UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T X.208

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

INTERCONEXIONES DE SISTEMAS ABIERTOS

MODELO Y NOTACIÓN

ESPECIFICACIÓN DE LA NOTACIÓN DE SINTAXIS ABSTRACTA UNO (NSA.1)

Recomendación UIT-T X.208

(Extracto del *Libro Azul*)

NOTAS

1 La Recomendación UIT-T X.208 se publicó en el fascículo VIII.4 del Libro Azul. Este fichero es un extracto del Libro Azul. Aunque la presentación y disposición del texto son ligeramente diferentes de la versión del Libro Azul, el contenido del fichero es idéntico a la citada versión y los derechos de autor siguen siendo los mismos (Véase a continuación).

2 Por razones de concisión, el término «Administración» se utiliza en la presente Recomendación para designar a una administración de telecomunicaciones y a una empresa de explotación reconocida.

© UIT 1988, 1993

Reservados todos los derechos. No podrá reproducirse o utilizarse la presente Recomendación ni parte de la misma de cualquier forma ni por cualquier procedimiento, electrónico o mecánico, comprendidas la fotocopia y la grabación en micropelícula, sin autorización escrita de la UIT.

Recomendación X.208

ESPECIFICACIÓN DE LA NOTACIÓN DE SINTAXIS ABSTRACTA UNO (NSA.1)[1)]

(Melbourne, 1988)

El CCITT,

considerando

a) la variedad y complejidad de los objetos de información transportados en la capa de aplicación;

b) la necesidad de una notación de alto nivel para especificar tales objetos de información;

c) la conveniencia de destacar y normalizar las reglas para codificar tales objetos de información;

recomienda por unanimidad

1) que la notación para definir la sintaxis abstracta de los objetos de información sea la que se define en la sección 1;

2) que los tipos de cadenas de caracteres sean los que se definen en la sección 2;

3) que otros tipos útiles sean que se definen en la sección 3;

4) que los subtipos sean los que se definen en la sección 4.

ÍNDICE

1) La Recomendación X.208 y la norma ISO 8824 ([Information processing systems – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)], ampliada por el Addéndum 1 a ISO 8824, fueron preparadas en estrecha colaboración y están técnicamente armonizadas.

0 Introducción

En las capas bajas del modelo básico de referencia (véase la Recomendación X.200), cada parámetro de datos de usuario de una primitiva de servicio se especifica como el valor binario de una secuencia de octetos.

En la capa de presentación, cambia la naturaleza de los parámetros de los datos de usuario. Las especificaciones de la capa de aplicación requieren que los datos de usuario del servicio de presentación (véase la Recomendación X.216) lleven el valor de tipos de gran complejidad, incluidas posiblemente cadenas de caracteres procedentes de una diversidad de juegos de caracteres. Con el fin de especificar el valor que es cursado, requieren una notación definida que no determina la representación del valor. Esta se complementa con la especificación de uno o más algoritmos denominados **reglas de codificación** que determinan el valor de los octetos de la capa de sesión que cursan tales valores de la capa de aplicación (denominada **sintaxis de transferencia**). El protocolo de la capa de presentación (véase la Recomendación X.226) puede negociar las sintaxis de transferencia que deben utilizarse.

El objetivo de especificar un valor es distinguirlo de otros valores posibles. La colección del valor junto con los valores de los que es diferenciado se llama **tipo**, y una instancia específica es un *valor* de aquel tipo. Mas en general un valor o tipo puede a menudo considerarse compuesto de varios valores o tipos más simples, junto con las relaciones existentes entre ellos. El término tipo de datos se usa a menudo como sinónimo de tipo.

Para interpretar correctamente la representación de un valor (ya sea por marcas en un papel o bits en una línea de comunicación) es necesario saber (por lo general a partir del contexto), el tipo del valor que se representa. Así pues, la identificación de un tipo es una parte importante de esta Recomendación.

Una técnica muy general para definir un tipo complejo es definir un pequeño número de **tipos simples** definiendo todos los posibles valores de los tipos simples, y luego combinar estos tipos simples de varias formas. Algunas de las formas de definir tipos nuevos son las siguientes:

a) dada una lista (ordenada) de tipos existentes, se puede formar un valor como una secuencia (ordenada) de valores, uno de cada uno de los tipos existentes; la colección de todos los valores posibles obtenida de esta forma es un tipo nuevo (si los tipos existentes en la lista son todos distintos, este mecanismo puede ser ampliado para permitir la omisión de algunos valores de la lista);

b) dada una lista de tipos existentes (diferentes), se puede formar un valor como un conjunto (desordenado) de valores, uno de cada uno de los tipos existentes; la colección de todos los valores posibles obtenida de esta forma es un tipo nuevo (el mecanismo puede ser de nuevo ampliado para permitir la omisión de algunos valores);

c) dado un tipo existente simple, se puede formar un valor como una secuencia (ordenada) o conjunto (desordenado) de cero, uno o más valores de tipos existentes; la colección (infinita) de todos los valores posibles así obtenidos es un tipo nuevo;

d) dada una lista de tipos (diferentes), se puede escoger un valor de cualquiera de ellos; el conjunto de todos los valores posibles así obtenido es un tipo nuevo;

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

Los tipos definidos de esta manera se denominan **tipos estructurados**.

A cada tipo definido usando la notación especificada en esta Recomendación se le asigna un **rótulo**. El rótulo lo define esta Recomendación o el usuario de la notación.

Es frecuente que el mismo rótulo se asigne a muchos tipos diferentes, y que el tipo particular sea identificado en el contexto en el que se usa el rótulo.

El usuario de la notación puede optar por asignar rótulos diferentes a dos ocurrencias de un tipo simple, creando para ello dos tipos distintos. Esto puede ser necesario cuando se exige distinguir qué elección se ha hecho en situaciones tales como las del apartado d) anterior.

Se especifican cuatro clases de rótulos en la notación.

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

a) se asigna a un tipo simple; o

b) se asigna a un mecanismo de construcción.

La segunda clase de rótulos es la clase **aplicación**. Los rótulos de la clase aplicación se asignan a tipos por otras normas o Recomendaciones. Dentro de una norma o Recomendación, un rótulo de clase aplicación se asigna solamente a un tipo.

La tercera clase es la clase **privada**. Los rótulos de clase privada no son nunca asignados por normas de la ISO ni por Recomendaciones del CCITT. Su utilización es específica de la aplicación.

La última clase de rótulos es la clase **contexto-específico**. Esta es libremente asignada dentro de cualquier utilización de esta notación e interpretada de acuerdo con el contexto en el que se utiliza.

Los rótulos están principalmente destinados a utilizarse en máquinas y no son esenciales para la notación humana definida en esta Recomendación. No obstante, cuando sea necesario exigir que ciertos tipos sean diferentes, ello se expresará exigiendo que tengan rótulos distintos. La asignación de los rótulos es pues una parte importante de la utilización de esta notación.

Nota 1 – Todos los tipos que pueden definirse en la notación de esta Recomendación tienen un rótulo. Dado un tipo cualquiera, el usuario de la notación puede definir un tipo nuevo con un rótulo diferente.

Nota 2 – Las reglas de codificación llevan siempre el rótulo de un tipo, implícita o explícitamente, con cualquier representación del valor del tipo. Las limitaciones impuestas al uso de la notación están concebidas para asegurar que el rótulo determine inequívocamente el tipo real, con tal que las definiciones aplicables del tipo estén disponibles.

Esta Recomendación especifica una notación que permite definir tipos complejos y especificar valores de estos tipos. Esto se realiza sin determinar la forma en la que haya de representarse (por una secuencia de octetos) una instancia de este tipo durante la transferencia. Una notación que proporciona esta posibilidad se denomina **notación para definición de sintaxis abstracta**.

La finalidad de esta Recomendación es especificar una notación para definición de sintaxis abstracta denominada **notación de sintaxis abstracta uno**, o NSA.1. La notación en sintaxis abstracta uno se utiliza como instrumento semiformal para definir protocolos. La utilización de la notación no impide necesariamente especificaciones ambiguas. Es responsabilidad de los usuarios de la notación asegurar que sus especificaciones no sean ambiguas.

Esta Recomendación se completa con otras normas y Recomendaciones que especifican *reglas de codificación*. La aplicación de reglas de codificación al valor de un tipo definido por la NSA.1 produce una especificación completa de la representación de valores de dicho tipo durante la transferencia (una sintaxis de transferencia).

Esta Recomendación está redaccional y técnicamente en armonía con las publicaciones ISO 8824 y el Addéndum 1 a ISO 8824.

La sección 1 de esta Recomendación define los tipos simples permitidos por la NSA.1 y especifica la notación que se utiliza para referirse a tipos simples y definir tipos estructurados. La sección 1 también especifica la notación que se utiliza para especificar valores de tipos definidos utilizando NSA.1.

La sección 2 de esta Recomendación define tipos adicionales [tipos character string (cadena de caracteres)] los cuales, mediante la aplicación de reglas de codificación para juegos de caracteres, pueden ser equiparados al tipo octetstring (cadena de octetos).

La sección 3 de esta Recomendación define ciertos tipos estructurados considerados de utilidad general pero que no necesitan reglas de codificación adicionales.

La sección 4 de esta Recomendación define una notación que permite definir subtipos a partir de los valores de un tipo progenitor.

El anexo A forma parte de esta Recomendación y especifica una notación para ampliar la notación básica NSA.1. A esto se le llama facilidad macro.

El anexo B forma parte de esta Recomendación y define el árbol de identificadores de objetos para autoridades admitidos por la ISO.

El anexo C forma parte de esta Recomendación y define el árbol de identificadores de objetos para autoridades admitidos por el CCITT.

El anexo D forma parte de esta Recomendación y define el árbol de identificadores de objetos para uso conjunto por el CCITT y la ISO.

El apéndice I no forma parte de esta Recomendación, y proporciona ejemplos y orientaciones sobre la utilización de la NSA.1.

El apéndice II no forma parte de esta Recomendación, y proporciona un resumen de la NSA.1 utilizando la notación del § 5.

El texto de esta Recomendación, y en particular los anexos B a D, son el resultado de un acuerdo entre la ISO y el CCITT.

1 Objeto y campo de aplicación

Esta Recomendación especifica una notación para la definición en sintaxis abstracta denominada notación de sintaxis abstracta uno (NSA.1).

Esta Recomendación define cierto número de tipos simples, con sus rótulos, y especifica una notación para referenciar estos tipos y para especificar valores de los mismos.

Esta Recomendación define mecanismos para construir tipos nuevos a partir de tipos más básicos y especifica una notación para definir tales tipos estructurados y asignarles rótulos, y para especificar valores de estos tipos.

Esta Recomendación define juegos de caracteres para su utilización dentro de la NSA.1.

Esta Recomendación define una cierta cantidad de tipos útiles (mediante NSA.1) a los cuales pueden referirse los usuarios de la NSA.1.

La notación NSA.1 puede aplicarse siempre que sea necesario para definir la sintaxis abstracta de información. Es particularmente, pero no exclusivamente, aplicable a los protocolos de aplicación.

Se hace referencia a la notación NSA.1 en otras normas y Recomendaciones relativas a la capa de presentación que definen reglas de codificación para los tipos simples, los tipos estructurados, los tipos cadena de caracteres y los tipos útiles definidos en NSA.1.

2 Referencias

[1] Recomendación X.200, *Modelo de referencia de interconexión de sistemas abiertos para aplicaciones del CCITT* (véase también la norma ISO 7498).

[2] Recomendación X.209, *Especificación de las reglas básicas de codificación de la notación de sintaxis abstracta uno (NSA.1)*, (véase también la norma ISO 8825).

[3] Recomendación X.216, *Definición del servicio de presentación para la interconexión de sistemas abiertos para aplicaciones del CCITT* (véase también la norma ISO 8822).

[4] Recomendación X.226, *Especificación del protocolo de presentación para la interconexión de sistemas abiertos para aplicaciones del CCITT* (véase también la norma ISO 8823).

[5] ISO 2014, *Writing of calendar dates in all-numeric form.*

[6] ISO 2375, *Data processing – Procedure for registration of escape sequences*.

[7] ISO 3166, *Codes for the representation of names of countries.*

[8] ISO 3307, *Information interchange – Representations of time of the day.*

[9] ISO 4031, *Information interchange – Representation of local time differentials.*

[10] ISO 6523, *Data interchange – Structure for identification of organizations.*

[11] Recomendación X.121, *Plan de numeración internacional por redes públicas de datos.*

3 Definiciones

En esta Recomendación se emplean las definiciones de la Recomendación X.200.

3.1 **valor**

Miembro diferenciado de un conjunto de valores.

3.2 **tipo**

Conjunto denominado de valores.

3.3 **tipo simple**

Tipo definido especificando directamente el conjunto de sus valores.

3.4 **tipo estructurado**

Tipo definido por referencia a uno o más tipos.

3.5 **tipo componente**

Uno de los tipos referenciados al definir un tipo estructurado.

3.6 **rótulo**

Indicación de tipo asociada a cada tipo NSA.1.

3.7 **rotulación**

Sustitución del rótulo existente (posiblemente por defecto) de un tipo por un rótulo especificado.

3.8 **juego de caracteres NSA.1**

Conjunto de caracteres, especificado en el § 7, utilizado en la notación NSA.1.

3.9 **elementos**

Secuencias denominadas de caracteres del juego de caracteres NSA.1, especificado en el § 8, utilizados para formar la notación NSA.1.

3.10 **nombre de referencia de tipo (o de valor)**

Nombre asociado unívocamente con un tipo (o valor) dentro de un contexto.

Nota – Los nombres de referencia se asignan a los tipos definidos en esta Recomendación; éstos están universalmente disponibles dentro de la NSA.1. Otros nombres de referencia están definidos en otras normas y Recomendaciones, y son aplicables solamente en el contexto de dichas normas o Recomendaciones.

3.11 **reglas de codificación NSA.1**

Reglas que especifican la representación durante la transferencia del valor de cualquier tipo NSA.1; las reglas de codificación NSA.1 permiten que la información transferida sea identificada por el destinatario como un valor específico de un tipo específico NSA.1.

3.12 **tipo characterstring (cadena de caracteres)**

Tipo cuyos valores son cadenas de caracteres de algún juego de caracteres definido.

3.13 **tipo boolean (booleano)**

Tipo simple con dos valores diferenciados.

3.14 **cierto**

Uno de los valores diferenciados del tipo booleano.

3.15 **falso**

El otro valor diferenciado del tipo booleano.

3.16 **tipo integer (entero)**

Tipo simple con valores diferenciados que son números enteros positivos y negativos, incluido el cero (como un valor único).

Nota – Las reglas de codificación particulares limitan la gama de un entero, pero tales limitaciones se escogen de tal forma que no afecten a los usuarios de la NSA.1.

3.17 **tipo enumerated (enumerado)**

Tipo simple a cuyos valores se les da identificadores distintos como parte de la notación de tipo.

3.18 **tipo real**

Tipo simple cuyos valores diferenciados (especificados en el § 16.2) son miembros del conjunto de números reales.

3.19 **tipo bitstring (cadena de bits)**

Tipo simple cuyos valores diferenciados son una secuencia ordenada de cero, uno o más bits.

Nota – Las reglas de codificación no limitan el número de bits de una cadena de bits.

3.20 **tipo octetstring (cadena de octetos)**

Tipo simple cuyos valores diferenciados son una secuencia ordenada de cero, uno o más octetos; cada octeto es una secuencia ordenada de ocho bits.

Nota – Las reglas de codificación no limitan el número de octeto de una cadena de octetos.

3.21 **tipo null (nulo)**

Tipo simple consistente en un valor único, también llamado nulo.

Nota – El valor nulo se utiliza generalmente cuando son posibles varias alternativas pero ninguna de ellas se aplica.

3.22 **tipo sequence (secuencia)**

Tipo estructurado, definido referenciando una lista de tipos, fijada y ordenada (algunos de los cuales pueden ser declarados opcionales); cada valor de un tipo nuevo es una lista ordenada de valores, uno de cada tipo componente.

Nota – Cuando un tipo componente es declarado opcional, un valor del tipo nuevo no necesita contener un valor de ese tipo componente.

3.23 **tipo sequence-of (secuencia-de)**

Tipo estructurado definido referenciando un único tipo existente; cada valor en el nuevo tipo es una lista ordenada de cero, uno o más valores del tipo existente.

Nota – Las reglas de codificación no limitan el número de valores en un valor secuencia-de.

3.24 **tipo set (conjunto)**

Tipo estructurado, definido referenciando una lista de tipos distintos, fijada y desordenada (algunos de los cuales pueden ser declarados opcionales); cada valor del tipo nuevo es una lista desordenada de valores, uno a partir de cada uno de los tipos componentes.

Nota – Cuando un tipo componente se declara opcional, el tipo nuevo no necesita contener el valor de ese tipo componente.

3.25 **tipo set-of (conjunto-de)**

Tipo estructurado, definido referenciando un tipo existente único; cada valor en el tipo nuevo es una lista desordenada de cero, uno o más valores del tipo existente.

Nota – Las reglas de codificación no limitan el número de valores en un valor conjunto-de.

3.26 **tipo tagged (rotulado)**

Tipo definido referenciando un tipo existente único y a un rótulo; el tipo nuevo es isomórfico del tipo existente, pero es distinto de él.

3.27 **tipo choice (elección)**

Tipo estructurado, definido referenciando una lista de tipos distintos, fijada y desordenada; cada valor del tipo nuevo es un valor de uno de los tipos componentes.

3.28 **tipo selection (selección)**

Tipo estructurado, definido referenciando un tipo componente de un tipo selección.

3.29 **tipo any (cualquiera)**

Tipo elección cuyos tipos componentes están sin especificar pero están restringidos al conjunto de tipos que pueden ser definidos utilizando NSA.1.

3.30 **tipo external (externo)**

Tipo cuyos valores diferenciados no pueden ser deducidos de su caracterización como externos, pero que pueden ser deducidos de la codificación de tal valor; los valores pueden, pero no necesitan, ser describibles utilizando la NSA.1, y así su codificación puede, pero no necesita, ajustarse a las reglas de codificación NSA.1.

3.31 **objeto de información**

Elemento de información, definición o especificación bien definido que requiere un nombre para identificar su utilización en una instancia de comunicación.

3.32 **identificador de objeto**

Valor (distinguible de todos los demás valores) que está asociado con un objeto de información.

3.33 **tipo object identifier (identificador de objeto)**

Tipo cuyos valores diferenciados son el conjunto de todos los identificadores de objeto atribuidos de acuerdo con las reglas de esta Recomendación.

Nota – Las reglas de esta Recomendación permiten a una amplia gama de autoridades asociar independientemente identificadores de objeto con objeto de información.

3.34 **tipo object descriptor (descriptor de objeto)**

Tipo cuyos valores diferenciados son textos legibles por el hombre que proporcionan una breve descripción de un objeto de información.

Nota – Un valor descriptor de objeto está generalmente, pero no siempre, asociado a un único objeto de información. Solamente un valor de identificador de objeto identifica inequívocamente un objeto de información.

3.35 **definiciones recurrentes**

Conjunto de definiciones NSA.1 que no puede ser reordenado, por lo que todos los tipos utilizados en una construcción están definidos antes de la definición de la construcción.

Nota – Las definiciones recurrentes están permitidas en NSA.1: el usuario de la notación tiene la responsabilidad de asegurar que aquellos valores (de los tipos resultantes) que son utilizados tienen una representación finita.

3.36 **módulo**

Una o más instancias de utilización de la notación NSA.1 para definición de tipos y valores, encasillados usando la notación módulo NSA.1 (véase el § 9).

3.37 **producción**

Parte de la notación formal utilizada para especificar la NSA.1, en la cual las secuencias permitidas de los ítems están asociadas con un nombre que puede ser utilizado para referenciar aquellas secuencias en la definición de conjuntos nuevos de secuencias permitidas.

3.38 **Tiempo Universal Coordinado (UTC)**

Escala de tiempo mantenida por el Bureau International de l'Heure (Oficina Internacional de la Hora) que sirve de base para una difusión coordinada de frecuencias patrón y señales horarias.

Nota 1 – El origen de esta definición es la Recomendación 460-2 del Comité Consultivo Internacional de Radiocomunicaciones (CCIR). El CCIR también ha definido que la abreviatura para Tiempo Universal Coordinado sea UTC.

Nota 2 – El UTC se llama también Hora Media de Greenwich, y se difunden regularmente señales horarias apropiadas.

3.39 **usuario (de NSA.1)**

Individuo u organización que define la sintaxis abstracta de una determinada parte de información utilizando NSA.1.

3.40 **subtipo (de un tipo progenitor)**

Tipo cuyos valores están especificados como un subconjunto de valores de algún otro tipo (el tipo progenitor).

3.41 **tipo progenitor (de un subtipo)**

Tipo utilizado para definir un subtipo.

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

3.42 **especificación de subtipo**

Notación que puede utilizarse en asociación con la notación de un tipo, para definir un subtipo de ese tipo.

3.43 **conjunto de valores de un subtipo**

Notación que forma parte de una especificación de subtipo, especificando un conjunto de valores del tipo progenitor que han de ser incluidos en el subtipo.

3.44 Esta Recomendación utiliza los siguientes términos definidos en la Recomendación X.216:

a) valor de datos de presentación;

b) (una) sintaxis abstracta;

c) nombre de sintaxis abstracta;

d) nombre de sintaxis de transferencia.

3.45 Esta Recomendación también utiliza los siguientes términos definidos en ISO 6523:

a) organización, emisora (issuing organization);

b) código de organización (organization code);

c) designador de código internacional (International Code Designator).

3.46 Esta Recomendación utiliza los siguientes términos definidos en la Recomendación X.226:

a) identificador de contexto de presentación.

4 Abreviaturas

NSA.1 Notación de sintaxis abstracta uno

UTC Tiempo Universal Coordinado

DCI Designador de código internacional

IPD Indicativo de país para datos

CIRD Código de identificación de red de datos

5 Notación usada en esta Recomendación

La notación NSA.1 consiste en una secuencia de caracteres del juego de caracteres NSA.1 especificado en el § 7.

Cada utilización de la notación NSA.1 contiene caracteres del juego de caracteres NSA.1 agrupados en ítems. La cláusula 8 especifica todas las secuencias de caracteres que forman parte de los ítems del juego de caracteres de la NSA.1, y los nombra.

La notación NSA.1 se especifica en el § 9 (y cláusulas siguientes) especificando la colección de secuencias de ítems que forman instancias válidas de la notación NSA.1, y especificando la semántica de tales secuencias.

Para especificar dichas colecciones, esta Recomendación utiliza una notación formal definida en las siguientes subcláusulas.

5.1 *Producciones*

Una colección nueva (más compleja) de secuencias NSA.1 se define por medio de una producción. Esta utiliza nombres de colecciones de secuencias definidas en esta Recomendación y forma una nueva colección de secuencias, especificando ya sea:

a) que la colección nueva de secuencias tiene que consistir en cualquier secuencia contenida en cualquiera de las colecciones originales; o

b) que la nueva colección tiene que consistir en cualquier secuencia que pueda ser generada tomando exactamente una secuencia de cada colección, y yuxtaponiéndola en un orden determinado.

Cada producción consta de las partes siguientes, en una o varias líneas, en orden a:

a) un nombre para la nueva colección de secuencias;

b) los caracteres

::=

c) una o más colecciones alternativas de secuencias, definidas en el § 5.2, separadas por el carácter

|

Una secuencia está presente en la nueva colección si está presente en una o más de las colecciones alternativas. La nueva colección está referenciada en esta Recomendación por el nombre indicado en el apartado anterior a).

Nota – Si la misma secuencia aparece en más de una alternativa, cualquier ambigüedad semántica en la notación resultante se resolverá en otras partes de la secuencia completa NSA.1.

5.2 *Colecciones alternativas*

Cada una de las colecciones alternativas de secuencias en "una o más colecciones alternativas de" se especifica por medio de una lista de nombres. Cada nombre es el nombre de un ítem o es nombre de una colección de secuencias definidas por una producción en esta Recomendación.

La colección de secuencias definidas por la alternativa consiste en todas las secuencias obtenidas al tomar cualquiera de las secuencias (o el ítem) asociadas al primer nombre, en combinación con (y seguida por) cualquiera de las secuencias (o el elemento) asociadas con el segundo nombre en combinación con (y seguida por) cualquiera de las secuencias (o el elemento) asociadas al tercer nombre, y así sucesivamente hasta incluir el último nombre (o el ítem) de la alternativa.

5.3 *Ejemplo de una producción*

BitString Value ::=

bstring |
hstring |
{IdentifierList}

es una producción que asocia con el nombre **BitStringValue** (ValorCadenaBits) las siguientes secuencias:

a) cualquier bstring (cadenab) (un elemento);

b) cualquier hstring (cadenah) (un elemento);

c) cualquier secuencia asociada con la **IdentifierList** (ListaIdentificadores), precedida por una {y seguida por una}.

Nota – {y} son los nombres de elementos que contienen caracteres únicos {y} (véase § 8).

En este ejemplo **IdentifierList** debería ser definida por una producción posterior, ya sea después o antes de definir la producción **BitStringValue** (ValorCadenaBits).

5.4 *Disposición*

Cada producción utilizada en esta Recomendación está precedida y seguida por una línea vacía. Las líneas vacías no aparecen dentro de las producciones. La producción puede estar en una sola línea o distribuida en varias líneas. La disposición no es significativa.

5.5 *Recurrencia*

Las producciones en esta Recomendación son frecuentemente recurrentes. En este caso las producciones han de ser continuamente reaplicadas hasta que no se generen nuevas secuencias.

Nota – En muchos casos tal reaplicación produce una colección ilimitada de secuencias permitidas, de las cuales alguna o todas ellas pueden ser ilimitadas. Esto no es un error.

5.6 *Referencias a una colección de secuencias*

Esta Recomendación referencia una colección de secuencias (parte de la NSA.1) referenciando el primer nombre (antes de ::=) en una producción; el nombre está entre “ para distinguirlo del texto en el lenguaje natural, a no ser que aparezca como parte de una producción.

5.7 *Referencias a un elemento*

Esta Recomendación referencia un elemento referenciando el nombre del elemento; el nombre está entre ” para distinguirlo del texto en el lenguaje natural, a no ser que aparezca como parte de una producción.

CUADRO 1/X.208

Asignaciones de rótulos de clase universal

UNIVERSAL 1	Tipo boolean
UNIVERSAL 2	Tipo integer
UNIVERSAL 3	Tipo bitstring
UNIVERSAL 4	Tipo octetstring
UNIVERSAL 5	Tipo null
UNIVERSAL 6	Tipo object identifier
UNIVERSAL 7	Tipo object descriptor
UNIVERSAL 8	Tipo external
UNIVERSAL 9	Tipo real
UNIVERSAL 10	Tipo enumerated
UNIVERSAL 12-15	Reservado para versiones futuras de esta Recomendación
UNIVERSAL 16	Tipos sequence y sequence-of
UNIVERSAL 17	Tipos set y set-of
UNIVERSAL 18-22, 25-27	Tipo characterstring
UNIVERSAL 23, 24	Tipo time
UNIVERSAL 28-. . .	Reservado para versiones futuras de esta Recomendación

5.8 *Rótulos*

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

universal
aplicación
privada
específico a un contexto

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

Las restricciones en los rótulos asignados por el usuario de NSA.1 se especifican en el § 26.

Los rótulos en la clase universal se asignan de tal forma que, para tipos estructurados, la estructura de alto nivel puede dedurcirse a partir del rótulo y para tipos simples el tipo puede deducirse a partir del rótulo. El cuadro 1/X.208 resume la asignación de rótulos de la clase universal especificados en esta Recomendación.

Nota – Se reservan los rótulos adicionales en la clase universal para ser asignados en futuras versiones de esta Recomendación.

6 Uso de la notación NSA.1

6.1 La notación NSA.1 para una definición de un tipo será "Type" (Tipo) (véase el § 12.1).

6.2 La notación NSA.1 para un valor de un tipo será "Value" (Valor) (véase § 12.7).

Nota – No es posible generalmente interpretar la notación del valor sin conocer la del tipo.

6.3 La notación NSA.1 para asignar un tipo a un nombre de referencia de tipo será "Typeassignment" (Asignacióntipo) (véase § 11.1).

6.4 La notación NSA.1 para asignar un valor a un nombre de referencia de valor será "Valueassignment" (Asignaciónvalor) (véase § 11.2).

6.5 La notación "Typeassignment" (Asignacióntipo) y "Valueassignment" (Asignaciónvalor) serán solamente utilizadas dentro de la notación "ModuleDefinition" (DefiniciónMódulo) (véase no obstante el § 9.1).

SECCIÓN 1 – ESPECIFICACIÓN DE LA NOTACIÓN NSA.1

7 Juego de caracteres NSA.1

7.1 Un ítem NSA.1 consistirá en una secuencia de los caracteres indicados en el cuadro 2/X.208, excepto en lo especificado en los § 7.2 y y § 7.3.

CUADRO 2/X.208

Caracteres NSA.1

A a Z
a a z
0 a 9
: = , { } < .
( ) [ ] - ' "

Nota 1 – En la notación macro se utilizan además los caracteres > y | .

Nota 2 – Cuando se desarrollen normas derivadas equivalentes por organismos de normalización nacionales, pueden aparecer nuevos caracteres en los siguientes ítems (los cinco últimos están definidos en el anexo A):

typereference	(§ 8.2.1)
identifier	(§ 8.3)
valuereference	(§ 8.4)
modulereference	(§ 8.5)
macroreference	(§ A.2.1)
productionreference	(§ A.2.2)
localtypereference	(§ A.2.3)
localvaluereference	(§ A.2.4)
astring	(§ A.2.7)

Nota 3 – Cuando se introducen 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 tiene que ser conseguida de alguna otra forma.

7.2 Cuando se utiliza la notación para especificar el valor de un tipo cadena de caracteres, todos los caracteres del juego de caracteres definidos pueden aparecer en la NSA.1, entre comillas (véase § 8.11).

7.3 En el ítem "comentario" (comment) pueden aparecer caracteres adicionales. (Véase § 8.6)

7.4 El significado no recaerá en el estilo tipográfico, tamaño, color, intensidad y otras características de visualización.

7.5 Las letras mayúsculas y minúsculas deberán considerarse distintas.

8 Elementos en la NSA.1

8.1 *Reglas generales*

8.1.1 Las siguientes subcláusulas especifican los caracteres en los elementos NSA.1. En cada caso se da el nombre del elemento, junto con la definición de las secuencias de caracteres que forman el ítem.

Nota – El anexo A especifica los elementos adicionales utilizados en la notación macro.

8.1.2 Cada elemento especificado en las siguientes subcláusulas aparecerá en una línea única y [salvo para el elemento "comment" (comentario)] no contendrá espacios.

8.1.3 La longitud de la línea no está limitada.

8.1.4 Los elementos de las secuencias especificadas por esta Recomendación (la NSA.1) pueden aparecer en una línea o en varias y pueden estar separados por uno o más espacios y por líneas vacías.

8.1.5 Un elemento estará separado del siguiente o por un espacio o por estar en una línea separada, si el carácter inicial (o caracteres) del elemento siguiente es un carácter (o caracteres) autorizado para su inclusión al final de los caracteres del elemento anterior.

8.2 *Referencias tipo*

Nombre de elemento-typereference

8.2.1 Una "typereference" constará de un número arbitrario de (una o más) letras, cifras y guiones. El carácter inicial será una letra mayúscula. El 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 ambigüedad en los comentarios (que posiblemente sigan).

8.2.2 Una "typereference" no será una de las secuencias reservadas de caracteres relacionados en el cuadro 3/X.208.

Nota – En el § A.2.9 se especifican secuencias reservadas adicionales de caracteres dentro de una definición macro.

8.3 *Identificadores*

Nombre de elemento-identifier

Un "identifier" estará constituido por un número arbitrario de (una o más) letras, dígitos y guiones. El carácter inicial será una letra minúscula. El 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 en los comentarios (que posiblemente sigan).

CUADRO 3/X.208

Secuencias de caracteres reservados

BOOLEAN	(BOOLEANO)	BEGIN	(COMIENZO)
INTEGER	(ENTERO)	END	(FINAL)
BIT	(BIT)	DEFINITIONS	(DEFINICIONES)
STRING	(CADENA)	EXPLICIT	(EXPLÍCITO)
OCTET	(OCTETO)	ENUMERATED	(ENUMERADO)
NULL	(NULO)	EXPORTS	(EXPORTACIONES)
SEQUENCE	(SECUENCIA)	IMPORTS	(IMPORTACIONES)
OF	(DE)	REAL	(REAL)
SET	(CONJUNTO)	INCLUDES	(INCLUSIONES)
IMPLICIT	(IMPLÍCITO)	MIN	(MÍNIMO)
CHOICE	(ELECCIÓN)	MAX	(MÁXIMO)
ANY	(CUALQUIERA)	SIZE	(TAMAÑO)
EXTERNAL	(EXTERNO)	FROM	(DESDE)
OBJECT	(OBJETO)	WITH	(CON)
IDENTIFIER	(IDENTIFICADOR)	COMPONENT	(COMPONENTE)
OPTIONAL	(OPCIONAL)	PRESENT	(PRESENTE)
DEFAULT	(PORDEFECTO)	ABSENT	(AUSENTE)
COMPONENTS	(COMPONENTES)	DEFINED	(DEFINIDO)
UNIVERSAL	(UNIVERSAL)	BY	(POR)
APPLICATION	(APLICACIÓN)	PLUS-INFINITY	(MÁS-INFINITO)
PRIVATE	(PRIVADO)	MINUS-INFINITY	(MENOS-INFINITO)
TRUE	(VERDADERO)	TAGS	(RÓTULOS)
FALSE	(FALSO)		

8.4 *Referencias valor*

Nombre de elemento-valuereference

Una “valuereference” estará constituida por la secuencia de caracteres especificada por un “identificador” (identifier) en el § 8.3. Al analizar una instancia de utilización de esta notación, una “valuereference” se distingue de un “identifier” por el contexto en el que aparece.

8.5 *Referencia módulo*

Nombre de elemento-modulereference

Una “modulereference” estará constituida por la secuencia de caracteres especificada por una “typereference” en el § 8.2. Al analizar una instancia de utilización de esta notación, una “modulereference” se distingue de una “typereference” por el contexto en el que aparece.

8.6 *Comentario*

Nombre de elemento-comment

8.6.1 Un “comment” no está referenciado en la definición de la notación NSA.1. Puede, sin embargo, aparecer en cualquier momento entre otros ítems y no tiene significación.

8.6.2 Un “comment” comenzará con un par de guiones adyacentes y terminará con el próximo par de guiones adyacentes o el final de la línea, lo que suceda primero. Un “comment” no contendrá un par de guiones adyacentes además del par que lo abre y el par (de haberlo), que lo cierra. Puede incluir caracteres que no están en el juego de caracteres especificados en el § 7.1 (véase el § 7.3).

8.7 *Elemento vacío*

Nombre de elemento-empty

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

8.8 *Elemento número*

Nombre de elemento-number

Un “number” constará de una o más cifras. La primera cifra no será cero a menos que el “number” sea de una sola cifra.

8.9 *Elemento cadena binaria*

Nombre de elemento-bstring

Una “bstring” constará de un número arbitrario (que puede ser cero) de ceros y unos, precedidos por un único ’ y seguidos por el par de caracteres:

’B

Ejemplo: ’01101100’B

8.10 *Elemento cadena hexadecimal*

Nombre de elemento-hstring

8.10.1 Una “hstring” constará de 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

precedidos por un único ’ y seguidos por el par de caracteres

’H

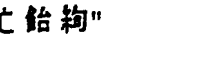
Ejemplo: ’AB0196’H

8.10.2 Cada carácter se utiliza para designar el valor de un semiocteto usando una representación hexadecimal.

8.11 *Elemento cadena de caracteres*

Nombre de elemento-cstring

Una “cstring” constará de un número arbitrario (que puede ser cero) de caracteres del juego de caracteres referenciados por un tipo bitstring, precedidos y seguidos de ”. Si el juego de caracteres incluye el carácter ”, éste estará representado en la “cstring” por un par de ”. El juego de caracteres involucrado no está limitado al juego de caracteres indicados en el cuadro 2/X.208, pero está determinado por el tipo para el cual la “cstring” es un valor (véase el § 7.2).

Ejemplo: 

8.12 *Elemento asignación*

Nombre de elemento-“::=”

Este elemento estará constituido en la secuencia de caracteres

::=

Nota – Esta secuencia no contiene ningún carácter espacio (véase el § 8.1.2).

8.13 *Elementos carácter único*

Nombres de los elementos:

{
}
<
,
.
(
)
[
]
- (guión)
;

Un elemento con cualquiera de los nombres enumerados anteriormente estará constituido por el carácter único que forma el nombre.

8.14 *Elementos palabras clave*

Nombres de los elementos –

BOOLEAN (BOOLEANO)
INTEGER (ENTERO)
BIT (BIT)
STRING (CADENA)
OCTECT (OCTETO)
NULL (NULO)
SEQUENCE (SECUENCIA)
OF (DE)
SET (CONJUNTO)
IMPLICIT (IMPLÍCITO)
CHOICE (ELECCIÓN)
ANY (CUALQUIERA)
EXTERNAL (EXTERNO)
OBJECT (OBJETO)
IDENTIFIER (IDENTIFICADOR)
OPTIONAL (OPCIONAL)
DEFAULT (PORDEFECTO)
COMPONENTS (COMPONENTES)
UNIVERSAL (UNIVERSAL)
APPLICATION (APLICACIÓN)
PRIVATE (PRIVADO)
TRUE (VERDADERO)
FALSE (FALSO)
BEGIN (COMIENZO)
END (FINAL)
DEFINITIONS (DEFINICIONES)
EXPLICIT (EXPLÍCITO)
ENUMERATED (ENUMERADO)
EXPORTS (EXPORTACIONES)
IMPORTS (IMPORTACIONES)
REAL (REAL)
INCLUDES (INCLUSIONES)

MIN (MÍNIMO)
MAX (MÁXIMO)
SIZE (TAMAÑO)
FROM (DESDE)
WITH (CON)
COMPONENT (COMPONENTE)
PRESENT (PRESENTE)
ABSENT (AUSENTE)
DEFINED (DEFINIDO)
BY (POR)
PLUS-INFINITY (MÁS-INFINITO)
MINUS-INFINITY (MENOS-INFINITO)
TAGS (RÓTULOS)

Los elementos con los nombres anteriores estarán constituidos por la secuencia de caracteres del nombre.

Nota 1 – No deben aparecer espacios en estas secuencias.

Nota 2 – Cuando estas secuencias no estén enumeradas como secuencias reservadas en el § 8.2.2, se diferenciarán de otros elementos que contengan los mismos caracteres por el contexto en el que aparezcan.

9 Definición de módulo

9.1 Una "ModuleDefinition" (DefiniciónMódulo) se especifica por las siguientes producciones:

ModuleDefinition ::=

ModuleIdentifier
DEFINITIONS
TagDefault
"::="
BEGIN
ModuleBody
END

TagDefault ::=

EXPLICIT TAGS |
IMPLICIT TAGS |
empty

ModuleIdentifier ::=

modulereference
AssignedIdentifier

AssignedIdentifier ::=

ObjectIdentifierValue |
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 ModuleIdentifier

SymbolList ::= Symbol, SymbolList | Symbol

Symbol ::=typereference | valuereference

AssignmentList ::=

Assignment AssignmentList | Assignment

Assignment ::=

TypeAssignment | ValueAssignment

Nota 1 – El anexo A especifica una secuencia "MacroDefinition" (DefiniciónMacro) que puede aparecer también en la "AssignmentList" (ListaAsignación). Las notaciones definidas por una definición macro pueden aparecer antes o después de la definición macro, dentro del mismo módulo.

Nota 2 – En casos individuales (aunque desaconsejados) y para ejemplos y para definición de tipos con rótulos de clase universal, puede utilizarse el "ModuleBody" (CuerpoMódulo) fuera de una "ModuleDefinition".

Nota 3 – Las producciones "Typeassignment" y "Valueassignment" están especificadas en el § 11.

Nota 4 – El agrupamiento de tipos de datos NSA.1 en módulos no determina necesariamente la formación de valores de datos de presentación en sintaxis abstractas nominadas con fines de definición del contexto de presentación.

Nota 5 – El valor de "TagDefault" (RótuloPorDefecto) de la definición módulo afecta solamente a aquellos tipos definidos explícitamente en el módulo. No afecta a la interpretación de tipos importados.

Nota 6 – Una "macroreference" (véase el anexo A), puede también aparecer como un "Symbol" (Símbolo).

9.2 El "TagDefault" es tomado por "EXPLICIT TAGS" (RÓTULOS EXPLÍCITOS) si está "empty" (vacío).

Nota – El § 26 da el significado de "EXPLICIT TAGS" (RÓTULOS EXPLÍCITOS) y de "IMPLICIT TAGS" (RÓTULOS IMPLÍCITOS).

9.3 La "modulereference" que aparece en la producción "ModuleDefinition" se llama nombre del módulo. Los nombres de módulos se escogen para asegurar la coherencia y acabado de todas las secuencias "Assignment" (Asignación) que aparecen dentro de las secuencias "ModuleBody" y "ModuleDefinition" con este nombre de módulo. Un conjunto de secuencias "Assignment" es coherente y completo si, para cada "typereference" o "valuereference" que aparecen dentro de él, hay exactamente una "Typeassignment" o "Valueassignment" respectivamente asociando el nombre con un tipo o valor (respectivamente), o exactamente un "SymbolsFromModule" en el que la "typereference" o "valuereference" (respectivamente) aparece como un "Symbol".

9.4 Los nombres de módulos sólo deben utilizarse una vez (salvo lo establecido en el § 9.10) dentro de la esfera de interés de la definición del módulo.

Nota – Se recomienda que los módulos definidos en las Normas ISO tengan nombres de módulo de la forma

ISOxxxx-yyyy

donde xxxx es el número de la Norma e yyyy es un acrónimo adecuado para la Norma (por ejemplo, JTM, FTAM, o CCR). Un acuerdo similar puede ser aplicado por otros organismos de normalización.

9.5 Si el "AssignedIdentifier" incluye un "ObjectIdentifierValue", el último identifica el módulo sin ambigüedad y unívocamente.

Nota – Se recomienda que sea asignado un identificador objeto para que otros puedan referirse al módulo sin ambigüedad.

9.6 El "ModuleIdentifier" en un "SymbolsFromModule" aparecerá en la "ModuleDefinition" de otro módulo, salvo si incluye un "ObjectIdentifierValue", la "modulereference" puede diferir en ambos casos.

Nota 1 – Una "modulereference" diferente de la usada en el otro módulo debe ser utilizada solamente cuando los símbolos tienen que ser importados de dos módulos con el mismo nombre (la denominación de los módulos no tiene en cuenta el § 9.4). La utilización de nombres alternativos distintos permite utilizar estos nombres en el cuerpo del módulo (véase el § 9.8).

Nota 2 – Cuando se utilizan una "modulereference" y un "ObjectIdentifierValue" para referirse a un módulo, el último será considerado definitivo.

9.7 Cada "Symbol" en "SymbolsExported" estará definido en el módulo que se construye.

Nota – Se recomienda que cada símbolo al que se referencia desde fuera del módulo, sea incluido en "SymbolsExported". Si no los hay, entonces se seleccionaría la alternativa "empty" de "SymbolsExported" (no de "Exports").

9.8 Cada "Symbol" en "SymbolsFromModule" estará definido en el módulo especificado por el "ModuleIdentifier" en "SymbolsFromModule". Si se utiliza "Exports" en la definición de ese módulo, "Symbol" aparecerá en su "SymbolsExported".

9.9 Un "Symbol" en un "SymbolsFromModule" puede aparecer en un "ModuleBody" en un "DefinedType" (si es una "typereference") o "DefinedValue" (si es una "valuereference"). El significado asociado con "Symbol" es el que tiene en el módulo designado por el correspondiente "ModuleIdentifier". Cuando aparezca también "Symbol" en una "AssignmentList" (desaconsejado), o en una o más instancias de "SymbolsFromModule", se utilizará solamente en una "ExternalTypeReference" o "ExternalValueReference" cuya "modulereference" es la "SymbolsFromModule" (véase el § 9.10). Cuando esto no aparezca pueden utilizarse un "DefinedType" o "DefinedValue" directamente.

9.10 Excepto en lo especificado en el § 9.9, una "typereference" o "valuereference" estará referenciada en un módulo diferente de aquel en el que está definida utilizando una "Externaltypereference" o "Externalvaluereference", especificada por medio de las siguientes producciones:

```
Externaltypereference ::=
        modulereference
        typereference
Externalvaluereference ::=
        modulereference
        valuereference
```

10 Referenciación de las definiciones tipo y valor

10.1 Las producciones

```
DefinedType ::=
        Externaltypereference |
        typereference
```

DefinedValue ::==

Externalvaluereference |
valuereference

especifican las secuencias que se utilizarán para referenciar las definiciones de tipo y valor.

10.2 Salvo lo especificado en el § 9.10 no se utilizarán las alternativas "typereference" y "valuereference" a no ser que la referencia esté dentro del módulo en el que está asignado un tipo o valor (véanse los § 11.1 y § 11.2) a la typereference o valuereference.

10.3 No se utilizarán "Externaltypereference" y "Externalvaluereference" a menos que a la correspondiente "typereference" o "valuereference" se les haya asignado un tipo o valor respectivamente (véanse los § 11.1 y § 11.2) dentro de la "modulereference" correspondiente.

11 Asignación de tipos y valores

11.1 La notación especificada por la producción "Typeassignment" asignará un tipo a "typereference":

Typeassignment ::= typereference

"::="
Type

"Typereference" no será uno de los nombres utilizados para referenciar los tipos de cadena de caracteres definidos en la sección 2, y no será uno de los nombres utilizados para referenciar los tipos definidos en la sección 3.

11.2 La notación especificada por la producción "Valueassignment" asignará un valor a "valuereference":

Valueassignment ::= valuereference

Type
"::="
Value

El valor "Value" asignado a la "Valuereference" será una notación válida (véase el § 12.7) para un valor del tipo definido por "Type".

12 Definición de tipos y valores

12.1 Un tipo se referenciará por una de las secuencias "Type":

Type ::= BuiltinType | DefinedType | Subtype
(véase § 10.1) (véase § 37)

BuiltinType ::=

BooleanType |
IntegerType |
BitStringType |
OctetStringType |
NullType |
SequenceType |
SequenceOfType |
SetType |
SetOfType |
ChoiceType |
SelectionType |
TaggedType |
AnyType |
ObjectIdentifierType |
CharacterStringType |

UsefulType |
EnumeratedType |
RealType |

Nota 1 – Una notación tipo definida en una macro puede ser utilizada también como una secuencia para "Type" (véase el anexo A).

Nota 2 – En futuras versiones de esta Recomendación se pueden definir nuevos tipos adicionales.

12.2 La notación "BuiltinType" está especificada en las siguientes cláusulas.

12.3 La notación "Subtype" está especificada en la sección 4.

12.4 El tipo referenciado es el tipo definido por el "BuiltinType" o "Subtype" asignado al "DefinedType".

12.5 En algunas notaciones dentro de las cuales está referenciado un tipo, éste puede ser denominado. En tales casos, esta Recomendación especifica la utilización de la notación "NamedType":

NamedType ::=

identifier Type |
Type |
SelectionType

La notación "SelectionType" y la notación del valor correspondiente están especificadas en el § 25.

Nota – La notación "SelectionType" contiene un "identifier" que puede formar parte de la notación del valor cuando se utiliza "SelectionType" "NamedType" (véase el § 25.1).

12.6 El "identifier" no es parte del tipo, y no afecta al tipo. El tipo referenciado por una secuencia "NamedType" es aquel que está referenciado por la secuencia "Type" contenida.

12.7 El valor del tipo estará especificado por una de las secuencias "Value":

Value ::= BuiltinValue | DefinedValue

BuiltinValue ::=

BooleanValue |
IntegerValue |
BitStringValue |
OctetStringValue |
NullValue |
SequenceValue |
SequenceOfValue |
SetValue |
SetOfValue |
ChoiceValue |
SelectionValue |
TaggedValue |
AnyValue |
ObjectIdentifierValue |
CharacterStringValue |
EnumeratedValue |
RealValue |

Nota – Una notación de valor definida en una macro puede también ser utilizada como una secuencia para "Value" (véase el anexo A).

12.8 Si el tipo está definido utilizando una de las notaciones indicadas a continuación en la columna izquierda, entonces el valor será especificado utilizando la notación de la columna derecha:

Type notation	**Value notation**
BooleanType	BooleanValue
IntegerType	IntegerValue
BitStringType	BitStringValue
OctetStringType	OctetStringValue
NullType	NullValue
SequenceType	SequenceValue
SequenceOfType	SequenceOfValue
SetType	SetValue
SetOfType	SetOfValue
ChoiceType	ChoiceValue
TaggedType	TaggedValue
AnyType	AnyValue
ObjectIdentifierType	ObjectIdentifierValue
CharacterStringType	CharacterStringValue
EnumeratedType	EnumeratedValue
RealType	RealValue

Nota – En futuras versiones de esta Recomendación pueden definirse nuevos valores.

Cuando el tipo es un DefinedType, la notación de valor será la notación para un tipo utilizado al producir el DefinedType.

12.9 La notación de valor para un tipo definido por la notación "UsefulType" está especificada en la sección 3.

12.10 La notación "BuiltinValue" está especificada en las cláusulas siguientes.

12.11 El valor de un tipo referenciado utilizando la notación "NamedType" estará definido por la notación "NamedValue":

NamedValue ::=

identifier Value │
Value

donde el "identifier" (si lo hay) es el mismo que el utilizado en la notación "NamedType". En el § 25.2 se especifican otras restricciones sobre el "NamedValue" cuando el "NamedType" era un "SelectionType".

Nota – El "identifier" es parte de la notación y no forma parte del valor en sí.

12.12 El "identifier" estará presente en "NamedValue" si y solamente si estaba presente en el "NamedType".

Nota – Un "identifier" está siempre presente en el caso de un "SelectionType".

13 Notación para el tipo boolean (booleano)

13.1 El tipo boolean (véase el § 3.13) estará referenciado por la notación "BooleanType".

BooleanType ::= BOOLEAN

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

13.3 El valor de un tipo boolean (véanse los § 3.14 y § 3.15) será definido por la notación "BooleanValue":

BooleanValue ::= TRUE │ FALSE

14 Notación para el tipo integer (entero)

14.1 El tipo integer (véase el § 3.16) será referenciado por la notación "IntegerType":

IntegerType ::=

INTEGER |
INTEGER{NamedNumberList}

NamedNumberList ::=

NamedNumber |
NamedNumberList, NamedNumber

NamedNumber ::=

identifier(SignedNumber) |
identifier(DefinedValue)

SignedNumber ::= number | -number

14.2 La segunda alternativa de "SignedNumber" no se utilizará si el "number" es cero.

14.3 La "NamedNumberList" no es significativa en la definición de un tipo. Se utiliza únicamente en la notación de valor especificada en el § 14.9.

14.4 El "DefinedValue" será una referencia a un valor de tipo entero o a un tipo derivado de un entero por rotulación o por subtipificación.

14.5 El valor de cada "SignedNumber" o "DefinedValue" que aparecerá en la "NamedNumberList" será diferente, y representa un valor diferenciado de un tipo entero.

14.6 Cada "identifier" que aparezca en la "NamedNumberList" será diferente.

14.7 El orden de las secuencias de "NamedNumber" en la "NamedNumberList" no es significativo.

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

14.9 El valor de un tipo entero estará definido por la notación "IntegerValue".

IntegerValue ::=

SignedNumber |
identifier

14.10 El "identifier" en "IntegerValue" será igual al de un "identifier" en la secuencia "IntegerType" con la que el valor está asociado, y representará el número correspondiente.

Nota – Se debe preferir el uso de la forma "identifier" de "IntegerValue" cuando se defina un valor entero para el cual se haya definido un "identifier".

15 Notación para el tipo enumerated (enumerado)

15.1 El tipo enumerated (véase el § 3.17) será referenciado por la notación "EnumeratedType":

EnumeratedType ::= ENUMERATED {Enumeration}

Enumeration ::=

NamedNumber |
NamedNumber, Enumeration

Nota 1 – Cada valor tiene un identificador el cual está asociado, en esta notación, con un entero distinto. Esto proporciona control de la representación del valor para facilitar ampliaciones compatibles, pero los valores por sí mismos no se prevé que tengan ninguna semántica de entero.

Nota 2 – Los valores numéricos dentro de los “NamedNumber” en la “Enumeration” no son necesariamente ordenados y contiguos.

15.2 Para cada “NamedNumber”, el “identifier” y el “SignedNumber” serán distintos de todos los demás "identifier" y "SignedNumber" en la “Enumeration”.

15.3 El tipo enumerated tiene un rótulo que es de clase universal, número 10.

15.4 El valor de un tipo enumerated estará definido por la notación “EnumeratedValue”.

EnumeratedValue ::= identifier

16 Notación para el tipo real

16.1 El tipo real (véase el § 3.18) estará referenciado por la notación “RealType”.

RealType ::= REAL

16.2 Los valores del tipo real son valores PLUS-INFINITY y MINUS-INFINITY junto con los números reales que pueden especificarse mediante la fórmula que involucra tres enteros, M, B y E:

$$M \times B^E$$

donde M se denomina mantisa, B la base, y E exponente. M y E pueden tomar cualquier valor entero, positivo o negativo, mientras que B puede tomar los valores 2 ó 10. Están autorizadas todas las combinaciones de M, B y E.

Nota 1 – Este tipo es capaz de cursar una representación exacta de cualquier número que pueda ser almacenado en un soporte material típico de coma flotante, y de cualquier número con una representación finita de caracteres decimales.

Nota 2 – La codificación (de este tipo) que está especificada en la Recomendación X.209 permite el uso de bases 2, 8 ó 16 con una representación binaria de valores reales, y base 10 con una representación de caracteres. La elección es una opción de emisor.

16.3 El tipo real tiene un rótulo cuya clase es universal, número 9.

16.4 La notación para definir un valor de tipo real será “RealValue”:

RealValue ::= NumericRealValue | SpecialRealValue

NumericRealValue ::=

{Mantissa, Base, Exponent} | 0

Mantissa ::= SignedNumber

Base ::= 2 | 10

Exponent ::= SignedNumber

SpecialRealValue ::= PLUS-INFINITY | MINUS-INFINITY

Se utilizará la forma “0” para valores cero, y la forma alternada para “NumericRealValue” no será usada para valores cero.

17 Notación para el tipo bitstring (cadena de bits)

17.1 El tipo bistring (véase el § 3.19) será referenciado por “BitStringType”:

BitStringType ::=

BIT STRING |
BIT STRING{NamedBitList}

NamedBitList ::=

NamedBit |
NamedBitList, NamedBit

NamedBit ::=

identifier(number) |
identifier(DefinedValue)

17.2 La "NamedBitList" no es significativa en la definición de un tipo. Se usa solamente en la notación del valor especificado en el § 17.8.

17.3 El primer bit de una bitstring tiene el número cero. El bit final de una bistring es llamado *bit postrero*.

Nota – Esta terminología se usa al especificar la notación de valor y las reglas de codificación.

17.4 El "DefinedValue" será una referencia a un valor no negativo de tipo entero o enumerado o a un tipo derivado de éstos por rotulación o por subtipificación.

17.5 El valor de cada "number" o "DefinedValue" que aparece en la "NamedBitList" será diferente, y es el número de un bit diferenciado en un valor bitstring.

17.6 Cada "identifier" que aparece en la "NamedBitList" será diferente.

Nota – El orden de las secuencias de "NamedBit" en la "NamedBitList" no es significativo.

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

17.8 El valor de un tipo bitstring estará definido por la notación "BitStringValue":

BitStringValue ::=

bstring |
hstring |
{IdentifierList} |
{}

IdentifierList ::=

identifier |
IdentifierList,identifier

17.9 Cada "identifier" en un "BitStringType" será el mismo que un "identifier" en la secuencia "BitStringType" con la que el valor está asociado.

17.10 El usuario de la notación determina, y puede indicar mediante comentarios, si la presencia o ausencia del bit postrero es o no pertinente.

Nota – Las reglas de codificación permiten la transferencia de una cadena de bits con un patrón y una longitud arbitrarios.

17.11 Las notaciones "{IdentifierList}" y "{}" para "BitStringValue" no serán usadas si la presencia o ausencia del bit postrero es pertinente. Esta notación indica un valor bitstring con unos en las posiciones de bit especificadas por los números correspondientes a las secuencias "identifier" y con todos los demás bits cero.

Nota – La secuencia "{}" se utiliza para denotar un valor bitstring que no tiene ningún bit.

17.12 Al especificar las reglas de codificación para una bitstring, los bits serán referenciados por los términos *primer bit* y *bit postrero*, tal como se ha definido anteriormente.

17.13 Cuando se utiliza la notación "bstring", el *primer bit* es el de la izquierda, y el *bit postrero* el de la derecha.

17.14 Cuando se usa la notación "hstring", el bit más significativo de cada cifra hexadecimal corresponde al primer bit (el que está más a la izquierda) de la bitstring.

Nota – Esta notación no debe en ningún caso limitar la forma en que las reglas de codificación colocan las cadenas de bits en octetos para su transferencia.

17.15 No se utilizará la notación "hstring" a menos que:

a) el valor de bitstring consista en un múltiplo de cuatro bits; o

b) la presencia o ausencia del bit postrero cero no sea significativa,

Ejemplo:

'A98A'H

y

'1010100110001010'B

son notaciones alternativas para el mismo valor de bitstring.

18 Notación para el tipo octetstring (cadena de octetos)

18.1 El tipo octetstring (véase el § 3.20) será referenciado por la notación "OctetStringType":

OctetStringType ::= OCTET STRING

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

18.3 El valor de un tipo octetstring estará definido por la notación "OctetStringValue":

OctetStringValue ::=

bstring |
hstring

18.4 Al especificar las reglas de codificación para una octetstring, los octetos se referenciarán por los términos *primer octeto* y *octeto postrero*, y los bits dentro de un octeto están definidos por los términos *bits más significativo* y *bit menos significativo*.

18.5 Cuando se usa 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.

18.6 Cuando se usa la notación "hstring", la cifra hexadecimal más a la izquierda será el semiocteto más significativo del primer octeto. Si la "hstring" no tiene un número par de cifras hexadecimales, se interpretará como si contuviera una sola cifra trasera hexadecimal adicional.

19 Notación para el tipo null (nulo)

19.1 El tipo null (véase el § 3.21) se referenciará por la notación "NullType":

NullType ::= NULL

19.2 El rótulo para este tipo es de clase universal, número 5.

19.3 El valor del tipo nulo se referenciará por la notación "NullValue":

NullValue ::= NULL

20 Notación para tipos sequence (secuencia)

20.1 La notación para definir un tipo sequence (véase el § 3.22) a partir de otros tipos será "SequenceType":

SequenceType ::=

SEQUENCE{ElementTypeList} |
SEQUENCE{}

ElementTypeList ::=

ElementType |
ElementTypeList,ElementType

ElementType ::=

NamedType |
NamedType OPTIONAL |
NamedType DEFAULT Value |
COMPONENTS OF Type

20.2 El "Type" en la cuarta alternativa del "ElementType" será un tipo sequence. Se utilizará la notación "COMPONENTS OF Type" para definir la inclusión, en este punto de la "ElementTypeList", de todas las secuencias "ElementType" que aparezcan en el tipo referenciado.

Nota – Esta transformación se realiza lógicamente antes de la cumplimentación de los requisitos de las siguientes cláusulas.

20.3 Para cada serie de uno o más "ElementTypes" consecutivos marcados como OPTIONAL o DEFAULT, los rótulos de aquellos "ElementTypes" y de cualquier otro que sigue inmediatamente al "ElementType" serán distintos (véase el § 26).

20.4 Si están presentes "OPTIONAL" o "DEFAULT", puede omitirse el valor correspondiente a partir de un valor del tipo nuevo, y de la información transferida por las reglas de codificación.

Nota 1 – La notación de valor puede ser ambigua en este caso, a no ser que estén presentes en cada NamedType secuencias de "identifier".

Nota 2 – Las reglas de codificación aseguran que la codificación para un valor de sequence en el que un valor de elemento "DEFAULT" u "OPTIONAL" omitido sea el mismo que para un valor secuencia de un tipo en cuya definición de tipo se omitió el elemento correspondiente. Esta característica puede ser útil al definir subconjuntos.

20.5 Si aparece "DEFAULT", la omisión de un valor para ese tipo será exactamente equivalente a la inserción del valor definido por "Value", que será una especificación de valor que sea válida para el tipo definido por "Type" en la secuencia "NamedType".

20.6 El "identifier", de haberlo, en todas las secuencias de "NamedType" de la "ElementTypeList" será diferente.

20.7 Todos los tipos sequence tienen un rótulo de clase universal, número 16.

Nota – Los tipos sequence-of tienen el mismo rótulo (véase el § 21.3).

20.8 La notación para definir el valor de un tipo sequence será "SequenceValue":

SequenceValue ::=

{ElementValueList} |
{}

ElementValueList ::=

NamedValue |
ElementValueList,NamedValue

20.9 La notación "{}" se utilizará solamente si:

a) todas las secuencias "ElementType" del "SequenceType" van marcadas con "DEFAULT" u "OPTIONAL"; y se han omitido todos los valores; o

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

20.10 Habrá un "NamedValue" para cada "NamedType" en el "SequenceType" que no está marcado con OPTIONAL o DEFAULT y los valores estarán en el mismo orden que las correspondientes secuencias "NamedType".

Nota – La utilización de secuencias "NamedType" que no contengan un identificador no está prohibida, pero puede hacer que la notación de valor sea ambigua si se utiliza "OPTIONAL" o "DEFAULT".

21 Notación para tipos sequence-of (secuencia-de)

21.1 La notación para definir un tipo sequence-of (véase el § 3.23) a partir de otro tipo será el "SequenceOfType".

SequenceOfType ::=

SEQUENCE OF Type |
SEQUENCE

21.2 La notación "SEQUENCE" es sinónima de la notación "SEQUENCE OF ANY" (véase el § 27).

21.3 Todos los tipos sequence-of tienen un rótulo de clase universal, número 16.

Nota – Los tipos sequence tienen el mismo rótulo (véase el § 20.7).

21.4 La notación para definir un valor de un tipo sequence-of será "SequenceOfValue":

SequenceOfValue ::= {ValueList} | {}

ValueList ::=

Value |
ValueList,Value

La notación "{}" se utilizará cuando no hay valores componentes en el valor de sequence-of.

21.5 Cada secuencia "Value" de la "ValueList" tendrá la notación para un valor del "Type" especificado en el "SequenceOfType".

Nota – El significado semántico puede colocarse siguiendo el orden de estos valores.

22 Notación para tipos set (conjunto)

22.1 La notación para definir un tipo set (véase el § 3.24) de otros tipos será "SetType".

SetType ::=

SET{ElementTypeList} |
SET{}

"ElementTypeList" se especifica en el § 20.1.

22.2 El "Type" de la cuarta alternativa de "ElementType" (véase el § 20.1) será un tipo set. Se utilizará la notación "COMPONENTS OF Type" para definir la inclusión de todas las secuencias "ElementType" que aparezcan en el tipo referenciado.

Nota – Esta transformación se realiza lógicamente antes de la cumplimentación de los requisitos de las siguientes cláusulas.

22.3 Los tipos "ElementType" de un tipo set tendrán rótulos diferentes (véase el § 26).

22.4 Las subcláusulas § 20.4, § 20.5 y § 20.6 también se aplican a los tipos conjunto.

22.5 Todos los tipos set tienen un rótulo de clase universal, número 17.

Nota – Los tipos set-of tienen el mismo rótulo (véase el § 23.3).

22.6 No habrá semánticas asociadas con el orden de valores de un tipo set.

22.7 La notación para definir el valor de un tipo set será "SetValue":

SetValue ::= {ElementValueList} | {}

La "ElementValueList" está especificada en el § 20.8.

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

a) todas las secuencias "ElementType" del "SetType" van marcadas "DEFAULT" u "OPTIONAL", y se omiten todos los valores; o

b) la notación de tipo es "SET"{}".

22.9 Habrá un "NamedValue" para cada "NamedType" del "SetType" que no esté marcado por "OPTIONAL" o "DEFAULT".

Nota 1 – Estos "NamedValue" pueden aparecer en cualquier orden.

Nota 2 – El uso de secuencias "NamedType" que no contengan un identificador no está prohibido, pero pueden hacer que resulte ambigua la notación de valor.

23 Notación para tipos set-of (conjunto-de)

23.1 La notación para definir un tipo set-of (véase el § 3.25) a partir de otro tipo será "SetOfType":

SetOfType::= SET OF Type │
SET

23.2 La notación "SET" es sinónima de la notación "SET OF ANY" (véase el § 27).

23.3 Todos los tipos set-of tienen un rótulo que es de clase universal, número 17.

Nota – Los tipos set tienen el mismo rótulo (véase el § 22.5).

23.4 La notación para definir un valor de un tipo set-of será "SetOfValue".

SetOfValue::= {ValueList} │ {}

La "ValueList" se especifica en el § 21.4.

Se usa la notación "{}" cuando no hay valores componentes en los valores set-of.

23.5 Cada secuencia "Value" de la "ValueList" será la notación para un valor del "Type" especificado en el "SetOfType".

Nota 1 – El significado semántico no debería colocarse en el orden de estos valores.

Nota 2 – No se requieren reglas de codificación para preservar el orden de estos valores.

24 Notación para tipos choice (elección)

24.1 La notación para definir un tipo choice (véase el § 3.27) a partir de otros tipos será "ChoiceType":

ChoiceType ::= CHOICE{AlternativeTypeList}

AlternativeTypeList ::=

NamedType │
AlternativeTypeList,NamedType

Nota 1 – Las reglas de codificación codifican la alternativa elegida de forma que no es distinguible a partir de un "Type" consistente solamente en el "Type" contenido en esa alternativa.

Nota 2 – La especificación de un "ChoiceType" con un único "NamedType" en la "AlternativeTypeList" no puede distinguirse en ninguna codificación de un valor del uso directo del "Type" en el "NamedType".

24.2 Los tipos definidos en la "AlternativeTypeList" tendrán todos rótulos diferentes (véase el § 26).

24.3 El rótulo del tipo elección se considerará variable. Cuando se selecciona un valor, el rótulo se hace igual al rótulo del "Type" en el "NamedType" en la "AlternativeTypeList" de donde se toma el valor.

24.4 Cuando se utilice este tipo en un lugar donde esta Recomendación requiere la utilización de tipos con rótulos diferentes (véanse los § 20.3, 22.3 y 24.2) los rótulos de todos los tipos definidos en la "AlternativeTypeList" diferirán de otros tipos (véase el § 26). Los siguientes ejemplos ilustran este requisito. Los ejemplos 1 y 2 son usos correctos de la notación. El ejemplo 3 es incorrecto, ya que los rótulos para tipos d y f, y e y g son idénticos.

Ejemplo 1:

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

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

Ejemplo 2:

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

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

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

Ejemplo 3:

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

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

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

24.5 El "identifier", (si lo hay), en todas las secuencias "NamedType" de la "AlternativeTypeList" será diferente.

24.6 Cuando se utilice este tipo en un lugar donde esta Recomendación requiere el uso de "NamedTypes" con distintos "identifiers", los "identifiers" (de haberlos) de todos los "NamedTypes" en la "AlternativeTypeList" diferirán de los correspondientes a los demás "NamedTypes" (si los hay).

24.7 La notación para definir el valor de un tipo choice será "ChoiceValue":

ChoiceValue ::= NamedValue

24.8 Si el "NamedValue" contiene un "identifier", será una notación para un valor del tipo en la "AlternativeTypeList" que está denominado por el mismo "identifier". Si el "NamedValue" no contiene un "identifier", será una notación para un valor de uno de los tipos en la "AlternativeTypeList" que no están denominados por un "identifier".

Nota – La omisión de un "identifier" en el "NamedType" puede hacer ambigua la notación de valor.

25 Notación para tipos selection (selección)

25.1 Un “NamedType” que aparece en la “AlternativeTypeList” de un “ChoiceType” puede ser referenciado por la notación “SelectionType”:

SelectionType ::= identifier<Type

donde “Type” es una notación para referenciar el “ChoiceType” y el “identifier” en el “NamedType”.

Nota – Puede utilizarse “SelectionType” como un “NamedType”, en cuyo caso el “identifier” se utiliza en la notación de valor, o como un “Type” dentro de un “NamedType”, en cuyo caso su “identifier” no se utiliza.

25.2 La notación para un valor de un tipo selection será “SelectionValue”:

SelectionValue ::= NamedValue

donde el “NamedValue” contiene el identificador que aparece en el correspondiente “SelectionType” si el “SelectionType” se utiliza como un “NamedType”, pero no de otra forma.

26 Notación para tipos tagged (rotulados)

Un tipo tagged (véase el § 3.26) es un tipo nuevo que es isomorfo de un tipo antiguo pero que tiene un rótulo diferente. En todos los esquemas de codificación un valor del tipo nuevo podrá distinguirse de un valor del tipo antiguo. El tipo tagged, se utiliza principalmente donde esta Recomendación requiere el uso de tipos con rótulos distintos. (Véanse los § 20.3, 22.3, 24.2, 24.4 y 27.6.)

Nota – Cuando un protocolo determine de entre varios tipos de datos, los valores que pueden transmitirse en cualquier momento, pueden necesitarse diferentes rótulos para permitir al receptor decodificar el valor correctamente.

26.1 La notación para un tipo tagged será “TaggedType”:

TaggedType ::=

Tag Type |
Tag IMPLICIT Type |
Tag EXPLICIT Type

Tag ::= [Class ClassNumber]

Class Number ::=

number |
DefinedValue

Class ::=

UNIVERSAL |
APPLICATION |
PRIVATE |
empty

26.2 El “DefinedValue” será una referencia a un valor no negativo de tipo entero, o de un tipo derivado de un tipo entero por rotulación.

26.3 El nuevo tipo es isomorfo del tipo antiguo, pero tiene un rótulo de clase “Class” y número “ClassNumber”, a no ser que “Class” esté “empty”, cuando el rótulo es de la clase específica de contexto, número “ClassNumber”.

26.4 La “Class” no será “UNIVERSAL” salvo para los tipos definidos en esta Recomendación.

Nota – El uso de rótulos de la clase universal se aprueba cada cierto tiempo por la ISO y el CCITT.

26.5 Si la “Class” es “APPLICATION” no se utilizará de nuevo el mismo “Tag” en el mismo módulo.

26.6 Si la “Class” es “PRIVATE” el “Tag” está disponible para su uso sobre una base específica de la aplicación.

26.7 La construcción rotulación especifica la rotulación explícita si se dan 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”;

c) se utiliza la alternativa “Tag Type” y el valor “TagDefault” para el módulo es “IMPLICIT TAGS”, pero el tipo definido en el “Type” es un tipo choice o un tipo any.

La construcción de rotulación especifica por otra parte la rotulación implícita.

26.8 Si la “Class” es “vacía” (empty), no hay otras restricciones en el uso de “Tag”, que las implicadas por los requisitos para los distintos rótulos en los § 20.3, 22.3 y 24.2.

26.9 La rotulación implícita indica, para aquellas reglas de codificación que proporcionan la opción que la identificación explícita del rótulo del “Type” en el “TaggedType” no se necesita durante la transferencia.

Nota – Puede ser útil retener el rótulo antiguo cuando éste sea de clase universal y por lo tanto identifique sin ambigüedad el tipo antiguo sin conocer la definición NSA.1 del tipo nuevo. La mínima transferencia de octetos es sin embargo, realizada normalmente con el uso de IMPLICIT. Un ejemplo de una codificación utilizando IMPLICIT se da en la Recomendación X.209.

26.10 No se utilizará la alternativa “IMPLICIT” si el tipo definido por “Type” es un tipo choice o un tipo any.

26.11 La notación para un valor de un “TaggedType” será “TaggedValue”:

TaggedValue ::= Value

donde “Value” es la notación para un valor del “Type” en el “Tagged Type”.

El “Tag” no aparece en esta notación.

27 Notación para el tipo any (cualquiera)

27.1 La notación para un tipo any (véase el § 3.29) es “AnyType”:

AnyType ::=

ANY |
ANY DEFINED BY identifier

Nota – El uso de “ANY” en una norma de la ISO o en una Recomendación del CCITT da una especificación incompleta a menos que se proporcionen especificaciones adicionales. La construcción “ANY DEFINED BY” proporciona los medios para especificar en una instancia de comunicación el tipo que rellena ANY y señala su semántica. Si se siguen las reglas siguientes para su uso, se puede obtener una especificación completa. No se aconseja el uso de ANY sin la construcción DEFINED BY.

27.2 La alternativa “DEFINED BY” se utilizará solamente cuando el tipo any o un tipo derivado de él por rotulación sea uno de los tipos componentes de un tipo sequence o un tipo set (tipo continente).

27.3 El “identifier” en la alternativa “DEFINED BY” aparecerá también en un “NamedType” que especifica a otro componente no opcional del tipo continente. El “NamedType” será un tipo integer o un tipo enumerated, o un tipo object identifier derivado de dichos tipos por rotulación o subtipificación.

27.4 Cuando el “NamedType” es un tipo integer o enumerated, o un tipo derivado de dichos tipos por rotulación o subtipificación, el documento que emplee la notación “DEFINED BY” contendrá, o hará referencia explícitamente, a una lista única que especifique el tipo NSA.1 cursado por el ANY para cada valor permitido del tipo integer. Habrá precisamente una de tales listas en todas las situaciones de comunicación del tipo continente.

27.5 Cuando el “NamedType” es un tipo identificador de objeto, o un tipo derivado de identificador de objeto por rotulación, se necesitan registros que, para cada valor identificador de objeto asignado, asocien un tipo único NSA.1 (que puede ser un tipo CHOICE) que vaya a ser cursado por el ANY.

Nota 1 – Puede haber un número arbitrario de registros que asocien un valor identificador de objeto con un tipo NSA.1 para este fin.

Nota 2 – Se espera que en el marco de las Normas de la ISO y las Recomendaciones del CCITT se efectúe el registro de valores para interconexión de sistemas abiertos usando la notación. Cuando una Autoridad de Registro Internacional diferente sea utilizada por cualquier instancia de "ANY DEFINED BY", esto debería indicarse en el documento que utiliza la notación.

Nota 3 – La diferencia principal entre las definiciones de un entero y de un identificador de objeto es que el uso del entero hace referencia a una lista única contenida en la norma o Recomendación que la utiliza, mientras que el uso del identificador de objeto permite un conjunto abierto de tipos determinados por cualquier autoridad capaz de asignar identificadores de objeto.

27.6 Este tipo tiene un rótulo indeterminado y no se utilizará cuando esta Recomendación requiera distintos rótulos. (Véanse los § 20.3, 22.3, 24.2 y 24.4)

27.7 La notación para el valor de un tipo cualquiera se definirá usando NSA.1, y es "AnyValue":

AnyValue::= Type Value

donde "Type" es la notación para el tipo elegido y "Value" es la notación para un valor de este tipo.

28 Notación para el tipo object identifier (identificador de objeto)

28.1 El tipo object identifier (véase el § 3.34) se referenciará por la notación "ObjectIdentifierType":

```
ObjectIdentifierType::=
        OBJECT IDENTIFIER
```

28.2 Este tipo tiene un rótulo de clase universal, número 6.

28.3 La notación de valor para un object identifier será "ObjectIdentifierValue":

```
ObjectIdentifierValue ::=
        {ObjIdComponentList} |
        {DefinedValue ObjIdComponentList}
ObjIdComponentList ::=
        ObjIdComponent |
        ObjIdComponent ObjIdComponentList
ObjIdComponent ::=
        NameForm |
        NumberForm |
        NameAndNumberForm
NameForm ::= identifier
NumberForm ::= number | DefinedValue
NameAndNumberForm ::=
        identifier(NumberForm)
```

28.4 El "DefinedValue" en "NumberForm" será una referencia a un valor de tipo entero o enumerado, o de un tipo derivado de éstos por rotulación o subtipificación.

28.5 El "DefinedValue" en "ObjectIdentifierValue" será una referencia a un valor de tipo identificador de objeto o de un tipo derivado de identificador de objeto por rotulación.

28.6 La "NameForm" se utilizará solamente para los componentes de identificador de objeto cuyo valor numérico e identificador están especificados en los anexos B a D, y serán uno de los identificadores especificados en los anexos B a D.

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

28.8 Se especificará el “identifier” en el “NameAndNumberForm” cuando se asigne un valor numérico al componente de identificador de objeto.

Nota – Las autoridades que asignan valores numéricos a componentes de identificador de objeto figuran en los anexos a esta Recomendación.

28.9 La semántica de un valor de object identifier se define por un *object identifier tree* (árbol de identificadores de objeto). Un árbol identificador de objeto es un árbol cuya raíz corresponde a esta Recomendación y cuyos vértices corresponden a autoridades administrativas responsables de la asignación de arcos desde este vértice. Cada arco del árbol está rotulado por un componente de identificador de objeto que es un valor numérico. A cada objeto de información se le asigna un vértice (normalmente una hoja) para ser identificado, y ningún otro objeto de información (del mismo tipo o de uno diferente) es asignado a ese mismo vértice. Así pues, un objeto de información está unívoca e inequívocamente identificado por la secuencia de valores numéricos (componentes de identificador de objeto) que rotulan los arcos del trayecto que va desde la raíz al vértice asignado al objeto de información.

Nota – Los valores de identificador de objeto contienen al menos dos componentes de identificador de objeto, tal como se especifica en los anexos B a D.

28.10 Semánticamente un valor de identificador de objeto es una lista ordenada de valores componentes de identificador de objeto. Empezando con la raíz del árbol de identificadores de objeto, cada valor de componente de identificador de objeto identifica un arco del árbol de identificador de objeto. El último valor de componente de identificador de objeto identifica un arco que conduce a un vértice al que se ha asignado un objeto de información. Es este objeto de información el que es identificado por el valor de identificador de objeto. La parte significativa del componente de identificador de objeto es la “NameForm” o la “NumberForm” a la que es consignado y que proporciona el valor numérico al componente de identificador de objeto.

Nota – En general, un objeto de información es una clase de información (por ejemplo el formato de un fichero) más que un elemento de tal clase (por ejemplo un fichero individual). Es pues la clase de información (definida por alguna especificación referenciable), antes que la información en sí misma, a la que se asigna un lugar en el árbol.

28.11 Donde el “ObjectIdentifierValue” incluya un “DefinedValue”, la lista de componentes de identificador de objeto a la que se refiere, se antepone a los componentes explícitamente presentes en el valor.

Ejemplos: Con identificadores asignados según se especifica en el anexo B, los valores:

{norma ISO 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)}

Nota – Se recomienda que, cuando una Recomendación del CCITT, o una Norma de la ISO u otro documento asigne valores de tipo OBJECT IDENTIFIER a objetos de información, debería haber un apéndice o anexo que resuma las asignaciones hechas en el mismo. Se recomienda también que, una autoridad que asigne valores de tipo OBJECT IDENTIFIER a un objeto de información, asigne también valores de tipo ObjectDescriptor a ese objeto de información.

29 Notación para tipos characterstring (cadena de caracteres)

29.1 La notación para referenciar un tipo characterstring (véanse el § 3.12 y sección 2) será:

CharacterStringType::= typereference

donde “typereference” es uno de los nombres de tipo characterstring enumerados en la sección 2.

29.2 El rótulo de cada tipo characterstring está especificado en la sección 2.

29.3 La notación para un valor characterstring será:

CharacterStringValue::= cstring

La definición del tipo characterstring determina los caracteres que aparecen en la “cstring”.

30 Notación para tipos definidos en la sección 3

30.1 La notación para referenciar un tipo definido en la sección 3 de esta Recomendación será:

UsefulType::= typereference

donde “typereference” es una de las definidas en la sección 3 utilizando la notación de la NSA.1.

30.2 El rótulo de cada “UsefulType” se especifica en la sección 3.

30.3 La notación para un valor de un “UsefulType” se especifica en la sección 3.

SECCIÓN 2 – TIPOS CHARACTER STRING (CADENA DE CARACTERES)

31 Definición de tipos character string (cadena de caracteres)

Esta cláusula define tipos cuyos valores diferenciados son secuencias de cero, uno o más caracteres de un juego de caracteres.

31.1 El tipo se define especificando:

a) el rótulo asignado al tipo;

b) un nombre mediante el que la definición del tipo pueda ser referenciada;

c) los caracteres del juego de caracteres usados al definir el tipo, bien por referencia a un cuadro que contenga los caracteres gráficos o por referencia a un número de registro en el Registro internacional ISO de juegos de caracteres codificados para ser utilizados con secuencias de escape.

El nombre citado en el apartado b) anterior puede utilizarse como una “Typereference” en la notación NSA.1 (véase el § 29).

31.2 El cuadro 6/X.208 enumera los nombres con los que cada una de estas definiciones tipo pueden ser referenciadas, el número del rótulo de clase universal asignado al tipo, la definición de los números de registro o el cuadro y cuando sea necesaria la identificación de una NOTA relativa a la entrada del cuadro. Cuando se define un sinónimo en la notación, éste se indica entre paréntesis.

Nota – El rótulo asignado a tipos character string identifica el tipo sin ambigüedad. Nótese, sin embargo, que si se utiliza NSA.1 para definir nuevos tipos de este tipo (particularmente al utilizar IMPLICIT), puede ser imposible reconocer estos tipos si no se conoce la definición de tipo de NSA.1.

31.3 El cuadro 4/X.208 indica los caracteres que pueden aparecer en el tipo NumericString.

CUADRO 4/X.208

NumericString

Nombre	Símbolo
Cifras	0, 1, ...9
Espacio	(espacio)

CUADRO 5/X.208

PrintableString

Nombres	Símbolo
Letras mayúsculas	A, B, . . . Z
Letras minúsculas	a, b, . . . z
Cifras	0, 1, . . . 9
Espacio	(espacio)
Apóstrofe	’
Paréntesis izquierdo	(
Paréntesis derecho	)
Signo más	+
Coma	,
Guión	-
Punto	.
Barra de fracción	/
Dos puntos	:
Signo igual	=
Signo de interrogación	?

CUADRO 6/X.208

Lista de tipos cadena de caracteres

Nombre para referenciar el tipo	Número de clase universal	Números de registro (véase ISO 2375) o número del cuadro que proporciona la definición	Notas
NumericString	18	cuadro 4	1
PrintableString	19	cuadro 5	1
TeletexString (CadenaT61)	20	87, 102, 103, 106, 107 + ESPACIO + BORRADO	2
VideotexString	21	1, 72, 73, 102, 108, 128, 129 + ESPACIO + BORRADO	3
VisibleString (CadenaISO646)	26	2 + ESPACIO	
IA5String	22	1, 2 + ESPACIO + BORRADO	
GraphicString	25	Todos los conjuntos G + ESPACIO	
GeneralString	27	Todos los conjuntos G y todos los C + ESPACIO + BORRADO	

Nota 1 – El estilo tipográfico, dimensión, color, intensidad, u otras características de visualización no son significativos.

Nota 2 – Las entradas que corresponden a estos números de registro hacen referencia a la Recomendación T.61 del CCITT para reglas relativas a su utilización.

Nota 3 – Las entradas correspondientes a estos números de registro dan la funcionalidad de las Recomendaciones T.100 y T.101 del CCITT.

31.4 El cuadro 5/X.208 enumera los caracteres que pueden aparecer en el TipoCadenaImprimible.

31.5 La notación para estos tipos será "cstring".

Nota – Esta notación sólo puede utilizarse con un medio capaz de visualizar los caracteres que están presentes en el valor. La notación para el valor en otros casos no está definida.

31.6 En todos los casos, la gama de caracteres permitidos puede ser restringida por un comentario, pero no será ampliada.

SECCIÓN 3 – DEFINICIONES ÚTILES

Esta sección contiene definiciones que se espera sean útiles en un cierto número de aplicaciones.

Nota – Se espera que esta sección será ampliada para incluir otros tipos de datos comunes tales como diagnósticos, información de autentificación, información para contabilidad, parámetros de seguridad y otros.

La notación valor y definición semántica para tipos definidos en esta sección se derivan de una definición de tipo utilizando la notación NSA.1. Esta definición de tipo puede referenciarse por normas o Recomendaciones que definen reglas de codificación con el fin de especificar codificaciones para estos tipos.

32 Generalized Time (Tiempo generalizado)

32.1 Este tipo se referenciará por el nombre

GeneralizedTime

32.2 El tipo consta de valores que representan

a) una fecha, tal como se define en ISO 2014 (Escritura de fechas de forma totalmente numérica);

b) una hora, para cualquiera de las precisiones definidas en la cláusula 2 de la norma ISO 3307 (Representación de la hora); y

c) el factor diferencial de la hora local en la forma definida en ISO 4031 (Representación de diferenciales de la hora local).

32.3 El tipo puede ser definido utilizando la NSA.1 de la forma siguiente:

GeneralizedTime::=

[UNIVERSAL 24] IMPLICIT
VisibleString

con los valores de "VisibleString" limitados a cadenas de caracteres que son:

a) una cadena que representa la fecha, según se especifica en ISO 2014, con una representación del año con cuatro cifras, una representación del mes con dos cifras, y una representación del día con dos cifras, sin utilizar separadores, seguida de una cadena que representa la hora tal como se especifica en ISO 3307, sin otros separadores que una coma decimal o un punto decimal (tal como está previsto en las cláusulas 2.3, 2.4 y 2.5 de ISO 3307) y sin terminar en Z (tal como se estipula en la cláusula 3 de ISO 3307); o

b) los caracteres del apartado anterior seguidos de la letra mayúscula Z; o

c) los caracteres del apartado anterior a) seguidos por una cadena que representa un diferencial de la hora local, tal como se especifica en ISO 4031, sin separadores.

En el caso a) la hora representará la hora local. En el caso b) la hora representará la hora UTC. 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 UTC de la siguiente forma:

la hora UTC es $t_1 - t_2$

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 antes.

Caso c) 19851106210627.3-0500
hora local como en el ejemplo a)
con la hora local retrasada 5 horas
con relación a la hora UTC.

32.4 El rótulo será el definido en el § 32.3.

32.5 La notación de valor será la notación de valor para la "VisibleString" definida en el § 32.3.

33 Tiempo universal

33.1 Este tipo se designará con el nombre

UTCtime

33.2 El tipo consiste en valores que representa:

a) una fecha;

b) una hora con una precisión de un minuto o un segundo; y

c) (opcionalmente) un diferencial de hora local con respecto al tiempo universal coordinado.

33.3 Puede definirse el tipo, utilizando la NSA.1, de la siguiente manera:

UTCTime::=

[UNIVERSAL 23] IMPLICIT
VisibleString

con los valores de la "VisibleString" restringidos a las cadenas de caracteres que son la yuxtaposición de:

a) las seis cifras YYMMDD donde YY son las dos cifras finales del año cristiano, MM es el mes (contando enero como 01), y DD es el día del mes (01 a 31);

b) o bien

1) las cuatro cifras hhmm donde hh es la hora (00 a 23) y mm son los minutos (00 a 59); o

2) las seis cifras hhmmss donde hh y mm son como en 1), y ss son los segundos (00 a 59); y

c) o bien

1) el carácter Z; o

2) uno de los caracteres + o –, seguidos de hhmm, donde hh es la hora y mm son los minutos.

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

En la alternativa c) 1), la hora es tiempo UTC. En la alternativa c) 2) el tiempo (t_1) especificado en a) y b) es la hora local; el diferencial de hora (t_2) especificado en c) 2) permite expresar el tiempo UTC de la forma siguiente:

el tiempo UTC es $t_1 - t_2$

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

TiempoUTC "8201021200Z"
TiempoUTC "8201020700-0500"

33.4 El rótulo será como se define en el § 33.3.

33.5 La notación de valor será la notación de valor para la "VisibleString" definida en el § 33.3.

34 Tipo external (externo)

34.1 La notación para un tipo external (véase el § 3.30) es "ExternalType":

ExternalType ::= EXTERNAL

34.2 El tipo consta de valores que representan:

a) una codificación de un valor de datos único que puede, pero no necesariamente, ser un valor de un datatype NSA.1 único; y

b) información de identificación que determina la semántica y las reglas de codificación; y

c) (opcionalmente) un descriptor objeto que describe el objeto.

El descriptor de objeto opcional no estará presente a no ser que sea permitido explícitamente por un comentario asociado con el uso de notación EXTERNAL.

34.3 El tipo EXTERNAL permite la inclusión de cualquier valor de datos de un conjunto identificador de valores de datos.

Nota 1 – La especificación de este conjunto de valores de datos, sus semánticas, la asignación de un identificador de objeto y (opcionalmente) un descriptor de objeto, y la distribución de esta información a todas las partes en comunicación se llama un *registro de sintaxis abstracta*. Esta operación puede ser realizada por cualquier autoridad con derecho a atribuir un valor de OBJECT IDENTIFIER, como se especifica en los anexos B a D.

Nota 2 – Un conjunto de valores de datos registrados como sintaxis abstracta (con reglas de codificación asociadas) no está bien-configurado a no ser que la codificación de cada valor de datos sea autoidentificadora dentro del conjunto de codificaciones de valor de datos. Cuando se utiliza la NSA.1 para definir una sintaxis abstracta, se usa la rotulación para proporcionar una autoidentificación. Cuando una sintaxis abstracta no está bien configurada, la utilización del canal de comunicaciones es dependiente del contexto o conduce a la ambigüedad.

34.4 El tipo EXTERNAL se puede definir, utilizando la NSA.1, de la siguiente manera:

```
EXTERNAL ::= [UNIVERSAL 8] IMPLICIT SEQUENCE

{direct-reference          OBJECT IDENTIFIER OPTIONAL,
 indirect-reference        INTEGER OPTIONAL,
 data-value-descriptor     ObjectDescriptor OPTIONAL,
 encoding                  CHOICE
    {single-ASN1-type      [0] ANY,
     octet-aligned         [1] IMPLICIT OCTET STRING,
     arbitrary             [2] IMPLICIT BIT STRING}}
```

34.5 Cuando no se utiliza la negociación de la capa de presentación de las reglas de codificación (por acuerdo previo de sintaxis de transferencia) para el valor de este EXTERNAL, estará presente el "direct-reference OBJECT IDENTIFIER". En este caso el identificador del conjunto de valores de datos es un *identificador de objeto que referencia* directamente a una sintaxis abstracta y rellena el campo del "direct-reference OBJECT IDENTIFIER" del "EXTERNAL". En este caso, el registro de sintaxis abstracta define también las reglas de codificación (sintaxis de transferencia) para el valor de datos y no se incluirá el "indirect-reference INTEGER".

34.6 Cuando se utiliza la negociación de la capa de presentación para el valor de este EXTERNAL, estará presente el "indirect-reference INTEGER". En este caso el identificador del conjunto de valores de datos es un entero que hace referencia a un caso de uso de una sintaxis abstracta. El entero se llama identificador de contexto de presentación y rellena el campo del "indirect-reference INTEGER" del "EXTERNAL". Si la negociación de la capa de presentación ha sido completada, el identificador de contexto de presentación también identifica las reglas de codificación (sintaxis de transferencia) para el valor de datos y el "direct-reference OBJECT IDENTIFIER" no estará incluido. Si la negociación de la capa de presentación no ha sido completada, se necesita también un valor de identificador de objeto que identifica las reglas de codificación (sintaxis de transferencia) usadas para la codificación. Cuando la negociación de la capa de presentación está en curso y cuando el elemento "direct-reference OBJECT IDENTIFIER" sea permitido para cursar tal valor, éste será identificado por un comentario asociado al uso de la notación "EXTERNAL", en otro caso el campo estará ausente.

Nota 1 – El efecto de lo indicado en los § 34.5 y 34.6 es hacer obligatoria la presencia de al menos una de las "direct-reference" e "indirect-reference".

Nota 2 – Ambas referencias están presentes durante la negociación de la capa de presentación mientras no haya concluido.

34.7 Si el valor de datos es el valor de un tipo de datos NSA.1 único, y si las reglas de codificación para este valor de datos son las mismas que para el tipo de datos "EXTERNAL" completo, la realización del envío se hará a través de cualquiera de las elecciones "Encoding":

single-ASN1-type
octet-aligned
arbitrary

como una opción de realización.

34.8 Si la codificación de los valores de datos, según la codificación negociada o acordada, es un número entero de octetos, la realización del envío utilizará cualquiera de las elecciones de "Encoding":

octet-aligned
arbitrary

como una opción de realización.

Nota – Un valor de datos, que es una serie de tipos NSA.1, y para el que la sintaxis de transferencia especifica una simple concatenación de las cadenas de octetos producidas al aplicar las reglas de codificación básicas NSA.1, es de esta categoría, y no de la del § 34.7.

34.9 Si la codificación del valor de datos, según la codificación acordada o negociada, no es un número entero de octetos, la elección "Encoding" será

arbitrary

34.10 Si la elección "Encoding" se hace como "single-NSA.1-type", el tipo NSA.1 reemplazará al "ANY", con un valor igual al valor de datos que se va a codificar.

Nota – La gama de valores que puede darse en un "ANY" está determinada por el registro del valor de identificador de objeto asociado a la "direct-reference", y/o al valor entero asociado a la "indirect-reference".

34.11 Si la elección "Encoding" se hace como "octet-aligned", el valor de datos estará codificado según la sintaxis de transferencia negociada o acordada, y los octetos resultantes formarán el valor de la octetstring.

34.12 Si la elección "Encoding" se hace como "arbitrary", el valor de datos será codificado según la sintaxis de transferencia negociada o acordada y el resultado formará un valor de la bitstring.

34.13 El rótulo será el definido en el § 34.4.

34.14 La notación de valor será la notación de valor del tipo definido en el § 34.4.

35 Tipo object descriptor (descriptor de objeto)

35.1 Este tipo se referenciará por el nombre

ObjectDescriptor

35.2 El tipo consiste en un texto legible por el hombre que sirve para describir un objeto de información. El texto no es una identificación inequívoca del objeto de información, si bien deberá ser poco habitual un mismo texto para objetos de información diferentes.

Nota – Se recomienda que una autoridad que asigne valores del tipo "OBJECT IDENTIFIER" a un objeto de información deberá también asignar valores del tipo "ObjectDescriptor" a ese objeto de información.

35.3 El tipo puede definirse, utilizando la notación NSA.1, de la forma siguiente:

```
ObjectDescriptor ::=
        [UNIVERSAL 7] IMPLICIT
        GraphicString
```

La "GraphicString" contiene el texto que describe el objeto de información.

35.4 El rótulo será el definido en el § 35.3.

35.5 La notación de valor será la notación de valor para la "GraphicString" definida en el § 37.3.

SECCIÓN 4 – SUBTIPOS

36 Notación de subtipo

36.1 Un subtipo está definido por la notación por un tipo progenitor seguido de una especificación adecuada del subtipo. La notación de especificación del subtipo está compuesta de valores de conjuntos de subtipo. Los valores del subtipo se determinan según se especifica en el § 36.7, tomando la unión de todos los conjuntos de valores de subtipo.

36.2 La notación de subtipo no se utilizará para producir un subtipo sin valores.

36.3 La notación para un subtipo será "Subtype":

```
Subtype ::=

        ParentType SubtypeSpec |
        SET SizeConstraint OF Type |
        SEQUENCE SizeConstraint OF Type

ParentType ::= Type
```

36.4 Cuando la notación "SubtypeSpec" sigue a la notación "SelectionType", el tipo progenitor es el "SelectionType", no el "Type" en la notación "SelectionType".

36.5 Cuando la notación "SubtypeSpec" sigue a una notación tipo sequence-of o set-of, se aplica al "Type" en la notación sequence-of o set-of, no al tipo sequence-of o set-of.

Nota – Se utiliza la notación especial "SET SizeConstraint OF" y "SEQUENCE SizeConstraint OF" para proporcionar un mecanismo alternativo (que es más legible que la notación del caso general) para casos simples. Casos más complejos requieren el mecanismo general.

36.6 La notación para la especificación de subtipo será “SubtypeSpec”:

SubtypeSpec ::=

(SubtypeValueSet SubtypeValueSetList)

SubtypeList ::=

“ | ”
SubtypeValueSet
SubtypeValueSetList |
empty

36.7 Cada “SubtypeValueSet” especifica un número (con la posibilidad del cero) de valores del tipo progenitor, que están incluidos en el subtipo. Un valor del tipo progenitor es un valor del subtipo si y solamente si pertenece a uno o más conjuntos del valor subtipo. El subtipo está formado por el conjunto unión de valores que pertenecen a conjuntos de valores de subtipo.

36.8 Se proporciona un cierto número de diferentes formas de notación para “SubtypeValueSet” y “SubtypeConstraint”, que se identifican a continuación. Su sintaxis y su semántica se definen en el § 37. Como se especifica en el § 37, y se resume en el cuadro 7/X.208, algunas notaciones sólo pueden aplicarse a tipos progenitores particulares.

CUADRO 7/X.208

Aplicabilidad de conjuntos de valores de subtipos

Tipo (o derivado del mismo por rotulación	Valor único	Subtipo contenido	Gama de valores	Gama de tamaño	Limitaciones alfabeto	Subtipificación interna
Boolean	/	/	x	x	x	x
Integer	/	/	/	x	x	x
Enumerated	/	/	x	x	x	x
Real	/	/	/	x	x	x
Object Identifier	/	/	x	x	x	x
Bit String	/	/	x	/	x	x
Octet String	/	/	x	/	x	x
CharacterString Types	/	/	x	/	/	x
Sequence	/	/	x	x	x	/
Sequence-of	/	/	x	/	x	/
Set	/	/	x	x	x	/
Set-of	/	/	x	/	x	/
Any	/	/	x	x	x	x
Choice	/	/	x	x	x	/

Subtype ValueSet ::=

SingleValue |
ContainedSubtype |
ValueRange |
PermittedAlphabet | SizeConstraint | InnerTypeConstraints

37 Conjuntos de valores de subtipo

37.1 *Single Value* (Valor único)

37.1.1 La notación "SingleValue" será:

SingleValue ::= Value

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

37.1.2 Un conjunto de valores "SingleValue" es el valor único del tipo progenitor especificado por "Value". Esta notación puede aplicarse a todos los tipos progenitores.

37.2 *Contained Subtype (Subtipo contenido)*

37.2.1 La notación "ContainedSubtype" será:

ContainedSubtype ::= INCLUDES Type

37.2.2 Un conjunto de valores "ContainedSubtype" consta de todos los valores del "Type" que ha de ser en sí mismo un subtipo del tipo progenitor. Esta notación puede ser aplicada a todos los tipos progenitores.

37.3 *Value Range (Gama de valores)*

37.3.1 La notación "ValueRange" será:

ValueRange ::= LowerEndpoint .. UpperEndpoint

37.3.2 Un conjunto de valores "ValueRange" consta de todos los valores en una gama de valores designados especificando los valores numéricos de los puntos extremos de la gama. Esta notación sólo puede aplicarse a tipos enteros, tipos reales y tipos derivados de dichos tipos por rotulación o subtipificación.

Nota – Con fines de subtipificación "PLUS-INFINITY" excede a todos los valores "NumericReal" y "MINUS-INFINITY" es menor que todos los valores "NumericReal".

37.3.3 Cada extremo final de la gama es cerrado (en cuyo caso ese punto extremo está incluido en el conjunto de valores) o abierto (en cuyo caso el punto extremo no está incluido). Cuando es abierto, la especificación del extremo final incluye un símbolo menor que ("<"):

LowerEndpoint ::= LowerEndValue | LowerEndValue <

UpperEndpoint ::= UpperEndValue | < UpperEndValue

37.3.4 Un punto extremo puede estar también sin especificar, en cuyo caso la gama se extiende en esta dirección tanto como el tipo progenitor lo permita:

LowerEndValue ::= Value | MIN

UpperEndValue ::= Value | MAX

37.4 *Size Constraint (Limitación de tamaño)*

37.4.1 La notación "SizeConstraint" será:

SizeConstraint ::= SIZE SubtypeSpec

37.4.2 Una "SizeConstraint" sólo puede aplicarse a tipos bitstring, tipos octetstring, tipos characterstring, tipos set-of o tipos sequence-of, o tipos formados a partir de cualquiera de estos tipos por rotulación o subtipificación.

37.4.3 El "SubtypeSpec" especifica los valores enteros permitidos por la longitud de los números del conjunto de valores, y adopta la forma de cualquier especificación de subtipo que pueda ser aplicada al tipo progenitor siguiente:

INTEGER (0. .MAX)

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

Type	**Unit of measure**
bit string	bit
octet string	octet
character string	character
set-of	component value
sequence of	component value

37.5 *Permitted Alphabet*

37.5.1 La notación "PermittedAlphabet" será:

PermittedAlphabet ::= FROM SubtypeSpec

37.5.2 Un conjunto de valores "PermittedAlphabet" consta de todos los valores que pueden construirse utilizando un sub-alfabeto de la cadena progenitora. Esta notación sólo puede aplicarse a los tipos cadena de caracteres, o a los tipos formados a partir de ellos por rotulación o subtipificación.

37.5.3 El "SubtypeSpec" especifica los caracteres que pueden aparecer en la cadena de caracteres, y es una especificación de subtipo cualquiera que puede aplicarse al subtipo obtenido por la aplicación de la especificación de subtipo "SIZE(1)" al tipo progenitor.

37.6 *Subtipificación interna*

37.6.1 La notación para "InnerTypeConstraints" será:

InnerTypeConstraints ::=

WITH COMPONENT SingleTypeConstraint │
WITH COMPONENTS MultipleTypeConstraints

37.6.2 Una "InnerTypeConstraints" incluye en el conjunto de valores sólo aquellos valores que satisfagan una colección de restricciones sobre la presencia y/o los valores componentes del tipo progenitor. Un valor del tipo progenitor no pertenecerá a un subtipo a no ser que satisfaga todas las restricciones expresadas o implicadas (véase el § 37.6.6). Esta notación puede aplicarse a los tipos set-of, sequence-of, set, tipos sequence y choice, o tipos formados a partir de ellos por rotulación o subtipificación.

37.6.3 Para los tipos que están definidos en términos de un tipo único (interno) distinto (set-of, sequence-of y tipos derivados de ellos por rotulación o subtipificación), se proporciona una restricción que toma la forma de una especificación de valor de subtipo. La notación para esto es "SingleTypeConstraint":

SingleTypeConstraint ::= SubtypeSpec

El "SubtypeSpec" define un subtipo de un tipo único (interno) distinto. Un valor de tipo progenitor es un miembro del conjunto de valor de subtipo, si y solamente si cada valor interno pertenece a un subtipo obtenido al aplicar la "SubtypeSpec" al tipo interno.

37.6.4 Para los tipos que están definidos en términos de tipos múltiples internos y distintos (choice, set, sequence y tipos derivados de ellos por rotulación o subtipificación), se puede proporcionar un cierto número de restricciones para estos tipos internos. La notación para esto es "MultipleTypeConstraints":

MultipleTypeConstraints ::=

FullSpecification │ PartialSpecification

FullSpecification ::= {TypeConstraints}

PartialSpecification ::= { . . ., TypeConstraints}

TypeConstraints ::=

NamedConstraint │
NamedConstraint, TypeConstraints

NamedConstraint ::= identifier Constraint │ Constraint

37.6.5 "TypeConstraints" contiene una lista de limitaciones de los tipos componentes del tipo progenitor. En un tipo sequence las limitaciones deben aparecer en orden. El tipo interno al que se aplica la limitación se identifica por medio de su identificador, si tiene uno, o por su posición, en el caso de tipos sequence.

Nota – Cuando el tipo interno no tiene identificador, la notación puede ser ambigua.

37.6.6 "MultipleTypeConstraints" comprende una "FullSpecification" o una "PartialSpecification". Cuando se usa "FullSpecification" está implicada la restricción de presencia "ABSENT" en todos los tipos internos que no están explícitamente enumerados (véase el § 37.6.8) y cada tipo interno que no está marcado con "OPTIONAL" o "DEFAULT" en el tipo progenitor estará enumerado explícitamente. Cuando se emplea "PartialSpecification" no hay limitaciones implicadas, y cualquier tipo interno puede ser omitido de la lista.

37.6.7 Un tipo interno particular puede estar limitado en términos de su presencia (en valores del tipo progenitor), su valor, o de ambos. La notación es "Constraint":

Constraint ::= ValueConstraint PresenceConstraint

37.6.8 Una limitación del valor de un tipo interno se expresa por la notación "ValueConstraint":

ValueConstraint ::= SubtypeSpec | empty

La limitación se cumple por un valor del tipo progenitor si y solamente si el valor interno pertenece al subtipo especificado en "SubtypeSpec" aplicado al tipo interno.

37.6.9 Una limitación a la presencia de un tipo interno se expresará por medio de la notación "PresenceConstraint":

PresenceConstraint ::= PRESENT | ABSENT | empty | OPTIONAL

El significado de estas alternativas y las situaciones en las que son permitidas se definen en los § 37.6.9.1 al 37.6.9.3.

37.6.9.1 Si el tipo progenitor es sequence o set, un elemento tipo marcado con "OPTIONAL" puede ser restringido a estar "PRESENT" (en cuyo caso la limitación se cumple si y sólo si el valor del elemento correspondiente está presente) o a estar "ABSENT" (en cuyo caso la limitación se cumple únicamente si el valor del elemento correspondiente está ausente) o estar "OPTIONAL" (en cuyo caso no se impone una limitación ante la presencia del valor del elemento correspondiente).

37.6.9.2 Si el tipo progenitor es choice, un tipo componente puede ser restringido a estar "ABSENT", en cuyo caso la limitación se cumple si y sólo si el tipo componente correspondiente no se usa en el valor.

37.6.9.3 El significado de una "PresenceConstraint" vacía, depende de si se utiliza una "FullSpecification" o una "PartialSpecification":

a) en una "FullSpecification", esto es equivalente a una restricción de "PRESENT";

b) en una "PartialSpecification", no se impone ninguna restricción.

ANEXO A

(a la Recomendación X.208)

La notación macro

A.1 *Introducción*

La NSA.1 proporciona al usuario de la NSA.1 un mecanismo para definir una nueva notación con la que pueda luego construir y referenciar tipos NSA.1 o especificar valores de tipos. La nueva notación se define utilizando la notación "MacroDefinition". Una "MacroDefinition" especifica simultáneamente una nueva notación para construir o referenciar un tipo y también una nueva notación para especificar un valor (véase en el § I.3 un ejemplo ilustrativo del uso de la notación macro).

Con una “MacroDefinition” el usuario NSA.1 especifica la nueva notación por medio de un conjunto de producciones de manera similar a la de esta Recomendación. El redactor de la definición macro:

a) especifica la sintaxis completa que ha de utilizarse para definir todos los tipos admitidos por la macro; (se invoca esta especificación de sintaxis para el análisis sintáctico en todas las ocasiones de aparición del nombre macro en la notación de tipo NSA.1); y

b) especifica la sintaxis completa que se usará para un valor de uno de estos tipos; (esta especificación de sintaxis se invoca para análisis sintáctico siempre que se espera un valor del tipo macro); y

c) especifica como valor de un tipo NSA.1 normalizado (de complejidad arbitraria), el tipo y valor resultantes para todos los casos de la notación de valor macro.

Un caso de la sintaxis definida por la notación macro puede contener instancias de tipos o valores (usando la notación NSA.1 normalizada). Estos tipos o valores (que aparecen en el uso de la notación macro) pueden ir asociados, durante el análisis sintáctico, con una local type reference o una local value reference por sentencias apropiadas en la definición macro. Es posible también incluir, dentro de la definición macro, asignaciones de tipo NSA.1 normalizadas. Estas asignaciones se activarán cuando la categoría sintáctica asociada se adapte a la del elemento o elementos en la situación de la nueva notación que se está analizando. Su tiempo de vida útil está limitado al del análisis.

Cuando se analiza un valor en la nueva notación, están disponibles las asignaciones efectuadas durante el análisis de la notación de tipo correspondiente. Tal análisis debe preceder lógicamente al análisis de cada caso de la notación de valor.

El tipo y valor resultantes de un caso de utilización de la notación de valor nueva están determinados por el valor (y el tipo del valor) asignado finalmente a la referencia valor local identificada por la palabra clave VALUE del elemento, de acuerdo con el tratamiento de la macrodefinición para la notación de tipo nuevo seguida por la notación de valor nuevo.

Cada “MacroDefinition” define una notación (una sintaxis) de definición de tipo y una notación (una sintaxis) de definición de valor. El tipo NSA.1 que se define por un caso de notación de tipo nuevo, puede, pero no necesariamente, depende del caso de la notación valor a la que el tipo está asociado. A este respecto, el uso de la notación de tipo nuevo es similar a CHOICE – el rótulo está indeterminado. Así en este caso la nueva notación no puede utilizarse en lugares donde se requiera un rótulo conocido, ni puede rotularse implícitamente.

A.2 *Ampliaciones a los elementos y juegos de caracteres NSA.1*

Los caracteres | y > se usan en la notación macro.

Se utilizan también los elementos especificados en las siguientes subcláusulas.

A.2.1 *Macroreference* Referenciamacro

Nombre del elemento – macroreference

Una “macroreference” constará de la secuencia de caracteres especificada para una “typereference” en el § 8.2 salvo en que todos los caracteres estarán en mayúsculas. Dentro de un módulo único, no se empleará a la vez la misma secuencia de caracteres para una typereference y una macroreference.

A.2.2 *Productionreference* Referenciaproducción

Nombre del elemento – productionreference

Una “productionreference” constará de la secuencia de caracteres especificada para una “typereference” en el § 8.2.

A.2.3 *Localtypereference* Referenciatipolocal

Nombre del elemento – localtypereference

Una “localtypereference” constará de la secuencia de caracteres especificados para una “typereference” en el § 8.2. Una “localtypereference” se utiliza como identificador para tipos que son reconocidos durante el análisis sintáctico de un caso del nuevo tipo o de una notación de valor.

A.2.4 *Localvaluereference* Referenciavalorlocal

Nombre del elemento – localvaluereference

Una "localvaluereference" constará de la secuencia de caracteres especificados para una "valuereference" en el § 8.2. Una "localvaluereference" se utiliza como identificador para valores que son reconocidos durante el análisis sintáctico de un caso del nuevo tipo o de una notación de valor.

A.2.5 *Elemento de alternación*

Nombre del elemento – " | "

Este elemento contendrá un único carácter |.

A.2.6 *Elemento finalizador de definición*

Nombre del elemento – >

Este elemento contendrá un único carácter >.

Nota – Este elemento < para el comienzo de las definiciones se define en el § 8.13.

A.2.7 *Elemento terminal sintáctico*

Nombre del elemento – astring

Una "astring" contendrá un número arbitrario (eventualmente cero) de caracteres, del juego de caracteres NSA.1 (véase el § 7) entrecomillados ". El carácter " se representará en una "astring" por un par de ".

Nota – El uso de la "astring" en la notación macro especifica la aparición, en el punto correspondiente de la sintaxis que se está analizando, de los caracteres incluidos entre comillas (").

CUADRO 8/X.208

Secuencia especificada por ítems

Nombre del elemento	Cláusula de definición
"string"	Cualquier secuencia de caracteres
"identifier"	8.3 – Identificadores
"number"	8.8 – Números
"empty"	8.7 – Vacío

A.2.8 *Elementos palabras clave de categoría sintáctica*

Nombres de los elementos: "string"
"identifier"
"number"
"empty"

Los elementos con los nombres anteriores constarán (en la notación macro) de secuencias de caracteres en el nombre, excluyendo las comillas ("). Estos elementos se utilizan en la notación macro al especificar la aparición en un caso de la nueva notación, de ciertas secuencias de caracteres. Las secuencias en la nueva notación especificadas por cada elemento se dan en el cuadro 8/X.208, indicándose el apartado de esta Recomendación que define la secuencia de caracteres que aparecen en la nueva notación.

Nota – La notación macro no admite la distinción entre identificadores y referencias basada en el tipo de la letra inicial. Esto se debe a razones históricas.

A.2.9 *Elementos palabras clave adicionales*

Nombres de los elementos: MACRO
TYPE
NOTATION
VALUE
value
type

Los elementos con los nombres anteriores constarán de la secuencia de caracteres en el nombre.

Los elementos especificados en los § A.2.2 a § A.2.4 inclusive no serán una de las secuencias del § A.2.9, salvo cuando sean utilizados como se especifica a continuación.

Se utiliza la palabra clave "MACRO" para introducir una definición macro. Se utilizará la palabra clave "TYPE NOTATION" como nombre de la producción que define la notación del tipo nuevo. La palabra clave "VALUE NOTATION" se utilizará como nombre de la producción que define la notación del valor nuevo. La palabra clave "VALUE" se utilizará como la "localvaluereference" a la que se le asigna el valor resultante. La palabra clave "value" se utilizará para especificar que cada caso de la nueva notación contiene en ese punto, utilizando la notación normalizada NSA.1, algún valor de un tipo (especificado en la definición macro). La palabra clave "type" se utilizará para especificar que cada caso de nueva notación contiene en ese punto algún "Type", utilizando la notación normalizada NSA.1.

A.3 *Notación de definición macro*

A.3.1 Se definirá una macro empleando la notación "MacroDefinition":

MacroDefinition ::=
    macroreference
    MACRO
    "::="
    MacroSubstance

MacroSubstance ::=
    BEGIN MacroBody END |
    macroreference |
    Externalmacroreference

MacroBody ::=
    TypeProduction
    ValueProduction
    SupportingProductions

TypeProduction ::=
    TYPE NOTATION
    "::="
    MacroAlternativeList

ValueProduction ::=
    VALUE NOTATION
    "::="
    MacroAlternativeList

SupportingProductions ::=
    ProductionList |
    empty

ProductionList ::=
    Production |
    ProductionListProduction

Production ::=
    productionreference
    "::="
    MacroAlternativeList

Externalmacroreference ::=

modulereference . macroreference

Nota – El objeto es permitir que una definición macro pueda referenciar (es decir, utilizar) otras macros. El método que ha de emplearse para que la notación lo permita requiere ulterior estudio.

A.3.2 Si se elige la alternativa "macroreference" de "MacroSubstance", el módulo que contiene la definición macro:

a) contendrá otra definición macro que defina esta "macroreference"; o

b) contendrá la "macroreference" en sus "SymbolsImported".

A.3.3 Si se elige la alternativa "ExternalMacroReference" de "MacroSubstance", el módulo denominado "modulereference" contendrá otra definición macro que defina "macroreference". La definición asociada está entonces también asociada con la "macroreference" que se define.

A.3.4 La cadena de definiciones que pueden surgir de las aplicaciones repetidas de las reglas de los § A.3.2 a A.3.3 finalizará con una "MacroDefinition" que utilice la alternativa "BEGIN MacroBody END" siendo esa "MacroBody" la que define la notación de valor y tipo de la macro que esta definiéndose.

A.3.5 Cada "productionreference" que aparezca en una "SymbolDefn" (véase el § A.3.9) figurará solamente una vez como primer elemento de una "Production".

A.3.6 Cada caso de la notación de tipo nuevo comenzará con la secuencia de caracteres “macroreference”, seguida por una de las secuencias de caracteres referenciadas por “TYPE NOTATION” después de aplicar las producciones especificadas en la definición macro.

A.3.7 Cada caso de la notación de valor nuevo constará de una de las secuencias de caracteres referenciados por “VALUE NOTATION” después de aplicar las producciones especificadas en la definición macro.

A.3.8 La “MacroAlternativeList” en una producción especifica los conjuntos posibles de secuencias de caracteres referenciados por esa producción. Se especifica por:

MacroAlternativeList ::=

MacroAlternative |
MacroAlternativeList “|” MacroAlternative

El conjunto de secuencias de caracteres identificadas por la “MacroAlternativeList” está formado por todas las secuencias de caracteres referenciadas por cualquiera de las producciones “MacroAlternative” en la “MacroAlternativeList”.

A.3.9 La notación para una “MacroAlternative” será:

MacroAlternative ::= SymbolList

SymbolList ::=
SymbolElement |
SymbolList SymbolElement

SymbolElement ::=
SymbolDefn |
EmbeddedDefinitions

SymbolDefn ::=
astring |
productionreference |
“string” |
“identifier” |
“number” |
“empty” |
type |
type(localtypereference) |
value(MacroType) |
value(localvaluereference MacroType) |
value(VALUE MacroType)

MacroType ::= localtypereference |
Type

Nota – En una macro, cualquier “MacroType” definido en esa macro puede aparecer en cualquier punto en que NSA.1 especifica un “Type”.

Una “MacroAlternative” referencia todas las cadenas de caracteres que se forman tomando cualquiera de las cadenas de caracteres referenciadas por la primera “SymbolDefn” de la “SymbolList” seguidas de cualquiera de las cadenas de caracteres referenciadas por la segunda “SymbolDefn” y así sucesivamente hasta la última “SymbolDefn” de la “SymbolList”, inclusive.

Nota – Las “EmbeddedDefinitions” (si las hay) no juegan un papel directo en la determinación de las cadenas.

A.3.10 Una “astring” referencia la secuencia de caracteres de la “astring” sin el entrecomillado”.

A.3.11 Una “productionreference” referencia cualquier secuencia de caracteres especificada por la “Production” a la que identifica.

A.3.12 Las secuencias de caracteres referenciadas por las próximas cuatro alternativas para “SymbolDefinition” están especificadas en el cuadro 8/X.208.

Nota – Las secuencias de caracteres referenciados por la "string" deberían estar terminadas en un caso de la notación macro por la aparición de una secuencia referenciada por la "SymbolDefn" de la "SymbolList" siguiente.

A.3.13 Un "type" referencia cualquier secuencia de símbolos que forman una notación "Type", como se especifica en el § 12.1.

Nota – El "DefinedType" del § 12.1 puede contener en este caso una "localtypereference" que referencia un tipo definido en la notación macro.

A.3.14 Un "type(localtypereference)" referencia cualquier secuencia de símbolos que forma un "Type" como se especifica en § 12.1, pero además asigna ese tipo a la "localtypereference". Puede darse una asignación posterior a la misma "localtypereference".

A.3.15 Un "value(MacroType)" referencia cualquier secuencias de símbolos que forma una notación de "Value" (como se especifica en el § 12.7) para el tipo especificado por "MacroType".

A.3.16 Un "value(localvaluereferenceMacroType)" referencia cualquier secuencia de símbolos que forma una notación de "Value" (como se especifica en el § 12.7) para el tipo especificado por "MacroType", pero además asigna el valor especificado por la notación de valor al "localvaluereference". Puede darse una asignación posterior al "localvaluereference".

A.3.17 Un "value(VALUE MacroType)" referencia cualquier secuencia de símbolo que forma una notación de "Value" (como se especifica en el § 12.7) para el tipo especificado por "MacroType", pero además devuelve al valor como el valor especificado por la notación de valor. El tipo de valor devuelto es el tipo identificado por MacroType.

A.3.18 Se da precisamente una asignación a VALUE (como se especifica en los § A.3.17 o A.3.19) en el análisis de cualquier caso correcto de la nueva notación de valor.

A.3.19 La notación para una "EmbeddedDefinitions" será:

```
EmbeddedDefinitions ::=
        <EmbeddedDefinitionList>

EmbeddedDefinitionList ::=
        EmbeddedDefinition |
        EmbeddedDefinitionList
        EmbeddedDefinition

EmbeddedDefinition ::=
        LocalTypeassignment |
        LocalValueassignment

LocalTypeassignment ::=
        localtypereference
        "::="
        MacroType

LocalValueassignment ::=
        localvaluereference
        MacroType
        "::="
        MacroValue

MacroValue ::=
        Value |
        localvaluereference
```

La asignación de un "MacroType" a una "localtypereference" (o de un "MacroValue" a una "localvaluereference") dentro de "EmbeddedDefinitions" produce efecto durante el análisis sintáctico de un caso de la nueva notación en el momento en que se entran las "EmbeddedDefinitions", y persiste hasta que produce la redefinición de la "localtypereference" o "localvaluereference".

Nota 1 – El uso de la "localtypereference" o "localvaluereference" asociada en cualquier punto de la "Alternative" implica suposiciones sobre la naturaleza del algoritmo de análisis. Tales suposiciones serían indicadas por un comentario. Por ejemplo, el uso de la "localtypereference" siguiendo textualmente a "EmbeddedDefinitions" implica el análisis de izquierda a derecha.

Nota 2 – La "localvaluereference" "VALUE" puede asignarse a un valor ya sea por la construcción "value (VALUE MacroType)" o por una "EmbeddedDefinition". En ambos casos el valor es devuelto, como se especifica en el § A.3.17.

A.4 *Uso de la nueva notación*

Siempre que una notación de "Type" (o "Value") sea requerida por esta Recomendación, se puede usar un caso de la notación de tipo (o notación de valor) definida por una macro siempre que la macro esté:

a) definida dentro del mismo módulo; o sea

b) importada al módulo, por medio de la aparición de la "macroreference" en el "SymbolsImported" del módulo.

Para permitir la última posibilidad, una "macroreference" puede aparecer como un "Symbol" en el § 9.1.

Nota 1 – Esta ampliación a la notación normalizada NSA.1 no figura en el texto de la Recomendación.

Nota 2 – Es posible construir módulos incluyendo secuencias de asignación de tipo y macrodefiniciones que hacen el análisis sintáctico del valor en valores DEFAULT arbitrariamente complejos.

ANEXO B

(a la Recomendación X.208)

Asignación por la ISO de valores de componentes OBJECT IDENTIFIER

B.1 Se especifican tres arcos desde el nodo raíz. La asignación de valores e identificadores, y la autoridad para la asignación de valores de componentes subsiguientes, es como sigue:

Valor	Identificador	Autoridad para asignaciones subsiguientes
0	ccitt	CCITT
1	iso	ISO
2	conjuntamente-iso-ccitt	véase el anexo D

Nota – El resto de este anexo concierne solamente a la asignación de valores por la ISO.

B.2 Cada uno de los identificadores "ccitt", "iso", y "conjuntamente-iso-ccitt", asignados anteriormente, puede utilizarse como una "NameForm".

B.3 Se especifican cuatro arcos desde el nodo identificado por "iso". La asignación de valores e identificadores es:

Valor	Identificador	Autoridad para asignaciones subsiguientes
0	norma	Véase el § B.4
1	autoridad-registro	Véase el § B.5
2	organismo-miembro	Véase el § B.6
3	organización-identificada	Véase el § B.7

Estos identificadores pueden ser usados como una "NameForm".

B.4 Los arcos situados debajo de "norma" tendrán cada uno el valor del número de una Norma internacional. Donde la Norma internacional es multiparte, habrá un arco adicional para el número de parte, a menos que esto esté explícitamente excluido en el texto de la Norma internacional. Los demás arcos tendrán los valores definidos en dicha Norma internacional.

Nota – Si una Norma internacional que no es multiparte atribuye identificadores de objeto, y después se convierte en una Norma internacional multiparte, continuará atribuyendo identificadores de objeto como si fuera una Norma internacional monoparte.

B.5 Los arcos situados debajo de "autoridad de registro" (registration authority) están reservados a un addéndum a esta Recomendación que progresará en el establecimiento de procedimientos para la identificación de Autoridades de registro ISA específicas.

B.6 Los arcos situados inmediatamente debajo de "organismo-miembro" (member-body) tendrán valores de un indicativo de país numérico de tres cifras, tal como se especifica en ISO 3166, que identifica el Organismo miembro de la ISO de ese país (véase la nota). La "NameForm" de un componente de identificador de objeto no se permite con esos identificadores. Los arcos situados debajo del "country code" no se definen en esta Recomendación.

Nota – La existencia de un indicativo de país en ISO 3166 no implica necesariamente que haya un Organismo miembro de la ISO que represente a ese país o que ese Organismo miembro de la ISO para dicho país administre un método para la asignación de componentes de identificador de objeto.

B.7 Los arcos situados inmediatamente debajo de "organización-identificada" (identified-organization) tendrán valores de un Designador de indicativo internacional (DII) atribuido por la Autoridad de registro para la ISO 6523 que identifica una organización emisora registrada específicamente por esa autoridad como atribuidora de componentes de identificador de objeto (véanse las notas 1 y 2). Los arcos situados inmediatamente debajo de DII tendrán valores de un "código de organización" (organization code) atribuido por la organización emisora de acuerdo con ISO 6523. Los arcos situados debajo de "código de organización" (organitation code) no se definen en esta Recomendación (véase la nota 3).

Nota 1 – El requisito de que las organizaciones emisoras estén registradas por la Autoridad de registro para ISO 6523 como atribuidoras de códigos de organización para componentes de identificador de objeto asegura que solamente se atribuyan valores numéricos de acuerdo con esta Recomendación.

Nota 2 – La declaración de que una organización emisora atribuye códigos de organización para códigos de componentes de identificador de objeto no excluye el uso de esos códigos para otros propósitos.

Nota 3 – Se da por supuesto que las organizaciones identificadas por "código de organización" (organization code) definirán los arcos ulteriores de forma que se asegure la atribución única de valores.

Nota 4 – El resultado de los indicado en el § B.7 es que cualquier organización puede obtener un código de organización emisora apropiada, y puede entonces asignar valores de OBJECT IDENTIFIER para sus propios fines con la seguridad de que esos valores no estarán en conflicto con valores asignados por otras organizaciones. De este modo, el fabricante podría, por ejemplo, asignar un OBJECT IDENTIFIER a sus propios formatos de información de su propiedad.

ANEXO C

(a la Recomendación X.208)

Asignación por el CCITT de valores de componentes de OBJECT IDENTIFIER

C.1 Se especifican tres arcos desde el nodo raíz. La asignación de valores e identificadores, y la autoridad para la asignación de valores de componentes subsiguientes, es como sigue:

Valor	Identificador	Autoridad para asignaciones subsiguientes
0	ccitt	CCITT
1	iso	ISO
2	conjuntamente-iso-ccitt	Véase el anexo D

Nota – El resto de este anexo concierne solamente a la asignación de valores por el CCITT.

C.2 Cada uno de los identificadores "ccitt", "iso" y "conjuntamente-iso-ccitt", asignados anteriormente, puede utilizarse, como una "NameForm".

C.3 Se especifican cuatro arcos desde el nodo identificado por "ccitt". La asignación de valores e identificadores es:

Valor	Identificador	Autoridad para asignaciones subsiguientes
0	recomendación	Véase el § C.4
1	cuestión	Véase el § C.5
2	administración	Véase el § C.6
3	operador-red	Véase el § C.7

Estos identificadores pueden ser usados como una "NameForm".

C.4 Los arcos situados debajo de "recomendación" tienen el valor de 1 a 26 con identificadores asignados de la a a la z. Los arcos situados debajo de éstos tienen los números de la Recomendaciones del CCITT de las series identificadas por la letra. Los arcos situados debajo de éstos se determinan según sea necesario por la Recomendación del CCITT. Los identificadores a a z pueden utilizarse como "NameForm".

C.5 Los arcos situados debajo de "cuestión" tienen valores correspondientes a las Comisiones de estudio del CCITT, calificadas por Periodos de estudios. El valor se calcula mediante la fórmula:

número de Comisión de estudio + (Periodo * 32)

donde "Periodo" tiene el valor 0 para 1984-1988, 1 para 1988-1992, etc., y el multiplicador 32 es decimal.

Los arcos situados debajo de cada Comisión de estudio tienen los valores correspondientes a las Cuestiones asignadas a dicha Comisión de estudio. Los arcos situados debajo de éstos están determinados como lo exija el Grupo (por ejemplo, Grupo de trabajo o Grupo del Relator especial) asignado al estudio de la Cuestión.

C.6 Los arcos situados debajo de "Administración" tienen los valores de los indicativos de país para datos IPD de la Recomendación X.121. Los arcos situados debajo de éstos se determinan como lo exija la Administración del país identificado por el IPD de la Recomendación X.121.

C.7 Los arcos situados debajo de "operador-red" tienen el valor de los códigos de identificación de redes de datos CIRD de la Recomendación X.121. Los arcos situados debajo de éstos están determinados como lo exija la Administración o empresa privada de explotación reconocida (EPER) identificada por el CIRD.

ANEXO D

(a la Recomendación X.208)

Asignación conjunta de valores de componentes de OBJECT IDENTIFIER

D.1 Se especifican tres arcos desde el nodo raíz. La asignación de valores e identificadores, y la autoridad para la asignación de valores de componentes subsiguientes es como sigue:

Valor	Identificador	Autoridad para asignaciones subsiguientes
0	ccitt	CCITT
1	iso	ISO
2	conjuntamente-iso-ccitt	Ver a continuación

Nota – El resto de este anexo concierne solamente a la asignación conjunta de valores por ISO-CCITT.

D.2 Cada uno de los identificadores "ccitt", "iso" y "conjuntamente-iso-ccitt", asignados anteriormente puede utilizarse como una "NameForm".

D.3 Los arcos situados debajo de "conjuntamente-iso-ccitt" "joint-iso-ccitt" tienen valores que están asignados y acordados de vez en cuando por la ISO y el CCITT para identificar sectores de actividad de normalización conjunta ISO-CCITT, de conformidad con los "Procedimientos para asignación de valores de componentes de identificador de objeto para uso conjunto ISO-CCITT"[2)].

D.4 Los arcos situados debajo de cada arco identificado por el mecanismo definido en el § D.3 serán atribuidos de acuerdo con los mecanismos establecidos cuando se atribuye el arco.

---

2) La autoridad de registro para la asignación de valores de componentes de identificador de objeto para uso conjunto ISO-CCITT es el American National Standards Institute (ANSI), 1430 Broadway, New York, NY 10018, USA.

Nota – Se espera que esto incluirá la delegación de autoridad para el acuerdo conjunto de los Relatores del CCITT y de la ISO para el ámbito conjunto de trabajo.

D.5 Las normas de la ISO y las Recomendaciones del CCITT iniciales en sectores de actividad conjunta ISO-CCITT requieren atribuir OBJECT IDENTIFIERS con anterioridad al establecimiento de los procedimientos del § D.3, y a partir de ahí proceder a la atribución de acuerdo con los anexos B y C. Los objetos de información codificados de esta forma por las normas de la ISO o Recomendaciones del CCITT no tendrán que cambiar sus OBJECT IDENTIFIERS cuando se establezcan los procedimientos del § D.3.

APÉNDICE I

(a la Recomendación X.208)

Ejemplos y sugerencias

Este apéndice contiene ejemplos del uso de la NSA.1 en la descripción de estructuras (ficticias) de datos. Contiene también sugerencias u orientaciones para el uso de las diferentes características de la NSA.1.

I.1 *Ejemplos de un registro de personal*

Se ilustra el uso de la NSA.1 mediante un registro simple y ficticio de personal.

I.1.1 *Descripción informal de un registro de personal*

A continuación se presenta la estructura de un registro de personal y su valor para un determinado individuo.

Nombre:	John P. Smith
Título:	Director
Empleado número:	51
Fecha del contrato:	17 septiembre 1971
Nombre del cónyuge:	Mary T. Smith
Número de hijos:	2
Información hijo	
Nombre:	Ralph T. Smith
Fecha de nacimiento:	11 noviembre 1957
Información hijo	
Nombre:	Susan B. Jones
Fecha de nacimiento:	17 julio 1959

I.1.2 *Descripción NSA.1 de la estructura de registro*

A continuación se describe formalmente la estructura de cada registro personal utilizando la notación normalizada para tipos de datos.

```
PersonnelRecord ::= [APPLICATION 0] IMPLICIT SET
{                                   Name,
  title                             [0] VisibleString,
  number                            EmployeeNumber,
  dateOfHire                        [1] Date,
  nameOfSpouse                      [2] Name,
  children                          [3] IMPLICIT
                                    SEQUENCE OF
                                    ChildInformation
                                    DEFAULT {}  }
```

```
ChildInformation ::= SET
    {                                   Name,
     dateOfBirth                        [0] Date}

Name ::= [APPLICATION 1] IMPLICIT SEQUENCE
{givenName                              VisibleString,
 initial                                VisibleString,
 familyName                             VisibleString}

EmployeeNumber ::= [APPLICATION 2] IMPLICIT INTEGER

Date ::= [APPLICATION 3] IMPLICIT VisibleString--YYYYMMDD
```

Este ejemplo ilustra un aspecto del análisis de la NSA.1. La construcción sintáctica "DEFAULT" sólo puede aplicarse a un elemento de una "SEQUENCE" o un "SET", no puede aplicarse a un elemento de una "SEQUENCE OF". Así pues el "DEFAULT {}" en "PersonnelRecord" se aplica a "children", no a "ChildInformation".

I.1.3 *Descripción NSA.1 de un valor de registro*

Se describe formalmente a continuación el valor del registro de personal de John Smith utilizando la notación normalizada para valores de datos.

```
{                               {givenName "John",initial "P",familyName "Smith"}
 title                          "Director"
 number                         51
 dataOfHire                     "19710917"
 nameOfSpouse                   {givenName "Mary", initial"T", familyName "Smith"}
 children
    {{{ givenName "Ralph",initial"T",familyName"Smith"}
                dateOfBirth "19571111"}
                {{givenName "Susan",initial"B",familyName"Jones"}
                dateOfBirth "19590717"}}}
```

I.2 *Directrices para la utilización de la notación*

Los tipos de datos y la notación formal definidos en la presente Recomendación son flexibles, permitiendo elaborar una amplia variedad de protocolos diseñados para utilizarlos. 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 la sección 3 no se tratan aquí.

I.2.1 *Boolean*

I.2.1.1 Se utilizará un tipo boolean para modelar los valores de una variable lógica (es decir, de dos estados), por ejemplo, la respuesta a una pregunta del tipo "si" o "no".

Ejemplo:

```
Employed ::= BOOLEAN
```

I.2.1.2 Al asignar un nombre de referencia a un tipo boolean, se elegirá uno que describa el estado *verdadero*.

Ejemplo:

```
Married ::= BOOLEAN

not

MaritalStatus ::= BOOLEAN
```

Véase también el § I.2.3.2

I.2.2 *Integer*

I.2.2.1 Se utilizará un tipo integer para modelar los valores (a todos los fines prácticos, sin limitación de magnitud) de una variable cardinal o entera.

Ejemplo:

```
CheckingAccountBalance ::= INTEGER
  -- en céntimos; negativo significa saldo deudor
```

I.2.2.2 Se definirán los valores máximo y mínimo admitidos de un tipo integer como valores diferenciales.

Ejemplo:

```
DayOfTheMonth ::= INTEGER {first(1),last(31)}
```

I.2.3 *Enumerated*

I.2.3.1 Se utilizará un tipo enumerated con valores diferenciados 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 claridad.

Ejemplo:

```
DayOfTheWeek ::= ENUMERATED
  {sunday(0),monday(1),tuesday(2),wednesday(3),thursday(4), friday(5),saturday(6)}
```

I.2.3.2 Se utilizará un tipo enumerated 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(0),married(1)}
```

en anticipación a

```
MaritalStatus ::= ENUMERATED {single(0),married(1),widowed(2)}
```

I.2.4 *Real*

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

Ejemplo:

```
AngleInRadians ::= REAL
pi REAL ::= {31415926535897932384626433832 79, 10, –30}
```

I.2.5 *Bit string*

I.2.5.1 Se utilizará un tipo bit string 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
  -- una secuencia de bits conforme con el CCITT
  -- Recomendación T.4.
```

I.2.5.2 Se definirán el primer y último bit significativos de una cadena de bits de longitud fija como bits diferenciados.

Ejemplo:

```
Nibble ::= BIT STRING {first(0),last(3)}
```

I.2.5.3 Se utilizará un tipo bit string para modelar los valores de una **bit map** (correspondencia de bits), colección ordenada de variables lógicas que indica si una condición particular se cumple para cada colección de objetos correspondientemente ordenada.

Ejemplo:

```
SunnyDaysOfTheMonth ::= BIT STRING {first(1),last(31)}
  -- El día i fue soleado si y sólo si el bit i es uno
```

I.2.5.4 Se utilizará un tipo bit string con valores diferenciados para modelar los valores de una colección de variables lógicas relacionadas.

Ejemplo:

```
PersonalStatus ::= BIT STRING
{casado(0),empleado(1),veterano(2),titulado(3)}
```

I.2.6 *Octet string*

I.2.6.1 Se utilizará un tipo octet string 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 múltiple de ocho.

Ejemplo:

```
G4FacsimileImage ::= OCTET STRING
  -- una secuencia de octetos de acuerdo con
  -- Recomendaciones T.5 y T.6 del CCITT
```

I.2.6.2 Se utilizará un tipo character string con preferencia a un tipo octet string cuando se disponga de uno apropiado.

Ejemplo:

```
Surname ::= PrintableString
```

I.2.6.3 Se utilizará un tipo octet string para modelar cualquier cadena de información que no pueda modelarse utilizando uno de los tipos de cadena de caracteres. Se han de especificar el repertorio de caracteres y su codificación en octetos.

Ejemplo:

```
PackedBCDString ::= OCTET STRING
  -- las cifras 0 a 9, dos cifras por octeto
  -- cada cifra codificada de 0000 a 1001
  -- 1111₂ se utiliza para relleno
```

I.2.7 *Null*

Se utilizará un tipo null para indicar la ausencia efectiva de un elemento de la secuencia.

Ejemplo:

```
PatientIdentifier ::= SEQUENCE
{name                   VisibleString,
 roomNumber             CHOICE
        {INTEGER,
        NULL -- if an out-patient --}}
```

Nota – La utilización "OPTIONAL" proporciona una facilidad equivalente.

I.2.8 *Sequence y sequence-of*

I.2.8.1 Se utilizará un tipo sequence-of para modelar una colección de variables del mismo tipo, de número elevado o impredecible, y cuyo orden sea significativo.

Ejemplo:

```
NamesOfMemberNations ::= SEQUENCE OF VisibleString
  -- en el orden de su incorporación
```

I.2.8.2 Se utilizará un tipo 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 del protocolo a la siguiente.

Ejemplo:

```
NamesOfOfficers ::= SEQUENCE
{president              VisibleString,
 vicePresident          VisibleString,
 secretary              VisibleString}
```

I.2.8.3 Se utilizará un tipo sequence 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 de protocolo a la siguiente.

Ejemplo:

```
Credentials ::= SEQUENCE
{userName               VisibleString,
 password               VisibleString,
 accountNumber          INTEGER}
```

I.2.8.4 Si los elementos de un tipo sequence son de diversos tipos pero su número es fijo, se deberá asignar a cada elemento un nombre de referencia cuya finalidad no pueda deducirse claramente de su tipo.

Ejemplo:

```
File ::= SEQUENCE
 {                      ContentType,
  other                 FileAttributes,
  content               ANY}
```

Véanse también los § I.2.5.3, § I.2.5.4, y § I.2.7.

I.2.9 *Set*

I.2.9.1 Se utilizará un tipo set para modelar una colección de variables cuyo número es conocido y pequeño y cuyo orden no es significativo. Se identificará cada variable con un rótulo específico del contexto.

Ejemplo:

```
UserName ::= SET
 {personalName          [0] IMPLICIT VisibleString,
  organisationName      [1] IMPLICIT VisibleString,
  countryName           [2] IMPLICIT VisibleString}
```

I.2.9.2 Se utilizará un tipo set con "OPTIONAL" para modelar una colección de variables que es un subconjunto (adecuado o inadecuado) de otra colección de variables de número conocido y razonablemente pequeño y cuyo orden no es significativo. Se identificará cada variable con un rótulo del contexto.

Ejemplo:

```
UserName ::= SET
 {personalName          [0] IMPLICIT VisibleString,
  organisationName      [1] IMPLICIT VisibleString OPTIONAL
    -- defaults to that of the local organisation --,

  countryName           [2] IMPLICIT VisibleString OPTIONAL
    -- por defecto el de la región --}
```

I.2.9.3 Se utilizará un tipo set para modelar una colección de variables cuya constitución pueda presumiblemente cambiar de una versión del protocolo a la siguiente. Se identificará cada variable rotulándola específicamente según el contexto.

Ejemplo:

```
UserName ::= SET
 {personalName          [0] IMPLICIT VisibleString,
  organisationName      [1] IMPLICIT VisibleString OPTIONAL
    -- defaults to that of the local organisation --,

 countryName            [2] IMPLICIT VisibleString OPTIONAL
    -- por defecto el de la región
    -- otros atributos opcionales para ulterior estudio --}
```

I.2.9.4 Si el tipo set tiene un número fijo de miembros, se tendrá que asignar a cada miembro un nombre de referencia cuya finalidad no pueda deducirse claramente de su tipo.

Ejemplo:

```
FileAttributes ::= SET
  {owner                    [0] IMPLICIT UserName,
    sizeOfContentInOctets   [1] IMPLICIT INTEGER,
                            [2] IMPLICIT AccessControls,
    . . .}
```

I.2.9.5 Se utilizará un tipo set para modelar una colección de variables cuyos tipos son los mismos y cuyo orden no es significativo.

Ejemplo:

```
Keywords ::= SET OF VisibleString -- en orden arbitrario
```

Véanse también los § I.2.5.4 y § I.2.10.3.

I.2.10 *Tagged*

I.2.10.1Se utilizará un tipo tagged universal para definir en esta Recomendación solamente, un tipo de datos, generalmente útil, con independencia de la aplicación, que deben ser diferenciables (por medio de su representación) de los demás tipos de datos.

Ejemplo:

```
EncryptionKey ::= [UNIVERSAL 30] IMPLICIT OCTET STRING
-- siete octetos
```

I.2.10.2Se utilizará un tipo tagged de aplicación amplia para definir un tipo de datos que se utiliza de una manera amplia y dispersa en el contexto de una presentación particular y que debe distinguirse (mediante su representación) de los demás tipos de datos utilizados en el contexto de presentación.

Ejemplo:

```
FileName ::= [APPLICATION 8] IMPLICIT SEQUENCE
  {directoryName                VisibleString,
    directoryRelativeFileName   VisibleString}
```

I.2.10.3Se utilizarán tagged específicos del contexto para distinguir los miembros de un conjunto. Se asignarán rótulos numéricos empezando desde el cero si su única limitación es la distinguibilidad.

Ejemplo:

```
CustomerRecord ::= SET
  {name                     [0] IMPLICIT VisibleString,
    mailingAddress          [1] IMPLICIT VisibleString,
    accountNumber           [2] IMPLICIT INTEGER,
    balanceDue              [3] IMPLICIT INTEGER -- in cents --}
```

I.2.10.4Cuando un miembro determinado de un conjunto ha sido rotulado con aplicación amplia, no es necesario utilizar otro rótulo específico del contexto a no ser que se requiera por razones de distinguibilidad (o pueda serlo en el futuro). Cuando un miembro del conjunto ha sido rotulado con carácter universal debe utilizarse otro rótulo específico del contexto.

Ejemplo:

```
ProductRecord ::= SET
  {                         UniformCode,
    description             [0] IMPLICIT VisibleString,
    inventoryNo             [1] IMPLICIT INTEGER,
    inventoryLevel          [2] IMPLICIT INTEGER}

UniformCode ::= [APPLICATION 13] IMPLICIT INTEGER
```

I.2.10.5Se utilizarán tipos tagged específicos del contexto para distinguir las alternativas de una CHOICE. Se asignarán rótulos numéricos empezando a partir de cero si su única limitación es la distinguibilidad.

Ejemplo:

```
CustomerAttribute ::= CHOICE
  {name                [0] IMPLICIT VisibleString,
    mailingAddress     [1] IMPLICIT VisibleString,
    accountNumber      [2] IMPLICIT INTEGER,
    balanceDue         [3] IMPLICIT INTEGER -- in cents --}
```

I.2.10.6Cuando se ha definido una alternativa determinada de CHOICE utilizando un tipo tagged de aplicación amplia, no es necesario utilizar otro rótulo específico del contexto a no ser que se requiera por razones de distinguibilidad (o pueda serlo en el futuro).

Ejemplo:

```
ProductDesignator ::= CHOICE
  {                    UniformCode,
    description        [0] IMPLICIT VisibleString,
    inventoryNo        [1] IMPLICIT INTEGER}

UniformCode ::= [APPLICATION 13] IMPLICIT INTEGER
```

I.2.10.7Cuando se ha rotulado con carácter universal una determinada alternativa CHOICE deberá utilizarse otro rótulo específico del contexto, a no ser que la elección tenga por fin proporcionar más de un tipo universal.

Ejemplo:

```
CustomerIdentifier ::= CHOICE
  {name                VisibleString,
    number             INTEGER}
```

I.2.10.8Se utilizará un tipo tagged de uso privado para definir un tipo de datos que se vaya a utilizar dentro de una organización o país determinado y que deba ser distinguible (por medio de su representación) de los demás tipos de datos utilizados por dicha organización o país.

Ejemplo:

```
AcmeBadgeNumber ::= [PRIVATE 2] IMPLICIT INTEGER
```

I.2.10.9En estas directrices se ha utilizando la rotulación implícita en los ejemplos en que está autorizado hacerlo. Dependiendo de las reglas de codificación, esto da 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 verificar los tipos. En el último caso, se utilizará la rotulación explícita.

Véanse también los § I.2.9.1, § I.2.9.2, § I.2.11.1 y § I.2.11.2.

I.2.11 *Choice*

I.2.11.1Se utilizará CHOICE para modelar una variable seleccionada de una colección de variables cuyo número es conocido y reducido. Se identificará cada variable posible rotulándola explícitamente según el contexto.

Ejemplo:

```
FileIdentifier ::= CHOICE
  {relativeName        [0] IMPLICIT VisibleString,
      -- name of file (for example, “MarchProgressReport”)

    absoluteName       [1] IMPLICIT VisibleString,
      -- name of file and containing directory
      -- (for example, “<Williams> MarchProgressReport”)

    serialNumber       [2] IMPLICIT INTEGER
      -- system-assigned identifier for file --}
```

I.2.11.2Se utilizará CHOICE para modelar una variable seleccionada de una colección de variables cuya constitución vaya a cambiar probablemente de una versión del protocolo a la siguiente. Se identificará cada variable posible rotulándola específicamente según el contexto.

Ejemplo:

```
FileIdentifier ::= CHOICE
 {relativeName              [0] IMPLICIT VisibleString,
    -- name of file (for example, “MarchProgressReport”)

  absoluteName              [1] IMPLICIT VisibleString,
    -- name of file and containing directory
    -- (for example, “<Williams> MarchProgressReport”)
  -- other forms of file identifiers are for further study --}
```

I.2.11.3Se asignará un nombre de referencia a cada alternativa cuya finalidad no puede deducirse claramente de su tipo.

Ejemplo:

```
FileIdentifier ::= CHOICE
 {relativeName              [0] IMPLICIT VisibleString,
    -- name of file (for example, “MarchProgressReport”)

  absoluteName              [1] IMPLICIT VisibleString,
    -- names of file and containing directory
    -- (for example, “<Williams> MarchProgressReport”)

                            [2] IMPLICIT SerialNumber
    -- system-assigned identifier for file --}
```

I.2.11.4Cuando la rotulación implícita sea la norma para una determinada aplicación de esta Recomendación se utilizará una CHOICE de un solo tipo cuando se haya previsto la posibilidad de permitir más de un tipo en el futuro. Esto excluye la posibilidad de que se produzca la rotulación implícita, y facilita de este modo la transición.

Ejemplo:

```
Greeting ::= [APPLICATION 12] CHOICE
                            {VisibleString}
```

en anticipación de:

```
Greeting ::= [APPLICATION 12] CHOICE
                            {VisibleString,
                              Voice}
```

I.2.12 *Tipo selection*

I.2.12.1Se utilizará un tipo selection para modelar una variable cuyo tipo sea el de una determinada alternativa de una CHOICE definida anteriormente.

I.2.12.2Considérese la definición:

```
FileAttribute ::= CHOICE
{date-last-used             INTEGER,
 file-name                  VisibleString}
```

entonces es posible la siguiente definición:

```
CurrentAttributes ::= SEQUENCE
{date-last-used             <FileAttribute,
 file-name                  <FileAttribute}
```

con una notación de valor posible de

```
{date-last-used 27
 file-name “PROGRAM”}
```

También es posible la siguiente definición:

```
AttributeList ::= SEQUENCE
{first-attribute date-last-used <FileAttribute,
 second-attribute file-name <FileAttribute}
```

con una notación de valor posible de

```
{first-attribute 27,
 second-attribute “PROGRAM”}
```

I.2.13 *Any*

I.2.13.1Se utilizará un tipo any para modelar una variable cuyo tipo no se ha especificado o lo ha sido en otra parte utilizando NSA.1.

Ejemplo:

```
MessageContents ::= ANY
   -- un elemento de datos cuyo tipo se ha especificado
   -- fuera de esta Recomendación utilizando la notación NSA.1.
```

I.2.14 *External*

I.2.14.1Se utilizará un tipo external para modelar una variable cuyo tipo no se ha especificado, o lo ha sido en otra parte sin limitación en la notación utilizada para especificar el tipo.

Ejemplo:

```
FileContents ::= EXTERNAL
DocumentList ::= SEQUENCE OF EXTERNAL
```

I.3 *Ejemplo de empleo de la notación macro*

Supóngase que se desea hacer una notación para la definición de tipo de la forma:

PAIR TYPEX = ... TYPEY = ...

con una notación del valor correspondiente que admita

(X = ---- , Y = ----)

El . . . . y el ---- se refieren a cualquier tipo NSA.1 y al valor correspondiente, respectivamente.

Esta notación de tipo macro podría utilizarse para definir tipos de valores de la forma siguiente:

```
T1 ::= PAIR
          TYPEX = INTEGER
          TYPEY = BOOLEAN

T2 ::= PAIR
          TYPEX = VisibleString
          TYPEY = T1
```

Entonces un valor de tipo T1 podría ser:

(X = 3, Y = TRUE)

y un valor de tipo T2 podría ser:

(X = “Name”, Y = (X = 4, Y = FALSE))

La siguiente definición macro podría utilizarse para establecer esta nueva notación, como una ampliación de la NSA.1 básica:

```
PAIR
MACRO ::= BEGIN
TYPE NOTATION ::=
                "TYPEX"
                "="
                type (Local-type-1)
                    -- Expects any ASN.1 type and assigns it
                    -- to the variable Local-type-1;
                "TYPEY"
                "="
                type (Local-type-2)
                    -- Expects a second ASN.1 type and assigns
                    -- it to the variable Local-type-2;

VALUE NOTATION ::=
                "("
                "X"
                "="
                value (Local-value-1 Local-type-1)
                    -- Expects a value for the type in
                    -- Local-type-1, and assigns it
                    -- to the variable Local-value-1;
                ","
                "Y"
                "="
                value (Local-value-2 Local-type-2)
                    -- Expects a value for the type in
                    -- Local-type-2 and assigns it
                    -- to the variable Local-value-2;
                <VALUE SEQUENCE {Local-type-1, Local-type-2}
                    ::= {Local-value-1, Local-value-2}>
                -- This "embedded definition" returns
                -- the final value as the value
                -- of a sequence of the two types.
                ")"

END
```

En este ejemplo, el tipo básico NSA.1 del valor devuelto es independiente de la instancia real de la notación de valor, pero depende de la instancia de la notación de tipo que se utilice. En otros casos, puede estar plenamente determinado por la definición macro, o puede depender de la instancia de la notación de valor que se utilice. Nótese sin embargo, que en todos los casos es la producción "VALUE NOTATION" la que debe examinarse para determinar el tipo de valor devuelto. La producción "TYPE NOTATION" simplemente define la sintaxis para la definición de tipo, y establecer el valor inicial de las variables locales que deben utilizarse al analizar una instancia de la notación de valor.

I.4 *Utilización durante la identificación de sintaxis abstractas*

I.4.1 El uso del servicio de presentación (Recomendación X.216) requiere la especificación de valores denominados valores de datos de presentación y el agrupamiento 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 NSA.1.

I.4.2 Puede utilizarse la notación NSA.1 como un instrumento general en la especificación de los valores de datos de presentación y su agrupamiento en sintaxis abstractas denominadas.

I.4.3 En el uso más sencillo, existe un tipo NSA.1 único, de manera que cada valor de datos de presentación en la sintaxis abstracta denominada es un valor de dicho tipo NSA.1. Este tipo normalmente será un tipo CHOICE y cada valor de datos de presentación será un tipo alternativo de este tipo CHOICE. En este caso se recomienda que se utilice la notación modular NSA.1 para contener dicho tipo CHOICE como el primer tipo definido, seguido por la definición de aquellos tipos (no universales) a que se hizo referencia directa o indirectamente por este tipo CHOICE.

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

I.4.4 A continuación se facilita un ejemplo de texto que podría aparecer en una aplicación normal. El final del ejemplo se identifica por la frase "END OF EXAMPLE" para evitar confusiones.

Ejemplo:

```
ISOxxxx-yyyy DEFINITIONS ::=
BEGIN
PDU ::== CHOICE
    {connect-pdu            . . .,
      data-pdu              CHOICE
                {     . . .,
              . . .}    ,
    . . .               }
. . .
END
```

Esta norma internacional asigna el valor de identificador de objetos NSA.1

{iso standard xxxx abstract-syntax (1)}

como un nombre de sintaxis abstracta para el conjunto de valores de datos de presentación, cada uno de los cuales es un valor del tipo NSA.1 "ISOxxxx-yyyy.PDU".

El valor descriptor de objeto NSA.1 correspondiente, será:

"....."

Los valores del identificador de objeto y del descriptor de objeto NSA.1:

{joint-iso-ccitt asn1 (1) basic-encoding (1)}

y

"Basic Encoding of a single ASN.1 type"

(asignados a un objeto de información en la Recomendación X.209) pueden utilizarse como nombre de sintaxis de transferencia con este nombre de sintaxis abstracta.

END OF EXAMPLE

I.4.5 La norma puede además admitir la sintaxis de transferencia que se obtiene aplicando:

{joint-iso-ccitt asn1 (1) basic-encoding (1)}

obligatoria para esta sintaxis abstracta.

I.5 *Subtipos*

I.5.1 Para limitar los valores de un tipo existente, se utilizarán los subtipos que se admitan en una situación particular.

Ejemplo:

```
AtomicNumber ::= INTEGER (1..104)

TouchToneString ::= IA5String (FROM
                                ("0"|"1"|"2"|"3"|"4"|"5"|"6"|
                                "7"|"8"|"9"|"*"| #)|
                                SIZE (1..63))

ParameterList ::= SET SIZE (0..63) OF Parameter

SmallPrime ::= INTEGER (2|3|5|7|11|13|17|19|23|29)
```

I.5.2 Cuando dos o más tipos relacionados tengan puntos significativos en común, 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 los puntos en común, 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 realización práctica común 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 pueden expresarse, en 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 aquéllos que son facultativos, el grado de las diferencias entre los tipos individuales y la posibilidad de una estrategia de evolución.

I.5.3 Se utilizará la subtipificación para definir parcialmente un valor, por ejemplo, una unidad de datos de protocolo que será sometida a una prueba de conformidad, cuando la prueba afecte únicamente a algunos de los componentes de la UDP.

Ejemplo:

Dado que:

```
PDU ::= SET
                  {alpha      [0]    INTEGER,
                    beta      [1]    IA5String OPTIONAL,
                    gamma     [2]    SEQUENCE OF Parameter,
                    delta     [3]    BOOLEAN}
```

cuando se componga una prueba que requiera que Boolean sea falso e integer negativo, escríbase:

```
TestPDU ::= PDU (WITH COMPONENTS
                  {. . .,
                    delta (FALSE),
                    alpha (MIN. . .<0)})
```

y si además debe estar presente cadena AI5 (IA5String), beta, con una longitud de 5 ó 12 caracteres, escríbase:

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

I.5.4 Si se ha definido un datatype para propósito general, 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)))
```

I.5.5 Para formar nuevos subtipos a partir de los subtipos existentes, se utilizarán subtipos contenidos:

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 (
                    INCLUDES First-quarter |
                    INCLUDES Second-quarter)
Second-half ::= Months (
                    INCLUDES Third-quarter |
                    INCLUDES Fourth-quarter)
```

APÉNDICE II

(a la Recomendación X.208)

Resumen de la notación NSA.1

En el § 8 se definen los elementos siguientes:

typereference	INTEGER	BEGIN
identifier	BIT	END
valuereference	STRING	DEFINITIONS
modulereference	OCTET	EXPLICIT
comment	NULL	ENUMERATED
empty	SEQUENCE	EXPORTS
number	OF	IMPORTS
bstring	SET	
hstring	IMPLICIT	REAL
cstring	CHOICE	INCLUDES
"::="	ANY	MIN
{	EXTERNAL	MAX
}	OBJECT	SIZE
<	IDENTIFIER	FROM
,	OPTIONAL	WITH
.	DEFAULT	COMPONENT
(	COMPONENTS	PRESENT
)	UNIVERSAL	ABSENT
[	APPLICATION	DEFINED
]	PRIVATE	BY
-	TRUE	PLUS-INFINITY
BOOLEAN	FALSE	MINUS-INFINITY

En la presente Recomendación, se utilizan las producciones siguientes con los ítems arriba mencionados como símbolos terminales:

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

TagDefault ::=
    EXPLICIT TAGS |
    IMPLICIT TAGS |
    empty

ModuleIdentifier ::=
    modulereference
    AssignedIdentifier

AssignedIdentifier ::=
    ObjectIdentifier Value |
    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 ModuleIdentifier

SymbolList ::= Symbol, SymbolList | Symbol

Symbol ::= typereference | valuereference

AssignmentList ::=
Assignment AssignmentList |
Assignment

Assignment ::= Typeassignment | Valueassignment

Externaltypereference ::=
modulereference
.
typereference

Externalvaluereference ::=
modulereference
.
valuereference

DefinedType ::= Externaltypereference | typereference

Definedvalue ::= Externalvaluereference | valuereference

Typeassignment ::=
typereference
“::=”
Type

Valueassignment ::=
valuereference
Type
“::=”
Value

Type ::= BuiltinType | DefinedType | Subtype

```
BuiltinType ::=
                BooleanType             |
                IntegerType             |
                BitStringType           |
                OctetStringType         |
                NullType                |
                SequenceType            |
                SequenceOfType          |
                SetType                 |
                SetOfType               |
                ChoiceType              |
                SelectionType           |
                TaggedType              |
                AnyType                 |
                ObjectIdentifierType    |
                CharacterStringType     |
                UsefulType                      |
                EnumeratedType          |
                RealType                |

NamedType ::= identifier Type | Type | Selection Type

Value ::= Builtin Value | DefinedValue
BuiltinValue ::=
                BooleanValue            |
                IntegerValue            |
                BitStringValue          |
                OctetStringValue        |
                NullValue               |
                SequenceValue           |
                SequenceOfValue         |
                SetValue                |
                SetOfValue              |
                ChoiceValue             |
                SelectionValue          |
                TaggedValue             |
                AnyValue                |
                ObjectIdentifierValue   |
                CharacterStringValue    |
                EnumeratedValue         |
                RealValue               |

NamedValue ::= identifier Value | 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 {Enumeration}

Enumeration ::=
NamedNumber |
NamedNumber, Enumeration

EnumeratedValue ::= identifier

RealType ::= REAL

RealValue ::= NumericRealValue | SpecialRealValue

NumericRealValue ::= {Mantissa, Base, Exponent} | 0

Mantissa ::= SignedNumber

Base ::= 2 | 10

Exponent ::= SignedNumber

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 {ElementTypeList}
SEQUENCE {}

ElementTypeList ::=
ElementType |
ElementTypeList,ElementType

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

SequenceValue ::= {ElementValueList} | {}

ElementValueList ::=
NamedValue |
ElementValueList,NamedValue

SequenceOfType ::= SEQUENCE OF Type | SEQUENCE

SequenceOfValue ::= {ValueList} |{}

ValueList ::= Value | ValueList,Value

SetType ::= SET{ElementTypeList} | SET {}

SetValue ::= {ElementValueList} | {}

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

SetOfValue ::= {ValueList} | {}

ChoiceType ::= CHOICE{AlternativeTypeList}

AlternativeTypeList ::=
                NamedType |
                AlternativeTypeList,NamedType

ChoiceValue ::= NamedValue

SelectionType ::= identifier < Type

SelectionValue ::= NamedValue

TaggedType ::=
                Tag Type |
                Tag IMPLICIT Type |
                Tag EXPLICIT Type

Tag ::= [Class ClassNumber]

ClassNumber ::= number | DefinedValue

Class ::=
                UNIVERSAL |
                APPLICATION |
                PRIVATE |
                empty

TaggedValue ::= Value

AnyType ::=
                ANY |
                ANY DEFINED BY identifier

AnyValue ::= Type Value

ObjectIdentifierType ::= OBJECT IDENTIFIER

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

ObjIdComponentList ::=
                ObjIdComponent |
                ObjIdComponent ObjIdComponentList

ObjIdComponent ::=
                NameForm |
                NumberForm |
                NameAndNumberForm

NameForm ::= identifier

NumberForm ::= number | DefinedValue

NameAndNumberForm ::= identifier(NumberForm)

CharacterStringType ::= typereference

CharacterStringValue ::= cstring

UsefulType ::= typereference
```

En el § 2 se definen los siguientes tipos de cadena de caracteres:

NumericString	VisibleString
PrintableString	ISO646String
TeletexString	IA5String
T61String	GraphicString
VideotexString	GeneralString

En el § 3 se definen los siguientes tipos útiles:

GeneralizedTime	EXTERNAL
UTCTime	ObjectDescriptor

En el § 4 se utilizan las siguientes producciones:

Subtype ::=
ParentType SubtypeSpec | SET SizeConstraint OF Type | SEQUENCE SizeConstraint OF Type

ParentType ::= Type

SubtypeSpec ::=
(SubtypeValueSet SubtypeValueSetList)

SubtypeValueSetList ::=
“ | ”
SubtypeValueSet
SubtypeValueSetList |
empty

SubtypeValueSet ::=
SingleValue |
ContainedSubtype |
ValueRange |
PermittedAlphabet | SizeConstraint | InnerTypeConstraint
SingleValue ::= Value
ContainedSubtype ::= INCLUDES Type
ValueRange ::= LowerEndPoint .. UpperEndPoint

LowerEndpoint ::= LowerEndValue | LowerEndValue <

UpperEndpoint ::= UpperEndValue | <UpperEndValue

LowerEndValue ::= Value | MIN

UpperEndValue ::= Value | MAX

SizeConstraint ::= SIZE SubtypeSpec

PermittedAlphabet ::= FROM SubtypeSpec
InnerTypeConstraints ::=

WITH COMPONENT SingleTypeConstraint |
WITH COMPONENTS MultipleTypeConstraints

SingleTypeConstraint ::= SubtypeSpec

MultipleTypeConstraints ::=
FullSpecification | PartialSpecification

FullSpecification ::= {TypeConstraints}

PartialSpecification ::= {. . ., TypeConstraints}

TypeConstraints ::=
NamedConstraint |
NamedConstraint, TypeConstraints

NamedConstraint ::= identifier Constraint | Constraint

Constraint ::= ValueConstraint PresenceConstraint

ValueConstraint ::= SubtypeSpec | empty

PresenceConstraint ::= PRESENT | ABSENT | empty | OPTIONAL

En el § A.2 se definen los elementos adicionales siguientes para su empleo en la notación macro:

macroreference	"number"
productionreference	"empty"
localtypereference	MACRO
localvaluereference	TYPE
"|"	NOTATION
>	VALUE
astring	value
"string"	type
"identifier"	

En el anexo A se definen las siguientes producciones junto con los elementos arriba mencionados y aquellos relacionados al principio de este apéndice, como símbolos terminales:

MacroDefinition ::=
    macroreference
    MACRO
    "::="
    MacroSubstance

MacroSubstance ::=
    BEGIN MacroBody END |
    macroreference |
    Externalmacroreference

MacroBody ::=
    TypeProduction
    ValueProduction
    SupportingProductions

TypeProduction ::=
    TYPE NOTATION
    "::="
    MacroAlternativeList

ValueProduction ::=
    VALUE NOTATION
    "::="
    MacroAlternativeList

SupportingProductions ::= ProductionList | empty

ProductionList ::= Production | ProductionList Production

Production ::=
    productionreference
    "::="
    MacroAlternativeList

Externalmacroreference ::=
    modulereference . macroreference

MacroAlternativeList ::=
    MacroAlternative |
    MacroAlternativeList "|" MacroAlternative

MacroAlternative ::= SymbolList

SymbolList ::= SymbolElement | SymbolList SymbolElement

SymbolElement ::= SymbolDefn | EmbeddedDefinitions

SymbolDefn ::=
astring |
productionreference |
“string” |
“identifier” |
“number” |
“empty” |
type |
type(localtypereference) |
value(MacroType) |
value(localvaluereference MacroType) |
value(VALUE MacroType)

MacroType ::= localtypereference | Type

EmbeddedDefinitions ::= <EmbeddedDefinitionList>

EmbeddedDefinitionList ::=
EmbeddedDefinition |
EmbeddedDefinitionList EmbeddedDefinition

EmbeddedDefinition ::=
LocalTypeassignment |
LocalValueassignment

LocalTypeassignment ::=
Localtypereference
“::=”
MacroType

LocalValueassignment ::=
localvaluereference
MacroType
“::=”
MacroValue

MacroValue ::= Value | localvaluereference