



UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

X.680

(12/97)

SERIE X: REDES DE DATOS Y COMUNICACIÓN
ENTRE SISTEMAS ABIERTOS

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

(Anteriormente Recomendación del CCITT)

RECOMENDACIONES DE LA SERIE X DEL UIT-T
REDES DE DATOS Y COMUNICACIÓN ENTRE SISTEMAS ABIERTOS

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

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

NORMA INTERNACIONAL 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 par 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 limitar de ningún modo la manera de codificar la información para la transmisión. Es aplicable especialmente, pero no exclusivamente, a los protocolos de capa de aplicación.

Orígenes

El texto de la Recomendación UIT-T X.680 se aprobó el 12 de diciembre de 1997. Su texto se publica también, en forma idéntica, 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 Conferencia Mundial de Normalización de las Telecomunicaciones (CMNT), 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 N.º 1 de la CMNT.

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 *empresa de explotación reconocida (EER)* designa a toda persona, compañía, empresa u organización gubernamental que explote un servicio de correspondencia pública. Los términos *Administración, EER y correspondencia pública* están definidos en la *Constitución de la UIT (Ginebra, 1992)*.

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.

© UIT 1998

Es propiedad. Ninguna parte de esta publicación puede reproducirse o utilizarse, de ninguna forma o por ningún medio, sea éste electrónico o mecánico, de fotocopia o de microfilm, sin previa autorización escrita por parte de la UIT.

ÍNDICE

Página

1	Alcance	1
2	Referencias normativas	1
2.1	Recomendaciones Normas Internacionales idénticas.....	1
2.2	Pares de Recomendaciones Normas Internacionales de contenido técnico equivalente	2
2.3	Referencias adicionales	2
3	Definiciones.....	2
3.1	Especificación de objeto de información.....	2
3.2	Especificación de construcción	3
3.3	Parametrización de la especificación de ASN.1	3
3.4	Definición del servicio de presentación.....	3
3.5	Especificación del protocolo de presentación.....	3
3.6	Estructura para la identificación de organizaciones.....	3
3.7	Conjunto de caracteres codificados multiocteto universal (UCS)	3
3.8	Definiciones adicionales.....	4
4	Abreviaturas.....	8
5	Notación.....	8
5.1	Producciones	9
5.2	Las colecciones alternativas	9
5.3	Ejemplo de una producción	9
5.4	Disposición.....	9
5.5	Recursión.....	10
5.6	Referencias a una colección de secuencias.....	10
5.7	Referencias a un ítem.....	10
5.8	Notaciones abreviadas	10
6	Modelo ASN.1 de extensión de tipo	11
7	Requisitos de extensibilidad en reglas de codificación.....	11
8	Rótulos.....	12
9	Utilización de la notación ASN.1	13
10	El conjunto (o juego) de caracteres ASN.1	13
11	Ítems ASN.1	14
11.1	Reglas generales	14
11.2	Referencias de tipo	14
11.3	Identificadores	14
11.4	Referencia de valor.....	14
11.5	Referencia de módulo.....	14
11.6	Comentario	14
11.7	Ítem vacío	15
11.8	Ítem número.....	15
11.9	Ítem cadena binaria.....	15
11.10	Ítem cadena hexadecimal.....	15
11.11	Ítem cadena de caracteres.....	15
11.12	Ítem asignación.....	16
11.13	Separador de gama	16

	<i>Página</i>
11.14 Elipsis	16
11.15 Corchetes de versión izquierdos	17
11.16 Corchetes de versión derechos	17
11.17 Ítems carácter único (o carácter individual)	17
11.18 Palabras reservadas.....	17
12 Definición de módulo	18
13 Referenciación de definiciones de tipo y valor	22
14 Notación para soportar referencias a componentes ASN.1	23
15 Asignación de tipos y valores	24
16 Definición de tipos y valores	24
17 Notación para el tipo booleano	27
18 Notación para el tipo entero.....	27
19 Notación para el tipo enumerado	28
20 Notación para el tipo real.....	29
21 Notación para el tipo cadena de bits	29
22 Notación para el tipo cadena de octetos.....	31
23 Notación para el tipo nulo.....	31
24 Notación para tipos secuencia.....	31
25 Notación para tipos secuencia de.....	34
26 Notación para tipos conjunto	35
27 Notación para tipos conjunto de	35
28 Notación para tipos elección.....	36
29 Notación para tipos selección	37
30 Notación para tipos rotulado	37
31 Notación para el tipo identificador de objeto.....	39
32 Notación para el tipo pdv incrustado	40
33 Notación para el tipo externo.....	42
34 Tipos cadena de caracteres	43
35 Notación para tipos cadena de caracteres	44
36 Definición de tipos cadena de caracteres restringida	44
37 Denominación de caracteres y colecciones definidas en ISO/CEI 10646.....	48
38 Orden canónico de los caracteres.....	51
39 Definición de tipos cadena de caracteres no restringida	52
40 Notación para tipos definidos en las cláusulas 41 a 43	53
41 Generalized time (tiempo generalizado) (u hora generalizada)	53
42 Tiempo universal	54
43 Tipo descriptor de objeto.....	55
44 Tipos constreñidos.....	55
45 Identificador de excepción.....	56
46 Especificación de conjunto de elementos	57
47 Marcador de extensión.....	58
48 Elementos subtipos	60

	<i>Página</i>
48.1 Generalidades	60
48.2 Single Value (valor único).....	61
48.3 Contained Subtype (subtipo contenido).....	62
48.4 Value Range (gama de valores).....	62
48.5 Size Constraint (constricción de tamaño)	62
48.6 Constricción de tipo.....	63
48.7 Alfabeto permitido.....	63
48.8 Inner Subtyping (subtipificación interior)	63
Anexo A – Utilización de la notación ASN.1-88/90	65
A.1 Mantenimiento.....	65
A.2 Combinación de la notación ASN.1-88/90 con la notación ASN.1 actual.....	65
A.3 Paso a la notación ASN.1 actual.....	65
Anexo B – Asignación de valores de identificador de objeto	68
Anexo C – Ejemplos y sugerencias	69
C.1 Ejemplo de un registro de personal	69
C.2 Directrices para la utilización de la notación.....	70
C.3 Identificación de sintaxis abstractas	81
C.4 Subtipos.....	82
Anexo D – Suplemento didáctico sobre cadenas de caracteres ASN.1	86
D.1 Soporte de cadenas de caracteres en ASN.1	86
D.2 Los tipos UniversalString, UTF8String y BMPString	86
D.3 Requisitos de conformidad de ISO/CEI 10646-1	87
D.4 Recomendaciones a los usuarios de la ASN.1 sobre conformidad de ISO/CEI 10646-1	87
D.5 Subconjuntos adoptados como parámetros de la sintaxis abstracta	88
D.6 El tipo CHARACTER STRING.....	89
Anexo E – Características reemplazadas.....	90
E.1 Utilización de identificadores ahora obligatoria.....	90
E.2 El valor de elección	90
E.3 El tipo cualquiera.....	90
E.4 La capacidad macro	91
Anexo F – Anexo explicativo del modelo ASN.1 de extensión de tipo	92
F.1 Visión general.....	92
F.2 Efectos sobre el número de la versión, etc.....	94
F.3 Requisitos sobre reglas de codificación.....	94
Anexo G – Sumario de la notación ASN.1.....	95

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 clase de información (por ejemplo, numérica, textual, de imagen fija o de vídeo). Un *valor de datos* (o simplemente *valor*) es un ejemplar de esa clase. 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.

Aunque esta notación normalizada se define dentro del marco de la OSI, puede utilizarse para muchos otros objetivos. En las capas inferiores del modelo de referencia básico OSI (véase la Rec. UIT-T X.200 | ISO/CEI 7498-1) y en muchas otras arquitecturas de protocolos, cada mensaje se especifica como el valor binario de una secuencia de octetos. En la capa de presentación de OSI (véase la Rec. UIT-T X.216 | ISO/CEI 8822), la naturaleza de los parámetros de datos de usuario cambia. Sin embargo, las normas relativas a la capa de aplicación requieren la definición de tipos de datos muy complejos para llevar sus mensajes, con independencia de su representación binaria. Para especificar los tipos de datos, precisan de una notación que no necesariamente determine la representación de cada valor. Esa notación ha de ser complementada mediante la especificación de uno o más algoritmos denominados **reglas de codificación (encoding rules)** que determinan el valor de los octetos de capa más baja que transportan los datos de aplicación [dicha especificación se denomina **sintaxis de transferencia (transfer syntax)**]. El protocolo de la capa de presentación de OSI (véase la Rec. UIT-T X.226 | ISO/CEI 8823-1) puede negociar las sintaxis de transferencia **codificaciones (encodings)** que habrán de utilizarse.

Fuera del contexto de OSI hay un reconocimiento creciente de la noción de valor abstracto de una clase (por ejemplo, una determinada imagen en color 256) con independencia de los detalles de cualquier codificación particular en la que, para interpretar correctamente la representación del valor mediante un esquema de bits, es necesario saber (usualmente a partir del contexto) el tipo (la clase) del valor que se representa, así como el mecanismo de codificación que se emplea. La identificación de un tipo es, por ello, una parte importante de la presente Recomendación | Norma Internacional.

Una técnica muy general para definir un tipo complicado a nivel abstracto consiste en definir un pequeño número de **tipos simples (simple types)** definiendo todos los valores posibles de los tipos simples, y combinar entonces estos tipos simples 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 no ordenado de tipos existentes (distintos), se puede formar un valor como un conjunto (no ordenado) de valores, formado por uno de cada uno de los tipos existentes; la colección de todos los posibles conjuntos de valores no ordenados 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 (no ordenado) 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 los tipos que se combinan de esta manera 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.

En la notación se especifican cuatro clases de rótulo.

La primera es la clase **universal**. Los rótulos de clase universal sólo se utilizan como se especifica en esta Recomendación | Norma Internacional, y cada rótulo:

- a) se asigna a un solo tipo; o
- b) se asigna a un mecanismo de construcción.

Los usuarios de esta notación no están autorizados a especificar explícitamente rótulos de clase universal en sus especificaciones ASN.1, ya que estos rótulos están incorporados (built-in) y sólo pueden ser especificados explícitamente en la presente Recomendación | Norma Internacional.

Las otras tres clases de rótulos se denominan rótulo de clase **aplicación (application)**, rótulo de clase **privada (private)** y rótulo de clase **específica del contexto (context-specific)**. No hay diferencia formal entre las utilizaciones de los rótulos de estas 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 C.2.12 se dan directrices respecto al modo de emplear normalmente las clases.

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 atribución de rótulos es una parte importante de la utilización de esta notación.

NOTA – En esta Recomendación | Norma Internacional, los valores de rótulo se asignan 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 etapa ulterior, sin embargo, es posible que sea necesario cambiar esos tipos (normalmente añadiendo componentes suplementarios en un tipo secuencia o un tipo conjunto). Para que sea posible de modo que las implementaciones que utilizan las definiciones de tipos antiguos puedan interfundirse con implementaciones que utilizan las definiciones de tipos nuevos, es necesario que las reglas de codificación den el apoyo apropiado. La notación ASN.1 apoya la inclusión de un **marcador de extensión (extension marker)** 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 (extension series)**, 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 éstos. Las cláusulas 10 a 31 especifican también la notación que ha de emplearse para especificar valores de tipos definidos mediante ASN.1.

Las cláusulas 32 y 33 (inclusive) definen los tipos soportados por la ASN.1 para llevar dentro de ellas la codificación completa de los tipos ASN.1.

Las cláusulas 34 a 39 (inclusive) definen los tipos cadena de caracteres.

Las cláusulas 40 a 43 (inclusive) definen ciertos tipos que se consideran de utilidad general, pero que no requieren reglas de codificación adicionales.

Las cláusulas 44 y 48 definen una notación que permite definir subtipos a partir de los valores de un tipo parent (progenitor).

El anexo A es parte integrante de la presente Recomendación | Norma Internacional y en él se dan directrices sobre cómo pueden los usuarios de esta Recomendación | Norma Internacional referirse a los tipos y valores de ASN.1 definidos utilizando la Rec. X.208 del CCITT | ISO/CEI 8824.

El anexo B es parte integrante de la presente Recomendación | Norma Internacional y contiene valores de identificador de objeto y descriptor de objeto asignados en esta Recomendación | Norma Internacional.

El anexo C no es parte integrante de la presente Recomendación | Norma Internacional y proporciona ejemplos y sugerencias sobre la utilización de la notación ASN.1.

El anexo D no es parte integrante de la presente Recomendación | Norma Internacional y contiene un suplemento didáctico sobre las cadenas de caracteres ASN.1.

El anexo E no es parte integrante de la presente Recomendación | Norma Internacional y describe características de la versión anterior de ASN.1 que han sido reemplazadas.

El anexo F no es parte integrante de la presente Recomendación | Norma Internacional y proporciona un suplemento didáctico sobre el modelo ASN.1 de tipo de extensión.

El anexo G no es parte integrante de la presente Recomendación | Norma Internacional y proporciona un sumario de ASN.1 utilizando 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 conjuntos (o juegos) de caracteres (por medio de referencias a otras Recomendaciones y/o Normas Internacionales) para uso dentro de la ASN.1;
- define varios tipos útiles (ASN.1), que pueden ser referenciados por usuarios de ASN.1.

La notación ASN.1 puede aplicarse cuando sea necesaria para definir la sintaxis abstracta de información. Es aplicable en particular, aunque no exclusivamente, a protocolos de aplicación.

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 esta 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 en la 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 UIT-T X.200 (1994) | ISO/CEI 7498-1:1994, *Tecnología de la información – Interconexión de sistemas abiertos – Modelo de referencia básico: El modelo básico.*
- Recomendación UIT-T X.216 (1994) | ISO/CEI 8822:1994, *Tecnología de la información – Interconexión de sistemas abiertos – Definición del servicio de presentación.*
- Recomendación UIT-T X.226 (1994) | ISO/CEI 8823-1:1994, *Tecnología de la información – Interconexión de sistemas abiertos – Protocolo de presentación con conexión: Especificación del protocolo.*
- Recomendación UIT-T X.660 (1992)/enm.2 (1997) | ISO/CEI 9834-1:1993/enm.2:1998, *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 1 y 2).*
- Recomendación UIT-T X.681 (1997) | ISO/CEI 8824-2:1998, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación del objeto de información.*

- Recomendación UIT-T X.682 (1997) | ISO/CEI 8824-3:1998, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de constricciones.*
- Recomendación UIT-T X.683 (1997) | ISO/CEI 8824-4:1998, *Tecnología de la información – Notación de sintaxis abstracta uno: Parametrización de las especificaciones de notación de sintaxis abstracta uno.*
- Recomendación UIT-T X.690 (1997) | ISO/CEI 8825-1:1998, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno: Especificación de las reglas de codificación básica, de las reglas de codificación canónica y de las reglas de codificación distinguida.*
- Recomendación UIT-T X.691 (1997) | ISO/CEI 8825-2:1998, *Tecnología de la información – Reglas de codificación de notación de sintaxis abstracta uno: Especificación de las reglas de codificación compactada.*

2.2 Pares de Recomendaciones | Normas Internacionales de contenido técnico equivalente

- Recomendación X.208 del CCITT (1988), *Especificación de la notación de sintaxis abstracta uno (ASN.1).*
ISO/CEI 8824:1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1).*

2.3 Referencias adicionales

- Recomendación del CCITT T.61 (1988), *Repertorio de caracteres y juegos de caracteres codificados para el servicio teletex internacional.*
- Recomendación del 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 6523:1984, *Data interchange – Structures for the identification of organizations.*
- ISO/CEI 7350:1991, *Information technology – Registration of repertoires of graphic characters from ISO 10367.*
- ISO 8601:1988, *Data elements and interchange formats – Information interchange – Representation of dates and times.*
- ISO/CEI 10646-1:1993, *Information technology – Universal Multiple-Octet Coded Character Set (UCS – Part 1: Architecture and Basic Multilingual Plane.*
- ISO/CEI 10646-1:1993/Amd.2:1996, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane – Amendment 2: UCS Transformation Format 8 (UTF-8).*

3 Definiciones

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

3.1 Especificación de objeto 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 Definición del servicio de presentación

Esta Recomendación | Norma Internacional utiliza los siguientes términos definidos en la Rec. UIT-T X.216 | ISO/CEI 8822:

- a) (una) sintaxis abstracta;
- b) nombre de sintaxis abstracta;
- c) conjunto de contextos definido;
- d) valor de datos de presentación;
- e) (una) sintaxis de transferencia;
- f) nombre de sintaxis de transferencia.

3.5 Especificación del protocolo de presentación

Esta Recomendación | Norma Internacional utiliza el siguiente término definido en la Rec. UIT-T X.226 | ISO/CEI 8823-1:

- identificador de contexto de presentación.

3.6 Estructura para la identificación de organizaciones

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

- a) organización emisora;
- b) código de organización;
- c) designador de indicativo internacional.

3.7 Conjunto de caracteres codificados multiocteto universal (UCS)

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

- a) plano multiidioma 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.

3.8 Definiciones adicionales

3.8.1 carácter abstracto: El conjunto de información asociada con una célula en una tabla que define un repertorio de caracteres.

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

- a) un símbolo gráfico;
- b) un nombre de carácter; o
- c) la definición de funciones asociadas con el carácter cuando se utiliza en entornos particulares.

3.8.2 valor abstracto: Un valor cuya definición se basa sólo en el tipo, con independencia de cómo se representa en cualquier regla de codificación.

NOTA – El empleo de la expresión "valor abstracto" es con frecuencia una declaración de lo que se dice; probablemente varíe en base a las reglas de codificación utilizadas.

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

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

3.8.5 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.

3.8.6 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, la utilización del tipo pdv incrustado proporcionará, por lo general, un mecanismo más flexible para el anuncio o el acuerdo sobre la naturaleza de la codificación que el tipo cadena de bits.

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

3.8.8 carácter: Un miembro de un conjunto de elementos utilizados para la organización, control o representación de datos.

NOTA – Esto significa, por ejemplo, que un carácter que combina acento y letra "e" son dos caracteres en la versión francesa de ISO 646 y no un solo carácter "é".

3.8.9 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.

3.8.10 repertorio de caracteres: Caracteres de un conjunto de caracteres sin indicación alguna de cómo se codifican.

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

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

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

3.8.13 tipos elección: Tipos definidos por referencia a una lista de tipos diferentes; cada valor del tipo elección es un valor de los tipos componentes.

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

3.8.15 constricción: Una notación que puede utilizarse en asociación con la notación para un tipo, para definir un subtipo de ese tipo.

3.8.16 caracteres de control: Caracteres que aparecen en algunos repertorios de caracteres que han recibido un nombre (y quizá una función definida en relación con determinados entornos) pero a los que no se ha asignado una forma de carácter, y que no son caracteres con avance de espacio.

NOTA – NEWLINE y TAB son ejemplos de caracteres de control a los que se les ha asignado una función de formateo en un entorno de impresión. DLE es un ejemplo de carácter de control al que se ha asignado una función en un entorno de comunicación.

3.8.17 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 Recomendación 460-2 del Comité Consultivo Internacional de Radiocomunicaciones (CCIR). El CCIR también ha definido UTC como el acrónimo para Tiempo Universal Coordinado.

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

3.8.18 elemento: Un miembro de una clase elemento que puede distinguirse de todos los elementos de la misma clase.

3.8.19 clase elemento: Un tipo (cuyos elementos son sus valores) o clase de objeto de información (cuyos elementos son todos los objetos posibles de esa clase).

3.8.20 conjunto de elementos: Uno o más elementos de la misma clase elemento.

3.8.21 tipo pdv inscrito: Un tipo cuyo conjunto de valores es la unión de los conjuntos de valores en todas las sintaxis abstractas posibles. Este tipo es una parte de una especificación ASN.1 que lleva un valor cuyo tipo puede ser definido externamente a esa especificación ASN.1. También lleva una identificación del tipo del valor que se transporta así como una identificación de la regla de codificación utilizada para codificar el valor.

3.8.22 codificación: El esquema de bits resultante de la aplicación de un conjunto de reglas de codificación a un valor de una sintaxis abstracta específica.

3.8.23 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.

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

3.8.25 adición de extensión: Una de las notaciones añadidas en una serie de extensión. Para 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 ulterior única. Para una constricción es la adición de un 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 ELECCIÓN, con rótulos crecientes).

3.8.26 grupo adición de extensión: Uno o más componentes de un tipo conjunto, secuencia o elección agrupados dentro de corchetes de versión. Un grupo 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.

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

3.8.28 constricción extensible: Una constricción de subtipo con un marcador de extensión.

3.8.29 punto de inserción de extensión: 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.

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

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

3.8.32 relacionados por 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.

3.8.33 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 espacios en blanco entre el marcador de extensión y el 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.

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

NOTA – Pueden extenderse tipos anidados y no anidados.

3.8.35 tipo extensible: Un tipo con un marcador de extensión.

3.8.36 referencia externa: Una referencia de tipo, referencia de valor, objeto, etc., que se define en un módulo distinto de aquel al que se hace referencia y que es referido prefijando el nombre del módulo al ítem referido.

EJEMPLO – `ModuleName.TypeReference`

3.8.37 tipo externo: Un tipo que es una parte de una especificación ASN.1 y que lleva un valor cuyo tipo puede ser definido externamente a esa especificación ASN.1. También lleva una identificación del tipo del valor que se transporta.

3.8.38 false: Uno de los valores distinguidos de tipo booleano (véase "verdadero").

3.8.39 gobernante; gobernador: En relación con algún objeto, conjunto de objetos, conjunto de valores, valor o subtipo, la clase de objeto de información o tipo que controla su interpretación restringiendo el ítem o los ítems que entran en juego para ser notación de valor de esa clase o tipo, respectivamente.

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

NOTA – La gama de un entero puede verse limitada por reglas de codificación particulares, pero tales limitaciones se establecen de tal modo que no afecten a ningún usuario de ASN.1.

3.8.41 ítems: Secuencias nombradas de caracteres tomadas en los juegos de caracteres ASN.1, especificadas en la cláusula 11, que se utilizan para formar la notación ASN.1.

3.8.42 módulo: Uno o más ejemplares de la utilización de la notación ASN.1 para definición de tipo, valor, etc., encapsuladas utilizando la notación de módulo ASN.1 (véase la cláusula 12).

3.8.43 tipo nulo: Un tipo simple que consta de un solo valor, que también se llama nulo.

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

3.8.45 tipo descriptor de objeto: Un tipo cuyos valores distinguidos son texto legible por el ser humano y que proporcionan una breve descripción de un objeto.

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

3.8.46 identificador de objeto: Un valor (distinguible de todos los demás) que está asociado con un objeto.

3.8.47 tipo identificador de objeto: Un valor simple cuyos valores distinguidos son el conjunto de todos los identificadores de objeto atribuidos de conformidad con las reglas de la Rec. UIT-T X.660 | ISO/CEI 9834-1.

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.

3.8.48 tipo cadena de octetos: Un 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.

3.8.49 notación de tipo abierto: Una notación ASN.1 utilizada para denotar 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 único. 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 del 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, donde el "FieldName" denota un campo de tipo o un campo de valor de tipo variable. La notación "ANY" definida en la Rec. X.208 del CCITT | ISO/CEI 8824, es una notación de tipo abierto.

3.8.50 tipo progenitor (de un subtipo): El tipo que se constriñe cuando se define un subtipo.

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

3.8.51 producción: Una parte de la notación formal utilizada para especificar ASN.1.

3.8.52 tipo real: Un tipo simple cuyos valores distinguidos (especificados en la cláusula 20) son miembros del conjunto de números reales.

3.8.53 definiciones recursivas: Un conjunto de definiciones ASN.1 que no pueden ser reordenadas de un 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.

3.8.54 tipo cadena de caracteres restringida: Un tipo cadena de caracteres cuyos caracteres se toman de un repertorio de caracteres fijo, identificado en la especificación de tipo.

3.8.55 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.

3.8.56 tipos secuencia: Tipos definidos por referencia a una lista ordenada de tipos (algunos de los cuales pueden ser declarados como opcionales); cada valor del tipo secuencia es una lista 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 secuencia contenga un valor de ese tipo componente.

3.8.57 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.

3.8.58 tipos conjunto: Tipos definidos por referencia a una lista fija, no ordenada, de tipos distintos (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 el nuevo tipo contenga un valor de ese tipo componente.

3.8.59 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.

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

3.8.61 carácter con avance de espacio: Un carácter de un repertorio de caracteres que se incluye con caracteres gráficos en la impresión de una cadena de caracteres, pero que está representado en la reproducción física por un espacio vacío; normalmente no se le considera como un carácter de control (véase 3.8.16).

NOTA – En un repertorio de caracteres puede haber un solo carácter con avance de espacio o múltiples caracteres con avance de espacio con anchuras variables.

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

3.8.63 rótulo: Una denotación de tipo que va asociada con todo tipo ASN.1.

3.8.64 tipos rotulados: Tipos definidos por referencia a un solo tipo existente y a un rótulo; el nuevo tipo es isomórfico con el tipo existente, pero es diferente de éste.

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

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

3.8.67 tipo: Un conjunto de valores que tiene un nombre.

3.8.68 nombre de referencia de tipo: Un 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.

3.8.69 tipo cadena de caracteres no restringida: Un tipo cuyos valores son valores de una sintaxis abstracta de caracteres identificada separadamente para cada ejemplar de uso de ese tipo.

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

3.8.71 valor: Un miembro distinguido de un conjunto de valores.

3.8.72 nombre de referencia de valor: Un nombre asociado de manera única con un valor dentro de algún contexto.

3.8.73 conjunto de valores: Una colección de valores de un tipo. Semánticamente equivalente a un subtipo.

3.8.74 corchetes de versión: Par de corchetes izquierdos y derechos adyacentes ([[o]]) utilizados para delimitar el arranque y el final de un grupo adición de extensión.

3.8.75 espacio en blanco: Cualquier acción de formato que da un espacio en una página impresa, tal como el carácter SPACE o el carácter TAB, o múltiples usos de esos caracteres.

4 Abreviaturas

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

ASN.1	Notación de sintaxis abstracta uno (<i>abstract syntax notation one</i>)
BER	Reglas de codificación básica de ASN.1 (<i>basic encoding rules of ASN.1</i>)
CEI	Comisión Electrotécnica Internacional
DCC	Indicativo de país de datos (<i>data country code</i>)
DNIC	Código de identificación de red de datos (<i>data network identification code</i>)
EER	Empresa de explotación reconocida
ICD	Designador de indicativo internacional (<i>international code designator</i>)
ISO	Organización Internacional de Normalización (<i>international organization for standardization</i>)
PDV	Valor de datos de presentación (<i>presentation data value</i>)
PER	Reglas de codificación básica de ASN.1 (<i>packed encoding rules of ASN.1</i>)
UCS	Conjunto de caracteres codificados multiocteto universal (<i>universal multiple-octet coded character set</i>)
UIT-T	Unión Internacional de Telecomunicaciones – Sector de Normalización de las Telecomunicaciones

5 Notación

La notación ASN.1 consiste en una secuencia de caracteres del conjunto (o juego) de caracteres ASN.1 especificados en la cláusula 10.

Cada vez que se utiliza la notación ASN.1, caracteres pertenecientes al conjunto de caracteres ASN.1 son agrupados para formar ítems. La cláusula 11 especifica todas las secuencias de caracteres que forman ítems ASN.1 y el nombre de cada ítem.

La notación ASN.1 se especifica en la cláusula 12 (y siguientes), para lo cual especifica la colección de secuencias de ítems que forman ejemplares válidos de la notación ASN.1, y se especifica la semántica de cada secuencia.

Para especificar esas colecciones, esta Recomendación | Norma Internacional utiliza una notación formal definida en las siguientes subcláusulas.

5.1 Producciones

Una colección nueva (más compleja) de secuencias ASN.1 se define mediante una producción. Ésta utiliza los nombres de colecciones de secuencias de producción definidas en la presente Recomendación | Norma Internacional y forma una nueva colección de secuencias de producción especificando:

- a) que la nueva colección de secuencias de producción habrá de consistir en cualquier secuencia contenida en cualquiera de las colecciones originales; o
- b) que la nueva colección habrá de consistir en cualquier secuencia de producción que pueda ser generada tomando exactamente un carácter de cada colección, y yuxtaponiéndolos en un orden especificado.

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

- a) un nombre para la nueva colección de secuencias de producción;
- b) los caracteres

$$::=$$
- c) una o más colecciones alternativas de secuencias, definidas en 5.2, separadas por el carácter

$$|$$

Una secuencia de producción está presente en la nueva colección si lo está en una o más de las colecciones alternativas. La nueva colección es referenciada en esta Recomendación | Norma Internacional por el nombre a que se refiere el anterior inciso a).

NOTA – Si la misma secuencia de producción aparece en más de una alternativa, cualquier ambigüedad semántica en la notación resultante es resuelta por otras partes de la secuencia de producción ASN.1 completa.

5.2 Las colecciones alternativas

Cada una de las colecciones alternativas de secuencias de producción en "una o más colecciones alternativas de" [véase 5.1 c)] se especifica por una lista de nombres. Cada nombre es, o bien el nombre de un ítem, o el nombre de una colección de secuencias de producción definida por una producción en esta Recomendación | Norma Internacional.

La colección de secuencias de producción definida por la alternativa consiste en todas las secuencias obtenidas tomando una cualquiera de las secuencias de producción (o el ítem) asociada con el primer nombre, en combinación con (y seguida por) cualquiera de las secuencias de producción (o ítem) asociada con el segundo nombre, en combinación con (y seguida por) cualquiera de las secuencias de producción (o ítem) asociada con el tercer nombre, y así sucesivamente hasta el último nombre (o ítem), inclusive, de la alternativa.

5.3 Ejemplo de una producción

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

es una producción que asocia al nombre "BitStringValue" las secuencias de producción siguientes:

- a) cualquier "bstring" (un ítem); o
- b) cualquier "hstring" (un ítem); o
- c) cualquier secuencia de producción asociada con "IdentifierList", precedida por un "{" y seguida por un "}".

NOTA – "{" y "}" son los nombres de ítems que contienen los caracteres únicos { y } (véase 11.17).

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

5.4 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.

5.5 Recursión

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

NOTA – En muchos casos, tal reaplicación da lugar a una colección no acotada de secuencias permitidas, algunas de las cuales, o todas ellas, pueden ser a su vez inacotadas. Esto no es un error.

5.6 Referencias a una colección de secuencias

Esta Recomendación | Norma Internacional hace referencia a una colección de secuencias (parte de la notación ASN.1) referenciando el primer nombre (antes de " ::= ") de una producción; el nombre va encerrado entre un par de " (comillas) para distinguirlo de texto en lenguaje natural, a menos que aparezca como parte de una producción.

5.7 Referencias a un ítem

Esta Recomendación | Norma Internacional hace referencia a un ítem referenciando el nombre del ítem; el nombre va encerrado entre un par de " (comillas) para distinguirlo de texto en lenguaje natural, a menos que aparezca como parte de una producción y no sea un ítem de un solo carácter, " ::= ", " .. ", o "...".

5.8 Notaciones abreviadas

Para obtener producciones más concisas y legibles se utilizan las siguientes notaciones abreviadas en la definición de las colecciones de secuencias de producción ASN.1 de 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 (no se utiliza en ningún otro lugar de la presente Recomendación | Norma Internacional):

- a) un asterisco (*) que sigue a dos nombres, "A" y "B", denota el ítem vacío (empty) (véase 11.7) o una secuencia de producción asociada con "A", o una serie alternante de secuencias de producción asociadas con "A" y "B" que empiezan y terminan con una secuencia de producción 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 ítem 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 ítem empty (véase 11.7) o una secuencia de producción asociada con "A". Así:

$F ::= A ?$

es equivalente a:

$F ::= \text{empty} \mid A$

6 Modelo ASN.1 de extensión de tipo

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

- la ausencia de adiciones de extensión esperadas en un tipo secuencia o conjunto; o
- 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 restricción 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 por extensión a "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 deben especificarse por el diseñador de la capa de aplicación.

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 no esperadas 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

7.1 Todas las reglas de codificación ASN.1 permitirán la codificación de valores de un tipo extensible "X" de manera que puedan ser decodificados utilizando un tipo extensible "Y" que esté relacionado por 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 por extensión con "Y" (y por lo tanto también con "X").

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

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 por extensión "Z") a otra aplicación que decodifica el valor utilizando el tipo extensible "Y" que está relacionado por extensión con "X", el decodificador que utiliza el tipo "Y" obtiene un valor abstracto compuesto por:

- un valor abstracto del tipo raíz de extensión;
- un valor abstracto de cada adición de extensión que está presente en "X" y en "Y";
- 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 por 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.

ISO/CEI 8824-1 : 1998 (S)

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).

En todos los casos, la presencia de adiciones de extensión no afectará la aptitud para reconocer información posterior cuando un tipo con un marcador de extensión está anidado dentro de algún otro tipo.

NOTA – 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.

8 Rótulos

8.1 Un rótulo se especifica dando su clase y el número dentro de la clase. La clase es una de las siguientes:

- universal;
- aplicación;
- privada;
- específica al contexto.

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

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

El cuadro 1 recapitula 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 incrustado
UNIVERSAL 12	Tipo UTF8String
UNIVERSAL 13-15	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

8.3 Algunas reglas de codificación exigen un orden canónico de los rótulos. En 8.4 se define dicho orden, a efectos de uniformidad.

- 8.4** El orden canónico de los rótulos se define de la siguiente manera:
- los elementos o alternativas con rótulos de la clase universal aparecerán en primer lugar, seguidos por aquellos que tengan rótulos de clase aplicación, seguidos a su vez por aquellos que tengan rótulos específicos del contexto, seguidos por último por los de rótulo de clase privada;
 - dentro de cada clase de rótulos, los elementos o alternativas aparecerán en el 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 un 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 15.4), "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 El conjunto (o juego) de caracteres ASN.1

10.1 Un ítem ASN.1 consistirá en una secuencia de los caracteres indicados en el cuadro 2, excepto lo especificado en 10.2 y 10.3.

Cuadro 2 – Caracteres ASN.1

A a Z
a a z
0 a 9
: = , { } < . @ () [] - ' " & ^ * ; !

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

- 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 ítems 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 símbolos gráficos del juego de caracteres definidos podrán aparecer en la notación ASN.1, entre comillas (") (véase 11.11).

10.3 En el ítem "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.

11 Ítems ASN.1

11.1 Reglas generales

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

11.1.2 Cada ítem especificado en las siguientes subcláusulas (excepto "bstring", "hstring" y "cstring") aparecerá en una sola línea y, (salvo para los ítems "comment", "bstring", "hstring" y "cstring"), no contendrá espacios en blanco (véanse 11.9, 11.10 y 11.11).

11.1.3 La longitud de la línea no está limitada.

11.1.4 Los ítems de las secuencias especificadas por esta Recomendación | Norma Internacional (la notación ASN.1) pueden aparecer en una línea o en varias y pueden estar separados por espacio en blanco, por líneas vacías o por comentarios.

11.1.5 Un ítem estará separado del siguiente por un espacio en blanco, una línea nueva o un comentario, si el carácter (o caracteres) inicial del ítem siguiente es un carácter (o caracteres) autorizado para su inclusión al final de los caracteres del ítem anterior.

11.2 Referencias de tipo

Nombre de ítem – 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 11.18.

11.3 Identificadores

Nombre de ítem – 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 Referencia de valor

Nombre de ítem – valuereference

Una "valuereference" consistirá en la secuencia de caracteres especificada para un "identifier" en 11.2. 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 Referencia de módulo

Nombre de ítem – 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 Comentario

Nombre de ítem – 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 ítems ASN.1 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 Un "comment" comenzará con un par de guiones adyacentes y terminará con el próximo par de guiones adyacentes o al final de la línea, lo que suceda primero. Un "comment" no contendrá más pares de guiones adyacentes que el par que lo abre y el par (de haberlo), que lo cierra. Puede incluir símbolos gráficos que no están en el juego de caracteres especificados en 10.1 (véase 10.3).

11.7 Ítem vacío

Nombre de ítem – empty

El ítem "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 Ítem número

Nombre de ítem – 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 ítem "number" se hace corresponder siempre con (a) un valor entero interpretándolo como notación decimal.

11.9 Ítem cadena binaria

Nombre de ítem – bstring

Una "bstring" estará constituida por un número arbitrario (que puede ser cero) de ceros, unos, espacios en blanco y caracteres de nuevo renglón, precedidos por un solo apóstrofo (') y seguidos por el par de caracteres:

'B

Los espacios en blanco y los caracteres de nuevo renglón que aparecen dentro de un ítem cadena binaria no tienen significado alguno.

EJEMPLO – '01101100'B

11.10 Ítem cadena hexadecimal

Nombre de ítem – hstring

11.10.1 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

o espacios en blanco o caracteres de nuevo renglón, precedidos por un solo apóstrofo (') y seguidos por el par de caracteres:

'H

Los espacios en blanco y los caracteres de nuevo renglón que aparecen dentro de un ítem cadena hexadecimal no tienen significado alguno.

EJEMPLO – 'AB0196'H

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

11.11 Ítem cadena de caracteres

Nombre de ítem – cstring

11.11.1 Una "cstring" estará constituida por un número arbitrario (que puede ser cero) de símbolos gráficos y caracteres con avance de espacio tomados del conjunto de caracteres referenciado por el tipo cadena de caracteres, precedido y seguido por comillas ("). Si el conjunto de caracteres incluye el carácter comillas, este carácter (si está presente en la cadena de caracteres representada por la "cstring") debe estar representado en la "cstring" por un par de

ISO/CEI 8824-1 : 1998 (S)

caracteres comillas 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 avance de espacio en la posición que precede o sigue al final de línea de la "cstring". El espacio en blanco que aparece 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 cadenas de caracteres a cada uno de cuyos caracteres se le ha asignado un símbolo gráfico o bien se trata de un carácter con avance de espacio. Se dispone de sintaxis ASN.1 alternativa para cuando sea preciso denotar una cadena de caracteres que contiene caracteres de control. (Véase la cláusula 34).

NOTA 2 – La cadena de caracteres representada por una "cstring" está constituida por los caracteres asociados con los símbolos gráficos impresos y los caracteres con avance de espacio. Los caracteres con avance de espacio 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 avance de espacio en la "cstring" o cuando símbolos gráficos del repertorio de caracteres no son inequívocos, la cadena de caracteres denotada por "cstring" puede resultar imprecisa.

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 si en la especificación se utiliza un tipo de carácter de espaciado proporcional (como el empleado más arriba).

11.11.2 Cuando un carácter sea carácter combinante, se denotará en 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.)

EJEMPLO – El carácter combinante acento y la letra minúscula "e" son dos caracteres en la versión francesa de ISO 646 y, por eso, en una "cstring" correspondiente se escriben como dos caracteres y no como un solo carácter.

11.11.3 La "cstring" no se utilizará para representar valores de cadena de caracteres que contengan caracteres de control. Sólo se permiten caracteres gráficos y caracteres con avance de espacio.

11.12 Ítem asignación

Nombre de ítem – "::~="

Este ítem estará constituido por la secuencia de caracteres:

```
::=
```

NOTA – Esta secuencia no contiene ningún carácter espacio en blanco (véase 11.1.2).

11.13 Separador de gama

Nombre de ítem – ".."

Este ítem estará constituido por la secuencia de caracteres:

```
..
```

NOTA – Esta secuencia no contiene ningún carácter espacio en blanco (véase 11.1.2).

11.14 Elipsis

Nombre de ítem – "..."

Este ítem estará constituido por la secuencia de caracteres:

```
...
```

NOTA – Esta secuencia no contiene ningún carácter espacio en blanco (véase 11.1.2).

11.15 Corchetes de versión izquierdos

Nombre de ítem – "[["

Este ítem estará constituido por la secuencia de caracteres:

[[

NOTA – Esta secuencia no contiene ningún carácter espacio en blanco (véase 11.1.2).

11.16 Corchetes de versión derechos

Nombre de ítem – "]]"

Este ítem estará constituido por la secuencia de caracteres:

]]

NOTA – Esta secuencia no contiene ningún carácter espacio en blanco (véase 11.1.2).

11.17 Ítems carácter único (o carácter individual)

Nombres de ítems –

"{"
 "}"
 "<"
 ">"
 "."
 "("
 ")"
 "["
 "]"
 "-" (guión)
 ":"
 ">"
 "@"
 "|"
 "!"
 "^"

Un ítem con cualquiera de estos nombres estará constituido por el carácter único sin las comillas.

11.18 Palabras reservadas

Nombres de palabras reservadas –

ABSENT	END	INSTANCE	REAL
ABSTRACT-SYNTAX	ENUMERATED	INTEGER	SEQUENCE
ALL	EXCEPT	INTERSECTION	SET
APPLICATION	EXPLICIT	ISO646String	SIZE
AUTOMATIC	EXPORTS	MAX	STRING
BEGIN	EXTENSIBILITY	MIN	SYNTAX
BIT	EXTERNAL	MINUS-INFINITY	T61String
BMPString	FALSE	NULL	TAGS
BOOLEAN	FROM	NumericString	TeletexString
BY	GeneralizedTime	OBJECT	TRUE
CHARACTER	GeneralString	ObjectDescriptor	TYPE-IDENTIFIER
CHOICE	GraphicString	OCTET	UNION

CLASS	IA5String	OF	UNIQUE
COMPONENT	IDENTIFIER	OPTIONAL	UNIVERSAL
COMPONENTS	IMPLICIT	PDV	UniversalString
CONSTRAINED	IMPLIED	PLUS-INFINITY	UTCTime
DEFAULT	IMPORTS	PRESENT	UTF8String
DEFINITIONS	INCLUDES	PrintableString	VideotexString
EMBEDDED		PRIVATE	VisibleString
			WITH

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

NOTA 1 – Estas secuencias no contienen espacios en blanco.

NOTA 2 – Las palabras clave CLASS, CONSTRAINED, 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" es especificada por 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 ";" |
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 ::=
typerreference |
valuereference |
objectclassreference |
objectreference |
objectsetreference

AssignmentList ::=
Assignment |
AssignmentList Assignment

Assignment ::=
TypeAssignment |
ValueAssignment |
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 rútilos de la clase universal), el "ModuleBody" puede utilizarse fuera de una "ModuleDefinition".

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

NOTA 4 – La agrupación de tipos de datos ASN.1 para constituir módulos no entraña necesariamente que valores de datos de presentación formen sintaxis abstractas denominadas, destinadas a la definición de contextos de presentación.

NOTA 5 – 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 6 – 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 por 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 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 tipos. No produce efecto sobre conjuntos de objetos.

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 es vacío, el valor de identificador de objeto denotado identifica inequívocamente y de forma única el módulo que se está definiendo. En la definición del valor de 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 es 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 ítems. 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" ("valuereferences") 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 un "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 exclusivamente 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; 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 vacío de "Exports", todo "Symbol" definido en el módulo puede ser referenciado desde otros módulos.

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

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 retrocompatibilidad.

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 exportado.

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 "AssignmentList" (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:

<i>Alternativa de Assignment</i>	<i>Subcláusula definidora</i>
"TypeAssignment"	15.1
"ValueAssignment"	15.2
"ValueSetTypeAssignment"	15.4
"ObjectClassAssignment"	Rec. UIT-T X.681 ISO/CEI 8824-2, 9.1
"ObjectAssignment"	Rec. UIT-T X.681 ISO/CEI 8824-2, 11.1
"ObjectSetAssignment"	Rec. UIT-T X.681 ISO/CEI 8824-2, 12.1
"ParameterizedAssignment"	Rec. 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 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.

13.3 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 elemento 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.

13.4 Una referencia externa solamente deberá utilizarse en un módulo para referenciar un ítem que está definido en un módulo diferente, y se especifica por 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).

13.5 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".

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 casos 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 ::= "@" GlobalModuleReference
 "."
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 La "GlobalModuleReference" identifica un módulo ASN.1 (véase 12.1).

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

14.6 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.

14.7 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.

14.8 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 y denominado **cuenta de iteraciones (iteration count)**] 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.

14.9 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 11.18).

15.2 Para asignar un valor a una "valuereference" se utiliza la notación especificada por la producción "ValueAssignment":

```

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

```

El "Value" que se asigne a la "valuereference" deberá ser una notación válida para el valor del tipo definido por "Type" (especificado en 15.3).

15.3 "Value" es una notación para un valor de un tipo si, o bien:

- a) "Value" es una notación "BuiltinValue" para el tipo (véase 16.8); o
- b) "Value" es una notación "DefinedValue" para un valor de ese tipo.

15.4 A una "typereference" le puede ser asignado un conjunto de valores por 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" y que contiene exactamente los valores que están especificados en, o permitidos por, "ValueSet". La "typereference" no será una palabra reservada ASN.1 (véase 11.18) y podrá ser referenciada como un tipo. "ValueSet" se define en 15.5.

15.5 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).

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           |
  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	35
ChoiceType	28
EmbeddedPDVType	32
EnumeratedType	19
ExternalType	33
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
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	40.1
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 44.

16.5 Esta Recomendación | Norma Internacional exige la utilización de la notación "NamedType" al especificar los componentes del tipo conjunto, del tipo 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 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":

Value ::= BuiltinValue | ReferencedValue | ObjectClassFieldValue

NOTA – ObjectClassFieldValue se define en la Rec. UIT-T X.681 | ISO/CEI 8824-2, 14.6.

16.8 Los valores de tipo incorporados de ASN.1 pueden especificarse por la notación "BuiltinValue", definida como sigue:

BuiltinValue ::=

BitStringValue	
BooleanValue	
CharacterStringValue	
ChoiceValue	
EmbeddedPDVValue	
EnumeratedValue	
ExternalValue	
InstanceOfValue	
IntegerValue	
NullValue	
ObjectIdentifierValue	
OctetStringValue	
RealValue	
SequenceValue	
SequenceOfValue	
SetValue	
SetOfValue	
TaggedValue	

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

16.9 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

16.10 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.

16.11 El valor de un tipo referenciado mediante la notación "NamedType" será definido por la notación "NamedValue":

NamedValue ::= identifier Value

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.

16.12 La presencia implícita o explícita de un marcador de extensión 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.

17 Notación para el tipo booleano

17.1 El tipo booleano (boolean) (véase 3.8.7) será referenciado 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 3.8.66 y 3.8.38) será definido por la notación "BooleanValue":

BooleanValue ::= TRUE | FALSE

18 Notación para el tipo entero

18.1 El tipo entero (integer) (véase 3.8.40) será referenciado 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 las secuencias "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":

IntegerValue ::=
SignedNumber |
identifier

18.10 El "identifíer" del "IntegerValue" 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 "identifíer", deberá preferirse la utilización de la forma "identifíer" de "IntegerValue".

19 Notación para el tipo enumerado

19.1 El tipo enumerado (enumerated) (véase 3.8.24) será referenciado por la notación "EnumeratedType":

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

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

RootEnumeration ::= Enumeration

AdditionalEnumeration ::= Enumeration

Enumeration ::=
    EnumerationItem | EnumerationItem "," Enumeration

EnumerationItem ::=
    identifíer | 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 "identifíer" y el "SignedNumber" serán distintos de todos los demás "identifíer"s y de los "SignedNumber"s 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 "identifíer" se le asigna un número entero no negativo distinto. Con este fin se asignan los enteros sucesivos comenzando por 0, pero se excluyen los que han sido empleados en "EnumerationItem"s y son "NamedNumber"s.

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 una "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 "AdditionalEnumeration" para la producción de "identifíer" (no de "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 será definido por la notación "EnumeratedValue":

EnumeratedValue ::= identifier

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

20 Notación para el tipo real

20.1 El tipo real (véase 3.8.52) será referenciado 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 que consta de tres enteros, M, B y E:

$$M \times B^E$$

donde M se denomina 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 codificado 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 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 (WITH 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":

```
RealValue ::=
    NumericRealValue | SpecialRealValue

NumericRealValue ::= 0 |
    SequenceValue          -- Value of the associated sequence type

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

La forma "0" deberá utilizarse para valores cero, y la forma alternativa para "NumericRealValue" no deberá utilizarse para valores cero.

21 Notación para el tipo cadena de bits

21.1 El tipo cadena de bits (bitstring) (véase 3.8.6) será referenciado 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 cero (bit zero)**. 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.

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 de este tipo. Están permitidos valores que tengan bits 1 distintos de los bits nombrados.

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, muchos 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":

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

IdentifierList ::=
    identifier |
    IdentifierList "," identifier
```

21.10 Cada "identifier" de "BitStringValue" deberá ser el mismo que un "identifier" de la secuencia "BitStringType" con la que está asociado el valor.

21.11 La notación "BitStringValue" 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 – La secuencia de producción "{" "}" se utiliza para denotar la cadena de bits que no contiene ningún bit.

21.12 Al especificar las reglas de codificación de una cadena de bits, los bits serán referenciados por los términos **primer bit** y **bit de cola**, en donde el primer bit tiene el número cero (véase 21.2).

21.13 Cuando se utiliza la notación "bstring", el **primer bit** está a la izquierda y el **bit de cola** está a la derecha.

21.14 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 construye en forma alguna la manera en que las reglas de codificación colocan una cadena de bits en octetos para la transferencia.

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

EJEMPLO

'A8A'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.

22 Notación para el tipo cadena de octetos

22.1 El tipo cadena de octetos (octetstring) (véase 3.8.48) será referenciado 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":

OctetStringValue ::=
 bstring |
 hstring

22.4 Al especificar las reglas de codificación para una cadena de octetos, los octetos son referenciados por los términos **primer octeto (first octet)** y **octeto final o de cola (trailing octet)**, y los bits en un octeto son referenciados por los términos **bit más significativo (most significant bit)** y **bit menos significativo (least significant bit)**.

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

22.6 Cuando se use la notación "hstring", el dígito hexadecimal más a la izquierda será el semiocteto más significativo del primer octeto.

22.7 Si la "hstring" no tiene un número par de dígitos hexadecimales, se interpretará como si contuviera un solo dígito hexadecimal cero postrero adicional.

23 Notación para el tipo nulo

23.1 El tipo nulo (null) (véase 3.8.43) será referenciado 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":

NullValue ::= NULL

24 Notación para tipos secuencia

24.1 La notación para definir un tipo secuencia (sequence) (véase 3.8.56) 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

RootComponentTypeList ::= ComponentTypeList

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

ExtensionAdditions ::= "," ExtensionAdditionList | empty

ExtensionAdditionList ::= ExtensionAddition |
    ExtensionAdditionList "," ExtensionAddition

ExtensionAddition ::= ComponentType | ExtensionAdditionGroup

ExtensionAdditionGroup ::= "[" ComponentTypeList "]"

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 de secuencia. 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, 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".)

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 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 satisfecerá siempre, si se ha aplicado la rotulación automática.

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

24.5.3 La subcláusula 24.5.1 se aplicará a la serie de "ComponentTypes" 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 elipsis).

24.6 Todos los "ComponentTypes" en adiciones de extensión tendrán rótulos distintos de los de los "ComponentTypes" que siguen textualmente y se encuentran en la raíz, hasta el primero de esos "ComponentType" (inclusive) que no esté marcado OPTIONAL o DEFAULT (si lo hay).

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 tipo rotulado.

24.9 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 rótulo de valor cero para el primer "ComponentType" de la "RootComponentTypeList" o "NamedType" de las "AlternativeTypeLists", uno para el segundo, y así sucesivamente, procediendo con 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 "TaggedTypes" 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:

$$T ::= SEQUENCE \{ a \ T_a, b \ T_b, c \ T_c \}$$

$$E ::= SEQUENCE \{ f1 \ E1, f2 \ T, f3 \ E3 \}$$

los rótulos atribuidos a a, b y c no resultan afectados por la posible rotulación automática aplicada a los componentes de E.

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:

$$T ::= SEQUENCE \{ a \ T_a, b \ SEQUENCE \{ b1 \ T1, b2 \ T2, b3 \ T3 \}, c \ T_c \}$$

$$W ::= SEQUENCE \{ x \ W_x, COMPONENTS OF \ T, y \ W_y \}$$

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 puede dar lugar a cambios en otros componentes, debido al efecto lateral de la modificación de los rótulos.

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 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 marcados 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 de datos de presentación 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 los tipos en una "ExtensionAdditionList" deben ser soportados por una aplicación como un grupo.

24.15 Todos los tipos secuencia (sequence) tienen un rótulo que es de clase universal, número 16.

NOTA – Los tipos secuencia de (sequence of) tienen el mismo rótulo que los tipos secuencia (véase 25.2).

24.16 La notación para la definición de un valor de un tipo secuencia será "SequenceValue":

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

ComponentValueList ::=
    NamedValue |
    ComponentValueList "," NamedValue
```

24.17 La notación "{" "}" 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{"}

24.18 Deberá haber un "NamedValue" para cada "NamedType" 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 3.8.57) a partir de otro tipo será "SequenceOfType".

```
SequenceOfType ::= SEQUENCE OF Type
```

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

NOTA – Los tipos secuencia de tienen el mismo rótulo que los tipos secuencia (véase 24.15).

25.3 La notación para definir un valor de un tipo secuencia de será "SequenceOfValue":

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

ValueList ::=
    Value |
    ValueList "," Value
```

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

25.4 Cada "Value" de la "ValueList" deberá ser del tipo especificado en el "SequenceOfType".

NOTA – Puede darse un significado semántico al orden de estos valores.

26 Notación para tipos conjunto

26.1 La notación para definir un tipo conjunto (set) (véase 3.8.58) 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, salvo para cualesquiera 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".)

NOTA – Esta notación se realiza lógicamente antes de la cumplimentación 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 "AdditionalComponentTypeList" será canónicamente mayor (véase 6.4) que los de las otras alternativas de la "AdditionalComponentTypeList".

NOTA – Cuando el "TagDefault" del módulo en que aparece esta notación es "AUTOMATIC TAGS", esto se consigue, cualesquiera que sean los "ComponentType", como resultado de la aplicación de 24.7.

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

26.5 Todos los tipos conjunto tienen un rótulo que es 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 asociado ningún significado.

26.7 La notación para definir un valor de un tipo conjunto será "SetValue":

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

"ComponentValueList" se especifica en 24.16.

26.8 El "SetValue" será solamente "{" "}" si:

- todas las secuencias "ComponentType" del "SetType" están marcadas "DEFAULT" u "OPTIONAL", y todos los valores han sido omitidos; o
- la notación de tipo era "SET {}".

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

NOTA – Estos "NamedValues" pueden aparecer en cualquier orden.

27 Notación para tipos conjunto de

27.1 La notación para definir un tipo conjunto de (set-of) (véase 3.8.59) a partir de otro tipo será "SetOfType".

```
SetOfType ::=
    SET OF Type
```

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

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

27.3 La notación para definir un valor de un tipo conjunto de será "SetOfValue":

```
SetOfValue ::= "{" ValueList "}" | "{" "}"
```

"ValueList" se especifica en 25.3.

La notación "{" "}" se utiliza cuando el SetOfValue es una lista vacía.

27.4 Cada secuencia "Value" de la "ValueList" deberá ser la notación de un valor del "Type" especificado en el "SetofType".

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, para "SET OF INTEGER", los valores "{ 1 }" y "{ 1 1 }" son distintos.

28 Notación para tipos elección

28.1 La notación para definir un tipo elección (choice) (véase 3.8.13) 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 ::= ExtensionAdditionAlternatives | NamedType

ExtensionAdditionAlternatives ::= "[" AlternativeTypeList "]"

AlternativeTypeList ::=

NamedType |

AlternativeTypeList "," NamedType

NOTA – T ::= CHOICE { a A } y A no son el mismo tipo y pueden estar codificados diferentemente.

28.2 Los tipos definidos en cualquiera de las producciones "AlternativeTypeList" en la lista "AlternativeTypeLists" deberán tener rótulos distintos (véase la cláusula 30). Si la rotulación automática no está en vigor, y el "NamedType" en la extensión de raíz no tiene rótulos, entonces ninguna "NamedType" dentro de la "ExtensionAdditionAlternativesList" será rotulada.

NOTA – Cuando el "TagDefault" del módulo en que aparece esta notación es "AUTOMATIC TAGS" los rótulos se hacen distintos como resultado de la aplicación de 24.7.

28.3 Cuando la producción "AlternativeTypeLists" ocurre 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 ella contiene un "Type" que sea una ocurrencia de "TaggedType", se selecciona la transformación de rotulación automática para la totalidad de la "AlternativeTypeLists"; de no ser así, no se selecciona. Cuando se selecciona, se aplica la transformación de rotulación automática de una "AlternativeTypeLists" a cada "NamedType" de la "AlternativeTypeLists" sustituyendo cada "Type", originalmente en la producción "NamedType", por una ocurrencia del "TaggedType" de sustitución especificado en 24.9.

28.4 El rótulo de cada nuevo "NamedType" añadido a la "ExtensionAdditionAlternativesList" será canónicamente mayor (véase 8.4) que los de las otras alternativas de la "ExtensionAdditionAlternativesList" y será el último "NamedType" en la "ExtensionAdditionAlternativesList".

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

28.6 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 (véanse 24.5 a 24.6, 26.3 y 28.2) 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 porque los rótulos de los tipos d y f, y e y g, son idénticos.

28.7 Los identificadores de todos los "NamedTypes" de la "AlternativeTypeLists" deberán ser diferentes de los de otros "NamedTypes" de esa lista.

28.8 La notación para definir el valor de un tipo elección será "ChoiceValue":

ChoiceValue ::= identifier ":" Value

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

29 Notación para tipos selección

29.1 La notación para definir un tipo selección (selection) (véase 3.8.55) será "SelectionType":

SelectionType ::= identifier "<" Type

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

29.2 Cuando el "SelectionType" se utiliza como un "NamedType", el "identifier" del "NamedType" está presente y se empleará como el "identifier" del "SelectionType".

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

29.4 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 rotulado

Un tipo rotulado (véase 3.8.64) es un tipo nuevo que es isomórfico 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 28.2). 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 isomórfico con el tipo antiguo, pero tiene un rótulo con la clase "Class" y el número "ClassNumber", a menos que la "Class" sea "empty" cuando el rótulo es de clase específica del contexto, número "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 es aprobado cada cierto tiempo por el UIT-T y la ISO.

NOTA 2 – La subcláusula C.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 proporciona 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 específica 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 "Type" es un tipo choice (elección), un tipo open (abierto) o una "DummyReference" (véase la Rec. UIT-T X.683 | ISO/CEI 8824-4, 8.3).

En todos los demás casos, la construcción de la rotulación específica 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 28.2.

30.8 La alternativa "IMPLICIT" no deberá utilizarse si el tipo definido por "Type" es un tipo choice o un tipo open o una "DummyReference" (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":

```

TaggedValue ::= Value
    
```

donde "Value" 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 3.8.47) será referenciado 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":

```
ObjectIdentifierValue ::=
    "{" ObjIdComponentList "}" |
    "{" DefinedValue ObjIdComponentList "}"
```

```
ObjIdComponentList ::=
    ObjIdComponent |
    ObjIdComponent ObjIdComponentList
```

```
ObjIdComponent ::= NameForm |
    NumberForm |
    NameAndNumberForm
```

NameForm ::= identifier

NumberForm ::= number | DefinedValue

```
NameAndNumberForm ::=
    identifier "(" NumberForm ")"
```

31.4 La "valuereference" en "DefinedValue" de "NumberForm" será de tipo 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 La "NameForm" sólo se utilizará para los componentes de identificador de objeto cuyo valor numérico e identificador están especificados en los anexos A a C a la Rec. UIT-T X.660 | ISO/CEI 9834-1, 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 identificador de objeto se tratará como un identificador de la Rec. UIT-T X.660 | ISO/CEI 9834-1.

31.7 El "number" en la "NumberForm" será el valor numérico asignado al componente identificador de objeto.

31.8 Se especificará el "identifier" en el "NameAndNumberForm" 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.

31.9 La semántica asociada con un valor de identificador de objeto se especifica en la Rec. UIT-T X.660 | ISO/CEI 9834-1.

31.10 La parte significativa del componente identificador de objeto es la "NameForm" o "NumberForm" al cual se reduce y que proporciona el valor numérico para el componente identificador de objeto. Salvo para los arcos especificados en los anexos A a C a la Rec. UIT-T X.660 | ISO/CEI 9834-1, el valor numérico del componente identificador de objeto siempre está presente en una instancia de notación de valor de identificador de objeto.

31.11 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.

Con la siguiente definición adicional:

ftam OBJECT IDENTIFIER ::= { iso standard 8571 }

el valor siguiente es también equivalente a los anteriores:

{ ftam pci(1) }

32 Notación para el tipo pdv incrustado

32.1 El tipo pdv incrustado (embedded-pdv) (véase 3.8.21) será referenciado por la notación "EmbeddedPDVType":

EmbeddedPDVType ::= EMBEDDED PDV

32.2 Este tipo tiene un rótulo que es de clase universal, número 11.

NOTA – Cuando se utiliza la negociación de la capa de presentación, la misma funcionalidad que EXTERNAL es proporcionada por EMBEDDED PDV (junto con la funcionalidad añadida), pero los bits en la línea serán diferentes. En este caso se recomienda que los cambios de versiones futuras de los protocolos de aplicación incorporen la sustitución de EXTERNAL por CHOICE {external EXTERNAL, embedded-pdv EMBEDDED PDV}. En la Rec. UIT-T X.681 | ISO/CEI 8824-2, Anexo C, se proponen sustituciones adicionales para utilizar EXTERNAL cuando no se utilice la negociación de la capa de presentación.

32.3 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 clase de valores que contienen ese valor de dato (una sintaxis abstracta); y
 - 2) la codificación utilizada (la sintaxis de transferencia) para distinguir ese valor de dato de otros valores en la misma clase.

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 a un contexto de presentación OSI para la identificación de 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 será codificada en una instancia de comunicación.

32.4 El tipo pdv incrustado tiene un tipo asociado. Este tipo se utiliza para dar precisión a la definición de los valores abstractos del tipo pdv incrustado y también para soportar las notaciones de valor y subtipo del tipo pdv incrustado.

32.5 El tipo asociado para la definición de valor y la subtipificación, suponiendo un entorno de rotulación automática, es (con comentarios normativos):

<pre>SEQUENCE { identification syntaxes abstract transfer -- Abstract and transfer syntax object identifiers --, syntax -- A single object identifier for identification of the class and encoding --, presentation-context-id -- (Applicable only to OSI environments) -- The negotiated presentation context identifies the class of the value and its encoding --,</pre>	<pre>CHOICE { SEQUENCE { OBJECT IDENTIFIER, OBJECT IDENTIFIER } OBJECT IDENTIFIER</pre>
---	---

```

context-negotiation                SEQUENCE {
  presentation-context-id          INTEGER,
  transfer-syntax                  OBJECT IDENTIFIER }
-- (Applicable only to OSI environments)
-- Context-negotiation in progress for a context to identify the class of the value
-- and its encoding --,

transfer-syntax                    OBJECT IDENTIFIER
-- The class 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 comunales que existen entre el tipo pdv incrustado (embedded-pdv), el tipo externo (external) y el tipo cadena de caracteres no restringida (unrestricted character string).

32.6 Para la alternativa "presentation-context-id", el valor entero será un identificador de contexto de presentación del conjunto de contextos. Esta alternativa no deberá ser utilizada en la petición P-CONEXIÓN ni en la petición P-ALTERACIÓN DE CONTEXTO para un contexto de presentación cuya adición o supresión está siendo propuesta por esas primitivas de petición.

NOTA – Incluso si existe una sola sintaxis de transferencia propuesta para un contexto de presentación en la lista de definiciones de contextos de presentación, la alternativa "presentation-context-id" no puede ser utilizada para ese contexto de presentación.

32.7 La alternativa "context-negotiation" deberá utilizarse solamente en la petición P-CONEXIÓN o en la petición P-ALTERACIÓN DE CONTEXTO y el valor entero será un identificador de contexto de presentación propuesto para su adición al conjunto de contextos definido. El identificador de objeto "transfer-syntax" identificará una sintaxis de transferencia propuesta para el contexto de presentación que se utiliza para codificar el valor.

32.8 La notación para un valor del tipo embedded-pdv será la notación del valor del tipo asociado definido en 32.5, donde el "data-value" OCTET STRING representa una codificación que utiliza la sintaxis de transferencia especificada en "identification".

EmbeddedPdvValue ::= SequenceValue *-- value of associated type defined in 32.5*

32.9 EJEMPLO 1 – Cuando un diseñador de aplicación desea que la codificación sea independiente de cualquier entorno de presentación (y, por tanto, que sea posible retransmitirla o almacenarla y recuperarla sin modificación), es necesario prohibir la utilización de las alternativas "presentation-context-id" y "context-negotiation". Esto puede hacerse escribiendo:

```

EMBEDDED PDV (WITH COMPONENTS {
  ... ,
  identification (WITH COMPONENTS {
    ... ,
    presentation-context-id          ABSENT,
    context-negotiation             ABSENT } ) } )

```

32.10 EJEMPLO 2 – 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 } ) } )

```

33 Notación para el tipo externo

33.1 El tipo externo (external) (véase 3.8.37) será referenciado por la notación "ExternalType":

ExternalType ::= EXTERNAL

33.2 Este tipo tiene un rótulo que es de clase universal, número 8.

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:
 - 1) una clase de valores que contienen ese valor de dato (una sintaxis abstracta); y
 - 2) la codificación utilizada (la sintaxis de transferencia) para distinguir ese valor de dato de otros valores en la misma clase; y
- c) (opcionalmente) un descriptor de objeto que proporciona una descripción legible por los seres humanos de la clase 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 32.3 también se aplica al tipo external.

33.4 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. En particular, la codificación en BER utiliza un tipo secuencia (sequence) equivalente, idéntico al que estaba presente en la definición del tipo externo de la Rec. X.208 del CCITT | ISO/CEI 8824, y las codificaciones de valores externos por BER permanecen inalteradas.

33.5 El tipo asociado para la definición de valor y la subtipificación, suponiendo un entorno de rotulación automática, es (con comentarios normativos):

<pre> SEQUENCE { identification syntaxes abstract transfer -- Abstract and transfer syntax object identifiers --, syntax -- A single object identifier for identification of the class and encoding --, presentation-context-id -- (Applicable only to OSI environments) -- The negotiated presentation context identifies the class of the value and its encoding --, context-negotiation presentation-context-id transfer-syntax -- (Applicable only to OSI environments) -- Context-negotiation in progress for a context to identify the class of the value -- and its encoding --, transfer-syntax -- The class 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 --, </pre>	<pre> CHOICE { SEQUENCE { OBJECT IDENTIFIER, OBJECT IDENTIFIER } OBJECT IDENTIFIER INTEGER SEQUENCE { INTEGER OBJECT IDENTIFIER } OBJECT IDENTIFIER </pre>
---	--

```

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 – El tipo externo no permite las alternativas "syntaxes", "transfer-syntax" o "fixed" de "identification". Estas alternativas no pueden autorizarse para el tipo externo por la necesidad de mantener la retrocompatibilidad con el tipo externo de la Rec. X.208 del CCITT | ISO/CEI 8824. 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.

33.6 El texto de 32.6 y 32.7 también se aplica al tipo external.

33.7 La notación para un valor de tipo externo deberá ser la notación de valor del tipo asociado definido en 33.5, donde el "data-value" OCTET STRING representa una codificación que utiliza la sintaxis de transferencia especificada en "identification".

ExternalValue ::= SequenceValue -- valor del tipo asociado definido en 33.5.

NOTA – Por razones históricas, los reglas de codificación pueden transferir valores incrustados en EXTERNAL, cuyas codificaciones no son múltiples exactos de ocho bits. Tales valores no pueden ser representados en la notación de valor utilizando el mencionado tipo asociado.

34 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 avance de espacio tales como THIN-SPACE y EN-SPACE de ISO/CEI 10646-1).

El término **carácter abstracto (abstract character)** denota la totalidad de la información asociada con una célula de una tabla repertorio de caracteres. 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 ASN.1 para los tipos cadenas de caracteres (character string) tiene tres variantes (que pueden combinarse), especificadas formalmente a continuación:

- Una representación impresa de los caracteres de la cadena utilizando el símbolo gráfico asignado, con la posible inclusión de caracteres con avance de espacio; ésta es la notación "cstring".

NOTA 1 – Una representación como esa puede resultar equívoca cuando se utilice la misma forma de carácter para más de un carácter del repertorio.

NOTA 2 – Una representación como esa puede resultar equívoca cuando se utilicen caracteres con avance de espacio o la especificación esté impresa con un tipo de carácter de espaciado proporcional.
- 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 ASN.1-CHARACTER-MODULE en la cláusula 37 de ISO/CEI 10646-1 para el repertorio 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 inciso anterior a) o en el c) siguiente.
- 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.

35 Notación para tipos cadena de caracteres

35.1 La notación para referenciar un tipo cadena de caracteres (véase 3.8.11) será:

CharacterStringType ::= RestrictedCharacterStringType | UnrestrictedCharacterStringType

"RestrictedCharacterStringType" es la notación para un tipo cadena de caracteres restringida y se define en la cláusula 36. "UnrestrictedCharacterStringType" es la notación para un tipo cadena de caracteres no restringidas y se define en 39.1.

35.2 El rótulo de cada tipo cadena de caracteres restringida se especifica en 36.1. El rótulo del tipo cadena de caracteres no restringida se especifica en 39.2.

35.3 La notación para un valor cadena de caracteres será:

CharacterStringValue ::= RestrictedCharacterStringValue | UnrestrictedCharacterStringValue

"RestrictedCharacterStringValue" se define en 36.7. "UnrestrictedCharacterStringValue" es la notación para un valor cadena de caracteres no restringida y se define en 39.6.

36 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 da los caracteres gráficos, por referencia a un número de registración en ISO International Register of Coded Character Sets (Registro internacional de conjuntos de caracteres codificados de la ISO) (véase *ISO International Register of Coded Character Sets*) o por referencia a la ISO/CEI 10646-1.

36.1 El cuadro 3 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 registración o la tabla 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 la tabla. Si la notación contiene la definición de un nombre sinónimo, éste aparecerá entre paréntesis.

NOTA – El rótulo asignado a tipos cadena de caracteres identifica inequívocamente el tipo. Obsérvese no obstante que si se emplea ASN.1 para definir nuevos tipos a partir de este tipo (en particular, utilizando IMPLICIT), puede resultar imposible reconocer estos tipos si no se conoce la definición del tipo ASN.1.

Cuadro 3 – Lista de tipos cadena de caracteres restringida

Nombre para referenciar el tipo	Número de clase universal	Número de registraci3n para la definici3n ^{a)} , número de cuadro o cláusula de la Rec. UIT-T X.680 ISO/CEI 8824-1	Notas
UTF8String	12	Véase 36.13	
NumericString	18	Cuadro 4	Nota 1
PrintableString	19	Cuadro 5	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	Nota 4
GeneralString	27	Todos los juegos G y C + SPACE + DELETE	
UniversalString	28	Véase 36.6	
BMPString	30	Véase 36.12	

a) Los números de registraci3n que proporcionan la definici3n 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 visualizaci3n no son significativos.

NOTA 2 – Las entradas que corresponden a estos números de registraci3n hacen referencia a la Rec. T.61 del CCITT para las reglas relativas a su utilizaci3n. Pueden utilizarse las entradas de registro 6 y 156 en vez de la 102 y 103.

NOTA 3 – Las entradas correspondientes a estos números de registraci3n dan la funcionalidad de la Rec. T.100 del CCITT y Rec. UIT-T T.101.

NOTA 4 – La referencia al registro 6 de "ISO International Register of Coded Character Sets to be used with Escape Sequences" constituye una referencia indirecta a ISO/CEI 646:1991. Esto representa un cambio con respecto a la Rec. X.208 del CCITT | ISO/CEI 8824, que referenciaba el registro 2 (referencia indirecta a ISO 646:1973). En las aplicaciones en las que se desee referenciar el número de registraci3n 2 se deberán emplear otras maneras de hacerlo [por ejemplo, utilizaci3n de la cadena de caracteres no restringida (véase la cláusula 39)] para llevar la definici3n antigua de VisibleString o referenciar la Rec. X.208 del CCITT | ISO/CEI 8824.

36.2 El cuadro 4 indica los caracteres que pueden aparecer en el tipo NumericString y en la sintaxis abstracta de caracteres NumericString.

Cuadro 4 – NumericString

Nombre	Símbolo gráfico
Dígitos	0, 1, ... 9
Espacio	(espacio)

36.3 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 identificaci3n del tipo cadena de caracteres separada del valor.

ISO/CEI 8824-1 : 1998 (S)

NOTA 2 – Un valor de una sintaxis abstracta de caracteres NumericString puede ser codificado por:

- 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 M.
- 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)}.

36.4 El cuadro 5 indica los caracteres que pueden aparecer en el tipo PrintableString y en la sintaxis abstracta de caracteres PrintableString.

Cuadro 5 – 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	-
Punto	.
Barra de fracción	/
Dos puntos	:
Signo igual	=
Signo de interrogación (final)	?

36.5 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:

- 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 M.
- 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)}.

36.6 Los caracteres que pueden aparecer en el tipo UniversalString son cualesquiera de los autorizados por ISO/CEI 10646-1, y la utilización de este tipo invoca los requisitos de conformidad especificados en ISO/CEI 10646-1, especialmente con respecto a la zona de uso restringido de ISO/CEI 10646-1.

NOTA 1 – La utilización de este tipo no constreñido está desaconsejada, pues la conformidad será generalmente imposible en la práctica.

NOTA 2 – La cláusula 37 define un módulo ASN.1 que contiene varios subtipos de este tipo para los "Collections of graphics characters for subsets" (Colecciones de caracteres gráficos para subconjuntos) definidas en ISO/CEI 10646-1, anexo A.

36.7 La notación de valor para los tipos restricted character string será "cstring" (véase 11.11), "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 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 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 37.1.2).

NOTA 2 – La cláusula 37 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), 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, 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 las letras "H", "O", "P" y "E" aparecen en los alfabetos LATÍN BÁSICO, CIRÍLICO y GRIEGO BÁSICO por lo que resultan ambiguas.

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"

36.8 El "DefinedValue" de "CharsDefn" deberá ser una referencia a un valor de ese tipo.

36.9 El "number" de las producciones "Plane", "Row" y "Cell" deberá ser inferior a 256 y en la producción "Group" deberá ser inferior a 128.

36.10 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 subtificación.

NOTA – Los diseñadores de aplicación deben analizar cuidadosamente las implicaciones de conformidad cuando se utilizan tipos cadena de caracteres abierta (open-ended character string), 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.

36.11 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*).

36.12 BMPString es un subtipo de UniversalString que tiene su propio rótulo exclusivo y modela el plano multilingüe básico (las primeras células 64K-2) 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 37) 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 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.

36.13 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 es diferente de la de UniversalString, y para la mayoría de los textos será menos extensa.

37 Denominación de caracteres y colecciones definidas en ISO/CEI 10646

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 carácter 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 de 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 37.1 son exportadas y tienen que ser importadas por cualquier módulo que las utilice.

37.1 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.

37.1.1 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}

```

37.1.2 Para cada entrada de cada lista de nombres de caracteres de los caracteres gráficos (glyphs) mostradas en las cláusulas 24 y 25 de ISO/CEI 10646-1, el módulo incluye un enunciado (statement) de la forma:

```

<namedcharacter> BMPString ::= <tablecell>
    -- represents the character <iso10646name>, see ISO/IEC 10646-1

```

donde:

- <iso10646name> es el nombre de carácter derivado de uno listado en ISO/CEI 10646-1;
- <namedcharacter> es una cadena obtenida aplicando a <iso10646name> los procedimientos especificados en 37.2;
- <tablecell> es el glyph 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, 145}
    -- represents the character GREEK CAPITAL LETTER SIGMA, see ISO/IEC 10646-1

```

37.1.3 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:

- <collectionstring> es el nombre para la colección de caracteres asignada en ISO/CEI 10646-1;
- <namedcollectionstring> se forma aplicando a <collectionstring> los procedimientos de 37.3;
- <alternativelist> se forma utilizando los <namedcharacter>s generados como se indica en 37.2 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 D.)

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.

```

37.1.4 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 CombiningCharactersB-2))

Level3 ::= BMPString

NOTA 1 – Los "CombiningCharacters" y los "CombiningCharactersB-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 restricción en una "ConstraintSpec". Véase un ejemplo en C.2.7.1.

NOTA 3 – Para más información sobre este tema véase D.2.5.

37.1.5 El módulo se termina con el enunciado:

END

37.1.6 Un equivalente al ejemplo de 37.1.3 definido por el usuario es:

BasicLatin ::= BMPString (FROM (space..tilde))

-- represents the collection of characters BASIC LATIN, see ISO/IEC 10646-1.

37.2 Un <namedcharacter> es la cadena obtenida tomando un <iso10646name> (véase 37.1.2) 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 tienen la representación gráfica "<" puede ser referenciado mediante el "DefinedValue" de:

less-thanSign

37.3 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.

38 Orden canónico de los caracteres

38.1 A efectos de subtipificación de "ValueRange" y posible utilización por las reglas de codificación se especifica un ordenamiento canónico de caracteres para UniversalString, BMPString, NumericString, PrintableString, VisibleString, e IA5String.

38.2 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.

38.3 El orden canónico de un carácter abstracto está definido por el orden canónico de su célula.

38.4 Para UniversalString, el orden canónico de las células se define (véase ISO/CEI 10646-1) como:

$$256*(256*(128*(\text{número de grupo})+(\text{número de plano}))+(\text{número de fila}))+(\text{número de célula})$$

El conjunto de caracteres contiene, en su totalidad, exactamente $128*256*256*256$ caracteres. 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) dando el símbolo gráfico de una "cstring" (ASN1-CHARACTER-MODULE se define en 37.1). No es posible especificar una célula como un punto extremo de una gama o identificar un carácter individual cuando no se hayan asignado nombres o símbolos gráficos a esa célula.

38.5 Para BMPString, el orden canónico de las células se define (véase ISO/CEI 10646-1) como:

$$256*(\text{número de fila})+(\text{número de célula})$$

El conjunto de caracteres contiene, en su totalidad, exactamente $256*256$ caracteres. Los puntos extremos de las "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) dando el símbolo gráfico de una "cstring". No es posible especificar una célula como un punto extremo de una gama o identificar un carácter individual cuando no se hayan asignado nombres o símbolos gráficos a esa célula.

38.6 Para NumericString, el orden canónico, creciente de izquierda a derecha, se define (véase el cuadro 4 de 36.2) 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.

38.7 Para PrintableString, el orden canónico, creciente de izquierda a derecha y de arriba abajo, se define (véase el cuadro 5 de 36.4) como:

(espacio) (apóstrofo) (paréntesis izquierdo) (paréntesis derecho) (signo más) (coma) (guión) (punto)
(barra de fracción) 0123456789 (dos puntos) (signo igual) (signo de interrogación final)
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.

38.8 Para VisibleString, el orden canónico de las células se define a partir de la codificación de ISO 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 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".

38.9 Para IA5String, el orden canónico de las células se define a partir de la codificación ISO 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 37.1.1.

39 Definición de tipos cadena de caracteres no restringida

Esta cláusula define un tipo cuyos valores son los valores de cualquier sintaxis abstracta de caracteres. Esta sintaxis abstracta puede formar parte del conjunto de contexto definido en un ejemplar de comunicación, o ser referenciada 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 instancias específicas o grupos de instancias de CHARACTER STRING. Por lo general, la referencia a las sintaxis soportadas se incluirá en un formulario de enunciado de conformidad de implementación de protocolo (PICS, *protocol implementation conformance statement*). Obsérvese que la agrupación de casos para fines de especificación de la capa aplicación puede conseguirse utilizando diferentes referencias de tipo ASN.1 (todas las cuales serían referencias para el tipo CHARACTER STRING).

39.1 El tipo cadena de caracteres no restringida (véase 3.8.69) será referenciado por la notación "CharacterStringType":

UnrestrictedCharacterStringType ::= CHARACTER STRING

39.2 Este tipo tiene un rótulo que es de clase universal, número 29.

39.3 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 clase de valores que contiene dicho valor cadena de caracteres (una sintaxis abstracta de caracteres); y
 - 2) la codificación utilizada (la sintaxis de transferencia de carácter) para distinguir dicho valor cadena de caracteres de otros valores en la misma clase.

39.4 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.

39.5 El tipo asociado para la definición de valor y la subtipificación, suponiendo un entorno de rotulación automática, es (con comentarios normativos):

<pre> SEQUENCE { identification syntaxes abstract transfer -- Abstract and transfer syntax object identifiers --, syntax -- A single object identifier for identification of the class and encoding --, presentation-context-id -- (Applicable only to OSI environments) -- The negotiated presentation context identifies the class of the value and its encoding --, context-negotiation presentation-context-id transfer-syntax -- (Applicable only to OSI environments) -- Context-negotiation in progress for a context to identify the class of the value -- and its encoding --, </pre>	<pre> CHOICE { SEQUENCE { OBJECT IDENTIFIER, OBJECT IDENTIFIER } OBJECT IDENTIFIER INTEGER SEQUENCE { INTEGER, OBJECT IDENTIFIER } </pre>
--	---

transfer-syntax**OBJECT IDENTIFIER**

-- The class 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 comunales que existen entre el tipo pdv incrustado, el tipo externo y el tipo cadena de caracteres no restringida.

39.6 La notación de valor debe ser la notación de valor del tipo asociado, donde el valor del OCTET STRING "string-value" representa una codificación que utiliza la sintaxis de transferencia especificada en "identification".

UnrestrictedCharacterStringValue ::= SequenceValue -- value of associated type defined in 39.4

39.7 En C.2.8 figura un ejemplo de tipo cadena de caracteres no restringida.

40 Notación para tipos definidos en las cláusulas 41 a 43

40.1 La notación para referenciar un tipo definido en las cláusulas 41 a 43 será:

UsefulType ::= typereference

donde "typereference" es una de las definidas en las cláusulas 41 a 43 utilizando la notación ASN.1.

40.2 El rótulo de cada "UsefulType" se especifica en las cláusulas 41 a 43.

41 Generalized time (tiempo generalizado) (u hora generalizada)

41.1 Este tipo será referenciado por el nombre:

GeneralizedTime

41.2 Está constituido por valores que representan:

- una fecha calendario (calendar date), definida en ISO 8601; y
- 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
- el factor diferencial de hora local, definido en ISO 8601.

41.3 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:

- una cadena que representa la fecha 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
- los caracteres señalados en el inciso a) anterior seguidos por una letra Z mayúscula; o
- los caracteres señalados en el inciso a) anterior seguidos por una cadena que representa un diferencial de hora local, como se especifica en ISO 8601, sin separadores.

ISO/CEI 8824-1 : 1998 (S)

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.

41.4 El rótulo será como se define en 41.3.

41.5 La notación de valor será la notación de valor para la "VisibleString" definida en 41.3.

42 Tiempo universal

42.1 Este tipo será referenciado por el nombre:

UTCTime

42.2 El tipo está constituido por valores que representan:

- a) una fecha 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.

42.3 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 hhmms 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.

La alternativa de b) permite variar la precisión en la especificación de la hora.

En la alternativa c) 1), la hora es hora de 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 de tiempo universal coordinado de la forma siguiente:

Hora de 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"; o
- "8201020700-0500".

EJEMPLO 2 – Si la hora local es las 7 de la mañana del 2 de enero de 2001 y la hora de tiempo universal coordinado es las doce del mediodía del 2 de enero de 2001, el valor de UTCTime es uno de los siguientes:

- "0101021200Z"; o
- "0101020700-0500".

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 Tipo descriptor de objeto

43.1 Este tipo será referenciado por el nombre:

ObjectDescriptor

43.2 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.

43.3 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.

43.4 El rótulo será como se define en 43.3.

43.5 La notación de valor será la notación de valor para la "GraphicString" definida en 43.3.

44 Tipos constreñidos

44.1 La notación "tipo constreñido" ("ConstrainedType") permite aplicar una restricción 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 restricción.

ConstrainedType ::=
Type Constraint |
TypeWithConstraint

En la primera alternativa, el tipo progenitor es "Type", y la restricción es especificada por "Constraint" como se define en 44.5. La segunda alternativa se define en 44.4.

44.2 Cuando la notación "Constraint" sigue a una notación de tipo conjunto de o secuencia de, 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 restricción "(SIZE(1..64))" se aplica a la VisibleString, no a la SEQUENCE OF:

NamesOfMemberNations ::= SEQUENCE OF VisibleString (SIZE(1..64))

44.3 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.

44.4 Como consecuencia de la interpretación especificada en 44.2, se proporciona una notación especial para permitir aplicar una restricción 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
```

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 primera y tercera, la restricción es "Constraint" (véase 44.5), mientras que en las alternativas segunda y cuarta es "SizeConstraint" (véase 48.5).

NOTA – Aunque las alternativas "Constraint" comprenden las correspondientes alternativas "SizeConstraint", estas últimas, que no van encerradas entre corchetes, se proporcionan a efectos de retrocompatibilidad con la Rec. X.208 del CCITT | ISO/CEI 8824.

44.5 Una restricción se especifica por la notación "Constraint":

```
Constraint ::= "(" ConstraintSpec ExceptionSpec ")"
ConstraintSpec ::=
    SubtypeConstraint |
    GeneralConstraint
```

"ExceptionSpec" se define en la cláusula 45. A menos que se utilice junto con un "marcador de extensión" (véase la cláusula 47), 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).

44.6 La notación "SubtypeConstraint" es la notación "ElementSetSpec" de propósito general (cláusula 46):

```
SubtypeConstraint ::= ElementSetSpec
```

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.

45 Identificador de excepción

45.1 En una especificación ASN.1 compleja hay un cierto número de sitios en los que se reconoce específicamente que los codificadores han de tratar material que no está completamente especificado en ella. Estos casos surgen, en particular, como consecuencia del empleo de una restricción que se ha definido utilizando un parámetro de la sintaxis abstracta (véase la Rec. UIT-T X.683 | ISO/CEI 8824-4, cláusula 10).

45.2 En tales casos, el diseñador de aplicaciones tiene que identificar las acciones que han de tomarse cuando se viole alguna restricción 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 un 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.

45.3 Si está presente un identificador de excepción, ello indica que hay texto en el cuerpo de la norma en el que se dice cómo tratar la violación de la restricción asociada con el "!". Si está ausente, 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 restricción.

45.4 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").

45.5 Donde un tipo esté constreñido por restricciones múltiples, de las cuales más de una tengan un identificador de excepción, el identificador de excepción de la restricción más exterior será considerado como el identificador de excepción para ese tipo.

45.6 Donde esté presente un marcador de excepción sobre tipos que se utilizan en aritmética de conjuntos, el identificador de excepción en la construcción más externa se ignora y no es heredado por el tipo que se está construyendo como resultado de la aritmética de conjuntos.

46 Especificación de conjunto de elementos

46.1 En algunas notaciones puede especificarse un conjunto de elementos de alguna clase de elemento identificada (el gobernador). En esos casos se utiliza la notación "ElementSetSpec":

```

ElementSetSpec ::=
    RootElementSetSpec |
    RootElementSetSpec "," "..." |
    "..." "," AdditionalElementSetSpec |
    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 signo 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 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 limitaciones sin la utilización del paréntesis entorno a "ALL EXCEPT xxx".

NOTA 3 – Dondequiera que aparezca "Elements", puede aparecer una construcción sin paréntesis [por ejemplo, INTEGER (1..4)] o una limitación 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 "ElementSetSpec" son la unión de los elementos referenciados por "RootElementSetSpec" y "AdditionalElementSetSpec".

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 La notación "Elements" se define como sigue:

```

Elements ::=
    SubtypeElements      |
    ObjectSetElements   |
    "(" ElementSetSpec ")"
    
```

Los elementos especificados por esta notación son:

- a) los descritos en la cláusula 48, 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, 12.6, 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.

47 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.

47.1 El marcador de extensión, elipsis, es una indicación de que se esperan adiciones de extensión. No establece enunciados de cómo deben manejarse dichas adiciones salvo que no deben tratarse como un error durante el proceso de decodificación.

47.2 La utilización conjunta de un marcador de extensión y de un identificador de excepción es una indicación de que se esperan adiciones de extensión y que por lo tanto no deben tratarse como un error en el proceso de decodificación y que las normas de aplicación prescriben que la aplicación tome medidas específicas si existe una violación de constricción. Se recomienda que esta notación se utilice en aquellas situaciones en las que se está utilizando el procedimiento de almacenamiento y retransmisión o cualquier otra forma de retransmisión, de manera que se indique que cualquier adición de extensión no reconocida se devuelva a la aplicación para una posible recodificación y retransmisión.

47.3 Si existe aritmética de conjuntos en la notación "ElementSetSpecs", se realizará sin tener en cuenta la presencia del marcador de extensión.

NOTA – En otras palabras, la presencia de un marcador de extensión no produce efecto en la aritmética de conjuntos.

47.4 Si un tipo definido con una constricción extensible se referencia en un "ContainedSubtype", el tipo definido después no hereda el marcador de extensión o cualquiera de sus adiciones de extensión. Si se desea que el tipo definido después sea extensible, deberá añadirse explícitamente un marcador de extensión a su "ElementSetSpecs". 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.
    
```

47.5 Si un tipo definido con una constricción extensible sufre una ulterior constricción con un "ElementSetSpecs" que no contiene un marcador de extensión, el tipo resultante es uno cuya constricción no es extensible y que no hereda ninguna adición de extensión que pueda estar presente en el tipo progenitor. Por ejemplo:

```

A ::= INTEGER (0..10, ...)               -- A is extensible.

B ::= A (2..5)                           -- B is inextensible.

C ::= A                                    -- C is extensible.
    
```

47.6 Los componentes de un tipo conjunto, secuencia o elección que hayan sido constreñidos 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.
```

47.7 Cuando esta Recomendación | Norma Internacional requiera r tulos distintos (v anse 24.5 a 24.6, 26.3 y 28.2), se aplicar n conceptualmente las siguientes transformaciones antes de efectuar la comprobaci n de la unicidad de r tulo:

47.7.1 Se a ade un nuevo componente o alternativa si:

- no hay marcadores de extensi n pero la extensibilidad est  impl cita en el encabezamiento del m dulo, y seguidamente se a ade un marcador de extensi n y se a ade el nuevo elemento como primera adici n despu s de ese marcador de extensi n; o
- en un CHOICE o SEQUENCE o SET hay un solo marcador de extensi n y seguidamente se a ade el nuevo elemento al final de CHOICE o SEQUENCE o SET inmediatamente antes del corchete de cierre; o
- hay dos marcadores de extensi n en CHOICE o SEQUENCE o SET y seguidamente se a ade el nuevo elemento inmediatamente antes del segundo marcador de extensi n.

47.7.2 Este elemento a adido conceptualmente s lo sirve para comprobar la legalidad mediante la aplicaci n de reglas que requieren r tulos distintos (v anse 24.5 a 24.6, 26.3 y 28.2). Se a ade conceptualmente *despu s* de la aplicaci n de la rotulaci n autom tica (si procede) y la expansi n de COMPONENTS OF.

47.7.3 El elemento a adido conceptualmente se define para que tenga un r tulo distinto del de todos los tipos ASN.1 normales, pero que corresponda al r tulo de todos los elementos a adidos conceptualmente, y corresponda al r tulo indeterminado del tipo abierto, como se especifica en la nota 2 de la Rec. UIT-T X.861 | ISO/CEI 8824-2, 14.2.

NOTA – Las reglas concernientes a la unicidad de r tulos relativos al elemento a adido conceptualmente y al tipo abierto, junto con las reglas que requieren distintos r tulos (v anse 24.5 a 24.6, 26.3 y 28.2) son necesarios y suficientes para asegurar que:

- cualquier adici n de extensi n desconocida puede inequ vocamente atribuirse a un  nico punto de inserci n cuando una codificaci n BER es decodificada; y
- adiciones de extensi n desconocidas no pueden ser confundidas con elementos OPTIONAL.

En PER las reglas mencionadas son suficientes pero no necesarias para asegurar estas propiedades. No est n impuestas como reglas de ASN.1 para asegurar la independencia de la notaci n partiendo de las reglas de codificaci n.

47.7.4 Si con estos elementos a adidos conceptualmente se violan las reglas que requieren tipos distintos, la especificaci n ha hecho uso ilegalmente de la notaci n de extensibilidad.

NOTA – La finalidad de estas reglas es efectuar restricciones precisas que surjan de la utilizaci n de puntos de inserci n (especialmente los que no est n al final de las SEQUENCE o SET o CHOICE). Las restricciones est n destinadas a garantizar que en BER, DER y CER es posible atribuir inequ vocamente a un punto de inserci n espec fico un elemento desconocido recibido por una primera versi n de sistema. Esto resultaría importante si el tratamiento de excepci n de esos  tems a adidos fuera diferente para puntos de inserci n diferentes.

47.8 Ejemplos

47.8.1 Ejemplo 1

```
A ::= SET {
  a  A,
  b  CHOICE {
    c  C,
    ... ,
    ... ,
    d  D
  }
}
```

es legal porque no hay ambigüedad, dado que cualquier material a adido debe ser parte de "b".

47.8.2 Ejemplo 2

```

A ::= SET {
  a  A,
  b  CHOICE {
      c      C,
      ... ,
      ... ,
      d      D
  },
  ... ,
  e  E
}

```

es ilegal porque el material añadido puede ser parte de "b", o puede estar en el nivel más exterior de "A", y una primera versión de sistema no puede determinar de cual se trata.

47.8.3 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".

47.8.4 Se pueden dar 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 que no está presente en la primera versión pueda ser atribuido inequívocamente por una primera versión de sistema a precisamente un punto de inserción.

48 Elementos subtipos

48.1 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 6 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

```

Cuadro 6 – Aplicabilidad de conjuntos de valores subtipo

Tipo	Valor único	Subtipo contenido	Gama de valores	Limitación de tamaño	Alfabeto permitido	Limitación de tipo	Subtipif. interior
Cadena de bits (bit string)	Sí	Sí	No	Sí	No	No	No
Booleano	Sí	Sí	No	No	No	No	No
Elección (Choice)	Sí	Sí	No	No	No	No	Sí
pdv incrustado (embedded-pdv)	Sí	No	No	No	No	No	Sí
Enumerado (enumerated)	Sí	Sí	No	No	No	No	No
Externo (external)	Sí	No	No	No	No	No	Sí
Ejemplar de (instance-of)	Sí	Sí	No	No	No	No	Sí
Entero (integer)	Sí	Sí	Sí	No	No	No	No
Nulo (null)	Sí	Sí	No	No	No	No	No
Tipo campo de clase de objeto (object class field type)	Sí	Sí	No	No	No	No	No
Identificador de objeto (object identifier)	Sí	Sí	No	No	No	No	No
Cadena de octetos (octet string)	Sí	Sí	No	Sí	No	No	No
Tipo abierto (open type)	No	No	No	No	No	Sí	No
Real (real)	Sí	Sí	Sí	No	No	No	Sí
Tipos cadena de caracteres restringida (restricted character string types)	Sí	Sí	Sí ^{a)}	Sí	Sí	No	No
Secuencia (sequence)	Sí	Sí	No	No	No	No	Sí
Secuencia de (sequence-of)	Sí	Sí	No	Sí	No	No	Sí
Conjunto (set)	Sí	Sí	No	No	No	No	Sí
Conjunto de (set-of)	Sí	Sí	No	Sí	No	No	Sí
Tipo cadena de caracteres no restringida (unrestricted character string type)	Sí	No	No	Sí	No	No	Sí

a) Sólo se permite dentro del "PermittedAlphabet" de BMPString, IA5String, NumericString, PrintableString, VisibleString y UniversalString.

48.2 Single Value (valor único)

48.2.1 La notación "SingleValue" será:

SingleValue ::= Value

donde "Value" es la notación de valor para el tipo progenitor.

48.2.2 Un "SingleValue" especifica el valor único del tipo progenitor especificado por "Value".

48.3 Contained Subtype (subtipo contenido)

48.3.1 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.

48.3.2 Un "ContainedSubtype" especifica todos los valores del tipo progenitor resultantes de la intersección del tipo progenitor y "Type". "Type" ha de derivarse del mismo tipo insertado que el tipo progenitor.

48.4 Value Range (gama de valores)

48.4.1 La notación "ValueRange" será:

ValueRange ::= LowerEndpoint ".." UpperEndpoint

48.4.2 Una "ValueRange" especifica todos los valores de una gama de valores que se designan especificando los valores de los puntos extremos de la gama. Esta notación sólo puede aplicarse a tipos enteros, al PermittedAlphabet de determinados tipos cadena de caracteres restringida (IA5String, NumericString, PrintableString, VisibleString, BMPString y UniversalString únicamente) y a tipos reales.

NOTA – A los fines de la subtipificación, "PLUS-INFINITY" es mayor que todos los valores "NumericReal" y "MINUS-INFINITY" es menor que todos los valores "NumericReal".

48.4.3 Cada punto extremo de la gama 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

48.4.4 Un punto extremo puede también no estar especificado, en cuyo caso la gama se extiende en ese sentido tanto como lo permita el tipo progenitor:

LowerEndValue ::= Value | MIN

UpperEndValue ::= Value | MAX

48.5 Size Constraint (constricción de tamaño)

48.5.1 La notación "SizeConstraint" será:

SizeConstraint ::= SIZE Constraint

48.5.2 Una "SizeConstraint" sólo puede aplicarse a tipos bit string, octet string, character string, set-of o sequence-of, o a tipos formados por rotulación a partir de cualquiera de estos tipos.

48.5.3 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".

48.5.4 La unidad de medida depende del tipo progenitor, como sigue:

<i>Tipo</i>	<i>Unidad de medida</i>
bit string	bit
octet string	octeto
character string	carácter
set-of	valor componente
sequence-of	valor componente

NOTA – La cuenta del número de caracteres especificado en esta subcláusula para determinar el tamaño de un valor 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 registración de un registro que puedan aparecer en esa definición.

48.6 Constricción de tipo

48.6.1 La notación "TypeConstraint" será:

TypeConstraint ::= Type

48.6.2 Esta notación sólo se aplica a una notación de tipo abierto y constriñe el tipo abierto a valores de "Type".

48.7 Alfabeto permitido

48.7.1 La notación "PermittedAlphabet" será:

PermittedAlphabet ::= FROM Constraint

48.7.2 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.

48.7.3 La "Constraint" es cualquiera que pueda aplicarse al tipo progenitor (véase el cuadro 6), 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".

48.8 Inner Subtyping (subtipificación interior)

48.8.1 La notación "InnerTypeConstraints" será:

InnerTypeConstraints ::=
WITH COMPONENT SingleTypeConstraint |
WITH COMPONENTS MultipleTypeConstraints

48.8.2 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 48.8.6). Esta notación puede aplicarse a los tipos set-of, sequence-of, set, sequence y choice, o a los tipos formados por rotulación a partir de ellos.

48.8.3 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.

48.8.4 Para los tipos que se definen en términos de múltiples otros tipos (interiores) (choice, set y sequence) puede proporcionarse un número de constricciones sobre estos tipos interiores. La notación para estas constricciones es "MultipleTypeConstraints":

MultipleTypeConstraints ::= FullSpecification | PartialSpecification

FullSpecification ::= "{" TypeConstraints "}"

PartialSpecification ::= "{" "... " ," TypeConstraints "}"

TypeConstraints ::=

NamedConstraint |
NamedConstraint "," TypeConstraints

NamedConstraint ::=

identifier ComponentConstraint

48.8.5 "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 a que se aplica la constricción se identifica por medio de su identificador. Para un componente dado deberá haber por lo menos una "NamedConstraint".

48.8.6 "MultipleTypeConstraints" comprende una "FullSpecification" o una "PartialSpecification". Cuando se utiliza "FullSpecification", hay una restricción de presencia implicada, de "ABSENT", en todos los tipos interiores que puedan ser restringidos a estar ausentes (véase 48.8.9) y que no está explícitamente listada. Cuando se emplea "PartialSpecification", no hay restricciones implicadas y cualquier tipo interior puede ser omitido de la lista.

48.8.7 Un determinado tipo interior puede ser restringido en cuanto a su presencia (en valores de tipo progenitor) o su valor, o en cuanto a ambos. La notación es "ComponentConstraint":

ComponentConstraint ::= ValueConstraint PresenceConstraint

48.8.8 Una restricción sobre el valor de un tipo interior se expresa por la notación "ValueConstraint":

ValueConstraint ::= Constraint | empty

La restricción se satisface por un valor del tipo progenitor únicamente si el valor interior pertenece al subtipo especificado por la "Constraint" aplicada al tipo interior.

48.8.9 Una restricción 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 48.8.9.1 a 48.8.9.3.

48.8.9.1 Si el tipo progenitor es sequence o set, un tipo componente marcado "OPTIONAL" puede ser restringido a estar "PRESENT" (en cuyo caso la restricción se satisface únicamente si el valor componente correspondiente está presente) o a estar "ABSENT" (en cuyo caso la restricción se satisface únicamente si el valor componente está ausente) o a ser "OPTIONAL" (en cuyo caso no se imponen restricciones a la presencia del valor componente correspondiente).

48.8.9.2 Si el tipo progenitor es una elección, un tipo componente puede ser restringido a estar "ABSENT" (en cuyo caso la restricción se satisface únicamente si el tipo componente correspondiente no se utiliza en el valor), o "PRESENT" (en cuyo caso la restricción se satisface únicamente si el tipo componente correspondiente se utiliza en el valor); deberá haber a lo sumo una palabra clave "PRESENT" en "MultipleTypeConstraints".

NOTA – Véase un ejemplo aclarador en C.4.6.

48.8.9.3 El significado de una "PresenceConstraint" vacía depende de que se esté empleando una "FullSpecification" o una "PartialSpecification":

- a) en una "FullSpecification", equivale a una restricción de "PRESENT" para un componente set o sequence marcado OPTIONAL y, en otro caso, no impone ninguna otra restricción;
- b) en una "PartialSpecification", no se imponen restricciones.

Anexo A

Utilización de la notación ASN.1-88/90

(Este anexo es parte integrante de esta Recomendación | Norma Internacional)

A.1 Mantenimiento

La expresión **notación ASN.1-88/90** se utiliza para referirse a la notación ASN.1 especificada en la Rec. X.208 del CCITT | ISO/CEI 8824. La expresión **notación ASN.1 actual** se utiliza para referirse a la especificada en la presente Recomendación | Norma Internacional.

En el momento de publicarse la presente Recomendación | Norma Internacional se mantenía todavía la Rec. X.208 del CCITT | ISO/CEI 8824. Este mantenimiento continuado depende de una Resolución anual por parte del JTC1/SC21 de ISO/CEI y no cabe esperar que sea indefinido. Se proporciona para dar tiempo a los usuarios de ASN.1 a sustituir las características (sobre todo ANY y la utilización de la notación macro) de la notación ASN.1-88/90 por la notación ASN.1 actual. (Esto puede hacerse sin cambios en los bits de la línea.)

A.2 Combinación de la notación ASN.1-88/90 con la notación ASN.1 actual

Tanto la notación ASN.1-88/90 como la notación ASN.1 actual especifican una construcción sintáctica de nivel superior que es un módulo ASN.1. Un usuario de ASN.1 escribe una colección de módulos ASN.1 y puede importar definiciones de otros módulos ASN.1.

Para cualquier módulo dado, la notación utilizada ha de estar (completamente) conforme con la notación ASN.1-88/90 o con la notación ASN.1 actual y una especificación de usuario deberá identificar claramente qué notación se está utilizando (por referencia a la Recomendación | Norma Internacional apropiada) para cada módulo incluido textualmente en la Especificación del usuario.

Obsérvese que podría ocurrir que un usuario deseara modificar parte de un módulo para utilizar la nueva notación, pero dejando otras partes en la notación antigua. Esto puede conseguirse (solamente) dividiendo el módulo en dos módulos.

Cuando un módulo esté conforme con la notación ASN.1-88/90, las referencias de tipo y valor pueden importarse desde un módulo que se definió utilizando la notación ASN.1 actual. Esos tipos y valores deben estar asociados con tipos que pueden ser definidos utilizando solamente la notación ASN.1-88/90. Por ejemplo, un módulo escrito utilizando la notación ASN.1-88/90 no puede importar un valor del tipo UniversalString, ya que este tipo está definido en la notación actual pero no en ASN.1-88/90; puede, sin embargo, importar valores cuyos tipos sean, por ejemplo, INTEGER, IA5String, etc.

Cuando un módulo es conforme con la notación ASN.1 actual, las referencias de tipo y valor pueden importarse desde un módulo que se definió utilizando la notación ASN.1-88/90. No se importará macro ASN.1. La notación de valor de un tipo importado solamente se utilizará en el módulo importador si están presentes identificadores para valores de SET y SEQUENCE y CHOICE utilizados en la notación de valor, y si en la notación de valor no se exige ningún valor del tipo ANY. No se aplicará una restricción de tipo interior a un tipo importado si el componente que se restringe no tiene un identificador.

A.3 Paso a la notación ASN.1 actual

Cuando se modifique un módulo (escrito originalmente para cumplir con la notación ASN.1-88/90) de modo que esté conforme con la notación actual, deberán tenerse en cuenta los siguientes puntos:

- a) A todos los componentes de SET y SEQUENCE y CHOICE se les deberá dar identificadores que sean inequívocos dentro de ese SET, SEQUENCE o CHOICE y esos identificadores deberán ser incluidos en la notación de valor.

NOTA 1 – La notación de valor para un tipo CHOICE contiene un signo de dos puntos (":").

- b) Todas las utilizaciones de ANY y ANY DEFINED BY deberán ser soportadas mediante una definición de clase de objeto de información adecuada, con el ANY y el ANY DEFINED BY (y los componentes referenciados) sustituidos por referencias apropiadas a los campos de esa clase de objeto. En la mayoría de los casos, la especificación puede mejorarse en buena medida fijándose atentamente en la inserción de constricciones de tabla y relación de componentes. En muchos casos, la especificación puede mejorarse más aún si en la construcción de tabla o relación de componentes se hace un parámetro del tipo.
- c) Todas las definiciones de macros serán sustituidas por la definición de una clase de objeto de información, un tipo parametrizado o un valor parametrizado. Si la cláusula WITH SYNTAX se diseña cuidadosamente en la definición de una clase de objeto de información, la notación utilizada para definir un objeto de esa clase puede hacerse muy similar a la notación definida por la utilización antigua de la notación de macro.
- d) Todos los casos de utilización de un macro deberán ser sustituidos por definiciones de objetos de información equivalentes o por referencias a "ObjectClassFieldType" equivalentes, tipos parametrizados o valores parametrizados. En la mayoría de los casos, la especificación de objetos de información puede mejorarse en gran medida agrupando esas informaciones en conjuntos de objetos de información e indicando claramente si es obligatorio soportar todos los objetos de información del conjunto y si las ampliaciones, dependientes de la implementación, de ese conjunto de objetos de información han de ser acomodadas por las implementaciones receptoras, y si es así, cómo deben tratar la recepción de valores "desconocidos". También puede ser conveniente considerar la posibilidad de que una versión posterior de la especificación del usuario amplíe el conjunto de objetos de información y orientar a los implementadores actuales respecto a cómo deben tratarse esas ampliaciones.
- e) Todas las ocurrencias de EXTERNAL deben examinarse cuidadosamente; si bien esa notación sigue siendo lícita en la ASN.1 actual, una Especificación de usuario podría mejorarse, probablemente, haciendo lo siguiente:
 - 1) Considerar la utilización de la notación INSTANCE OF (preferiblemente con una construcción de tabla que puede ser un parámetro del tipo, según se examinó anteriormente para ANY y ANY DEFINED BY) en lugar de la notación EXTERNAL; en muchos casos esto no cambiará los bits en la línea.
 - 2) Cuando se retenga EXTERNAL, la utilización de la subtipificación interior del tipo asociado (véase 33.5) puede ayudar a dar precisión a la especificación de si el empleo de identificadores del contexto de presentación está o no permitido. También son aplicables aquí comentarios anteriores (véase la cláusula 33) en los que se dan directrices respecto a qué valores de EXTERNAL se han de soportar y qué implementaciones deben hacerlo si se reciben valores no soportados.
 - 3) Considerar un cambio a:

CHOICE {external EXTERNAL, embedded-pdv EMBEDDED PDV}

(de nuevo con subtipificación interior, si procede) para permitir una migración por fases de las aplicaciones pares distribuidas hacia la notación actual. Esto puede influir en los bits de la línea y normalmente debería hacerse como parte de un cambio de versión en el protocolo. La utilización de EMBEDDED PDV (sobre todo para especificaciones nuevas) dará normalmente una mayor flexibilidad, según puede verse por comparación de los tipos asociados; además, EMBEDDED PDV es codificado más eficazmente que EXTERNAL por todas las reglas de codificación especificadas en la Rec. UIT-T X.690 | ISO/CEI 8825-1.
- f) Es posible mejorar la legibilidad de la notación en los módulos ASN.1 existentes (sin cambios en los bits en la línea) mediante la inserción de AUTOMATIC TAGS en el encabezamiento del módulo y la supresión de algunos rótulos o de la totalidad de los mismos.

NOTA 2 – Esto es algo que hay que hacer con cuidado y con conocimiento del funcionamiento de la rotulación automática ya que, si se aplica de manera incorrecta, se producirá un cambio de los bits de la línea.

- g) Si no se aplica AUTOMATIC TAGS, como se describe en f), a los módulos existentes, normalmente convendrá no añadir nuevas definiciones de tipos al módulo existente sino más bien crear un nuevo módulo (con rotulación automática) para las definiciones de tipos nuevos. De esta manera es posible aprovechar las ventajas de la rotulación automática sin afectar a los bits de la línea.
- h) Deberá prestarse atención a los campos que contengan cadenas de caracteres para ver si debe emplearse la notación CHARACTER STRING, BMPString o UniversalString. Sin embargo esto cambiará, normalmente, los bits de la línea y debería hacerse como parte de un cambio de versión.
- i) Los identificadores "mantissa", "base" y "exponent" han de añadirse a cualquier notación de valor real que utilice la alternativa "NumericRealValue" de la producción "RealValue". Deberá considerarse la restricción de la "base" a 2 ó 10 en la notación de tipo.

Por lo general, pueden producirse mejoras importantes en lo tocante a legibilidad, eficacia, precisión y flexibilidad al pasar a utilizar la nueva notación ASN.1 (sobre todo si se aprovechan plenamente las ventajas de la utilización de las constricciones de tabla y relación de componentes y la parametrización, así como de la utilización de los nuevos tipos cadena de caracteres). Se insta encarecidamente a todos los usuarios de ASN.1-88/90 a que acometan el cambio cuando revisen una Especificación, o como una actividad independiente, cuando no se prevean revisiones durante algún tiempo.

Por lo general se considera un error efectuar adiciones a módulos existentes utilizando una notación no conforme con la especificación ASN.1 actual, incluso si se retienen referencias a las especificaciones ASN.1-88/90 para esos módulos. Deberán evitarse, en particular, nuevas utilizaciones de macros y de ANY o ANY DEFINED BY o nuevas construcciones de SET, SEQUENCE o CHOICE sin identificadores inequívocos.

Anexo B**Asignación de valores de identificador de objeto**

(Este anexo es parte integrante de esta Recomendación | Norma Internacional)

En esta Recomendación | Norma Internacional se asignan los siguientes valores:

Subcláusula	Valor de identificador de objeto
36.3	{ joint-iso-itu-t asn1(1) specification(0) characterStrings(1) numericString(0) }
	Valor de descriptor de objeto
	"NumericString ASN.1 type"
Subcláusula	Valor de identificador de objeto
36.5	{ joint-iso-itu-t asn1(1) specification(0) characterStrings(1) printableString(1) }
	Valor de descriptor de objeto
	"PrintableString ASN.1 type"
Subcláusula	Valor de identificador de objeto
37.1	{ joint-iso-itu-t asn1(1) specification(0) modules(0) iso10646(0) }
	Valor de descriptor de objeto
	"ASN1 Character Module"

Anexo C

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.

C.1 Ejemplo de un registro de personal

La utilización de ASN.1 se ilustra por medio de un registro de personal hipotético simple.

C.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

C.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".

C.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" }
          }
}
```

C.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; su utilización permite diseñar una amplia gama de protocolos. Esta flexibilidad, sin embargo, puede llevar a veces a confusión, 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 41 a 43 no se tratan aquí.

C.2.1 Booleano

C.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

C.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

not

MaritalStatus ::= BOOLEAN

C.2.2 Entero

C.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

C.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

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 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
```

C.2.3 Enumerado

C.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
```

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.

C.2.3.2 Se utilizará un tipo enumerado extensible para modelar los valores de una variable que tiene en ese momento dos estados, pero que puede 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
```

C.2.4 Real

C.2.4.1 Se utilizará un tipo real para modelar un número aproximado.

EJEMPLO

```
AngleInRadians ::= REAL
```

```
pi REAL ::= {mantissa 3141592653589793238462643383279, base 10, exponent -30}
```

C.2.4.2 Los diseñadores de aplicaciones quizás deseen asegurar 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}
```

C.2.5 Cadena de bits

C.2.5.1 Se utilizará un tipo de cadena de bits para modelar datos binarios cuyo formato y longitud no están especificados, o lo están en alguna otra parte, y cuya longitud en bits no es necesariamente un múltiplo de ocho.

EJEMPLO

```
G3FacsimilePage ::= BIT STRING
-- a sequence of bits conforming to Recommendation T.4.

image G3FacsimilePage ::= '100110100100001110110'B

trailer BIT STRING ::= '0123456789ABCDEF'H

body1 G3FacsimilePage ::= '1101'B

body2 G3FacsimilePage ::= '1101000'B
```

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).

C.2.5.2 Se utilizará un tipo cadena de bits con una restricción 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.
```

Obsérvese que "map1" y "map2" son el mismo valor abstracto, porque los cuatro bits finales de "map2" no son significativos.

C.2.5.3 Se utilizará un tipo cadena de bits para modelar los valores de un **bit map (mapa de bits)**, una 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 – violates size constraint.
```

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.

C.2.5.4 Se utilizará un tipo cadena de bits para modelar los valores de un **bit map**, una 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.

C.2.5.5 Se utilizará un tipo cadena de bits con bits denominados 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

Obsérvese que "billClinton" y "hillaryClinton" tienen los mismos valores abstractos.

C.2.6 Cadena de octetos

C.2.6.1 Se utilizará un tipo cadena de octetos para modelar datos binarios cuyo formato y longitud no están especificados, o están especificados en alguna otra parte, y cuya longitud en bits es un múltiplo de ocho.

EJEMPLO

G4FacsimileImage ::= OCTET STRING

-- *a sequence of octets conforming to*

-- *Recommendations T.5 and T.6.*

image G4FacsimilePage ::= '3FE2EBAD471005'H

C.2.6.2 Se utilizará un tipo cadena de caracteres con preferencia a un tipo cadena de octetos cuando se disponga de uno apropiado.

EJEMPLO

Surname ::= PrintableString

president Surname ::= "Clinton"

C.2.7 UniversalString y BMPString

Se utilizará el tipo BMPString para modelar cualquier cadena de información que conste únicamente de caracteres procedentes del plano multiidioma básico (BMP, *basic multilingual plane*) de ISO/CEI 10646-1, y UniversalString para modelar cualquier cadena que conste de caracteres de ISO/CEI 10646-1 no limitados al BMP.

C.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.*

C.2.7.2 Puede expandirse una colección para que sea un subconjunto seleccionado (es decir, incluir todos los caracteres en la colección LATÍN BÁSICO) utilizando la "UnionMark" (véase la cláusula 46).

EJEMPLO

KatakanaAndBasicLatin ::= UniversalString (FROM(Katakana | BasicLatin))

C.2.8 CHARACTER STRING

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. Hay que asegurarse de especificar 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 packedBCDStringAbstractSyntax and
                                -- packedBCDStringTransferSyntax defined below.
                                } )
```

```
-- object identifier value for a character abstract syntax (character set) whose alphabet
-- is the digits 0 through 9.
```

```
packedBCDStringAbstractSyntaxId OBJECT IDENTIFIER ::=
    { joint-iso-itu-t xxx(999) yyy(999) zzz(999) packedBCD(999) 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.
```

```
packedBCDStringTransferSyntaxId OBJECT IDENTIFIER ::=
    { joint-iso-itu-t xxx(999) yyy(999) zzz(999) packedBCD(999) 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.
```

NOTA – Las reglas de codificación no necesariamente codifican valores del tipo CHARACTER STRING en una forma tal que incluya siempre los valores de identificadores de objeto, si bien garantizan la preservación del valor abstracto en la codificación.

C.2.9 Nulo

Se utilizará un 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
}
```

C.2.10 Secuencia y secuencia de

C.2.10.1 Se utilizará un 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"}
```

C.2.10.2 Se utilizará un tipo secuencia (sequence) para modelar una colección de variables del mismo tipo, de número conocido y pequeño, y cuyo orden es 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"}
```

C.2.10.3 Se utilizará un tipo secuencia inextensible para modelar una colección de variables de tipos diferentes, de número conocido y pequeño, y cuyo orden es 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}
```

C.2.10.4 Se utilizará un tipo secuencia extensible para modelar una colección de variables cuyo orden es significativo, cuyo número es actualmente conocido y pequeño pero que se espera que aumente.

EJEMPLO

```
Record ::= SEQUENCE {
    userName      VisibleString,
    password      VisibleString,
    accountNumber INTEGER,
    ...,
    ...
}
-- First version of Record
```

en anticipación de:

```
Record ::= SEQUENCE {
    userName      VisibleString,
    password      VisibleString,
    accountNumber INTEGER,
    ...,
    [[
        lastLoggedIn      GeneralizedTime OPTIONAL,
        minutesLastLoggedIn INTEGER
    ]],
    ...
}
-- Second version of Record
-- Extension addition added in version 2
```

y aún posteriormente:

```
Record ::= SEQUENCE {
    userName      VisibleString,
    password      VisibleString,
    accountNumber INTEGER,
    ...,
    [[
        lastLoggedIn      GeneralizedTime OPTIONAL,
        minutesLastLoggedIn INTEGER
    ]],
    ...
}
-- Third version of Record
-- Extension addition added in version 2
```

```

    ||
    certificate          Certificate,
    thumb               ThumbPrint OPTIONAL
    ||,
    ...
}

```

-- Extension addition added in version 3

C.2.11 Conjunto y conjunto de

C.2.11.1 Se utilizará un tipo conjunto para modelar una colección de variables cuyo número es conocido y pequeño y cuyo orden no es 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."}

```

C.2.11.2 Se utilizará un tipo conjunto con "OPTIONAL" para modelar una colección de variables que es un subconjunto (propio o impropio) de otra colección de variables de número conocido y razonablemente pequeño y cuyo orden no es 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 OPTIONAL
    -- defaults to that of the local organization -- ,
    countryName           [2] VisibleString OPTIONAL
    -- defaults to that of the local country -- }

```

C.2.11.3 Se utilizará un tipo conjunto extensible para modelar una colección de variables cuya constitución pueda presumiblemente cambiar en una versión del protocolo a la siguiente. A continuación se supone que AUTOMATIC TAGS se ha especificado en la definición de módulo.

EJEMPLO

```

UserName ::= SET {
    personalName          VisibleString,
    organizationName      VisibleString OPTIONAL,
    countryName           VisibleString OPTIONAL,
    ...,
    ...
}
user UserName ::= { personalName "Jonas Maruba" }

```

-- First version of UserName

en anticipación de:

```

UserName ::= SET {
    personalName          VisibleString,
    organizationName      VisibleString OPTIONAL,
    countryName           VisibleString OPTIONAL,
    ...,
    ||
    internetEmailAddress VisibleString,
    faxNumber             VisibleString OPTIONAL
    ||,
    ...
}

```

-- Second version of UserName

-- Extension addition added in version 2

```

user UserName ::= {
    personalName          "Jonas Maruba",
    internetEmailAddress  "jonas@meteor.ngo.com"
}

```

y aún posteriormente:

```

UserName ::= SET {
    personalName          VisibleString,
    organizationName     VisibleString OPTIONAL,
    countryName          VisibleString OPTIONAL,
    ...,
    [[
        internetEmailAddress VisibleString,
        faxNumber            VisibleString OPTIONAL
    ]],
    phoneNumber          VisibleString OPTIONAL,
    ...
}
-- Third version of UserName

-- Extension addition added in version 2

-- Extension addition added in version 3

user UserName ::= {
    personalName          "Jonas Maruba",
    internetEmailAddress  "jonas@meteor.ngo.com"
}

```

C.2.11.4 Se utilizará un tipo conjunto de para modelar una colección de variables cuyos tipos son los mismos y cuyo orden no es significativo.

EJEMPLO

```

Keywords ::= SET OF VisibleString -- in arbitrary order

someASN1Keywords Keywords ::= {"INTEGER", "BOOLEAN", "REAL"}

```

C.2.12 Rotulado

Antes de la introducción de la construcción AUTOMATIC TAGS, las especificaciones ASN.1 contenían rótulos frecuentemente. En las subcláusulas que siguen se describe la manera según la cual se aplicaba la rotulación típicamente. Con la introducción de AUTOMATIC TAGS, las nuevas especificaciones ASN.1 de usuario no necesitan hacer uso de la notación de rótulo, si bien aquellas que modifiquen notación antigua quizá tengan que ocuparse de los rótulos.

C.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 tipos útiles normalizados internacionalmente. No deberá utilizarse en otro sitio.

C.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 más 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}

```

C.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 --}

```

C.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
```

C.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 compacidad puede ser menos importante que, por ejemplo, la aptitud para efectuar una verificación de tipos fuerte. En el último caso, se utilizará 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 --}

```

C.2.12.6 El criterio respecto a la utilización de rótulos en las nuevas especificaciones ASN.1 de usuario 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.

C.2.13 Elección

C.2.13.1 Se utilizará una CHOICE para modelar una variable seleccionada de una colección de variables cuyo número es 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

```

C.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 {
    relativeName  VisibleString,
    absoluteName  VisibleString,
    ..., ...
}
fileId1 FileIdentifier ::= relativeName : "MarchProgressReport.doc"
```

-- First version of FileIdentifier

en anticipación de:

```
FileIdentifier ::= CHOICE {
    relativeName  VisibleString,
    absoluteName  VisibleString,
    ...,
    serialNumber  INTEGER,
    ...
}
fileId1 FileIdentifier ::= relativeName : "MarchProgressReport.doc"
fileId2 FileIdentifier ::= serialNumber : 214
```

-- Second version of FileIdentifier
-- Extension addition added in version 2

y aún posteriormente:

```
FileIdentifier ::= CHOICE {
    relativeName  VisibleString,
    absoluteName  VisibleString,
    ...,
    serialNumber  INTEGER,
    [
        vendorSpecific  VendorExt,
        unidentified  NULL
    ],
    ...
}
fileId1 FileIdentifier ::= relativeName : "MarchProgressReport.doc"
fileId2 FileIdentifier ::= serialNumber : 214
fileId3 FileIdentifier ::= unidentified : NULL
```

-- Third version of FileIdentifier
-- Extension addition added in version 2
-- Extension addition added in version 3

C.2.13.3 Se utilizará una CHOICE extensible de un solo tipo donde se haya previsto la posibilidad de permitir más de un tipo en el futuro.

EJEMPLO

```
Greeting ::= CHOICE {
    postCard  VisibleString,
    ..., ...
}

```

-- First version of Greeting

en anticipación de:

```
Greeting ::= CHOICE {
    postCard  VisibleString,
    ...,
    [
        audio  Audio,
        video  Video
    ],
    ...
}

```

-- Second version of Greeting
-- Extension addition added in version 2

C.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
```

C.2.14 Tipo selección

C.2.14.1 Se utilizará un tipo selección para modelar una variable cuyo tipo sea el de algunas alternativas determinadas de una CHOICE definida anteriormente.

C.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" }
```

C.2.15 Tipo campo de clase de objeto

C.2.15.1 Se utilizará un tipo campo de clase de objeto, es decir, un tipo abierto, 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.

EJEMPLO

```
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 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.

C.2.16 pdv incrustado

C.2.16.1 Se utilizará un tipo pdv incrustado para modelar una variable cuyo tipo no se ha especificado o lo ha sido en otro sitio sin limitación en la notación utilizada para especificar el tipo.

EJEMPLO

FileContents ::= EMBEDDED PDV

DocumentList ::= SEQUENCE OF EMBEDDED PDV

C.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.

C.2.18 Ejemplar de

C.2.18.1 Se utilizará un ejemplar de para especificar un tipo que contenga un campo de identificador de objeto y un tipo abierto cuyo valor es un tipo determinado por el identificador de objeto. El tipo ejemplar de se limita a llevar un valor de la clase TYPE-IDENTIFIER (véanse los anexos A y C a la Rec. UIT-T X.681 | ISO/CEI 8824-2).

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**

}

Get-Invoke entonces 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 utilizando 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 a esa Recomendación | Norma Internacional, cómo utilizar un conjunto de objetos de información para constreñir un tipo ejemplar de. Se señala que la codificación de la INSTANCE OF ACCESS-CONTROL-CLASS es la misma que para un valor EXTERNAL que tiene sólo un identificador de objeto y un valor de datos.

C.3 Identificación de sintaxis abstractas

C.3.1 El uso del servicio de presentación definido en la Rec. UIT-T X.216 | ISO/CEI 8822 requiere la especificación de valores denominados valores de datos de presentación y la agrupación de dichos valores de datos de presentación en conjuntos que se denominan sintaxis abstractas. Cada uno de estos conjuntos recibe un nombre de sintaxis abstracta como identificador de objeto tipo ASN.1.

C.3.2 Puede utilizarse la notación ASN.1 como un instrumento general en la especificación de los valores de datos de presentación y su agrupación en sintaxis abstractas denominadas.

C.3.3 En el uso más sencillo, existe un tipo ASN.1 único tal que cada valor de datos de presentación en la sintaxis abstracta denominada es un valor de dicho tipo ASN.1. Este tipo normalmente será un tipo elección y cada valor de datos de presentación será un tipo alternativo de este tipo elección. En este caso se recomienda que se utilice la notación modular ASN.1 para contener dicho tipo elección como el primer tipo definido, seguido por la definición de aquellos tipos (no universales) referenciados directa o indirectamente por este tipo elección.

NOTA – No se tiene el propósito de excluir las referencias a los tipos definidos en otros módulos.

C.3.4 Se recomienda que la asignación de un identificador de objeto y un descriptor de objeto a una sintaxis abstracta se haga utilizando la clase de objeto de información útil ABSTRACT-SINTAX definida en la Rec. UIT-T X.681 | ISO/CEI 8824-2. Se recomienda también que todas las utilizaciones de la ABSTRACT-SYNTAX se agrupen en un solo módulo " raíz" que identifique todas las sintaxis abstractas utilizadas por una norma de aplicación.

C.3.5 A continuación se presenta un ejemplo de texto que podría aparecer en una aplicación normal.

EJEMPLO

```
ISOxxxx-yyyy {iso standard xxxx asn1-modules(...)} DEFINITIONS ::=
BEGIN
    EXPORTS YYYYY-PDU;

    YYYYY-PDU ::= CHOICE {
        connect-pdu ..... ,
        data-pdu CHOICE {
            ..... ,
            .....
        },
        .....
    }
    .....
END

ISOxxxx-yyyy-Abstract-Syntax-Module {iso standard xxxx asn1-modules(...)} DEFINITIONS ::=
BEGIN
    IMPORTS YYYYY-PDU FROM ISOxxxx-yyyy {iso standard xxxx asn1-modules(...)} yyy-pdu(...);

    -- This Recommendation | International Standard defines the following abstract syntax:

    YYYYY-Abstract-Syntax ABSTRACT-SYNTAX ::=
        { YYYYY-PDU IDENTIFIED BY yyy-abstract-syntax-object-id }

    yyy-abstract-syntax-object-id OBJECT IDENTIFIER ::= {iso standard yyyy(xxxx) abstract-syntax(...)}

    -- The corresponding object descriptor is:

    yyy-abstract-syntax-descriptor ObjectDescriptor ::= "....."

    -- 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,
    -- ISOxxx-yyyy-Abstract-Syntax.

END
```

C.3.6 Para garantizar el interfuncionamiento, la norma puede además hacer obligatoria la admisión de la sintaxis de transferencia obtenida aplicando las reglas de codificación mencionadas en su módulo de sintaxis abstracta.

C.4 Subtipos

C.4.1 Se utilizarán subtipos para limitar los valores de un tipo existente que se admitan 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)

C.4.2 Se definirá el uso de una restricción de subtipo extensible para modelar un tipo INTEGER cuyo conjunto de valores permitidos es pequeño y bien definido, pero que se espera que 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 restricción de subtipo (es decir, valores que aparecen antes del "...") y una codificación menos optimizada para valores adición de extensión de restricción de subtipo (es decir, valores que aparecen después del "..."), mientras que algunas otras reglas de codificación (por ejemplo, BER) para restricciones de subtipo no afectan a la codificación.

C.4.3 Cuando dos o más tipos relacionados tengan en común aspectos significativos (dícese "comunalidad significativa"), 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 la comunalidad, a la vez que alienta (aunque no obliga) a que continúe esta práctica conforme evolucionan los tipos. Así, facilita el uso de los enfoques 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 posibilidad de una estrategia de evolución.

C.4.4 Se utilizará la subtipificación para definir parcialmente un valor, por ejemplo, una unidad de datos de protocolo (PDU) que será sometida a una prueba de conformidad, cuando la prueba afecte únicamente a algunos de los componentes de la PDU.

EJEMPLO

Dado:

PDU ::= SET
 {alpha INTEGER,
 beta IA5String OPTIONAL,
 gamma SEQUENCE OF Parameter,
 delta BOOLEAN}

cuando se componga, entonces, 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 AI5 (AI5String), beta, con una longitud de 5 ó 12 caracteres, escríbase:

```
FurtherTestPDU ::= TestPDU (WITH COMPONENTS { ... , beta (SIZE (5|12)) PRESENT } )
```

C.4.5 Si un tipo de datos para 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)))
```

C.4.6 Si un tipo de datos para 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 48.8.9.2).

```
X ::= Z (WITH COMPONENTS { a PRESENT })
```

-- only 'a' can be present (see 48.8.9.2).

```
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.

C.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 D

Suplemento didáctico sobre cadenas de caracteres ASN.1

(Este anexo no es parte integrante de esta Recomendación | Norma Internacional)

D.1 Soporte de cadenas de caracteres en ASN.1

D.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, 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 que llevan un nombre (caracteres denominados).

NOTA 1 – La utilización de estos tipos no constreñidos conduce a una violación de los requisitos de conformidad para el intercambio de información especificados en ISO/CEI 10646-1, ya que no se ha especificado ningún subconjunto adoptado.

NOTA 2 – No obstante lo antes expuesto, la utilización de este tipo con una restricción de subtipo simple que utiliza un parámetro de la sintaxis abstracta (restringido a un subtipo definido de UniversalString) puede proporcionar un medio potente y flexible para el tratamiento de caracteres, basándose en perfiles para determinar el valor del parámetro a fin de satisfacer las necesidades de comunidades particulares de interés. Sin embargo, por lo general, debe preferirse la utilización de CHARACTER STRING en Recomendaciones | Normas Internacionales (véase más adelante).

- 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 conjunto de caracteres que habrá de utilizarse (o anuncio del conjunto 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; (unos perfiles pueden imponer requisitos o restricciones a los conjuntos de caracteres – las sintaxis abstractas de caracteres – que habrán de utilizarse).

D.2 Los tipos UniversalString, UTF8String y BMPString

D.2.1 Los tipos UniversalString y UTF8String llevan cualquier carácter de ISO/CEI 10646-1. El conjunto de caracteres de 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 normalizadas en ISO/CEI 10646-1, anexo A.

D.2.2 El tipo BMPString lleva cualquier carácter del plan multilingüe básico de ISO/CEI 10646-1 (los primeros 62K caracteres). Normalmente se recurre a un subconjunto del plan multilingüe básico formado por una combinación de colecciones de caracteres normalizadas en ISO/CEI 10646-1, anexo A.

D.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 37). El mecanismo "subtype constraint" (restricción de subtipo) permite definir nuevos subtipos de UniversalString que son combinaciones de subtipos existentes.

D.2.4 Ejemplos de referencias de tipo definidas en ASN1-CHARACTER-MODULE y sus correspondientes de colecciones de ISO/CEI 10646-1 son:

BasicLatin	LATÍN BÁSICO
Latin-1Supplement	SUPLEMENTO DE LATÍN 1
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

D.2.5 ISO/CEI 10646-1 especifica tres "niveles de implementación" y exige que todas las utilizaciones 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* en el 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 los caracteres combinantes y normalmente, existe una correspondencia biunívoca entre caracteres abstractos (referencias a células) de las cadenas de caracteres ASN.1 y caracteres impresos de una reproducción física 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 cuya utilización está prohibida.

En el nivel 3 de implementación no hay restricciones a la utilización de los caracteres combinantes.

D.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)))
```

D.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 (digamos, 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.

D.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.

ISO/CEI 8824-1 : 1998 (S)

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.
```

El PICS 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 subtipificados) en todas las normas en que se tuvieran que incluir cadenas de ISO/CEI 10646-1.

EJEMPLO DE PICS

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  
}
```

D.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 constreñir 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 restricción 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  
-- receive 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 }
```

D.6 El tipo CHARACTER STRING

D.6.1 El tipo CHARACTER STRING da completa flexibilidad en la elección del conjunto de caracteres y del método de codificación. Cuando una conexión única proporcione transferencia de datos de extremo a extremo (es decir, cuando no se trate una aplicación relevadora), la negociación de los conjuntos de caracteres que han de utilizarse y de la codificación puede efectuarse como parte de la definición de los contextos de presentación para sintaxis abstractas de caracteres.

D.6.2 Es importante comprender que 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 registración de esas sintaxis, y la negociación de un contexto de presentación se efectúan de manera normal.

D.6.3 La codificación de CHARACTER STRING permite también el anuncio de la sintaxis abstracta y de la sintaxis de transferencia utilizadas, sin negociación, en el caso de entornos en que esto sea apropiado.

NOTA 1 – En lo que respecta a la utilización de negociación de presentación para estos campos, los diseñadores de aplicaciones pueden prohibirla, exigirla, o dejarla a la voluntad del emisor.

NOTA 2 – Cuando se emplee anuncio en lugar de negociación, el diseñador de la aplicación debe considerar dos cosas: en primer lugar, la manera en que el emisor puede determinar qué sintaxis abstractas de caracteres (y qué sintaxis de transferencia) podrían ser aceptables por el receptor (por ejemplo, mediante la utilización del servicio de directorio o como resultado del establecimiento de perfiles) y, en segundo lugar, las acciones que deberá ejecutar el receptor cuando reciba un valor CHARACTER STRING de una sintaxis abstracta de caracteres no soportada por él.

D.6.4 Si se utiliza negociación, el diseñador de capa de aplicación puede controlar dicha negociación especificando cuándo deberán establecerse esos contextos de presentación, y el parámetro de datos de usuario de las primitivas P-ALTERACIÓN DE CONTEXTO, o, simplemente suponer que algún perfil ha determinado qué sintaxis de caracteres habrá de utilizarse, y establecer un contexto de presentación para la misma en el momento de P-CONEXIÓN.

D.6.5 Las facilidades de gestión de contexto del servicio de presentación permiten a un iniciador (en una P-CONEXIÓN o dentro de una conexión establecida utilizando P-ALTERACIÓN DE CONTEXTO) proponer una lista de nuevas sintaxis abstractas (que pueden incluir sintaxis abstractas de caracteres), o retirar del uso sintaxis abstractas, de modo que el respondedor seleccione dentro de esa lista.

D.6.6 El iniciador puede expresar preferencia por medio del orden de la sintaxis abstracta en la lista, o de la utilización del parámetro datos de usuario, que puede ser empleado por el diseñador de aplicación para aclarar la finalidad de proponer la utilización de la nueva sintaxis abstracta. Podría indicar, por ejemplo, que todas las sintaxis abstractas (de caracteres) se están proponiendo para uso con una sola finalidad determinada, o que se tiene el propósito de permitir la selección de una sola sintaxis abstracta (de caracteres) para uso con varios fines.

D.6.7 Se han definido sintaxis abstractas de caracteres (y las correspondientes sintaxis de transferencia de caracteres) en Recomendaciones UIT-T y Normas Internacionales, y pueden ser definidas sintaxis abstractas de caracteres adicionales (y/o sintaxis de transferencia de caracteres) por cualquier organización capaz de atribuir identificadores de objetos.

D.6.8 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 E

Características reemplazadas

(Este anexo no es parte integrante de esta Recomendación | Norma Internacional)

Un número de prestaciones que fueron incluidas en ediciones anteriores de esta Recomendación | Norma Internacional (a saber, la Rec. X.208 del CCITT | ISO/CEI 8824) han sido reemplazadas y no forman parte de ASN.1. No obstante pueden encontrarse en algunos módulos ASN.1 existentes. Este anexo describe estas prestaciones, así como la forma de obtener sus capacidades utilizando las que han venido a reemplazarlas.

E.1 Utilización de identificadores ahora obligatoria

La notación para un NamedType y un NamedValue que inicialmente era:

NamedType ::= identifier Type | Type | SelectionType
NamedValue ::= identifier Value | Value

pero ha sido reemplazada por:

NamedType ::= identifier Type
NamedValue ::= identifier Value

porque la primera podría dar lugar a una gramática ambigua.

Pueden añadirse identificadores a los "NamedType" de las especificaciones ASN.1 antiguas sin afectar a la codificación de tipo (aunque serán necesarios cambios en la ASN.1 para cualquier utilización de notación de valor relacionado). Esa modificación deberá introducirse en el marco de un informe de faltas o como parte de una nueva revisión de la Recomendación | Norma Internacional que se modifica.

E.2 El valor de elección

La notación de valor para el tipo elección era inicialmente:

ChoiceValue ::= NamedValue
NamedValue ::= identifier Value | Value

pero ha sido reemplazada por:

ChoiceValue ::= identifier ":" Value

porque la primera podría dar lugar a una gramática ambigua.

E.3 El tipo cualquiera

El tipo cualquiera (any) fue definido en ediciones anteriores de la presente Recomendación | Norma Internacional.

El uso normal del tipo cualquiera (any) tenía por finalidad dejar un "hueco", en la especificación, que sería llenado por alguna otra especificación. La notación era "AnyType", permitida como una alternativa para "Type", y se especificaba así:

AnyType ::= ANY | ANY DEFINED BY identifier

Se recomendaba insistentemente utilizar la segunda alternativa de la notación. En esta alternativa, que sólo estaba permitida donde el tipo cualquiera era uno de los tipos componentes de un tipo conjunto o secuencia, algún otro componente del conjunto o de la secuencia (aquel cuyo "identifier" hubiese sido referenciado) indicaría por su valor entero o de identificador de objeto (o una opción de estos) el tipo que gobernaría en efecto al componente any. La correspondencia entre esos valores y tipos ASN.1 particulares podría visualizarse como una especie de "tabla" que formaría parte de la sintaxis abstracta. En ausencia del "DEFINED BY identifier" (la primera alternativa notacional), no había en la notación ninguna indicación sobre la forma en que podría determinarse el tipo. Esto conducía frecuentemente a especificaciones en las que el "hueco" continuaba existiendo incluso en la etapa en que se esperaban implementaciones.

El tipo cualquiera está ahora reemplazado por la aptitud para especificar clases de objeto de información y, seguidamente, referir a los campos de objetos de información desde el interior de definiciones de tipo (véase la Rec. UIT-T X.681 | ISO/CEI 8824-2). Puesto que pueden definirse campos para permitir un tipo ASN.1 arbitrario, se proporciona la aptitud básica para dejar "huecos". Sin embargo, la nueva prestación permite también la especificación de una "constricción por tabla", en la cual se indica explícitamente que un determinado "conjunto de objetos de información" (un conjunto de objetos de información de la clase apropiada) constriñe el tipo. Esta última capacidad comprende la ofrecida por "**ANY DEFINED BY** identifier".

Se proporcionan asimismo algunos usos predefinidos de las nuevas capacidades (véase la Rec. UIT-T X.681 | ISO/CEI 8824-2), que corresponden a diversos patrones de utilización del tipo cualquiera que suelen presentarse con frecuencia. Por ejemplo, una secuencia que contiene un identificador de objeto y un any, que solía utilizarse anteriormente para transportar algún valor arbitrario junto con una indicación de su tipo, puede ahora describirse como:

INSTANCE OF MUMBLE

donde **MUMBLE** se define como una clase de objeto de información (no un tipo ASN.1):

MUMBLE ::= TYPE-IDENTIFIER

Esta notación hace que "INSTANCE OF MUMBLE" sea sustituida por un identificador de objeto para un objeto de clase **MUMBLE**, junto con el tipo identificado por el identificador de objeto. Véase un ejemplo en C.2.18.

Emparejamientos particulares de identificador de objeto y tipo se definen como objetos de información de clase **MUMBLE**, y, si es necesario, conjuntos particulares de éstos pueden también definirse y utilizarse para constreñir la construcción **INSTANCE OF** de modo que en el conjunto sólo puedan aparecer esos objetos.

La capacidad macro se utilizaba a menudo como una manera semiformal de definir tablas de objetos de información para gobernar un uso asociado de un tipo cualquiera, y ha sido también reemplazada por las nuevas capacidades.

E.4 La capacidad macro

La capacidad macro permitía al usuario de ASN.1 ampliar la notación definiendo macros.

La capacidad macro se ha utilizado sobre todo para definir una notación para la especificación de objetos de información. Esta capacidad ha sido ahora incluida en ASN.1 directamente (véase la Rec. UIT-T X.681 | ISO/CEI 8824-2), por lo que no es necesario recurrir a la solución general de una notación definida por el usuario (con los peligros consiguientes).

Aparte de esto, la única utilización que pueden tener los macros parece ser la de definir expresiones a las que deben suministrarse algunos parámetros antes de que se conviertan en tipos ASN.1 completamente definidos. Esto se consigue ahora mediante la capacidad más general de parametrización (véase la Rec. UIT-T X.683 | ISO/CEI 8824-4).

Anexo F

Anexo explicativo del modelo ASN.1 de extensión de tipo

(Este anexo no es parte integrante de esta Recomendación | Norma Internacional)

F.1 Visión general

F.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**.

F.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.

F.1.3 Las definiciones de tipos ASN.1 de esta serie se dice que están **relacionadas por extensión** (véase 3.8.32 para una definición más precisa de "relacionada por extensión") y se precisan reglas de codificación para codificar tipos relacionados por extensión de manera que si dos sistemas están utilizando dos tipos diferentes que están relacionados por extensión, las transmisiones entre los dos sistemas transfieran con éxito el contenido de la información de las partes de los tipos relacionados por extensión que son comunes a ambos sistemas. También se requiere que aquellas partes que no son comunes a ambos sistemas puedan ser delimitadas y retransmitidas (quizá 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.

F.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 por extensión (lo que puede necesitar más bits en la línea), dichos tipos (incluido el tipo raíz de extensión) necesitan estar marcados sintácticamente. La marca es una elipsis (...) que se denomina **marcador de extensión**.

EJEMPLO

Extension root type	1st extension	2nd extension	3rd extension
A ::= SEQUENCE { a INTEGER, ... }	A ::= SEQUENCE { a INTEGER, ..., b BOOLEAN, c INTEGER }	A ::= SEQUENCE { a INTEGER, ..., b BOOLEAN, c INTEGER, d SEQUENCE { e INTEGER, ..., ..., f IA5String } }	A ::= SEQUENCE { a INTEGER, ..., b BOOLEAN, c INTEGER, d SEQUENCE { e INTEGER, ..., ..., g BOOLEAN OPTIONAL, h BMPString, ..., f IA5String } }

F.1.5 Todas las adiciones de extensión están insertadas entre pares de marcadores de extensión. Solamente se permite un marcador de extensión si en el tipo raíz de extensión aparece como el último ítem 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.

F.1.6 Un tipo que tiene un marcador de extensión puede anidarse dentro de un tipo que no tenga 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.

F.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 1ª extensión define un grupo adición de extensión en el cual b y c deben estar ambos presentes o estar ambos ausentes en un valor de tipo "A". La segunda extensión define un tipo componente único, d, el cual puede estar ausente en un valor de tipo "A". La tercera extensión define un grupo adición de extensión en el cual h debe estar presente en un valor de tipo "A" siempre que el grupo adición de extensión añadido después se encuentre presente en el valor.

EJEMPLO

Extension root type	1 st extension	2 nd extension	3 rd extension
A ::= SEQUENCE { a INTEGER, ... }	A ::= SEQUENCE { a INTEGER, ... b BOOLEAN, c INTEGER }	A ::= SEQUENCE { a INTEGER, ... b BOOLEAN, c INTEGER , d SEQUENCE { e INTEGER, f IA5String } }	A ::= SEQUENCE { a INTEGER, ... b BOOLEAN, c INTEGER , d SEQUENCE { e INTEGER, ... g BOOLEAN OPTIONAL, h BMPString , ... f IA5String }

F.1.8 Aunque lo normal será añadir las adiciones de extensión en el tiempo, el modelo ASN.1 y la especificación subyacentes no consideran el tiempo. Dos tipos están relacionados por 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.

F.1.9 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**. Podemos posteriormente añadir adiciones de extensión al tipo sin modificar los bits en la línea para valores anteriores, y manejando de una manera definida los valores extendidos por sistemas anteriores. Es sin embargo 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) **cambiará** generalmente los bits en la línea e impedirá el interfuncionamiento. Dicho cambio requiere normalmente un cambio en el número de versión en todos los protocolos afectados.

F.1.10 El cuadro F.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 hacer múltiples adiciones de extensión en sucesión o juntas).

Cuadro F.1 – Adiciones de extensión

Tipo raíz de extensión	Naturaleza de la adición de extensión
ENUMERATED	Adición de una enumeración ulterior única al final de las "AdditionalEnumeration" con un valor de enumeración mayor que el de cualquier enumeración ya añadida.
SEQUENCE y SET	Adición de un tipo o grupo adición de extensión único al final de la "ExtensionAdditionList". No es obligatorio marcar los "ComponentType" que son adiciones de extensión (no contenidas en un grupo adición de extensión) con OPTIONAL o DEFAULT, aunque normalmente se hace.
CHOICE	Adición de un "NamedType" único al final de la "ExtensionAdditionAlternativesList".
Notación constricción	Adición de una única "AdditionalElementSetSpec" a la notación "ElementSetSpecs".

F.2 Efectos sobre el número de la versión, etc.

F.2.1 Cuando se hace una nueva publicación de una especificación ASN.1 en la que las definiciones de tipo han cambiado con respecto a las definiciones de tipos relacionados con la extensión, entonces, para todos efectos, estos cambios, por sí mismos, no requieren un cambio en el identificador de objeto del módulo del número de versión del protocolo.

F.2.2 Puede darse el caso de que, por otros motivos, tales cambios pudieran ir acompañados de cambios en el número de la versión, pero esto no es obligatorio.

F.2.3 Por el contrario, la adición de un marcador de extensión a un tipo que antes no lo tenía, o la adición de componentes a un tipo secuencia o conjunto (por ejemplo) sin ningún marcador de extensión, crea un nuevo tipo que **no** está relacionado con el tipo antiguo en lo que respecta a la extensión, el módulo que lo contiene debe recibir un nuevo identificador de objeto y un nuevo número de versión será apropiado en el protocolo asociado.

F.3 Requisitos sobre reglas de codificación

F.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.

F.3.2 Un conjunto de reglas de codificación bien formado para una sintaxis abstracta relacionada por extensión cumple los requisitos adicionales indicados en F.3.3 a F.3.5.

NOTA – Todas las reglas de codificación ASN.1 cumplen estos requisitos.

F.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 reconocerá la posibilidad de que el emisor y el receptor estén utilizando sintaxis abstractas que no son idénticas pero que están relacionadas por extensión.

F.3.4 En este caso, las reglas de codificación asegurarán que cuando 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.

F.3.5 Las reglas de codificación asegurarán que, cuando el emisor tenga una especificación de tipo que sea anterior, en la serie de extensión, a la del receptor, será posible la transferencia de valores de este tipo al receptor.

Anexo G

Sumario de la notación ASN.1

(Este anexo no es parte integrante de esta Recomendación | Norma Internacional)

Los siguientes ítems se definen en la cláusula 11:

typereference	BEGIN	MAX
identifier	BIT	MIN
valuereference	BMPString	MINUS-INFINITY
modulereference	BOOLEAN	NULL
comment	BY	NumericString
empty	CHARACTER	OBJECT
number	CHOICE	ObjectDescriptor
bstring	CLASS	OCTET
hstring	COMPONENT	OF
cstring	COMPONENTS	OPTIONAL
"::="	CONSTRAINED	PDV
[[DEFAULT	PLUS-INFINITY
]]	DEFINITIONS	PRESENT
".."	EMBEDDED	PrintableString
"..."	END	PRIVATE
"{"	ENUMERATED	REAL
"}"	EXCEPT	SEQUENCE
"<"	EXPLICIT	SET
","	EXPORTS	SIZE
":"	EXTENSIBILITY	STRING
"("	EXTERNAL	SYNTAX
")"	FALSE	T61String
"["	FROM	TAGS
"]"	GeneralizedTime	TeletexString
"_"	GeneralString	TRUE
":"	GraphicString	TYPE-IDENTIFIER
","	IA5String	UNION
"@"	IDENTIFIER	UNIQUE
" "	IMPLICIT	UNIVERSAL
"!"	IMPLIED	UniversalString
"^"	IMPORTS	UTCTime
ABSENT	INCLUDES	UTF8String
ABSTRACT-SYNTAX	INSTANCE	VideotexString
ALL	INTEGER	VisibleString
APPLICATION	INTERSECTION	WITH
AUTOMATIC	ISO646String	

En esta Recomendación | Norma Internacional se utilizan las siguientes producciones con los ítems 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 ";" |
    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 | Symbol "," SymbolList

Symbol ::= Reference | ParameterizedReference

Reference ::=
    typerference |
    valuereference |
    objectclassreference |
    objectreference |
    objectsetreference

```

AssignmentList ::= Assignment | AssignmentList Assignment

Assignment ::=

TypeAssignment |
ValueAssignment |
ValueSetTypeAssignment |
ObjectClassAssignment |
ObjectAssignment |
ObjectSetAssignment |
ParameterizedAssignment

Externaltypereference ::=

modulereference
"."
typereference

Externalvaluereference ::=

modulereference
"."
valuereference

DefinedType ::=

Externaltypereference |
typereference |
ParameterizedType |
ParameterizedValueSetType

DefinedValue ::=

Externalvaluereference |
valuereference |
ParameterizedValue

AbsoluteReference ::= "@" GlobalModuleReference

."
ItemSpec

ItemSpec ::=

typereference |
ItemId "." ComponentId

ItemId ::= ItemSpec

ComponentId ::=

identifier | number | "*"

TypeAssignment ::= typereference

::="

Type

ValueAssignment ::= valuereference

Type
::="

Value

ValueSetTypeAssignment ::= typereference

Type
::="

ValueSet

ValueSet ::= "{" ElementSetSpecs "}"

Type ::= BuiltinType | ReferencedType | ConstrainedType

BuiltinType ::=

BitStringType |
BooleanType |
CharacterStringType |
ChoiceType |
EmbeddedPDVType |
EnumeratedType |
ExternalType |
InstanceOfType |
IntegerType |
NullType |

ISO/CEI 8824-1 : 1998 (S)

ObjectClassFieldType |
ObjectIdentifierType |
OctetStringType |
RealType |
SequenceType |
SequenceOfType |
SetType |
SetOfType |
TaggedType

NamedType ::= identifier Type

ReferencedType ::=

DefinedType |
UsefulType |
SelectionType |
TypeFromObject |
ValueSetFromObjects

Value ::= BuiltinValue | ReferencedValue

BuiltinValue ::=

BitStringValue |
BooleanValue |
CharacterStringValue |
ChoiceValue |
EmbeddedPDVValue |
EnumeratedValue |
ExternalValue |
InstanceOfValue |
IntegerValue |
NullValue |
ObjectClassFieldValue |
ObjectIdentifierValue |
OctetStringValue |
RealValue |
SequenceValue |
SequenceOfValue |
SetValue |
SetOfValue |
TaggedValue

ReferencedValue ::=

DefinedValue |
ValueFromObject

NamedValue ::= identifier Value

BooleanType ::=BOOLEAN

BooleanValue::= TRUE | FALSE

IntegerType ::=

INTEGER |
INTEGER "{" NamedNumberList "}"

NamedNumberList ::=

NamedNumber |
NamedNumberList "," NamedNumber

NamedNumber ::=

identifier "(" SignedNumber ")" |
identifier "(" DefinedValue ")"

SignedNumber ::= number | "-" number

IntegerValue ::= SignedNumber | identifier

EnumeratedType ::=

ENUMERATED "{" Enumerations "}"

Enumerations ::= RootEnumeration |

RootEnumeration "," "..." |
RootEnumeration "," "..." "," AdditionalEnumeration

RootEnumeration ::= Enumeration
AdditionalEnumeration ::= Enumeration
Enumeration ::=
 EnumerationItem | EnumerationItem "," Enumeration
EnumerationItem ::=
 identifier | NamedNumber
EnumeratedValue ::=
 identifier
RealType ::= REAL
RealValue ::=
 NumericRealValue | SpecialRealValue
NumericRealValue ::= 0 |
 SequenceValue -- *Value of the associated sequence type*
SpecialRealValue ::=
 PLUS-INFINITY | MINUS-INFINITY
BitStringType ::= BIT STRING | BIT STRING "{" NamedBitList "}"
NamedBitList ::= NamedBit | NamedBitList "," NamedBit
NamedBit ::= identifier "(" number ")" |
 identifier "(" DefinedValue ")"
BitStringValue ::= bstring | hstring | "{" IdentifierList "}" | "{" "}"
IdentifierList ::= identifier | IdentifierList "," identifier
OctetStringType ::= OCTET STRING
OctetStringValue ::= bstring | hstring
NullType ::= NULL
NullValue ::= NULL

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
RootComponentTypeList ::= ComponentTypeList
ExtensionEndMarker ::= "," "..."
ExtensionAdditions ::= "," ExtensionAdditionList | empty
ExtensionAdditionList ::= ExtensionAddition |
 ExtensionAdditionList "," ExtensionAddition
ExtensionAddition ::= ComponentType | ExtensionAdditionGroup
ExtensionAdditionGroup ::= "[|" ComponentTypeList "|]"
ComponentTypeList ::= ComponentType |
 ComponentTypeList "," ComponentType

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

SequenceValue ::= "{" ComponentValueList "}" | "{" "}"
ComponentValueList ::= NamedValue |
 ComponentValueList "," NamedValue
SequenceOfType ::= SEQUENCE OF Type
SequenceOfValue ::= "{" ValueList "}" | "{" "}"
ValueList ::= Value | ValueList "," Value
SetType ::= SET "{" "}" |
 SET "{" ExtensionAndException OptionalExtensionMarker "}" |
 SET "{" ComponentTypeLists "}"
SetValue ::= "{" ComponentValueList "}" | "{" "}"
SetOfType ::= SET OF Type
SetOfValue ::= "{" ValueList "}" | "{" "}"
ChoiceType ::= CHOICE "{" AlternativeTypeLists "}"
AlternativeTypeLists ::=
 RootAlternativeTypeList |
 RootAlternativeTypeList " ,"
 ExtensionAndException ExtensionAdditionAlternatives OptionalExtensionMarker
RootAlternativeTypeList ::= AlternativeTypeList
ExtensionAdditionAlternatives ::= " , " ExtensionAdditionAlternativesList | empty
ExtensionAdditionAlternativesList ::= ExtensionAdditionAlternative |
 ExtensionAdditionAlternativesList " , " ExtensionAdditionAlternative
ExtensionAdditionAlternative ::= ExtensionAdditionAlternatives | NamedType
ExtensionAdditionAlternatives ::= "[" AlternativeTypeList "]"
AlternativeTypeList ::= NamedType |
 AlternativeTypeList " , " NamedType
ChoiceValue ::= identifier ":" Value
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
EmbeddedPDVType ::= EMBEDDED PDV
EmbeddedPDVValue ::= SequenceValue
ExternalType ::= EXTERNAL
ExternalValue ::= SequenceValue
ObjectIdentifierType ::= OBJECT IDENTIFIER
ObjectIdentifierValue ::= "{" ObjIdComponentList "}" |
 "{" DefinedValue ObjIdComponentList "}"
ObjIdComponentList ::= ObjIdComponent |
 ObjIdComponent ObjIdComponentList

```

ObjIdComponent ::= NameForm          |
                   NumberForm        |
                   NameAndNumberForm

NameForm      ::= identifier

NumberForm    ::= number | DefinedValue

NameAndNumberForm ::= identifier "(" NumberForm ")"

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

UnrestrictedCharacterStringType ::= CHARACTER STRING

CharacterStringValue ::= RestrictedCharacterStringValue | UnrestrictedCharacterStringValue

UnrestrictedCharacterStringValue ::= SequenceValue

UsefulType ::= typereference

```

Los siguientes tipos cadena de caracteres se definen en 36.1:

```

NumericStringVisibleString
PrintableString      ISO646String
TeletexString        IA5String
T61String             GraphicString
VideotexString       GeneralString
UniversalString       BMPString

```

Los siguientes tipos útiles se definen en las cláusulas 41 a 43:

```

GeneralizedTime
UTCTime
ObjectDescriptor

```

Las siguientes producciones se utilizan en las cláusulas 44 a 48:

```

ConstrainedType ::=
  Type Constraint |
  TypeWithConstraint

```

ISO/CEI 8824-1 : 1998 (S)

TypeWithConstraint ::=
SET Constraint OF Type |
SET SizeConstraint OF Type |
SEQUENCE Constraint OF Type |
SEQUENCE SizeConstraint OF Type

Constraint ::= "(" ConstraintSpec ExceptionSpec ")"

ConstraintSpec ::=
SubtypeConstraint |
GeneralConstraint

ExceptionSpec ::= "!" ExceptionIdentification | empty

ExceptionIdentification ::= SignedNumber |
DefinedValue |
Type ":" Value

SubtypeConstraint ::= ElementSetSpecs

ElementSetSpecs ::=
RootElementSetSpec |
RootElementSetSpec "," "..." |
"..." "," AdditionalElementSetSpec |
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

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

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie B	Medios de expresión: definiciones, símbolos, clasificación
Serie C	Estadísticas generales de telecomunicaciones
Serie D	Principios generales de tarificación
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedios
Serie I	Red digital de servicios integrados
Serie J	Transmisiones de señales radiofónicas, de televisión y de otras señales multimedios
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	RGT y mantenimiento de redes: sistemas de transmisión, circuitos telefónicos, telegrafía, facsímil y circuitos arrendados internacionales
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de transmisión telefónica, instalaciones telefónicas y redes locales
Serie Q	Conmutación y señalización
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	Redes de datos y comunicación entre sistemas abiertos
Serie Y	Infraestructura mundial de la información
Serie Z	Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación