ASN.1 para asignar un tipo a un nombre de referencia de tipo será "TypeAssignment" (véase 15.1), "ValueSetTypeAssignment" (véase 15.6), "ParameterizedTypeAssignment" (véase la Rec. UIT-T X.683 | ISO/CEI 8824-4, 8.2) o "ParameterizedValueSetTypeAssignment" (véase la Rec. UIT-T X.683 | ISO/CEI 8824-4, 8.2).

9.4 La notación ASN.1 para asignar un valor a un nombre de referencia de valor será "ValueAssignment" (véase 15.2) o "ParameterizedValueAssignment" (véase la Rec. UIT-T X.683 | ISO/CEI 8824-4, 8.2).

9.5 Las alternativas de producción de la notación "Assignment" sólo se utilizarán dentro de la notación "ModuleDefinition" (excepto lo especificado en la nota 2 de 12.1).

10 Juego de caracteres ASN.1

10.1 Un elemento léxico consistirá en una secuencia de caracteres del cuadro 2, sin perjuicio de lo especificado en 10.2 y 10.3. En el cuadro 2, los caracteres se identifican con los nombres que reciben en ISO/CEI 10646-1.

Cuadro 2 – Caracteres ASN.1

A a Z	(A MAYÚSCULA a Z MAYÚSCULA)
a a z	(A MINÚSCULA a Z MINÚSCULA)
0 a 9	(DÍGITO 0 a DÍGITO 9)
!	(SIGNO DE EXCLAMACIÓN)
"	(COMILLAS)
&	(SIGNO DE Y)
'	(APÓSTROFO)
(	(PARÉNTESIS DE APERTURA)
)	(PARÉNTESIS DE CIERRE)
*****	(ASTERISCO)
,	(COMA)
-	(GUIÓN-MENOS)
.	(PUNTO)
/	(BARRA DE FRACCIÓN)
:	(DOS PUNTOS)
;	(PUNTO Y COMA)
<	(SIGNO DE MENOR)
=	(SIGNO IGUAL)
>	(SIGNO DE MAYOR)
@	(ARROBA)
[	(CORCHETE DE APERTURA)
]	(CORCHETE DE CIERRE)
^	(ACENTO CIRCUNFLEJO)
_	(CARÁCTER DE SUBRAYADO)
{	(LLAVE DE APERTURA)
|	(LÍNEA VERTICAL)
}	(LLAVE DE CIERRE)

NOTA – Cuando los organismos nacionales de normalización hayan elaborado normas de derivación equivalentes, podrán aparecer caracteres adicionales en los siguientes elementos léxicos:

- typereference (véase 11.2);
- identifier (véase 11.3);
- valuereference (véase 11.4);
- modulereference (véase 11.5).

Cuando se introduzcan caracteres adicionales para acomodar un lenguaje en el que la distinción entre letras mayúsculas y minúsculas no es significativa, la distinción sintáctica conseguida imponiendo que el primer carácter de alguno de los elementos léxicos mencionados anteriormente sea una letra mayúscula o minúscula tendrá que conseguirse de alguna otra forma. Con esto se pretende facilitar la escritura de especificaciones ASN.1 válidas en diversos idiomas.

10.2 Cuando se utilice la notación para especificar el valor de un tipo cadena de caracteres, todos los caracteres del juego definido podrán aparecer en la notación ASN.1, entre caracteres QUOTATION MARK (34) (") (COMILLAS) (véase 11.14).

10.3 En el elemento léxico "comment" pueden aparecer formas de caracteres adicionales alternativas (véase 11.6).

10.4 No se dará significado al estilo tipográfico, tamaño, color, intensidad y otras características de visualización.

10.5 Las letras mayúsculas y minúsculas deberán considerarse distintas.

10.6 Las definiciones ASN.1 también pueden incluir espacios en blanco (véase 11.1.6) entre los elementos léxicos.

11 Elementos léxicos ASN.1

11.1 Reglas generales

11.1.1 Las siguientes subcláusulas especifican los caracteres de elementos léxicos ASN.1. En cada caso se da el nombre del elemento léxico, junto con la definición de las secuencias de caracteres que forman el elemento léxico.

11.1.2 Los elementos léxicos especificados en las subcláusulas de esta cláusula 11 (excepto "bstring", "hstring", "cstring" y "comment" de varias líneas) no contendrán espacios en blanco (véanse 11.6, 11.10, 11.12 y 11.14).

11.1.3 La longitud de la línea no está limitada.

11.1.4 Los elementos léxicos podrán separarse mediante uno o varios espacios en blanco (véase 11.1.6) o comentarios (véase 11.6) excepto cuando se utilice el indicador "**&**" de sin espaciado (véase 5.4). Dentro de una producción "XMLTypedValue" (véase 15.2), podrán aparecer espacios en blanco entre los elementos léxicos, pero el elemento léxico "comment" no debe estar presente.

NOTA – Esto permitirá evitar la ambigüedad creada por la presencia de guiones adyacentes o asteriscos y barras de fracción dentro de un elemento léxico "xmlcstring". Dichos caracteres nunca indican el principio de un elemento léxico "comment" cuando aparecen dentro de una producción "XMLTypedValue".

11.1.5 Un elemento léxico estará separado del siguiente por uno o varios espacios en blanco o comentarios si el carácter (o caracteres) inicial del elemento léxico siguiente es un carácter (o caracteres) autorizado para su inclusión al final de los caracteres del elemento léxico anterior.

11.1.6 Esta Recomendación | Norma Internacional utiliza los términos "nueva línea" y "espacio en blanco". Al representar caracteres espacio en blanco y nueva línea (final de línea) en especificaciones legibles por las máquinas, se podrá utilizar cualquiera o varios de los siguientes caracteres en cualquier combinación (por cada carácter, se asignan el nombre del carácter y el código del carácter especificados en la Norma Unicode):

Para espacio en blanco:

TABULACIÓN HORIZONTAL (9)

CAMBIO DE RENGLÓN (10)

TABULACIÓN VERTICAL (11)

PÁGINA SIGUIENTE (12)

RETROCESO DEL CARRO (13)

ESPACIO (32)

Para nueva línea:

CAMBIO DE RENGLÓN (10)

TABULACIÓN VERTICAL (11)

PÁGINA SIGUIENTE (12)

RETROCESO DEL CARRO (13)

NOTA – Cualquier carácter o secuencia de caracteres que represente una nueva línea válida también representará un espacio en blanco válido.

11.2 Referencias de tipo

Nombre de elemento léxico – typereference

11.2.1 Una "typereference" estará constituida por un número arbitrario de letras (una o más), dígitos y guiones. El carácter inicial será una letra mayúscula. Un guión no será el último carácter. Un guión no irá seguido inmediatamente de otro guión.

NOTA – Las reglas referentes a los guiones están pensadas para evitar ambigüedades con comentarios (que posiblemente sigan).

11.2.2 Una "typereference" no será una de las secuencias de caracteres reservadas indicadas en 11.27.

11.3 Identificadores

Nombre de elemento léxico – identifier

Un "identifier" estará constituido por un número arbitrario de (una o más) letras, dígitos y guiones. El carácter inicial será una letra minúscula. Un guión no será el último carácter. Un guión no irá seguido inmediatamente de otro guión.

NOTA – Las reglas referentes a los guiones están pensadas para evitar ambigüedades con comentarios (que posiblemente sigan).

11.4 Referencias de valor

Nombre de elemento léxico – valuereference

Una "valuereference" consistirá en la secuencia de caracteres especificada para un "identifier" en 11.3. Al analizar un ejemplar de utilización de esta notación, una "valuereference" se distingue de un "identifier" por el contexto en el que aparece.

11.5 Referencias de módulo

Nombre de elemento léxico – modulereference

Una "modulereference" consistirá en la secuencia de caracteres especificada para una "typereference" en 11.2. Al analizar un ejemplar de utilización de esta notación, una "modulereference" se distingue de un "typereference" por el contexto en el que aparece.

11.6 Comentarios

Nombre de elemento léxico – comment

11.6.1 Un "comment" no está referenciado en la definición de la notación ASN.1. Puede, sin embargo, aparecer en cualquier momento entre otros elementos léxicos y no tiene significado sintáctico.

NOTA – No obstante, en el contexto de una Recomendación | Norma Internacional que utiliza ASN.1, un comentario ASN.1 puede contener texto normativo relacionado con la semántica de la aplicación o constricciones a la sintaxis.

11.6.2 El elemento léxico "comment" puede adoptar dos formas:

a) Comentarios de una línea que comienzan con "`--`" como se define en 11.6.3;

b) Comentarios de varias líneas que comienzan con "`/*`" como se define en 11.6.4.

11.6.3 Cuando comience un "comentario" con un par de guiones adyacentes, terminará con el siguiente par de guiones adyacentes o en el final de la línea, lo que suceda primero. Un comentario no debe contener un par de guiones adyacentes que no sea el par que lo inicia y el par, si existe, que lo termina. Si un comentario que comienza con "`--`" incluye los caracteres adyacentes "`/*`" o "`*/`", éstos no tienen significado especial y se consideran parte del comentario. El comentario puede incluir símbolos gráficos que no estén en el conjunto de caracteres especificados en 10.1 (véase 10.3).

11.6.4 Cuando comienza un "comentario" con "`/*`", debe terminar con un "`*/`" correspondiente, esté o no en la misma línea. Si se encuentra otro "`/*`" antes de un "`*/`", entonces el comentario termina cuando se encuentra un "`*/`" que corresponda con cada "`/*`". Si un comentario que comienza con "`/*`" incluye dos guiones adyacentes "`--`", estos guiones no tienen un significado especial y se consideran parte del comentario. El comentario puede incluir símbolos gráficos que no estén en el conjunto de caracteres especificado en 10.1 (véase 10.3).

NOTA – Esto facilita que el usuario comente partes de un módulo ASN.1 que ya contiene comentarios (ya sea que comiencen con "`--`" o con "`/*`") insertando simplemente "`/*`" al comienzo de la parte que se va a comentar y "`*/`" al final, siempre que no haya valores de cadena de caracteres en la parte que ha de comentarse y que contiene "`/*`" o "`*/`".

11.7 Elemento léxico vacío

Nombre de elemento léxico – empty

El elemento "empty" no contiene caracteres. Se usa en la notación de la cláusula 5 cuando se especifican conjuntos alternativos de secuencias de producción, para indicar que es posible la ausencia de todas las alternativas.

11.8 Números

Nombre de elemento léxico – number

Un "number" estará constituido por uno o más dígitos. El primer dígito no será cero a menos que el "number" tenga un solo dígito.

NOTA – El elemento léxico "number" se hace corresponder siempre con (a) un valor entero interpretándolo como notación decimal.

11.9 Números reales

Nombre de elemento léxico – realnumber

Un "realnumber" estará constituido por una parte entera de uno o más dígitos y, opcionalmente, un solo punto decimal (.). El punto decimal puede, opcionalmente, ir seguido de una parte decimal de uno o más dígitos. La parte entera, el punto decimal o la parte decimal (el último que aparezca) pueden, opcionalmente, ir seguidos de una e o E y opcionalmente por un exponente con signo -, de uno o más dígitos. El primer dígito del exponente no será cero a menos que el exponente sea de un solo dígito.

11.10 Cadenas binarias

Nombre de elemento léxico – bstring

Una "bstring" estará constituida por un número arbitrario (que puede ser cero) de los caracteres:

0 1

posiblemente entremezclados con espacios en blanco, precedidos de un carácter APOSTROPHE (39) (apóstrofo) (') y seguido por el par de caracteres.

'B

EJEMPLO – **'01101100'B**

Las ocurrencias de espacios en blanco dentro de un elemento léxico cadena binaria no tienen significado alguno.

11.11 Elemento cadena binaria XML

Nombre de elemento léxico – xmlbstring

Una "xmlbstring" estará constituida por un número arbitrario (que puede ser cero) de ceros, unos o espacios en blanco. Los espacios en blanco que aparecen dentro de un elemento cadena binaria no tienen significado alguno.

EXAMPLE – 01101100

Esta secuencia de caracteres es también un ejemplar válido de "xmlhstring" y "xmlcstring". En el análisis de un ejemplar de utilización de esta notación, una "xmlbstring" se distingue de una "xmlhstring" o "xmlcstring" por el contexto en que aparece.

11.12 Cadenas hexadecimales

Nombre de elemento léxico – hstring

11.12.1 Una "hstring" estará constituida por un número arbitrario (que puede ser cero) de los caracteres:

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

posiblemente entremezclados con espacios en blanco, precedidos de un carácter APOSTROPHE (39) (apóstrofo) (') y seguido por el par de caracteres:

'H

EJEMPLO – **'AB0196'H**

Las ocurrencias de espacios en blanco dentro de un elemento léxico de cadena hexadecimal no tienen significado alguno.

11.12.2 Cada carácter se utiliza para denotar el valor de un semiocteto en representación hexadecimal.

11.13 Elemento cadena hexadecimal XML

Nombre de elemento léxico – xmlhstring

11.13.1 Una "xmlhstring" estará constituida por un número arbitrario (que puede ser cero) de los caracteres:

0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f

o espacios en blanco. Cualquier espacio en blanco dentro de un elemento cadena hexadecimal no tienen significado alguno.

EJEMPLO – Ab0196

11.13.2 Cada carácter se utiliza para denotar el valor de un semiocteto utilizando una representación hexadecimal.

11.13.3 Algunos ejemplares de "xmlhstring" también son ejemplos válidos de "xmlbstring" y "xmlcstring". En el análisis de un ejemplo de utilización de esta notación, una "xmlhstring" se distingue de una "xmlbstring" o "xmlcstring" por el contexto en que aparece.

11.14 Cadenas de caracteres

Nombre de elemento léxico – cstring

11.14.1 Una "cstring" estará constituida por un número arbitrario (que puede ser cero) de símbolos gráficos y caracteres con espaciado tomados del conjunto de caracteres referenciado por el tipo cadena de caracteres, precedido y seguido del carácter QUOTATION MARK (34) ("). Si el conjunto de caracteres incluye dicho carácter, éste (si está presente en la cadena de caracteres representada por la "cstring") debe estar representado en la "cstring" por un par de caracteres QUOTATION MARK (34) (") en la misma línea, sin espacio separador. La "cstring" puede abarcar más de una línea de texto, en cuyo caso la cadena de caracteres representada no incluirá caracteres con espaciado en la posición que precede o sigue al final de línea de la "cstring". Cualquier carácter con espaciado inmediatamente antes o después del final de la línea de la "cstring" no tiene significado alguno.

NOTA 1 – La "cstring" sólo puede utilizarse para representar inequívocamente cadenas de caracteres (impresas en papel) a cada uno de cuyos caracteres se le ha asignado un símbolo gráfico o bien se trata de un carácter con espaciado. Se dispone de sintaxis ASN.1 alternativa para cuando sea preciso denotar (en forma impresa) una cadena de caracteres que contiene caracteres de control. (Véase la cláusula 35).

NOTA 2 – La cadena de caracteres representada por una "cstring" está constituida por los caracteres asociados a los símbolos gráficos y los caracteres con espaciado. Los caracteres con espaciado que preceden o siguen inmediatamente cualquier fin de línea de la "cstring" no son parte de la cadena de caracteres que se representa (son ignorados). Cuando se incluyen caracteres con espaciado en la "cstring" o cuando símbolos gráficos del repertorio de caracteres no son inequívocos en una representación impresa, la cadena de caracteres denotada por "cstring" puede resultar imprecisa en esa representación impresa.

EJEMPLO 1 – **"屎 屍 市 弒"**

EJEMPLO 2 – La "cstring":

```
"ABCDE   FGH
IJK""XYZ"
```

puede utilizarse para representar un valor de cadena de caracteres de tipo **IA5String.** El valor representado consta de los caracteres:

```
ABCDE   FGHIJK"XYZ
```

en donde el número exacto de espacios que se pretende que haya entre **E** y **F** puede resultar poco preciso en una representación impresa si en la especificación impresa se utiliza un tipo de carácter de espaciado proporcional (como el empleado más arriba), o si el repertorio de caracteres contiene varios caracteres con espaciado de diferentes anchuras.

11.14.2 Cuando un carácter sea combinante (véase el anexo F), se denotará en una representación impresa de la "cstring" como un carácter individual. No se sobreimprimirá con los caracteres con los que se combina. (Así se asegura que el orden de los caracteres combinantes en el valor cadena queda definido de manera inequívoca en la versión impresa.)

EJEMPLO – El carácter combinante letra minúscula "**e**" y acento son dos caracteres en ISO/CEI 10646-1 y, por eso, una "cstring" correspondiente debe imprimirse como dos caracteres y no como un solo carácter **é**.

11.15 Elemento cadena de caracteres XML

Nombre de elemento léxico – xmlcstring

11.15.1 Un "xmlcstring" consiste en un número arbitrario (posiblemente cero) de los siguientes caracteres ISO/CEI 10646-1:

a) TABULACIÓN HORIZONTAL (9);

b) CAMBIO DE RENGLÓN (10);

c) RETORNO DE CARRO (13);

d) cualquier carácter cuyo código de carácter ISO/CEI 10646-1 se encuentre en el intervalo de 32 (20 hex) a 55295 (D7FF hex), inclusive;

e) cualquier carácter cuyo código de carácter ISO/CEI 10646-1 se encuentre en el intervalo de 57344 (E000 hex) a 65533 (FFFD hex), inclusive;

f) cualquier carácter cuyo código de carácter ISO/CEI 10646-1 se encuentre en el intervalo de 65536 (10000 hex) a 1114111 (10FFFF hex), inclusive.

NOTA – El requisito de que "xmlcstring", en un ejemplar de utilización, debe contener sólo caracteres permitidos por el tipo de cadena de caracteres gobernante, impone restricciones adicionales.

11.15.2 Los caracteres "&" (AMPERSAND), "<" (SIGNO DE MENOR) o ">" (SIGNO DE MAYOR) deben aparecer sólo como parte de una de las secuencias de caracteres especificadas en 11.15.4 y 11.15.5.

11.15.3 Se utiliza "xmlcstring" para representar el valor de una cadena de caracteres restringida (véase 37.9), y puede emplearse para representar todas las combinaciones de caracteres ISO/CEI 10646-1, bien sea directamente, o usando las secuencias de escape que se especifican a continuación.

NOTA 1 – Un "xmlcstring" no puede emplearse para representar caracteres no presentes en ISO/CEI 10646-1, como es el caso de algunos de los caracteres de control que pueden aparecer en **`GeneralString`**, ni caracteres que pudieran definirse con los códigos de caracteres ISO/CEI 10646-1 por encima de 10FFFF hex.

NOTA 2 – Los caracteres CAMBIO DE RENGLÓN (10) y RETORNO DE CARRO (13) y el par RETORNO DE CARRO + CAMBIO DE RENGLÓN no pueden distinguirse cuando son tratados por procesadores XML conformes.

11.15.4 Si los caracteres "&" (AMPERSAND), "<" (SIGNO DE MENOR) o ">" (SIGNO DE MAYOR) están presentes en un valor de cadena de caracteres abstracto que se representa mediante "xmlcstring" (véase 37.9), se representarán en "xmlcstring" mediante

a) las secuencia de escape especificadas en 11.15.8; o

b) las secuencias de escape "&", "<" o ">" respectivamente. Estas secuencias de escape no deben contener espacios en blanco (véase 11.1.6).

11.15.5 Si un carácter con un código de carácter ISO/CEI 10646-1 en la columna 1 del cuadro 3 está presente en el valor de cadena de caracteres abstracto representado por "xmlcstring" (véase 37.9), se representará mediante la secuencia de caracteres de la segunda columna del cuadro 3. Estas secuencias de caracteres no deben contener espacios en blanco (véase 11.1.6).

NOTA – Eso no incluye los caracteres con códigos de carácter decimal 9, 10 y 13, y todas las letras en esas secuencias de caracteres son minúsculas.

Cuadro 3 – Secuencias de escape para los caracteres de control en una "xmlcstring"

Código de carácter ISO/CEI 10646-1	Representación de "xmlcstring"	Código de carácter ISO/CEI 10646-1	Representación de "xmlcstring"
0 (0 hex)	<nul/>	17 (11 hex)	<dc1/>
1 (1 hex)	<soh/>	18 (12 hex)	<dc2/>
2 (2 hex)	<stx/>	19 (13 hex)	<dc3/>
3 (3 hex)	<etx/>	20 (14 hex)	<dc4/>
4 (4 hex)	<eot/>	21 (15 hex)	<nak/>
5 (5 hex)	<enq/>	22 (16 hex)	<syn/>
6 (6 hex)	<ack/>	23 (17 hex)	<etb/>
7 (7 hex)	<bel/>	24 (18 hex)	<can/>
8 (8 hex)	<bs/>	25 (19 hex)	
11 (B hex)	<vt/>	26 (1A hex)	<sub/>
12 (C hex)	<ff/>	27 (1B hex)	<esc/>
14 (E hex)	<so/>	28 (1C hex)	<is4/>
15 (F hex)	<si/>	29 (1D hex)	<is3/>
16 (10 hex)	<dle/>	30 (1E hex)	<is2/>
		31 (1F hex)	<is1/>

11.15.6 Cuando se utiliza "xmlcstring" dentro de un "XMLTypedValue" (véase 15.2) que forma parte de una codificación XER (véase la Rec. UIT-T X.693 | ISO/CEI 8825-4), el mismo podrá contener caracteres GUIÓN-MENOS (45). Cuando se usa dentro de un ejemplar de notación de valor XML en un módulo ASN.1, no debe contener dos caracteres GUIÓN-MENOS adyacentes. Si esta secuencia de caracteres está presente en un valor de cadena de caracteres abstracto que se representa mediante "xmlcstring" en un módulo ASN.1, en ese caso, al menos uno de los caracteres GUIÓN-MENOS adyacentes debe representarse mediante las secuencias de escape especificadas en 11.15.8.

11.15.7 Cuando se utiliza "xmlcstring" dentro de un "XMLTypedValue" que forma parte de una codificación XER (véase la Rec. UIT-T X.693 | ISO/CEI 8825-4), el mismo podrá contener caracteres ASTERISCO (42) y BARRA DE

FRACCIÓN (47) en cualquier orden. Cuando se usa dentro de un ejemplar de notación de valor XML en un módulo ASN.1, no debe contener caracteres ASTERISCO ni BARRA DE FRACCIÓN (en cualquier orden). Si esta secuencia de caracteres está presente en un valor de cadena de caracteres abstracto que se representa mediante "xmlcstring", en ese caso, al menos uno de los caracteres ASTERISCO ni BARRA DE FRACCIÓN debe representarse mediante las secuencias de escape especificadas en 11.15.8.

11.15.8 Cualquier carácter que pueda aparecer directamente en una "xmlcstring" también podrá representarse en la "xmlcstring" mediante una secuencia de escape de la forma "&#n;" (donde n es el código de carácter ISO/CEI 10646-1 en notación decimal) o de la forma "&#xn;" (donde n es el código de carácter ISO/CEI 10646-1 en notación hexadecimal). Estas secuencias de escape no deben contener espacios en blanco (véase 11.1.6).

NOTA 1 – En los valores decimales y hexadecimales de "n" se permiten ceros iniciales y en el valor hexadecimal pueden utilizarse minúsculas y mayúsculas "A"-"F".

NOTA 2 – Si las secuencias de escape "&#n" y "&#xn" se emplean para los caracteres ISO/CEI 10646-1 que no pertenecen al plano plurilingüe básico (BMP), el valor de "n" será mayor que 65535 (FFFF hex).

EJEMPLO – La "xmlcstring":

ABCDé FGHîJK&XYZ

se puede utilizar para representar un valor de cadena de caracteres del tipo UTF8String. El valor representado está constituido por los caracteres:

ABCDé FGHîJK&XYZ

donde los caracteres de espacio precisos entre é y F pueden ser ambiguos en los medios de impresión si en la especificación se utiliza un tipo de caracteres con espaciado proporcional (tal como el anterior).

11.16 Elemento léxico asignación

Nombre de elemento léxico – "**::=**"

Este elemento léxico estará constituido por la secuencia de caracteres:

::=

NOTA – Esta secuencia no contiene espacios en blanco (véase 11.1.2).

11.17 Separador de rango

Nombre de elemento léxico – "**..**"

Este elemento léxico estará constituido por la secuencia de caracteres:

..

NOTA – Esta secuencia no contiene espacios en blanco (véase 11.1.2).

11.18 Puntos suspensivos

Nombre de elemento léxico – "**...**"

Este elemento léxico estará constituido por la secuencia de caracteres:

...

NOTA – Esta secuencia no contiene espacios en blanco (véase 11.1.2).

11.19 Corchetes de versión de apertura

Nombre de elemento léxico – "**[[**"

Este elemento léxico estará constituido por la secuencia de caracteres:

[[

NOTA – Esta secuencia no contiene espacios en blanco (véase 11.1.2).

11.20 Corchetes de versión de cierre

Nombre de elemento léxico – "**]]**"

Este elemento léxico estará constituido por la secuencia de caracteres:

```
]]
```

NOTA – Esta secuencia no contiene espacios en blanco (véase 11.1.2).

11.21 Elemento de comienzo de rótulo final XML

Nombre de elemento léxico – "</"

Este elemento léxico estará constituido por la secuencia de caracteres:

```
</
```

NOTA – Esta secuencia no contiene espacios en blanco (véase 11.1.2).

11.22 Elemento de fin de rótulo simple XML

Nombre de elemento léxico – "/>"

Este elemento léxico estará constituido por la secuencia de caracteres:

```
/>
```

NOTA – Esta secuencia no contiene espacios en blanco (véase 11.1.2).

11.23 Elemento booleano verdadero XML

Nombre de elemento léxico – "true"

11.23.1 Este elemento léxico estará constituido por la secuencia de caracteres:

true

11.23.2 Al analizar un ejemplar de utilización de esta notación, se distingue un elemento "true" de uno "valuereference" o de uno "identifier" por el contexto en el que aparece.

NOTA – Esta secuencia no contiene espacios en blanco (véase 11.1.2).

11.24 Elemento booleano falso XML

Nombre de elemento léxico – "false"

11.24.1 Este elemento léxico estará constituido por la secuencia de caracteres:

false

11.24.2 Al analizar un ejemplar de utilización de esta notación, se distingue un elemento "false" de uno "valuereference" o de uno "identifier" por el contexto en el que aparece.

NOTA – Esta secuencia no contiene espacios en blanco (véase 11.1.2).

11.25 Nombres de rótulos XML para tipos ASN.1

Nombre de elemento léxico – xmlasn1typename

11.25.1 Esta Recomendación | Norma Internacional utiliza el elemento léxico "xmlasn1typename" cuando han de emplearse tipos incorporados ASN.1 como nombres de rótulos XML.

11.25.2 En el cuadro 4 se relacionan las secuencias de caracteres que permitirán formar el "xmlasn1typename" de cada uno de los tipos incorporados ASN.1 enumerados en 16.2. El tipo incorporado ASN.1 se identifica en la columna 1 del cuadro 4 mediante su nombre de producción. La secuencia de caracteres que será utilizada para "xmlasn1typename" se identifica en la columna 2 del cuadro 4, sin espacio en blanco antes o después de estas secuencias de caracteres.

11.25.3 El "xmlasn1typename" para los "UsefulType" (véase 41.1) será el "typereference" utilizado en su definición.

11.25.4 La secuencia de caracteres en el elemento "xmlasn1typename" para el "ObjectClassFieldType" y para la "InstanceOfType" se especifican en la Rec. UIT-T X.681 | ISO/CEI 8824-2, 14.1 y el anexo C.

11.25.5 Si el tipo incorporado ASN.1 es un "TaggedType" el tipo que determina el "xmlasn1typename" será "Type" en el "TaggedType" (véase 30.1). Si éste es a su vez un "TaggedType", en ese caso esta subcláusula 11.25.5 se aplicará recurrentemente.

Cuadro 4 – Caracteres en xmlasn1typename

Nombre de producción tipo ASN.1	Caracteres en xmlasn1typename
BitStringType	BIT_STRING
BooleanType	BOOLEAN
ChoiceType	CHOICE
EmbeddedPDVType	SEQUENCE
EnumeratedType	ENUMERATED
ExternalType	SEQUENCE
InstanceOfType	SEQUENCE
IntegerType	INTEGER
NullType	NULL
ObjectClassFieldType	*Véase la Rec. UIT-T X.681 | ISO/CEI 8824-2, 14.10 y 14.11*
ObjectIdentifierType	OBJECT_IDENTIFIER
OctetStringType	OCTET_STRING
RealType	REAL
RelativeOIDType	RELATIVE_OID
RestrictedCharacterStringType	*El nombre deltipo (por ejemplo.* `IA5String`*)*
SequenceType	SEQUENCE
SequenceOfType	SEQUENCE_OF
SetType	SET
SetOfType	SET_OF
TaggedType	*Véase 11.25.5*
UnrestrictedCharacterStringType	SEQUENCE

11.26 Elementos léxicos de carácter simple

Nombres de elementos léxicos –

"{"
"}"
"<"
">"
","
"."
"("
")"
"["
"]"
"-" (GUIÓN-MENOS)
":"
"="
""" (COMILLAS)
"'" (APÓSTROFO)
" " (ESPACIO)
"."
"@"
"|"
"!"
"^"

Un elemento léxico con cualquiera de los nombres antes indicados estará constituido por el carácter simple sin comillas.

11.27 Palabras reservadas

Nombres de palabras reservadas -

ABSENT	**ENCODED**	**INTEGER**	**RELATIVE-OID**
ABSTRACT-SYNTAX	**END**	**INTERSECTION**	**SEQUENCE**
ALL	**ENUMERATED**	**ISO646String**	**SET**
APPLICATION	**EXCEPT**	**MAX**	**SIZE**
AUTOMATIC	**EXPLICIT**	**MIN**	**STRING**
BEGIN	**EXPORTS**	**MINUS-INFINITY**	**SYNTAX**
BIT	**EXTENSIBILITY**	**NULL**	**T61String**
BMPString	**EXTERNAL**	**NumericString**	**TAGS**
BOOLEAN	**FALSE**	**OBJECT**	**TeletexString**
BY	**FROM**	**ObjectDescriptor**	**TRUE**
CHARACTER	**GeneralizedTime**	**OCTET**	**TYPE-IDENTIFIER**
CHOICE	**GeneralString**	**OF**	**UNION**
CLASS	**GraphicString**	**OPTIONAL**	**UNIQUE**
COMPONENT	**IA5String**	**PATTERN**	**UNIVERSAL**
COMPONENTS	**IDENTIFIER**	**PDV**	**UniversalString**
CONSTRAINED	**IMPLICIT**	**PLUS-INFINITY**	**UTCTime**
CONTAINING	**IMPLIED**	**PRESENT**	**UTF8String**
DEFAULT	**IMPORTS**	**PrintableString**	**VideotexString**
DEFINITIONS	**INCLUDES**	**PRIVATE**	**VisibleString**
EMBEDDED	**INSTANCE**	**REAL**	**WITH**

Los elementos léxicos con cualquiera de estos nombres estarán constituidos por la secuencia de caracteres que forman el nombre y son secuencias de caracteres reservadas.

NOTA 1 – Estas secuencias no contienen espacios en blanco.

NOTA 2 – Las palabras clave **CLASS**, **CONSTRAINED**, **CONTAINING**, **ENCODED**, **INSTANCE**, **SYNTAX** y **UNIQUE** no se utilizan en esta Recomendación | Norma Internacional; se utilizan en la Rec. UIT-T X.681 | ISO/CEI 8824-2, en la Rec. UIT-T X.682 | ISO/CEI 8824-3 y en la Rec. UIT-T X.683 | ISO/CEI 8824-4.

12 Definición de módulo

12.1 Una "ModuleDefinition" se especifica mediante las siguientes producciones:

ModuleDefinition ::=
ModuleIdentifier
DEFINITIONS
TagDefault
ExtensionDefault
"::="
BEGIN
ModuleBody
END

ModuleIdentifier ::=
modulereference
DefinitiveIdentifier

DefinitiveIdentifier ::=
"{" DefinitiveObjIdComponentList "}"
| **empty**

DefinitiveObjIdComponentList ::=
DefinitiveObjIdComponent
| **DefinitiveObjIdComponent DefinitiveObjIdComponentList**

```
DefinitiveObjIdComponent ::=
        NameForm
    |   DefinitiveNumberForm
    |   DefinitiveNameAndNumberForm

DefinitiveNumberForm    ::= number

DefinitiveNameAndNumberForm ::= identifier "(" DefinitiveNumberForm ")"

TagDefault ::=
        EXPLICIT TAGS
    |   IMPLICIT TAGS
    |   AUTOMATIC TAGS
    |   empty

ExtensionDefault ::=
        EXTENSIBILITY IMPLIED
    |   empty

ModuleBody ::=
        Exports Imports AssignmentList
    |   empty

Exports ::=
        EXPORTS SymbolsExported ";"
    |   EXPORTS ALL ";"
    |   empty

SymbolsExported ::=
        SymbolList
    |   empty

Imports ::=
        IMPORTS SymbolsImported ";"
    |   empty

SymbolsImported ::=
        SymbolsFromModuleList
    |   empty

SymbolsFromModuleList ::=
        SymbolsFromModule
    |   SymbolsFromModuleList  SymbolsFromModule

SymbolsFromModule ::=
        SymbolList FROM GlobalModuleReference

GlobalModuleReference ::=
        modulereference AssignedIdentifier

AssignedIdentifier ::=
        ObjectIdentifierValue
    |   DefinedValue
    |   empty

SymbolList ::=
        Symbol
    |   SymbolList "," Symbol

Symbol ::=
        Reference
    |   ParameterizedReference

Reference ::=
        typereference
    |   valuereference
    |   objectclassreference
    |   objectreference
    |   objectsetreference
```

AssignmentList ::=
    Assignment
  | **AssignmentList Assignment**

Assignment ::=
    TypeAssignment
  | **ValueAssignment**
  | **XMLValueAssignment**
  | **ValueSetTypeAssignment**
  | **ObjectClassAssignment**
  | **ObjectAssignment**
  | **ObjectSetAssignment**
  | **ParameterizedAssignment**

NOTA 1 – La utilización de una "ParameterizedReference" en las listas "Exports" e "Imports" se especifica en la Rec. UIT-T X.683 | ISO/CEI 8824-4.

NOTA 2 – En los ejemplos (y en la definición de esta Recomendación | Norma Internacional de tipos con rótulos de la clase universal), el "ModuleBody" puede utilizarse fuera de una "ModuleDefinition".

NOTA 3 – Las producciones "TypeAssignment", "ValueAssignment" y "ValueSetTypeAssignment" se especifican en la cláusula 15.

NOTA 4 – El valor de "TagDefault" para la definición de módulo sólo afecta a los tipos definidos explícitamente en el módulo. No afecta a la interpretación de tipos importados.

NOTA 5 – El carácter punto y coma (;) no aparece en la especificación de la lista de asignaciones ni en ninguna de sus producciones subordinadas, y está reservado para uso de los diseñadores de herramientas ASN.1.

12.2 El "TagDefault" se toma como **`EXPLICIT TAGS`** si es "vacío".

NOTA – La cláusula 30 da el significado de **`EXPLICIT TAGS`**, **`IMPLICIT TAGS`** y **`AUTOMATIC TAGS`**.

12.3 Cuando se selecciona la alternativa **`AUTOMATIC TAGS`** de "TagDefault", se dice que se ha seleccionado la rotulación automática del módulo; de no ser así, se dice que no se ha seleccionado. La rotulación automática es una transformación sintáctica que se aplica (con condiciones adicionales) a las producciones "ComponentTypeLists" y "AlternativeTypeLists" que aparecen en la definición del módulo. Esta transformación está especificada formalmente en 24.7 a 24.9, 26.3 y 28.2 a 28.5 a propósito de las notaciones de los tipos secuencia, los tipos conjunto y los tipos elección, respectivamente.

12.4 La opción **`EXTENSIBILITY IMPLIED`** es equivalente a la inserción textual de un marcador de extensión ("**`...`**") en la definición de cada tipo en el módulo para el cual eso está permitido. La ubicación del marcador de extensión implicado es la última posición en el tipo en el cual se permite un marcador de extensión especificado explícitamente. La ausencia de **`EXTENSIBILITY IMPLIED`** significa que sólo se proporciona extensibilidad para aquellos tipos dentro del módulo en los que está explícitamente presente un marcador de extensión.

NOTA – **`EXTENSIBILITY IMPLIED`** afecta sólo a los tipos. No produce efecto sobre conjuntos de objetos ni constricciones de subtipos.

12.5 A la "modulereference" que aparece en la producción "ModuleIdentifier" se le llama nombre del módulo.

NOTA – La posibilidad de definir un módulo ASN.1 único haciendo aparecer varios "ModuleBody" con la misma "modulereference" asignada fue (discutiblemente) permitida en especificaciones anteriores. No se permite en la presente Recomendación | Norma Internacional.

12.6 Los nombres de módulo deberán utilizarse una vez solamente (salvo lo especificado en 12.9) dentro de la esfera de interés de la definición del módulo.

12.7 Si el "identificador definitivo" ("DefinitiveIdentifier") no está vacío, el valor del identificador de objeto denotado identifica inequívocamente y de forma única el módulo que se está definiendo. En la definición del valor del identificador de objeto no podrá utilizarse un valor definido.

NOTA – En esta Recomendación | Norma Internacional no se trata la cuestión de los cambios en un módulo exigidos por un nuevo "DefinitiveIdentifier".

12.8 Si el "identificador asignado" ("AssignedIdentifier") no está vacío, las alternativas "valor de identificador de objeto" ("ObjectIdentifierValue") y "valor definido" ("DefinedValue") identifican inequívocamente y de forma única el módulo del cual se están importando nombres de referencia. Cuando se utilice la alternativa "DefinedValue" de "AssignedIdentifier", deberá ser un valor de identificador de objeto de tipo. Cada "valuereference" que aparezca textualmente dentro de un "AssignedIdentifier" deberá satisfacer una de las reglas siguientes:

a) Está definida en la "lista de asignaciones" ("AssignmentList") del módulo que se está definiendo, y todas las "referencias de valor" ("valuereference"s) que aparecen textualmente en el lado derecho del enunciado de asignación también satisfacen esta regla (regla "a") o la regla siguiente (regla "b").

b) Aparece como "símbolo" ("Symbol") en un "símbolos procedentes de módulo" ("SymbolsFromModule") cuyo "AssignedIdentifier" no contiene textualmente ninguna "valuereference".

NOTA – Se recomienda asignar un identificador de objeto de tal modo que otros puedan hacer referencia inequívocamente al módulo.

12.9 La "GlobalModuleReference" de un "SymbolsFromModule" deberá aparecer en la "ModuleDefinition" de otro módulo, con la excepción de que, si incluye un "DefinitiveIdentifier" no vacío, la "referencia de módulo" ("modulereference") puede no ser la misma en los dos casos.

NOTA – Sólo deberá utilizarse una "modulereference" diferente a la utilizada en el otro módulo cuando deban importarse símbolos de dos módulos que tienen el mismo nombre (por haber sido denominados los módulos sin tener en cuenta 12.6). Cuando se utilizan otros nombres distintos, dichos nombres pueden emplearse en el cuerpo del módulo (véase 12.15).

12.10 Cuando en la referencia de un módulo se utiliza una "modulereference" y un "AssignedIdentifier" no vacío, éste deberá considerarse definitivo.

12.11 Cuando el módulo referenciado tenga un "DefinitiveIdentifier" no vacío, la "GlobalModuleReference" que referencia ese módulo no podrá tener un "AssignedIdentifier" vacío.

12.12 Cuando se seleccione la alternativa "SymbolsExported" de "Exports":

a) cada "Symbol" de "SymbolsExported" deberá satisfacer una, y solamente una, de las condiciones siguientes:

i) está definido únicamente en el módulo que se está construyendo; o

ii) aparece exactamente una vez en la alternativa "SymbolsImported" de "Imports";

b) todo "Symbol" al que se haga referencia desde el exterior del módulo de manera apropiada deberá incluirse en el "SymbolsExported" y sólo estos "símbolos" podrán ser referenciados desde el exterior del módulo (supeditado a la relajación especificada en 12.13); y

c) si no existen tales símbolos, deberá seleccionarse entonces la alternativa "vacío" de "SymbolsExported" (no de "Exports").

12.13 Cuando se selecciona la alternativa "empty" de la alternativa **`EXPORTS ALL`** de "Exports", todo "Symbol" definido en el módulo o importado por el módulo puede ser referenciado desde otros módulos a reserva de la restricción especificada en 12.12 a).

NOTA – La alternativa "empty" de "Exports" se incluye para asegurar la compatibilidad ascendente.

12.14 Los identificadores que aparecen en una "NamedNumberList", "Enumeration" o "NamedBitList" son exportados implícitamente si la referencia de tipo que los define es exportada o aparece como un componente (o un subcomponente) dentro de un tipo exportado.

12.15 Cuando se seleccione la alternativa "SymbolsImported" de "Imports":

a) Cada "Symbol" de "SymbolsFromModule" deberá estar definido en el cuerpo del módulo, o presente en la cláusula "Imports", del módulo denotado por la "GlobalModuleReference" en "SymbolsFromModule". La importación de un "Symbol" presente en la cláusula "Imports" del módulo referenciado se permite únicamente si el "Symbol" aparece solamente una vez en esa cláusula y no está definido en el módulo referenciado.

NOTA 1 – Esto no impide que el mismo nombre de símbolo definido en dos módulos diferentes se importe en otro módulo. Sin embargo, si el mismo nombre de "Symbol" aparece más de una vez en la cláusula "Imports" del módulo A, ese nombre de "Symbol" no puede ser exportado de A para importarlo en otro módulo B.

b) Si la alternativa "SymbolsExported" de "Exports" está seleccionada en la definición del módulo denotado por la "GlobalModuleReference" en "SymbolsFromModule", el "Symbol" deberá aparecer en su "SymbolsExported".

c) Sólo los "símbolos" que aparezcan en la "SymbolList" de un "SymbolsFromModule" podrán aparecer como el símbolo en cualquier "External<X>Reference" que tenga la "modulereference" denotada por "GlobalModuleReference" de ese "SymbolsFromModule" (donde <X> es "Value", "Type", "Object", "Objectclass" u "Objectset").

d) Si no existen tales "símbolos", deberá seleccionarse entonces la alternativa "empty" de "SymbolsImported".

NOTA 2 – Una consecuencia de c) y d) es que "**`IMPORTS`**" implica que el módulo no puede contener una "External<X>Reference".

e) Todos los "SymbolsFromModule" de la "SymbolsFromModuleList" deberán incluir ocurrencias de "GlobalModuleReference" de tal modo que:

i) cada "modulereference" en ellos sea diferente de cada una de las demás, y también de la "modulereference" asociada con el módulo referenciante; y

ii) el "AssignedIdentifier", cuando no esté vacío, denota valores de identificador de objeto cada uno de los cuales será diferente de cada uno de los demás, y también del valor de identificador de objeto (si existe) asociado con el módulo referenciante.

12.16 Cuando se selecciona la alternativa "empty" de "Imports", el módulo puede, aun así, referenciar "Symbols" definidos en otros módulos mediante una "External<X>Reference".

NOTA – La alternativa "empty" de "Imports" se incluye para asegurar la compatibilidad ascendente.

12.17 Los identificadores que aparecen en una "NamedNumberList", "Enumeration" o "NamedBitList" son importados implícitamente si la referencia de tipo que los define ha sido importada o aparece como un componente (o subcomponente) dentro de un tipo importado.

12.18 Un "Symbol" de un "SymbolsFromModule" puede aparecer en "ModuleBody" como una "Reference". El significado asociado con el "Symbol" es el que tiene en el módulo denotado por la correspondiente "GlobalModuleReference".

12.19 Cuando el "Symbol" aparezca también en una "AssigmentList" (lo cual se desaconseja), o en otro u otros varios ejemplares de "SymbolsFromModule", sólo deberá utilizarse en una "External<X>Reference". Cuando no aparezca de esta forma, se utilizará directamente como una "Reference".

12.20 Las diversas alternativas de "Assignment" se definen en la siguientes subcláusulas de esta Recomendación | Norma Internacional, excepto cuando se indica otra cosa:

Alternativa de Assignment	*Subcláusula definidora*
"TypeAssignment"	15.1
"ValueAssignment"	15.2
"XMLValueAssignment"	15.2
"ValueSetTypeAssignment"	15.6
"ObjectClassAssignment"	Recomendación UIT-T X.681 | ISO/CEI 8824-2, 9.1
"ObjectAssignment"	Recomendación UIT-T X.681 | ISO/CEI 8824-2, 11.1
"ObjectSetAssignment"	Recomendación UIT-T X.681 | ISO/CEI 8824-2, 12.1
"ParameterizedAssignment"	Recomendación UIT-T X.683 | ISO/CEI 8824-4, 8.1

El primer símbolo de cada "Assignment" es una de las alternativas de "Reference", que denota el nombre de referencia que se está definiendo. En una "AssignmentList" no podrá haber dos asignaciones que tengan los mismos nombres de referencia.

13 Referenciación de definiciones de tipo y valor

13.1 Las producciones tipo definido y valor definido:

```
DefinedType ::=
        ExternalTypeReference
    |   Typereference
    |   ParameterizedType
    |   ParameterizedValueSetType

DefinedValue ::=
        ExternalValueReference
    |   Valuereference
    |   ParameterizedValue
```

especifican las secuencias que deberán utilizarse para referenciar definiciones de tipo y valor. El tipo identificado por un "tipo parametrizado" ("ParameterizedType") y por un "tipo de conjunto de valores parametrizados" ("ParameterizedValueSetType") y el valor identificado por un "valor parametrizado" ("ParameterizedValue") se especifican en la Rec. UIT-T X.683 | ISO/CEI 8824-4.

13.2 La producción "NonParameterizedTypeName":

NonParameterizedTypeName ::=
ExternalTypeReference
| typereference
| xmlasn1typename

se utiliza cuando se necesita un nombre de rótulo XML para representar un tipo ASN.1.

13.3 La tercera alternativa no se utilizará como el "NonParameterizedTypeName" en el "XMLTypedValue" de "XMLValueAssignment" (véase 15.2) o de "XMLOpenTypeFieldVal" (véase la Rec. UIT-T X.681 | ISO/CEI 8824-2, 14.6) cuando se emplea la notación de valor XML en un módulo ASN.1 si el "xmlasn1typename" es "CHOICE", "ENUMERATED", "SEQUENCE", "SEQUENCE_OF", "SET" o "SET_OF".

NOTA – Esta restricción se impone a la notación de valor XML utilizada en un módulo ASN.1 debido a que estos "xmlasn1typename" no definen un tipo ASN.1. La restricción no se impone cuando esta notación se emplea en las reglas de codificación (tales como XER, véase la Rec. UIT-T X.693 | ISO/CEI 8825-4) debido a que los rótulos XML formados a partir de los "xmlasn1typename" no se usan para determinar los tipos que han de codificarse.

13.4 Salvo lo especificado en 12.18, las alternativas "typereference", "valuereference", "ParameterizedType", "ParameterizedValueSetType" o "ParameterizedValue" no deberán utilizarse a menos que la referencia esté dentro del "ModuleBody" en que se ha asignado un tipo o valor (véanse 15.1 y 15.2) a la "typereference" o la "valuereference".

13.5 La "ExternalTypeReference" y la "ExternalValueReference" no deberán utilizarse a menos que a la correspondiente "typereference" o "valuereference":

a) se le haya asignado un tipo o un valor, respectivamente (véanse 15.1 y 15.2); o

b) están presentes en la cláusula "Imports",

dentro del "ModuleBody" utilizado para definir el "modulereference" correspondiente, sólo se permitirá la referenciación de un nombre en la cláusula "Imports" de otro módulo si el "Symbol" no aparece más que una vez en dicha cláusula.

NOTA – Esto no impide que el mismo "Symbol" definido en dos módulos diferentes sea importado en otro módulo. Sin embargo, si el mismo "Symbol" aparece más de una vez en la cláusula **IMPORTS** de un módulo **A**, ese "Symbol" no puede ser referenciado utilizando el módulo **A** en una referencia externa.

13.6 Una referencia externa solamente deberá utilizarse en un módulo para referenciar un nombre de referencia que está definido en un módulo diferente, y se especifica mediante las producciones siguientes:

ExternalTypeReference ::=
modulereference
"."
typereference

ExternalValueReference ::=
modulereference
"."
valuereference

NOTA – En la Rec. UIT-T X.681 | ISO/CEI 8824-2 se especifican producciones de referencia externa adicionales ("ExternalClassReference", "ExternalObjectReference" y "ExternalObjectSetReference").

13.7 Cuando el módulo referenciante se haya definido utilizando la alternativa "SymbolsImported" de "Imports", la "modulereference" de la referencia externa deberá aparecer en la "GlobalModuleReference" de exactamente uno de los "SymbolsFromModule" de los "SymbolsImported". Cuando el módulo referenciante se haya definido utilizando la alternativa "empty" de "Imports", la "modulereference" de la referencia externa deberá aparecer en la "ModuleDefinition" del módulo (distinto del módulo referenciante) en que está definida la "Reference".

13.8 Cuando se utiliza un "DefinedType" como parte de una notación gobernada por un "Type" (por ejemplo, en una "SubtypeConstraint"), el "DefinedType" deberá ser compatible con el "Type" gobernante, tal como se especifica en la cláusula B.6.2.

13.9 Siempre que aparece un "DefinedValue" en una especificación ASN.1, está gobernado por un "Type", y ese "DefinedValue" deberá hacer referencia a un valor de un tipo que sea compatible con el "Type" gobernante, tal como se especifica en cláusula B.6.2.

14 Notación para soportar referencias a componentes ASN.1

14.1 También se requiere hacer referencia formal a componentes de tipos, valores, etc. ASN.1 para muchos otros fines. Uno de estos ejemplares se da cuando es necesario, al escribir un texto, identificar un tipo específico dentro de algún módulo ASN.1. Esta cláusula define una notación que puede utilizarse para proporcionar tales referencias.

14.2 La notación permite identificar cualquier componente de un tipo conjunto o secuencia (cuya presencia en el tipo puede ser obligatoria u opcional).

14.3 Cualquier parte de cualquier definición de tipo ASN.1 puede ser referenciada mediante la utilización de la construcción sintáctica "AbsoluteReference":

```
AbsoluteReference ::= "@" ModuleIdentifier
                      "."
                      ItemSpec

ItemSpec ::=
                      typereference
                  |   ItemId "." ComponentId

ItemId ::= ItemSpec

ComponentId ::=
                      identifier
                  |   number
                  |   "*"
```

NOTA – La producción AbsoluteReference no se utiliza en ningún otro punto de esta Recomendación | Norma Internacional. Se proporciona a los fines indicados en 14.1.

14.4 El "ModuleIdentifier" identifica un módulo ASN.1 (véase 12.1).

14.5 Cuando se utiliza la primera alternativa de "DefinitiveIdentifier" como parte del "ModuleIdentifier", el "DefinitiveIdentifier" identifica inequívocamente y de modo único el módulo desde el cual se hace referencia a un nombre.

14.6 La "typereference" hace referencia a cualquier tipo ASN.1 definido en el módulo identificado por "ModuleIdentifier".

14.7 El "ComponentId" de cada "ItemSpec" identifica un componente del tipo que ha sido identificado por "ItemId". Será el último "ComponentId" si el componente que identifica no es un tipo conjunto, secuencia, conjunto de, secuencia de o elección.

14.8 La forma "identificador" ("identifier") de "ComponentId" puede utilizarse si el "ItemId" progenitor es un tipo conjunto o secuencia y es preciso que sea uno de los identificadores del "NamedType" de la "ComponentTypeLists" de ese conjunto o de esa secuencia. También puede utilizarse si el "ItemId" identifica un tipo elección, y se requiere entonces que sea uno de los identificadores de un "NamedType" de la "AlternativeTypeLists" de ese tipo elección. No puede utilizarse en ninguna otra circunstancia.

14.9 La forma "número" ("number") de "ComponentId" puede utilizarse únicamente si el "ItemId" es un tipo secuencia de o conjunto de. El valor del número identifica el ejemplar del tipo en la secuencia de o en el conjunto de, identificando el valor "1" el primer ejemplar del tipo. El valor cero identifica un componente tipo entero conceptual (no explícitamente presente en transferencia) que contiene una cuenta del número de ejemplares del tipo en la secuencia de o en el conjunto de que están presentes en el valor del tipo circundante.

14.10 La forma "*****" de "ComponentId" puede utilizarse únicamente si el "ItemId" es una secuencia de o un conjunto de. Toda semántica asociada con la utilización de la forma "*****" de "ComponentId" se aplica a todos los componentes de la secuencia de y del conjunto de.

NOTA – En el siguiente ejemplo:

```
M DEFINITIONS ::= BEGIN
      T ::= SEQUENCE {
            a     BOOLEAN,
            b     SET OF INTEGER
      }
END
```

los componentes de "**T**" podrían ser referenciados por texto fuera de un módulo ASN.1 (o en un comentario), tal como:

```
-- si (@M.T.b.0 es impar) entonces:
--        (@M.T.b.* será un entero impar)
```

que se utiliza para indicar que si el número de componentes en **b** es impar, todos los componentes de **b** deben ser impares.

15 Asignación de tipos y valores

15.1 Para asignar un tipo a una "typereference" se utiliza la notación especificada por la producción "TypeAssignment":

```
TypeAssignment ::=
        typereference
        "::="
        Type
```

La "typereference" no será una palabra reservada en ASN.1 (véase 11.27).

15.2 Para asignar un valor a una "valuereference" se utiliza la notación especificada por las producciones "ValueAssignment" o "XMLValueAssignment":

```
ValueAssignment ::=
        valuereference
        Type
        "::="
        Value

XMLValueAssignment ::=
        valuereference
        "::="
        XMLTypedValue

XMLTypedValue ::=
        "<" & NonParameterizedTypeName  ">"
        XMLValue
        "</" & NonParameterizedTypeName ">"
    |   "<" & NonParameterizedTypeName "/>"
```

El valor que se asigna a la "valuereference" en el "ValueAssignment" es "Value" y está gobernado por "Type" y deberá ser una notación de un valor del tipo definido por "Type" (especificado en 15.3). El valor que se asigna a la "valuereference" en la "XMLValueAssignment" es "XMLValue" (véase 16.7) y debe ser una notación para un valor del tipo definido por "NonParameterizedTypeName" (como se especifica en 15.4). Si se trata del elemento "xmlasn1typename", entonces identifica el tipo incorporado ASN.1 en la fila correspondiente del cuadro 4 (véase también 13.3).

15.3 "Value" es una notación para un valor de un tipo si como el especificado en 16.7.

15.4 "XMLValue" es una notación para un valor de un tipo si "XMLValue" es una notación "XMLBuiltinValue" para el tipo (véase 16.10).

15.5 La segunda alternativa de "XMLTypedValue" (utilización de un rótulo elemento-vacío XML) puede emplearse sólo si está vacío un ejemplar de la producción "XMLValue".

NOTA – Si la producción "XMLValue" era una "xmlcstring" que contenía sólo espacios en blanco, ésta no estaría vacía y no podría emplearse la segunda alternativa.

15.6 A una "typereference" se le puede asignar un conjunto de valores mediante la notación especificada por la producción "ValueSetTypeAssignment":

```
ValueSetTypeAssignment ::=
        typereference
        Type
        "::="
        ValueSet
```

Esta notación asigna a la "typereference" el tipo definido como un subtipo del tipo denotado por "Type" que contiene exactamente los valores que están especificados en, o permitidos por, "ValueSet". La "typereference" no será una palabra reservada ASN.1 (véase 11.27) y podrá ser referenciada como un tipo. "ValueSet" se define en 15.7.

15.7 Un conjunto de valores gobernado por algún tipo será especificado por la notación "ValueSet":

```
ValueSet ::= "{" ElementSetSpecs "}"
```

El conjunto de valores comprende todos los valores, de los que habrá por lo menos uno, especificado por "ElementSetSpecs" (véase la cláusula 46).

15.8 La producción " ValueSetTypeAssignment " se amplia a:

typereference
Type
":: ="
"{" ElementSetSpecs "}"

A todos los efectos, incluida la aplicación de las reglas de codificación, esto se define exactamente equivalente a la utilización de la producción:

typereference
":: ="
Type
"(" ElementSetSpecs ")"

con las mismas especificaciones de "Type" y "ElementSetSpecs".

16 Definición de tipos y valores

16.1 Un tipo deberá especificarse por la notación "Type":

Type ::= BuiltinType | ReferencedType | ConstrainedType

16.2 Los tipos incorporados de ASN.1 se especifican por la notación "BuiltinType", que se define como sigue:

BuiltinType ::=
BitStringType
| **BooleanType**
| **CharacterStringType**
| **ChoiceType**
| **EmbeddedPDVType**
| **EnumeratedType**
| **ExternalType**
| **InstanceOfType**
| **IntegerType**
| **NullType**
| **ObjectClassFieldType**
| **ObjectIdentifierType**
| **OctetStringType**
| **RealType**
| **RelativeOIDType**
| **SequenceType**
| **SequenceOfType**
| **SetType**
| **SetOfType**
| **TaggedType**

Las diversas notaciones "BuiltinType" se definen en las siguientes cláusulas (de esta Recomendación | Norma Internacional, a menos que se indique otra cosa):

BitStringType	21
BooleanType	17
CharacterStringType	36
ChoiceType	28
EmbeddedPDVType	33
EnumeratedType	19
ExternalType	34
InstanceOfType	Rec. UIT-T X.681 | ISO/CEI 8824-2, anexo C
IntegerType	18
NullType	23
ObjectClassFieldType	Rec. UIT-T X.681 | ISO/CEI 8824-2, 14.1
ObjectIdentifierType	31
OctetStringType	22
RealType	20
RelativeOIDType	32
SequenceType	24

SequenceOfType	25
SetType	26
SetOfType	27
TaggedType	30

16.3 Los tipos referenciados de ASN.1 se especifican por la notación "ReferencedType":

ReferencedType ::=
DefinedType
| **UsefulType**
| **SelectionType**
| **TypeFromObject**
| **ValueSetFromObjects**

La notación "ReferencedType" proporciona otro posible medio de hacer referencia a algún otro tipo (y, en último término, a un tipo incorporado). Las diversas notaciones "ReferencedType" y la forma en que se determina el tipo a que ellas se refieren, se especifican en los siguientes lugares de esta Recomendación | Norma Internacional a menos que se indique otra cosa:

DefinedType	13.1
UsefulType	41.1
SelectionType	29
TypeFromObject	Rec. UIT-T X.681 | ISO/CEI 8824-2, cláusula 15
ValueSetFromObjects	Rec. UIT-T X.681 | ISO/CEI 8824-2, cláusula 15

16.4 El "ConstrainedType" se define en la cláusula 45.

16.5 Esta Recomendación | Norma Internacional exige la utilización de la notación "NamedType" al especificar los componentes de los tipos conjunto, de los tipos secuencia y de los tipos elección. La notación para "NamedType" es:

NamedType ::= identifier Type

16.6 El "identifier" se utiliza para referirse de manera inequívoca a los componentes de un tipo conjunto, un tipo secuencia o un tipo elección en la notación de valor, en las limitaciones de subtipo interiores y en las constricciones de la relación de componentes (véase la Rec. UIT-T X.682 | ISO/CEI 8824-3). No forma parte del tipo y no influye en él.

16.7 Un valor de algún tipo será especificado por la notación "Value" o por la notación "XMLValue":

Value ::=
BuiltinValue
| **ReferencedValue**
| **ObjectClassFieldValue**

XMLValue ::=
XMLBuiltinValue
| **XMLObjectClassFieldValue**

NOTA 1 – "ObjectClassFieldValue" y "XMLObjectClassFieldValue" se definen en la Rec. UIT-T X.681 | ISO/CEI 8824-2, 14.6.

NOTA 2 – "XMLValue" se utiliza sólo en "XMLTypedValue".

16.8 Si cualquier parte de la producción "XMLValue" resulta en un rótulo de comienzo XML inmediatamente seguido por un rótulo de final XML, posiblemente separado por un espacio en blanco insertado conforme a lo permitido en 11.1.4 (por ejemplo, <field1></field1>), estos dos rótulos XML y cualquier espacio en blanco presentes, podrán sustituirse por un solo rótulo de elemento vacío XML (<field1/>).

NOTA – Si está presente cualquier espacio en blanco, excepto el espacio en blanco insertado según lo permitido en 11.1.4, entre el carácter final ">" del rótulo de comienzo y el carácter inicial "<" del rótulo de final, la condición anterior no se cumple.

16.9 Los valores de tipo incorporados de ASN.1 pueden especificarse por la notación "XMLBuiltinValue" (véase 16.10) o "XMLBuiltinValue" , definidas como sigue:

BuiltinValue ::=
BitStringValue
| **BooleanValue**
| **CharacterStringValue**
| **ChoiceValue**
| **EmbeddedPDVValue**
| **EnumeratedValue**
| **ExternalValue**
| **InstanceOfValue**

| **IntegerValue**
| **NullValue**
| **ObjectIdentifierValue**
| **OctetStringValue**
| **RealValue**
| **RelativeOIDValue**
| **SequenceValue**
| **SequenceOfValue**
| **SetValue**
| **SetOfValue**
| **TaggedValue**

Cada una de las diversas notaciones "BuiltinValue" se define en la misma subcláusula que la correspondiente notación "BuiltinType", como se ha indicado en 16.2.

16.10 "XMLBuiltinValue" se define como sigue:

XMLBuiltinValue ::=
XMLBitStringValue
| **XMLBooleanValue**
| **XMLCharacterStringValue**
| **XMLChoiceValue**
| **XMLEmbeddedPDVValue**
| **XMLEnumeratedValue**
| **XMLExternalValue**
| **XMLInstanceOfValue**
| **XMLIntegerValue**
| **XMLNullValue**
| **XMLObjectIdentifierValue**
| **XMLOctetStringValue**
| **XMLRealValue**
| **XMLRelativeOIDValue**
| **XMLSequenceValue**
| **XMLSequenceOfValue**
| **XMLSetValue**
| **XMLSetOfValue**
| **XMLTaggedValue**

Cada una de las diversas notaciones "XMLBuiltinValue" se define en la misma cláusula que la notación "BuiltinType" correspondiente, como se indica en 16.2.

16.11 Los valores referenciados de ASN.1 se especifican por la notación "ReferencedValue":

ReferencedValue ::=
DefinedValue
| **ValueFromObject**

La notación "ReferencedValue" proporciona otro posible medio de referir a algún otro valor (y, en último término, a un valor incorporado). Las diversas notaciones "ReferencedValue", y la forma en que se determina el valor a que ellas se refieren, se especifican en los siguientes lugares de esta Recomendación | Norma Internacional (a menos que se indique otra cosa):

DefinedValue	13.1
ValueFromObject	Rec. UIT-T X.681 | ISO/CEI 8824-2, cláusula 15

16.12 Con independencia de si un tipo es un "BuiltinType", "ReferencedType" o "ConstrainedType", sus valores pueden ser especificados por un "BuiltinValue" o un "ReferencedValue" de ese tipo.

16.13 El valor de un tipo referenciado mediante la notación "NamedType" será definido por la notación "NamedValue" o cuando se utiliza como parte de un "XMLValue", mediante la notación "XMLNamedValue". Estas producciones son:

NamedValue ::= identifier Value

XMLNamedValue ::= "<" & identifier ">" XMLValue "</" & identifier ">"

donde el "identifier" es el mismo que se utiliza en la notación "NamedType".

NOTA – El "identifier" forma parte de la notación, no del valor propiamente dicho. Se utiliza para referirse inequívocamente a los componentes de un tipo conjunto, un tipo secuencia o un tipo elección.

16.14 La presencia implícita (véase 12.4) o explícita de un marcador de extensión (véase la cláusula 6) en la definición de un tipo no produce efecto sobre la notación de valor. Es decir, la notación de valor para un tipo con marcador de extensión es exactamente la misma que si no estuviera el marcador de extensión.

NOTA – La subcláusula 46.8 prohíbe que la notación de valor que se utilice en una constricción de subtipo haga referencia a un valor inexistente en la raíz de extensión del tipo progenitor.

17 Notación para el tipo booleano

17.1 El tipo booleano (boolean) (véase 3.6.7) se referenciará por la notación "BooleanType":

BooleanType ::= `BOOLEAN`

17.2 El rótulo para tipos definidos por esta notación es de clase universal, número 1.

17.3 El valor de un tipo booleano (véanse 3.6.73 y 3.6.38) se definirá por la notación "BooleanValue", o cuando se utiliza como un "XMLValue", mediante la notación "XMLBooleanValue". Estas producciones son:

BooleanValue ::= `TRUE` | `FALSE`

XMLBooleanValue ::=
"<" & "true" "/>"
| "<" & "false" "/>"

18 Notación para el tipo entero

18.1 El tipo entero (integer) (véase 3.6.41) se referenciará por la notación "IntegerType":

IntegerType ::=
`INTEGER`
| `INTEGER` "{" NamedNumberList "}"

NamedNumberList ::=
NamedNumber
| NamedNumberList "," NamedNumber

NamedNumber ::=
identifier "(" SignedNumber ")"
| identifier "(" DefinedValue ")"

SignedNumber ::=
number
| "-" number

18.2 La segunda alternativa de "SignedNumber" no se utilizará si el "número" ("number") es cero.

18.3 La "NamedNumberList" no es significativa en la definición de un tipo. Se utiliza solamente en la notación de valor especificada en 18.9.

18.4 La "valuereference" en "DefinedValue" deberá ser de tipo entero (integer).

NOTA – Puesto que un "identifier" no puede utilizarse para especificar el valor asociado con "NamedNumber", el "DefinedValue" nunca puede interpretarse erróneamente como un "IntegerValue". Por eso, en el caso siguiente:

```
a INTEGER ::= 1
T1 ::= INTEGER { a(2) }
T2 ::= INTEGER { a(3), b(a) }
c T2 ::= b
d T2 ::= a
```

`c` denota el valor 1, ya que no puede ser una referencia a la segunda ni a la tercera ocurrencia de **`a`**, y **`d`** denota el valor 3.

18.5 El valor de cada "SignedNumber" o "DefinedValue" que aparece en la "NamedNumberList" deberá ser diferente, y representa un valor distinguido del tipo entero.

18.6 Cada "identifier" que aparece en la "NamedNumberList" deberá ser diferente.

18.7 El orden de los "NamedNumber" en "NamedNumberList" no es significativo.

18.8 El rótulo para tipos definidos por esta notación es de clase universal, número 2.

18.9 El valor de un tipo entero será definido por la notación "IntegerValue", o cuando se utiliza como un "XMLValue", mediante la notación "XMLIntegerValue". Estas producciones son:

IntegerValue ::=
SignedNumber
| identifier

XMLIntegerValue ::=
SignedNumber
| "<" & identifier "/>"

18.10 El "identifier" del "IntegerValue" y del "XMLIntegerValue" será uno de los identificadores del "IntegerType" con el que está asociado el valor, y representará el número correspondiente.

NOTA – Cuando se referencia un valor entero para el cual se ha definido un "identifier", deberá preferirse la utilización de la forma "identifier" del "IntegerValue" y del "XMLIntegerValue".

18.11 Con un ejemplar de notación de valor para un tipo entero con un "NamedNumberList", cualquier ocurrencia de un nombre que sea tanto un "identifier" de la "NamedNumberList" como un nombre de referencia se interpretará como el "identifier".

19 Notación para el tipo enumerado

19.1 El tipo enumerado (enumerated) (véase 3.6.24) se referenciará por la notación "EnumeratedType":

EnumeratedType ::=
ENUMERATED "{" Enumerations "}"

Enumerations ::=
RootEnumeration
| RootEnumeration "," "..." ExceptionSpec
| RootEnumeration "," "..." ExceptionSpec "," AdditionalEnumeration

RootEnumeration ::= Enumeration

AdditionalEnumeration ::= Enumeration

Enumeration ::= EnumerationItem | EnumerationItem "," Enumeration

EnumerationItem ::= identifier | NamedNumber

NOTA 1 – Cada valor de un "EnumeratedType" tiene un identificador que está asociado con un entero distinto. Sin embargo, no se espera que los valores propiamente dichos tengan una semántica de entero. La especificación de la alternativa "NamedNumber" de "EnumerationItem" proporciona el control de la representación del valor, a fin de facilitar las extensiones compatibles.

NOTA 2 – No es necesario que los valores numéricos dentro de los "NamedNumber" en la "RootEnumeration" estén ordenados o sean contiguos, y los valores numéricos dentro de los "NamedNumber" en la "AdditionalEnumeration" están ordenados pero no son necesariamente contiguos.

19.2 Para cada "NamedNumber", el "identifier" y el "SignedNumber" serán distintos de todos los demás "identifier" y de los "SignedNumber" de la "Enumeration". Las subcláusulas 18.2 y 18.4 se aplican también a cada "NamedNumber".

19.3 A cada "EnumerationItem" (de un "EnumeratedType") que sea un "identifier" se le asigna un número entero no negativo distinto. Para la "RootEnumeration" se asignan los enteros sucesivos comenzando por 0, pero se excluyen los que han sido empleados en los "EnumerationItem" y son "NamedNumber".

NOTA – Un valor entero está asociado con un "EnumerationItem" para ayudar en la definición de las reglas de codificación. De no ser así, no se utiliza en la especificación ASN.1.

19.4 El valor de cada nueva "AdditionalEnumeration" será mayor que el de todas las "AdditionalEnumeration" definidas anteriormente en el tipo.

19.5 Cuando se utiliza un "NamedNumber" en la definición de un "EnumerationItem" en la "AdditionalEnumeration", los valores asociados con él serán diferentes del valor de todos los "EnumerationItem" definidos anteriormente (en este tipo), independientemente de que los "EnumerationItem" anteriormente definidos aparecieron o no en la raíz de enumeración. Por ejemplo:

```
A ::= ENUMERATED {a, b, ..., c(0)}      -- invalid, since both 'a' and 'c' equal 0
B ::= ENUMERATED {a, b, ..., c, d(2)}   -- invalid, since both 'c' and 'd' equal 2
C ::= ENUMERATED {a, b(3), ..., c(1)}   -- valid, 'c' = 1
D ::= ENUMERATED {a, b, ..., c(2)}      -- valid, 'c' = 2
```

19.6 El valor asociado con la primera alternativa de "EnumerationItem" en la "AdditionalEnumeration" que sea un "identifier" (no un "NamedNumber") será el valor más pequeño para el cual no está definido un "EnumerationItem" en la "RootEnumeration" y todos los anteriores "EnumerationItem" en la "AdditionalEnumeration" (si existen) son pequeños. Por ejemplo, todo lo que sigue es válido:

```
A ::= ENUMERATED {a, b, ..., c}         -- c = 2
B ::= ENUMERATED {a, b, c(0), ..., d}   -- d = 3
C ::= ENUMERATED {a, b, ..., c(3), d}   -- d = 4
D ::= ENUMERATED {a, z(25), ..., d}     -- d = 1
```

19.7 El tipo enumerado tiene un rótulo de clase universal, número 10.

19.8 El valor de un tipo enumerado se definirá por la notación "EnumeratedValue", o cuando se utiliza como un "XMLValue", mediante la notación "XMLEnumeratedValue". Estas producciones son:

EnumeratedValue ::= identifier

XMLEnumeratedValue ::= "<" & identifier "/>"

19.9 El "identifier" de "EnumeratedValue" y de "XMLEnumeratedValue" será igual al de un "identifier" de la secuencia "EnumeratedType" con la que está asociado el valor.

19.10 Dentro de un ejemplar de notación de valor para un tipo enumerado, cualquier ocurrencia de un nombre que sea tanto un "identifier" de la "Enumeration" como un nombre de referencia se interpretará como el "identifier".

20 Notación para el tipo real

20.1 El tipo real (véase 3.6.54) se referenciará por la notación "RealType":

RealType ::= `REAL`

20.2 El tipo real tiene un rótulo de clase universal número 9.

20.3 Los valores del tipo real son los valores **`PLUS-INFINITY`** y **`MINUS-INFINITY`** junto con los números reales que pueden ser especificados por la siguiente fórmula en la que M, B y E son enteros:

$$M \times B^{E}$$

siendo M la mantisa, B la base y E el exponente.

20.4 El tipo real tiene un tipo asociado que se utiliza para dar precisión a la definición de los valores abstractos del tipo real y que se utiliza también para soportar las notaciones de valor y subtipo del tipo real.

NOTA – Las reglas de codificación pueden definir un tipo diferente que se utiliza para especificar codificaciones, o puede especificar codificaciones sin referencia a los tipos asociados. En particular, la codificación en BER y PER proporciona una codificación decimal codificada en binario (BCD, *binary-coded decimal*) si la "base" es 10 y una codificación que permite la transformación eficiente hacia y desde las representaciones de coma flotante de los soportes físicos si la "base" es 2.

20.5 El tipo asociado para la definición de valores y a efectos de subtipificación es (con comentarios normativos):

```
SEQUENCE {
      mantissa    INTEGER,
      base  INTEGER (2|10),
      exponent    INTEGER
-- The associated mathematical real number is "mantissa"
-- multiplied by "base" raised to the power "exponent"
}
```

NOTA 1 – Los valores distintos de cero representados por "base" 2 y por "base" 10 se consideran valores abstractos distintos incluso si equivalen al mismo valor en números reales y pueden llevar semánticas de aplicación diferentes.

NOTA 2 – La notación **REAL (WHITH COMPONENTS { ... , base (10)})** puede utilizarse para restringir el conjunto de valores a valores abstractos en base 10 (y de manera similar para valores abstractos en base 2).

NOTA 3 – Este tipo es capaz de llevar una representación finita exacta de cualquier número que pueda ser almacenado en soportes físicos típicos de coma flotante y de cualquier número que tenga una representación finita decimal en caracteres.

20.6 El valor de un tipo real se definirá por la notación "RealValue", o cuando se utiliza en un "XMLValue", mediante la notación "XMLRealValue":

```
RealValue ::=
        NumericRealValue
    |   SpecialRealValue

NumericRealValue ::=
        realnumber
    |   "-" realnumber
    |   SequenceValue          -- Value of the associated sequence type

SpecialRealValue ::=
        PLUS-INFINITY
    |   MINUS-INFINITY
```

La segunda y tercera alternativas de "NumericReal value" no deben utilizarse para valores cero.

```
XMLRealValue ::=
    XMLNumericRealValue | XMLSpecialRealValue

XMLNumericRealValue ::=
        realnumber
    |   "-" realnumber
```

La segunda alternativa de "XMLNumericReal value" no debe utilizarse para valores cero.

```
XMLSpecialRealValue ::=
    "<" & PLUS-INFINITY "/>" | "<" & MINUS-INFINITY "/>"
```

20.7 Cuando la notación "realnumber" es utilizada, identifica el valor abstracto "base" 10 correspondiente. Si el "RealType" está constreñido a "base" 2, el "realnumber" identifica el valor abstracto "base" 2 correspondiente al valor decimal especificado por el "realnumber" o a una precisión definida localmente si no es posible una representación exacta.

21 Notación para el tipo cadena de bits

21.1 El tipo cadena de bits (bitstring) (véase 3.6.6) se referenciará por la notación "BitStringType":

```
BitStringType ::=
        BIT STRING
    |   BIT STRING "{" NamedBitList "}"

NamedBitList ::=
        NamedBit
    |   NamedBitList "," NamedBit

NamedBit ::=
        identifier "(" number ")"
    |   identifier "(" DefinedValue ")"
```

21.2 El primer bit de una cadena de bits se denomina bit al comienzo. El último bit de una cadena de bits se denomina bit final o bit de cola (trailing bit).

NOTA – Esta terminología se utiliza en la especificación de la notación de valor y en la definición de las reglas de codificación.

21.3 El "DefinedValue" deberá ser una referencia a un valor no negativo de tipo entero.

21.4 El valor de cada "number" o "DefinedValue" que aparezca en la "NamedBitList" deberá ser diferente y será el número (de la posición) de un bit distinguido en un valor bistring. El bit al comienzo de la cadena de bits se identifica mediante el "number" cero, y los bits subsiguientes tendrán valores consecutivos.

21.5 Cada "identifier" que aparece en la "NamedBitList" deberá ser diferente.

NOTA 1 – El orden de las secuencias de producción "NamedBit" de la "NamedBitList" no es significativo.

NOTA 2 – Puesto que un "identifier" que aparezca dentro de la "NamedBitList" no puede ser utilizado para especificar el valor asociado con un "NamedBit", el "DefinedValue" nunca podrá interpretarse erróneamente como un "IntegerValue". Por eso, en el caso siguiente:

```
a  INTEGER ::= 1
T1 ::= INTEGER { a(2) }
T2 ::= BIT STRING { a(3), b(a) }
```

la última ocurrencia de **a** denota el valor 1, ya que no puede ser una referencia a la segunda ni a la tercera ocurrencia de **a**.

21.6 La presencia de una "NamedBitList" no tiene efecto alguno en el conjunto de valores abstractos de este tipo. Están permitidos valores que tengan bits 1 distintos de los bits con nombre.

21.7 Cuando se utiliza una "NamedBitList" en la definición de un tipo cadena de bits, las reglas de codificación ASN.1 pueden añadir (o eliminar) libremente, de manera arbitraria, los bits 0 de cola a (o de) los valores que están siendo codificados o decodificados. Los diseñadores de aplicaciones deben asegurarse, por consiguiente, de que no se asocian semánticas diferentes con unos valores que sólo difieren en el número de bits 0 de cola.

21.8 Este tipo tiene un rótulo que es de clase universal, número 3.

21.9 El valor de un tipo cadena de bits se definirá por la notación "BitStringValue", o cuando se utiliza como un "XMLValue", mediante la notación "XMLBitStringValue". Estas producciones son:

BitStringValue ::=
bstring
| **hstring**
| **"{" IdentifierList "}"**
| **"{" "}"**
| **CONTAINING Value**

IdentifierList ::=
identifier
| **IdentifierList "," identifier**

XMLBitStringValue ::=
XMLTypedValue
| **Xmlbstring**
| **XMLIdentifierList**
| **empty**

XMLIdentifierList ::=
"<" & identifier "/>"
| **XMLIdentifierList "<" & identifier "/>"**

21.10 No se utilizará la alternativa "XMLTypedValue" a menos que la cadena de bits tenga una constricción de contenido que incluya un tipo ASN.1 y no incluya un **ENCODED BY**. Si se utiliza esta alternativa, el "XMLTypedValue" será un valor del tipo ASN.1 en la constricción de contenido.

21.11 No se utilizará la alternativa "XMLIdentifierList" a menos que la cadena de bits tenga una "NamedBitList".

21.12 Cada "identifier" de "BitStringValue" o "XMLBitStringValue" deberá ser el mismo que un "identifier" de la secuencia de producción "BitStringType" con la que está asociado el valor.

21.13 La alternativa "empty" denota una cadena de bits que no tiene bits.

21.14 Si la cadena de bits tiene bits con nombres, la notación "BitStringValue" o "XMLBitStringValue" denota un valor cadena de bits con unos en las posiciones de bits especificadas por los números correspondientes a los "identifier" y con ceros en todas las demás posiciones de bits.

NOTA – Para un "BitStringType que tiene una "NamedBitList", la secuencia de producción "**{**" "**}**" en "BitStringValue" y "empty" en "XMLBitStringValue" se utilizan para denotar la cadena de bits que no contiene ningún bit.

21.15 Cuando se utiliza la notación "bstring" o "XMLbstring", el bit inicial del valor de la cadena de bits está a la izquierda y el bit de cola del valor de la cadena de bits está a la derecha.

21.16 Cuando se utiliza la notación "hstring", el bit más significativo de cada dígito hexadecimal corresponde al bit situado más a la izquierda en la cadena de bits.

NOTA – Esta notación no constriñe en forma alguna la manera en que las reglas de codificación colocan una cadena de bits en octetos para la transferencia.

21.17 La notación "hstring" no debe utilizarse a menos que el valor cadena de bits esté constituido por un múltiplo de cuatro bits.

EJEMPLO

```
'A98A'H
```

y

```
'1010100110001010'B
```

son dos notaciones posibles para el mismo valor cadena de bits. Si el tipo se definió utilizando una "NamedBitList", el cero final (único) no forma parte del valor que, de esta forma, tiene una longitud de 15 bits. Si el tipo se definió sin una "NamedBitList", el cero final sí forma parte del valor, que tiene entonces una longitud de 16 bits.

21.18 La alternativa **CONTAINING** sólo puede utilizarse si se aplica una constricción de contenido al tipo cadena de bits que incluye **CONTAINING**. El "Value" debe ser, por consiguiente, la notación de valor de un valor del "Type" en el "ContentsConstraint" (véase la Rec. UIT-T X.682 | ISO/CEI 8824-3, cláusula 11).

NOTA – Esta notación de valor no puede aparecer nunca en una constricción de subtipo debido a que la Rec. UIT-T X.682 | ISO/CEI 8824-3, cláusula 11.3 prohíbe constricciones adicionales después de "ContentsConstraint", y el texto antes mencionado prohíbe su uso a menos que el gobernador tenga una "ContentsConstraint".

21.19 La alternativa **CONTAINING** debe utilizarse si hay una constricción de contenido en el tipo cadena de bits que no contiene **ENCODED BY**.

22 Notación para el tipo cadena de octetos

22.1 El tipo cadena de octetos (octetstring) (véase 3.6.49) se referenciará por la notación "OctetStringType":

OctetStringType ::= OCTET STRING

22.2 Este tipo tiene un rótulo que es de clase universal, número 4.

22.3 El valor de un tipo cadena de octetos se definirá por la notación: "OctetStringValue", o cuando se utiliza como un "XMLValue", mediante la notación "XMLOctetStringValue". Estas producciones son:

OctetStringValue ::=
bstring
| **hstring**
| **CONTAINING Value**

XMLOctetStringValue ::=
XMLTypedValue
| **xmlhstring**

22.4 No se utilizará la alternativa "XMLTypedValue" a menos que la cadena de octetos tenga una constricción de contenido que incluya un tipo ASN.1 y no incluya un **ENCODED BY**. Si se utiliza esta alternativa, el "XMLTypedValue" debe ser un valor del tipo ASN.1 en la constricción de contenido.

22.5 Al especificar las reglas de codificación para una cadena de octetos, los octetos se referencian por los términos primer octeto y octeto final o de cola, y los bits en un octeto se referencian por los términos bit más significativo y bit menos significativo.

22.6 Cuando se use la notación "bstring", el bit situado más a la izquierda de la notación "bstring" será el bit más significativo del primer octeto del valor de la cadena de octetos. Si la "bstring" no es múltiplo de ocho bits, se interpretará como si contuviera bits finales cero adicionales para hacer el próximo múltiplo de ocho.

22.7 Cuando se usa la notación "hstring" o "xmlhstring" el dígito hexadecimal más a la izquierda será el semiocteto más significativo del primer octeto..

22.8 Si la "hstring" es un número impar de dígitos hexadecimales, se interpretará como si contuviera un solo dígito hexadecimal cero final adicional. La "xmlhstring" no debe ser un número impar de dígitos hexadecimales.

22.9 La alternativa **CONTAINING** sólo puede utilizarse si hay una constricción de contenidos en el tipo cadena de octetos que incluye **CONTAINING**. El "Value" será, por consiguiente, la notación de valor de un valor del "Type" en la "ContentsConstraint" (véase la Rec. UIT-T X.682 | ISO/CEI 8824-3, cláusula 11).

NOTA – Esta notación de valor no puede aparecer nunca en una constricción de subtipo debido a que la Rec. UIT-T X.682 | ISO/CEI 8824-3, cláusula 11.3 prohíbe constricciones adicionales después de una "ContentsConstraint", y el texto antes mencionado prohíbe su uso a menos que el gobernador tenga una "ContentsConstraint".

22.10 La alternativa **CONTAINING** debe utilizarse si hay una constricción de contenido en el tipo cadena de octetos que no contiene **ENCODED BY**.

23 Notación para el tipo nulo

23.1 El tipo nulo (null) (véase 3.6.44) se referenciará por la notación "NullType":

NullType ::= NULL

23.2 Este tipo tiene un rótulo que es de clase universal, número 5.

23.3 El valor del tipo nulo se referenciará por la notación "NullValue", o cuando se utiliza como un "XMLValue", mediante la notación "XMLNullValue". Estas producciones son:

NullValue ::= NULL

XMLNullValue ::= empty

24 Notación para tipos secuencia

24.1 La notación para definir un tipo secuencia (sequence) (véase 3.6.60) será "SequenceType":

```
SequenceType ::=
        SEQUENCE "{" "}"
    |   SEQUENCE "{" ExtensionAndException OptionalExtensionMarker "}"
    |   SEQUENCE "{" ComponentTypeLists "}"

ExtensionAndException ::= "..." | "..." ExceptionSpec

OptionalExtensionMarker ::= "," "..." | empty

ComponentTypeLists ::=
        RootComponentTypeList
    |   RootComponentTypeList "," ExtensionAndException ExtensionAdditions
            OptionalExtensionMarker
    |   RootComponentTypeList "," ExtensionAndException ExtensionAdditions
            ExtensionEndMarker "," RootComponentTypeList
    |   ExtensionAndException ExtensionAdditions ExtensionEndMarker ","
            RootComponentTypeList
    |   ExtensionAndException ExtensionAdditions OptionalExtensionMarker

RootComponentTypeList ::= ComponentTypeList

ExtensionEndMarker ::= "," "..."

ExtensionAdditions ::=
        "," ExtensionAdditionList
    |   empty

ExtensionAdditionList ::=
        ExtensionAddition
    |   ExtensionAdditionList "," ExtensionAddition

ExtensionAddition ::=
        ComponentType
    |   ExtensionAdditionGroup

ExtensionAdditionGroup ::= "[[" VersionNumber ComponentTypeList "]]"

VersionNumber ::= empty | number ":"

ComponentTypeList ::=
        ComponentType
    |   ComponentTypeList "," ComponentType
```

ComponentType ::=
NamedType
| **NamedType** **OPTIONAL**
| **NamedType** **DEFAULT** **Value**
| **COMPONENTS OF** **Type**

24.2 Cuando aparece una producción "ComponentTypeLists" dentro de la definición de un módulo para el que se ha seleccionado rotulación automática (véase 12.3) y ninguna de las ocurrencias de "NamedType" en cualquiera de las tres primeras alternativas para "ComponentType" contiene un "TaggedType", se selecciona la transformación de rotulación automática para la totalidad de la "ComponentTypeLists"; de no ser así, no se selecciona.

NOTA 1 – La utilización de la notación "TaggedType" dentro de la definición de la lista de componentes de un tipo secuencia da el control de los rótulos al especificador, al contrario de lo que ocurre con la asignación automática por el mecanismo de rotulación automática. Por ello, en el siguiente caso:

```
T ::= SEQUENCE { a INTEGER, b [1] BOOLEAN, c OCTET STRING }
```

no se aplica rotulación automática a la lista de componentes a, b, c, incluso si esta definición de tipo secuencia T se produce dentro de un módulo para el que ha sido seleccionada la rotulación automática.

NOTA 2 – Sólo las ocurrencias de la producción "ComponentTypeLists" dentro de un módulo en el que se ha seleccionado la rotulación automática son candidatas a la transformación por rotulación automática.

24.3 La decisión de aplicar la transformación de rotulación automática se toma individualmente para cada ocurrencia de la "ComponentTypeLists" y *con anterioridad* a la transformación **COMPONENTS OF** especificada por 24.4. Sin embargo, tal como se especifica en 24.7 a 24.9, la transformación de rotulación automática, si se aplica, lo es *después* de la transformación **COMPONENTS OF**.

NOTA – Lo anterior tiene como efecto que la aplicación de rótulos automáticos queda suprimida por los rótulos presentes de manera explícita en la "ComponentTypeLists", pero no por los rótulos presentes en el "Type" que sigue a **COMPONENTS OF**.

24.4 "Type" de la notación "**COMPONENTS OF** Type" deberá ser un tipo secuencia. La notación "**COMPONENTS OF** Type" deberá utilizarse para definir la inclusión, en este punto de la lista de componentes, de todos los tipos componente del tipo referenciado, salvo para cualesquiera marcador de extensión y adiciones de extensión que puedan estar presentes en el "Type". (Solamente se incluye la "RootComponentTypeList" del "Type" del "**COMPONENTS OF** Type"; los marcadores de extensión y las adiciones de extensión, si existen, son ignorados por la notación "**COMPONENTS OF** Type".) Toda constricción de subtipo aplicada al tipo referenciado es ignorada por esta transformación.

NOTA – Esta notación se realiza lógicamente antes de la cumplimentación de los requisitos de las siguientes subcláusulas.

24.5 En cada una de las subcláusulas siguientes se identifica una serie de apariciones de "ComponentType" en las adiciones de raíz o de extensión, o en ambas. La regla de 24.5.1 se aplicará a todas esas series.

24.5.1 Cuando se produzcan una o varias apariciones consecutivas de "ComponentType" y todas estén marcadas **OPTIONAL** o **DEFAULT**, los rótulos de esos "ComponentType" y de cualquier tipo componente que los siga en la serie serán distintos (véase la cláusula 30). Si se seleccionó rotulación automática, la exigencia de que los rótulos sean distintos sólo es aplicable después de que se haya efectuado la rotulación automática y se satisfará siempre, si se ha aplicado la rotulación automática.

24.5.2 La subcláusula 24.5.1 se aplicará a la serie de los "ComponentType" en la raíz.

24.5.3 La subcláusula 24.5.1 se aplicará a la serie de los "ComponentType" en las adiciones de raíz o de extensión, en el orden textual de su aparición en la definición de tipo (ignorando todos los corchetes de versión y la notación de puntos suspensivos). (Véase asimismo 48.7.)

24.6 Cuando se utiliza la tercera o cuarta alternativa de "ComponentTypeLists", todos los "ComponentType" en adiciones de extensión tendrán rótulos distintos de los "ComponentType" que siguen textualmente, hasta el primero de esos "ComponentType" (inclusive) que no esté marcado **OPTIONAL** o **DEFAULT** en el "RootComponentTypeList" final (si lo hay). (Véase asimismo 48.7.)

24.7 La transformación de rotulación automática de una ocurrencia de "ComponentTypeLists" se efectúa lógicamente *después* de la transformación especificada por 24.4, pero solamente si 24.2 determina que debe aplicarse a esa ocurrencia de "ComponentTypeLists". La transformación de rotulación automática repercute en cada "ComponentType" de la "ComponentTypeLists" al sustituir el "Type" que figura originalmente en la producción "NamedType" por una ocurrencia del "TaggedType" de sustitución especificado en 24.9.

24.8 Si está en vigor la rotulación automática y los "ComponentType" en la raíz de extensión no tienen rótulos, entonces ningún "ComponentType" dentro de la "ExtensionAdditionList" deberá ser un "TaggedType" (tipo rotulado).

24.9 Si está en vigor la rotulación automática , el "TaggedType" de sustitución se especifica de la siguiente manera:

a) la notación de "TaggedType" de sustitución utiliza la alternativa "Tag Type";

b) la "Class" del "TaggedType" de sustitución es vacío (es decir, la rotulación es específica del contexto);

c) el "ClassNumber" del "TaggedType" de sustitución es el rótulo de valor cero para el primer "ComponentType" de la "RootComponentTypeList", uno para el segundo, y así sucesivamente, procediendo por números de rótulo crecientes;

d) el "ClassNumber" del "TaggedType" de sustitución del primer "ComponentType" en la "ExtensionAdditionList" es cero si falta la "RootComponentTypeList", o bien es una unidad superior al "ClassNumber" más grande de la "RootComponentTypeList", teniendo el siguiente "ComponentType" de la "ExtensionAdditionList" un "ClassNumber" una unidad superior al primero, y así sucesivamente, procediendo con números de rótulo crecientes;

e) el "Type" del "TaggedType" de sustitución es el "Type" original que se sustituye.

NOTA 1 – Las reglas que rigen la especificación de la rotulación implícita o de la rotulación explícita de los "TaggedType" de sustitución se indican en 30.6. La rotulación automática es siempre rotulación implícita, a menos que el "Type" sea un tipo elección o una notación de tipo abierto o una referencia ficticia "DummyReference" (véase la Rec. UIT-T X.683 | ISO/CEI 8824-4, 8.3), en cuyo caso se trata de rotulación explícita.

NOTA 2 – Una vez satisfecha 24.7, los rótulos de los componentes quedan determinados por completo y no son modificados incluso cuando el tipo secuencia es referenciado en la definición de un componente dentro de otra "ComponentTypeLists", para el que es aplicable la transformación de rotulación automática. Así pues, en el caso siguiente:

```
T ::= SEQUENCE { a   Ta,  b   Tb,  c   Tc }
E ::= SEQUENCE { f1  E1,  f2  T,   f3  E3 }
```

la rotulación automática aplicada a los componentes de **E** nunca afecta a los rótulos atribuidos a los componentes **a**, **b** y **c** de **T**, cualquiera que sea el entorno de rotulación de **T**. Si **T** se define en un entorno de rotulación automática y E no está en un entorno de rotulación automática, se sigue aplicando rotulación automática a los componentes **a**, **b** y **c** de **T**.

NOTA 3 – Cuando aparece un tipo secuencia como el "Type" en "**COMPONENTS OF** Type", cada ocurrencia de "ComponentType" en él es reproducida por la aplicación de 24.4 antes de la posible aplicación de la rotulación automática al tipo secuencia referenciador. Así pues, en el caso siguiente:

```
T ::= SEQUENCE { a  Ta,  b  SEQUENCE { b1  T1,  b2  T2,  b3  T3},  c Tc }
W ::= SEQUENCE { x  Wx,  COMPONENTS OF  T, y  Wy }
```

no es preciso que los rótulos de **a**, **b** y **c** dentro de **T** sean los mismos que los rótulos de **a**, **b** y **c** dentro de **W** si **W** se ha definido en un entorno de rotulación automática, pero los rótulos de **b1**, **b2** y **b3** son los mismos tanto en **T** como en **W**. En otras palabras, la transformación de rotulación automática sólo se aplica una vez a una "ComponentTypeLists" dada.

NOTA 4 – La subtipificación no influye en la rotulación automática.

NOTA 5 – En presencia de rotulación automática, la inserción de componentes nuevos en cualquier ubicación que no sea el punto de inserción de extensión (véase 3.6.29), puede dar lugar a cambios en otros componentes debido al efecto lateral de la modificación de los rótulos, causando así problemas de interfuncionamiento con una versión más antigua de la especificación.

24.10 Si están presentes **OPTIONAL** o **DEFAULT**, puede omitirse el valor correspondiente en un valor del tipo nuevo.

24.11 Si ocurre **DEFAULT**, la omisión de un valor para ese tipo deberá equivaler exactamente a la inserción del valor definido por "Value", que será una notación de valor para un valor del tipo definido por "Type" en la secuencia de producción "NamedType".

24.12 El valor correspondiente a un "ExtensionAdditionGroup" (todos los componentes juntos) es facultativo. Sin embargo, si este valor está presente, el valor correspondiente a los componentes dentro de la "ComponentTypeList" entre corchetes que no están marcados como **OPTIONAL** o **DEFAULT** deberá estar presente.

24.13 Los identificadores de todas las secuencias de producción "NamedType" de la "ComponentTypeLists" (junto con los obtenidos por expansión de **COMPONENTS OF**) serán distintos.

24.14 No se especificará un valor para un tipo adición de extensión dado a menos que se especifiquen valores para todos los tipos adición de extensión que no están marcados como **OPTIONAL** o **DEFAULT** que estén lógicamente entre el tipo adición de extensión y la raíz de extensión.

NOTA 1 – Cuando el tipo haya crecido de la raíz de extensión (versión 1) a la versión 2 y la versión 3 por adición de adiciones de extensión, la presencia en una codificación de cualquier adición de la versión 3 exigirá la presencia de una codificación de todas las adiciones en la versión 2 que no estén marcadas **OPTIONAL** o **DEFAULT**.

NOTA 2 – Los "ComponentType" que son adiciones de extensión pero no están contenidos dentro de un "ExtensionAdditionGroup" deberán codificarse siempre si no son marcados como **OPTIONAL** o **DEFAULT**, salvo cuando el valor abstracto se está reenviando desde un emisor que utiliza una versión anterior de la sintaxis abstracta en la cual no se define el "ComponentType".

NOTA 3 – Se recomienda el uso de la producción "ExtensionAdditionGroup" porque:

a) puede dar como resultado codificaciones más compactas dependiendo de las reglas de codificación (por ejemplo, PER),

b) la sintaxis es más precisa, lo que indica claramente que un valor de un tipo definido en la "ExtensionAdditionList" y que no está marcado como **OPTIONAL** o **DEFAULT** debe siempre estar presente en una codificación si el grupo adición de extensión en el que se define está codificado (comparar con la nota 1),

c) la sintaxis clarifica que tipos en una "ExtensionAdditionList" deben ser soportados por una aplicación como un grupo.

24.15 Se utilizará una producción "VersionNumber" únicamente si todas las "ExtensionAdditions" y las "ExtensionAdditionAlternatives" dentro del módulo son "ExtensionAdditionGroup" o ExtensionAdditionAlternativesGroup" con "VersionNumbers". El "number" en cada "VersionNumber" de un "ExtensionAdditionGroup" debe ser mayor que o igual a dos, y será mayor que el "number" en cualquier "ExtensionAdditionGroup" precedente dentro de un punto de inserción.

NOTA 1 – El convenio seguido aquí es que la especificación sin grupos de adición de extensión es la versión 1, por lo que el primer grupo de adición de extensión añadido tendrá un número mayor que o igual a 2. Cuando se necesita una sola producción "ExtensionAddition" para una "ExtensionAdditions", se puede utilizar un "ExtensionAdditionGroup" con una sola "ExtensionAddition".

NOTA 2 – Las restricciones a la utilización de "VersionNumber" se aplican solamente dentro de un módulo único y no imponen limitaciones sobre tipos importados.

24.16 Todos los tipos secuencia tienen un rótulo que es de clase universal, número 16.

NOTA – Los tipos Sequence-of (secuencia de) tiene el mismo rótulo que los tipos secuencia (véase 25.2).

24.17 La notación para la definición de un valor de un tipo secuencia será "SequenceValue", o cuando se utiliza como un "XMLValue", "XMLSequenceValue". Estas producciones son:

```
SequenceValue ::=
        "{" ComponentValueList "}"
    |   "{" "}"

ComponentValueList ::=
        NamedValue
    |   ComponentValueList "," NamedValue

XMLSequenceValue ::=
        XMLComponentValueList
    |   empty

XMLComponentValueList ::=
        XMLNamedValue
    |   XMLComponentValueList XMLNamedValue
```

24.18 La notación "**{**" "**}**" o "empty" sólo se utilizará si:

a) todas las secuencias "ComponentType" del "SequenceType" están marcadas **DEFAULT** u **OPTIONAL** y todos los valores han sido omitidos; o

b) la notación de tipo era **SEQUENCE{}**.

24.19 Deberá haber un "NamedValue" o "XMLNamedValue" para cada "NamedType" del "SequenceType" que no esté marcado **OPTIONAL** o **DEFAULT**, y los valores deberán estar en el mismo orden que las secuencias "NamedType" correspondientes.

25 Notación para tipos secuencia de

25.1 La notación para definir un tipo secuencia de (sequence-of) (véase 3.6.61) a partir de otro tipo será "SequenceOfType".

SequenceOfType ::= SEQUENCE OF Type | SEQUENCE OF NamedType

NOTA – Si se necesita una letra inicial mayúscula para un nombre de rótulo XML que se utiliza en una notación de valor XML para el "SequenceOfType", entonces debería utilizarse la primera alternativa. (El nombre del rótulo XML se forma, por consiguiente, a partir del nombre del "Type".)

25.2 Todos los tipos secuencia de tienen un rótulo que es de clase universal, número 16.

NOTA – Los tipos secuencia tienen el mismo rótulo que los tipos secuencia de (véase 24.16).

25.3 La notación para definir un valor de un tipo secuencia de será "SequenceOfValue", o cuando se utiliza como un "XMLValue", "XMLSequenceOfValue". Estas producciones son:

```
SequenceOfValue ::=
        "{" ValueList "}"
    |   "{" NamedValueList "}"
    |   "{"  "}"

ValueList ::=
        Value
    |   ValueList "," Value

NamedValueList ::=
        NamedValue
    |   NamedValueList "," NamedValue

XMLSequenceOfValue ::=
        XMLValueList
    |   XMLDelimitedItemList
    |   XMLSpaceSeparatedList
    |   empty

XMLValueList ::=
        XMLValueOrEmpty
    |   XMLValueOrEmpty XMLValueList

XMLValueOrEmpty ::=
        XMLValue
    |   "<" & NonParameterizedTypeName "/>"

XMLSpaceSeparatedList ::=
        XMLValueOrEmpty
    |   XMLValueOrEmpty " " XMLSpaceSeparatedList

XMLDelimitedItemList ::=
        XMLDelimitedItem
    |   XMLDelimitedItem XMLDelimitedItemList

XMLDelimitedItem ::=
            "<" & NonParameterizedTypeName ">" XMLValue
            "</" & NonParameterizedTypeName ">"
    |   "<" & identifier ">" XMLValue "</" & identifier ">"
```

La notación "`{`" "`}`" se utiliza cuando el "SequenceOfValue" o "XMLSequenceOfValue" es una lista vacía.

NOTA 1 – La significación de la semántica puede colocarse en el orden de estos valores.

NOTA 2 – La producción "XMLSpaceSeparatedList" no se utiliza en esta Recomendación | Norma Internacional ni en la notación de valor XML. Se proporciona a fin de facilitar la especificación del empleo de "XMLSpaceSeparatedList" en las codificaciones de "IntegerType", "RealType", "ObjectIdentifierType", "RelativeOIDType", y de los tipos útiles **`GeneralizedTime`** y **`UTCTime`**. Asimismo, es posible especificar la utilización de "XMLValueList" en lugar de "XMLDelimitedItemList" para algunos ejemplares de "**`SEQUENCE OF SEQUENCE`**" y "**`SEQUENCE OF SET`**".

25.4 Si el "XMLValue" para el componente es "empty", se seleccionará la segunda alternativa de "XMLValueOrEmpty" para representar ese valor del componente.

25.5 Las producciones "XMLValueList", o "XMLDelimitedItemList" se utilizarán de acuerdo con la columna 2 del cuadro 5, donde el "Type" del componente está indicado en la columna 1.

Cuadro 5 – Notación "XMLSequenceOfValue" y "XMLSetOfValue" para tipos ASN.1

Tipo ASN.1	Notación de valor XML
BitStringType	XMLDelimitedItemList
BooleanType	XMLValueList
CharacterStringType	XMLDelimitedItemList
ChoiceType	XMLValueList
ConstrainedType	*Véase 25.7*
DefinedType	*Véase 25.9*
EmbeddedPDVType	XMLDelimitedItemList
EnumeratedType	XMLValueList
ExternalType	XMLDelimitedItemList
InstanceOfType	*Véase la Rec. UIT-T X.681 | ISO/CEI 8824-2, C.9*
IntegerType	XMLDelimitedItemList
NullType	XMLValueList
ObjectClassFieldType	*Véase la Rec. UIT-T X.681 | ISO/CEI 8824-2, 14.10 y 14.11*
ObjectIdentifierType	XMLDelimitedItemList
OctetStringType	XMLDelimitedItemList
RealType	XMLDelimitedItemList
RelativeOIDType	XMLDelimitedItemList
SelectionType	*Véase 25.8*
SequenceType	XMLDelimitedItemList
SequenceOfType	XMLDelimitedItemList
SetType	XMLDelimitedItemList
SetOfType	XMLDelimitedItemList
TaggedType	*See 25.6*
UsefulType (GeneralizedTime)	XMLDelimitedItemList
UsefulType (UTCTime)	XMLDelimitedItemList
UsefulType (ObjectDescriptor)	XMLDelimitedItemList
TypeFromObject	*Véase la Rec. UIT-T X.681 | ISO/CEI 8824-2, 15.6*
ValueSetFromObjects	*Véase la Rec. UIT-T X.681 | ISO/CEI 8824-2, 15.6*

25.6 Si el "Type" del componente es un "TaggedType", el tipo que determina la notación "XMLSequenceOfValue" será el "Type" en el "TaggedType" (véase 30.1). Si éste es él mismo un "TaggedType", esta subcláusula 25.6 se aplicará recurrentemente.

25.7 Si el "Type" del componente es un "ConstrainedType", el tipo que determina la notación "XMLSequenceOfValue" será el "Type" en el "ConstrainedType" (véase 45.1). Si éste es él mismo un "ConstrainedType", esta subcláusula 25.7 se aplica recursivamente.

25.8 Si el "Type" del componente es un "SelectionType", el tipo que determina la notación "XMLSequenceOfValue" será el tipo a que hace referencia el "SelectionType" (véase la cláusula 29).

25.9 Si el "Type" del componente es un "DefinedType" el tipo que determina la notación "XMLSequenceOfValue" será el tipo a que hace referencia el "DefinedType" (véase 13.1).

25.10 La segunda alternativa de "XMLDelimitedItem" será utilizada únicamente si "SequenceOfType" contiene un "identifier", y el "identifier" en el "XMLDelimitedItem" será ese "identifier".

25.11 Si se utiliza la primera alternativa de "XMLDelimitedItem" y si el componente del tipo "sequence-of" (sin tener en cuenta eventuales rótulos) es una "typereference" o una "ExternalTypeReference", el "NonParameterizedTypeName" será esa "typereference" o "ExternalTypeReference"; de lo contrario, será el "xmlasn1typename" especificado en el cuadro 4 correspondiente al tipo built-in (incorporado) del componente.

25.12 Si se utiliza la primera alternativa de "SequenceOfType" deberá utilizarse la primera alternativa de "SequenceOfValue". Cada "Value" de la "ValueList" de "SequenceOfValue", y cada "XMLValue" en las alternativas de "XMLSequenceValue" deberá ser del tipo especificado en el "SequenceOfType".

25.13 Si se utiliza la segunda alternativa de "SequenceOfType" deberá utilizarse la segunda alternativa de "SequenceOfValue", y cada "NamedValue" de la "NamedValueList" deberá contener un "Value" del tipo especificado en el "NamedType" del "SequenceOfType". El "identifier" en los "NamedValue" será el "identifier" en el "NamedType" del "SequenceOfType".

26 Notación para tipos conjunto

26.1 La notación para definir un tipo conjunto (set) (véase 3.6.64) a partir de otros tipos será "SetType":

```
SetType ::=
        SET "{" "}"
    |   SET "{" ExtensionAndException OptionalExtensionMarker "}"
    |   SET "{" ComponentTypeLists "}"
```

"ComponentTypeLists", "ExtensionAndException" y "OptionalExtensionMarker" se especifican en 24.1.

26.2 El "Type" de la notación "**COMPONENTS OF** Type" deberá ser un tipo conjunto. La notación "**COMPONENTS OF** Type" deberá utilizarse para definir la inclusión, en este punto de la lista de componentes, de todos los tipos de componentes del tipo referenciado, excepto para las adiciones de extensión y marcadores de extensión que puedan estar presentes en el "Type". (Solamente se incluye la "RootComponentTypeList" del "Type" del "**COMPONENTS OF** Type"; los marcadores de extensión y las adiciones de extensión, si existen, son ignorados por la notación "**COMPONENTS OF** Type".) Esta transformación ignora toda limitación de subtipo aplicada al tipo referenciado.

NOTA – Esta transformación se realiza lógicamente antes de la satisfacción de los requisitos de las siguientes subcláusulas.

26.3 Todos los tipos "ComponentType" de un tipo conjunto deberán tener rótulos diferentes (véase la cláusula 30). El rótulo de cada nuevo "ComponentType" añadido a la "ExtensionAdditions" será canónicamente mayor (véase 8.6) que los de los otros componentes de la "ExtensionAdditions".

NOTA – Cuando el "TagDefault" del módulo en que aparece esta notación es **AUTOMATIC TAGS**, esto se consigue con independencia de los "ComponentType" reales, como resultado de la aplicación de 24.7. (Véase asimismo 48.7)

26.4 Las subcláusulas 24.2 y 24.7 a 24.13 se aplican también a los tipos conjunto.

26.5 Todos los tipos conjunto tienen un rótulo de clase universal, número 17.

NOTA – Los tipos conjunto de (Set-of) tienen el mismo rótulo que los tipos conjunto (véase 27.2).

26.6 El orden de los valores en un tipo conjunto no debe tener asociada ninguna semántica.

26.7 La notación para definir un valor de un tipo conjunto será "SetValue", o cuando se utiliza como un "XMLValue", "XMLSetValue". Estas producciones son:

```
SetValue ::=
        "{" ComponentValueList "}"
    |   "{" "}"

XMLSetValue ::=
        XMLComponentValueList
    |   empty
```

"ComponentValueList" y "XMLComponentValueList" se especifican en 24.17.

26.8 El "SetValue" y "XMLSetValue" serán solamente "**{**" "**}**" y "empty" respectivamente si:

a) todas las secuencias "ComponentType" del "SetType" están marcadas **DEFAULT** u **OPTIONAL**, y todos los valores han sido omitidos; o

b) la notación de tipo era **SET{}**.

26.9 Deberá haber un "NamedValue" o "XMLNamedValue" para cada "NamedType" en el "SetType" que no esté marcado **OPTIONAL** o **DEFAULT**.

NOTA – Estos "NamedValue" o "XMLNamedValue" pueden aparecer en cualquier orden.

27 Notación para tipos conjunto de

27.1 La notación para definir un tipo conjunto de (set-of) (véase 3.6.65) a partir de otro tipo será "SetOfType".

SetOfType ::=
    SET OF Type
  | **SET OF NamedType**

NOTA – Si se necesita una letra inicial mayúscula para un nombre de rótulo XML que se utiliza en una notación de valor XML para el "SetOfType", entonces debería utilizarse la primera alternativa. (El nombre del rótulo XML se forma, por consiguiente, a partir del nombre del "Type".)

27.2 Todos los tipos conjunto de tienen un rótulo de clase universal, número 17.

NOTA – Los tipos conjunto tienen el mismo rótulo que los tipos conjunto de (Set-of) (véase 26.5).

27.3 La notación para definir un valor de un tipo conjunto de será "SetOfValue", o cuando se utiliza como un "XMLValue", "XMLSetOfValue". Estas producciones son:

SetOfValue ::=
    "{" ValueList "}"
  | **"{" NamedValueList "}"**
  | **"{" "}"**

XMLSetOfValue ::=
    XMLValueList
  | **XMLDelimitedItemList**
  | **XMLSpaceSeparatedList**
  | **empty**

"ValueList", "NamedValueList" y las alternativas de "XMLSetOfValue" se especifican en 25.3. La notación "**{**" "**}**" o "empty" se utiliza cuando el "SetOfValue" o el "XMLSetOfValue" es una lista vacía.

NOTA 1 – No debe darse significado semántico al orden de estos valores.

NOTA 2 – No es necesario que las reglas de codificación conserven el orden de estos valores.

NOTA 3 – El tipo conjunto de no es un conjunto matemático de valores, por lo tanto, como un ejemplo, para **SET OF INTEGER** los valores **{ 1 }** y **{ 1 1 }** son distintos.

27.4 Si se utiliza la primera alternativa de "SetOfType" deberá utilizarse la primera alternativa de "SetOfValue". Cada "Value" de la "ValueList" de "SetOfValue", y cada "XMLValue" en las alternativas de "XMLSetValue" deberá ser del tipo especificado en el "SetOfType".

27.5 Si se utiliza la segunda alternativa de "SetOfType" deberá utilizarse la segunda alternativa de "SetOfValue", y cada secuencia "NamedValue" en la "NamedValueList" deberá contener un "Value" del tipo especificado en el "NamedType" del "SetOfType". El "identifier" en los "NamedValue" será el "identifier" en el "NamedType" del "SetOfType".

28 Notación para tipos elección

28.1 La notación para definir un tipo elección (choice) (véase 3.6.13) a partir de otros tipos será "ChoiceType":

ChoiceType ::= CHOICE "{" AlternativeTypeLists "}"

AlternativeTypeLists ::=
    RootAlternativeTypeList
  | **RootAlternativeTypeList "," ExtensionAndException ExtensionAdditionAlternatives OptionalExtensionMarker**

RootAlternativeTypeList ::= AlternativeTypeList

ExtensionAdditionAlternatives ::=
    "," ExtensionAdditionAlternativesList
  | **empty**

ExtensionAdditionAlternativesList ::=
    ExtensionAdditionAlternative
  | **ExtensionAdditionAlternativesList "," ExtensionAdditionAlternative**

ExtensionAdditionAlternative ::=
ExtensionAdditionAlternativesGroup
| **NamedType**

ExtensionAdditionAlternativesGroup ::=
"[[" VersionNumber AlternativeTypeList "]]"

AlternativeTypeList ::=
NamedType
| **AlternativeTypeList "," NamedType**

NOTA – "`T ::= CHOICE { a A }`" y **A** no son el mismo tipo y pueden estar codificados diferentemente mediante reglas de codificación.

28.2 Cuando la producción "AlternativeTypeLists" ocurra dentro de la definición de un módulo para el que se haya seleccionado rotulación automática (véase 12.3), y ninguna de las ocurrencias de "NamedType" de las "AlternativeTypeList" contenga un "TaggedType", se seleccionará la transformación de rotulación automática para la totalidad de la "AlternativeTypeLists"; de no ser así, no se seleccionará.

28.3 Los tipos definidos en cualquiera de las producciones "AlternativeTypeList" en "AlternativeTypeLists" deberán tener rótulos distintos (véanse las cláusulas 30 y 48.7). Si se seleccionó rotulación automática, la exigencia de que los rótulos sean distintos sólo es aplicable después de que se haya efectuado la rotulación automática y se satisfará siempre.

28.4 Si se está aplicando rotulación automática y los "NamedType" en la raíz de extensión no tienen rótulos, ningún "NamedType" de la "ExtensionAdditionAlternativesList" deberá ser un tipo rotulado.

28.5 La transformación de rotulación automática repercute en el "NamedType" de las "AlternativeTypeLists" sustituyendo el "Type" originalmente en la producción "NamedType" por un "TaggedType" de sustitución. El "TaggedType" de sustitución se especifica como sigue:

a) la notación "TaggedType" de sustitución utiliza la alternativa "Tag Type";

b) la "Class" de "TaggedType" de sustitución está vacía (es decir, la rotulación es específica del contexto);

c) el "ClassNumber" en el "TaggedType" de sustitución es el rótulo de valor cero para el primer "NamedType" de la "RootAlternativeTypeList", uno para el segundo, y así sucesivamente, procediendo por números de rótulo crecientes;

d) el "ClassNumber" en el "TaggedType" de sustitución del primer "NamedType" de la "ExtensionAdditionAlternativesList" es superior en uno al mayor "ClassNumber" de la "RootAlternativeTypeList", teniendo el siguiente "NamedType" de la "ExtensionAdditionAlternativesList" un "ClassNumber" superior en uno al primero, y así sucesivamente, procediendo por números de rótulo crecientes;

e) el "Type" en el "TaggedType" de sustitución es el "Type" que se sustituye.

NOTA 1 – Las reglas que rigen la especificación de rotulación implícita o rotulación explícita para los "TaggedType" de sustitución se especifican en 30.6. La rotulación automática es siempre rotulación implícita, a menos que el "Type" sea una notación de tipo elegido sin rotulación o de tipo abierto sin rotulación, o una "DummyReference" sin rotulación (véase la Rec. UIT-T X.683 | ISO/CEI 8824-4, 8.3), en cuyo caso es la rotulación explícita.

NOTA 2 – Una vez que se ha aplicado rotulación automática, los rótulos de los componentes están completamente determinados, y no se modifican ni siquiera cuando el tipo elegido está referenciado en la definición de una alternativa dentro de otras "AlternativeTypeLists", a las que se aplica transformación de rotulación automática. Así pues, en el caso siguiente:

```
T ::= CHOICE { a Ta, b Tb, c Tc }
E ::= CHOICE {f1 E1, f2 T, f3 E3}
```

la rotulación automática aplicada a los componentes de **E** nunca afecta a los rótulos atribuidos a los componentes **a**, **b** y **c** de **T**, cualquiera que sea el entorno de rotulación de **T**. Si **T** se define en un entorno de rotulación automática y **E** no está en un entorno de rotulación automática, se sigue aplicando rotulación automática a los componentes **a**, **b** y **c** de **T**.

NOTA 3 – La subtipificación no afecta a la rotulación automática.

NOTA 4 – En presencia de rotulación automática, la inserción de nuevas alternativas en cualquier ubicación que no sea el punto de inserción de extensión (véase 3.6.29), puede dar lugar a cambios en otras alternativas debido al efecto lateral de la modificación de los rótulos, causando así problemas de interfuncionamiento con una versión más antigua de la especificación.

28.6 "VersionNumber" se define en 24.1, y las restricciones sobre la utilización consecuente de "VersionNumber" a través de un módulo, especificadas en 24.15, se aplicarán a la utilización de "number" dentro de esta producción.

28.7 El rótulo de cada nuevo "NamedType" añadido a la "ExtensionAdditionAlternativesList" será canónicamente mayor (véase 8.6) que los de las otras alternativas de la "ExtensionAdditionAlternativesList" y será el último "NamedType" en la "ExtensionAdditionAlternativesList".

28.8 El tipo elección contiene valores cuyos rótulos no son todos iguales. (El rótulo depende de la alternativa que aportó el valor del tipo elección.)

28.9 Cuando este tipo no tiene marcador de extensión y se emplea en un lugar en que la presente Recomendación | Norma Internacional exige la utilización de tipos con rótulos distintos (véase 28.3) deben tenerse en cuenta en dicha exigencia todos los posibles rótulos de valores del tipo elección. Los ejemplos siguientes, en los que se supone que "TagDefault" no es **AUTOMATIC TAGS**, ilustran ese requisito.

EJEMPLOS

1
```
A ::= CHOICE {
b     B,
c     NULL}

B ::= CHOICE {
d     [0] NULL,
e     [1] NULL}
```

2
```
A ::= CHOICE {
b     B,
c     C}

B ::= CHOICE {
d     [0] NULL,
e     [1] NULL}

C ::= CHOICE {
f     [2] NULL,
g     [3] NULL}
```

3 (Incorrect)
```
      A ::= CHOICE {
b     B,
c     C}

      B ::= CHOICE {
d     [0] NULL,
e     [1] NULL}

      C ::= CHOICE {
f     [0] NULL,
g     [1] NULL}
```

Los ejemplos 1 y 2 presentan una utilización correcta de la notación. La utilización presentada en el ejemplo 3 es incorrecta sin rotulación automática porque los rótulos de los tipos **d** y **f**, y **e** y **g**, son idénticos.

28.10 Los "Identifier" (identificadores) de todos los "NamedType" de "AlternativeTypeLists" deberán ser diferentes de los de otros "NamedType" de esa lista.

28.11 La notación para definir el valor de un tipo elección deberá ser "ChoiceValue", o cuando se utiliza como un "XMLValue", "XMLChoiceValue". Estas producciones son:

ChoiceValue ::= identifier ":" Value

XMLChoiceValue ::= "<" & identifier ">" XMLValue "</" & identifier ">"

28.12 "Value" o "XMLValue" deberá ser una notación para un valor del tipo de "AlternativeTypeLists" que esté nombrado por el "identifier".

29 Notación para tipos selección

29.1 La notación para definir un tipo selección (selection) (véase 3.6.59) será "SelectionType":

SelectionType ::= identifier "<" Type

donde "Type" denota un tipo choice (elección), e "identifier" es el de algún "NamedType" que aparece en "AlternativeTypeLists" de la definición de ese tipo elección.

29.2 Cuando "Type" denota un tipo limitado, la selección se realiza en el tipo progenitor ignorando cualquier limitación de subtipo en el tipo progenitor.

29.3 Cuando el "SelectionType" se utiliza como un "NamedType", el "identifier" del "NamedType" y el "identifier" del "SelectionType" estarán presentes.

29.4 Cuando el "SelectionType" se utiliza como un "Type", se retiene el "identifier" y el tipo denotado es el de la alternativa seleccionada.

29.5 La notación para un valor de un tipo selection será la notación para un valor del tipo referenciado por el "SelectionType".

30 Notación para tipos rotulados

Un tipo rotulado (véase 3.6.70) es un tipo nuevo que es isomorfo con un tipo antiguo, pero que tiene un rótulo diferente. El tipo rotulado se utiliza principalmente donde esta Recomendación | Norma Internacional requiera el empleo de tipos con rótulos distintos (véanse 24.5 a 24.6, 26.3 y 28.3). Mediante el empleo de un "TagDefault" de **AUTOMATIC TAGS** en un módulo puede conseguirse esto sin que aparezca explícitamente en ese módulo una notación de tipo rotulado.

NOTA – Cuando un protocolo determina que pueden transmitirse valores de varios tipos de datos en cualquier momento, pueden necesitarse rótulos distintos para hacer posible que el receptor decodifique correctamente el valor.

30.1 La notación para un tipo rotulado será "TaggedType":

```
TaggedType ::=
        Tag Type
    |   Tag IMPLICIT Type
    |   Tag EXPLICIT Type

Tag ::= "[" Class ClassNumber "]"

ClassNumber ::=
        number
    |   DefinedValue

Class ::=
        UNIVERSAL
    |   APPLICATION
    |   PRIVATE
    |   empty
```

30.2 La "valuereference" de "DefinedValue" deberá ser de tipo entero y se le asignará un valor no negativo.

30.3 El nuevo tipo es isomorfo con el tipo antiguo, pero tiene un rótulo con la clase "Class" y el número "ClassNumber", a menos que la "Class" sea "empty", en cuyo caso el rótulo es de clase específica del contexto y el número es "ClassNumber".

30.4 La "Class" no será **UNIVERSAL** salvo para los tipos definidos en esta Recomendación | Norma Internacional.

NOTA 1 – El uso de rótulos de la clase universal se acuerda cada cierto tiempo entre el UIT-T y la ISO.

NOTA 2 – La subcláusula E.2.12 contiene directrices y sugerencias sobre la utilización estilística de las clases de rótulos.

30.5 Toda aplicación de rótulos es rotulación implícita o rotulación explícita. La rotulación implícita indica, para aquellas reglas de codificación que proporcionan la opción, que la identificación explícita del rótulo del "Type" del "TaggedType" no se necesita durante la transferencia.

NOTA – Puede ser útil retener el rótulo antiguo cuando sea de clase universal y, por lo tanto, identifique sin ambigüedad el tipo antiguo sin conocer la definición ASN.1 del tipo nuevo. La transferencia mínima de octetos, sin embargo, se consigue normalmente con el uso de **IMPLICIT**. En la Rec. UIT-T X.690 | ISO/CEI 8825-1 se da un ejemplo de una codificación utilizando **IMPLICIT**.

30.6 La construcción de la rotulación especifica rotulación explícita si se cumple cualquiera de los siguientes requisitos:

a) se utiliza la alternativa "Tag **EXPLICIT** Type";

b) se utiliza la alternativa "Tag Type" y el valor de "TagDefault" para el módulo es **EXPLICIT TAGS** o está vacío;

c) se utiliza la alternativa "Tag Type" y el valor de "TagDefault" para el módulo es **IMPLICIT TAGS** o **AUTOMATIC TAGS**, pero el tipo definido por "Type" es un tipo choice (elección) sin rotulación, un tipo open (abierto) sin rotulación o una "DummyReference" sin rotulación (véase la Rec. UIT-T X.683 | ISO/CEI 8824-4, 8.3).

En todos los demás casos, la construcción de la rotulación especifica rotulación implícita.

30.7 Si la "Class" es "empty", no hay otras restricciones a la utilización del "Tag" que las implicadas por los requisitos de rótulos distintos de 24.5 a 24.6, 26.3 y 28.3.

30.8 La alternativa **IMPLICIT** no deberá utilizarse si el tipo definido por "Type" es un tipo choice sin rotulación, un tipo open sin rotulación o una "DummyReference" sin rotulación (véase la Rec. UIT-T X.683 | ISO/CEI 8824-4, 8.3).

30.9 La notación para un valor de un "TaggedType" será "TaggedValue", o cuando se utiliza como un "XMLValue", "XMLTaggedValue". Estas producciones son:

TaggedValue ::= Value

XMLTaggedValue ::= XMLValue

donde "Value" o "XMLValue" es una notación para un valor del "Type" en el "TaggedType".

NOTA – El "Tag" no aparece en esta notación.

31 Notación para el tipo identificador de objeto

31.1 El tipo identificador de objeto (object identifier) (véase 3.6.48) se referenciará por la notación "ObjectIdentifierType":

```
ObjectIdentifierType ::=
        OBJECT IDENTIFIER
```

31.2 Este tipo tiene un rótulo que es de clase universal, número 6.

31.3 La notación de valor para un identificador de objeto será "ObjectIdentifierValue", o cuando se utiliza como un "XMLValue", "XMLObjectIdentifierValue". Estas producciones son:

```
ObjectIdentifierValue ::=
        "{" ObjIdComponentsList "}"
    |   "{" DefinedValue ObjIdComponentsList "}"

ObjIdComponentsList ::=
        ObjIdComponents
    |   ObjIdComponents ObjIdComponentsList

ObjIdComponents ::=
        NameForm
    |   NumberForm
    |   NameAndNumberForm
    |   DefinedValue

NameForm ::= identifier

NumberForm ::= number | DefinedValue

NameAndNumberForm ::=
        identifier "(" NumberForm ")"

XMLObjectIdentifierValue ::=
        XMLObjIdComponentList

XMLObjIdComponentList ::=
        XMLObjIdComponent
    |   XMLObjIdComponent & "." & XMLObjIdComponentList

XMLObjIdComponent ::=
        NameForm
    |   XMLNumberForm
    |   XMLNameAndNumberForm

XMLNumberForm ::= number

XMLNameAndNumberForm ::=
        identifier & "(" & XMLNumberForm & ")"
```

31.4 La "valuereference" en "DefinedValue" de "NumberForm" será de tipo entero (integer), y se le asignará un valor no negativo.

31.5 La "valuereference" en "DefinedValue" de "ObjectIdentifierValue" será de tipo identificador de objeto.

31.6 El "DefinedValue" de "ObjIdComponents" será de tipo identificador de objeto relativo, e identificará un conjunto ordenado de arcos de un nodo inicial en el árbol de identificador de objeto a algún nodo posterior en el mismo árbol. El nodo inicial viene identificado por los "ObjIdComponents" anteriores, y posteriormente los "ObjIdComponents" (si los hubiere) identifican arcos del nodo posterior. Es preciso que el nodo de inicio no sea ni la raíz ni el nodo inmediatamente debajo de la raíz.

NOTA – Un valor de identificador de objeto relativo tiene que estar asociado con un valor de identificador de objeto específico para identificar inequívocamente un objeto. Se requieren valores de identificador de objeto (véase 31.10) que tengan al menos dos componentes. Esta es la razón por la cual hay una restricción relativa al nodo inicial.

31.7 La "NameForm" sólo se utilizará para los componentes de identificador de objeto cuyo valor numérico e identificador están especificados en la Rec. UIT-T X.660 | ISO/CEI 9834-1, anexos A a C (véase asimismo el anexo D de esta Recomendación | Norma Internacional y será uno de los identificadores especificados en esos mismos anexos. Cuando se especifiquen identificadores sinónimos en la Rec. UIT-T X.660 | ISO/CEI 9834-1, podrá utilizarse cualquier sinónimo con la misma semántica. Cuando el mismo nombre sea un identificador especificado en la Rec. UIT-T X.660 | ISO/CEI 9834-1 y una referencia de valor ASN.1 dentro del módulo que contiene la "NameForm", el nombre dentro del valor del identificador de objeto se tratará como un identificador de la Rec. UIT-T X.660 | ISO/CEI 9834-1.

31.8 El "number" en la "NumberForm" y en la "XMLNumberForm" será el valor numérico asignado al componente identificador de objeto.

31.9 Se especificará el "identifier" en el "NameAndNumberForm" y "XMLNameAndNumberForm" cuando se asigne un valor numérico al componente identificador de objeto.

NOTA – Las autoridades que atribuyen valores numéricos a componentes de identificador de objeto figuran en la Rec. UIT-T X.660 | ISO/CEI 9834-1.

31.10 La semántica asociada con un valor de identificador de objeto se especifica en la Rec. UIT-T X.660 | ISO/CEI 9834-1.

NOTA – La Rec. UIT-T X.660 | ISO/CEI 9834-1 exige que todo valor de identificador de objeto contenga al menos dos arcos.

31.11 La parte significativa del componente identificador de objeto es la "NameForm", "NumberForm" o "XMLNumberForm" a la cual se reduce y que proporciona el valor numérico para el componente identificador de objeto. Salvo para los arcos especificados en la Rec. UIT-T X.660 | ISO/CEI 9834-1, anexos A a C (véase asimismo el anexo D de esta Recomendación | Norma Internacional) el valor numérico del componente identificador de objeto siempre está presente en una instancia de notación de valor de identificador de objeto.

31.12 Cuando el "ObjectIdentifierValue" incluye un "DefinedValue" para un valor de identificador de objeto, la lista de componentes identificador de objeto a la que se refiere es prefijada a los componentes explícitamente presentes en el valor.

NOTA – En la Rec. UIT-T X.660 | ISO/CEI 9834-1 se recomienda que, siempre que se asigne un valor de identificador de objeto para identificar un objeto, también se asigne un valor de descriptor de objeto.

EJEMPLOS

Con identificadores asignados según se especifica en la Rec. UIT-T X.660 | ISO/CEI 9834-1, los valores:

```
{ iso standard 8571 pci (1) }
```

y

```
{ 1 0 8571 1 }
```

identificarían cada uno un objeto, **pci**, definido en ISO 8571, como lo haría

```
iso.standard.8571.pci(1)
```

y

```
1.0.8571.1
```

en un "XMLObjectIdentifierValue".

Con la siguiente definición adicional:

```
ftam OBJECT IDENTIFIER ::= { iso standard 8571 }
```

el valor siguiente es equivalente a los anteriores:

```
{ ftam pci(1) }
```

32 Notación para el tipo identificador de objeto relativo

32.1 El tipo identificador de objeto relativo (véase 3.6.57) se referenciará por la notación "RelativeOIDType":

RelativeOIDType ::= `RELATIVE-OID`

32.2 Este tipo tiene un rótulo que es de clase universal, número 13.

32.3 La notación de valor para un identificador de objeto relativo será "RelativeOIDValue", o cuando se utilice como "XMLValue", "XMLRelativeOIDValue". Estas producciones son:

RelativeOIDValue ::=
"{" RelativeOIDComponentsList "}"

RelativeOIDComponentsList ::=
RelativeOIDComponents
| **RelativeOIDComponents RelativeOIDComponentsList**

RelativeOIDComponents ::=
NumberForm
| **NameAndNumberForm**
| **DefinedValue**

XMLRelativeOIDValue ::=
XMLRelativeOIDComponentList

XMLRelativeOIDComponentList ::=
XMLRelativeOIDComponent
| **XMLRelativeOIDComponent & "." & XMLRelativeOIDComponentList**

XMLRelativeOIDComponent ::=
XMLNumberForm
| **XMLNameAndNumberForm**

32.4 Las producciones "NumberForm", "NameAndNumberForm", "XMLNumberForm", "XMLNameAndNumberForm" y su semántica se definen en las subcláusulas 31.3 a 31.11.

32.5 El "DefinedValue" de "RelativeOIDComponents" será de tipo identificador de objeto relativo, e identificará un conjunto ordenado de arcos desde algún nodo inicial en el árbol de identificador de objeto a algún nodo posterior en el mismo árbol. El nodo inicial viene identificado por los "RelativeOIDComponents" anteriores (si los hubiere), y los "RelativeOIDComponents" posteriores (si los hubiere), identifican arcos a partir de los nodos posteriores.

32.6 El primer "RelativeOIDComponents" o "XMLRelativeOIDComponents" identifica uno o más arcos desde algún nodo inicial del árbol identificador de objeto a algún nodo posterior en el mismo árbol. El punto inicial puede ser definido por comentarios asociados con la definición de tipo. Si no hay definición del nodo inicial en los comentarios asociados con la definición de tipo, ésta deberá ser transmitida como un valor de identificador de objeto en un ejemplar de comunicación (véase E.2.19). El nodo inicial no debe ser ni la raíz, ni un nodo inmediatamente por debajo de la raíz.

NOTA – Un valor de identificador de objeto relativo tiene que estar asociado con un valor de identificador de objeto específico para identificar inequívocamente un objeto. Se requieren valores de identificador de objeto (véase 31.10) que tengan al menos dos componentes. Ésta es la razón por la cual hay una restricción relativa al nodo inicial.

EJEMPLO

Con las siguientes definiciones:

```
thisUniversity  OBJECT IDENTIFIER       ::=
            {iso member-body country(29) universities(56) thisuni(32)}

firstgroup   RELATIVE-OID   ::=  {science-fac(4) maths-dept(3)}
```

o en notación de valor XML:

```
thisUniversity ::= <OBJECT_IDENTIFIER>1.2.29.56.32</OBJECT_IDENTIFIER>

firstgroup ::= <RELATIVE_OID>4.3</RELATIVE_OID>
```

el identificador de objeto relativo:

```
relOID   RELATIVE-OID  ::=  {firstgroup room(4) socket(6)}
```

o en notación de valor XML:

```
relOID ::= <RELATIVE_OID>4.3.4.6</RELATIVE_OID>
```

puede utilizarse en lugar del valor de **OBJECT IDENTIFIER {1 2 29 56 32 4 3 4 6}** si la raíz existente (conocida o transmitida por la aplicación) es **thisUniversity**.

33 Notación para el tipo pdv incrustado

33.1 El tipo pdv incrustado (embedded-pdv) (véase 3.6.21) se referenciará por la notación "EmbeddedPDVType":

EmbeddedPDVType ::= EMBEDDED PDV

NOTA – El término "Embedded PDV" significa un valor abstracto de una sintaxis abstracta posiblemente diferente (esencialmente, el valor y codificación de un mensaje definido (e identificado) en un protocolo separado) que está incorporado en un mensaje. Históricamente significa "Embedded Presentation Data Value" a partir de su utilización en la capa de presentación OSI, pero esta ampliación no se utiliza actualmente, y debería interpretarse como "embedded value".

33.2 Este tipo tiene un rótulo que es de clase universal, número 11.

33.3 El tipo está constituido por valores que representan:

a) la codificación de un valor de dato único que puede ser, aunque no necesariamente, el valor de un tipo ASN.1; y

b) la identificación (de manera separada o conjunta) de:

1) una sintaxis abstracta; y

2) la sintaxis de transferencia.

NOTA 1 – El valor de dato puede ser el valor de un tipo ASN.1 o puede ser, por ejemplo, la codificación de una imagen fija o una imagen en movimiento. La identificación consta de uno o dos identificadores de objeto o hace referencia (en un entorno OSI) a un identificador de contexto de presentación OSI que especifica las sintaxis abstracta y de transferencia.

NOTA 2 – La identificación de la sintaxis abstracta y/o la codificación también puede estar determinada por la del diseñador de la aplicación como un valor fijo, en cuyo caso no se codifica en un ejemplar de comunicación.

33.4 El tipo pdv incrustado tiene un tipo asociado. Este tipo se utiliza para soportar las notaciones de valor y subtipo del tipo pdv incrustado.

33.5 El tipo asociado para la definición de valor y la subtipificación, suponiendo un entorno de rotulación automática, es (con comentarios normativos):

```
SEQUENCE {
      identification                          CHOICE {
            syntaxes                             SEQUENCE {
                  abstract                          OBJECT IDENTIFIER,
                  transfer                          OBJECT IDENTIFIER }
            -- Abstract and transfer syntax object identifiers --,

            syntax                               OBJECT IDENTIFIER
            -- A single object identifier for identification of the abstract
            -- and transfer syntaxes --,

            presentation-context-id              INTEGER
            -- (Applicable only to OSI environments)
            -- The negotiated OSI presentation context identifies the
            -- abstract and transfer syntaxes --,

            context-negotiation                  SEQUENCE {
                  presentation-context-id           INTEGER,
                  transfer-syntax                   OBJECT IDENTIFIER }
            -- (Applicable only to OSI environments)
            -- Context-negotiation in progress, presentation-context-id
            -- identifies only the abstract syntax
            -- so the transfer syntax shall be specified --,

            transfer-syntax                      OBJECT IDENTIFIER
            -- The type of the value (for example, specification that it is
            -- the value of an ASN.1 type)
            -- is fixed by the application designer (and hence known to both
            -- sender and receiver). This
            -- case is provided primarily to support
            -- selective-field-encryption (or other encoding
            -- transformations) of an ASN.1 type --,
```

```
            fixed                                   NULL
            -- The data value is the value of a fixed ASN.1 type (and hence
            -- known to both sender
            -- and receiver) -- },

        data-value-descriptor                   ObjectDescriptor  OPTIONAL
        -- This provides human-readable identification of the class of the
        -- value --,
        data-value                              OCTET STRING }

    ( WITH COMPONENTS {
            ... ,
            data-value-descriptor  ABSENT } )
```

NOTA – El tipo pdv incrustado no permite la inclusión de un valor **data-value-descriptor**. Sin embargo, la definición del tipo asociado que aquí se da sustenta las comunalidades que existen entre el tipo pdv incrustado, el tipo externo y el tipo cadena de caracteres no restringida.

33.6 La alternativa **presentation-context-id** sólo puede aplicarse en un entorno OSI, cuando el valor entero sea un identificador de contexto de presentación OSI del conjunto de contextos definido por OSI. Esta alternativa no deberá ser utilizada durante la negociación del contexto OSI.

33.7 La alternativa **context-negotiation** sólo puede aplicarse en un entorno OSI, y deberá utilizarse solamente durante la negociación del contexto OSI. El valor entero será un identificador de contexto de presentación OSI propuesto para su adición al conjunto de contextos definido por OSI. El identificador de objeto **transfer-syntax** identificará una sintaxis de transferencia propuesta para el contexto de presentación OSI que habrá de utilizarse para codificar el valor.

33.8 La notación para un valor del tipo embedded-pdv será la notación del valor del tipo asociado definido en 33.5, donde el valor del componente **data-value** del tipo **OCTET STRING** representa una codificación que utiliza la sintaxis de transferencia especificada en identification.

EmbeddedPdvValue ::= SequenceValue *-- value of associated type defined in 33.5*

XMLEmbeddedPDVValue ::= XMLSequenceValue *-- value of associated type defined in 33.5*

EJEMPLO – Si ha de implementarse una sola opción, tal como la utilización de **syntaxes**, esto puede hacerse escribiendo:

```
EMBEDDED PDV (WITH COMPONENTS {
                ... ,
                identification  (WITH COMPONENTS {
                      syntaxes  PRESENT } ) } )
```

34 Notación para el tipo externo

34.1 El tipo externo (external) (véase 3.6.37) se referenciará por la notación "ExternalType":

ExternalType ::= EXTERNAL

34.2 Este tipo tiene un rótulo que es de clase universal, número 8.

34.3 El tipo está constituido por valores que representan:

a) la codificación de un valor de dato único que puede ser, aunque no necesariamente, el valor de un tipo ASN.1; y

b) la identificación de:

   1) una sintaxis abstracta; y

   2) la sintaxis de transferencia; y

c) (opcionalmente) un descriptor de objeto que proporciona una descripción legible por los seres humanos de la categoría del valor de dato. El descriptor de objeto opcional no estará presente, a menos que lo permitan de manera explícita comentarios asociados con la utilización de la notación "ExternalType".

NOTA – La nota 1 de 33.3 también se aplica al tipo external.

34.4 El tipo externo tiene un tipo asociado. Este tipo se utiliza para dar precisión a la definición de los valores abstractos del tipo externo y se utiliza también para soportar las notaciones de valor y subtipo del tipo externo.

NOTA – Las reglas de codificación pueden definir un tipo diferente, utilizado para derivar codificaciones, o pueden especificar codificaciones sin referencia a ningún tipo asociado. Por ejemplo, la codificación en BER utiliza un tipo secuencia diferente, por motivos históricos.

34.5 El tipo asociado para la definición de valor y la subtipificación, suponiendo un entorno de rotulación automática, es (con comentarios normativos):

```
SEQUENCE {
      identification                          CHOICE {
            syntaxes                             SEQUENCE {
                  abstract                          OBJECT IDENTIFIER,
                  transfer                          OBJECT IDENTIFIER }
            -- Abstract and transfer syntax object identifiers --,

            syntax                                OBJECT IDENTIFIER
            -- A single object identifier for identification of the abstract
            -- and transfer syntaxes --,

            presentation-context-id               INTEGER
            -- (Applicable only to OSI environments)
            -- The negotiated OSI presentation context identifies the
            -- abstract and transfer syntaxes --,

            context-negotiation                   SEQUENCE {
                  presentation-context-id            INTEGER,
                  transfer-syntax                    OBJECT IDENTIFIER }
            -- (Applicable only to OSI environments)
            -- Context-negotiation in progress, presentation-context-id
            -- identifies only the abstract syntax
            -- so the transfer syntax shall be specified --,

            transfer-syntax                       OBJECT IDENTIFIER
            -- The type of the value (for example, specification that it is
            -- the value of an ASN.1 type)
            -- is fixed by the application designer (and hence known to both
            -- sender and receiver). This
            -- case is provided primarily to support
            -- selective-field-encryption (or other encoding
            -- transformations) of an ASN.1 type --,

            fixed                                 NULL
            -- The data value is the value of a fixed ASN.1 type (and hence
            -- known to both sender
            -- and receiver) -- },

data-value-descriptor                   ObjectDescriptor  OPTIONAL
            -- This provides human-readable identification of the class of
            -- the value --,

data-value                              OCTET STRING }
      ( WITH COMPONENTS {
            ... ,
            identification  (WITH COMPONENTS {
                  ... ,
                  syntaxes                                  ABSENT,
                  transfer-syntax                           ABSENT,
                  fixed                           ABSENT } ) } )
```

NOTA – Por motivos históricos, el tipo externo no permite las alternativas **syntaxes**, **transfer-syntax** o **fixed** de **identification**. Los diseñadores de aplicaciones que exijan estas opciones deberán utilizar el tipo pdv-incrustado. La definición del tipo asociado que aquí se proporciona sustenta las comunalidades existentes entre el tipo externo, el tipo cadena de caracteres no restringida y el tipo pdv-incrustado.

34.6 El texto de 33.6 y 33.7 también se aplica al tipo external.

34.7 La notación para un valor de tipo externo (external) deberá ser la notación de valor del tipo asociado definido en 34.5, donde el valor del componente **data-value** del tipo **OCTET STRING** representa una codificación que utiliza la sintaxis de transferencia especificada en **identification**.

ExternalValue ::= SequenceValue *-- value of associated type defined in 34.5*

XMLExternalValue ::= XMLSequenceValue *-- value of associated type defined in 34.5*

NOTA – Por razones históricas, las reglas de codificación pueden transferir valores incrustados en **EXTERNAL**, cuyas codificaciones no sean múltiples exactos de ocho bits. Tales valores no pueden ser representados en la notación de valor utilizando el mencionado tipo asociado.

35 Tipos cadena de caracteres

Estos tipos están constituidos por cadenas de caracteres procedentes de algún repertorio de caracteres especificado. Es normal definir un repertorio de caracteres y su codificación mediante el uso de células de una o más tablas, correspondiendo cada célula a un carácter del repertorio. A cada célula se le asigna también, normalmente, un símbolo gráfico y un nombre de carácter, si bien en algunos repertorios se dejan células vacías o tienen nombres pero no formas (ejemplos de células con nombre pero sin forma son los caracteres de control, tales como el EOF de ISO/CEI 646 y los caracteres de espaciado tales como THIN-SPACE y EN-SPACE de ISO/CEI 10646-1).

Por lo general, la información asociada con una célula denota un carácter abstracto distinto del repertorio incluso si esa información es nula (no hay un símbolo gráfico o nombre de carácter asignado a esa célula).

La notación de valor básico ASN.1 para los tipos cadenas de caracteres tiene tres variantes (que pueden combinarse), especificadas formalmente a continuación:

a) Una representación de los caracteres de la cadena utilizando símbolos gráficos asignados, con la posible inclusión de caracteres con espaciado; ésta es la notación "cstring".

   NOTA 1 – Una representación como esa puede resultar equívoca en una representación impresa cuando se utilice el mismo símbolo gráfico para más de un carácter del repertorio.

   NOTA 2 – Una representación como esa puede resultar equívoca en una representación impresa cuando estén presentes caracteres con espaciado y anchuras diferentes o la especificación esté impresa con un tipo de carácter de espaciado proporcional.

b) Un listado de los caracteres del valor cadena de caracteres dando una serie de referencias de valores ASN.1 que han sido asignados al carácter; un conjunto de tales referencias de valores se define en el módulo **`ASN1-CHARACTER-MODULE`** en la cláusula 38 para el repertorio de ISO/CEI 10646-1 y el de caracteres **`IA5String`**; esta forma no está disponible para otros repertorios de caracteres a menos que el usuario efectúe asignaciones a esas referencias de valores utilizando la notación de valor descrita en el apartado anterior a) o en el c) siguiente.

c) Un listado de caracteres del valor cadena de caracteres identificando cada uno de los caracteres abstractos por la posición de su célula en la tabla o tablas repertorio de caracteres; esta forma está disponible solamente para **`IA5String`**, **`UniversalString`**, **`UTF8String`** y **`BMPString`**.

La notación de valor XML ASN.1 de los tipos cadena de caracteres utiliza la notación "xmlcstring", que incluye la capacidad para emplear secuencias de escape para determinados caracteres especiales, y para la especificación de caracteres mediante representación decimal o hexadecimal (véase 11.15).

36 Notación para tipos cadena de caracteres

36.1 La notación para referenciar un tipo cadena de caracteres (véase 3.6.11) será:

```
CharacterStringType ::=
        RestrictedCharacterStringType
    |   UnrestrictedCharacterStringType
```

"RestrictedCharacterStringType" es la notación para un tipo cadena de caracteres restringida y se define en la cláusula 37. "UnrestrictedCharacterStringType" es la notación para un tipo cadena de caracteres no restringida y se define en 40.1.

36.2 El rótulo de cada tipo cadena de caracteres restringida se especifica en 37.1. El rótulo del tipo cadena de caracteres no restringida se especifica en 40.2.

36.3 La notación para un valor cadena de caracteres será:

```
CharacterStringValue ::=
        RestrictedCharacterStringValue
    |   UnrestrictedCharacterStringValue

XMLCharacterStringValue ::=
        XMLRestrictedCharacterStringValue
    |   XMLUnrestrictedCharacterStringValue
```

"RestrictedCharacterStringValue" y "XMLRestrictedCharacterStringValue" se definen en 37.8 y 37.9 respectivamente. "UnrestrictedCharacterStringValue" y "XMLUnrestrictedCharacterStringValue" son notaciones para un valor cadena de caracteres no restringida y se definen en 40.7.

37 Definición de tipos cadena de caracteres restringida

En esta cláusula se definen tipos cuyos valores están limitados a secuencias de cero, uno o más caracteres de un repertorio especificado de caracteres. La notación para referenciar un tipo cadena de caracteres restringida será "RestrictedCharacterStringType":

```
RestrictedCharacterStringType ::=
        BMPString
    |   GeneralString
    |   GraphicString
    |   IA5String
    |   ISO646String
    |   NumericString
    |   PrintableString
    |   TeletexString
    |   T61String
    |   UniversalString
    |   UTF8String
    |   VideotexString
    |   VisibleString
```

Cada una de las alternativas de "RestrictedCharacterStringType" se define especificando:

a) el rótulo asignado al tipo; y

b) un nombre (por ejemplo, **`NumericString`**) por el cual puede referenciarse el tipo; y

c) los caracteres de la colección de caracteres utilizada para definir el tipo, por referencia a una tabla que recoge los caracteres gráficos o por referencia a un número de registro en ISO International Register of Coded Character Sets (Registro internacional de juegos de caracteres codificados de la ISO) (véase *ISO International Register of Coded Character Sets to be used with Escape Sequences*) o por referencia a la ISO/CEI 10646-1.

37.1 El cuadro 6 da el nombre por el cual se referencia cada tipo cadena de caracteres restringida, el número del rótulo de la clase universal asignada al tipo, el número de registro o el cuadro que proporcionan la definición o la cláusula del texto que proporciona la definición y, donde sea necesario, la identificación de una nota relativa a la entrada en el cuadro. Si la notación contiene la definición de un nombre sinónimo, éste aparecerá entre paréntesis.

Cuadro 6 – Lista de tipos cadena de caracteres restringida

Nombre para referenciar el tipo	Número de clase universal	Número de registro para la definición[a], número de cuadro o cláusula de la Rec. UIT-T X.680 | ISO/CEI 8824-1	Notas
`UTF8String`	12	Subcláusula 37.16	
`NumericString`	18	Cuadro 7	(Nota 1)
`PrintableString`	19	Cuadro 8	(Nota 1)
`TeletexString` (**`T61String`**)	20	6, 87, 102, 103, 106, 107, 126, 144, 150, 153, 156, 164, 165, 168 + SPACE + DELETE	(Nota 2)
`VideotexString`	21	1, 13, 72, 73, 87, 89, 102, 108, 126, 128, 129, 144, 150, 153, 164, 165, 168 + SPACE + DELETE	(Nota 3)
`IA5String`	22	1, 6 + SPACE + DELETE	
`GraphicString`	25	Todos los juegos G + SPACE	
`VisibleString` (**`ISO646String`**)	26	6 + SPACE	
`GeneralString`	27	Todos los juegos G y C + SPACE + DELETE	
`UniversalString`	28	Véase 37.6	
`BMPString`	30	Véase 37.15	

[a] Los números de registro que proporcionan la definición se dan en ISO International Register of Coded Character Sets to be used with Escape Sequences.

NOTA 1 – El estilo tipográfico, tamaño, color, intensidad u otras características de visualización no son significativos.

NOTA 2 – Pueden utilizarse las entradas de registro 6 y 156 en vez de las 102 y 103.

NOTA 3 – Las entradas correspondientes a estos números de registro dan la funcionalidad de la Rec. T.100 del CCITT y Rec. UIT-T T.101.

37.2 El cuadro 7 indica los caracteres que pueden aparecer en el tipo **NumericString** y en la sintaxis abstracta de caracteres **NumericString**.

Cuadro 7 – NumericString

Nombre	Símbolo gráfico
Dígitos	**0, 1, ... 9**
Espacio	(espacio)

37.3 Los siguientes valores de identificador de objeto y de descriptor de objeto se asignan para identificar y describir la sintaxis abstracta de caracteres **NumericString**:

```
{ joint-iso-itu-t asn1(1) specification(0) characterStrings(1) numericString(0) }
```

y

```
"NumericString character abstract syntax"
```

NOTA 1 – Este valor de identificador de objeto puede utilizarse en valores de **CHARACTER STRING** y en otros casos en los que es necesario llevar la identificación del tipo cadena de caracteres separada del valor.

NOTA 2 – Un valor de una sintaxis abstracta de caracteres **NumericString** puede ser codificado por:

a) Una de las reglas dadas en ISO/CEI 10646-1 para la codificación de los caracteres abstractos. En este caso, la sintaxis de transferencia de caracteres es identificada por el identificador de objeto asociado con las reglas de ISO/CEI 10646-1, anexo N.

b) Las reglas de codificación ASN.1 para el tipo **NumericString** incorporado. En este caso, la sintaxis de transferencia de caracteres es identificada por el valor de identificador de objeto **{joint-iso-itu-t asn1(1) basic-encoding(1)}**.

37.4 El cuadro 8 indica los caracteres que pueden aparecer en el tipo **PrintableString** y en la sintaxis abstracta de caracteres **PrintableString**.

Cuadro 8 – PrintableString

Nombre	Símbolo gráfico
Letras mayúsculas	**A**, **B**, ... **Z**
Letras minúsculas	**a**, **b**, ... **z**
Dígitos	**0**, **1**, ... **9**
ESPACIO	(espacio)
APÓSTROFO	**'**
PARÉNTESIS IZQUIERDO	**(**
PARÉNTESIS DERECHO	**)**
SIGNO MÁS	**+**
COMA	**,**
GUIÓN-MENOS	**-**
PUNTO	**.**
BARRA DE FRACCIÓN	**/**
DOS PUNTOS	**:**
SIGNO IGUAL	**=**
SIGNO DE INTERROGACIÓN (FINAL)	**?**

37.5 Los siguientes valores de identificador de objeto y descriptor de objeto se asignan para identificar y describir la sintaxis abstracta de caracteres **PrintableString**:

```
{ joint-iso-itu-t asn1(1) specification(0) characterStrings(1) printableString(1) }
```

y

```
"PrintableString character abstract syntax"
```

NOTA 1 – Este valor de identificador de objeto puede utilizarse en valores de **CHARACTER STRING** y en otros casos en los que es necesario llevar la identificación del tipo cadena de caracteres separada del valor.

NOTA 2 – Un valor de una sintaxis abstracta de caracteres **PrintableString** puede ser codificado por:

a) Una de las reglas dadas en ISO/CEI 10646-1 para la codificación de los caracteres abstractos. En este caso, la sintaxis de transferencia de caracteres es identificada por el identificador de objeto asociado con las reglas de ISO/CEI 10646-1, anexo N.

b) Las reglas de codificación ASN.1 para el tipo **PrintableString** incorporado. En este caso, la sintaxis de transferencia de caracteres es identificada por el identificador de objeto **{ joint-iso-itu-t asn1(1) basic-encoding(1)}**.

37.6 Los caracteres que pueden aparecer en el tipo **UniversalString** son cualesquiera de los autorizados por ISO/CEI 10646-1.

37.7 La utilización de este tipo invoca los requisitos de conformidad especificados en ISO/CEI 10646-1.

NOTA – La cláusula 38 define un módulo ASN.1 que contiene varios subtipos de este tipo para las "Collections of graphics characters for subsets" (Colecciones de caracteres gráficos para subconjuntos) definidas en ISO/CEI 10646-1, anexo A.

37.8 La notación "RestrictedCharacterStringValue" para los tipos cadena de caracteres restringida será "cstring" (véase 11.14), "CharacterStringList", Quadruple" o "Tuple". "Quadruple" sólo es capaz de definir una cadena de caracteres de longitud uno y sólo se puede utilizar en notación de valor para tipos **UniversalString**, **UTF8String** o **BMPString**. "Tuple" sólo puede definir una cadena de caracteres de longitud uno y sólo se puede utilizar en notación de valor para tipos **IA5String**.

```
RestrictedCharacterStringValue ::=
        cstring
    |   CharacterStringList
    |   Quadruple
    |   Tuple

CharacterStringList ::=  "{" CharSyms "}"

CharSyms ::=
        CharsDefn
    |   CharSyms "," CharsDefn

CharsDefn ::=
        cstring
    |   Quadruple
    |   Tuple
    |   DefinedValue

Quadruple ::= "{" Group "," Plane "," Row "," Cell "}"

Group   ::= number

Plane   ::= number

Row     ::= number

Cell    ::= number

Tuple   ::= "{" TableColumn "," TableRow "}"

TableColumn ::= number

TableRow ::= number
```

NOTA 1 – La notación "cstring" sólo puede utilizarse inequívocamente en un medio capaz de visualizar los símbolos gráficos de los caracteres que están presentes en el valor. Por el contrario, si el medio no tiene esa capacidad, la única posibilidad de especificar inequívocamente un valor cadena de caracteres que utiliza esos símbolos gráficos es mediante la notación "CharacterStringList" y solamente si el tipo es **UniversalString**, **UTF8String**, **BMPString** o **IA5String** y se utiliza la alternativa "DefinedValue" de "CharsDefn" (véase 38.1.2).

NOTA 2 – La cláusula 38 define varias "valuereference" que denotan caracteres únicos (cadenas de tamaño 1) de tipo **BMPString** (y, por tanto, **UniversalString** y **UTF8String**) e **IA5String**.

EJEMPLO – Supóngase que se desea especificar un valor de "abcΣdef" para una **UniversalString** (cadena universal), y no siendo representable el carácter "Σ" en el medio disponible, este valor puede expresarse también como:

```
IMPORTS BasicLatin, greekCapitalLetterSigma FROM ASN1-CHARACTER-MODULE
      { joint-iso-itu-t asn1(1) specification(0) modules(0) iso10646(0) };
```

```
MyAlphabet ::= UniversalString (FROM (BasicLatin | greekCapitalLetterSigma))

mystring MyAlphabet ::= { "abc" , greekCapitalLetterSigma , "def" }
```

NOTA 3 – Cuando se especifique el valor de un tipo **UniversalString, UTF8String** o **BMPString**, no deberá emplearse la notación "cstring", a menos que se hayan resuelto las ambigüedades derivadas de los caracteres gráficos diferentes con formas similares.

EJEMPLO – No deberá utilizarse la siguiente notación "cstring" porque los símbolos gráficos "H", "O", "P" y "E" aparecen en los alfabetos LATÍN BÁSICO, CIRÍLICO y GRIEGO BÁSICO por lo que resultan ambiguos.

```
IMPORTS BasicLatin, Cyrillic, BasicGreek FROM ASN1-CHARACTER-MODULE
     { joint-iso-itu-t asn1(1) specification(0) modules(0) iso10646(0) };

MyAlphabet ::= UniversalString (FROM (BasicLatin | Cyrillic | BasicGreek))

mystring MyAlphabet ::= "HOPE"
```

Una definición alternativa inequívoca de **mystring** sería:

```
mystring  MyAlphabet(BasicLatin) ::= "HOPE"
```

Formalmente, **mystring** es una referencia de valor a un valor de un subconjunto de **MyAlphabet**, pero puede utilizarse, mediante las reglas de correspondencia de valor del anexo B, siempre que se necesite una referencia de valor a este valor dentro de **MyAlphabet**.

37.9 La notación "XMLRestrictedCharacterStringValue" es:

XMLRestrictedCharacterStringValue ::= xmlcstring

37.10 Hay algunos caracteres que no pueden ser representados directamente en "xmlcstring". Éstos se representarán utilizando las secuencias de escape que se especifican en 11.15.

NOTA – Si el valor de la cadena de caracteres restringida contiene caracteres que no pertenecen a los de ISO/CEI 10646-1 que se especifican en 11.15.1, éstos no podrán ser representados en "xmlcstring", y esos valores no pueden ser transferidos mediante las reglas de codificación XML (véase la Rec. UIT-T X.693 | ISO/CEI 8825-4).

37.11 El "DefinedValue" de "CharsDefn" deberá ser una referencia a un valor de ese tipo.

37.12 El "number" de las producciones "Plane", "Row" y "Cell" deberá ser inferior a 256 y en la producción "Group" deberá ser inferior a 128.

37.13 El "Group" especifica un grupo en el espacio de codificación del UCS, el "Plane" especifica un plano dentro del grupo, la "Row" especifica una fila dentro del plano y la "Cell" especifica una célula dentro de la fila. El carácter abstracto identificado por esta notación es el carácter abstracto de la célula especificado por los valores de "Group", "Plane", "Row" y "Cell". En todos los casos, el conjunto de caracteres permitidos puede ser restringido por subtipificación.

NOTA – Los diseñadores de aplicaciones deben analizar cuidadosamente las implicaciones de conformidad cuando se utilizan tipos cadena de caracteres abierta, tales como **GeneralString**, **GraphicString** y **UniversalString**, sin la aplicación de constricciones. Se necesita también un texto cuidadosamente elaborado sobre conformidad para tipos cadena de caracteres que, aunque acotados, son extensos, como **TeletexString**.

37.14 El "number" de la producción "TableColumn" deberá estar en la gama de cero a siete y el "number" de la producción "TableRow" deberá estar en la gama de cero a quince. La "TableColumn" especifica una columna y la "TableRow" especifica una fila de una tabla de códigos de caracteres, de conformidad con la figura 1 de ISO/CEI 2022. Esta notación se emplea solamente para **IA5String** cuando la tabla de códigos contiene la entrada de registro 1 en las columnas 0 y 1 y la entrada de registro 6 en las columnas 2 a 7 (véase *ISO International Register of Coded Character Sets to be used with Escape Sequences*).

37.15 **BMPString** es un subtipo de **UniversalString** que tiene su propio rótulo exclusivo y contiene sólo los caracteres en el plano plurilingüe básico (los que corresponden a las primeras células 64K-2, células menos importantes cuya codificación se emplea para tratar caracteres que no pertenecen al plano plurilingüe básico) de ISO/CEI 10646-1. Tiene un tipo asociado definido como:

```
UniversalString (Bmp)
```

donde **Bmp** se define en el módulo **ASN.1 ASN1-CHARACTER-MODULE** (véase la cláusula 38) como el subtipo de **UniversalString** correspondiente al nombre de colección "BMP" definido en ISO/CEI 10646-1, anexo A.

NOTA 1 – Puesto que **BMPString** es un tipo incorporado, no se define en **ASN1-CHARACTER-MODULE**.

NOTA 2 – La finalidad de definir **BMPString** como un tipo incorporado es permitir que las reglas de codificación (tales como BER), que no tienen en cuenta las constricciones, utilicen codificaciones de 16 bits en vez de 32 bits.

NOTA 3 – En la notación de valor todos los valores **BMPString** son valores **UniversalString** y **UTF8String** válidos.

36.16 **`UTF8String`** es sinónimo de **`UniversalString`** a nivel abstracto y se puede utilizar siempre que se utilice **`UniversalString`** (a reserva de que ciertas reglas exijan rótulos distintos) pero tiene un rótulo diferente y un tipo distinto.

NOTA – La codificación de **`UTF8String`** utilizada por BER y PER es diferente de la de **`UniversalString`** y para la mayoría de los textos será menos extensa.

38 Denominación de los caracteres y colecciones definidos en ISO/CEI 10646-1

Esta cláusula especifica un módulo incorporado ASN.1 que contiene la definición de un nombre de referencia de valor para cada carácter de ISO/CEI 10646-1, donde cada nombre referencia un valor **`UniversalString`** de tamaño 1. Este módulo contiene también la definición de un nombre de referencia de tipo para cada colección de caracteres de ISO/CEI 10646-1, donde cada nombre referencia un subconjunto del tipo **`UniversalString`**.

NOTA – Estos valores están disponibles para uso en la notación de valor del tipo **`UniversalString`** y de los tipos derivados de éste. Todas las referencias de valores y tipos definidas en el módulo especificado en 38.1 son exportadas y tienen que ser importadas por cualquier módulo que las utilice.

38.1 Especificación del módulo ASN.1 "ASN1-CHARACTER-MODULE"

Este módulo no se reproduce aquí en su totalidad. En su lugar se especifica el medio para definirlo.

38.1.1 El módulo comienza de esta manera:

```
ASN1-CHARACTER-MODULE { joint-iso-itu-t asn1(1) specification(0) modules(0)
iso10646(0) }
      DEFINITIONS ::= BEGIN
      -- All of the value references and type references defined within this
      -- module are implicitly exported, and are available for import by any
         module.
      -- ISO/IEC 646 control characters:

      nul   IA5String ::= {0, 0}
      soh   IA5String ::= {0, 1}
      stx   IA5String ::= {0, 2}
      etx   IA5String ::= {0, 3}
      eot   IA5String ::= {0, 4}
      enq   IA5String ::= {0, 5}
      ack   IA5String ::= {0, 6}
      bel   IA5String ::= {0, 7}
      bs    IA5String ::= {0, 8}
      ht    IA5String ::= {0, 9}
      lf    IA5String ::= {0,10}
      vt    IA5String ::= {0,11}
      ff    IA5String ::= {0,12}
      cr    IA5String ::= {0,13}
      so    IA5String ::= {0,14}
      si    IA5String ::= {0,15}
      dle   IA5String ::= {1, 0}
      dc1   IA5String ::= {1, 1}
      dc2   IA5String ::= {1, 2}
      dc3   IA5String ::= {1, 3}
      dc4   IA5String ::= {1, 4}
      nak   IA5String ::= {1, 5}
      syn   IA5String ::= {1, 6}
      etb   IA5String ::= {1, 7}
      can   IA5String ::= {1, 8}
      em    IA5String ::= {1, 9}
      sub   IA5String ::= {1,10}
      esc   IA5String ::= {1,11}
      is4   IA5String ::= {1,12}
      is3   IA5String ::= {1,13}
      is2   IA5String ::= {1,14}
      is1   IA5String ::= {1,15}
      del   IA5String ::= {7,15}
```

38.1.2 Para cada entrada de cada lista de nombres de caracteres de los caracteres gráficos (glifos) mostradas en las cláusulas 24 y 25 de ISO/CEI 10646-1, el módulo incluye un enunciado de la forma:

```
<namedcharacter> BMPString ::= <tablecell>
-- represents the character <iso10646name>, see ISO/IEC 10646-1
```

donde:

a) <*iso10646name*> es el nombre de carácter derivado de uno listado en ISO/CEI 10646-1;

b) <*namedcharacter*> es una cadena obtenida aplicando a <*iso10646name*> los procedimientos especificados en 38.2;

c) <*tablecell*> es el glifo de la célula de tabla de ISO/CEI 10646-1 correspondiente a la entrada en la lista.

EJEMPLO

```
latinCapitalLetterA  BMPString  ::=  {0, 0, 0, 65}
-- represents the character LATIN CAPITAL LETTER A, see ISO/IEC 10646-1

greekCapitalLetterSigma BMPString  ::=  {0, 0, 3, 163}
-- represents the character GREEK CAPITAL LETTER SIGMA, see ISO/IEC 10646-1
```

38.1.3 Para cada nombre de una colección de caracteres gráficos especificados en ISO/CEI 10646-1, anexo A, se incluye un enunciado en el módulo de la forma:

```
<namedcollectionstring> ::= BMPString
      (FROM (<alternativelist>))
      -- represents the collection of characters <collectionstring>,
      -- see ISO/IEC 10646-1.
```

donde:

a) <*collectionstring*> es el nombre para la colección de caracteres asignada en ISO/CEI 10646-1;

b) <*namedcollectionstring*> se forma aplicando a <*collectionstring*> los procedimientos de 38.3;

c) <*alternativelist*> se forma utilizando los <*namedcharacter*> generados como se indica en 38.2 para cada uno de los caracteres especificados por ISO/CEI 10646-1.

La referencia de tipo resultante, <*namedcollectionstring*>, forma un subconjunto limitado. (Véase el suplemento didáctico del anexo F.)

NOTA – Un subconjunto limitado es una lista de caracteres de un subconjunto especificado. Un subconjunto seleccionado es, en cambio, una colección de caracteres enumerados en ISO/CEI 10646-1, anexo A, más la colección LATÍN BÁSICO.

EJEMPLO (parcial)

```
      space BMPString              ::= {0, 0, 0, 32}
      exclamationMark BMPString    ::= {0, 0, 0, 33}
      quotationMark BMPString      ::= {0, 0, 0, 34}
      ...                          -- and so on
      tilde BMPString              ::= {0, 0, 0, 126}

      BasicLatin ::= BMPString
            (FROM (space
            | exclamationMark
            | quotationMark
            | ...       -- and so on
            | tilde)
            )
-- represents the collection of characters BASIC LATIN, see ISO/IEC 10646-1.
-- The ellipsis in this example is used for brevity and means "and so on";
-- you cannot use this in an actual ASN.1 module.
```

38.1.4 La Norma ISO/CEI 10646-1 define tres niveles de implementación. Por defecto, todos los tipos definidos en **ASN1-CHARACTER-MODULE**, excepto para **Level1** y **Level2**, se atienen a la implementación de nivel 3, porque dichos tipos no tienen restricciones a la utilización de los caracteres combinantes. **Level1** indica que se requiere la implementación de nivel 1, **Level2** indica que se requiere la implementación de nivel 2 y **Level3** indica que se requiere la implementación de nivel 3. Así pues, en **ASN1-CHARACTER-MODULE** se define lo siguiente:

```
      Level1 ::= BMPString (FROM (ALL EXCEPT CombiningCharacters))

      Level2 ::= BMPString (FROM (ALL EXCEPT CombiningCharactersType-2))
```

```
Level3 ::= BMPString
```

NOTA 1 – Los **CombiningCharacters** y los **CombiningCharactersType-2**" son las <namedcollectionstring> correspondientes a "COMBINING CHARACTERS" y "COMBINING CHARACTERS B-2", respectivamente, definidas en ISO/CEI 10646-1, anexo A.

NOTA 2 – "**Level1**" y "**Level2**" se utilizarán siguiendo una "IntersectionMark" (véase la cláusula 46) o como la única constricción en una "ConstraintSpec". (Véase un ejemplo en E.2.7.1.)

NOTA 3 – Para más información sobre este tema véase F.2.5.

38.1.5 El módulo se termina con el enunciado:

```
END
```

38.1.6 Un equivalente al ejemplo de 38.1.3 definido por el usuario es:

```
BasicLatin ::= BMPString (FROM (space..tilde))
-- represents the collection of characters BASIC LATIN,
-- see ISO/IEC 10646-1.
```

38.2 Un <*namedcharacter*> es la cadena obtenida tomando un <*iso10646name*> (véase 38.1.2) y aplicando el siguiente algoritmo:

a) cada letra mayúscula del <*iso10646name*> se transforma en la correspondiente letra minúscula, a menos que la letra mayúscula esté precedida por un SPACE (carácter de espacio), en cuyo caso se mantiene la mayúscula;

b) se mantiene sin cambiar cada dígito y cada HYPHEN-MINUS (guión signo menos);

c) se suprime cada SPACE (carácter de espacio).

NOTA – Este algoritmo, junto con las directrices relativas a la denominación de caracteres del anexo K de ISO/CEI 10646-1, permite obtener siempre una notación de valor inequívoca para todos los nombres de carácter indicados en ISO/CEI 10646-1.

EJEMPLO – El carácter de ISO/CEI 10646-1, fila 0, célula 60, que lleva por nombre "LESS-THAN SIGN" (signo menor que) y tiene la representación gráfica "<" puede ser referenciado mediante el "DefinedValue" de:

```
less-thanSign
```

38.3 Una <*namedcollectionstring*> es la cadena obtenida tomando <*collectionstring*> y aplicándole el siguiente algoritmo:

a) cada letra mayúscula del nombre de colección de ISO/CEI 10646-1 se transforma en la correspondiente letra minúscula, a menos que la letra mayúscula esté precedida por un SPACE (carácter de espacio), o sea la primera letra del nombre, en cuyo caso se mantiene la mayúscula;

b) se mantiene sin cambiar cada dígito y cada HYPHEN-MINUS (guión signo menos);

c) se suprime cada SPACE (carácter de espacio).

EJEMPLOS

1) La colección identificada en el anexo A de ISO/CEI 10646-1 como:

BASIC LATIN

tiene la referencia de tipo ASN.1:

```
BasicLatin
```

2) Un tipo cadena de caracteres constituido por los caracteres pertenecientes a la colección LATÍN BÁSICO, junto con la colección ÁRABE BÁSICO, podría definirse como sigue:

```
My-Character-String ::= BMPString (FROM (BasicLatin | BasicArabic) )
```

NOTA – La construcción anterior es necesaria porque la construcción, aparentemente más sencilla, de:

```
My-Character-String ::= BMPString (BasicLatin | BasicArabic)
```

sólo permitiría cadenas que fuesen enteramente LATÍN BÁSICO o ÁRABE BÁSICO, pero no una mezcla de ambos.

39 Orden canónico de los caracteres

39.1 A efectos de subtipificación de "ValueRange" y posible utilización por las reglas de codificación se especifica un ordenamiento canónico de caracteres para **UniversalString**, **UTF8String**, **BMPString**, **NumericString**, **PrintableString**, **VisibleString**, e **IA5String**.

39.2 Para los fines de esta cláusula solamente, un carácter es una correspondencia de uno a uno con una célula en una tabla de códigos, tanto si a la célula se le ha asignado un nombre como una forma de carácter y tanto si es un carácter de control como si es un carácter de impresión, un carácter combinante o un carácter no combinante.

39.3 El orden canónico de un carácter abstracto está definido por el orden canónico de su valor en la representación de 32 bits de ISO/CEI 10646-1, donde los números inferiores aparecen en primer lugar y los números superiores aparecen al final en el orden canónico.

39.4 Los puntos extremos de los "ValueRanges" de las notaciones "PermittedAlphabet" (o caracteres individuales) pueden especificarse utilizando la referencia de valor ASN.1 definida en el módulo **ASN1-CHARACTER-MODULE** o (cuando el símbolo gráfico sea inequívoco en el contexto de la especificación y el medio utilizado para representarlo) dando el símbolo gráfico de una "cstring" (**ASN1-CHARACTER-MODULE** se define en 38.1) o utilizando la notación de "Quadruple" o "Tuple" de 37.8.

39.5 Para **NumericString**, el orden canónico, creciente de izquierda a derecha, se define (véase el cuadro 7 de 37.2) como:

(espacio) **0 1 2 3 4 5 6 7 8 9**

El conjunto de caracteres contiene, en su totalidad, exactamente 11 caracteres. El punto extremo de una "ValueRange" (o caracteres individuales) puede especificarse utilizando el símbolo gráfico de una "cstring".

NOTA – Este orden es el mismo que el de los caracteres correspondientes de una colección LATÍN BÁSICO de ISO/CEI 10646-1.

39.6 Para **PrintableString**, el orden canónico, creciente de izquierda a derecha y de arriba abajo, se define (véase el cuadro 8 de 37.4) como:

(ESPACIO) (APÓSTROFO) (PARÉNTESIS IZQUIERDO) (PARÉNTESIS DERECHO)
(SIGNO MÁS) (COMA) (GUIÓN MENOS) (PUNTO) (BARRA DE FRACCIÓN) **0123456789**
(DOS PUNTOS) (SIGNO IGUAL) (INTERROGACIÓN)
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

El conjunto de caracteres contiene, en su totalidad, exactamente 74 caracteres. El punto extremo de una "ValueRange" (o caracteres individuales) puede especificarse utilizando el símbolo gráfico de una "cstring".

NOTA – Este orden es el mismo que el de los caracteres correspondientes de una colección LATÍN BÁSICO de ISO/CEI 10646-1.

39.7 Para **VisibleString**, el orden canónico de las células se define a partir de la codificación de ISO/CEI 646 (llamada ISO 646 ENCODING) de la siguiente manera:

(ISO 646 ENCODING) - 32

NOTA – Es decir, el orden canónico es el mismo que el de los caracteres de las células 2/0-7/14 de la tabla de códigos de ISO/CEI 646.

El conjunto de caracteres contiene, en su totalidad, exactamente 95 caracteres. El punto extremo de una "ValueRange" (o caracteres individuales) puede especificarse utilizando el símbolo gráfico de una "cstring".

39.8 Para **IA5String**, el orden canónico de las células se define a partir de la codificación ISO/CEI 646 de la siguiente manera:

(ISO 646 ENCODING)

El conjunto de caracteres contiene, en su totalidad, exactamente 128 caracteres. El punto extremo de una "ValueRange" (o caracteres individuales) puede especificarse utilizando el símbolo gráfico de una "cstring" o una referencia de valor de carácter de control de ISO 646 definida en 38.1.1.

40 Definición de tipos cadena de caracteres no restringida

Esta cláusula define un tipo cuyos valores son los valores de cualquier sintaxis abstracta de caracteres. En un entorno OSI, esta sintaxis abstracta puede formar parte del conjunto de contexto definido por OSI. De lo contrario, la referencia se establece directamente para cada ejemplar de uso del tipo cadena de caracteres no restringida.

NOTA 1 – Una sintaxis abstracta de caracteres (y una o más sintaxis de transferencia de caracteres correspondientes) puede ser definida por cualquier organización capaz de atribuir los **OBJECT IDENTIFIER** de ASN.1.

NOTA 2 – Perfiles producidos por comunidades de interés determinarán normalmente las sintaxis abstractas de caracteres y las sintaxis de transferencia de caracteres que son soportadas para ejemplares específicos o grupos de ejemplares de **CHARACTER STRING**. Por lo general, en aplicaciones OSI será habitual incluir una referencia a las sintaxis soportadas en un enunciado de conformidad de implementación de protocolo (PICS, *protocol implementation conformance statement*) de OSI.

40.1 El tipo cadena de caracteres no restringida (véase 3.6.76) se referenciará por la notación "UnrestrictedCharacterStringType":

UnrestrictedCharacterStringType ::= `CHARACTER STRING`

40.2 Este tipo tiene un rótulo que es de clase universal, número 29.

40.3 El tipo consta de valores que representan:

a) un valor cadena de caracteres que puede, pero que no necesita, ser el valor de un tipo cadena de caracteres ASN.1; y

b) identificación (por separado o junto) de:

1) una sintaxis abstracta de caracteres; y

2) la sintaxis de transferencia de carácter.

40.4 El tipo cadena de caracteres no restringida tiene un tipo asociado. Este tipo asociado se utiliza para soportar su valor y las notaciones de subtipo.

40.5 El tipo asociado para la definición de valor y la subtipificación, suponiendo un entorno de rotulación automática, es (con comentarios normativos):

```
SEQUENCE {
     identification                          CHOICE {
          syntaxes                             SEQUENCE {
               abstract                           OBJECT IDENTIFIER,
               transfer                           OBJECT IDENTIFIER }
               --  Abstract and transfer syntax object identifiers --,

          syntax                                  OBJECT IDENTIFIER
               --  A single object identifier for identification of the
               -- abstract and transfer syntaxes --,
          presentation-context-id                 INTEGER
               --  (Applicable only to OSI environments)
               --  The negotiated OSI presentation context identifies the
               -- abstract and transfer syntaxes  --,

          context-negotiation                     SEQUENCE {
               presentation-context-id               INTEGER,
               transfer-syntax                       OBJECT IDENTIFIER }
          -- (Applicable only to OSI environments)
          -- Context-negotiation in progress, presentation-context-id
          -- identifies only the
          -- abstract-syntax, so the transfer syntax shall be specified --,

          transfer-syntax                           OBJECT IDENTIFIER
          --  The type of the value (for example, specification that it is
          -- the value of an ASN.1 type) is fixed by the application
          -- designer (and hence known to both sender and receiver). This
          -- case is provided primarily to support
          -- selective-field-encryption (or other encoding
          --  transformations) of an ASN.1 type  --,

          fixed                                     NULL
          -- The data value is the value of a fixed ASN.1 type (and hence
          -- known to both sender and receiver)  -- },

     data-value-descriptor                   ObjectDescriptor  OPTIONAL
          -- This provides human-readable identification of the class of
          -- the value  --,
     string-value                             OCTET STRING }
     ( WITH COMPONENTS {
          ... ,
          data-value-descriptor  ABSENT } )
```

NOTA – El tipo cadena de caracteres no restringida no permite la inclusión de un valor **`data-value-descriptor`** (descriptor de valor de dato) junto con la **`identification`**. Sin embargo, la definición del tipo asociado que aquí se da sustenta las comunalidades que existen entre el tipo pdv incrustado, el tipo externo y el tipo cadena de caracteres no restringida.

40.6 El texto de 33.6 y 33.7 se aplica también al tipo cadena de caracteres no restringida.

40.7 La notación de valor debe ser la notación de valor del tipo asociado, donde el valor del componente **`string-value`** del tipo **`OCTET STRING`** representa una codificación que utiliza la sintaxis de transferencia especificada en **`identification`**.

UnrestrictedCharacterStringValue ::= SequenceValue -- *value of associated type defined in 40.5*

XMLUnrestrictedCharacterStringValue ::=
XMLSequenceValue -- *value of associated type defined in 40.5*

40.8 En E.2.8 figura un ejemplo de tipo cadena de caracteres no restringida.

41 Notación para tipos definidos en las cláusulas 42 a 44

41.1 La notación para referenciar un tipo definido en las cláusulas 42 a 44 será:

UsefulType ::= typereference

donde "typereference" es una de las definidas en las cláusulas 42 a 44 utilizando la notación ASN.1.

41.2 El rótulo de cada "UsefulType" se especifica en las cláusulas 42 a 44.

42 Generalized time (hora generalizada)

42.1 Este tipo se referenciará por el nombre:

```
GeneralizedTime
```

42.2 Está constituido por valores que representan:

a) una fecha de calendario, definida en ISO 8601; y

b) una hora del día, con cualquiera de las precisiones definidas en ISO 8601, excepto que el valor 24 para las horas no se utilizará; y

c) el factor diferencial de hora local, definido en ISO 8601.

42.3 El tipo puede definirse, utilizando la notación ASN.1, de la forma siguiente:

```
GeneralizedTime ::=    [UNIVERSAL 24] IMPLICIT VisibleString
```

con los valores de la `VisibleString` restringidos a cadenas de caracteres que son o bien:

a) una cadena que representa la fecha de calendario, como se especifica en ISO 8601, con una representación de cuatro dígitos para el año, una representación de dos dígitos para el mes y una representación de dos dígitos para el día, sin separadores, seguida de una cadena que representa la hora del día, especificada en ISO 8601, sin otros separadores que la coma o punto decimal (como se prescribe en ISO 8601) y sin ninguna Z de terminación (como se prescribe en ISO 8601); o

b) los caracteres señalados en el apartado a) anterior seguidos por una letra `Z` mayúscula; o

c) los caracteres señalados en el apartado a) anterior seguidos por una cadena que representa un diferencial de hora local, como se especifica en ISO 8601, sin separadores.

En el caso a), la hora representará la hora local. En el caso b), la hora representará la hora de tiempo universal coordinado. En el caso c), la parte de la cadena formada como en el caso a) representa la hora local (t_1), y el diferencial de hora (t_2) permite determinar la hora de tiempo universal coordinado como sigue:

Hora de tiempo universal coordinado: $t_1 - t_2$

EJEMPLOS

Caso a)

`"19851106210627.3"`
hora local 21 horas, 6 minutos, 27,3 segundos del 6 de noviembre de 1985.

Caso b)

`"19851106210627.3Z"`
hora UTC igual que la anterior.

Caso c)

`"19851106210627.3-0500"`
hora local como en el ejemplo a) con la hora local retrasada 5 horas con relación a la hora de tiempo universal coordinado.

42.4 El rótulo será como se define en 42.3.

42.5 La notación de valor será la notación de valor para la **`VisibleString`** definida en 42.3.

43 Hora universal

43.1 Este tipo se referenciará por el nombre:

```
UTCTime
```

43.2 El tipo está constituido por valores que representan:

a) una fecha de calendario; y

b) una hora con una precisión de un minuto o un segundo; y

c) (opcionalmente) un diferencial de hora local con respecto a la hora de tiempo universal coordinado.

43.3 El tipo puede definirse, utilizando la notación ASN.1, de la forma siguiente:

```
UTCTime ::= [UNIVERSAL 23] IMPLICIT VisibleString
```

con los valores de la **`VisibleString`** restringidos a cadenas de caracteres que son la yuxtaposición de:

a) los seis dígitos YYMMDD donde YY son los dos dígitos de orden inferior del año cristiano, MM es el mes (contando enero como 01), y DD es el día del mes (01 a 31); y

b) o bien:

   1) los cuatro dígitos hhmm donde hh son la hora (00 a 23) y mm son los minutos (00 a 59); o

   2) los seis dígitos hhmmss donde hh y mm son como en 1), y ss son los segundos (00 a 59); y

c) o bien:

   1) el carácter **`z`**; o

   2) uno de los caracteres + o –, seguidos de hhmm, donde hh es hora y mm es minutos.

Las alternativas de b) permiten variar la precisión en la especificación de la hora.

En la alternativa c) 1), la hora es hora del tiempo universal coordinado. En la alternativa c) 2), la hora (t_1) especificada en a) y b) es la hora local; el diferencial de hora (t_2) especificado en c) 2) permite expresar la hora del tiempo universal coordinado de la forma siguiente:

Hora del tiempo universal coordinado: $t_1 - t_2$

EJEMPLO 1 – Si la hora local es las 7 de la mañana del 2 de enero de 1982 y la hora de tiempo universal coordinado es las doce del mediodía del 2 de enero de 1982, el valor de **`UTCTime`** es uno de los siguientes:

– **`"8201021200Z"`**; ó
– **`"8201020700-0500"`**.

EJEMPLO 2 – Si la hora local es las 7 de la mañana del 2 de enero de 2001 y la hora del tiempo universal coordinado es las doce del mediodía del 2 de enero de 2001, el valor de **`UTCTime`** es uno de los siguientes:

– **`"0101021200Z"`**; ó
– **`"0101020700-0500"`**.

43.4 El rótulo será como se define en 43.3.

43.5 La notación de valor será la notación de valor para la **`VisibleString`** definida en 43.3.

44 Tipo descriptor de objeto

44.1 Este tipo se referenciará por el nombre:

```
ObjectDescriptor
```

44.2 El tipo descriptor de objeto está constituido por texto legible por los seres humanos que sirve para describir un objeto. El texto no es una identificación inequívoca del objeto, si bien deberá ser poco habitual un mismo texto para objetos diferentes.

NOTA – Se recomienda que una autoridad que asigne valores de tipo **`OBJECT IDENTIFIER`** a un objeto asigne también valores de tipo **`ObjectDescriptor`** a ese objeto.

44.3 El tipo puede definirse, utilizando la notación ASN.1, de la forma siguiente:

```
ObjectDescriptor ::= [UNIVERSAL 7] IMPLICIT GraphicString
```

La **GraphicString** contiene el texto que describe el objeto.

44.4 El rótulo será como se define en 44.3.

44.5 La notación de valor será la notación de valor para la **GraphicString** definida en 44.3.

45 Tipos constreñidos

45.1 La notación "tipo constreñido" ("ConstrainedType") permite aplicar una constricción a un tipo progenitor (parent), sea para restringir su conjunto de valores a algún subtipo del progenitor, sea (dentro de un tipo conjunto o un tipo secuencia) para especificar que unas relaciones de componentes se aplican a valores del tipo progenitor y a valores de algún otro componente en el mismo valor conjunto o secuencia. Permite también asociar un identificador de excepción a una constricción.

ConstrainedType ::=
Type Constraint
| **TypeWithConstraint**

En la primera alternativa, el tipo progenitor es "Type", y la constricción es especificada por "Constraint" como se define en 45.6. La segunda alternativa se define en 45.5.

45.2 Cuando la notación "Constraint" sigue a una notación de tipo conjunto de (set-of) o secuencia de (sequence-of), se aplica al "Type" en la notación set-of o sequence-of (más interior), no al tipo conjunto de o secuencia de.

NOTA – Por ejemplo, en lo que sigue la limitación **(SIZE(1..64))** se aplica a la **VisibleString**, no a la **SEQUENCE OF**:

```
NamesOfMemberNations ::= SEQUENCE OF VisibleString (SIZE(1..64))
```

45.3 Cuando la notación "Constraint" sigue a la notación de tipo selección (selection), se aplica al tipo elección (choice), y no al tipo de la alternativa seleccionada. Dicho tipo de constricción se ignora (véase 29.2).

NOTA – En el siguiente ejemplo, la constricción **(WITH COMPONENTS {..., a ABSENT})** se aplica al tipo **CHOICE T**, y no al tipo **SEQUENCE** seleccionado, y no produce efecto sobre los valores de **V**.

```
T ::= CHOICE {
      a SEQUENCE {
            a INTEGER OPTIONAL,
            b BOOLEAN
      },
      b NULL
}

V ::= a < T (WITH COMPONENTS {..., a ABSENT})
```

45.4 Cuando la notación "Constraint" sigue a una notación "TaggedType", la interpretación de la notación global es la misma independientemente de que el "TaggedType" o el "Type" se considere como el tipo progenitor.

45.5 Como consecuencia de la interpretación especificada en 45.2, se proporciona una notación especial para permitir aplicar una constricción a un tipo conjunto de o secuencia de. Esta notación especial es "TypeWithConstraint":

TypeWithConstraint ::=
SET Constraint OF Type
| **SET SizeConstraint OF Type**
| **SEQUENCE Constraint OF Type**
| **SEQUENCE SizeConstraint OF Type**
| **SET Constraint OF NamedType**
| **SET SizeConstraint OF NamedType**
| **SEQUENCE Constraint OF NamedType**
| **SEQUENCE SizeConstraint OF NamedType**

En las alternativas primera y segunda, el tipo progenitor es "**SET OF** Type", mientras que en las alternativas tercera y cuarta es "**SEQUENCE OF** Type". En las alternativas quinta y sexta el tipo progenitor es "**SET OF** NamedType", y en la séptima y octava es "**SEQUENCE OF** NamedType". En la primera, tercera, quinta y séptima alternativas, la constricción es "Constraint" (véase 45.6), mientras que en las alternativas segunda, cuarta, sexta y octava es "SizeConstraint" (véase 47.5).

NOTA – Aunque las alternativas "Constraint" comprenden las correspondientes alternativas "SizeConstraint", las alternativas "SizeConstraint" se proporcionan por motivos históricos.

45.6 Una constricción se especifica por la notación "Constraint":

Constraint ::= "(" ConstraintSpec ExceptionSpec ")"

ConstraintSpec ::=
SubtypeConstraint
| **GeneralConstraint**

"ExceptionSpec" se define en la cláusula 49. A menos que se utilice junto con un "marcador de extensión" (véase la cláusula 48), sólo deberá estar presente si la "ConstraintSpec" incluye una ocurrencia de "DummyReference" (véase la Rec. UIT-T X.683 | ISO/CEI 8824-4, 8.3) o es una "UserDefinedConstraint" (véase la Rec. UIT-T X.682 | ISO/CEI 8824-3, cláusula 9). La "GeneralConstraint" se define en la Rec. UIT-T X.682 | ISO/CEI 8824-3, 8.1.

45.7 La notación "SubtypeConstraint" es la notación "ElementSetSpecs" de propósito general (véase la cláusula 46):

SubtypeConstraint ::= ElementSetSpecs

En este contexto, los elementos son valores del tipo progenitor (el gobernador del conjunto de elementos es el tipo progenitor). Deberá haber por lo menos un elemento del conjunto.

46 Especificación de conjunto de elementos

46.1 En algunas notaciones puede especificarse un conjunto de elementos de algún tipo identificado o de alguna clase de objeto de información identificada (el gobernador). En esos casos se utiliza la notación "ElementSetSpec":

ElementSetSpecs ::=
RootElementSetSpec
| **RootElementSetSpec "," "..."**
| **RootElementSetSpec "," "..." "," AdditionalElementSetSpec**

RootElementSetSpec ::= ElementSetSpec

AdditionalElementSetSpec ::= ElementSetSpec

ElementSetSpec ::= Unions
| **ALL Exclusions**

Unions ::= Intersections
| **UElems UnionMark Intersections**

UElems ::= Unions

Intersections ::= IntersectionElements
| **IElems IntersectionMark IntersectionElements**

IElems ::= Intersections

IntersectionElements ::= Elements | Elems Exclusions

Elems ::= Elements

Exclusions ::= EXCEPT Elements

UnionMark ::= "|" | UNION

IntersectionMark ::= "^" | INTERSECTION

NOTA 1 – El carácter de intercalación "^" y la palabra **INTERSECTION** son sinónimos. El carácter "|" y la palabra **UNION** son sinónimos. Se recomienda que, por cuestión de estilo, a lo largo de la especificación del usuario se utilicen o bien los caracteres o bien las palabras. **EXCEPT** puede utilizarse con cualquiera de los estilos.

NOTA 2 – El orden de precedencia de mayor a menor es: **EXCEPT**, "^", "|". Se señala que se ha especificado **ALL EXCEPT** por lo que no puede entremezclarse con las otras constricciones sin la utilización del paréntesis en torno a "**ALL EXCEPT** xxx".

NOTA 3 – Dondequiera que aparezca "Elements", puede aparecer una constricción sin paréntesis [por ejemplo, **INTEGER (1..4)**] o una constricción de subtipo entre paréntesis [por ejemplo, **INTEGER ((1..4 | 9))**].

NOTA 4 – Se señala que dos operadores **EXCEPT** deben tener "|", "^", "(" o ")" separándoles, por lo que no está permitido (**A EXCEPT B EXCEPT C**) que debe cambiarse a ((**A EXCEPT B**) **EXCEPT C**) o (**A EXCEPT** (**B EXCEPT C**)).

NOTA 5 – Se señala que ((**A EXCEPT B**) **EXCEPT C**) es lo mismo que (**A EXCEPT (B | C)**).

NOTA 6 – Los elementos que se referencian por "ElementSetSpecs" son la unión de los elementos referenciados por "RootElementSetSpec" y "AdditionalElementSetSpec" (si los hubiere).

NOTA 7 – Cuando los elementos son objetos de información (es decir, el gobernador es una clase de objeto de información), se utiliza la notación "ObjectSetElements" que se define en la Rec. UIT-T X.681 | ISO/CEI 8824-2, 12.3.

46.2 Los elementos que forman el conjunto son:

a) si se selecciona la primera alternativa de la "ElementSetSpec", los especificados en las "Unions" [véase b)]; de no ser así, todos los elementos del gobernador excepto los especificados en la notación "Elements" de las "Exclusions";

b) si se selecciona la primera alternativa de "Unions", los especificados en las "Intersections" [véase c)]; de no ser así, los especificados por lo menos una vez en los UElems" o en las "Intersections";

c) si se selecciona la primera alternativa de "Intersections", los especificados en los "IntersectionElements" [véase d)]; de no ser así, los especificados por "IElems" que también son especificados por "IntersectionElements";

d) si se selecciona la primera alternativa de "IntersectionElements", los especificados en los "Elements"; de no ser así, los especificados en los "Elems" excepto los especificados en las "Exclusions".

46.3 El conjunto de valores se define como extensible si se mantienen las condiciones siguientes:

a) para los "Elements": hay un marcador de extensión en el nivel exterior;

NOTA – Esto se aplica incluso si todos los valores del progenitor están incluidos en la raíz del nuevo tipo constreñido.

b) para las "Unions": al menos uno de los "UElems" es extensible;

c) para las "Intersections": al menos uno de los "Unions" es extensible;

d) para las "Exclusions": el conjunto de elementos que precede a **`EXCEPT`** es extensible.

En los demás casos, el conjunto de valores no es extensible (véase asimismo G.4).

46.4 Si el conjunto de valores es extensible, los valores raíz podrán determinarse ejecutando la aritmética de conjuntos y usando sólo los valores raíz de los conjuntos de valores que intervienen en la aritmética de conjuntos, como se especifica en 46.2. Las adiciones de extensión podrán determinarse ejecutando la aritmética de conjuntos y empleando los valores raíz aumentados por las adiciones de extensión, para cada conjunto de valores que interviene en la aritmética de conjuntos, y a continuación excluyendo los valores que se determinaron como los valores raíz.

46.5 La notación "Elements" se define como sigue:

```
Elements ::=
        SubtypeElements
    |   ObjectSetElements
    |   "(" ElementSetSpec ")"
```

Los elementos especificados por esta notación son:

a) los descritos en la cláusula 47, si se utiliza la alternativa "SubtypeElements". Esta notación sólo se utilizará cuando el gobernador sea un tipo, y el tipo que efectivamente interviene constreñirá más aún las posibilidades notacionales. En este contexto, se hace referencia al gobernador como el tipo progenitor;

b) los descritos en la Rec. UIT-T X.681 | ISO/CEI 8824-2, 12.10, si se utiliza la notación "ObjectSetElements". Esta notación sólo se utilizará cuando el gobernador sea una clase de objeto de información;

c) los especificados por la "ElementSetSpec" si se utiliza la tercera alternativa.

46.6 Si se aplica la aritmética de conjuntos dentro de una constricción de subtipo o un conjunto de valores cuando el tipo gobernante es extensible, en la aritmética de conjuntos se utilizan solamente los valores abstractos que están en la raíz de extensión del tipo gobernante. En este caso, se requieren todos los ejemplares de la notación de valor (incluidas las referencias de valor) que se emplean en la aritmética de conjuntos para hacer referencia a un valor abstracto de la raíz de extensión del tipo gobernante. Se requieren los puntos extremo de una constricción del rango para hacer referencia a los valores que están presentes en la raíz de extensión del tipo gobernante, y la especificación de gama como un todo hace referencia a todos (y únicamente) los valores del rango que están dentro de la raíz de extensión del tipo gobernante.

46.7 Si se ejecuta la aritmética de conjuntos en la que participan conjuntos de objetos de información, en la aritmética de conjuntos se utilizan todos los conjuntos de información. Si alguno de los conjuntos de objetos de información que contribuyen a la aritmética de conjuntos es extensible, o si hay un marcador de extensión en el nivel exterior de un "ElementSetSpecs", el resultado de la aritmética de conjuntos es extensible.

46.8 Si se aplica una constricción de subtipo en serie a un tipo progenitor que es extensible a través de la aplicación de una constricción extensible, la notación de valor utilizada dentro de ella no hará referencia a valores que no están en la raíz de extensión del tipo progenitor. El resultado de la segunda constricción (que se aplica en serie) se define de la misma forma como si la constricción se hubiese aplicado al tipo progenitor sin su marcador de extensión y sus posibles adiciones de extensión.

EJEMPLO

```
Foo ::= INTEGER ( 1..6, ..., 73..80)
Bar ::= Foo (73) -- illegal
foo Foo ::= 73 -- legal since it is value notation for Foo, not part of a
                  constraint
```

Bar es ilegal ya que 73 no está en la raíz de extensión de **Foo**. Si 73 hubiera estado en la raíz de extensión de **Foo**, el ejemplo habría sido legal, y **Bar** hubiera contenido el valor simple 73.

47 Elementos subtipos

47.1 Generalidades

Se proporciona un cierto número de formas diferentes de notación para "SubtypeElements". Estas formas se identifican más adelante, y sus sintaxis y semántica se definen en las subcláusulas siguientes. El cuadro 9 muestra de forma resumida cuáles son las notaciones que pueden aplicarse y a qué tipos progenitor pueden aplicarse.

SubtypeElements ::=
SingleValue
| **ContainedSubtype**
| **ValueRange**
| **PermittedAlphabet**
| **SizeConstraint**
| **TypeConstraint**
| **InnerTypeConstraints**
| **PatternConstraint**

Cuadro 9 – Aplicabilidad de los conjuntos de valores de subtipo

Tipo (o derivación del mismo por rotulación o subtipificación)	Valor único	Subtipo contenido	Gama de valores	Constricción de tamaño	Alfabeto permitido	Constricción de tipo	Subtipif. interior	Constricción de patrón
Cadena de bits	Sí	Sí	No	Sí	No	No	No	No
Booleano	Sí	Sí	No	No	No	No	No	No
Elección	Sí	Sí	No	No	No	No	Sí	No
pdv incrustado	Sí	No	No	No	No	No	Sí	No
Enumerado)	Sí	Sí	No	No	No	No	No	No
Externo	Sí	No	No	No	No	No	Sí	No
Ejemplar de	Sí	Sí	No	No	No	No	Sí	No
Entero	Sí	Sí	Sí	No	No	No	No	No
Nulo	Sí	Sí	No	No	No	No	No	No
Tipo campo de clase de objeto	Sí	Sí	No	No	No	No	No	No
Descriptor de objeto	Si	Si	No	Si	Si	No	No	No
Identificador de objeto	Sí	Sí	No	No	No	No	No	No
Cadena de octetos	Sí	Sí	No	Sí	No	No	No	No
Tipo abierto	No	No	No	No	No	Sí	No	No
Real	Sí	Sí	Sí	No	No	No	Sí	No
Identificador de objeto relativo	Si[b)]	Si[b)]	No	No	No	No	No	No
Tipos cadena de caracteres restringida	Sí	Sí	Sí[a)]	Sí	Sí	No	No	Sí
Secuencia	Sí	Sí	No	No	No	No	Sí	No
Secuencia de	Sí	Sí	No	Sí	No	No	Sí	No
Conjunto	Sí	Sí	No	No	No	No	Sí	No
Conjunto de	Sí	Sí	No	Sí	No	No	Sí	No
Tipos tiempo	Si	Si	No	No	No	No	No	No
Tipo cadena de caracteres no restringida	Sí	No	No	Sí	No	No	Sí	No

a) Sólo se permite dentro del "PermittedAlphabet" de **`BMPString`**, **`IA5String`**, **`NumericString`**, **`PrintableString`**, **`VisibleString`**, **`UTF8String`** y **`UniversalString`**.

b) El nodo inicial para todos los tipos o valores del identificador de objeto relativo en las constricciones o en los conjuntos de valores será el mismo que el nodo inicial del gobernador.

47.2 Single Value (valor único)

47.2.1 La notación "SingleValue" será:

SingleValue ::= Value

donde "Value" es la notación de valor para el tipo progenitor.

47.2.2 Un "SingleValue" especifica el valor único del tipo progenitor especificado por "Value".

47.3 Contained Subtype (subtipo contenido)

47.3.1 La notación "ContainedSubtype" será:

ContainedSubtype ::= Includes Type

Includes ::= `INCLUDES` | empty

La alternativa "empty" de la producción "Includes" no deberá utilizarse cuando "Type", en "ContainedSubtype", sea la notación para el tipo nulo.

47.3.2 Un "ContainedSubtype" especifica todos los valores en la raíz del tipo progenitor (parent) que están también en la raíz de "Type". Es necesario que "Type" se obtenga del mismo tipo incorporado, que el tipo progenitor.

47.3.3 El conjunto de valores referenciado por un "Type" extensible utilizado en una constricción de subtipo contenida no hereda el marcador de extensión del "Type". Todo valor en "Type" que no esté en la raíz de extensión de ese tipo será ignorado, y no contribuirá a los valores del tipo constreñido.

NOTA – La utilización de un "Type" extensible no logra por si misma que el tipo constreñido sea extensible.

47.4 Value Range (rango de valores)

47.4.1 La notación "ValueRange" será:

ValueRange ::= LowerEndpoint " . . " UpperEndpoint

47.4.2 Una "ValueRange" especifica los valores de un rango de valores que se designan especificando los valores de los puntos extremos del rango. Esta notación sólo puede aplicarse a tipos enteros, al "PermittedAlphabet" de determinados tipos cadena de caracteres restringida (**IA5String**, **NumericString**, **PrintableString**, **VisibleString**, **BMPString**, **UniversalString** y **UTF8String** únicamente) y a tipos reales. Es necesario que todos los valores especificados en la "ValueRange" estén en la raíz del tipo progenitor.

NOTA – A los fines de la subtipificación, **PLUS-INFINITY** es mayor que todos los valores reales y **MINUS-INFINITY** es menor que todos los valores reales.

47.4.3 Cada punto extremo del rango es o bien cerrado (en cuyo caso el punto extremo está especificado) o abierto (en cuyo caso el punto extremo no está especificado). Cuando es abierto, la especificación del punto extremo incluye un símbolo menor que ("<"):

LowerEndpoint ::= LowerEndValue | LowerEndValue "<"

UpperEndpoint ::= UpperEndValue | "<" UpperEndValue

47.4.4 Un punto extremo puede también no estar especificado, en cuyo caso el rango se extiende en ese sentido tanto como lo permita el tipo progenitor:

LowerEndValue ::= Value | MIN

UpperEndValue ::= Value | MAX

NOTA – Cuando se utiliza una "ValueRange" como una constricción "PermittedAlphabet", "LowerEndValue" y "UpperEndValue" serán de tamaño 1.

47.5 Constricción de tamaño

47.5.1 La notación "SizeConstraint" será:

SizeConstraint ::= SIZE Constraint

47.5.2 Una "SizeConstraint" sólo puede aplicarse a tipos cadena de bits, cadena de octetos, cadena de caracteres, conjunto de o secuencia de.

47.5.3 La "Constraint" especifica los valores enteros permitidos para la longitud de los valores especificados, y adopta la forma de cualquier constricción que pueda aplicarse al siguiente tipo progenitor:

```
INTEGER (0 .. MAX)
```

La "Constraint" utilizará la alternativa "SubtypeConstraint" de "ConstraintSpec".

47.5.4 La unidad de medida depende del tipo progenitor, como sigue:

Tipo	*Unidad de medida*
bit string	bit
octet string	octeto
character string	carácter
set-of	valor componente
sequence-of	valor componente

NOTA – La cuenta del número de caracteres especificada en esta subcláusula para determinar el tamaño de un valor de cadena de caracteres deberá distinguirse claramente de una cuenta de octetos. Una cuenta de caracteres se interpretará de acuerdo con la definición de la colección de caracteres utilizada en el tipo, en particular, en relación con las referencias a las normas, cuadros o números de inscripción de un registro que puedan aparecer en esa definición.

47.6 Constricción de tipo

47.6.1 La notación "TypeConstraint" será:

TypeConstraint ::= Type

47.6.2 Esta notación sólo se aplica a una notación de tipo abierto y constriñe el tipo abierto a valores de "Type".

47.7 Alfabeto permitido

47.7.1 La notación "PermittedAlphabet" será:

PermittedAlphabet ::= FROM Constraint

47.7.2 Un "PermittedAlphabet" especifica todos los valores que pueden construirse utilizando un subalfabeto de la cadena progenitora. Esta notación sólo puede aplicarse a tipos cadena de caracteres restringida.

47.7.3 La "Constraint" es cualquiera que pueda aplicarse al tipo progenitor (véase el cuadro 9), con la salvedad de que deberá utilizar la alternativa "SubtypeConstraint" de "ConstraintSpec". El subalfabeto incluye exactamente los caracteres que aparecen en uno o más de los valores del tipo cadena progenitor autorizados por la "Constraint".

47.7.4 Si la "Constraint" es extensible, el conjunto de valores seleccionado por la constricción del alfabeto permitido es extensible. El conjunto de valores en la raíz es el permitido por la raíz de "Constraint" y las adiciones de extensión son aquellos valores permitidos por la raíz junto con las adiciones de extensión de "Constraint", salvo aquellos valores que ya están en la raíz.

47.8 Subtipificación interior

47.8.1 La notación "InnerTypeConstraints" será:

InnerTypeConstraints ::=
WITH COMPONENT SingleTypeConstraint
| WITH COMPONENTS MultipleTypeConstraints

47.8.2 Un "InnerTypeConstraints" especifica solamente los valores que satisfacen una colección de constricciones sobre la presencia y/o valores de los componentes del tipo progenitor. Un valor del tipo progenitor no se especifica, a menos que satisfaga todas las constricciones expresadas o implicadas (véase 47.8.6). Esta notación puede aplicarse a los tipos set-of, sequence-of, set, sequence y choice.

NOTA – Los **COMPONENTS OF** de la transformación ignoran un "InnerTypeConstraints" que se aplica a un tipo conjunto (set) o secuencia (sequence) (véase 24.4 y 26.2).

47.8.3 Para los tipos que se definen en términos de otro tipo (interior) único (set-of y sequence-of), se proporciona una constricción en forma de una especificación de valor de subtipo. La notación para esta constricción es "SingleTypeConstraint":

SingleTypeConstraint ::= Constraint

La "Constraint" define un subtipo del otro tipo (interior) único. Se especifica un valor del tipo progenitor únicamente si cada valor interior pertenece al subtipo obtenido al aplicar la "Constraint" al tipo interior.

47.8.4 Para los tipos que se definen en términos de varios otros tipos (interiores) (choice, set y sequence) puede proporcionarse un número de constricciones sobre los tipos interiores. La notación para estas constricciones es "MultipleTypeConstraints":

MultipleTypeConstraints ::=
FullSpecification
| PartialSpecification

FullSpecification ::= "{" TypeConstraints "}"

PartialSpecification ::= "{" "..." "," TypeConstraints "}"

TypeConstraints ::=
NamedConstraint
| NamedConstraint "," TypeConstraints

NamedConstraint ::=
identifier ComponentConstraint

47.8.5 "TypeConstraint" contiene una lista de constricciones sobre los tipos componentes del tipo progenitor. Para un tipo sequence, las constricciones tienen que aparecer en orden. El tipo interior al que se aplica la constricción se identifica por medio de su identificador. Para un componente dado deberá haber por lo menos una "NamedConstraint".

47.8.6 "MultipleTypeConstraints" comprende una "FullSpecification" o una "PartialSpecification". Cuando se utiliza "FullSpecification", hay una constricción de presencia supuesta, de **ABSENT**, en todos los tipos interiores que puedan constreñirse a estar ausentes (véase 47.8.9) y que no está explícitamente listada. Cuando se emplea "PartialSpecification", no hay constricciones supuestas y cualquier tipo interior puede ser omitido de la lista.

47.8.7 Un determinado tipo interior puede constreñirse en cuanto a su presencia (en valores de tipo progenitor), su valor, o en cuanto a ambos. La notación es "ComponentConstraint":

ComponentConstraint ::= ValueConstraint PresenceConstraint

47.8.8 Una constricción sobre el valor de un tipo interior se expresa por la notación "ValueConstraint":

ValueConstraint ::= Constraint | empty

La constricción se satisface por un valor del tipo progenitor únicamente si el valor interior pertenece al subtipo especificado por la "Constraint" aplicada al tipo interior.

47.8.9 Una constricción sobre la presencia de un tipo interior se expresará por la notación "PresenceConstraint":

PresenceConstraint ::= `PRESENT` | `ABSENT` | `OPTIONAL` | empty

El significado de estas alternativas, en la situaciones en que se permiten, se define en 47.8.9.1 a 47.8.9.3.

47.8.9.1 Si el tipo progenitor es sequence o set, un tipo componente marcado **`OPTIONAL`** puede constreñirse a estar **`PRESENT`** (en cuyo caso la constricción se satisface únicamente si el valor componente correspondiente está presente) o a estar **`ABSENT`** (en cuyo caso la constricción se satisface únicamente si el valor componente correspondiente está ausente) o a ser **`OPTIONAL`** (en cuyo caso no se imponen constricciones a la presencia del valor componente correspondiente).

47.8.9.2 Si el tipo progenitor es una elección, un tipo componente puede constreñirse a estar **`ABSENT`** (en cuyo caso la constricción se satisface únicamente si el tipo componente correspondiente no se utiliza en el valor), o a estar **`PRESENT`** (en cuyo caso la constricción se satisface únicamente si el tipo componente correspondiente se utiliza en el valor); deberá haber a lo sumo una palabra clave **`PRESENT`** en un "MultipleTypeConstraints".

NOTA – Véase un ejemplo en E.4.6.

47.8.9.3 El significado de una "PresenceConstraint" vacía depende de que se esté empleando una "FullSpecification" o una "PartialSpecification":

a) en una "FullSpecification", equivale a una constricción de **`PRESENT`** para un componente set o sequence marcado **`OPTIONAL`** y, en otro caso, no impone ninguna otra constricción;

b) en una "PartialSpecification", no se imponen constricciones.

47.9 Limitación de patrón

47.9.1 La notación "PatternConstraint" será:

PatternConstraint ::= `PATTERN` Value

47.9.2 "Value" será una "cstring" de tipo **`UniversalString`** (o una referencia a esa cadena de caracteres) que contiene una expresión regular ASN.1 definida en el anexo A. La "PatternConstraint" selecciona aquellos valores del tipo del progenitor que satisfacen la expresión regular ASN.1. El valor total deberá satisfacer la expresión regular ASN.1 en su totalidad, es decir, "PatternConstraint" no selecciona valores cuyos caracteres iniciales concuerden con la expresión regular ASN.1 (en su totalidad), pero que contengan más caracteres de cola.

NOTA – "Value" se define formalmente como el valor de tipo **`UniversalString`**, pero los conjuntos de valores de tipo **`UniversalString`** y **`UTF8String`** son los mismos (véase 37.16). Así pues, una definición totalmente equivalente podría haber consistido en decir que "Value" es un valor de tipo **`UTF8String`**.

48 Marcador de extensión

NOTA – Al igual que la notación constricción en general, el marcador de extensión no produce efecto sobre algunas reglas de codificación ASN.1, como las reglas de codificación básica, pero sí lo produce en otras como las reglas de codificación empaquetada. El efecto que produce sobre codificaciones definidas utilizando ECN se determina mediante la especificación de ECN.

48.1 El marcador de extensión, puntos suspensivos, es una indicación de que se esperan adiciones de extensión. No establece enunciados de cómo deben manejarse dichas adiciones salvo el de que no deben tratarse como un error durante el proceso de decodificación.

48.2 La utilización conjunta de un marcador de extensión y de un identificador de excepción (véase la cláusula 49) es una indicación de que se esperan adiciones de extensión y asimismo ofrece un mecanismo para identificar las medidas que habrá de tomar la aplicación si existe una violación de constricción. Se recomienda que esta notación se utilice en aquellas situaciones en las que se esté utilizando el procedimiento de almacenamiento y retransmisión o

cualquier otra forma de retransmisión, de manera que se indique (por ejemplo) que cualquier adición de extensión no reconocida se devuelva a la aplicación para su posible recodificación y retransmisión.

48.3 El resultado de la aritmética de conjuntos con constricciones de subtipos, conjuntos de valores o conjuntos de objetos de información que son extensibles, se especifica en la cláusula 46.

48.4 Si un tipo definido con una constricción extensible se referencia en un "ContainedSubtype", el nuevo tipo definido no hereda el marcador de extensión ni ninguna de sus adiciones de extensión (véase 47.3.3). El nuevo tipo definido puede volverse extensible al incluir un marcador de extensión en el nivel exterior de su "ElementSetSpecs" (véase asimismo 46.3). Por ejemplo:

```
A ::= INTEGER (0..10, ..., 12) -- A is extensible.
B ::= INTEGER (A)              -- B is inextensible and is constrained to 0-10.
C ::= INTEGER (A, …)            - C is extensible and is constrained to 0-10.
```

48.5 Si un tipo definido con una constricción extensible se constriñe aún más con un "ElementSetSpecs", el tipo resultante no hereda el marcador de extensión ni ninguna adición de extensión que pueda estar presente en la primera constricción (véase 46.8). Por ejemplo:

```
A ::= INTEGER (0..10, ...)   -- A is extensible.
B ::= A (2..5)               -- B is inextensible.
C ::= A                      -- C is extensible.
```

48.6 Los componentes de un tipo conjunto, secuencia o elección que se hayan constreñido a estar ausentes no podrán estar presentes, independientemente de que el tipo conjunto, secuencia o elección sea un tipo extensible.

NOTA – Las constricciones de tipo interior no producen efecto sobre la extensibilidad.

Por ejemplo:

```
A ::= SEQUENCE   {
a     INTEGER
b     BOOLEAN OPTIONAL,
...
}

B ::= A (WITH COMPONENTS {b ABSENT})
                              -- B is extensible, but 'b' shall not be
                              -- present in any of its values.
```

48.7 Cuando esta Recomendación | Norma Internacional requiera rótulos distintos (véanse 24.5 a 24.6, 26.3 y 28.3), se aplicarán conceptualmente las siguientes transformaciones antes de efectuar la comprobación de la unicidad de rótulo:

48.7.1 Se añade conceptualmente un nuevo elemento o alternativa (denominado el elemento añadido conceptualmente, véase 48.7.2) en el punto de inserción de extensión si:

a) no hay marcadores de extensión pero la extensibilidad está implícita en el encabezamiento del módulo, y seguidamente se añade un marcador de extensión y se añade el nuevo elemento como primera adición después de ese marcador de extensión; o

b) en un **CHOICE** o **SEQUENCE** o **SET** hay un solo marcador de extensión y seguidamente se añade el nuevo elemento al final de **CHOICE** o **SEQUENCE** o **SET** inmediatamente antes del corchete de cierre; o

c) hay dos marcadores de extensión en **CHOICE** o **SEQUENCE** o **SET** y seguidamente se añade el nuevo elemento inmediatamente antes del segundo marcador de extensión.

48.7.2 Este elemento añadido conceptualmente sólo sirve para comprobar la legalidad mediante la aplicación de reglas que requieren rótulos distintos (véanse 24.5 a 24.6, 26.3 y 28.3). Se añade conceptualmente *después* de la aplicación de la rotulación automática (si procede) y la expansión de **COMPONENTS OF**.

48.7.3 El elemento añadido conceptualmente se define con un rótulo distinto del de todos los tipos ASN.1 normales, pero en concordancia con el rótulo de todos los elementos añadidos conceptualmente, y el rótulo indeterminado del tipo abierto, como se especifica en la Rec. UIT-T X.681 | ISO/CEI 8824-2, 14.2, nota 2.

NOTA – Las reglas sobre unicidad de rótulos relativos al elemento añadido conceptualmente y al tipo abierto, junto con las reglas que requieren distintos rótulos (véanse 24.5 a 24.6, 26.3 y 28.3) son necesarias y suficientes para asegurar que:

a) las adiciones de extensión desconocida pueden atribuirse inequívocamente a un único punto de inserción al decodificar una codificación BER; y

b) las adiciones de extensión desconocidas no pueden ser confundidas con elementos **OPTIONAL**.

En PER las reglas mencionadas son suficientes pero no necesarias para asegurar estas propiedades. Sin embargo, se imponen como reglas de ASN.1 para garantizar la independencia de la notación con respecto a las reglas de codificación.

48.7.4 Si con estos elementos añadidos conceptualmente se violan las reglas que requieren tipos distintos, significa que la especificación ha hecho uso ilegal de la notación de extensibilidad.

NOTA – La finalidad de estas reglas es efectuar restricciones precisas que surjan de la utilización de puntos de inserción (especialmente los que no están al final de las **SEQUENCE** o **SET** o **CHOICE**). Las restricciones están destinadas a garantizar que en BER, DER y CER es posible atribuir inequívocamente a un punto de inserción específico un elemento desconocido recibido por una primera versión de sistema. Esto resultaría importante si el tratamiento de excepción de esos elementos añadidos fuera diferente para puntos de inserción diferentes.

48.8 Ejemplos

48.8.1 Ejemplo 1

```
A ::= SET {
      a     A,
      b     CHOICE {
            c     C,
            d     D,
            ...
      }
}
```

es legal porque no hay ambigüedad, dado que cualquier material añadido debe ser parte de **b**.

48.8.2 Ejemplo 2

```
A ::= SET {
      a     A,
      b     CHOICE {
            c     C,
            d     D,
            ...
      },
      ... ,
      d     D
}
```

es ilegal porque el material añadido puede ser parte de **b**, o puede estar en el nivel exterior de **A**, y una primera versión de sistema no puede determinar de cual se trata.

48.8.3 Ejemplo 3

```
A ::= SET {
      a     A,
      b     CHOICE {
            c     C,
            ...
      } ,
      d     CHOICE {
            e     E,
            ...
      }
}
```

también es ilegal porque el material añadido puede ser parte de **b** o **d**.

48.8.4 Se pueden construir ejemplos más complejos, con elecciones extensibles dentro de elecciones extensibles, o elecciones extensibles dentro de elementos de una secuencia marcada **OPTIONAL** o **DEFAULT**, pero estas reglas son necesarias y suficientes para garantizar que un elemento no presente en la versión 1 pueda atribuirse inequívocamente por un sistema de la versión 1 a un punto de inserción exactamente.

49 Identificador de excepción

49.1 En una especificación ASN.1 compleja hay un cierto número de sitios en los que se reconoce específicamente que los decodificadores han de tratar material que no está completamente especificado en aquélla. Estos casos surgen, en particular, como consecuencia del empleo de una constricción que se ha definido utilizando un parámetro de la sintaxis abstracta (véase la Rec. UIT-T X.683 | ISO/CEI 8824-4, cláusula 10).

49.2 En tales casos, el diseñador de la aplicación tiene que identificar las acciones que han de tomarse cuando se viole alguna constricción dependiente de la implementación. El identificador de excepción se proporciona como una manera inequívoca de referirse a partes de una especificación ASN.1 para indicar las acciones que deben efectuarse. El identificador consiste en el carácter "**!**", seguido de un tipo ASN.1 opcional y un valor de ese tipo. En ausencia del tipo, se supone que **INTEGER** es el tipo del valor.

49.3 La presencia de una "ExceptionSpec", indica que hay texto en el cuerpo de la norma en el que se dice cómo tratar la violación de la constricción asociada al carácter "**!**" . Si no está, los implementadores necesitarán identificar texto que describa la acción que han de efectuar o realizarán acciones dependientes de la implementación cuando se produzca una violación de la constricción.

49.4 La notación "ExceptionSpec" se define como sigue:

ExceptionSpec ::= "!" ExceptionIdentification | empty

ExceptionIdentification ::=
SignedNumber
| DefinedValue
| Type ":" Value

Las dos primeras alternativas denotan identificadores de excepción de tipo integer. La tercera alternativa denota un identificador de excepción ("Value") de tipo arbitrario ("Type").

49.5 Donde un tipo esté constreñido por constricciones múltiples, de las cuales más de una tenga un identificador de excepción, el identificador de excepción de la constricción exterior se considerará el identificador de excepción para ese tipo.

49.6 Si hay un marcador de excepción sobre tipos utilizados en aritmética de conjuntos, se ignora el identificador de excepción y no lo hereda el tipo que se está constriñendo como resultado de la aritmética de conjuntos.

Anexo A

Expresiones ASN.1 regulares

(Este anexo es parte integrante de esta Recomendación | Norma Internacional)

A.1 Definición

A.1.1 Una expresión ASN.1 regular es un patrón que describe un conjunto de cadenas cuyo formato se ajusta a aquél. Una expresión regular es en sí misma una cadena; se construye de manera análoga a las expresiones aritméticas, utilizando diversos operadores para combinar expresiones más pequeñas. Las expresiones de menor tamaño, que constan (normalmente) de uno o dos caracteres, son variables que representan a un conjunto de caracteres.

Las expresiones regulares que aquí se presentan son muy similares a las de los lenguajes de escritura de guiones como Perl y las de los esquemas XML, en los que pueden encontrarse algunos otros ejemplos de uso.

A.1.2 La mayoría de los caracteres, entre ellos todas las cifras y las letras, son expresiones regulares que concuerdan consigo mismas.

EJEMPLO

La expresión regular "**fred**" concuerda solamente con la cadena "**fred**".

A.1.3 Dos expresiones regulares pueden estar concatenadas; la expresión regular resultante concuerda con cualquier cadena formada por dos subcadenas concatenantes que concuerden respectivamente con las subexpresiones concatenadas.

A.2 Metacaracteres

A.2.1 Una secuencia de metacaracteres (o metacarácter) es un conjunto de uno o más caracteres contigüos con un significado especial en el contexto de una expresión regular. La lista siguiente contiene todas las secuencias de metacaracteres. Su significado se explica en las cláusulas que siguen.

[ ]		Concuerda con cualquier carácter del conjunto en el que el rango se indica mediante "**-**". Un "**^**" después del corchete de apertura complementa el conjunto que le sigue.
{g,p,r,c}		Cuádruplo que identifica un carácter de ISO/CEI 10646-1 (véase 37.8)
\N{name}		Concuerda con el carácter denominado (o cualquier carácter del juego de caracteres con nombre) definido en 38.1
.		Concuerda con cualquier carácter (salvo que sea uno de los caracteres de nueva línea definidos en 11.1.6)
\d		Concuerda con cualquier número (equivalente a "**[0-9]**")
\w		Concuerda con cualquier carácter alfanumérico (equivalente a "**[a-zA-Z0-9**]")
\t		Concuerda con el carácter TABULACIÓN HORIZONTAL (9) (véase 11.1.6)
\n		Concuerda con cualquiera de los caracteres de nueva línea definidos en 11.1.6
\r		Concuerda con el carácter RETROCESO DEL CARRO (13) (véase 11.1.6)
\s		Concuerda con cualquiera de los caracteres de espacio en blanco (véase 11.1.6)
\b		Concuerda con un límite de palabra
****	(prefijo)	Introduce el metacarácter siguiente y hace que se interprete literalmente
****		Concuerda con el carácter BARRA DE FRACCIÓN INVERTIDA (92) "\"
""		Concuerda con el carácter COMILLAS (34) (")
|	(infijo)	Alternativa entre dos expresiones
( )		Agrupación de expresión encerrada
*****	(sufijo)	Concuerda con la expresión anterior cero, una o varias veces
+	(sufijo)	Concuerda con la expresión anterior una o varias veces
?	(sufijo)	Concuerda con la expresión anterior una vez o ninguna
#n	(sufijo)	Concuerda con la expresión anterior exactamente n veces (siendo n un número de una sola cifra)
#(n)	(sufijo)	Concuerda con la expresión anterior exactamente n veces
#(n,)	(sufijo)	Concuerda con la expresión anterior al menos n veces
#(n,m)	(sufijo)	Concuerda con la expresión anterior al menos n veces pero no más de m veces
#(,m)	(sufijo)	Concuerda con la expresión anterior no más de m veces

NOTA 1 – Los caracteres ACENTO CIRCUNFLEJO (94) "^" y GUIÓN SIGNO MENOS (45) "-" son metacaracteres adicionales en determinadas posiciones de la cadena definida en A.2.2.

NOTA 2 – El valor entre paréntesis después de un nombre de carácter en este anexo es el valor decimal de los caracteres en ISO/CEI 10646-1.

NOTA 3 – Esta notación no prevé los metacaracteres "^" y "**$**" en concordancia, con el comienzo y el final de una cadena, respectivamente. Por ello, una cadena deberá concordar con una expresión regular en su totalidad excepto si esta última incluye "**.***" en su comienzo, en su final o en ambos lados.

NOTA 4 – Las siguientes secuencias de metacaracteres no pueden incluir un espacio en blanco (véase 11.1.6), salvo si éste aparece inmediatamente antes o después de una línea nueva:

```
{g,p,r,c}
\N{name}
#n
#(n)
#(n,)
#(n,m)
#(,m)
```

Si una expresión regular contiene una línea nueva, cualesquiera caracteres con espaciado que aparezcan inmediatamente antes o después de la línea nueva no poseen ningún significado y no corresponden a nada (véase 11.14.1).

A.2.2 Una lista de caracteres encerrada entre "**[**" y "**]**" corresponde a un solo carácter de esa lista. Si el primer carácter de la lista es el signo de intercalación "^", quiere decir que concuerda con cualquier carácter que no figure en la lista. Se puede especificar un rango de caracteres dando el primero y el último de ellos, separados por un guión (de acuerdo con la relación de orden definida en 39.3). Todas las secuencias de metacaracteres, excepto "**]**" y "\", pierden su significado especial dentro de una lista. Para incluir un literal de ACENTO CIRCUNFLEJO (94) "^", hay que situarlo en cualquier lugar excepto en la primera posición, o bien precedido por una barra de fracción invertida. Para incluir un literal GUIÓN SIGNO MENOS (45) "-", hay que situarlo en el primero o en el último lugar de la lista, o bien precedido por una barra de fracción invertida. Para incluir un literal CORCHETE DE CIERRE (93) "**]**", hay que situarlo en primer lugar. Si el primer carácter en la lista es el signo de intercalación "^", entonces los caracteres "**-**" y "**]**" también concuerdan cuando siguen inmediatamente ese carácter. Las secuencias metacaracteres definidas en A.2.3, A.2.4, A.2.6 y A.2.7 se pueden utilizar entre corchetes conservando su significado.

EJEMPLOS

La expresión regular "**[0123456789]**", o el equivalente "**[0-9]**", concuerda con una sola cifra, cualquiera que sea.

La expresión regular "**[^0]**" concuerda con un solo carácter, cualquiera que sea excepto **0**.

La expresión regular "**[\d^.-]**" concuerda con una sola cifra, signo de intercalación, guión o punto.

A.2.3 Para evitar cualquier ambigüedad entre dos caracteres ISO/CEI 10646-1 que tienen el mismo glifo, se prevén dos notaciones. Una notación de la forma "**{**grupo, plano, fila, celda**}**" hace referencia a un (único) carácter de acuerdo con la producción "Cuádruplo" definida en 37.8.

A.2.4 Una notación de la forma "**\N{**valuereference**}**" concuerda con el carácter referenciado si "valuereference" es una referencia a un valor de cadena de caracteres restringido de tamaño 1 (véase la cláusula 37) que se define o se importa al módulo actual. Una notación de la forma "**\N{**typereference**}**" concuerda con cualquier carácter del conjunto de caracteres referenciado si "typereference" es una referencia a un subtipo de un "RestrictedCharacterStringType" que se define en el módulo actual, o es uno de los "RestrictedCharacterStringType" definidos en la cláusula 37.

NOTA – En particular, "valuereference" o "typereference" pueden ser una de las referencias definidas en el módulo **ASN1-CHARACTER-MODULE** (véase 38.1) e importadas al módulo actual (véase 37.8).

EJEMPLOS

La expresión regular **"\N{greekCapitalLetterSigma}"** concuerda con GREEK CAPITAL LETTER SIGMA.

La expresión regular **"\N{BasicLatin}"** concuerda con cualquier carácter (único) del conjunto de caracteres BASIC LATIN.

"[\N{BasicLatin}\N{Cyrillic}\N{BasicGreek}]+" o de manera equivalente **"(\N{BasicLatin}** | **\N{Cyrillic}** | **\N{BasicGreek})+"**, son expresiones regulares que concuerdan con una cadena formada por cualquier número (no nulo) de caracteres de los tres conjuntos de caracteres especificados.

A.2.5 El punto "**.**" concuerda con cualquier carácter sencillo, salvo que sea uno de los caracteres de nueva línea definidos en 11.1.6.

A.2.6 El símbolo "**\d**" es un sinónimo de "**[0-9]**", es decir, concuerda con una sola cifra, cualquiera que sea. El símbolo "**\t**" concuerda con el carácter TABULACIÓN HORIZONTAL (9). El símbolo "**\w**" es un sinónimo de "**[a-zA-Z0-9]**", es decir, concuerda con un solo carácter, cualquiera que sea (minúscula o mayúscula) o con una sola cifra.

EJEMPLO

La expresión regular "**\w+(\s\w+)*\.**" concuerda con una oración constituida por al menos una palabra (alfanumérica). Las palabras se separan mediante un carácter de espacio en blanco como se define en 11.1.6. No hay carácter de espacio en blanco antes del punto final.

A.2.7 El símbolo "**\r**" concuerda con el carácter RETROCESO DEL CARRO (13). El símbolo "**\n**" concuerda con cualquiera de los caracteres de nueva línea definidos en 11.1.6. El símbolo "**\s**" concuerda con cualquiera de los caracteres de espacio en blanco definidos en 11.1.6. El símbolo "**\b**" concuerda con la cadena vacía al principio o al final de una palabra.

EJEMPLO

La expresión regular "**.*\bfred\b.***" concuerda con cualquier cadena que incluya la palabra "**fred**" (esta palabra no sólo es una serie de cuatro caracteres sino que además está delimitada). Por ello concuerda con cadenas tales como **fred"** o **"I am fred the first"**, pero no en cadenas tales como **"My name is freddy"** o **"I am afred I don't know how to spell 'afraid'!"**.

A.2.8 Un carácter que funciona normalmente como metacarácter puede interpretarse literalmente colocándole como prefijo un "\". Si la expresión regular incluye COMILLAS (34) (**"**), este carácter deberá estar representado por un par caracteres COMILLAS.

EJEMPLOS

La expresión regular "**\.**" concuerda con la cadena (simple) "**.**", pero no con una cadena de un solo carácter.

La expresión regular "**""**" concuerda con la cadena que contiene un solo carácter COMILLAS.

La expresión regular "**\)**" concuerda con la cadena "**)**".

La expresión regular "**\a**" concuerda con el carácter "**a**".

NOTA – El cuarto ejemplo muestra que se permite que la barra de fracción invertida preceda a los caracteres que no sean metacaracteres, pero sólo se desaconseja (puesto que en futuras versiones de esta Recomendación | Norma Internacional podrían permitirse otros metacaracteres).

A.2.9 Dos o más expresiones regulares pueden estar unidas por el operador infijo "**|**". La expresión regular resultante concuerda con cualquier cadena que concuerde con cualquiera de las subexpresiones.

A.2.10 Una expresión regular puede ir seguida de un operador de repetición. Si el operador es "**?**", el elemento precedente es facultativo y hay concordancia con él una vez como máximo. Si el operador es "*****", hay concordancia con el elemento precedente cero o más veces. Si el operador es "**+**", hay concordancia con el elemento precedente una o más veces. Si el operador es de la forma "**#(n)**", hay concordancia con el elemento precedente exactamente n veces; en este caso particular, se puede omitir el paréntesis si n consiste en una cifra. Si es de la forma "**#(n,)**", hay concordancia con el elemento n o más veces. Si es de la forma "**#(,m)**", el elemento es facultativo y hay concordancia con él m veces como máximo. Por último, si es de la forma "**#(n,m)**", hay concordancia con el elemento al menos n veces, pero no más de m veces.

NOTA – Es ilegal utilizar los metacaracteres "*****", "**+**", "**?**" o "**#**" como primer carácter de una expresión regular. También, es ilegal utilizar los metacaracteres "**#**" o "**|**"como último carácter de una expresión regular.

EJEMPLOS

Con un número telefónico tal como "**555-1212**" concuerda la expresión regular "**\d#3-d#4**", o de manera equivalente "**\d#(3)-\d#(4)**".

Con un precio en dólares tal como "**$12345.90**" concuerda la expresión regular "**$\d#(1,)(\.\d#(1,2))?**". Se señala que los paréntesis se necesitan después del símbolo "**#**" cuando va seguido de una gama.

Con un número de seguridad social tal como **"123-45-5678"** concuerda la expresión regular **"\d#3-?\d#2-?\d#4"**.

A.2.11 La repetición (véase A.2.10) tiene precedencia con respecto a la concatenación (véase A.1.3), que a su vez tiene precedencia con respecto a la alternancia (véase A.2.9). Se puede encerrar entre paréntesis una subexpresión completa para alterar estas reglas de precedencia.

A.2.12 Cuando una expresión regular contiene subexpresiones entre paréntesis, a cada paréntesis de apertura (sin comillas) se le asigna sucesivamente un entero distinto (estrictamente positivo) de la izquierda a la derecha de la expresión regular. A continuación, se puede hacer referencia a cada subexpresión dentro de un comentario con una notación tal como "*\1*", "*\2*", que utiliza el entero asociado. La subexpresión vacía "**()**" no se permite.

EJEMPLO

```
"((\d#2)(\d#2)(\d#4))" -- \1 es una fecha en la que \2 es el mes, \3 es el día
                       -- y \4 es el año.
```

NOTA – En muchos casos es preciso que haya una referencia formal a las subexpresiones de una expresión regular. Uno de tales casos sería la necesidad de escribir texto para documentar la expresión regular dentro del módulo ASN.1. Se trata de una notación que puede ser utilizada para proporcionar esa referencia. Esta notación no se emplea en ningún otro sitio de la presente Recomendación | Norma Internacional.

Anexo B

Reglas para la compatibilidad de tipos y de valores

(Este anexo es parte integrante de esta Recomendación | Norma Internacional)

El presente anexo se ha concebido principalmente para uso de los constructores de herramientas, y tiene por objeto garantizar que interpretan el lenguaje de forma idéntica. Se trata de especificar claramente qué ASN.1 son legales y cuáles no, y de poder especificar el valor exacto que cada nombre de referencia de valor identifica, y el conjunto exacto de valores que cada nombre de referencia de conjunto de tipos o de valores identifica. El objetivo no es proporcionar la definición de transformaciones válidas de notaciones ASN.1 para otros propósitos que no sean los mencionados más arriba.

B.1 Necesidad del concepto de correspondencia de valores (introducción didáctica)

B.1.1 Considérense las siguientes definiciones ASN.1:

```
A ::= INTEGER

B ::= [1] INTEGER

C ::= [2] INTEGER (0..6,...)

D ::= [2] INTEGER (0..6,...,7)

E ::= INTEGER (7..20)

F ::= INTEGER {red(0), white(1), blue(2), green(3), purple(4)}

a A ::= 3

b B ::= 4

c C ::= 5

d D ::= 6

e E ::= 7

f F ::= green
```

B.1.2 Es evidente que las referencias de valor **a**, **b**, **c**, **d**, **e** y **f** pueden utilizarse en una notación de valor gobernada por **A**, **B**, **C**, **D**, **E** y **F** respectivamente. Por ejemplo:

```
W ::= SEQUENCE {w1  A  DEFAULT a}
```

y:

```
x A ::= a
```

y:

```
Y ::= A(1..a)
```

son válidas teniendo en cuenta las definiciones de B.1.1. Sin embargo, si **A** fuera reemplazada por **B**, o **C**, o **D**, o **E** o **F**, ¿serían ilegales las sentencias resultantes? De igual forma, si la referencia de valor a fuera reemplazada en cada uno de estos casos por **b**, o **c**, o **d**, o **e** o **f**, ¿serían legales los enunciados resultantes?

B.1.3 Una cuestión más compleja sería considerar en cada caso la sustitución de la referencia de tipo por el texto explícito a la derecha de su asignación. Considérese por ejemplo:

```
f INTEGER {red(0), white(1), blue(2), green(3), purple(4)} ::= green

W ::= SEQUENCE {
      w1    INTEGER {red(0), white(1), blue(2), green(3),  purple(4)}
      DEFAULT f}

x INTEGER {red(0), white(1), blue(2), green(3),  purple(4)} ::= f

Y ::= INTEGER {red(0), white(1), blue(2), green(3),  purple(4)}(1..f)
```

¿Serían estas notaciones ASN.1 legales?

B.1.4 Algunos de los ejemplos anteriores, aun siendo legales (como lo son la mayoría de ellos – véase el texto a continuación), son casos en los que no puede aconsejarse a los usuarios que escriban textos similares, porque son, como mínimo, oscuros, y en el peor de los casos confusos. Sin embargo, con frecuencia se usa una referencia a un valor de algún tipo [no necesariamente un tipo **INTEGER** (ENTERO)] como valor por defecto para ese tipo con la rotulación o la subtipificación aplicada en el gobernador. El concepto de *correspondencia de valores* se ha introducido con el fin de proporcionar un medio claro y preciso de determinar qué construcciones – como las anteriores – son legales.

B.1.5 Considérese de nuevo:

```
C ::= [2] INTEGER (0..6,...)

E ::= INTEGER (7..20)

F ::= INTEGER {red(0), white(1), blue(2), green(3), purple(4)}
```

En cada caso se crea un nuevo tipo. Para **F** se puede identificar claramente una correspondencia biunívoca entre sus valores y los valores del tipo universal **INTEGER**. En el caso de **C** y **E** se puede identificar claramente una correspondencia biunívoca entre sus valores y un subconjunto de los valores de tipo universal **INTEGER**. Esta relación se denomina *correspondencia de valores* entre valores de los dos tipos. Además, puesto que entre los valores de **F**, **C** y **E** y los de **INTEGER** existe una correspondencia biunívoca, se puede utilizar esa relación para establecer la correspondencia entre los propios valores de **F**, **C** y **E**. Esto se ilustra en la figura B.1 para **F** y **C**.

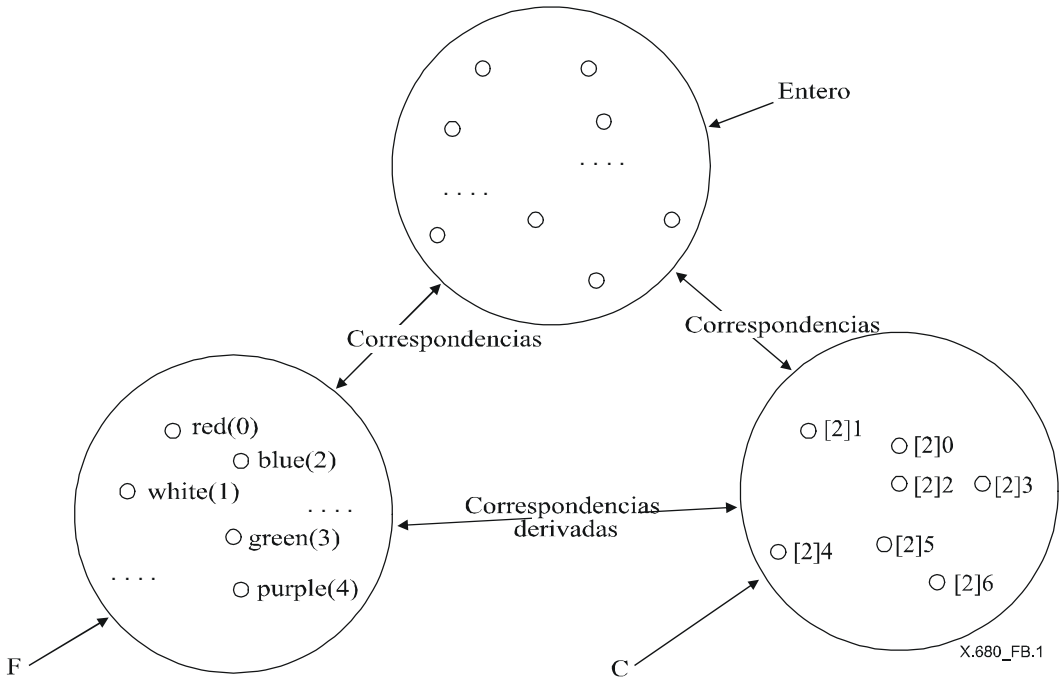

Figura B.1

B.1.6 Ahora, cuando se tiene una referencia de valor como:

```
c C ::= 5
```

para un valor de **C** requerido en algún contexto para identificar un valor de **F**, siempre que exista una correspondencia de valores entre dicho valor de **C** y un (único) valor de **F** se puede definir **c** como referencia legal al valor de **F**. Esto se ilustra en la figura B.2, en la que la referencia de valor **c** se utiliza para identificar un valor de **F**, y puede utilizarse en lugar de una referencia directa **f1** que de otro modo habría que definir:

```
f1 F ::= 5
```

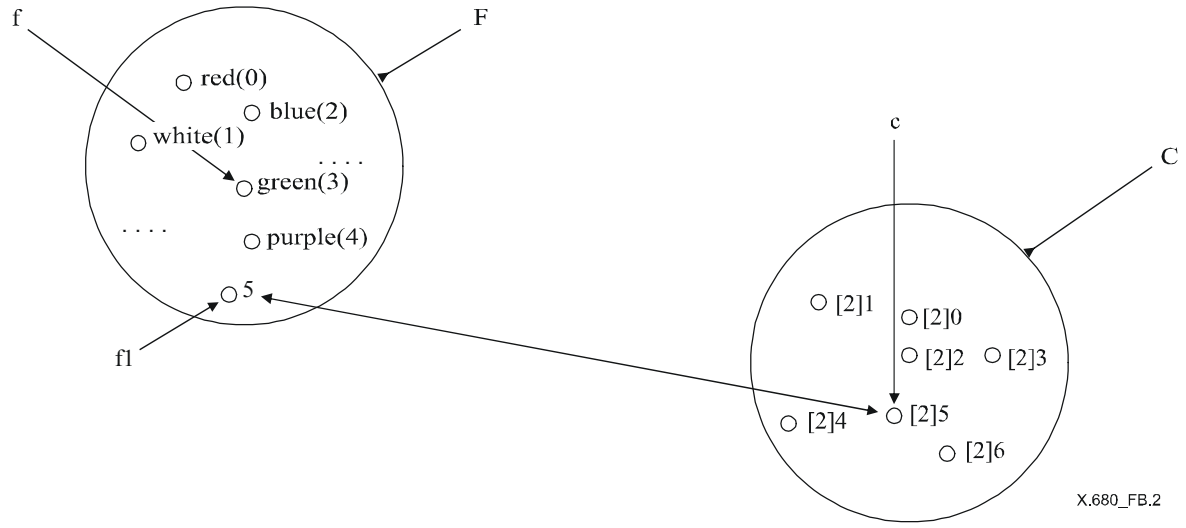

Figura B.2

B.1.7 Debe indicarse que en algunos casos habrá valores de un tipo (7 a 20 de **A** en B.1.1, por ejemplo) que tengan correspondencias con otros valores de otro tipo (7 a 20 de **E** en B.1.1, por ejemplo), pero habrá otros valores que no las tengan (de 21 en adelante, de **A**). Las referencias a tales valores de **A** no proporcionarían una referencia válida a valores de **E**. (En este ejemplo, todo el conjunto **E** tiene una correspondencia de valores con un subconjunto de **A**. En general, puede haber un subconjunto de valores de ambos tipos que tengan correspondencias, mientras que otros valores de ambos tipos no la tengan.)

B.1.8 En la parte principal de las normas ASN.1 se utiliza texto en inglés corriente para especificar la legalidad de los casos anteriores y casos similares. En la subcláusula B.6 se indican los requisitos exactos de legalidad, a los que habrá que remitirse siempre que existan dudas sobre una construcción compleja.

NOTA – El hecho de que se definan correspondencias entre dos casos de la construcción "Type" permite utilizar referencias de valor establecidas usando una construcción "Type" para identificar valores en otra construcción "Type" que se asemeje lo suficiente a la anterior. Ello permite tipificar parámetros ficticios y reales utilizando dos construcciones "Type" textualmente separadas sin quebrantar las reglas de compatibilidad de los parámetros ficticios y reales. Y permite además especificar campos de clases de objeto de información utilizando una construcción "Type", y especificar el valor correspondiente en un objeto de información utilizando una construcción "Type" distinta que se asemeje lo suficiente. (Estos ejemplos no pretenden ser exhaustivos.) No obstante, se recomienda que se aproveche esta libertad únicamente en casos simples como **SEQUENCE OF INTEGER**, o **CHOICE {int INTEGER, id OBJECT IDENTIFIER}**, y no para construcciones "Type" más complejas.

B.2 Correspondencias de valores

B.2.1 El modelo subyacente consta de tipos que, a modo de contenedores no solapados, contienen valores de forma que cada vez que aparece la construcción "Type" ASN.1 se define un nuevo tipo distinto (véanse las figuras B.1 y B.2). En el presente anexo se especifica cuándo existen *correspondencias de valores* entre esos tipos, permitiendo la utilización de una referencia a un valor de un tipo cuando se necesita una referencia a un valor en cualquier otro tipo.

EJEMPLO: Considérese:

```
X ::= INTEGER

Y ::= INTEGER
```

X e **Y** son nombres de referencia de tipo (punteros) de dos tipos diferentes, pero existen correspondencias de valores entre estos dos tipos, por lo que puede utilizarse cualquier referencia a un valor de **X** cuando gobierna **Y** (por ejemplo, después de **DEFAULT**).

B.2.2 En el conjunto de todos los valores ASN.1 posibles, una correspondencia pone en relación dos valores. El conjunto completo de correspondencias de valores es una relación matemática. Esta relación posee las siguientes propiedades: es reflexiva (cada valor ASN.1 está en relación consigo mismo), es simétrica (si se ha definido la existencia de una correspondencia entre un valor **x1** y un valor **x2**, también existe, de manera automática, una correspondencia entre el valor **x2** y el valor **x1**) y es transitiva (si existe una correspondencia entre el valor **x1** y el valor **x2**, y otra correspondencia entre el valor **x2** y el valor **x3**, existe automáticamente una correspondencia entre el valor **x1** y el valor **x3**).

B.2.3 Además, dados dos tipos **X1** y **X2**, considerados conjuntos de valores, el conjunto de correspondencias entre los valores de **X1** y los valores de **X2** es una relación de uno a uno, es decir, que para todos los valores **X1** de **X1** y **X2** de **X2**, si existe una correspondencia entre **X1** y **X2**:

a) no existe ninguna correspondencia entre **X1** y otro valor de **X2** diferente de **X2**; y

b) no existe ninguna correspondencia entre ningún valor de **X1** (distinto de **X1**) con **X2**.

B.2.4 Cuando existe una correspondencia entre un valor **X1** y un valor **X2**, la referencia de valor a cualquiera de los dos puede utilizarse automáticamente como referencia al otro si así lo requiere un tipo gobernante.

NOTA – El hecho de que se defina la existencia de correspondencias entre valores de algunas construcciones "Type" tiene únicamente por objeto flexibilizar la utilización de la notación ASN.1. La existencia de tales correspondencias no implica en absoluto que los dos tipos tengan la misma semántica de aplicación, pero se recomienda que las construcciones ASN.1 que pudieran ser ilegales sin correspondencias de valores se utilicen únicamente si los tipos correspondientes llevan realmente la misma semántica de aplicación. Se señala que con frecuencia existirán correspondencias de valores en cualquier especificación amplia entre dos tipos que sean construcciones ASN.1 idénticas, pero que lleven semánticas de aplicación completamente diferentes, y cuando la existencia de estas correspondencias de valores no se utilice nunca para determinar la legalidad de la especificación completa.

B.3 Definiciones de tipo idénticas

B.3.1 El concepto de definiciones de tipo idénticas se utiliza para hacer posible la definición de correspondencias de valores entre dos ejemplares de "Type" que sean idénticos o se asemejen lo suficiente como para que pueda esperarse razonablemente un uso intercambiable. Con el fin de precisar el significado de la expresión "se asemejen lo suficiente", en la presente subcláusula se especifica una serie de transformaciones que se aplican a cada uno de los ejemplares de "Type" para producir una *forma normal* de dichos ejemplares. Se dice que las definiciones de los dos ejemplares de "Type" son definiciones de tipo idénticas única y exclusivamente cuando sus formas normales sean listas ordenadas idénticas de los mismos elementos léxicos ASN.1 (véase la cláusula 11).

B.3.2 Cada ocurrencia de "Type" en una especificación ASN.1 es una lista ordenada de elementos léxicos definidos en la cláusula 11. La forma normal se obtiene aplicando las transformaciones definidas en B.3.2.1 a B.3.2.6, en ese orden.

B.3.2.1 Se eliminarán todos los comentarios (véase 11.6).

B.3.2.2 Las siguientes transformaciones no son recursivas y, por consiguiente, sólo es necesario aplicarlas una vez, en cualquier orden:

a) Para un tipo definido por una "ValueSetTypeAssignment", su definición se sustituye por una "TypeAssignment" utilizando el mismo "Type" y una constricción de subtipo que es el contenido del "ValueSet" que se especifica en 15.6.

b) Para cada tipo entero: "NamedNumberList" (véase 18.1), si existe, se reordena de forma que los "identifiers" estén en orden alfabético ("a" en primer lugar y "z" en último lugar).

c) Para cada tipo enumerado: se añaden números, tal como se indica en 19.3, a cualquier "EnumerationItem" (véase 19.1) que sea un "identifier" (sin número); a continuación, "RootEnumeration" se reordena de forma que los "identifiers" estén en orden alfabético ("a" en primer lugar, y "z" en último lugar).

d) Para cada tipo cadena de bits: "NamedBitList" (véase 21.1), si existe, se reordena de forma que los "identifiers" estén en orden alfabético ("a" en primer lugar, y "z" en último lugar).

e) Para cada valor identificador de objeto: cada "ObjectIdComponent" se transforma en sus "NumberForm" correspondientes de conformidad con la semántica de la cláusula 31 (véase el ejemplo de 31.12).

f) Para cada valor de identificador de objeto relativo (véase 32.3): cada "RelativeOIDComponents" se transforma en su correspondiente "NumberForm" de acuerdo con la semántica de la cláusula 32.

g) Para tipos secuencia (véase la cláusula 24) y tipos conjunto (véase la cláusula 26): cualquier extensión de la forma "ExtensionAndException", "ExtensionAdditions", se quita y coloca al final de "ComponentTypeLists"; se elimina "OptionalExtensionMarker", si está presente.

   Si "TagDefault" es **IMPLICIT TAGS**, se añade la palabra clave **IMPLICIT** a todos los ejemplares de "Tag" (véase la cláusula 30) a menos que:

   - ya esté presente; o
   - la palabra reservada **EXPLICIT** esté presente; o
   - el tipo que se esté rotulando sea un tipo **CHOICE**; o
   - se trate de un tipo abierto.

Si "TagDefault" es **AUTOMATIC TAGS**, la decisión de si se aplica o no la rotulación automática se adopta de conformidad con 24.2 (la rotulación automática se realizará más adelante).

NOTA – En las subcláusulas 24.3 y 26.2 se especifica que la presencia de un "Tag" en un "ComponentType" que haya sido insertado como resultado de la sustitución de "Components of Type" no impide en sí misma la transformación por rotulación automática.

Si "ExtensionDefault" es **EXTENSIBILITY IMPLIED**, se añaden puntos suspensivos ("**...**") después de "ComponentTypeLists", si no está presente.

h) Para tipos elección (véase la cláusula 28): "RootAlternativeTypeList" se reordena de forma que los "identifiers" de los "NameType" estén en orden alfabético ("a" en primer lugar, y "z" en último lugar). Si está presente "OptionalExtensionMarker", se suprime. Si "TagDefault" es **IMPLICIT TAGS**, se añade la palabra clave **IMPLICIT** a todos los ejemplares de "Tags" (véase la cláusula 30) a menos que:

- ya esté presente; o
- la palabra reservada **EXPLICIT** esté presente; o
- el tipo que se esté rotulando sea un tipo **CHOICE**; o
- se trate de un tipo abierto.

Si "TagDefault" es **AUTOMATIC TAGS**, la decisión de si se aplica o no la rotulación automática se adopta de conformidad con 28.5 (la rotulación automática se realizará más adelante). Si "ExtensionDefault" es **EXTENSIBILITY IMPLIED**, se añaden puntos suspensivos ("**...**") después de "AlternativeTypeLists", si no está presente.

B.3.2.3 Las siguientes transformaciones se aplicarán recursivamente en el orden indicado, hasta que se alcance un punto fijo:

a) Para cada valor de identificador de objeto (véase 31.3): si la definición de valor empieza con un "DefinedValue", éste se sustituye por su definición.

b) Para cada valor de identificador de objeto relativo (véase 32.3): si la definición de valor contiene "DefinedValue", éstos se sustituyen por su definición.

c) Para tipos secuencia y tipos conjunto: todos los ejemplares de "**COMPONENTS OF** Type" (véase la cláusula 24) se transforman de conformidad con las cláusulas 24 y 26.

d) Para tipos secuencia, tipos conjunto y tipos elección: si anteriormente se ha decidido realizar la rotulación automática [véanse los ítems g) y h) de B.3.2.2], ésta se aplicará de conformidad con las cláusulas 24, 26 y 28.

e) Para tipo elección: la construcción se sustituye por la alternativa seleccionada de conformidad con la cláusula 29.

f) Todas las referencias de tipos son sustituidas por sus definiciones, de acuerdo con las siguientes reglas:

- Si el tipo que se sustituye es una referencia al tipo que se está transformando, dicha referencia se sustituye por un elemento especial que se corresponde únicamente consigo mismo.
- Si el tipo que se sustituye es un tipo secuencia de o un tipo conjunto de, las constricciones que siguen al tipo sustituido, si existen, se desplazan frente a la palabra clave **OF**.
- Si el tipo sustituido es un tipo parametrizado o un conjunto de valores parametrizados (véase la Rec. UIT-T X.683 | ISO/CEI 8824-4, 8.2), cada "DummyReference" se sustituye por el "ActualParameter" correspondiente.

g) Todas las referencias de valores se sustituyen por sus definiciones; si el valor sustituido es un valor parametrizado (véase la Rec. UIT-T X.683 | ISO/CEI 8824-4, 8.2), cada "DummyReference" se sustituye por el "ActualParameter" correspondiente.

NOTA – Antes de sustituir cualquier referencia de valor, se aplicarán los procedimientos de este anexo para asegurar que la referencia de valor identifica, mediante correspondencias de valores o directamente, un valor en su tipo gobernante.

B.3.2.4 Para el tipo conjunto: "RootComponentTypeList" se reordena de forma que los "ComponentType" estén en orden alfabético ("a" en primer lugar, y "z" en último lugar).

B.3.2.5 Se aplicarán las siguientes transformaciones a las definiciones de valor:

a) Si se define un valor entero, ese identificador se sustituye por el número asociado.

b) Si se define un valor de cadena de bits utilizando identificadores, se sustituye por la "bstring" correspondiente suprimiendo todos los bits de cola.

c) Se suprimen todos los espacios en blanco inmediatamente antes y después de cada nueva línea (incluida la nueva línea) de una "cstring".

d) Se suprimen todos los espacios en blanco de "bstring" y "hstring".

e) Cada valor real definido con base 2 se normaliza de forma tal que la mantisa sea impar, y cada valor real definido con base 10 se normaliza de tal manera que el último dígito de la mantisa no sea 0.

f) Cada valor **GeneralizedTime** y **UTCTime** se sustituye por una cadena que sea conforme con las reglas utilizadas cuando se codifica en DER y CER (véase la Rec. UIT-T X.690 | ISO/CEI 8825-1, 11.7 y 11.8).

g) Tras haber aplicado el apartado c), cada valor **UTF8String**, **NumericString**, **PrintableString**, **IA5String**, **VisibleString** (**ISO646String**), **BMPString** y **UniversalString** se sustituye por el valor equivalente del tipo **UniversalString** escrito utilizando la notación "Quadruple" (véase la cláusula 37.8).

B.3.2.6 Toda ocurrencia de "realnumber" se transformará en un "SequenceValue" asociado de "base" 10. Toda ocurrencia del "RealValue" asociado con "SequenceValue" se transformará en el "SequenceValue" asociado de la misma "base", de forma tal que el último dígito de la mantisa no sea cero.

B.3.3 Si dos ejemplares de "Type", al ser transformados en su forma normal, se convierten en listas idénticas de elementos léxicos ASN.1 (véase la cláusula 11), se dice que las definiciones de esos dos ejemplares de "Type" son definiciones de tipo idénticas con la siguiente excepción: si una "objectclassreference" (véase la Rec. UIT-T X.681 | ISO/CEI 8824-2, 7.1), una "objectreference" (véase la Rec. UIT-T X.681 | ISO/CEI 8824-2, 7.2) o una "objectsetreference" (véase la Rec. UIT-T X.681 | ISO/CEI 8824-2, 7.3) aparece dentro de la forma normalizada del "Type", no se considera que las definiciones de los dos tipos sean idénticas, y no existirán correspondencias de valores (véase B.4) entre ellos.

NOTA – Se ha introducido esta excepción para evitar la necesidad de proporcionar reglas de transformación a forma normal en el caso de elementos de sintaxis que tengan que ver con notación de clase de objeto de información, de objeto de información y de conjunto de objetos de información. De igual forma, no se ha incluido en este momento una especificación de la normalización de toda la notación de valor y de la notación aritmética de conjuntos. Si dicha especificación se requiriera, podría proporcionarse en una versión futura de la presente Recomendación | Norma Internacional. Se han introducido los conceptos de definiciones de tipo idénticas y de correspondencia de valores con el fin de que puedan utilizarse construcciones ASN.1 sencillas con nombres de referencia o copiando texto. Se ha considerado innecesario proporcionar esta funcionalidad para ejemplares más complejos de "Type" que incluyan clases de objeto de información, etc.

B.4 Especificación de correspondencias de valores

B.4.1 Si dos ocurrencias de "Type" son definiciones de tipo idénticas según las reglas de B.3, existirán correspondencias entre cada valor de un tipo y el correspondiente valor del otro tipo.

B.4.2 Para un tipo **X1**, creado a partir de otro tipo, **X2**, por medio de rotulación (véase la cláusula 30), se define la existencia de correspondencia de valores entre todos los miembros de **X1** y los miembros correspondientes de **X2**.

NOTA – Si bien se define la existencia de correspondencias entre los valores de **X1** y **X2** en B.4.2 y entre los valores de **X3** y **X4** en B.4.3, cuando tales tipos estén incorporados en definiciones de tipo no idénticas, sino distintas (como **SEQUENCE** o **CHOICE**), las definiciones de tipo resultantes (los tipos **SEQUENCE** o **CHOICE**) no serán idénticas y no habrá correspondencias de valores entre ellas.

B.4.3 Para un tipo **X3**, creado seleccionando valores de cualquier tipo gobernante **X4** por medio de la construcción de conjunto de elementos o de la subtipificación, se define la existencia de correspondencias de valores entre sus miembros y los miembros del tipo gobernante que fueron seleccionados por la construcción del conjunto de elementos o por la subtipificación. La presencia o ausencia de un marcador de extensión no afecta a esta regla.

B.4.4 En B.5 se especifican correspondencias de valores adicionales entre algunos de los tipos de cadenas de caracteres.

B.4.5 Se define la existencia de una correspondencia entre todos los valores de cualquier tipo definido como entero con valores con nombre y cualquier tipo entero definido sin valores con nombre, o con valores con nombre diferentes, o con nombres diferentes para los valores con nombre, o todos ellos.

NOTA – La existencia de correspondencias de valores no afecta a ningún requisito de la regla de ámbito de aplicación sobre la utilización de los nombres de los valores con nombre. Sólo pueden utilizarse en un ámbito gobernado por el tipo en el que se han definido, o por un nombre de referencia a ese tipo.

B.4.6 Se define la existencia de una correspondencia entre todos los valores de cualquier tipo definido como tipo de cadena de bits con bits con nombre y de cualquier tipo de cadena de bits definido sin bits con nombre, o con diferentes bits con nombre, con nombres diferentes para los bits con nombre, o todos ellos.

NOTA – La existencia de correspondencias de valores no afecta a ningún requisito de la regla de ámbito de aplicación sobre la utilización de los nombres de los valores con nombre. Sólo pueden utilizarse en un ámbito gobernado por el tipo en el que se han definido, o por un nombre de referencia a ese tipo.

B.5 Correspondencias de valores adicionales definidas para los tipos cadenas de caracteres

B.5.1 Hay dos grupos de tipos cadena de caracteres restringidos, el grupo A (véase B.5.2) y el grupo B (véase B.5.3). Se define la existencia de correspondencias de valores entre todos los tipos del grupo A, y las referencias a los valores de cada tipo pueden utilizarse cuando éste está gobernado por cualquiera de los otros tipos. En cuanto a los tipos del grupo B, no existen correspondencias de valores entre ellos, ni entre los tipos del grupo A y los tipos del grupo B.

B.5.2 El grupo A consiste en:

```
UTF8String
NumericString
PrintableString
IA5String
VisibleString (ISO646String)
UniversalString
BMPString
```

B.5.3 El grupo B consiste en:

```
TeletexString (T61String)
VideotexString
GraphicString
GeneralString
```

B.5.4 Las correspondencias de valores del grupo A se especifican estableciendo las correspondencias entre los valores de cadena de caracteres de cada tipo y la **UniversalString** y aplicando después la propiedad transitiva de las correspondencias de valores. Para establecer la correspondencia entre valores de uno de los tipos del grupo A y la **UniversalString**, se sustituye la cadena por una **UniversalString** de la misma longitud, asociando cada carácter como se especifica a continuación.

B.5.5 Formalmente, el conjunto de valores abstractos de **UTF8String** es el mismo que el que tiene lugar en **UniversalString** pero con un rótulo diferente (véase 37.16), y cada valor abstracto de **UTF8String** se define en correspondencia con el valor abstracto de **UniversalString** asociado.

B.5.6 Los glifos (caracteres imprimibles) utilizados para formar los tipos **NumericString** y **PrintableString** tienen correspondencias reconocibles y sin ambigüedad con un subconjunto de los glifos asignados a los primeros 128 caracteres de ISO/CEI 10646-1. La correspondencia de estos tipos se define utilizando esta correspondencia de glifos.

B.5.7 La correspondencia entre **IA5String** y **VisibleString** y la **UniversalString** se establece asociando cada carácter con el carácter de la **UniversalString** que tenga un valor idéntico (32 bits) en la codificación BER de **UniversalString** como el valor (8 bits) de la codificación BER de **IA5String** y **VisibleString**.

B.5.8 **BMPString** es formalmente un subconjunto de **UniversalString**, y los valores abstractos correspondientes tienen correspondencias de valores.

B.6 Requisitos específicos de compatibilidad de tipo y de valor

En la presente subcláusula se utiliza el concepto de correspondencia de valores para hacer precisiones textuales respecto a la legalidad de ciertas construcciones ASN.1.

B.6.1 Toda ocurrencia de "Value", *x-notation* (*notación x*), con un tipo gobernante **Y** identifica el valor, *y-val* (*valor y*), de dicho tipo gobernante que tiene una correspondencia de valor con el valor *x-val* (*valor x*) especificado por la *x-notation*. Se requiere que dicho valor exista.

Considérese, por ejemplo, la **x** de la última línea de lo siguiente:

```
X   ::= [0] INTEGER (0..30)

x X ::= 29

Y   ::= [1] INTEGER (25..35)

Z1  ::= Y (x | 30)
```

Estas construcciones ASN.1 son legales, y en la última asignación la *x-notation* **x** hace referencia al *x-val* 29 de **X** y, a través de la correspondencia de valores, identifica el *y-val* 29 de **Y**. La *x-notation* 30 hace referencia al *y-val* 30 de **Y**, y **Z1** es el conjunto de valores 29 y 30. Por otra parte, la asignación:

```
Z2 ::= Y (x | 20)
```

es ilegal porque no existe un *y-val* al que pueda referirse la *x-notation* 20.

B.6.2 Siempre que aparezca "Type", *t-notation* (*notación t*), con un tipo gobernante **V** identifica al conjunto completo de valores en la raíz de este tipo gobernante que tiene correspondencia de valor con cualquiera de los valores en la raíz del "Type" *t-notation*. Este conjunto ha de contener al menos un valor.

Considérese por ejemplo, **W** de la última línea de lo siguiente:

```
V  ::= [0] INTEGER (0..30)

W  ::= [1] INTEGER (25..35)

Y  ::= [2] INTEGER (31..35)

Z1 ::= V (W | 24)
```

W aporta los valores 25-30 a la aritmética del conjunto, con el resultado de que **Z1**, tiene los valores 24-30. Por otra parte, la asignación:

```
Z2 ::= V (Y | 24)
```

es ilegal porque no hay valores en **Y** que se correspondan con los valores de **V**.

B.6.3 Se requiere que el tipo de cualquier valor proporcionado como parámetro real tenga una correspondencia de valor con uno de los valores del tipo que gobierna el parámetro ficticio, y el valor que se identifica pertenece a ese tipo gobernante.

B.6.4 Si se proporciona un "Type" como parámetro real para un parámetro ficticio de un conjunto de valores, todos los valores de ese "Type" habrán de tener correspondencia con los valores del gobernador del parámetro ficticio del conjunto de valores. El parámetro real selecciona todo el conjunto de valores del gobernador que tienen correspondencia con el "Type".

B.6.5 La especificación del tipo, **A**, de un parámetro ficticio de valor o de conjunto de valores es ilegal a menos que todos los valores de **A** y todos los ejemplares de utilización de **A** en la parte derecha de la asignación puedan aplicarse legalmente en lugar del parámetro ficticio.

B.7 Ejemplos

B.7.1 En la presente subcláusula se dan ejemplos para ilustrar B.3 y B.4.

B.7.2 Ejemplo 1

```
X ::= SEQUENCE
      {name VisibleString,
      age  INTEGER}

X2 ::= [8] SEQUENCE
      {name VisibleString,
      age  INTEGER}
```

```
X1 ::=  SEQUENCE
   {name VisibleString,
   -- comment --
         age  INTEGER}

X3 ::=  SEQUENCE
   {name VisibleString,
   age  AgeType}

AgeType ::= INTEGER
```

X, **X1**, **X2** y **X3** son definiciones de tipos idénticas. Las diferencias entre espacios en blanco y comentarios no son visibles, y la utilización de la referencia de tipo **AgeType** en **X3** no afecta a la definición de tipo. Se señala, sin embargo, que si se modificara cualquiera de los identificadores de los elementos de la secuencia, los tipos dejarían de ser definiciones idénticas, y no habría correspondencia de valores entre ellos.

B.7.3 Ejemplo 2

```
B ::= SET
      {name VisibleString,
      age  INTEGER}

B1 ::= SET
       {age INTEGER,
       name VisibleString}
```

estas definiciones son definiciones de tipo idénticas, con tal de que ninguna de ellas esté en un módulo con **AUTOMATIC TAGS** en su encabezamiento; de no ser así, no serán definiciones de tipo idénticas, y no existirá correspondencia de valores entre ellas. Pueden obtenerse ejemplos similares utilizando **CHOICE** y **ENUMERATED** (con la forma "identifier" de "EnumerationItem").

B.7.4 Ejemplo 3

```
C ::= SET
      {name [0]VisibleString,
      age INTEGER}

C1 ::= SET
       {name VisibleString,
       age INTEGER (1..64)}
```

no son definiciones de tipo idénticas entre sí, ni son idénticas tampoco a las de **B** o **B1**, y no hay correspondencia entre los valores de **C** y de **C1**, ni entre los valores de ninguno de ellos y los valores de **B** o **B1**.

B.7.5 Ejemplo 4

```
x  INTEGER { y (2) } ::= 3
z  INTEGER ::= x
```

es legal, y asigna el valor 3 a **z** según la correspondencia de valores definida en B.4.5.

B.7.6 Ejemplo 5

```
b1 BIT STRING ::= '101'B
b2 BIT STRING {version1(0), version2(1), version3(2)} ::= b1
```

es legal, y asigna el valor **{version1, version3}** a **b2**.

B.7.7 Ejemplo 6

Con las definiciones de B.1.1, los elementos de **SEQUENCE** de la forma:

```
X DEFAULT y
```

son legales cuando **X** es **A**, **B**, **C**, **D**, **E**, o **F**, o cualquiera de los textos a la derecha de las asignaciones de tipo a estos nombres, **e** y es cualquiera de **a**, **b**, **c**, **d**, **e**, o **f**, con las siguientes excepciones: **E DEFAULT** y es ilegal para **a**, **b**, **c**, **d**, **f**, y **C DEFAULT e** es ilegal, porque en estos casos no existe correspondencia de valores entre la referencia del valor que falta y el tipo al que se suple.

Anexo C

Valores de identificador de objetos asignado

(Este anexo es parte integrante de esta Recomendación | Norma Internacional)

En este anexo se registran los valores de los identificadores de objetos y de los descriptores de objetos asignados en la serie de Recomendaciones | Normas Internacionales sobre ASN.1, y ofrece un módulo ASN.1 que puede utilizarse para referenciar esos valores de identificadores de objetos.

C.1 Identificadores de objetos asignados en esta Recomendación | Norma Internacional

En la presente Recomendación | Norma Internacional se asignan los siguientes valores:

Subcláusula 37.3

Object Identifier Value:
{ joint-iso-itu-t asn1(1) specification(0) characterStrings(1) numericString(0) }

Object Descriptor Value: **"NumericString ASN.1 type"**

Subcláusula 37.5

Object Identifier Value:
{ joint-iso-itu-t asn1(1) specification(0) characterStrings(1) printableString(1) }

Object Descriptor Value: **"PrintableString ASN.1 type"**

Subcláusula 38.1

Object Identifier Value:
{ joint-iso-itu-t asn1(1) specification(0) modules(0) iso10646(0) }

Object Descriptor Value: **"ASN.1 Character Module"**

Subcláusula C.2

Object Identifier Value:
{ joint-iso-itu-t asn1(1) specification(0) modules(0) object-identifiers(1) }

Object Descriptor Value: **"ASN.1 Object Identifier Module"**

C.2 Identificadores de objetos en las normas sobre ASN.1 y reglas de codificación

En esta cláusula se especifica un módulo ASN.1 que contiene la definición de un nombre de referencia para cada valor de identificador de objetos en las normas sobre ASN.1 (Rec. UIT-T X.680 | ISO/CEI 8824-1 a Rec. UIT-T X.693 | ISO/CEI 8825-4).

NOTA – Estos valores pueden utilizarse en la notación de valor del tipo OBJECT IDENTIFIER y de los tipos que se deducen a partir del mismo. Todas las referencias de valor definidas en el módulo que se especifica en esta cláusula son exportadas y tienen que ser importadas por cualquier módulo que necesite utilizarlas.

```
ASN1-Object-Identifier-Module { joint-iso-itu-t asn1(1) specification(0) modules(0)
object-identifiers(1) }
      DEFINITIONS ::= BEGIN

      -- NumericString ASN.1 type (see 37.3) --
      numericString OBJECT IDENTIFIER ::=
            { joint-iso-itu-t asn1(1) specification(0) characterStrings(1)
      numericString(0) }

      -- PrintableString ASN.1 type (see 37.5) --
      printableString OBJECT IDENTIFIER ::=
            { joint-iso-itu-t asn1(1) specification(0) characterStrings(1)
      printableString(1) }

      -- ASN.1 Character Module (see 38.1) --
      asn1CharacterModule OBJECT IDENTIFIER ::=
            { joint-iso-itu-t asn1(1) specification(0) modules(0) iso10646(0) }
```

```
        -- ASN.1 Object Identifier Module (this module) --
        asn1ObjectIdentifierModule OBJECT IDENTIFIER ::=
              { joint-iso-itu-t asn1(1) specification(0) modules(0)
        object-identifiers(1) }

        -- BER encoding of a single ASN.1 type --
        ber OBJECT IDENTIFIER ::=
              { joint-iso-itu-t asn1(1) basic-encoding(1) }

        -- CER encoding of a single ASN.1 type --
        cer OBJECT IDENTIFIER ::=
              { joint-iso-itu-t asn1(1) ber-derived(2) canonical-encoding(0) }

        -- DER encoding of a single ASN.1 type --
        der OBJECT IDENTIFIER ::=
              { joint-iso-itu-t asn1(1) ber-derived(2) distinguished-encoding(1) }

        -- PER encoding of a single ASN.1 type (basic aligned) --
        perBasicAligned OBJECT IDENTIFIER ::=
              { joint-iso-itu-t asn1(1) packed-encoding(3) basic(0) aligned(0) }

        -- PER encoding of a single ASN.1 type (basic unaligned) --
        perBasicUnaligned OBJECT IDENTIFIER ::=
              { joint-iso-itu-t asn1(1) packed-encoding(3) basic(0) unaligned(1) }

        -- PER encoding of a single ASN.1 type (canonical aligned) --
        perCanonicalAligned OBJECT IDENTIFIER ::=
              { joint-iso-itu-t asn1(1) packed-encoding(3) canonical(1) aligned(0) }

        -- PER encoding of a single ASN.1 type (canonical unaligned) --
        perCanonicalUnaligned OBJECT IDENTIFIER ::=
              { joint-iso-itu-t asn1(1) packed-encoding(3) canonical(1) unaligned(1)
        }

        -- XER encoding of a single ASN.1 type (basic) --
        xerBasic OBJECT IDENTIFIER ::=
              {joint-iso-itu-t asn1(1) xml-encoding(5) basic(0) }

        -- XER encoding of a single ASN.1 type (canonical) --
        xerCanonical OBJECT IDENTIFIER ::=
              {joint-iso-itu-t asn1(1) xml-encoding(5) canonical(1) }

    END -- ASN1-Object-Identifier-Module --
```

Anexo D

Asignación de valores componentes de identificadores de objetos

(Este anexo no es parte integrante de esta Recomendación | Norma Internacional)

En este anexo se describen los arcos de nivel superior del árbol de registro de los identificadores de objetos. No se proporciona explicación alguna sobre la manera en la que se añaden nuevos arcos, ni sobre las reglas que habrán de observar las autoridades en materia de registro. Las explicaciones correspondientes se especifican en la Rec. UIT-T X.660 | ISO/CEI 9834-1.

D.1 Asignación raíz de valores de componentes de identificadores de objetos

D.1.1 A partir del nodo raíz se especifican tres arcos. La asignación de valores e identificadores , y la autoridad de asignación de valores componentes subsiguientes, son:

Valor	*Identificador*	*Autoridad encargada de las asignaciones subsiguientes*
`0`	**`itu-t`**	UIT-T (Véase D.2)
`1`	**`iso`**	UIT-T (Véase D.3)
`2`	**`joint-iso-itu-t`**	Véase D.4

D.1.2 Los identificadores **`itu-t`**, **`iso`** y **`joint-iso-itu-t`**, asignados en el cuadro anterior, pueden utilizarse, cada uno de ellos, como una "NameForm" (véase 31.3).

D.1.3 Los identificadores **`ccitt`** y **`joint-iso-ccitt`** son sinónimos de **`itu-t`** y **`joint-iso-itu-t`**, respectivamente, y pueden aparecer en valores de identificadores de objetos.

D.2 Asignación del UIT-T de valores de componentes de identificadores de objetos

D.2.1 A partir del nodo identificado por **`itu-t`** se especifican cinco arcos. La asignación de valores e identificadores es:

Valor	*Identificador*	*Autoridad encargada de las asignaciones subsiguientes*
`0`	**`recommendation`**	Véase D.2.2
`1`	**`question`**	Véase D.2.3
`2`	**`administration`**	Véase D.2.4
`3`	**`network-operator`**	Véase D.2.5
`4`	**`identified-organization`**	Véase D.2.6

Estos identificadores pueden utilizarse como una "NameForm" (véase 31.3).

D.2.2 Los arcos indicados por debajo de **`recommendation`** pueden tener los valores **`1`** a **`26`** con identificadores asignados de **`a`** a **`z`**. Los arcos por debajo de éstos tienen los números de las Recomendaciones UIT-T (y CCITT) que corresponden a las series identificadas por la letra. Los arcos por debajo de éstos se determinan mediante las Recomendaciones UIT-T (y CCITT), según proceda. Los identificadores de **`a`** a **`z`** pueden utilizarse como una "NameForm".

D.2.3 Los arcos por debajo de **`question`** tienen los valores que corresponden a las Comisiones de Estudio del UIT-T, calificados por el periodo de estudios. El valor se calcula mediante la fórmula:

Número de la Comisión de Estudio + (periodo * 32)

donde "periodo" tiene el valor 0 para 1984-1988, 1 para 1988-1992, etc., y el multiplicador es el 32 decimal.

Los arcos por debajo de cada Comisión de Estudio tienen los valores que corresponden a las Cuestiones asignadas a esa Comisión de Estudio. Los arcos por debajo de éstos son determinados por el Grupo (por ejemplo, el Grupo de Trabajo o el Grupo de Relator Especial) asignado para que estudie la Cuestión.

D.2.4 Los arcos por debajo de **`administration`** tienen los valores de los DCC X.121. Los arcos por debajo de éstos son determinados, según proceda, por la Administración del país identificado por el DCC X.121.

D.2.5 Los arcos por debajo de **`network-operator`** tienen el valor de los DNIC X.121. Los arcos por debajo de éstos son determinados, según proceda, por la Administración o la EER identificada por el DNIC.

D.2.6 Los arcos por debajo de **`identified-organization`** son valores asignados por la Oficina de Normalización de las Telecomunicaciones de la UIT (TSB). Los arcos por debajo de éstos son determinados, según proceda, por las organizaciones identificadas.

NOTA – Entre las organizaciones para las que puede ser útil este arco se encuentran:

- las empresas de explotación reconocidas que no se encargan de la explotación de una red pública de datos;
- las organizaciones científicas e industriales;
- las organizaciones de normalización regionales; y
- las organizaciones multinacionales.

D.3 Asignación de ISO de valores de componentes de identificadores de objetos

D.3.1 A partir del nodo identificado como **`iso(1)`** se especifican tres arcos. La asignación de valores e identificadores es:

Valor	*Identificador*	*Autoridad encargada de las asignaciones subsiguientes*
`0`	**`standard`**	Véase D.3.2
`2`	**`member-body`**	Véase D.3.3
`3`	**`identified-organization`**	Véase D.3.4

Estos identificadores pueden utilizarse como una "NameForm".

NOTA – La utilización del arco **`registration-authority(1)`** ha sido suprimida.

D.3.2 Los arcos por debajo de **`standard`** tendrán, cada uno, el valor del número de una Norma Internacional. Cuando la Norma Internacional esté constituida por varias partes, habrá un arco adicional para el número de parte, a menos de que éste se excluya específicamente en el texto de la Norma Internacional. Los arcos adicionales tendrán los valores definidos en esa Norma Internacional.

D.3.3 Los arcos justo por debajo de **`member-body`** tendrán valores de un indicativo de país numérico con tres dígitos, como se especifica en ISO 3166, que permiten identificar el organismo nacional de ISO en ese país. Con estos identificadores no se permite la "NameForm" del componente identificador de objeto.

D.3.4 Los arcos justo por debajo de **`identified-organization`** tendrán los valores de un designador de indicativo internacional (ICD, *international code designator*) atribuido por la autoridad de registro de ISO/CEI 6523 que permite identificar a una organización emisora registrada específicamente por esa autoridad para atribuir componentes identificadores de objetos. Los arcos justo por debajo del ICD tendrán valores de un "indicativo de organización" atribuido por la organización emisora de conformidad con ISO/CEI 6523.

D.4 Asignación conjunta de valores de componentes de identificadores de objetos

D.4.1 Los arcos por debajo de **`joint-iso-itu-t`** tienen valores que son asignados y aprobados cada cierto tiempo por una autoridad de registro establecida por ISO/CEI y UIT-T para identificar ámbitos de actividad de normalización conjunta de ISO/CEI | UIT-T, de conformidad con la Rec. UIT-T X.662 | ISO/CEI 9834-3.

Anexo E

Ejemplos y sugerencias

(Este anexo no es parte integrante de esta Recomendación | Norma Internacional)

Este anexo contiene ejemplos de la utilización de ASN.1 en la descripción de estructuras de datos (hipotéticas). Contiene también sugerencias, o directrices, para el uso de diversas prestaciones de ASN.1. Salvo que se indique otra cosa, se supone un entorno de **AUTOMATIC TAGS**.

E.1 Ejemplo de un registro de personal

La utilización de ASN.1 se ilustra por medio de un registro de personal hipotético simple.

E.1.1 Descripción informal de registro de personal

A continuación se presenta la estructura del registro de personal y su valor para un determinado individuo.

Name: John P Smith

Title: Director

Employee Number: 51

Date of Hire: 17 September 1971

Name of Spouse: Mary T Smith

Number of Children: 2

Child Information

Name: Ralph T Smith

Date of Birth 11 November 1957

Child Information

Name: Susan B Jones

Date of Birth 17 July 1959

E.1.2 Descripción ASN.1 de la estructura de registro

La estructura de cada registro de personal se describe formalmente a continuación utilizando la notación normalizada para tipos de datos.

```
PersonnelRecord ::= [APPLICATION 0] SET
{     name              Name,
      title             VisibleString,
      number            EmployeeNumber,
      dateOfHire        Date,
      nameOfSpouse      Name,
      children          SEQUENCE OF ChildInformation DEFAULT {}
}

ChildInformation ::= SET
{     name              Name,
      dateOfBirth       Date
}

Name ::= [APPLICATION 1] SEQUENCE
{     givenName         VisibleString,
      initial           VisibleString,
      familyName        VisibleString
}

EmployeeNumber ::= [APPLICATION 2] INTEGER

Date ::= [APPLICATION 3] VisibleString  --  YYYY MMDD
```

Este ejemplo ilustra un aspecto de la descomposición analítica de la sintaxis ASN.1. La construcción sintáctica **DEFAULT** sólo puede aplicarse a un componente de una **SEQUENCE** o un **SET**, no puede aplicarse a un elemento de una **SEQUENCE OF**. Así, el **DEFAULT {}** en **PersonnelRecord** se aplica a **children**, no a **ChildInformation**.

E.1.3 Descripción ASN.1 de un valor de registro

El valor del registro de personal de John Smith se describe formalmente a continuación utilizando la notación normalizada para valores de datos.

```
{     name             {givenName "John", initial "P", familyName "Smith"},
      title            "Director",
      number           51,
      dateOfHire       "19710917",
      nameOfSpouse     {givenName "Mary", initial "T", familyName "Smith"},
      children
      {  {name  {givenName "Ralph", initial "T", familyName "Smith"} ,
          dateOfBirth "19571111"},
         {name  {givenName "Susan", initial "B", familyName "Jones"} ,
          dateOfBirth "19590717" }
      }
}
```

o en notación de valor XML:

```
person ::=
      <PersonnelRecord>
         <name>
            <givenName>John</givenName>
            <initial>P</initial>
            <familyName>Smith</familyName>
         </name>
         <title>Director</title>
         <number>51</number>
         <dateOfHire>19710917</dateOfHire>
         <nameOfSpouse>
            <givenName>Mary</givenName>
            <initial>T</initial>
            <familyName>Smith</familyName>
         </nameOfSpouse>
         <children>
            <ChildInformation>
               <name>
                  <givenName>Ralph</givenName>
                  <initial>T</initial>
                  <familyName>Smith</familyName>
               </name>
               <dateOfBirth>19571111</dateOfBirth>
            </ChildInformation>
            <ChildInformation>
               <name>
                  <givenName>Susan</givenName>
                  <initial>B</initial>
                  <familyName>Jones</familyName>
               </name>
               <dateOfBirth>19590717</dateOfBirth>
            </ChildInformation>
         </children>
      </PersonnelRecord>
```

E.2 Directrices para la utilización de la notación

Los tipos de datos y la notación formal definidos en la presente Recomendación | Norma Internacional son flexibles y su utilización permite diseñar una amplia gama de protocolos. Esta flexibilidad, sin embargo, puede confundir a veces, especialmente cuando la notación se utiliza por primera vez. Este anexo procura minimizar la confusión dando directrices y ejemplos para el empleo de la notación. Se ofrecen una o más directrices de uso para cada uno de los tipos de datos incorporados. Los tipos cadena de caracteres (por ejemplo **VisibleString**) y los tipos definidos en las cláusulas 42 a 44 no se tratan aquí.

E.2.1 Booleano

E.2.1.1 Se utilizará un tipo booleano para modelar los valores de una variable lógica (es decir, de dos estados), por ejemplo la respuesta a una pregunta del tipo sí o no.

EJEMPLO

```
Employed ::= BOOLEAN
```

E.2.1.2 Al asignar un nombre de referencia a un tipo booleano se elegirá uno que describa el estado *true* (verdadero).

EJEMPLO

```
Married ::= BOOLEAN
```

no

```
MaritalStatus ::= BOOLEAN
```

E.2.2 Entero

E.2.2.1 Se utilizará un tipo entero para modelar los valores (para todos los fines prácticos, sin limitación de magnitud) de una variable cardinal o entera.

EJEMPLO

```
CheckingAccountBalance ::= INTEGER --  in cents; negative means
                                       overdrawn.

balance CheckingAccountBalance ::= 0
```

o utilizando notación de valor XML:

```
balance ::= <CheckingAccountBalance>0</CheckingAccountBalance>
```

E.2.2.2 Se definirán los valores máximo y mínimo permitidos de un tipo entero como valores distinguidos.

EJEMPLO

```
DayOfTheMonth ::= INTEGER {first(1), last(31)}

today DayOfTheMonth ::= first

unknown DayOfTheMonth ::= 0
```

o utilizando notación de valor XML:

```
today ::= <DayOfTheMonth><first/></DayOfTheMonth>

unknown ::= <DayOfTheMonth>0</DayOfTheMonth>
```

Se señala que los números denominados **primero** y **último** se eligieron debido a su significado semántico para el lector y no excluyen la posibilidad de que **DayOfTheMonth** tenga otros valores que pueden ser menores que 1, superiores a 31 o comprendidos entre 1 y 31.

Para restringir el valor de **DayOfTheMonth** a **primero** y **último** simplemente, habría que escribir:

```
DayOfTheMonth ::= INTEGER {first(1), last(31)} (first | last)
```

y para restringir el valor de **DayOfTheMonth** a todos los valores entre 1 y 31, inclusive, habría que escribir:

```
DayOfTheMonth ::= INTEGER {first(1), last(31)} (first .. last)

dayOfTheMonth DayOfTheMonth ::= 4
```

o utilizando notación de valor XML:

```
dayOfTheMonth ::= <DayOfTheMonth>4</DayOfTheMonth>
```

E.2.3 Enumerado

E.2.3.1 Se utilizará un tipo enumerado para modelar los valores de una variable con tres o más estados. Se asignarán valores a partir de cero si la única restricción es la distinción.

EJEMPLO

```
DayOfTheWeek ::= ENUMERATED {sunday(0), monday(1), tuesday(2),
                             wednesday(3), thursday(4), friday(5),
                             saturday(6)}

firstDay DayOfTheWeek ::= sunday
```

o utilizando notación de valor XML:

```
firstDay ::= <DayOfTheWeek><sunday/></DayOfTheWeek>
```

Se señala que, si bien las enumeraciones **sunday**, **monday**, etc., se eligieron debido a su significado semántico para el lector, **DayOfTheWeek** está limitado a suponer uno de estos valores y no otro. Además, sólo el nombre **sunday**, **monday**, etc., puede ser asignado a un valor; no se permiten los valores enteros equivalentes.

E.2.3.2 Se utilizará un tipo enumerado extensible para modelar los valores de una variable que tenga en ese momento dos estados, pero que pueda tener más estados en una futura versión del protocolo.

EJEMPLO

```
MaritalStatus ::= ENUMERATED {single, married}
          -- First version of MaritalStatus
```

en anticipación de:

```
MaritalStatus ::= ENUMERATED {single, married, …, widowed}
          -- Second version of MaritalStatus
```

y aún posteriormente:

```
MaritalStatus ::= ENUMERATED {single, married, …, widowed, divorced}
          -- Third version of MaritalStatus
```

E.2.4 Real

E.2.4.1 Se utilizará el tipo real para modelar un número aproximado.

EJEMPLO

```
AngleInRadians ::=  REAL

pi REAL ::= {mantissa 31415926535897932384626433832 79, base 10, exponent –30}
```

o utilizando la notación de valor alternativa para **REAL**:

```
pi   REAL ::= 3.14159265358979323846264338327
```

o utilizando la notación de valor XML:

```
pi ::=
      <REAL>
            3.14159265358979323846264338327
      </REAL>
```

E.2.4.2 Los diseñadores de aplicaciones quizás deseen garantizar el pleno interfuncionamiento con valores reales a pesar de las diferencias entre equipos físicos de coma flotante y entre decisiones relativas a la implementación respecto a la utilización, por ejemplo, de coma flotante de longitud simple o longitud doble para una aplicación. Esto puede conseguirse como se indica a continuación:

```
App-X-Real ::= REAL (WITH COMPONENTS {
                       mantissa (–16777215..16777215),
                       base (2),
                       exponent (–125..128) } )
/*
  Senders shall not transmit values outside these ranges
  and conforming receivers shall be capable of receiving
  and processing all values in these ranges.
*/

girth App-X-Real ::= {mantissa 16, base 2, exponent 1}
```

o utilizando la notación de valor XML:

```
girth ::=
      <App-X-Real>
         32
      </App-X-Real>
```

E.2.5 Cadena de bits

E.2.5.1 Se utilizará el tipo de cadena de bits para modelar datos binarios cuyo formato y longitud no estén especificados, o lo esté en alguna otra parte, y cuya longitud en bits no sea necesariamente múltiplo de ocho.

EJEMPLO

```
G3FacsimilePage ::= BIT STRING
--  a sequence of bits conforming to ITU-T Rec. T.4.

image G3FacsimilePage ::= '100110100100001110110'B

trailer BIT STRING ::= '0123456789ABCDEF'H

body1 G3FacsimilePage ::= '1101'B

body2 G3FacsimilePage ::= '1101000'B
```

o utilizando la notación de valor XML:

```
image ::= <G3FacSimile>100110100100001110110</G3FacSimile>

trailer ::=
            <BIT_STRING>
                  0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011
                  1100 1101 1110 1111
            </BIT_STRING>

body1 ::= <G3FacSimile>1101</G3FacSimile>

body2 ::= <G3FacSimile>1101000</G3FacSimile>
```

Obsérvese que **body1** y **body2** son valores abstractos distintos, porque los bits 0 finales son significativos (ya que no hay "NamedBitList" en la definición de **G3FacsimilePage**).

E.2.5.2 Se utilizará el tipo cadena de bits con una limitación de tamaño para modelar los valores de un campo de bits de tamaño fijo.

EJEMPLO

```
BitField ::= BIT STRING (SIZE (12))

map1 BitField ::= '100110100100'B

map2 BitField ::= '9A4'H

map3 BitField ::= '1001101001'B    --  Illegal - violates size constraint.
```

o utilizando la notación de valor XML:

```
map1 ::= <BitField>100110100100</BitField>
```

Obsérvese que **map1** y **map2** son el mismo valor abstracto, porque los cuatro bits finales de **map2** no son significativos.

E.2.5.3 Se utilizará el tipo cadena de bits para modelar los valores de un **bit map** (mapa de bits), colección ordenada de variables lógicas que indican si una codificación particular se cumple para cada colección de objetos correspondientemente ordenada.

```
DaysOfTheWeek ::= BIT STRING {
            sunday(0), monday (1), tuesday(2),
            wednesday(3), thursday(4), friday(5),
            saturday(6) } (SIZE (0..7))

sunnyDaysLastWeek1 DaysOfTheWeek ::= {sunday, monday, wednesday}
      sunnyDaysLastWeek2 DaysOfTheWeek ::= '1101'B
      sunnyDaysLastWeek3 DaysOfTheWeek ::= '1101000'B

sunnyDaysLastWeek4 DaysOfTheWeek ::= '11010000'B -- Illegal
```

o utilizando notación de valor XML:

```
sunnyDaysLastWeek1 ::=
      <DaysOfTheWeek>
          <sunday/><monday/><wednesday/>
      </DaysOfTheWeek>

sunnyDaysLastWeek2 ::= <DaysOfTheWeek>1101</DaysOfTheWeek>

sunnyDaysLastWeek3 ::= <DaysOfTheWeek>1101000</DaysOfTheWeek>
```

Obsérvese que si el valor de la cadena de bits tiene una longitud inferior a 7 bits, los bits que faltan indican día nublado para esos días, por lo que los tres primeros valores anteriores tienen el mismo valor abstracto.

E.2.5.4 Se utilizará el tipo cadena de bits para modelar los valores de un *bit map*, colección ordenada de tamaño fijo de variables lógicas que indican si una condición particular se cumple para cada colección de objetos correspondientemente ordenada.

```
DaysOfTheWeek ::= BIT STRING {
            sunday(0), monday (1), tuesday(2),
            wednesday(3), thursday(4), friday(5),
            saturday(6) } (SIZE (7))
sunnyDaysLastWeek1 DaysOfTheWeek ::= {sunday, monday, wednesday}
sunnyDaysLastWeek2 DaysOfTheWeek ::= '1101'B -- Illegal
                                  -- violates size constraint.
sunnyDaysLastWeek3 DaysOfTheWeek ::= '1101000'B

sunnyDaysLastWeek4 DaysOfTheWeek ::= '11010000'B --  Illegal
                                       -- violates size constraint.
```

Obsérvese que los valores primero y tercero tienen el mismo valor abstracto.

E.2.5.5 Se utilizará el tipo cadena de bits con nombre para modelar los valores de una colección de variables lógicas relacionadas.

EJEMPLO

```
PersonalStatus ::= BIT STRING
            {married(0), employed(1), veteran(2), collegeGraduate(3)}

billClinton PersonalStatus ::= {married, employed, collegeGraduate}

hillaryClinton PersonalStatus ::= '110100'B
```

o utilizando notación de valor XML:

```
billClinton ::=
      <PersonalStatus>
         <married/>
         <employed/>
         <collegeGraduate/>
      </PersonalStatus>

hillaryClinton ::= <PersonalStatus>110100</PersonalStatus>
```

Obsérvese que **billClinton** y **hillaryClinton** tienen los mismos valores abstractos.

E.2.6 Cadena de octetos

E.2.6.1 Se utilizará el tipo cadena de octetos para modelar datos binarios cuyo formato y longitud no estén especificados, o están especificados en alguna otra parte, y cuya longitud en bits sea múltiplo de ocho.

EJEMPLO

```
G4FacsimileImage ::= OCTET STRING
--  a sequence of octets conforming to ITU-T Rec. T.5 and CCITT Rec. T.6

image G4FacsimilePage ::= '3FE2EBAD471005'H
```

o utilizando notación de valor XML:

```
image ::= <G4FacSimileImage>3FE2EBAD471005</G4FacSimileImage>
```

E.2.6.2 Se utilizará el tipo cadena de caracteres restringida con preferencia a un tipo cadena de octetos cuando se disponga de uno apropiado.

EJEMPLO

```
Surname ::= PrintableString

president Surname ::= "Clinton"
```

o utilizando notación de valor XML:

```
president ::= <Surname>Clinton</Surname>
```

E.2.7 UniversalString, BMPString y UTF8String

Se utilizará el tipo **BMPString** o el tipo **UTF8String** para modelar cualquier cadena de información que conste únicamente de caracteres procedentes del plano plurilingüe básico (BMP, *basic multilingual plane*) de ISO/CEI 10646-1, y **UniversalString** o **UTF8String** para modelar cualquier cadena que conste de caracteres de ISO/CEI 10646-1 no limitados al BMP.

E.2.7.1 Se utilizará **Level1** o **Level2** para denotar que el nivel de implementación impone restricciones a la utilización de caracteres combinantes.

EJEMPLO

```
RussianName ::= Cyrillic (Level1)
      -- RussianName uses no combining characters.

SaudiName ::= BasicArabic (SIZE (1..100) ^ Level2)
      --  SaudiName uses a subset of combining characters.
```

Representación de la letra Σ:

```
greekCapitalLetterSigma BMPString ::= {0, 0, 3, 163}
```

o utilizando la notación de valor XML:

```
greekCapitalLetterSigma ::= <BMPString>&#x03a3;</BMPString>
```

Representación de la cadena "f → ∝":

```
rightwardsArrow UTF8String ::= {0, 0, 33, 146}

infinity UTF8String ::= {0, 0, 34, 30}

property UTF8String ::= {"f  ", rightwardsArrow, " ", infinity}
```

o utilizando la notación de valor XML:

```
property ::= <UTF8String>f &#x2192; &#x221E;</UTF8String>
```

E.2.7.2 Puede expandirse una colección para que sea un subconjunto seleccionado (es decir, incluir todos los caracteres de la colección LATÍN BÁSICO) utilizando la "UnionMark" (véase la cláusula 46).

EJEMPLO

```
KatakanaAndBasicLatin ::= UniversalString (FROM (Katakana | BasicLatin))
```

E.2.8 CADENA DE CARACTERES

Se utilizará el tipo cadena de caracteres no restringida para modelar cualquier cadena de información que no pueda modelarse utilizando uno de los tipos cadena de caracteres restringida. Debe especificarse el repertorio de caracteres y su codificación en octetos.

EJEMPLO

```
PackedBCDString ::= CHARACTER STRING (WITH COMPONENTS {
                                                identification  (WITH
COMPONENTS {
                                                   fixed  PRESENT })
      /* The abstract and transfer syntaxes shall be
         packedBCDString-AbstractSyntaxId and
         packedBCDString-TransferSyntaxId defined below.
      */
                           } )
```

```
        /* object identifier value for a character abstract syntax
           (character set) whose alphabet
           is the digits 0 through 9.
        */
        PackedBCDString-AbstractSyntaxId  OBJECT  IDENTIFIER ::=
              { joint-iso-itu-t asn1(1) examples(123) packedBCD(2) charSet(0) }

        /* object identifier value for a character transfer syntax that
           packs two digits per octet, each digit encoded as 0000 to
           1001, 1111₂ used for padding.
        */
        PackedBCDString-TransferSyntaxId  OBJECT IDENTIFIER ::=
              { joint-iso-itu-t asn1(1) examples(123) packedBCD(2)
                    characterTransferSyntax(1) }
/* The encoding of PackedBCDString will contain only the defined
   encoding of the characters, with any necessary length field, and in
   the case of BER with a field carrying the tag. The object
   identifier values are not carried, as "fixed" has been specified.
*/
```

o utilizando la notación de valor XML:

```
packedBCDString-AbstractSyntaxId ::=
      <OBJECT_IDENTIFIER>
            joint-iso-itu-t.asn1(1).examples(123).packedBCD(2).charSet(0)
      </OBJECT_IDENTIFIER>


packedBCDString-TransferSyntaxId ::=
 <OBJECT_IDENTIFIER>
joint-iso-itu-t.asn1(1).examples(123).packedBCD(2).characterTransferSyntax(1)
 </OBJECT_IDENTIFIER>
```

o:

```
packedBCDString-AbstractSyntaxId ::=
      <OBJECT_IDENTIFIER>2.1.123.2.0</OBJECT_IDENTIFIER>

PackedBCDString-TransferSyntaxId ::=
      <OBJECT_IDENTIFIER>2.1.123.2.1</OBJECT_IDENTIFIER>
```

NOTA – Las reglas de codificación no necesariamente codifican valores del tipo **CHARACTER STRING** de modo tal que incluya siempre los valores de identificadores de objeto, si bien garantizan la preservación del valor abstracto en la codificación.

E.2.9 Nulo

Se utilizará el tipo nulo para indicar la ausencia efectiva de un componente de una secuencia.

EJEMPLO

```
PatientIdentifier ::= SEQUENCE {
      name              VisibleString,
      roomNumber        CHOICE {
            room        INTEGER,
            outPatient  NULL  --  if an out-patient  --
      }
}

lastPatient PatientIdentifier ::= {
      name  "Jane Doe",
      roomNumber  outPatient : NULL
}
```

o utilizando notación de valor XML:

```
lastPatient ::=
      <PatientIdentifier>
         <name>Jane Doe</name>
         <roomNumber><outPatient/></roomNumber>
      </PatientIdentifier>
```

E.2.10 Secuencia y secuencia de

E.2.10.1 Se utilizará el tipo secuencia de para modelar una colección de variables del mismo tipo, de número elevado o impredecible, y cuyo orden sea significativo.

EJEMPLO

```
NamesOfMemberNations ::= SEQUENCE OF VisibleString
--  in alphabetical order

firstTwo  NamesOfMemberNations ::= {"Australia", "Austria"}
```

o utilizando el identificador opcional:

```
NamesOfMemberNations2 ::= SEQUENCE OF memberNation VisibleString
--  in alphabetical order

firstTwo2  NamesOfMemberNations2 ::=
      {memberNation "Australia", memberNation "Austria"}
```

Utilizando la notación de valor XML los dos valores anteriores son:

```
firstTwo ::=
      <NamesOfMemberNations>
          <VisibleString>Australia</VisibleString>
          <VisibleString>Austria</VisibleString>
      </NamesOfMemberNations>

firstTwo2 ::=
      <NamesOfMemberNations2>
          <memberNation>Australia</memberNation>
          <memberNation>Austria</memberNation>
      </NamesOfMemberNations2>
```

E.2.10.2 Se utilizará el tipo secuencia (sequence) para modelar una colección de variables del mismo tipo, de número conocido y pequeño, y cuyo orden sea significativo, siempre que sea poco probable que la constitución de la colección cambie de una versión de protocolo a la siguiente.

EJEMPLO

```
NamesOfOfficers ::= SEQUENCE {
            president              VisibleString,
            vicePresident          VisibleString,
            secretary              VisibleString}

acmeCorp NamesOfOfficers ::= {
            president              "Jane Doe",
            vicePresident          "John Doe",
            secretary              "Joe Doe"}
```

o utilizando la notación de valor XML:

```
acmeCorp ::=
      <NamesOfOfficers>
          <president>Jane Doe</president>
           <vicePresident>John Doe</vicePresident>
           <secretary>Joe Doe</secretary>
      </NamesOfOfficers>
```

E.2.10.3 Se utilizará el tipo secuencia inextensible para modelar una colección de variables de tipos diferentes, de número conocido y pequeño, y cuyo orden sea significativo, siempre que sea poco probable que la constitución de la colección cambie de una versión del protocolo a la siguiente.

EJEMPLO

```
Credentials ::= SEQUENCE {
            userName               VisibleString,
            password               VisibleString,
            accountNumber          INTEGER}
```

E.2.10.4 Se utilizará el tipo secuencia extensible para modelar una colección de variables cuyo orden sea significativo, cuyo número sea actualmente conocido y pequeño pero que se espera aumente.

EJEMPLO

```
Record ::= SEQUENCE {  -- First version of protocol containing "Record"
      userName                    VisibleString,
      password                    VisibleString,
      accountNumber               INTEGER,
      ...,
      ...
}
```

en anticipación de:

```
Record ::= SEQUENCE {  -- Second version of protocol containing "Record"
      userName                    VisibleString,
      password                    VisibleString,
      accountNumber               INTEGER,
      ...,
      [[2:              -- Extension addition added in protocol version 2
         lastLoggedIn             GeneralizedTime OPTIONAL,
         minutesLastLoggedIn      INTEGER
      ]],
      ...
}
```

y aún posteriormente (la versión 3 del protocolo no añadió nada a **Record**:

```
Record ::= SEQUENCE {  -- Third version of protocol containing "Record"
      userName                    VisibleString,
      password                    VisibleString,
      accountNumber               INTEGER,
      ...,
      [[2:              -- Extension addition added in protocol version 2
         lastLoggedIn             GeneralizedTime OPTIONAL,
         minutesLastLoggedIn      INTEGER
      ]],
      [[4:              -- Extension addition added in protocol version 3

         certificate              Certificate,
         thumb                    ThumbPrint OPTIONAL
      ]],
      ...
      }
```

E.2.11 Conjunto y conjunto de

E.2.11.1 Se utilizará el tipo conjunto para modelar una colección de variables cuyo número sea conocido y pequeño y cuyo orden no sea significativo. Si la rotulación automática no está en vigor, se identificará cada variable con un rótulo específico al contexto como se indica a continuación. (Con rotulación automática, los rótulos no son necesarios.)

EJEMPLO

```
UserName ::= SET {
            personalName                  [0] VisibleString,
            organizationName            [1] VisibleString,
            countryName                 [2] VisibleString}

user UserName ::= {
            countryName                 "Nigeria",
            personalName                  "Jonas Maruba",
            organizationName            "Meteorology, Ltd."}
```

o utilizando la notación de valor XML:

```
user ::=
      <UserName>
         <countryName>Nigeria</countryName>
         <personalName>Jonas Maruba</personalName>
         <organizationName>Meteorology, Ltd.</organizationName>
      </UserName>
```

E.2.11.2 Se utilizará el tipo conjunto con **OPTIONAL** para modelar una colección de variables que sea un subconjunto (propio o impropio) de otra colección de variables de número conocido y razonablemente pequeño y cuyo orden no sea significativo. Si la rotulación automática no está en vigor, se identificará cada variable con un rótulo específico del contexto como se indica a continuación. (Con rotulación automática, los rótulos no son necesarios.)

EJEMPLO

```
UserName ::= SET {
      personalName                        [0] VisibleString,
      organizationName                    [1] VisibleString OPTIONAL
          --  defaults to that of the local organization  -- ,
      countryName                         [2] VisibleString OPTIONAL
          --  defaults to that of the local country  --  }
```

E.2.11.3 Se utilizará un tipo conjunto extensible para modelar una colección de variables cuya constitución pueda presumiblemente cambiar de una versión del protocolo a la siguiente. En el ejemplo siguiente se supone que **AUTOMATIC TAGS** se ha especificado en la definición de módulo.

EJEMPLO

```
UserName ::= SET {
      personalName           VisibleString,        -- First version of
"UserName"
      organizationName       VisibleString OPTIONAL ,
      countryName            VisibleString OPTIONAL,
      ...,
      ...
}

user UserName ::= { personalName  "Jonas Maruba" }
```

o utilizando notación de valor XML:

```
user ::=
      <UserName>
         <personalName>Jonas Maruba</personalName>
      </UserName>
```

en anticipación de:

```
UserName ::= SET {           -- Second version of "UserName"
      personalName                VisibleString,
      organizationName            VisibleString OPTIONAL,
      countryName                 VisibleString OPTIONAL,
      ...,
      [[2:                   -- Extension addition added in protocol version 2
         internetEmailAddress     VisbleString,
         faxNumber                VisibleString OPTIONAL
      ]],
      ...
}

user  UserName ::= {
      personalName                "Jonas Maruba",
      internetEmailAddress        "jonas@meteor.ngo.com"
}
```

o utilizando notación de valor XML:

```
 user ::=
<UserName>
   <personalName>Jonas Maruba</personalName>
   <internetEmailAddress>jonas@meteor.ngo.com</internetEmailAddress>
</UserName>
```

y aún posteriormente (las versiones 3 y 4 del protocolo no introdujeron adiciones a **UserName**):

```
UserName ::= SET {     -- Fifth version of protocol containing "UserName"
      personalName                VisibleString,
      organizationName            VisibleString OPTIONAL,
      countryName                 VisibleString OPTIONAL,
      ...,
      [[2:                        -- Extension addition added in version 2
         internetEmailAddress     VisbleString,
         faxNumber                VisibleString OPTIONAL
      ]],
      [[5:                        -- Extension addition added in version 5
      phoneNumber                 VisibleString OPTIONAL
      ]],
```

```
      ...
}

user  UserName ::= {
      personalName                "Jonas Maruba",
      internetEmailAddress        "jonas@meteor.ngo.com"
}
```

o utilizando la notación de valor XML:

```
user ::=
      <UserName>
         <personalName>Jonas Maruba</personalName>
         <internetEmailAddress>jonas@meteor.ngo.com</internetEmailAddress>
      </UserName>
```

E.2.11.4 Se utilizará el tipo conjunto de para modelar una colección de variables cuyos tipos sean los mismos y cuyo orden no sea significativo.

EJEMPLO

```
Keywords ::= SET OF VisibleString  --  in arbitrary order

someASN1Keywords Keywords ::= {"INTEGER", "BOOLEAN", "REAL"}
```

o utilizando el identificador opcional:

```
Keywords2 ::= SET OF keyword VisibleString  --  in arbitrary order

someASN1Keywords2 Keywords2 ::= {keyword "INTEGER", keyword "BOOLEAN",
      keyword "REAL"}
```

Utilizando la notación XML los dos valores anteriores son:

```
someASN1Keywords ::=
      <Keywords>
         <VisibleString>INTEGER</VisibleString>
         <VisibleString>BOOLEAN</VisibleString>
         <VisibleString>REAL</VisibleString>
      </Keywords>

someASN1Keywords2 ::=
      <Keywords2>
         <keyword>INTEGER</keyword>
         <keyword>BOOLEAN</keyword>
         <keyword>REAL</keyword>
      </Keywords2>
```

E.2.12 Rotulado

Antes de la introducción de la construcción **AUTOMATIC TAGS**, las especificaciones ASN.1 solían contener rótulos. En las subcláusulas que siguen se describe la manera según la cual se aplicaba la rotulación normalmente. Con la introducción de **AUTOMATIC TAGS**, las nuevas especificaciones ASN.1 no necesitan hacer uso de la notación de rótulo, si bien aquellas que modifiquen notación antigua quizá tengan que ocuparse de los rótulos. Se insta a los nuevos usuarios de la notación ASN.1 a utilizar **AUTOMATIC TAGS** para que la notación sea más legible.

E.2.12.1 Los rótulos de clase universal sólo se utilizan en esta Recomendación | Norma Internacional. La notación **[UNIVERSAL 30]** (por ejemplo) se proporciona únicamente para facilitar la precisión de la definición de los "UsefulTypes" (véase 41.1). No deberá utilizarse en otro sitio.

E.2.12.2 Un estilo de utilización de los rótulos, que se da frecuentemente, consiste en asignar un rótulo de clase de aplicación exactamente una vez en la totalidad de la especificación, utilizándolo para especificar un tipo que se emplea de una manera amplia y dispersa dentro de la especificación. También se utiliza frecuentemente (solamente una vez) un rótulo de clase de aplicación para rotular los tipos de la **CHOICE** externa de una aplicación, proporcionando la identificación de mensajes individuales mediante el rótulo de clase de aplicación. Lo que sigue es un ejemplo de utilización en el primer caso.

EJEMPLO

```
FileName ::= [APPLICATION 8] SEQUENCE {
           directoryName               VisibleString,
           directoryRelativeFileName   VisibleString}
```

E.2.12.3 La rotulación específica al contexto se aplica frecuentemente de manera algorítmica a todos los componentes de un **SET**, **SEQUENCE** o **CHOICE**. Se señala, no obstante, que la prestación **AUTOMATIC TAGS** facilita esto al usuario.

EJEMPLO

```
CustomerRecord ::= SET {
            name                        [0] VisibleString,
            mailingAddress              [1] VisibleString,
            accountNumber               [2] INTEGER,
            balanceDue                  [3] INTEGER  --  in cents  --}

CustomerAttribute ::= CHOICE {
            name                        [0] VisibleString,
            mailingAddress              [1] VisibleString,
            accountNumber               [2] INTEGER,
            balanceDue                  [3] INTEGER  --  in cents  --}
```

E.2.12.4 La rotulación de clase privada no deberá utilizarse, normalmente, en las especificaciones normalizadas internacionalmente (aunque esto no puede prohibirse). Las aplicaciones producidas por una empresa utilizarán normalmente clases de rótulos de aplicación y específicos al contexto. No obstante, puede haber alguna vez casos en los que una especificación específica a la empresa trate de ampliar una especificación normalizada internacionalmente y, en tal caso, la utilización de rótulos de clase privada puede resultar ventajosa al proteger parcialmente la especificación específica a la empresa frente a los cambios de la especificación normalizada internacionalmente.

EJEMPLO

```
AcmeBadgeNumber ::= [PRIVATE 2] INTEGER

badgeNumber AcmeBadgeNumber ::= 2345
```

o utilizando la notación de valor XML:

```
badgeNumber ::= <AcmeBadgeNumber>2345</AcmeBadgeNumber>
```

E.2.12.5 La utilización textual de **IMPLICIT** en cada uno de los rótulos se encuentra, por lo general, únicamente en las especificaciones antiguas. BER produce una representación menos compacta cuando se emplea la rotulación explícita que cuando se emplea la implícita. PER produce la misma codificación compacta en ambos casos. Con BER y rotulación explícita, hay más visibilidad del tipo subyacente (**INTEGER**, **REAL**, **BOOLEAN**, etc.) en los datos codificados. En estas directrices se ha utilizado la rotulación implícita en los ejemplos en que está autorizado a hacerlo. Dependiendo de la regla de codificación, esto puede dar como resultado una representación compacta, muy conveniente en algunas aplicaciones. En otras aplicaciones, la compactibilidad puede ser menos importante que, por ejemplo, la aptitud para efectuar una verificación de tipos fuerte. En el último caso, puede utilizarse la rotulación explícita.

EJEMPLO

```
CustomerRecord ::= SET {
            name                      [0] IMPLICIT VisibleString,
            mailingAddress            [1] IMPLICIT VisibleString,
            accountNumber             [2] IMPLICIT INTEGER,
            balanceDue                [3] IMPLICIT INTEGER  --  in cents  --
}

CustomerAttribute ::= CHOICE {
            name                      [0] IMPLICIT VisibleString,
            mailingAddress            [1] IMPLICIT VisibleString,
            accountNumber             [2] IMPLICIT INTEGER,
            balanceDue                [3] IMPLICIT INTEGER  --  in cents  --
}
```

E.2.12.6 El criterio respecto a la utilización de rótulos en las nuevas especificaciones ASN.1 que hagan referencia a esta Recomendación | Norma Internacional es muy simple: NO UTILIZAR RÓTULOS. Poner **AUTOMATIC TAGS** en el encabezamiento del módulo y olvidarse a continuación de los rótulos. Si se necesita añadir componentes nuevos a los **SET**, **SEQUENCE** o **CHOICE** en una versión posterior, se añaden al final.

E.2.13 Elección

E.2.13.1 Se utilizará una **CHOICE** para modelar una variable seleccionada de una colección de variables cuyo número sea conocido y reducido.

EJEMPLO

```
FileIdentifier ::= CHOICE {
            relativeName     VisibleString,
```

```
                -- name of file (for example, "MarchProgressReport")
                  absoluteName     VisibleString,
                -- name of file and containing directory
                -- (for example, "<Williams>MarchProgressReport")
                  serialNumber     INTEGER
                -- system-assigned identifier for file --}

        file FileIdentifier ::= serialNumber : 106448503
```

o utilizando la notación de valor XML:

```
fileIdentifier ::=
            <FileIdentifier>
               <serialNumber>106448503</serialNumber>
            </FileIdentifier>
```

E.2.13.2 Se utilizará una **CHOICE** extensible para modelar una variable seleccionada de una colección de variables cuya constitución vaya a cambiar probablemente de una versión del protocolo a la siguiente.

EJEMPLO

```
      FileIdentifier ::= CHOICE {             -- First version of FileIdentifier
                  relativeName    VisibleString,

                  absoluteName    VisibleString,

                  ...,   ...
      }

      fileId1  FileIdentifier ::= relativeName : "MarchProgressReport.doc"
```

o utilizando la notación de valor XML:

```
fileId1 ::=
      <FileIdentifier>
         <relativeName>MarchProgressReport.doc</relativeName>
      </FileIdentifier>
```

en anticipación de:

```
      FileIdentifier ::= CHOICE {       -- Second version of FileIdentifier
            relativeName     VisibleString,
            absoluteName     VisibleString,
            ...,
            serialNumber     INTEGER,    -- Extension addition added in version 2
            ...
      }

      fileId1  FileIdentifier ::= relativeName : "MarchProgressReport.doc"

      fileId2  FileIdentifier ::= serialNumber : 214
```

o utilizando la notación de valor XML:

```
fileId1 ::=
      <FileIdentifier>
         <relativeName>MarchProgressReport.doc</relativeName>
      </FileIdentifier>

fileId2 ::=
      <FileIdentifier>
         <serialNumber>214</serialNumber>
      </FileIdentifier>
```

y aún posteriormente:

```
      FileIdentifier ::= CHOICE {       -- Third version of FileIdentifier
            relativeName     VisibleString,
            absoluteName     VisibleString,
            ...,
            serialNumber     INTEGER,    -- Extension addition added in version 2
            [[                           -- Extension addition added in version 3
               vendorSpecificVendorExt,
               unidentified  NULL
            ]],
```

```
      ...
}

fileId1  FileIdentifier ::= relativeName : "MarchProgressReport.doc"

fileId2  FileIdentifier ::= serialNumber : 214

fileId3  FileIdentifier ::= unidentified : NULL
```

o utilizando la notación de valor XML:

```
fileId1 ::=
      <FileIdentifier>
         <relativeName>MarchProgressReport.doc</relativeName>
      </FileIdentifier>

fileId2 ::=
      <FileIdentifier>
         <serialNumber>214</serialNumber>
      </FileIdentifier>

fileId3 ::=
      <FileIdentifier>
         <unidentified/>
      </FileIdentifier>
```

E.2.13.3 Se utilizará una **CHOICE** extensible de un solo tipo cuando se contemple la posibilidad de permitir más de un tipo en el futuro.

EJEMPLO

```
Greeting ::= CHOICE {              -- First version of "Greeting"
      postCard         VisibleString,
      ...,
      ...
}
```

en anticipación de:

```
Greeting ::= CHOICE {              -- Second version of "Greeting"
      postCard    VisibleString,
      ...,
      [[2:                         -- Extension addition added in version 2
         audio          Audio,
         video          Video
      ]],
      ...
}
```

E.2.13.4 Cuando un valor elección está anidado dentro de otro valor elección, se requieren múltiples caracteres dos puntos (:).

EJEMPLO

```
Greeting ::= [APPLICATION 12] CHOICE {
            postCard         VisibleString,
            recording        Voice }

Voice ::= CHOICE {
            english          OCTET STRING,
            swahili          OCTET STRING }

myGreeting Greeting ::= recording : english : '019838547E0'H
```

o utilizando la notación de valor XML:

```
myGreeting ::=
      <Greeting>
         <recording><english>019838547E0</english></recording>
      </Greeting>
```

E.2.14 Tipo selección

E.2.14.1 Se utilizará el tipo selección para modelar una variable cuyo tipo sea el de algunas alternativas determinadas de una **CHOICE** definida anteriormente.

E.2.14.2 Considérese la definición:

```
FileAttribute ::= CHOICE {
            date-last-used   INTEGER,
            file-name        VisibleString}
```

es posible, entonces, la siguiente definición:

```
AttributeList ::= SEQUENCE {
            first-attribute  date-last-used  < FileAttribute,
            second-attribute file-name < FileAttribute }
```

con una posible notación de valor de:

```
listOfAttributes AttributeList ::=  {
            first-attribute  27,
            second-attribute "PROGRAM" }
```

o utilizando la notación de valor XML:

```
listOfAttributes ::=
      <AttributeList>
         <first-attribute>27</first-attribute>
         <second-attribute>PROGRAM</second-attribute>
      </AttributeList>
```

E.2.15 Tipo campo de clase de objeto

E.2.15.1 Se utilizará el tipo campo de clase de objeto para identificar un tipo definido mediante una clase de objeto de información (véase la Rec. UIT-T X.681 | ISO/CEI 8824-2). Por ejemplo, campos de la clase de objeto de información **ATTRIBUTE** pueden ser utilizados en la definición de un tipo, **Attribute**.

EXAMPLE

```
ATTRIBUTE ::= CLASS {
      &AttributeType,
      &attributeId           OBJECT IDENTIFIER UNIQUE
            }

Attribute ::= SEQUENCE {
      attributeID       ATTRIBUTE.&attributeId,-- this is normally constrained.
      attributeValue    ATTRIBUTE.&AttributeType -- this is normally constrained.
            }
```

Tanto **ATTRIBUTE.&attributeID** como **ATTRIBUTE.&AttributeType** son tipos campo de clase de objeto puesto que son tipos definidos por referencia a una clase de objeto de información (**ATTRIBUTE**). El tipo **ATTRIBUTE.&attributeId** está fijo porque está definido explícitamente en **ATTRIBUTE** como un **OBJECT IDENTIFIER**. Sin embargo, el tipo **ATTRIBUTE.&AttributeType** puede llevar un valor de cualquier tipo definido utilizando ASN.1, puesto que su tipo no está fijo en la definición de la clase de objeto de información **ATTRIBUTE**. Las notaciones que poseen la propiedad de poder llevar un valor de cualquier tipo reciben el nombre de "notaciones de tipo abierto", por lo que **ATTRIBUTE.&AttributeType** es de tipo abierto.

E.2.16 pdv incrustado

E.2.16.1 Se utilizará el tipo pdv incrustado para modelar una variable cuyo tipo no se haya especificado o lo haya sido en otro sitio sin limitación en la notación utilizada para especificar el tipo.

EJEMPLO

```
FileContents ::= EMBEDDED PDV

DocumentList ::= SEQUENCE OF document EMBEDDED PDV
```

E.2.17 Externo

El tipo externo es similar al tipo pdv incrustado, pero tiene menos opciones de identificación. En las especificaciones nuevas se preferirá, por lo general, utilizar pdv incrustado debido a su mayor flexibilidad y al hecho de que algunas reglas de codificación codifican su valor de una manera más eficiente.

E.2.18 Ejemplar de

E.2.18.1 Se utilizará un ejemplar de para especificar un tipo que contenga un campo de identificador de objeto y un valor de tipo abierto cuyo tipo venga determinado por el identificador de objeto. El tipo ejemplar de sólo puede utilizarse si la asociación entre el valor del identificador de objeto y el tipo se especifica utilizando un objeto de

información de una clase obtenida a partir de **TYPE-IDENTIFIER** (véase la Rec. UIT-T X.681 | ISO/CEI 8824-2, anexos A y C).

EJEMPLO

```
ACCESS-CONTROL-CLASS ::= TYPE-IDENTIFIER

Get-Invoke ::= SEQUENCE {
      objectClass       ObjectClass,
      objectInstance    ObjectInstance,
      accessControl     INSTANCE OF ACCESS-CONTROL-CLASS, --  this is normally
                                                          --  constrained.
      attributeID       ATTRIBUTE.&attributeId
      }
```

Entonces Get-Invoke es equivalente a:

```
Get-Invoke ::= SEQUENCE {
      objectClass       ObjectClass,
      objectInstance    ObjectInstance,
      accessControl     [UNIVERSAL 8] IMPLICIT SEQUENCE {
            type-id     ACCESS-CONTROL-CLASS.&id,    --  this is normally
                                                     --  constrained.
            value       [0] ACCESS-CONTROL-CLASS.&Type --  this is normally
                                                       --  constrained.
      },
      attributeID ATTRIBUTE.&attributeId
}
```

La verdadera utilidad del tipo ejemplar de no se ve sino hasta que se constriñe mediante un conjunto de objetos de información, pero ese ejemplo va más allá del alcance de la presente Recomendación | Norma Internacional. Véase en la Rec. UIT-T X.682 | ISO/CEI 8824-3 la definición de conjunto de objetos de información, y en el anexo A de la Rec. UIT-T X.682 | ISO/CEI 8824-3, cómo utilizar un conjunto de objetos de información para constreñir un tipo ejemplar de.

E.2.19 Identificador de objeto relativo

E.2.19.1 Se utilizará un tipo identificador de objeto relativo para transmitir valores de identificador de objeto de forma más compacta en contextos en los que se conoce la parte anterior del valor de identificador de objeto. Pueden plantearse tres situaciones:

a) La primera parte del valor de identificador de objeto es fija para una determinada especificación (se trata de una norma industrial específica) y todos los OID están en relación con un OID atribuido al organismo de normalización. En este caso se utilizará:

```
RELATIVE-OID      --  The relative object identifier value is
                  --  relative to {iso identified-organization  set(22)}
```

b) La primera parte del valor de identificador de objeto es con frecuencia un valor conocido en el momento de la especificación, pero en ocasiones puede tratarse de un valor más general. En este caso, se utilizará:

```
CHOICE
      {a    RELATIVE-OID    --  The value is relative to {1  3  22}--,
       b    OBJECT IDENTIFIER  -- Any object identifier value--}
```

c) La primera parte del valor de identificador de objeto no se conoce hasta el momento de las comunicaciones, pero con frecuencia será común a muchos valores que es necesario enviar, y muy a menudo se tratará de un valor conocido en el momento de la especificación. En este caso, se utilizará (por ejemplo):

```
SEQUENCE
      {oid-root   OBJECT IDENTIFIER DEFAULT {1  3  22},
       reloids    SEQUENCE OF RELATIVE-OID --relative to oid-root--}
```

E.3 Identificación de sintaxis abstractas

E.3.1 Es común definir los protocolos asociando semántica con cada uno de los valores de un tipo ASN.1 simple, por lo general un tipo choice. (Algunas veces se hace referencia, informalmente, a este tipo ASN.1 como "el tipo de nivel superior de la aplicación"). Este conjunto de valores abstractos se denomina formalmente la sintaxis abstracta de la aplicación. Una sintaxis abstracta puede identificarse asignando un nombre de sintaxis abstracta como identificador de objeto tipo ASN.1.

E.3.2 La asignación de un identificador de objeto a una sintaxis abstracta puede efectuarse utilizando la clase de objeto de información incorporada **ABSTRACT-SINTAX** definida en la Rec. UIT-T X.681 | ISO/CEI 8824-2. Esto también es útil para poder identificar claramente el tipo de nivel superior de la aplicación.

E.3.3 A continuación se presenta un ejemplo de texto que podría aparecer en una aplicación de una aplicación.

EXAMPLE

```
Application-ASN1 DEFINITIONS ::=
      BEGIN
      EXPORTS Application-PDU;

      Application-PDU ::= CHOICE {
                  connect-pdu    ..... ,
                  data-pdu    CHOICE {
                        ..... ,
                        .....
                  },
                  .....
            }
            ......
      END

Abstract-Syntax-Module DEFINITIONS ::=
      BEGIN
      IMPORTS Application-PDU FROM Application-ASN1;

--  This application defines the following abstract syntax:

      Abstract-Syntax ABSTRACT-SYNTAX ::=
                  { Application-PDU  IDENTIFIED BY
                          application-abstract-syntax-object-id }

      application-abstract-syntax-object-id OBJECT IDENTIFIER ::=
                  {joint-iso-itu-t asn1(1) examples(123)
                        application-abstract-syntax(3) }
--  The corresponding object descriptor is:

      application-abstract-syntax-descriptor ObjectDescriptor ::=
                        "Example Application Abstract Syntax"

--  The ASN.1 object identifier and object descriptor values:
                  --  encoding rule object identifier
                  --  encoding rule object descriptor
--  assigned to encoding rules in ITU-T Rec. X.690 | ISO/IEC 8825-1
-- and ITU-T Rec. X.691 | ISO/IEC 8825-2 can be used as the transfer
-- syntax identifier in conjunction with this transfer syntax.

END
```

E.3.4 Para garantizar el interfuncionamiento, la norma puede además identificar una sintaxis de transferencia obligatoria (por lo general una de las que se definen en las reglas de codificación de la Rec. UIT-T X.690 | ISO/CEI 8825-1, la Rec. X.691 | ISO/CEI 8825-2 o la Rec. X.692 | ISO/CEI 8825-3).

E.4 Subtipos

E.4.1 Se utilizarán subtipos para limitar los valores permisibles de un tipo existente en una situación particular.

EJEMPLOS

```
AtomicNumber    ::=   INTEGER (1..104)

TouchToneString ::=  IA5String
                  (FROM ("0123456789" | "*" | "#")) (SIZE (1..63))

ParameterList   ::=  SET SIZE (1..63) OF Parameter

SmallPrime   ::=  INTEGER (2|3|5|7|11|13|17|19|23|29)
```

E.4.2 Se definirá el uso de una constricción de subtipo extensible para modelar un tipo **INTEGER** cuyo conjunto de valores permitidos sea pequeño y bien definido, pero que se espere aumente.

EJEMPLO

```
SmallPrime ::= INTEGER (2 | 3, ...)      -- First version of SmallPrime
```

en anticipación de:

```
SmallPrime ::= INTEGER (2 | 3, ..., 5 | 7 | 11)
-- Second version of SmallPrime
```

y aún posteriormente:

```
SmallPrime ::= INTEGER (2 | 3, ..., 5 | 7 | 11 | 13 | 17 | 19)
-- Third version of SmallPrime
```

NOTA – Para determinados tipos, algunas reglas de codificación (por ejemplo, PER) proporcionan una codificación altamente optimizada para valores raíz de extensión de constricción de subtipo (es decir, valores que aparecen antes del "**...**") y una codificación menos optimizada para valores adición de extensión de constricción de subtipo (es decir, valores que aparecen después del "**...**"), mientras que algunas otras reglas de codificación (por ejemplo, BER) para constricciones de subtipo no afectan a la codificación.

E.4.3 Cuando dos o más tipos relacionados tengan en común aspectos significativos, se considerará la definición explícita de su progenitor común como un tipo y se utilizará la subtipificación para los tipos individuales. Este enfoque permite aclarar la relación y sus coincidencias, a la vez que alienta (aunque no obliga) a que continúe esta práctica conforme evolucionan los tipos. Así, facilita el uso de soluciones de implementación comunes para el tratamiento de valores de estos tipos.

EJEMPLO

```
Envelope    ::=  SET {
                        typeA  TypeA,
                        typeB  TypeB   OPTIONAL,
                        typeC  TypeC   OPTIONAL}
             --  the common parent

ABEnvelope  ::=  Envelope (WITH  COMPONENTS
                        {... ,
                              typeB   PRESENT, typeC   ABSENT})
             --  where typeB must always appear and typeC must not

ACEnvelope  ::=  Envelope (WITH  COMPONENTS
                        {... ,
                              typeB   ABSENT, typeC   PRESENT})
             --  where typeC must always appear and typeB must not
```

Las últimas definiciones podrían expresarse, de forma alternativa, como:

```
ABEnvelope  ::= Envelope (WITH COMPONENTS {typeA, typeB})

ACEnvelope  ::= Envelope (WITH COMPONENTS {typeA, typeC})
```

La elección entre las alternativas debe basarse en factores tales como el número de componentes en el tipo progenitor, y el número de aquellos que son opcionales, el grado de diferencia entre los tipos individuales y la estrategia de evolución probable.

E.4.4 Se utilizará la subtipificación para definir parcialmente un valor, por ejemplo, una unidad de datos de protocolo (PDU) vaya a someterse a una prueba de conformidad, cuando la prueba afecte únicamente a algunos de los componentes de la PDU.

EJEMPLO

Dada:

```
PDU  ::=  SET
        {alpha          INTEGER,
         beta           IA5String  OPTIONAL,
         gamma          SEQUENCE OF Parameter,
         delta          BOOLEAN}
```

cuando se componga una prueba que requiera que el booleano sea falso y el entero negativo, escríbase:

```
TestPDU   ::=  PDU (WITH COMPONENTS
                {... ,
                 delta (FALSE),
                 alpha (MIN..<0)})
```

y si, además, debe estar presente la cadena **IA5String**, **beta**, con una longitud de 5 ó 12 caracteres, escríbase:

```
FurtherTestPDU ::= TestPDU (WITH COMPONENTS {... , beta (SIZE (5|12)) PRESENT
} )
```

E.4.5 Si un tipo de datos de propósito general se ha definido como **SEQUENCE OF**, se utilizará la subtipificación para definir un subtipo restringido del tipo general.

EJEMPLO

```
Text-block ::=  SEQUENCE OF VisibleString

Address   ::=  Text-block (SIZE (1..6)) (WITH COMPONENT (SIZE (1..32)))
```

E.4.6 Si un tipo de datos de propósito general se ha definido como **CHOICE**, se utilizará la subtipificación para definir un subtipo restringido del tipo general.

EJEMPLO

```
Z ::= CHOICE {
            a           A,
            b           B,
            c           C,
            d           D,
            e           E
      }

V ::= Z (WITH COMPONENTS { ..., a  ABSENT,  b  ABSENT }) --  'a' and 'b' must
                              -- be absent, either 'c',
                              --  'd' or 'e' may be present in a value.

W ::= Z (WITH COMPONENTS { ..., a  PRESENT }) --  only 'a' can be present
                                    -- (see 47.8.9.2).

X ::= Z (WITH COMPONENTS { a  PRESENT })   --  only 'a' can be present
                                 -- (see 47.8.9.2).

Y ::= Z (WITH COMPONENTS { a  ABSENT,  b,  c }) --  'a', 'd' and 'e' must be
                                 -- absent, either 'b' or 'c' may
                                 -- be present in a value.
```

NOTA – **W** y **X** son semánticamente idénticas.

E.4.7 Se utilizarán subtipos contenidos para formar nuevos subtipos a partir de los subtipos existentes.

EJEMPLO

```
Months      ::= ENUMERATED {
                        january          (1),
                        february         (2),
                        march            (3),
                        april            (4),
                        may              (5),
                        june             (6),
                        july             (7),
                        august           (8),
                        september        (9),
                        october          (10),
                        november         (11),
                        december         (12) }

First-quarter ::= Months ( january | february  | march )

Second-quarter ::= Months ( april | may | june )

Third-quarter ::= Months ( july | august | september )

Fourth-quarter ::= Months ( october | november | december )

First-half ::= Months ( First-quarter | Second-quarter )

Second-half ::= Months ( Third-quarter | Fourth-quarter )
```

Anexo F

Suplemento didáctico sobre cadenas de caracteres ASN.1

(Este anexo no es parte integrante de esta Recomendación | Norma Internacional)

F.1 Soporte de cadenas de caracteres en ASN.1

F.1.1 ASN.1 soporta cuatro grupos de cadenas de caracteres, a saber:

a) Tipos cadena de caracteres basados en el *ISO International Register of Coded Character Sets to be used with Escape Sequences* (es decir, basados en la estructura de ISO/CEI 646) y el International Register of Coded Character Sets asociado, y proporcionados por los tipos **`VisibleString`**, **`IA5String`**, **`TeletexString`**, **`VideotexString`**, **`GraphicString`** y **`GeneralString`**.

b) Tipos cadena de caracteres basados en ISO/CEI 10646-1 y proporcionados creando subconjuntos del tipo **`UniversalString`**, **`UTF8String`** o **`BMPString`**, estando los subconjuntos definidos en ISO/CEI 10646-1, o utilizando caracteres con nombre.

c) Tipos cadena de caracteres que proporcionan una colección pequeña y simple de caracteres especificados en esta Recomendación | Norma Internacional y destinados a uso especializado; estos son los tipos **`NumericString`** y **`PrintableString`**.

d) Utilización del tipo **`CHARACTER STRING`**, con negociación del juego de caracteres que habrá de utilizarse (o anuncio del juego de caracteres que se está utilizando); esto permite a una implementación utilizar cualquier colección de caracteres y codificaciones para las cuales se hayan asignado los **`OBJECT IDENTIFIER`**, incluidos los de *ISO International Register of Coded Character Sets to be used with Escape Sequence*, ISO/CEI 7350, ISO/CEI 10646-1, y colecciones privadas de caracteres y codificaciones (los perfiles pueden imponer requisitos o restricciones a los juegos de caracteres – las sintaxis abstractas de caracteres – que habrán de utilizarse).

F.2 Los tipos UniversalString, UTF8String y BMPString

F.2.1 Los tipos **`UniversalString`** y **`UTF8String`** llevan cualquier carácter de ISO/CEI 10646-1. El juego de caracteres ISO/CEI 10646-1 es generalmente demasiado grande para que tenga sentido exigir una conformidad y, normalmente, se recurre a un subconjunto del mismo formado por una combinación de las colecciones de caracteres normales de ISO/CEI 10646-1, anexo A.

F.2.2 El tipo **`BMPString`** lleva cualquier carácter del plan plurilingüe básico de ISO/CEI 10646-1. Normalmente se recurre a un subconjunto del plan plurilingüe básico formado por una combinación de colecciones de caracteres normalizadas en ISO/CEI 10646-1, anexo A.

F.2.3 Para las colecciones definidas en ISO/CEI 10646-1, anexo A hay referencias de tipos definidas en el módulo ASN.1 incorporado **`ASN1-CHARACTER-MODULE`** (véase la cláusula 38). El mecanismo "subtype constraint" (limitación de subtipo) permite definir nuevos subtipos de **`UniversalString`** que son combinaciones de subtipos existentes.

F.2.4 Ejemplos de referencias de tipo definidas en **`ASN1-CHARACTER-MODULE`** y sus correspondientes de nombres de colección de ISO/CEI 10646-1 son:

`BasicLatin`	LATÍN BÁSICO
`Latin-1Supplement`	SUPLEMENTO DE LATÍN1
`LatinExtended-a`	LATÍN AMPLIADO A
`LatinExtended-b`	LATÍN AMPLIADO B
`IpaExtensions`	AMPLIACIONES DEL ALFABETO FONÉTICO INTERNACIONAL
`SpacingModifierLetters`	LETRAS MODIFICADORAS DEL ESPACIAMIENTO
`CombiningDiacriticalMarks`	PUNTOS DIACRÍTICOS COMBINANTES

F.2.5 ISO/CEI 10646-1 especifica tres "niveles de implementación" y exige que todos los usos de ISO/CEI 10646-1 especifiquen el nivel de implementación.

El nivel de implementación está relacionado con el grado de soporte que se da a los *caracteres combinantes* del repertorio de caracteres y, por consiguiente, en términos de ASN.1, define un subconjunto de los tipos cadena de caracteres restringida **UniversalString** y **BMPString**.

En el nivel 1 de implementación, no se permiten caracteres combinantes y normalmente, existe una correspondencia biunívoca entre caracteres abstractos de las cadenas de caracteres ASN.1 y caracteres impresos de una representación de la cadena.

En el nivel 2 de implementación están disponibles para su utilización determinados caracteres combinantes (indicados en ISO/CEI 10646-1, anexo B), pero hay otros cuyo uso está prohibido.

En el nivel 3 de implementación no hay restricciones de uso de los caracteres combinantes.

F.2.6 Una **BMPString** o **UniversalString** puede restringirse para excluir todas las funciones de control mediante la utilización de la notación de subtipo como sigue:

```
VanillaBMPString ::= BMPString (FROM (ALL EXCEPT ({0,0,0,0}..{0,0,0,31} |
                                          {0,0,0,128}..{0,0,0,159})))
```

o, de forma equivalente:

```
C0 ::= BMPString (FROM ({0,0,0,0} .. {0,0,0,31}))   -- C0 control functions
C1 ::= BMPString (FROM ({0,0,0,128} .. {0,0,0,159}))  -- C1 control functions
VanillaBMPString ::= BMPString (FROM (ALL EXCEPT (C0 | C1)))
```

F.3 Requisitos de conformidad de ISO/CEI 10646-1

La utilización de **UniversalString**, **BMPString** o **UTF8String** (o subtipos de los mismos) en una definición de tipo ASN.1 exige que se tengan en cuenta los requisitos de conformidad de ISO/CEI 10646-1.

Según dichos requisitos de conformidad, los implementadores de una norma (por ejemplo, X) que utilice esos tipos ASN.1, han de proporcionar (en los enunciados de conformidad de implementación de protocolos) una declaración del subconjunto adoptado de ISO/CEI 10646-1 para su implementación de la norma X, y del nivel (soporte de caracteres combinantes) de la implementación.

La utilización de un subtipo ASN.1 de **UniversalString**, **UTF8String** o **BMPString** en una especificación exige que la implementación soporte todos los caracteres de ISO/CEI 10646-1 incluidos en ese subtipo ASN.1 y, por consiguiente, que esos caracteres (por lo menos) estén presentes en el subconjunto adoptado para la implementación. También se requiere el soporte del nivel establecido para todos esos subtipos ASN.1.

NOTA – Una especificación ASN.1 (en ausencia de parámetros de la sintaxis abstracta y de especificaciones de excepción) determina el conjunto (máximo) de caracteres que puede ser transmitido y el conjunto (mínimo) de caracteres que ha de ser tratado en recepción. El conjunto adoptado de ISO/CEI 10646-1 exige la no transmisión de los caracteres que excedan de este conjunto y que todos los caracteres del mismo sean soportados en recepción. Es necesario, por consiguiente, que el conjunto adoptado sea precisamente el conjunto de todos los caracteres permitidos por la especificación ASN.1. El caso en que está presente un parámetro de la sintaxis abstracta se examina más adelante.

F.4 Recomendaciones a los usuarios de la ASN.1 sobre conformidad de ISO/CEI 10646-1

Los usuarios de ASN.1 deben establecer claramente el conjunto de caracteres de ISO/CEI 10646-1 que formará el subconjunto adoptado de las implementaciones (y el nivel de implementación requerido) para que se satisfagan los requisitos de su norma.

Esto puede hacerse convenientemente definiendo un subtipo ASN.1 de **UniversalString**, **UTF8String** o **BMPString** que contenga todos los caracteres necesarios para la norma y restringiéndolo a **Level1** (nivel 1) o **Level2** (nivel 2) si procede. Un nombre adecuado para este tipo podría ser **ISO-10646-String**.

EJEMPLO

```
ISO-10646-String ::= BMPString
      (FROM (Level2 INTERSECTION (BasicLatin UNION HebrewExtended UNION Hiragana)))
      --  This is the type that defines the minimum set of characters in
      --  the adopted subset for an implementation of this standard. The
      --  implementation level is required to be at least level 2.
```

En un entorno OSI, la declaración de conformidad de implementación del protocolo OSI contendría entonces una simple declaración de que el subconjunto adoptado de ISO/CEI 10646-1 es el subconjunto limitado (y el nivel) definido por **ISO-10646-String**, y de que se utilizaría **ISO-10646-String** (posiblemente subtipificada) en todas las normas en que se tuvieran que incluir cadenas de ISO/CEI 10646-1.

EJEMPLO DE DECLARACIÓN DE CONFORMIDAD

El subconjunto adoptado de ISO/CEI 10646-1 es el subconjunto limitado formado por todos los caracteres del tipo ASN.1 **ISO-10646-String** definido en el módulo <your module name goes here>, con un nivel de implementación de 2.

EJEMPLO DE UTILIZACIÓN EN PROTOCOLO

```
Message ::=  SEQUENCE {
      first-field ISO-10646-String,--  all characters in the adopted
                                   -- subset can appear
      second-field     ISO-10646-String
            (FROM (latinSmallLetterA .. latinSmallLetterZ)),   --  lower case
                                                          -- latin letters only
      third-field ISO-10646-String
            (FROM (digitZero .. digitNine)) --  digits only

}
```

F.5 Subconjuntos adoptados como parámetros de la sintaxis abstracta

ISO/CEI 10646-1 exige que el subconjunto adoptado y el nivel de una implementación se definan *explícitamente*. Si un usuario de la ASN.1 no desea limitar la gama de caracteres ISO/CEI 10646-1 en alguna parte de la norma que se define, esto puede expresarse definiendo **ISO-10646-String** (por ejemplo) como un subtipo de **UniversalString**, **BMPString** o **UTF8String** con una limitación de subtipo que conste de (o que incluya) **ImplementorsSubset** que se deja como un parámetro de la sintaxis abstracta.

Se advierte a los usuarios de ASN.1 que, en este caso, un emisor conforme puede transmitir a un receptor conforme caracteres que no pueden ser tratados por el receptor, porque quedan fuera del subconjunto adoptado (dependiente de la implementación) o del nivel del receptor, y se recomienda que se incluya entonces una especificación de tratamiento de excepción en la definición de **ISO-10646-String**.

EJEMPLO

```
ISO-10646-String {UniversalString : ImplementorsSubset, ImplementationLevel} ::=
            UniversalString (FROM((ImplementorsSubset UNION BasicLatin)
                        INTERSECTION ImplementationLevel) !characterSetProblem)
-- The adopted subset of ISO/IEC 10646-1 shall include "BasicLatin", but
-- may also include any additional characters specified in
-- "ImplementorsSubset", which is a parameter of the abstract syntax.
-- "ImplementationLevel", which is a parameter of the abstract
-- syntax defines the implementation level. A conforming receiver must be
-- prepared to recieve characters outside of its adopted subset and
-- implementation level.  In this case the exception handling specified in
-- clause <add your clause number here> for "characterSetProblem" is
-- invoked. Note that this can never be invoked by a conforming
-- receiver if the actual characters used in an instance of communication
-- are restricted to "BasicLatin".

My-Level2-String ::= ISO-10646-String { { HebrewExtended UNION Hiragana }, Level2 }
```

F.6 El tipo CHARACTER STRING

F.6.1 El tipo **CHARACTER STRING** da completa flexibilidad en la elección del conjunto de caracteres y del método de codificación.

NOTA – Cuando una conexión única proporcione transferencia de datos de extremo a extremo sin retransmisión y se estén empleando los protocolos OSI, la negociación de los juegos de caracteres que han de utilizarse y de su codificación puede efectuarse como parte de la definición de los contextos de presentación de OSI para sintaxis abstractas de caracteres. De no ser así, la sintaxis de caracteres abstracta y de transferencia (repertorio y codificaciones de caracteres) se anuncian mediante un par de valores de identificadores de objetos.

F.6.2 En términos formales una sintaxis abstracta de caracteres es una sintaxis abstracta ordinaria con algunas restricciones en cuanto a los posibles valores (todos son cadenas de caracteres y, por cierto, cadenas de caracteres formadas a partir de alguna colección de caracteres). Así, la atribución de valores de identificadores de objetos para la sintaxis de caracteres abstracta y de transferencia se efectúan de manera normal.

F.6.3 La codificación de **CHARACTER STRING** anuncia la sintaxis abstracta y la sintaxis de transferencia del repertorio de caracteres que se está utilizando (es decir, el juego de caracteres y la codificación). En los entorno OSI, es posible la negociación estas dos sintaxis.

F.6.4 Se han definido sintaxis abstractas de caracteres (y las correspondientes sintaxis de transferencia de caracteres) en Recomendaciones UIT-T y Normas Internacionales, y pueden definirse sintaxis abstractas de caracteres adicionales (y/o sintaxis de transferencia de caracteres) por parte de cualquier organización capaz de atribuir identificadores de objetos.

F.6.5 En ISO/CEI 10646-1 se define una sintaxis abstracta de caracteres (y se asignan identificadores de objetos) para la colección completa de caracteres, para cada una de las colecciones definidas de caracteres para subconjuntos (LATÍN BÁSICO, SÍMBOLOS BÁSICOS, etc.), y para toda posible combinación de las colecciones definidas de caracteres. Se definen también dos sintaxis de transferencia de caracteres para identificar las diversas opciones (en particular, 16 bits y 32 bits) en ISO/CEI 10646-1.

Anexo G

Suplemento didáctico sobre el modelo ASN.1 de extensión de tipo

(Este anexo no es parte integrante de esta Recomendación | Norma Internacional)

G.1 Visión general

G.1.1 Puede ocurrir que un tipo ASN.1 evolucione en el tiempo desde un tipo **raíz de extensión** mediante una serie de extensiones denominadas **adiciones de extensión**.

G.1.2 Un tipo ASN.1 disponible para una implementación determinada puede ser el tipo raíz de extensión o el tipo raíz de extensión junto con uno más adiciones de extensión. Cada uno de estos tipos ASN.1 que contiene una adición de extensión también incluye todas las adiciones de extensión definidas con anterioridad.

G.1.3 Las definiciones de tipos ASN.1 de esta serie se dice que están **relacionadas por extensión** (véase 3.6.32 para una definición más precisa de "relacionada por extensión") y se precisan reglas de codificación para codificar tipos relacionados en extensión de manera que si dos sistemas están utilizando dos tipos diferentes que están relacionados en extensión, las transmisiones entre los dos sistemas transfieran con éxito el contenido de la información de las partes de los tipos relacionados en extensión comunes a ambos sistemas. También se requiere que aquellas partes no comunes a ambos sistemas puedan delimitarse y retransmitirse (quizás a un tercero) en una transmisión posterior, siempre que se utilice la misma sintaxis de transferencia.

NOTA – El emisor puede estar utilizando un tipo que se encuentre antes o después en la serie de adiciones de extensión.

G.1.4 La serie de tipos obtenidos por adición progresiva a un tipo raíz se denomina **serie de extensión**. Para que las reglas de codificación tengan debidamente en cuenta las transmisiones de tipos relacionados en extensión (lo que puede necesitar más bits en la línea), dichos tipos (incluido el tipo raíz de extensión) necesitan estar marcados sintácticamente. La marca unos puntos suspensivos (**...**) que se denomina **marcador de extensión**.

EJEMPLOS

```
Extension root type     1st extension       2nd extension        3rd extension

A ::= SEQUENCE {        A ::= SEQUENCE {    A ::= SEQUENCE {     A ::= SEQUENCE {
      a INTEGER,          a INTEGER,          a INTEGER,           a INTEGER,
        ...                 ...,                ...,                 ...,
      }                   b BOOLEAN,          b BOOLEAN,           b BOOLEAN,
                          c INTEGER           c INTEGER,           c INTEGER,
                        }                     d SEQUENCE {         d SEQUENCE {
                                                e INTEGER,           e INTEGER,
                                                  ...,                 ...,
                                                  ...,               g BOOLEAN OPTIONAL,
                                                f IA5String          h BMPString,
                                              }                        ...,
                                            }                        f IA5String
                                                                   }
                                                                 }
```

G.1.5 Todas las adiciones de extensión en la secuencia, tipos conjunto y elección, están insertadas entre pares de marcadores de extensión. Solamente se permite un marcador de extensión simple si (en el tipo raíz de extensión) éste aparece como el último elemento del tipo, en cuyo caso se supone que existe un marcador de extensión de adaptación inmediatamente antes del corchete de cierre del tipo; en tales casos todas las adiciones de extensión se insertan al final del tipo.

G.1.6 Un tipo con un marcador de extensión puede anidarse dentro de un tipo sin ninguno, o puede anidarse dentro de un tipo en una raíz de extensión, o dentro de un tipo adición de extensión. En estos casos las series de extensión se tratan independientemente y el tipo anidado con el marcador de extensión no produce efecto sobre el tipo dentro del cual está anidado. Sólo puede aparecer un **punto de inserción de extensión** (el final del tipo si se utiliza un marcador de extensión único, o inmediatamente antes del segundo marcador de extensión si se utiliza un par de marcadores de extensión) en una construcción específica.

G.1.7 Una adición de extensión nueva en la serie de extensión se define en términos de un único **grupo adición de extensión** (uno o más tipos anidados dentro de "**[[**" "**]]**") o un único tipo añadido en el punto de inserción de extensión. En el ejemplo a continuación la primera extensión define un grupo adición de extensión en el cual **b** y **c** deben estar ambos presentes o ambos ausentes en un valor de tipo **A**. La segunda extensión define un tipo componente único, **d**, que puede estar ausente en un valor de tipo **A**. La tercera extensión define un grupo adición de extensión en el que **h** debe estar presente en un valor de tipo **A** siempre que el grupo adición de extensión recién añadido se encuentre presente en el valor.

EJEMPLO

```
Extension root type     1st extension       2nd extension          3rd extension

A ::= SEQUENCE {        A ::= SEQUENCE {    A ::= SEQUENCE {    A ::= SEQUENCE {
      a INTEGER,              a INTEGER,          a INTEGER,          a INTEGER,
        ...                     ...,                ...,                ...,
      }                       [[                  [[                  [[
                                b BOOLEAN,          b BOOLEAN,          b BOOLEAN,
                                c INTEGER           c INTEGER           c INTEGER
                              ]]                  ]],                 ]],
                            }                     d SEQUENCE {        d SEQUENCE {
                                                    e INTEGER,          e INTEGER,
                                                      ...,                ...,
                                                      ...,              [[
                                                    f IA5String         g BOOLEAN
      OPTIONAL,                                                       h BMPString
                                                    }                 ]],
                                                  }                     ...,
                                                                      f IA5String
                                                                    }
                                                                  }
```

G.1.8 También es posible añadir el número de versión a los corchetes de versión, pero sólo si está presente en todos los corchetes dentro de un módulo, y solamente si todas las extensiones en el módulo están dentro de corchetes de versión. Se recomienda utilizar números de versión. La posibilidad de omitir números y corchetes de versión se debe a razones históricas. (No se permitieron corchetes de versión ni números de versión en las anteriores versiones de esta Recomendación | Norma Internacional.) (Véase también G.3.)

G.1.9 Aunque lo normal sea añadir las adiciones de extensión en el tiempo, el modelo ASN.1 y la especificación subyacentes no consideran el tiempo. Dos tipos están relacionados en extensión si uno puede ser "acrecentado" a partir del otro mediante adiciones de extensión. Es decir, uno contiene todos los componentes del otro. Pueden existir tipos que tienen que ser "acrecentados" en la dirección opuesta (aunque esto es poco probable). Puede incluso ocurrir que, con el tiempo, un tipo *empiece* con muchas adiciones de extensión que se quitaron progresivamente! De lo único que se ocupan la ASN.1 y sus reglas de codificación es de si un par de especificaciones de tipo están relacionadas por extensión o no. Si lo están, entonces *todas* las reglas de codificación ASN.1 asegurarán el interfuncionamiento entre sus usuarios.

G.1.10 Empezamos con un tipo y luego decidimos si vamos a querer interfuncionar con versiones anteriores y tenemos que ampliarlo posteriormente. Si es así, incluimos el marcador de extensión *ahora* (*now*). Podemos posteriormente añadir adiciones de extensión al tipo manejando de una manera definida los valores extendidos por sistemas anteriores. Sin embargo es importante destacar que la adición de un marcador de extensión a un tipo que no lo tenía (o la supresión de un marcador de extensión) puede impedir el interfuncionamiento.

NOTA – Cuando se utiliza ECN, existe la posibilidad de añadir extensiones de la versión 2 en aquellos sitios que no tenían marcadores de extensión de la versión 1 manteniendo, no obstante, el interfuncionamiento entre las versiones 1 y 2.

G.1.11 El cuadro G.1 muestra los tipos ASN.1 que pueden formar el tipo raíz de extensión de una serie de extensión ASN.1 y la naturaleza de la adición de extensión única permitida para este tipo (por supuesto se pueden formar múltiples adiciones de extensión en sucesión o juntas, como un grupo de extensión).

Cuadro G.1 – Adiciones de extensión

Tipo raíz de extensión	Tipo raíz de extensión
ENUMERATED	Adición de una sola enumeración al final de las "AdditionalEnumeration" con un valor de enumeración mayor que el de cualquier enumeración ya añadida.
SEQUENCE y **SET**	Adición de una sola enumeración al final de las "AdditionalEnumeration" con un valor de enumeración mayor que el de cualquier enumeración ya añadida.
CHOICE	Adición de un solo "NamedType" al final de la "ExtensionAdditionAlternativesList".
Notación constricción	Adición de una sola "AdditionalElementSetSpec" a la notación "ElementSetSpecs".

G.2 Significado de los números de versión

G.2.1 En las codificaciones BER o PER no se utilizan números de versión. Su utilización (de haberla) en codificaciones ECN la determina la especificación ECN.

G.2.2 Los números de versión son más útiles cuando se relacionan con los medios para decodificar una PDU completa, y no con un tipo individual. Cuando se trata de un tipo que se utiliza como un componente de varios protocolos y por lo tanto contribuye a distintas PDU completas, una adición a ese tipo requerirá normalmente que se incremente el número de versión para todas las PDU a las cuales contribuye.

G.2.3 Cuando se utiliza para proporcionar interfuncionamiento entre sistemas desplegados, los números de versión se deben utilizar sobre grupos de adición de extensión de tal manera que los sistemas desplegados tengan conocimiento de la sintaxis y la semántica para todos los grupos de adición de extensión con un determinado número de versión (cualquiera que sea el lugar en que aparezcan dentro del protocolo), y de todos los grupos de adición de extensión con un número de versión anterior. Los especificadores de ECN suponen normalmente que se han atribuido números de versión (a todas las partes de los tipos a los cuales se aplica ECN) de conformidad con este principio.

G.3 Requisitos sobre reglas de codificación

G.3.1 Se puede definir una sintaxis abstracta como los valores de un tipo ASN.1 único que sea un tipo extensible. Incluye entonces todos los valores que pueden obtenerse por adición o supresión de adiciones de extensión. Esta sintaxis abstracta se denomina una sintaxis abstracta relacionada por extensión.

G.3.2 Un conjunto de reglas de codificación bien formado para una sintaxis abstracta relacionada por extensión cumple los requisitos adicionales indicados en G.3.3 a G.3.5.

NOTA – Todas las reglas de codificación ASN.1 cumplen estos requisitos.

G.3.3 La definición de los procedimientos para transformar un valor abstracto en una codificación para transferencia, y para transformar una codificación recibida en un valor abstracto deberá reconocer la posibilidad de que el emisor y el receptor estén utilizando sintaxis abstractas que no sean idénticas pero que estén relacionadas por extensión.

G.3.4 En este caso, las reglas de codificación deberán garantizar que el emisor tenga una especificación de tipo que sea anterior, en la serie de extensión, a la del receptor, los valores del emisor se transferirán de manera que el receptor pueda determinar que no hay adiciones de extensión.

G.3.5 Las reglas de codificación deberán garantizar que, cuando el emisor tenga una especificación de tipo que sea posterior, en la serie de extensión, a la del receptor, sea posible la transferencia de valores de este tipo al receptor.

G.4 Combinación de constricciones (que pueden ser extensibles)

G.4.1 Modelo

G.4.1.1 El modelo ASN.1 básico para aplicar constricciones es sencillo: Un tipo es un conjunto de valores abstractos, y una aplicación que se aplica a dicho tipo selecciona un subconjunto de esos valores abstractos. Si el tipo no constreñido no fuera extensible, en ese caso, el tipo resultante se define como extensible únicamente si la constricción aplicada se define como extensible.

G.4.1.2 Aun en este caso simple, es necesario aclarar una característica: Un tipo puede ser extensible formalmente, aun cuando no puedan haber nunca adiciones de extensión. Considérese lo siguiente:

```
A ::= INTEGER (MIN .. MAX, ... , 1..10)
```

Como sucede en muchos ejemplos de este anexo, esto es algo que nadie escribiría, no obstante los fabricantes de herramientas tienen que escribir el código correspondiente debido a que la norma ASN.1 es sencilla y genérica y este ejemplo es, por consiguiente, una especificación ASN.1 válida. En este ejemplo, A es formalmente un **INTEGER** (ENTERO) extensible, con toda la gama de valores enteros en la raíz.

G.4.1.3 La complejidad se debe principalmente a tres motivos:

- La aplicación de una constricción a un tipo al que ya se ha aplicado una constricción extensible (aplicación de constricciones en serie – véase G.4.2).
- La combinación de constricciones extensibles utilizando **UNION** e **INTERSECTION** y **EXCEPT** (aritmética de conjuntos – véase G.4.3).
- la utilización de una referencia de tipo (un subtipo contenido) en la aritmética de conjuntos de una constricción, cuando la referencia de tipo suprime la referencia a un tipo extensible (probablemente con adiciones de extensión reales – véase G.4.4).

G.4.2 Aplicación de constricciones en serie

G.4.2.1 La aplicación de constricciones en serie se produce cuando un tipo es constreñido (en una asignación a una referencia de tipo) y posteriormente la referencia de tipo se utiliza con una nueva constricción que se aplica a ese tipo.

G.4.2.2 Asimismo, puede suceder, aunque menos comúnmente, cuando a un tipo se le aplican diversas constricciones directamente en serie. Esta última forma se utiliza en muchos de los ejemplos de este anexo (para facilitar la exposición), aunque en las especificaciones reales, la forma que suele adoptar la aplicación en serie es la de una referencia de tipo que vincula a las dos (o más) constricciones.

G.4.2.3 Hay dos puntos esenciales en la aplicación de constricciones en serie:

– Si un tipo constreñido es extensible (y quizá extendido), se descartan la marca "extensible" y todas las adiciones de extensión si posteriormente se aplica en serie una nueva constricción. La extensibilidad de un tipo constreñido (y de las posibles adiciones de extensión) depende únicamente de la última constricción aplicada, que puede referenciar sólo valores en la raíz del tipo que está siendo constreñido nuevamente (el tipo progenitor). Los valores incluidos en la raíz o en las adiciones de extensión del tipo resultante sólo pueden ser aquellos que estén en la raíz del tipo progenitor.

– La aplicación en serie de constricciones (en casos complejos) no es igual a una intersección de la aritmética de conjuntos, aun cuando no intervenga la extensibilidad. En primer lugar, el entorno en el que se interpretan **MIN** y **MAX**, y en segundo lugar los valores abstractos que pueden ser referenciados en la segunda constricción son muy diferentes en el caso de una aplicación en serie y en el que la que las dos constricciones se especifican como una intersección de valores de un progenitor común.

NOTA – La utilización de un rango como **20..28** en una constricción de un tipo entero es legal si (y sólo si) tanto **20** como **28** existen en (la raíz del) el tipo progenitor, pero los valores referenciados por esta especificación de rango son sólo aquellos en (la raíz de) el progenitor. Por tal razón, si el progenitor ya ha sido constreñido de modo que excluya los valores **24** y **25**, el rango **20..28** hace referencia sólo a **20** a **23** y **26** a **28**.

A continuación se presentan algunos ejemplos:

```
A1 ::= INTEGER (1..32, ... , 33..128)
      -- A1 is extensible, and contains values 1 to 128 with 1 to 32
      -- in the root and 33 to 128 as extension additions.

B1 ::= A1 (1..128)
      -- or equivalently

B1 ::= INTEGER (1..32, ... , 33..128) (1..128)
      -- These are illegal, as 128 is not in the parent, which
      -- lost its extension additions when it was further constrained

B2 ::= A1 (1..16)
      -- This is legal. B2 is not extensible, and contains 1 to 16.

A2 ::= INTEGER (1..32) (MIN .. 63)
      -- MIN is 1, and 63 is illegal

A3 ::= INTEGER ( (1..32) INTERSECTION (MIN..63) )
      -- This is legal.  MIN is minus infinity and A3 contains 1 to 32
```

G.4.3 Utilización de la aritmética de conjuntos

G.4.3.1 Los resultados son en gran parte intuitivos, y siguen las reglas matemáticas normales en cuanto a la intersección, unión y diferencia de conjuntos (**EXCEPT**). En particular, la intersección y la unión son conmutativas, es decir:

```
( <some set 1 of values> INTERSECTION <some set 2 of values> )
```

es igual a

```
( <some set 2 of values> INTERSECTION <some set 1 of values> )
```

del mismo modo para **UNION**.

G.4.3.2 La conmutatividad es verdadera, independientemente de qué conjuntos sean extensibles, y de qué adiciones de extensión estén presentes.

G.4.3.3 Si una intersección imposibilita que ocurran valores de adición de extensión podrían producirse algunos errores de comprensión. Esto es similar al caso de **INTEGER (MIN..MAX, ...)**.

G.4.3.4 Por ejemplo:

```
A ::= INTEGER ((1..256, ... , B) INTERSECTION (1..256))
-- A always contains (only) the values 1..256, no matter what values
-- B contains, but is nonetheless formally extensible
```

G.4.3.5 También es importante recordar que mientras que los progenitores pierden su extensibilidad y sus adiciones de extensión cuando se constriñen nuevamente, y los subtipos contenidos pierden su extensibilidad y sus adiciones de

extensión, los conjuntos de valores especificados directamente en la aritmética de conjuntos no pierden ni su extensibilidad ni sus adiciones de extensión.

G.4.3.6 Las reglas de extensibilidad de los conjuntos de valores producidos por la aritmética de conjuntos se establecen con claridad en 46.3 y 46.4, y no dependen de que la aritmética de conjuntos posibilite las adiciones de extensión reales o no las posibilite.

G.4.3.7 En esta cláusula se resumen las reglas por cuestiones de integridad, utilizando **E** para denotar un conjunto de valores con el marcador "extensible" fijado y **N** para denotar un conjunto de valores que formalmente no son extensibles. Los valores en la raíz de cada conjunto se denotan mediante **R**, y las adiciones de extensión (si las hubiere) mediante **X**, y los contenidos del resultado se muestran en cada caso.

NOTA 1 – A los efectos de este anexo y para facilitar la exposición, si un conjunto de valores no es extensible, se describen todos sus valores como valores raíz.

NOTA 2 – Si la raíz de cualquier conjunto de valores resultante utilizado en una constricción aplicada en serie está vacía, se trata de una especificación ilegal.

NOTA 3 – Para evitar verbosidad a continuación, se utiliza "Extensions" en lugar del término más correcto "Extension additions".

G.4.3.8 Las reglas son:

```
N1 INTERSECTION N2 => N
        Root: R1 INTERSECTION R2
N1 INTERSECTION E2 => E
        Root: R1 INTERSECTION R2, Extensions: R1 INTERSECTION X2
E1 INTERSECTION E2 => E
        Root: R1 INTERSECTION R2, Extensions: ((R1 UNION X1)
                                               INTERSECTION
                                               (R2 UNION X2))
                                               EXCEPT
                                               (R1 INTERSECTION R2)
N1 UNION N2 => N
        Root: R1 UNION R2
N1 UNION E2 => E
        Root: R1 UNION R2, Extensions: X2
E1 UNION E2 => E
        Root: R1 UNION R2, Extensions: (R1 UNION X1 UNION R2 UNION X2)
                                       EXCEPT
                                       (R1 UNION R2)
N1 EXCEPT N2 => N
        Root: R1 EXCEPT R2
N1 EXCEPT E2 => N
        Root: R1 EXCEPT R2
E1 EXCEPT N2 => E
        Root: R1 EXCEPT R2, Extensions: (X1 EXCEPT R2)
                                         EXCEPT
                                         (R1 EXCEPT R2)
E1 EXCEPT E2 => E
        Root: R1 EXCEPT R2, Extensions: (X1 EXCEPT (R2 UNION X2) )
                                         EXCEPT
                                         (R1 EXCEPT R2)
N1 ... N2 => E
        Root: R1, Extensions: R2 EXCEPT R1
E1 ... N2 => E
        Root: R1, Extensions: X1 UNION R2
                              EXCEPT
                              R1
N1 ... E2 => E
        Root: R1, Extensions: R2 UNION X2
                              EXCEPT
                              R1
E1 ... E2 => E
        Root: R1, Extensions: X1 UNION R2 UNION E2
                              EXCEPT
                              R1
```

NOTA – Si el resultado de la aplicación de la aritmética de conjuntos a conjuntos extensibles de valores no tiene adiciones de extensión reales, o incluso no puede tenerlas nunca (independientemente de las adiciones de extensión que se añadan a las entradas extensibles), el resultado sigue definiéndose formalmente como extensible para los resultados **E** antes mencionados.

G.4.4 Utilización de la notación de subtipo contenido

Un subtipo contenido puede ser extensible o no, pero cuando se utiliza en una aritmética de conjuntos se trata siempre como no extensible y se descartan todas sus adiciones de extensión.

Anexo H

Resumen de la notación ASN.1

(Este anexo no es parte integrante de esta Recomendación | Norma Internacional)

Los siguientes elementos léxicos se definen en la cláusula 11:

typereference
identifier
valuereference
modulereference
comment
empty
number
realnumber
bstring
hstring
cstring
xmlbstring
xmlhstring
xmlcstring
xmlasn1typename
"true"
"false"
"::="
"[["
"]]"
".."
"..."
"</"
"/>"
"{"
"}"
"<"
">"
","
"."
"("
")"
"["
"]"
"-"
":"
"="
""" (QUOTATION MARK)
"'" (APOSTROPHE)
" " (SPACE)
";"
"@"
"|"
"!"
"^"

ABSENT
ABSTRACT-SYNTAX
ALL
APPLICATION
AUTOMATIC
BEGIN
BIT
BMPString
BOOLEAN
BY
CHARACTER
CHOICE
CLASS
COMPONENT
COMPONENTS
CONSTRAINED
CONTAINING
DEFAULT
DEFINITIONS
EMBEDDED
ENCODED
END
ENUMERATED
EXCEPT
EXPLICIT
EXPORTS
EXTENSIBILITY
EXTERNAL
FALSE
FROM
GeneralizedTime
GeneralString
GraphicString
IA5String
IDENTIFIER
IMPLICIT
IMPLIED
IMPORTS
INCLUDES
INSTANCE
INTEGER
INTERSECTION
ISO646String
MAX
MIN
MINUS-INFINITY
NULL
NumericString
OBJECT
ObjectDescriptor
OCTET
OF
OPTIONAL
PATTERN
PDV
PLUS-INFINITY
PRESENT
PrintableString
PRIVATE
REAL
RELATIVE-OID
SEQUENCE
SET
SIZE
STRING
SYNTAX

T61String	TYPE-IDENTIFIER	UTCTime
TAGS	UNION	UTF8String
TeletexString	UNIQUE	VideotexString
TRUE	UNIVERSAL	VisibleString
	UniversalString	WITH

En esta Recomendación | Norma Internacional se utilizan las siguientes producciones con los elementos léxicos mencionados como símbolos terminales:

```
ModuleDefinition ::= ModuleIdentifier
                     DEFINITIONS
                     TagDefault
                     ExtensionDefault
                     "::="
                     BEGIN
                     ModuleBody
                     END

ModuleIdentifier ::= modulereference
                         DefinitiveIdentifier

DefinitiveIdentifier ::= "{" DefinitiveObjIdComponentList "}"
    |                    empty

DefinitiveObjIdComponentList ::=
          DefinitiveObjIdComponent
    |     DefinitiveObjIdComponent DefinitiveObjIdComponentList

DefinitiveObjIdComponent ::=
          NameForm
    |     DefinitiveNumberForm
    |     DefinitiveNameAndNumberForm

DefinitiveNumberForm     ::= number

DefinitiveNameAndNumberForm ::= identifier "(" DefinitiveNumberForm ")"

TagDefault ::=
          EXPLICIT TAGS
    |     IMPLICIT TAGS
    |     AUTOMATIC TAGS
    |     empty

ExtensionDefault ::=
          EXTENSIBILITY IMPLIED | empty

ModuleBody ::=   Exports Imports AssignmentList
    |            empty

Exports ::=      EXPORTS SymbolsExported ";"
    |            EXPORTS ALL ";"
    |            empty

SymbolsExported ::= SymbolList
    |               empty

Imports ::=      IMPORTS SymbolsImported ";"
    |            empty

SymbolsImported ::= SymbolsFromModuleList
    |               empty
```

```
SymbolsFromModuleList ::=
                    SymbolsFromModule
    |               SymbolsFromModuleList  SymbolsFromModule

SymbolsFromModule ::=    SymbolList FROM GlobalModuleReference

GlobalModuleReference ::= modulereference AssignedIdentifier

AssignedIdentifier ::=  ObjectIdentifierValue
    |                   DefinedValue
    |                   empty

SymbolList   ::= Symbol | SymbolList "," Symbol

Symbol  ::= Reference | ParameterizedReference

Reference ::=
          typereference
    |     valuereference
    |     objectclassreference
    |     objectreference
    |     objectsetreference

AssignmentList ::=          Assignment  | AssignmentList Assignment

Assignment ::=
          TypeAssignment
    |     ValueAssignment
    |     XMLValueAssignment
    |     ValueSetTypeAssignment
    |     ObjectClassAssignment
    |     ObjectAssignment
    |     ObjectSetAssignment
    |     ParameterizedAssignment

DefinedType ::=
          ExternalTypeReference
    |     typereference
    |     ParameterizedType
    |     ParameterizedValueSetType

ExternalTypeReference ::=
                    modulereference
                    "."
                    typereference

NonParameterizedTypeName ::=
          ExternalTypeReference
    |     typereference
    |     xmlasn1typename

DefinedValue ::=
          ExternalValueReference
    |     valuereference
    |     ParameterizedValue

ExternalValueReference ::=
                    modulereference
                    "."
                    valuereference

AbsoluteReference ::=
                    "@" ModuleIdentifier
                    "."
                    ItemSpec

ItemSpec ::=
          typereference
|         ItemId "." ComponentId
```

```
ItemId ::= ItemSpec

ComponentId ::=
        identifier | number | "*"

TypeAssignment ::=    typereference
                      "::="
                      Type

ValueAssignment ::=   valuereference
                      Type
                      "::="
                      Value

XMLValueAssignment ::=
        valuereference
        "::="
        XMLTypedValue

XML     TypedValue ::=
        "<" & NonParameterizedTypeName  ">"
        XMLValue
        "</" & NonParameterizedTypeName ">"
    |   "<" & NonParameterizedTypeName "/>"

ValueSetTypeAssignment ::=    typereference
                      Type
                      "::="
                      ValueSet

ValueSet ::= "{" ElementSetSpecs "}"

Type ::= BuiltinType | ReferencedType | ConstrainedType

BuiltinType ::=
        BitStringType
    |   BooleanType
    |   CharacterStringType
    |   ChoiceType
    |   EmbeddedPDVType
    |   EnumeratedType
    |   ExternalType
    |   InstanceOfType
    |   IntegerType
    |   NullType
    |   ObjectClassFieldType
    |   ObjectIdentifierType
    |   OctetStringType
    |   RealType
    |   RelativeOIDType
    |   SequenceType
    |   SequenceOfType
    |   SetType
    |   SetOfType
    |   TaggedType

NamedType ::= identifier   Type

ReferencedType ::=
        DefinedType
    |   UsefulType
    |   SelectionType
    |   TypeFromObject
    |   ValueSetFromObjects

Value  ::=   BuiltinValue | ReferencedValue | ObjectClassFieldValue

XMLValue ::= XMLBuiltinValue | XMLObjectClassFieldValue
```

```
BuiltinValue ::=
        BitStringValue
    |   BooleanValue
    |   CharacterStringValue
    |   ChoiceValue
    |   EmbeddedPDVValue
    |   EnumeratedValue
    |   ExternalValue
    |   InstanceOfValue
    |   IntegerValue
    |   NullValue
    |   ObjectIdentifierValue
    |   OctetStringValue
    |   RealValue
    |   RelativeOIDValue
    |   SequenceValue
    |   SequenceOfValue
    |   SetValue
    |   SetOfValue
    |   TaggedValue

XMLBuiltinValue ::=
        XMLBitStringValue
    |   XMLBooleanValue
    |   XMLCharacterStringValue
    |   XMLChoiceValue
    |   XMLEmbeddedPDVValue
    |   XMLEnumeratedValue
    |   XMLExternalValue
    |   XMLInstanceOfValue
    |   XMLIntegerValue
    |   XMLNullValue
    |   XMLObjectIdentifierValue
    |   XMLOctetStringValue
    |   XMLRealValue
    |   XMLRelativeOIDValue
    |   XMLSequenceValue
    |   XMLSequenceOfValue
    |   XMLSetValue
    |   XMLSetOfValue
    |   XMLTaggedValue

ReferencedValue ::=
        DefinedValue
    |   ValueFromObject

NamedValue ::= identifier Value

XMLNamedValue ::=
        "<" & identifier ">" XMLValue "</" & identifier ">"

BooleanType ::=BOOLEAN

BooleanValue::= TRUE | FALSE

XMLBooleanValue ::=
        "<" & "true" "/>"
    |   "<" & "false" "/>"

IntegerType ::=
        INTEGER
    |   INTEGER "{" NamedNumberList "}"

NamedNumberList ::=
        NamedNumber
    |   NamedNumberList "," NamedNumber
```

```
NamedNumber ::=
        identifier "(" SignedNumber ")"
    |   identifier "(" DefinedValue ")"

SignedNumber ::= number | "-" number

IntegerValue ::=  SignedNumber | identifier

XMLIntegerValue ::=
        SignedNumber
    |   "<" & identifier "/>"

EnumeratedType ::=
        ENUMERATED "{" Enumerations "}"

Enumerations ::= RootEnumeration
    |   RootEnumeration  ","  "..." ExceptionSpec
    |   RootEnumeration  ","  "..." ExceptionSpec  ","  AdditionalEnumeration

RootEnumeration ::= Enumeration

AdditionalEnumeration ::= Enumeration

Enumeration ::= EnumerationItem | EnumerationItem "," Enumeration

EnumerationItem ::= identifier | NamedNumber

EnumeratedValue  ::= identifier

XMLEnumeratedValue ::= "<" & identifier "/>"

RealType ::= REAL

RealValue ::=
        NumericRealValue | SpecialRealValue

NumericRealValue ::=
        realnumber
    |   "-" realnumber
    |   SequenceValue              -- Value of the associated sequence type

SpecialRealValue ::=
        PLUS-INFINITY | MINUS-INFINITY

XMLRealValue ::=
        XMLNumericRealValue | XMLSpecialRealValue

XMLNumericRealValue ::=
        realnumber
    |   "-" realnumber

XMLSpecialRealValue ::=
        "<" & PLUS-INFINITY "/>" | "<" & MINUS-INFINITY "/>"

BitStringType   ::=         BIT STRING | BIT STRING "{" NamedBitList "}"

NamedBitList::=       NamedBit | NamedBitList "," NamedBit

NamedBit    ::=  identifier "(" number ")"
    |            identifier "(" DefinedValue ")"

BitStringValue  ::=    bstring | hstring | "{" IdentifierList "}" | "{"  "}" | CONTAINING Value

IdentifierList  ::=    identifier | IdentifierList "," identifier

XMLBitStringValue ::=
        XMLTypedValue
    |   xmlbstring
    |   XMLIdentifierList
    |   empty
```

```
XMLIdentifierList ::=
        "<" & identifier "/>"
    |   XMLIdentifierList "<" & identifier "/>"

OctetStringType ::= OCTET STRING

OctetStringValue ::= bstring | hstring | CONTAINING Value

XMLOctetStringValue ::=
        XMLTypedValue
    |   xmlhstring

NullType    ::= NULL

NullValue   ::= NULL

XMLNullValue ::= empty

SequenceType ::=
        SEQUENCE "{" "}"
    |   SEQUENCE "{" ExtensionAndException OptionalExtensionMarker "}"
    |   SEQUENCE "{" ComponentTypeLists "}"

ExtensionAndException ::= "..." | "..." ExceptionSpec

OptionalExtensionMarker ::= "," "..." | empty

ComponentTypeLists ::=
        RootComponentTypeList
    |   RootComponentTypeList "," ExtensionAndException ExtensionAdditions
            OptionalExtensionMarker
    |   RootComponentTypeList "," ExtensionAndException ExtensionAdditions
            ExtensionEndMarker "," RootComponentTypeList
    |   ExtensionAndException ExtensionAdditions ExensionEndMarker ","
            RootComponentTypeList
    |   ExtensionAndException ExtensionAdditions OptionalExtensionMarker

RootComponentTypeList ::= ComponentTypeList

ExtensionEndMarker ::= "," "..."

ExtensionAdditions ::= "," ExtensionAdditionList | empty

ExtensionAdditionList ::= ExtensionAddition
    |   ExtensionAdditionList "," ExtensionAddition

ExtensionAddition ::= ComponentType | ExtensionAdditionGroup

ExtensionAdditionGroup ::= "[[" VersionNumber ComponentTypeList "]]"

VersionNumber ::= empty | number ":"

ComponentTypeList ::=   ComponentType
    |                   ComponentTypeList "," ComponentType

ComponentType ::=
        NamedType
    |   NamedType OPTIONAL
    |   NamedType DEFAULT Value
    |   COMPONENTS OF Type

SequenceValue   ::=   "{" ComponentValueList "}" | "{" "}"

ComponentValueList ::=
        NamedValue
    |   ComponentValueList "," NamedValue

XMLSequenceValue ::=
        XMLComponentValueList
    |   empty
```

```
XMLComponentValueList ::=
        XMLNamedValue
    |   XMLComponentValueList XMLNamedValue

SequenceOfType  ::=  SEQUENCE OF Type | SEQUENCE OF NamedType

SequenceOfValue ::=  "{" ValueList "}" | "{" NamedValueList "}" | "{"  "}"

ValueList   ::=    Value | ValueList "," Value

XMLSequenceOfValue ::=
        XMLValueList
    |   XMLDelimitedItemList
    |   XMLSpaceSeparatedList
    |   empty

XMLValueList ::=
        XMLValueOrEmpty
    |   XMLValueOrEmpty XMLValueList

XMLValueOrEmpty ::=
        XMLValue
    |   "<" & NonParameterizedTypeName "/>"

XMLSpaceSeparatedList ::=
        XMLValueOrEmpty
    |   XMLValueOrEmpty " " XMLSpaceSeparatedList

XMLDelimitedItemList ::=
        XMLDelimitedItem
    |   XMLDelimitedItem XMLDelimitedItemList

XMLDelimitedItem ::=
        "<" & NonParameterizedTypeName ">" XMLValue
            "</" & NonParameterizedTypeName ">"
    |   "<" & identifier ">" XMLValue "</" & identifier ">"

SetType ::=  SET "{" "}"
    |        SET "{" ExtensionAndException OptionalExtensionMarker "}"
    |        SET "{" ComponentTypeLists "}"

SetValue    ::=   "{" ComponentValueList "}" | "{"  "}"

XMLSetValue ::= XMLComponentValueList   |   empty

SetOfType   ::=  SET OF Type | SET OF NamedType

SetOfValue  ::=  "{" ValueList "}" | "{" NamedValueList "}" | "{"  "}"

XMLSetOfValue ::=
        XMLValueList
    |   XMLDelimitedItemList
    |   XMLSpaceSeparatedList
    |   empty

ChoiceType  ::= CHOICE "{" AlternativeTypeLists "}"

AlternativeTypeLists ::=
        RootAlternativeTypeList
    |   RootAlternativeTypeList ","
            ExtensionAndException ExtensionAdditionAlternatives OptionalExtensionMarker

RootAlternativeTypeList ::= AlternativeTypeList

ExtensionAdditionAlternatives ::= "," ExtensionAdditionAlternativesList | empty

ExtensionAdditionAlternativesList ::= ExtensionAdditionAlternative
    |       ExtensionAdditionAlternativesList "," ExtensionAdditionAlternative

ExtensionAdditionAlternative ::= ExtensionAdditionAlternativesGroup | NamedType
```

```
ExtensionAdditionAlternativesGroup ::= "[[" VersionNumber AlternativeTypeList "]]"

AlternativeTypeList ::=             NamedType
    |                               AlternativeTypeList "," NamedType

ChoiceValue ::=       identifier ":" Value

XMLChoiceValue ::= "<" & identifier ">" XMLValue "</" & identifier ">"

SelectionType ::=         identifier "<" Type

TaggedType ::= Tag Type
    |                   Tag IMPLICIT Type
    |                   Tag EXPLICIT Type

Tag    ::=              "[" Class ClassNumber "]"

ClassNumber ::= number | DefinedValue

Class ::=   UNIVERSAL
    |       APPLICATION
    |       PRIVATE
    |       empty

TaggedValue ::= Value

XMLTaggedValue ::= XMLValue

EmbeddedPDVType ::=             EMBEDDED PDV

EmbeddedPDVValue ::=            SequenceValue

XMLEmbeddedPDVValue ::= XMLSequenceValue

ExternalType ::= EXTERNAL

ExternalValue ::= SequenceValue

XMLExternalValue ::= XMLSequenceValue

ObjectIdentifierType ::= OBJECT IDENTIFIER

ObjectIdentifierValue ::= "{" ObjIdComponentsList "}"
    |                     "{" DefinedValue ObjIdComponentsList "}"

ObjIdComponentsList ::= ObjIdComponents
    |                   ObjIdComponents ObjIdComponentsList

ObjIdComponents ::= NameForm
    |               NumberForm
    |               NameAndNumberForm
    |               DefinedValue

NameForm    ::=     identifier

NumberForm    ::=       number | DefinedValue

NameAndNumberForm ::=     identifier "(" NumberForm ")"

XMLObjectIdentifierValue ::=
        XMLObjIdComponentList

XMLObjIdComponentList ::=
        XMLObjIdComponent
    |   XMLObjIdComponent & "." & XMLObjIdComponentList

XMLObjIdComponent ::=
        NameForm
    |   XMLNumberForm
    |   XMLNameAndNumberForm

XMLNumberForm ::= number
```

```
XMLNameAndNumberForm ::=
        identifier & "(" & XMLNumberForm & ")"

RelativeOIDType ::=
                RELATIVE-OID

RelativeOIDValue ::=
            "{" RelativeOIDComponentsList "}"

RelativeOIDComponentsList ::=
        RelativeOIDComponents
    |   RelativeOIDComponents RelativeOIDComponentsList

RelativeOIDComponents ::=   NumberForm
    |                       NameAndNumberForm
    |                       DefinedValue

XMLRelativeOIDValue ::=
        XMLRelativeOIDComponentList

XMLRelativeOIDComponentList ::=
        XMLRelativeOIDComponent
    |   XMLRelativeOIDComponent & "." & XMLRelativeOIDComponentList

XMLRelativeOIDComponent ::=
        XMLNumberForm
    |   XMLNameAndNumberForm

CharacterStringType ::= RestrictedCharacterStringType | UnrestrictedCharacterStringType

RestrictedCharacterStringType ::=
        BMPString
    |   GeneralString
    |   GraphicString
    |   IA5String
    |   ISO646String
    |   NumericString
    |   PrintableString
    |   TeletexString
    |   T61String
    |   UniversalString
    |   UTF8String
    |   VideotexString
    |   VisibleString

RestrictedCharacterStringValue ::= cstring | CharacterStringList | Quadruple | Tuple

CharacterStringList  ::=  "{" CharSyms "}"

CharSyms  ::=  CharsDefn | CharSyms "," CharsDefn

CharsDefn  ::=  cstring | Quadruple | Tuple | DefinedValue

Quadruple    ::= "{"  Group  ","  Plane  ","  Row  ","  Cell "}"

Group        ::= number

Plane        ::= number

Row          ::= number

Cell         ::= number

Tuple ::= "{" TableColumn "," TableRow "}"

TableColumn ::= number

TableRow ::= number

XMLRestrictedCharacterStringValue ::= xmlcstring

UnrestrictedCharacterStringType ::= CHARACTER STRING
```

CharacterStringValue ::= RestrictedCharacterStringValue | UnrestrictedCharacterStringValue

XMLCharacterStringValue ::=
XMLRestrictedCharacterStringValue
| XMLUnrestrictedCharacterStringValue

UnrestrictedCharacterStringValue ::= SequenceValue

XMLUnrestrictedCharacterStringValue ::= XMLSequenceValue

UsefulType ::= typereference

The following character string types are defined in 37.1:

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

The following useful types are defined in clauses 42 to 44:

```
GeneralizedTime

UTCTime

ObjectDescriptor
```

The following productions are used in clauses 45 to 47:

ConstrainedType ::=
Type Constraint
| TypeWithConstraint

TypeWithConstraint ::=
SET Constraint OF Type
| SET SizeConstraint OF Type
| SEQUENCE Constraint OF Type
| SEQUENCE SizeConstraint OF Type
| SET Constraint OF NamedType
| SET SizeConstraint OF NamedType
| SEQUENCE Constraint OF NamedType
| SEQUENCE SizeConstraint OF NamedType

Constraint ::= "(" ConstraintSpec ExceptionSpec ")"

ConstraintSpec ::= SubtypeConstraint
| GeneralConstraint

ExceptionSpec ::= "!" ExceptionIdentification | empty

ExceptionIdentification ::= SignedNumber
| DefinedValue
| Type ":" Value

SubtypeConstraint ::= ElementSetSpecs

ElementSetSpecs ::=
RootElementSetSpec
| RootElementSetSpec "," "..."
| RootElementSetSpec "," "..." "," AdditionalElementSetSpec

RootElementSetSpec ::= ElementSetSpec

AdditionalElementSetSpec ::= ElementSetSpec

ElementSetSpec ::= Unions | ALL Exclusions

```
Unions ::=      Intersections
      |         UElems UnionMark Intersections

UElems ::= Unions

Intersections ::=   IntersectionElements
      |             IElems IntersectionMark IntersectionElements

IElems ::= Intersections

IntersectionElements ::= Elements | Elems Exclusions

Elems ::= Elements

Exclusions ::= EXCEPT Elements

UnionMark ::= "|"   |   UNION

IntersectionMark ::= "^" |   INTERSECTION

Elements ::= SubtypeElements
      |      ObjectSetElements
      |      "(" ElementSetSpec ")"

SubtypeElements ::=
            SingleValue
      |     ContainedSubtype
      |     ValueRange
      |     PermittedAlphabet
      |     SizeConstraint
      |     TypeConstraint
      |     InnerTypeConstraints
      |     PatternConstraint

SingleValue   ::= Value

ContainedSubtype ::= Includes Type

Includes ::= INCLUDES | empty

ValueRange ::= LowerEndpoint ".." UpperEndpoint

LowerEndpoint ::= LowerEndValue | LowerEndValue "<"

UpperEndpoint ::= UpperEndValue | "<" UpperEndValue

LowerEndValue ::= Value | MIN

UpperEndValue ::= Value | MAX

SizeConstraint ::= SIZE Constraint

PermittedAlphabet ::= FROM Constraint

TypeConstraint ::= Type

InnerTypeConstraints ::=
            WITH COMPONENT SingleTypeConstraint
      |     WITH COMPONENTS MultipleTypeConstraints

SingleTypeConstraint::= Constraint

MultipleTypeConstraints ::= FullSpecification | PartialSpecification

FullSpecification   ::= "{" TypeConstraints "}"

PartialSpecification ::= "{" "..." "," TypeConstraints "}"

TypeConstraints ::=   NamedConstraint
      |               NamedConstraint "," TypeConstraints

NamedConstraint ::=   identifier ComponentConstraint

ComponentConstraint ::= ValueConstraint PresenceConstraint
```

ValueConstraint ::= Constraint | empty

PresenceConstraint ::= `PRESENT` | `ABSENT` | `OPTIONAL` | empty

PatternConstraint ::= `PATTERN` Value

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie D	Principios generales de tarificación
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedios
Serie I	Red digital de servicios integrados
Serie J	Redes de cable y transmisión de programas radiofónicos y televisivos, y de otras señales multimedios
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	Gestión de las telecomunicaciones, incluida la RGT y el mantenimiento de redes
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de transmisión telefónica, instalaciones telefónicas y redes locales
Serie Q	Conmutación y señalización
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	**Redes de datos, comunicaciones de sistemas abiertos y seguridad**
Serie Y	Infraestructura mundial de la información, aspectos del protocolo Internet y Redes de la próxima generación
Serie Z	Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación

Impreso en Suiza
Ginebra, 2006