(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 272 429**

(51) Int. Cl.:
H04N 7/24 (2006.01)
H03M 7/30 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Número de solicitud europea: **01460047 .2**
(86) Fecha de presentación : **13.07.2001**
(87) Número de publicación de la solicitud: **1276324**
(87) Fecha de publicación de la solicitud: **15.01.2003**

(54) Título: **Método para comprimir un árbol jerárquico, señal correspondiente y método para descodificar una señal.**

(45) Fecha de publicación de la mención BOPI:
01.05.2007

(45) Fecha de la publicación del folleto de la patente:
01.05.2007

(73) Titular/es: **FRANCE TELECOM**
6, place d'Alleray
75015 Paris, FR
Groupe des Ecoles des Telecommunications y Expway

(72) Inventor/es: **Concolato, Cyril;**
Seyrat, Claude;
Pau, Grégoire;
Thienot, Cédric y
Cotarmanac'h, Alexandre

(74) Agente: **Elzaburu Márquez, Alberto**

DESCRIPCIÓN

Método para comprimir un árbol jerárquico, señal correspondiente y método para descodificar una señal.

El campo del invento es el de la compresión de datos. Más precisamente, el invento se refiere a la compresión de documentos basados en XML ("Lenguaje de Markup Extendido").

El invento tiene aplicaciones, en particular, pero no solamente, en los campos siguientes:

- aplicaciones multimedia;

- herramientas de indexación;

- herramientas de manipulación de datos meta;

- la especificación MPEG-7 (véase particularmente NACK y col. "Todo lo que quería saber acerca de MPEG-7: Parte 2" Multimedia IEEE, Sociedad de Ordenadores de IEEE, Estados Unidos de Norteamérica, volumen 6, nº 4, octubre de 1999, páginas 64-73).

- SMIL;

- TV en cualquier instante;

- la tercera generación de radiocomunicaciones (3GPP).

Las técnicas de compresión de la técnica anterior para XML tienen varios inconvenientes. En particular, no soportan al mismo tiempo acceso rápido a datos, relaciones de alta compresión y construcción progresiva del documento. En otras palabras, la mayoría del tiempo, cuando una de las características antes mencionadas es soportada, las demás características se pierden.

Una de las técnicas de compresión de la técnica anterior es conocida como BiM (MPEG binario) y está notablemente descrita en Texto de ISO/IEC FCD 15938-1 Enlace de Descripción de Contenido Multimedia de Tecnología de la Información - Parte 1 Sistemas "ISO/IEC" JTC1/SC 29/WG11 "MPEG" 01/N4001, marzo de 2001. Tal técnica proporciona un método para comprimir un documento XML haciendo binaria la estructura del documento, es decir los nodos de una estructura de árbol asociados al documento XML. Por tanto, la relación de compresión conseguida empleando la técnica BiM es muy pobre, aunque la técnica BiM permite un rápido acceso a datos, construcción progresiva del documento y capacidad para omitir. Liefke y col., en "XMILL: Un compresor eficiente para datos XML". (Registro Sigmod, Asociación para Maquinaria de Ordenadores, Nueva York, Estados Unidos de Norteamérica, vol. 29, nº 2, Junio de 2000, páginas 153-164) ha descrito también un método para comprimir datos en un árbol XML. Un inconveniente de esta técnica es que no permite identificar un subárbol, asignar una técnica de codificación de compresión específica a un tipo de datos del subárbol identificado, e implantar esta técnica de codificación de compresión sólo para las hojas del subárbol cuyos datos son del tipo de técnica de codificación de compresión asociada.

Es un propósito del invento compensar los inconvenientes de las técnicas de la técnica anterior.

Más precisamente, el invento tiene por objetivo proporcionar una técnica de compresión eficiente para documentos basados en XML.

El invento también tiene por objetivo proporcionar una técnica de compresión para XML que proporciona la capacidad para omitir, relaciones de alta compresión y construcción progresiva del documento.

El invento también tiene por objetivo comprimir eficientemente descriptores MPEG-7.

Otro propósito del invento es poner en práctica un método para comprimir un documento XML que mejora mucho la relación de compresión proporcionada por una técnica del tipo BIM, pero que proporciona las mismas funcionalidades que las proporcionadas por el BIM.

Los propósitos antes mencionados del invento, así como otros propósitos del invento que aparecerán después, son conseguidos, de acuerdo con el invento, por medio de un método para comprimir un árbol jerárquico que describe una señal multimedia, comprimiendo dicho árbol nodos y hojas, que pueden estar asociadas a contenidos de al menos dos tipos distintos, en el que dicho método emplea una compresión de contenido para al menos alguna de dichas hojas por medio de al menos dos técnicas de codificación por compresión, estando selectivamente asociada cada una de dichas técnicas al menos a uno de dichos tipos de contenido.

De acuerdo con una realización preferida del invento, tal método comprende una operación de identificar al menos un subárbol y una operación de asignar una de dichas técnicas de codificación por compresión a dicho subárbol.

Ventajosamente, tal método comprende una operación de implantar dicha técnica de codificación por compresión asignada a dicho subárbol solamente para las hojas de dicho subárbol cuyo contenido es del tipo asociado a dicha técnica de codificación por compresión, y las otras hojas de dicho subárbol no experimentan ninguna codificación por compresión.

De acuerdo con una característica ventajosa del invento, tal método emplea una descripción paramétrica de dichas técnicas de codificación por compresión.

Preferentemente, tal método comprende también una operación de comprimir la estructura de dicho árbol.

Ventajosamente, dicho árbol es del tipo BiM (MPEG binario) de acuerdo con el estándar MPEG7.

Preferentemente, una de dichas técnicas de codificación por compresión emplea una cuantificación lineal.

Ventajosamente, una de dichas técnicas de codificación por compresión emplea un algoritmo de compresión estadístico.

Preferentemente, dicho algoritmo es del tipo GZip.

Ventajosamente, dicho algoritmo es empleado simultáneamente para un conjunto de datos correspondiente al contenido de al menos dos hojas.

Preferentemente, dicho árbol representa la estructura de un documento de tipo XML (Lenguaje de Markup extendido).

El invento se refiere también a un método para descodificar una señal multimedia comprimida de acuerdo con el método antes mencionado para comprimir un árbol jerárquico.

Ventajosamente, tal método pone en práctica una operación de refrescar un contexto de descodificación actual de acuerdo con la información del contexto de codificación transportada por dicha señal.

Preferentemente, dicho contexto actual define al menos un tipo de contenido, comprendiendo dicho método una operación de puesta en práctica de una técnica de descodificación por compresión asociada dicho tipo de contenido para las hojas que tienen un contenido de dicho tipo de contenido.

El invento se refiere también a una señal generada por el método antes mencionado para comprimir un árbol jerárquico.

Otras características y ventajas del invento aparecerán más claramente a la luz de la siguiente descripción, dada como ejemplo meramente ilustrativo pero no restrictivo, y de las siguientes figuras:

La fig. 1 ilustra el concepto de contexto de codificación

La fig. 2 describe la estructura de un elemento cuando es codificado de acuerdo con la técnica BiM;

La fig. 3 ilustra alguna de las operaciones llevadas a cabo de acuerdo con el invento para comprimir el contenido de las hojas de un árbol jerárquico.

Antes de describir en detalle como poner en práctica el invento, primero recordaremos las principales características de la técnica de compresión de la técnica anterior, conocida como la técnica BiM.

Para facilitar la comprensión del documento, algunas referencias están reunidas en el anexo 9 y se ha hecho referencia a ellas a lo largo de todo el documento.

Todos los anexos proporcionados con el presente documento son parte de la descripción actual del invento.

1. **Técnica anterior**

Introducción

Un contexto de codificación, ilustrado en la fig. 1, es un conjunto de información de descodificación, necesario mientras se descodifica la corriente o flujo de bits. Un contexto de codificación es aplicable al subárbol completo del nodo donde está definido. En todos los nodos del árbol, el contexto de codificación puede ser modificado: conduciendo a la creación de un nuevo contexto de codificación, aplicable al subárbol correspondiente.

Un contexto puede llevar varias informaciones que editan características aplicables al subárbol referido. Actualmente, en el formato de codificación de subárbol BiM [1], estas características tienen la capacidad de omitir de un subárbol/contexto y codificación de múltiples esquemas de un subárbol/contexto (con el fin de proporcionar la carac-

terística de compatibilidad hacia atrás y hacia adelante). Al final, el mecanismo de contenido puede ser deshabilitado en cada subárbol con el fin de ahorrar ancho de banda; este es el modo congelado de contexto.

El mecanismo de contexto de codificación proporciona máxima flexibilidad en cada subárbol de un árbol de documento y permite que características extensibles sean enchufadas en el mecanismo de codificación BiM. Conocemos ahora el mecanismo el contexto actual usado en BiM.

Mecanismo de contexto de codificación actual

Definiciones

Contexto de codificación

Un codingContext (contexto de codificación) es un conjunto de información, la información contextual, necesitada por el descodificador para descodificar la corriente de bits. Un contexto de codificación es aplicable al nodo donde ha sido definido, y al subárbol completo correspondiente a este nodo.

El codingContext (contexto de codificación) actual, (es decir el contexto aplicable a un nodo especificado de una descripción) puede ser modificado dentro del documento (es decir, una modificación de su conjunto subyacente de información). Cada modificación de un codingContext (contexto de codificación) conduce a la creación de un nuevo codingContext (contexto de codificación), que transportará el conjunto de información modificado. Todos los codingContexts (contextos de codificación) se espera que sean apilados, con el fin de recogerlos de nuevo, cuando el descodificador ha terminado de descodificar un contexto de subárbol correspondiente.

Descodificador

El descodificador BiM está compuesto de dos descodificadores:

- el descodificador de contexto: este descodificador está dedicado a descodificar la información contextual. Como se ha establecido antes, la información contextual no es parte de la descripción. Esto es un conjunto de información que soporta algunas características externas, compatibilidad hacia atrás y hacia adelante, omisión rápida...

- el descodificador de elementos; este descodificador, el BiM regular [1] está dedicado va a descodificar la información del elemento.

Fragmentos

Cada elemento de una descripción es codificado como el 3-plet ilustrado en la fig. 2, donde la parte de encabezamiento está constituida por dos fragmentos, cuyos tamaños pueden ser nulos:

- MC es el fragmento de meta-contexto;

- C es el fragmento de contexto;

- Elemento es el fragmento de elemento, que es el fragmento de codificación de elemento regular, véase [1].

El fragmento de meta-contexto MC contiene la información necesitada por el descodificador para descodificar el siguiente fragmento C. Es decir que el fragmento C es el fragmento de contexto del fragmento de contexto C.

El fragmento de contexto C contiene la información capaz de cambiar el conjunto de información de contexto de codificación actual, y necesitada por el descodificador para descodificar el siguiente fragmento de elemento. Es decir que el fragmento C es el fragmento de contexto del fragmento de elemento.

Conjunto de información

El contexto de codificación BiM actual lleva un conjunto de información, la información contextual, que puede estar dividida en las dos siguientes clases principales:

- la sección de meta-contexto; la información contenida en esta sección, influye sólo en el proceso de descodificación de contexto

- la sección de contexto; si la información contenida en esta sección, influye sólo en el proceso de descodificación de elemento

El conjunto actual de información es el siguiente conjunto de variables:

Clase	Variable	Valor	Descripción
MetaContexto	freezing_state	Booleano	Falso: el contexto puede ser cambiado dentro de este subárbol Verdadero: El contexto no puede ser cambiado más en este subárbol
Contexto	allows_skip	(obligatorio opcional, prohibido)	¿Es obligatoria, opcional o está prohibida la característica de omitir en este contexto?
	schema_mode	(mono, multi)	¿Está el elemento actual codificado con múltiples esquemas?
	allows_partial_instantiation	Booleano	¿Es la instalación parcial permitida en este contexto?
	allows_subtyping	Booleano	¿Está permitido el subtecleado en este contexto?

Las clases definidas son codificando exactamente los fragmentos descritos antes (el fragmento de meta-contexto MC y el fragmento de contexto C).

Fragmento de meta-contexto

Definición

El fragmento de meta-contexto MC, cuyo tamaño puede ser nulo, contiene información para saber si el descodificador tiene que leer el siguiente fragmento de contexto C, descrito en la siguiente sección.

Variable	Valor	Descripción
freezing_state	booleano	Falso: El contexto puede ser cambiado dentro de este subárbol. El fragmento MC será codificado en la corriente de bits. Verdadero: el contexto ya no puede ser cambiado en este subárbol. El fragmento MC no será codificado en la corriente de bits.

Valores por Defecto

El valor por defecto del freezing_state es falso: es decir que, por defecto, el contexto de raíz puede ser cambiado dinámicamente.

Reglas de propagación

En la creación de un nuevo contexto:

- el valor de freezing_state es ajustado al valor de freezing_state de su contexto padre.

Reglas de modificación dinámica

En cada nodo de una descripción y con el fin de crear un nuevo contexto:

- el valor de freezing_state puede ser intercambiado desde el valor falso al valor verdadero.

Reglas de descodificación

Si el valor de freezing_state es verdadero, el fragmento de meta-contexto MC (y el fragmento de contexto C próximo) no es codificado a la corriente de bits. De otro modo, la parte del fragmento de meta-contexto MC del encabezamiento es codificada como sigue:

MC 0 {	# de bits	Mnemonic
Freeze type	1-3	Vlclbf
}		

El context_chunk es una variable local, inicializada en falso.

Tipo de congelación	Implicación
110	context_chunk = verdadero y freezing_state = verdadero
10	context_chunk = verdadero
111	context_chunk = falso y freezing_state = verdadero
0	context_chunk = falso

Como se ha indicado en la sección previa, la modificación de las variables del contexto actual implica la creación de un nuevo contexto.

Influencia en el proceso de descodificación de contexto

Si el valor del context_chunk es verdadero, el descodificador tiene que leer el siguiente fragmento de contexto C.

Fragmento de Contexto

Definición

El fragmento de contexto C, cuyo tamaño puede ser nulo, contiene un conjunto de información capaz de cambiar dinámicamente las variables de contexto actuales. Estas variables son llamadas Propiedades de codificación ya que influyen en el proceso de descodificación de elemento BiM.

codingProperty	Valor	Descripción
allows_skip	(obligatorio, opcional, prohibido)	¿Es obligatoria, opcional o está prohibida la característica de omitir en este contexto?
schema_mode	(mono, multi)	¿Está el elemento actual codificado con esquemas múltiples?
allows_partial_instatiation	Booleano	¿Está la creación de caso parcial permitida en este contexto?
allows_subtyping	Booleano	¿Está el subtecleado permitido en este contexto?

Valores por defecto

La variable allows_skip es inicializada al comienzo de la corriente de bits por los dos primeros bits del campo de bit especial de 4 bits, como se ha definido en el documento de Sistemas FCD [1].

La variable allows_partial_instantiation es inicializada al comienzo de la corriente de bits por los dos terceros bits del campo de bit especial de 4 bits.

La variable allows_subtyping es inicializada al comienzo de la corriente de bits por los dos cuartos bits del campo de bit especial de 4 bits.

El valor por defecto de scheme_mode es mono; es decir que, por defecto, la raíz del subárbol/contexto es codificada con un esquema

Reglas de propagación

En la creación de un nuevo contexto:

- el valor allows_skip es configurado al valor allows_skip de su contexto padre
- el valor allows_partial_instantiation es configurado al valor de su contexto padre
- el valor allows_subtyping es configurado al valor de su contexto padre
- el valor del schema_mode es configurado a su valor por defecto

Reglas de modificación dinámica

En cada nodo de una descripción y con el fin de crear un nuevo contexto:

- allows_skip puede ser dinámicamente modificado
- allows_partial_instantiation no puede ser dinámicamente modificado
- allows_subtyping no puede ser dinámicamente modificado
- el scheme_mode puede ser dinámicamente modificado

Reglas de descodificación

El fragmento de contexto C está presente solo si el fragmento de meta-contexto MC está ya presente y su variable local previa context_chunk es verdadera.

La modificación dinámica del contexto actual está descrita con un elemento XML que es codificado por el esquema de codificación regular BiM. El elemento global modifyContext desde el esquema BiM es usado. El esquema de codificación http://www.mpeg7.org/2001/BiMCoding está descrito en el anexo 1.

El fragmento de contexto C ha de ser descodificado con el esquema regular BiM, con el esquema anterior.

Como se ha indicado antes, la modificación de codingProperties actual en el contexto implica la creación de un nuevo contexto. Por ello, la presencia del fragmento de contexto C, implica la creación de un nuevo contexto, que llevará las codingProperties modificadas.

Si el elemento allows_skip es instantiado dentro del elemento de modificación de contexto, entonces el valor allows_skip será actualizado en el nuevo contexto.

Si el elemento schema_mode es instantiado dentro del elemento modifyContext, entonces el valor del schema_mode será actualizado en el nuevo contexto.

Influencia sobre el proceso de descodificación del elemento

Los valores allows_skip y schema_mode influyen sobre el proceso de descodificación del elemento, cuando está relacionado con la característica de omitir. Este comportamiento está descrito en [1].

El valor de schema_mode influye sobre el proceso de descodificación del elemento, con el fin de saber si el elemento está codificado con o solo uno o varios esquemas. Este mecanismo está descrito en [1].

El valor allows_partial_instantiation influye sobre el proceso de descodificación del elemento, añadiendo un tipo especial partiallyInstantiated a los subtipos posibles del elemento. Véase [1].

El valor de allow_subtyping influye sobre el proceso de descodificación del elemento, y permite a un elemento o a un atributo tener diferentes tipos posibles, en caso de polimorfismo de elemento (con el atributo xsi:type) o unión. Véase [1].

2. **Descripción del invento**

2.1. *Extensión del mecanismo de contexto para proporcionar una estructura para la compresión de la hoja*

El invento propone extender el mecanismo de contexto BiM actual con el fin de soportar una nueva e interesante característica: el uso de compresores locales para comprimir hojas de un documento, con el fin de reducir el tamaño de la corriente de bits resultante.

Esta sección describe como extender el mecanismo de contexto BiM actual para soportar el uso de compresores locales. Esto es típicamente un nuevo conjunto de variables, codingProperties, enlazadas con reglas de semántica, propagación y codificación específicas. Por ello, este nuevo conjunto de codingProperties extenderá el fragmento de contexto actual.

Introducción

En un subárbol, todos los casos de uno o varios tipos simples especificados pueden ser comprimidos con uno o varios compresores especificados. Esto define básicamente un cartografiado entre un compresor y uno o varios tipos simples. Además:

- en algunos casos, un compresor puede necesitar algunos parámetros externos.
- un cartografiado puede ser activado/desactivado, con el fin de usar un compresor en algunos subárboles pero no en otros.

Por último, con el fin de ser activado/desactivado, un cartografiado debería ser referenciable y por ello, debe tener un identificador único en un contexto.

Por ello, cada contexto puede llevar cero, uno o varios codecTypeMapper; donde un codecTypeMapper es un 4-plet, que consiste de un identificador, uno o varios tipos simples, un codec, parámetros de codec opcional externos y un estado de activación.

Definiciones

CodecTypeMapper

Un codecTypeMapper es un 4-plet, que consiste en:

- un identificador, usado como una clave de referencia única en un subárbol/contexto
- uno o varios tipos simples, para los que es aplicable el cartografiado
- un codec
- parámetros de codec opcionales externos (depende del codec)
- un estado de activación

Identificador

El identificador es un número único que identifica un cartografiado dentro de un contexto en un modo no ambiguo. El esquema de codificación BiM restringe el número máximo de codecTyoeMappers en un contexto a 32.

Tipo simple

Todos los tipos simples definidos en un esquema podrían ser *a priori* codificados por cada codec pero, cada codec puede restringir esta elección. Por ejemplo, un cuantificador lineal, como se ha definido de aquí en adelante en el documento, puede ser solamente usado con tipos numéricos simples.

Un tipo simple es identificado por su nombre, y por la URL del esquema al que pertenece. Los prefijos de esquema XML deberían ser usados, con el fin de apuntar al esquema correcto. El esquema de codificación BiM define un tipo especial para codificar esta pareja; este tipo debería ser codificado como pareja de enteros; el primer entero está restringido al número actual de esquemas conocido (esta pieza de información puede ser traída en la parte DecoderConfig [1]) y el segundo entero está restringido al número de tipos simples globales presentes en el esquema correspondiente.

Codec

Un codec, abreviaturade compresor/descompresor, es un módulo que toma bits de entrada, y escribe bits de salida. Puede necesitar algunos parámetros externos opcionales.

Un codec es identificado por un nombre, entre los nombres de los codecs no abstractos definidos en el esquema de codificación BiM. El esquema de codificación BiM actual, definido en una sección anterior, no define ningún codec no abstractos, pero el §2.2 del presente documento lo hace.

Estado de activación

El estado de activación es una bandera booleana.

Reglas semánticas

CodecTypeMapper

Cada contexto:

- puede llevar 0,1 o varios codecTypeMapper
- puede definir uno o varios codecTypeMapper
- puede activar o desactivar uno o varios codecTypeMapper existentes. Si un codecTypeMapper está definido en un contexto, permanece en todos sus subcontextos.

Un codecTypeMapper existente, dentro de un contexto, no puede ser borrado ni modificado (excepto su estado de activación).

Identificador

El identificador de un cartografiado debe ser único entre todos los codecTypeMapper de un contexto.

Tipo simple

Cuando se asocia en un codecTypeMapper uno o varios tipos simples y un codec, el codec codificará/descodificará los propios tipos simples y todo los tipos simples que sean derivados de ellos.

En un contexto, debe haber como mucho un tipo simple que pueda ser aplicable con un codec.

Codec

Hay dos tipos de codecs: codecs sin memoria y codecs contextuales.

Un codec sin memoria es un módulo que codifica siempre los mismos bytes de entrada en los mismos bytes de salida: independientemente de la historia del codec. Un codec sin memoria típico es un cuantificador lineal. La compresión de hoja BiM (véase §2.2 del documento presente) describe tal codec.

Un codec contextual es un módulo que usa los bytes previos alimentados en él, (cambiando así el contexto del codec).

Tal codec no genera los mismos bytes de salida para los mismos bytes de entrada que recibe. Un codec típicamente contextual es un codec local a modo de Zip, uno está descrito en el §2.2 del documento presente.

Un codec sin memoria no induce ningún problema en la arquitectura de contexto actual pero un codec contextual si lo hace, en caso de subárbol omitible. En tales casos, un codec contextual es repuesto, con el fin de no confundir al descodificador, cuando este anterior ha omitido el subárbol.

Estado de activación

En cada subárbol/contexto, un codecTypeMapper puede ser activado o desactivado.

Este mecanismo permite definir un codecTypeMapper a un nivel más elevado del árbol de documentos, y activarlo solamente en los subárboles en los que es usado, sin redefinir el codecTypeMapper.

Un nuevo codingProperty: codecTypeMapper

En esta parte, añadiremos un nuevo codingProperty al conjunto previo de variables de la sección de contexto, descrito antes. Este nuevo codingProperty es denominado codecTypeMapper y es una lista del codecTypeMapper previo descrito en la sección previa.

Nuevo CodingProperty implicado

El contexto lleva una lista de codecTypeMapper:

CodingProperties	Valor	Descripción
codecTypeMapper[1].identifier	Int	Identificador numérico para referenciar un codingProperty en un contexto en un modo no ambiguo.
codecTypeMapper[1].simple_type[1]	Int;int	Lista de 2 plet (Índice Numérico de un esquema, índice numérico de un tipo simple en el esquema actual)
codecTypeMapper[1].codec	int	Índice Numérico de un codec en el esquema de contexto de codificación actual
codecTypeMapper[1].docec_parameters		Parámetros codec externos opcionales
codecTypeMapper[1].activation_state	booleano	Estado de activación de un codingProperty

Nuevos valores por defecto

Por defecto, no hay codecTypeMapper en un subárbol/contexto.

Si un codecTypeMapper está definido dentro de un contexto, su identificador, codec y el valor simple_type debe ser definido. Si no es especificado, el estado de activación de un codecTypeMapper nuevamente definido es ajustado a verdadero por defecto; es decir que un codecTypeMapper nuevamente definido es activado por defecto.

Nuevas reglas de propagación

Regla 1: En la creación de un nuevo contexto, la lista de codecTypeMapper es la copia de su contexto padre:

- el valor de identificador es copiado
- el valor simple_type es copiado
- el valor de codec es copiado de acuerdo con la regla 2
- el valor de codec_parameters son copiados
- el activation_state es copiado

Regla 2: el valor de codec es copiado y si:

- el codec padre codingProperty[i].codec es un codec contextual
- y si el contexto actual es omitible

Entonces,

el descodificador se espera que cree un nuevo caso del codec copiando el caso del codec padre (no solamente su valor) y reponiéndolo.

Por ejemplo, un codec Zlib sería copiado y reinicializado cuando entre en un nodo omitible.

Nuevas reglas de modificación dinámica

La lista de codecTypeMapper puede ser modificada dinámicamente, dentro de la descripción:

- un nuevo codecTypeMapper puede ser definido
- el activation_state de un codecTypeMapper existente (a continuación referenciable) puede ser modificado dinámicamente.

Un codecTypeMapper existente no puede ser eliminado y sus miembros no pueden ser modificados dinámicamente (excepto su activation_state).

Nuevas reglas de descodificación

Las mismas reglas previas se aplican a la descodificación del fragmento de contexto C, pero el nuevo esquema, descrito en el anexo 2, debería ser usado, con el fin de añadir las funcionalidades de modificación dinámica del nuevo codecTypeMapper.

Parte informativa

Ejemplos

El ejemplo, ilustrado en el anexo 3, presenta la definición de un cuantificador lineal activado (véase §2.2 del documento presente) en una descripción.

El ejemplo, ilustrado en el anexo 4, presenta la definición de un cuantificador lineal desactivado en una descripción.

2.2 *Compresión de hoja BiM*

Presentamos ahora el mecanismo empleado por el invento para codificar datos con diferentes codecs. Más precisamente, ilustramos ahora dos ejemplos, uno en el que un codec de cuantificación lineal es usado para comprimir, por ejemplo, valores de coma flotante, y el otro en el que un codec gzip es usado para comprimir, por ejemplo, valores de cadena.

Tal mecanismo está estrechamente relacionado con el contexto de codificación y permite el uso de varios otros tipos de codecs. Además, permite relacionarse apropiadamente con las características de contexto de codificación, por ejemplo subárboles omitibles. Finalmente permite volver a utilizar codecs en diferentes contextos de codificación.

Introducción

La codificación de subárbol BiM [1] no comprime las hojas de datos de una descripción. Actualmente, los valores de hoja son codificados con respecto a sus tipos (IEEE 754 flotantes y dobles, cadenas UTF...).

En muchos casos, podría ser útil usar algunas técnicas de compresión clásica como cuantificación lineal o compresión estadística para mejorar la relación de compresión de BiM sin perder sus características principales (análisis de la línea de corriente, característica de omisión rápida, descodificación tecleada).

Este documento presenta cómo puede hacerse la compresión de hojas de datos de un documento dentro del mecanismo de codificación de contexto descrito en §2.1 con el fin de conseguir mejores relaciones de compresión.

2.2.1 *Cuantificación lineal*

Definición

La cuantificación lineal es un modo usual y con pérdidas de reducir el tamaño de números codificados en la corriente de bits, cuando la fuente de la información es conocida y por ello, cuando las pérdidas pueden ser controladas.

Como ejemplo, la envolvente de una señal de audio muestreada es a menudo conocida con una cuantificación de tamaño de bit precisa, y esta técnica podría ser usada satisfactoriamente para codificar descripciones de audio MPEG-7.

Si es un número real, puede ser codificado con v_q con bits nbits donde:

$$v_q = \frac{v - v_{min}}{v_{max} - v_{min}}(2^{nbits} - 1)$$

donde:

- v_q es el valor cuantificado, codificado de v

- nbits es la precisión requerida en bits

- v_{min} es el valor mínimo inclusivo que v puede alcanzar

- v_{max} es el valor máximo incluido que v puede alcanzar

Y el valor descodificado desde v es $\overline{v}$, con:

$$\overline{v} = v_{min} + v_q \frac{v_{max} - v_{min}}{2^{nbits} - 1}$$

donde;

- $\overline{v}$ es el valor descodificado, aproximado de v

Integración con el mecanismo de contexto: el LinearQunatizerCodec

La cuantificación lineal puede ser usada como un codec, como se ha definido en el mecanismo de contexto de codificación descrito en §2.1 del presente documento. Con este mecanismo, la cuantificación lineal puede ser aplicada en hojas de datos numéricos, de un tipo simple deseado, en cualquier subárbol de una descripción.

Usados como éste, el mecanismo de contexto de codificación, asociado con el codec de cuantificación lineal, está actuando como el nodo QuatizationParameter, usado en MPEG-4 BIFS [3].

Restricción en tipos simples aplicables

De acuerdo con la definición de un codec en §2.1 del documento presente, este codec es un codec sin memoria, que puede ser aplicado en cada uno de los tipos numéricos simple atómicos y no atómicos; cuyo tipo primitivo de Esquema XML es flotante, doble o decimal.

Parámetros de codec externo

El codec cuantificador lineal necesita los 3 siguientes parámetros obligatorios:

- bitSize: la variable nbits descrita antes
- minInclusive: la variable v_{min} descrita antes
- maxInclusive: la variable v_{max} descrita antes

Definición de esquema del codec

El codec de cuantificación lineal es un nuevo codec de tipo LinearQuantizerCodecType, basado en el tipo CodecType abstracto (véase §2.1) y definido por el esquema dado en el anexo 5, en el contexto de codificación espacio de nombre URL xmins: cc=http://www.mpeg7.org/2001/BiMCoding.

Codificación (informativa)

Una hoja de datos numéricos de valor v es codificada con el entero sin signo v_q en bits nbits donde:

$$v_q = \frac{v - v_{min}}{v_{max} - v_{min}}(2^{nbits} - 1)$$

Descodificación

El entero sin signo v_q codificado en bits nbits, debería ser descodificado como v donde:

$$\overline{v} = v_{min} + v_q \frac{v_{max} - v_{min}}{2^{nbits} - 1}$$

Ejemplo

(Informativo)

El ejemplo ilustrado en el anexo 6 presenta la definición de un cuantificador lineal en una descripción.

2.2.2 Compresión estadística

Algoritmos de compresión sin pérdida estadística clásicos pueden ser usados como codecs, como se ha definido en el mecanismo de contexto de codificación (véase §2.1). Con este mecanismo, las hojas de datos, de un tipo simple deseado, pueden ser comprimidas de manera eficiente en cualquier subárbol de una descripción.

Este codec es útil para reducir significativamente el tamaño de la corriente de bits, especialmente cuando la descripción contiene muchas cadenas repetitivas o similares.

Definición

Algoritmos de compresión estadística sin pérdidas clásicos (como Zip o GZip) pueden ser usados en BiM para comprimir cualesquiera hojas de una descripción.

Pero, en la mayoría de los casos, cuando las hojas de datos son cadenas cortas que contiene menos de 10 caracteres, esto conduce a pobres rendimientos ya que los algoritmos de compresión estadística usuales requieren una mayor memoria tampón para mirar hacia delante.

Con el fin de conseguir una relación de compresión óptima, las hojas de un documento han de ser almacenadas en una pequeña memoria tampón antes de ser comprimidas. La siguiente sección define tal codificador estadístico almacenado, basándose en un algoritmo de compresión estadística sin pérdidas subyacentes.

Definiciones de un codificador estadístico almacenado

Un codificador estadístico almacenado se basa en un codificador estadístico subyacente que debería contener los siguientes métodos primitivos genéricos:

- initialize_stream(); que inicializada una corriente de compresión o descompresión
- reset_model(); que repone el modelo estadístico actual del codificador
- feed_input_bytes(); que toma bytes descomprimidos de entrada y los pone en la corriente de compresión
- flush_output_bytes(); que descarga la corriente de compresión comprimiendo los bytes de entrada ya procesados y emitiendo los correspondientes bytes de salida comprimidos
- decompress_input_bytes(); que toma una cantidad específica de bytes de entrada comprimidos y los descodifica emitiendo los bytes de salida descomprimidos correspondientes.

Un codec almacenado tiene una longitud de bytes bufferSize, estructura FIFO de memoria tampón de matriz de byte.

Desde el lado del codificador, el valor bufferSize indica cuántos bytes de entrada puede procesar el codificador antes de descargar. Desde el lado del descodificador, este es el tamaño de memoria tampón mínimo en bytes, necesario para descodificar la corriente de bits, a través del codificador estadístico subyacente API.

La memoria tampón tiene también una variable fillingLevel, que contiene el nivel de llenado real, en bytes, de la memoria tampón.

Usar el ZLib API como un codificador estadístico

La librería pública Zlib API [4], usada en el esquema de compresión GZip, proporciona un API eficiente y útil para usar compresión estadística en hojas de documentos.

El API ZLib satisface los métodos genéricos previos, con el siguiente cartografiado:

- initialize_stream() puede ser cartografiado con las funciones inflateInit() o deflateInit() de Zlib, con el parámetro de valor de eficiencia Z_DEFAULT_COMPRESSION.
- reset_model() puede ser cartografiado con una llamada inflateEnd() o deflateEnd() de ZLib y una siguiente llamada initialize_stream().
- feed_input_bytes() puede ser cartografiado con el método deflate() de ZLib con el parámetro Z_NO_FLUSH.
- flush_output_bytes() puede ser cartografiado con el método deflate() de ZLib con el parámetro Z_SYNC_FLUSH.

El codec almacenado ZLib debería ser inicializado con el valor de eficiencia Z_DEFAULT_COMPRESSION, como se ha definido en [4], que proporciona un buen resultado entre los requisitos de huella de memoria y la eficiencia de compresión.

Integración con el mecanismo de contexto: el ZLibCodec

Esta sección describe la integración de un codificador estadístico con memoria tampón previamente definido, basándose en el API ZLib, dentro del mecanismo de contexto de codificación.

Restricción en tipos simples aplicables

De acuerdo con la definición de un codec en §2.1, este codec es un codec contextual, que puede ser aplicado en cada tipo de cadena atómica y no atómica. EL ZLibCodec está basándose en la codificación primitiva subyacente de hojas de un documento, como se ha descrito en [1]. Por ejemplo, hojas int son codificadas con un entero sin signo de

32 bits, cadena con una codificación UTF-8, flotante y doble son codificados con el formato IEEE 754,... Por ello, el ZLibCodec comprimirá la hoja codificada.

Parámetros externos de codec

El codec ZLib con memoria tampón no necesita ningún parámetro externo, ya que la eficiencia del ZLib subyacente está ajustada a Z_DEFAULT_COMPRESSION y como el parámetro bufferSize no es necesario desde el lado del descodificador.

Definición de esquema del codec

El codec ZLib es un codec nuevo del tipo ZLibCodecType, basado en el tipo CodecType abstracto (véase §2.1) y definido por el esquema ilustrado en el anexo 7, en el espacio de nombre de contexto de codificación.

Codificación (informativa)

En la activación/creación de caso del codec:

- la estructura de memoria tampón FIFO se ha supuesto que está vacía, su fillinLevel es ajustado a 0
- la variable global referencable_chunk es inicializada a cero.

El referencable_chunk debería contener un fragmento referenciable de bits, que debe ser contenido por el codificador, ya que su valor será conocido después durante el proceso de codificación. La función de señalización signal_reference_chunk_known() podría ser llamada cuando este fragmento es conocido.

El tamaño, en bytes, de cada fragmento no nulo debería ser escrito antes que el propio fragmento, durante la llamada flush_output_bytes(), con la codificación de entero infinito estándar sin signo 4+1, como se ha definido en [1].

Una hoja de entrada es el valor codificado de una hoja textual, con respecto a su tipo primitivo. La longitud de la hoja, en bytes, está dada por el campo leaf.length. Por ejemplo [1], una hoja de cadena es un código UTF-8, precedido por el tamaño en bytes de la cadena (codificado con la codificación de entero infinito [1]); una hoja doble es el valor de 64 bits del estándar IEEE 754 correspondiente...

La siguiente función encode_leaf es capaz de codificar una hoja en un fragmento de bytes de salida:

```
chunk encode_leaf(chunk leaf){
  while (hoja no está vacía){
  if (fillingLevel + leaf-length < bufferSize){
      feed_input_bytes(leaf,leaf.legth)
      fillingLevel = fillingLevel + leaf.length
      if (referencable_chunk es nulo) {
            referencable_chunk = nuevo fragmento
            return referencable_chunk
      }else{
remaining_bytes = bufferSize - fillingLevel - leaf.legth
feed_inpunt_bytes(leaf,remaining_bytes)
referencable_chunk = flush_output_bytes()
signal_reference_chunk_known()
```

```
            fillingLevel = 0
            leaf = leaf.remove_beginning_bytes(remaining_bytes)
            referencable_chunk = null
          }
        }
    }
```

Descodificación

Supóngase que string_file es una cadena FIFO.

Los siguientes métodos dar() y poner() tomar respectivamente un elemento fuera de resp. Poner un elemento desde resp en el FIFO.

El método es señales Empty() si el FIFO está vacío.

Supóngase que sub es la función que toma una subcadena.

Supóngase que concat sea la función de concatenación.

Supóngase que getData() es la función que devuelve el carácter que contiene los datos de gzip desde una hoja.

Supóngase que split(char), char sep, Fifo, char remainder) sea el método que divide una matriz de caracteres en cada separador “sep” y almacena los elementos de cadena separados en la FIFO y devuelve el resto (es decir el último fragmento que no tiene “sep” para terminarlo).

```
split(char[] data, char sep, FIFO strin_fifo,char[]
remainder)
{
      if (data=null) return;
      int BEGIN=0;
      for (Int I=0;I<data.length;I++)
      {
            if(data[l]==sep)
            {
             char[]                                    str=
concat(remainder,sub(data.BEGIN,I));
             put(string_fifo, str);
             BEGIN=I+1;
            }
      }
      if(BEGIN!=data.length)
```

```
        {
            //hay un resto.
            remainder = sub(data,BEGIN,data.length);
        }

          }
          String decode()
          {
               if(isEmpty(string_fifo)
               {
                    data = getData();
                     split(data,0x00,string_fifo,remainder);
               }
               return get(string_fifo);
          }
```

En inicialización, string_fifo está vacío.

En la activación/creación de caso del codec;

- la estructura FIFO se ha supuesto que está vacía, su numberOfLeaves es ajustado a 0.
- el first_chunk variable es ajustado a verdadero.

Supóngase que el byte separador es 0x00.

La siguiente función decode_leaf es capaz de descodificar una hoja comprimida desde la corriente de bits:

```
                    string decode_leaf() {
                      if (numberOfLeaves ==0){
                          read_and_decompresses_byte()
          numberOfLeaves                                    =
count_number_of_leave_in_buffer()
         }else{

          }
}
```

La descodificación está definida por:

1. Si La FIFO está vacía:
    a. Descodificar los datos codificados
    b. Apilar en una FIFO todos los elementos separados por 0x00
    c. Si el último carácter no es 0x00 almacenar la cadena sin terminar temporalmente.

d. Si “last_element” no está vacío, insertarlo al comienzo del primer elemento en FIFO.

e. Ponga la cadena sin terminar de este redondeo en last_element.

f. Retirar y devolver el primer elemento.

2. Si el FIFO está vacío, entonces retire y devuelva el primer elemento, es equivalente a decir: “el FIFO no está vacío” y decir “no hay datos codificados en una hoja actual”.

Ejemplo

(Informativo)

La descripción dada en el anexo 8 es un ejemplo del uso del codec ZlibCodecType, cartografiado con la cadena y los tipos anyURI.

Resultados (informativo)

Las siguientes cifras muestran los rendimientos de usar el ZlibCodec de modo que comprima las hojas de texto de descripciones (las derivadas de la cadena y los tipos primitivos anyURI XML Esquema). Una memoria tampón de bufferSize = 256 bytes fue usada durante el proceso de codificación.

Los archivos usados fueron proporcionados por el subgrupo MPEG-7 MDS.

Nombre de archivo	Tamaño original (bytes)	Tamaño BiM (bytes)	BiM con el tamaño Zlibcodec (bytes)	Tamaño de archivo comprimido (bytes)
mdsExamplesClause11-12.xml	160658	22512	10602	17019
mdsExamplesClause13-15.xml	81133	11627	8538	9698
mdsExamplesClause17.xml	142426	57583	22489	21444
mdsExamplesClause4-7.xml	37208	6536	3748	7623
mdsExamplesClause8-10.xml	8179	2081	1389	2416

Ahora describimos rápidamente las operaciones llevadas a cabo de acuerdo con el invento para comprimir el contenido de las hojas de un árbol jerárquico.

La operación 1 consiste en asociar una técnica de codificación por compresión a un tipo de contenido. Por ejemplo, la cuantificación lineal puede ser asociada a valores de coma flotante.

En la operación 2, un subárbol es identificado dentro del árbol jerárquico correspondiente a la estructura del documento XML considerado.

La operación 3 consiste en asignar una técnica de codificación por compresión al subárbol identificado.

La operación 4 consiste entonces en comprobar si el codec que pone en práctica la técnica de codificación por compresión está o no activada, si no, no se consigue compresión (5) de las hojas en el subárbol.

Si así sucede, el invento emplea (6) la compresión del contenido de las hojas del subárbol cuyo contenido es del tipo de contenido asociado (1) a la técnica de codificación por compresión.

Anexo 1

```
<schema  targetNamespace = "http://www.mpeg7.org/2001/BiMCoding"
    xmlns:cc =  http://www.mpeg7.org/2001/BiMCoding
    xmlns = "http://www.w3.org/2000/10/XMLSchema">
    <element name = "modifyContext">
      <complexTyp>
       <sequence>
         <element name = "allowsSkip" minOccurs ="0">
           <simpleType>
             <restriction base = "string">
              <enumeration value="mandatory">
              <enumeration value="optional"/>
              <enumeration value="forbidden"/>
            </restriction>
         </simpleType>
       </element>
       <element name="schemaMode[n] minOccurs="0">
          <simpleType>
             <restriction base="string">
                <enumeration value="mono"/>
                <enumeration value="multi"/>
            </restriction>
          </simpleType>
        </element>
       </sequence>
      </complexType>
     </element>
    </schema>
```

Anexo 2

```
<schema targetNamespace=http://www.mpeg7.org/2001/BiMCoding
xmlns:cc=" http://www .mpeg7.org/2001/BiMCoding"
xmlns="http://www.w3.org/2000/10/XMLSchema"
>
      <element name="context">
        <complexType>
          <sequence>
            <element name="allowsSkip" minOccurs="0">
              <simpleType>
                <restriction base="string">
                  <enumeration
value="mandatory"/>
                  <enumeration
value="optional"/>
                  <enumeration
value="forbidden"/>
                </restriction>
            </simpletype>
       </element>

   <element name="schemaMode" minOccurs="0">
     <simpleType>
       <restriction base="string">
         <enumeration value="mono"/>
         <enumeration value="multi"/>
       </restriction>
```

```
      </simpletype>
    </element>
            <element name="codecTypeMappers"
minOccurs="0">
              <complexType>
                 <sequence>
                    <element
name="codecTypeMapper"
      type="cc:codecTypeMapper"
                                        maxOccurs="unbounded"/>
                    </sequence>
                 </complexType>
              </element>

          </sequence>
</complexType>
      </element>
    <!-- puede ser indicado un tipo con la ayuda de un prefijo
de Esquema XML a fin de conocer el esquema al que pertenece-->
    <simpleType name="coupleSchemaTypeType">
         <restriction base="string"/>
    </simpleType>
    <!—restricción de 32 codecTypeMappers como máximo en un
 contexto -->
    <simpleType name="codecIDType">
           <restriction base="integer">
               <minInclusive="0"/>
               <maxInclusive="31"/>
           </restriction>
        </simpleType>

        <complexType name="codecTypeMapperType">
            <sequence>
              <element name="type" type^^coupleSchemaTypeType"
 maxOccurs="unbounded"/>
              <element name»"codec" type="cc:codecType"/>
            </sequence >
            <attribute name="id" use="required"
```

```
type="codecIType"/>
                <attribute name="State">
                    <simpleType>
                        <restriction base="string">
                          <enumeration value="activated"/>
                          <enumeration value="deactivated"/>
                        </restriction>
                    </simpleType>
                </attribute>
            </complexType>
            <complexType name="codecType" abstract="true"/>
</schema>
```

Anexo 3

```
<Example xmlns:cc="http://www.mpeg7.org/2001/BiMCoding">
     <AudioEnvelope>
         <cc:modifyContext>
            <codingProperties>
               <codingProperty id="1" state="desactivated">>
                 <type>audioFloat</type>
                  <codec xsi:type="LinearQuantifierType">
                     <bitSize>8</bitSize>
                     <minInclusive>-1.0</minInclusive>
                     <maxInclusive>1.0</maxInclusive>
                  </codec>
               </codingProperty>
            </codingProperties>
         </cc:modifyContext>
      <Values>
         <Raw>
           -0,25411 0.88541 0.2141946 0.3652541 -0.148941 0.8814
0,145544-0.847
         </Raw>
       <Values>
    <(AudioEnvelope>
</Example>
```

Anexo 4

```
<Example xmlns:cc="http://www.mpeg7.org/2001/BiMCoding">
    <AudioEnvelope>
       <cc:modifyContext>
          <codingProperties>
            <codingProperty id="1" state="deactivated">>
              <type>audioFloat</type>
               <codec xsi:type="LinearQuantifierType">
                  <bitSize>8</bitSize>
                  <minInclusive>-1.0</minInclusive>
                  <maxInclusive>1.0</maxInclusive>
               </codec>
            </codingProperty>
       </codingProperties>
   </cc:modifyContext>
<Values>
   <Raw>  <!-- Cuantificación aquí -->
      </cc:modifyContext>
         </codingProperties>
            <codingProperty id="1" state="activated"/>
         </codingProperties>
      </cc:modifyContext>
       -0,25411 0.88541 0.2141946 0.3652541 -0.148941 0.8814 0,145544
-0.847
       </Raw>
       <Variance> <!-pero no cuantificación aquí -- >
      0.1777441 0.2094511 0.349411 0.548444 -0.445445 -0.3654847
0.9451
       </Variance>
                        </Values>
                    </AudioEnvelope>
                     </Example>
```

Anexo 5

```
<complexType name=MLinearQuantizerCodecType”>
<complexContent>
    <extension base=”cc:CodecType">
       <element name="bitSize”>
          <SimpleType>
             <restriction base="int”>
                <minInclusive value=”1”/>
                <maxInclusive value=”32”/>
             <restriction>
             <SimpleType>
       </element>
       <element name=”minInclusive” value=”float”/>
       <element name="maxInclusive” value=”float”/>
    </extension>
  </complexContent>
<complexType>
```

Anexo 6

```
<Example xmlns:cc=http://www.mpg7.org/2002/BiMCoding>
   <AudioEnvelope>
      <cc:modifyContext>
         <codecTypeMappers>
            <codecTypeMappers> id=”1” state=”deactivated”>
              <type>audioFloat</type>
              <codec xsi:type=”LinearQuantizerCodecType”>
                  <bitSize>8<c/bitSize>
                  <minInclusive>-1.0</minInclusive>
                  <maxInclusive>1.0</maxInclusive>
              </codec>
            <codecTypeMapper>
```

```
            </codecTypeMappers>
          </cc:modifyContext>
      <Values>
         <Raw> <! - cuantificación aquí -->
            <cc:modifyContext>
               <codecTypeMappers>
                  <codecTypeMappers> id="1" state="activated">
               <codecTypeMappers>
             </cc:modifyContext>
        -0.25411 0.88541 0.2141946 0.3652541 -0.148941 0.8814
0.145544 -0.847
      </Raw>
      <Variance> > <!- pero no cuantificación aquí -->
0.1777441 0.2094511 O.349411 O.548444 -0.445445 -0.3654847
0.9541
      </Variance>
    </Values>
</AudioEnvelope>

                          </Example>
```

Anexo 7

```
            <complexType name="ZlibCodecType">
                  <complexContent>
                        <extension base="cc:CodecType"/>
                  <complexContent>
            </complexType>
```

Anexo 8

```
<MdsExample/Test          xmlns="http://www.mpeg7.org/2001/MPEG-
7_Schema"
   xmlns:cc="http://www.mpeg7.org/2001/BiMCoding"
   <cc:modifyContext>
       <codecTypeMappers>
          <codecTypeMapper id="1">
            <type>string</type>
            <type>anyURI</type>
            <codec xsi:type="ZlibCodecType"/>
          </codecTypeMapper>
        </codecTypeMappers>
     </cc:modifyCoDtext>

<!-los atributos termId, elementos de Nombre y elementos de
Definición serán cogidos por el ZlibCodecType -->

<ClassificationScheme uri="urn:i:mpeg;MPEG7AudioDomainCS="/..">
   <Term termid="1">
     <Name xml:lang="en">Source</Name>
     <Definition xml:lang="en">Type of audio source</Definition>
     <Term termid="1">
        <Name xml:lang="en">Synthetic</Name>
     </Term>
     <Term termid="1.2">
        <Namexml:lang="en">Natural</Name>
     </Term>
   </Term>
   <Term termid="2">
       <Namexml:lang="en">Acquisition</Name>
     <Definition xml:lang="en">Type of Content</Definition>
     <Term termid="2.1">
        <Name xml:lang="en">Music</Name>
     </Term>
     <Term termid="2.2">
        <Namexml:lang="en">Speech</Name>
     </Term>
     <Term termid="2.3">
```

```
        <Namexml:lang="en">Mixed</Name>
      </Term>
      <Term termid="2.4">
        <Namexml:lang="en">Multi-track</Name>
      <Term>
    <Term>
```

Anexo 9

Referencias

1- MPEG-7 Systems FCD, N4001, MPEG Reunión de Singapore. Marzo de 2001.

3- ISO/IEC 14496-1, MPEG-4 Systems, N3850.

4- La Zlib API, http://www.gzip.org/zlib/, RFC 1950, RFC 1951, RFC 1952 disponible en http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1950.html

REIVINDICACIONES

1. Un método para comprimir un árbol jerárquico que describe una señal multimedia, comprendiendo dicho árbol nodos y hojas, que pueden estar asociados a datos de al menos dos naturalezas distintas, llamados tipos de datos, en el que dicho método pone en práctica una compresión de datos para al menos algunas de dichas hojas, y en el que dicho método comprende: una operación de identificar (2) al menos un subárbol; una operación de asignar (3) una técnica de codificación por compresión asignada a dicho subárbol sólo para las hojas de dicho subárbol cuyos datos son del tipo asociado a dicha técnica de codificación por compresión, mientras las otras hojas de dicho subárbol no sufren ninguna codificación por compresión.

2. Un método según la reivindicación 1ª, que emplea una descripción paramétrica de dichas técnicas de codificación por compresión.

3. Un método según cualquiera de las reivindicaciones 1ª y 2ª, que comprende también una operación de comprimir la estructura de dicho árbol.

4. Un método según cualquiera de las reivindicaciones 1ª a 3ª, en el que dicho árbol es del tipo BiM (MPEG binario) de acuerdo con el estándar MPEG7.

5. Un método según la reivindicación 4ª, en el que dichos tipos de datos son tipos simples de acuerdo con el estándar MPEG7.

6. Un método según cualquiera de las reivindicaciones 1ª a 5ª, en el que una de dichas técnicas de codificación por compresión emplea la cuantificación lineal.

7. Un método según cualquiera de las reivindicaciones 1ª a 6ª, en el que en una de dichas técnicas de codificación por compresión emplea un algoritmo de compresión estadística.

8. Un método según la reivindicación 7ª, en el que dicho algoritmo es del tipo GZip.

9. Un método según cualquiera de las reivindicaciones 7ª y 8ª, en el que dicho algoritmo es empleado simultáneamente para un conjunto de datos correspondiente a los datos de al menos dos hojas.

10. Un método según cualquiera de las reivindicaciones 1ª a 9ª, en el que dicho árbol representa la estructura de un tipo de documento XML (lenguaje Markup extendido).

11. Un método según cualquiera de las reivindicaciones 1ª a 10ª, que comprende también una operación de asociar al menos un contexto de codificación a dicho subárbol, comprendiendo dicho contexto de codificación piezas de información que permiten omitir dicho subárbol mientras descodifica dicho árbol jerárquico.

12. Un método según la reivindicación 11ª, en el que dichas piezas de información comprende: una pieza de información que indica la técnica de codificación por compresión usada; y/o una pieza de información que indica si el subárbol correspondiente ha sido comprimido; y/o una pieza de información que indica si el subárbol correspondiente es omitible; y/o una pieza de información que indica que al menos, un parámetro de la técnica de codificación por compresión usada ha sido modificado.

13. Un método para descodificar una señal multimedia comprimida de acuerdo con el método de cualquiera de las reivindicaciones 1ª a 12ª.

14. Un método según la reivindicación 13ª, que pone en práctica una operación de refrescar un contexto de descodificación actual de acuerdo con la información de contexto de codificación transportada por dicha señal.

15. Un método según la reivindicación 14ª, en el que dicho contexto presente define al menos un tipo de datos, comprendiendo dicho método una operación de emplear una técnica de descodificación por compresión asociada a dicho tipo de datos para las hojas que comprenden datos de dicho tipo de datos.

16. Una señal representativa de un árbol jerárquico comprimido según el método de cualquiera de las reivindicaciones 1ª a 12ª, comprendiendo dicho árbol nodos y hojas, que pueden ser asociados a datos de al menos dos naturalezas distintas, llamados tipos de datos, comprendiendo también dicho árbol al menos un subárbol al que ha sido asignada una técnica de codificación por compresión para al menos un tipo de datos, en el que dicha señal comprende al menos un campo que contiene: las hojas de dicho subárbol cuyos datos son del tipo de datos asociados a dicha técnica de codificación por compresión y que han sufrido dicha codificación por compresión; las otras hojas de dicho subárbol cuyos datos no son del tipo de datos asociados a dicha técnica de codificación por compresión y que no han sufrido dicha codificación por compresión.

17. Una señal según la reivindicación 16ª, en el que dicha señal comprende también: al menos un campo que indica dicha técnica de codificación por compresión asignada a dicho o dichos subárboles para dicho al menos uno de dicho

tipo de datos; al menos un campo que indica al menos dicho tipo de datos asociado a dicha técnica de codificación por compresión para dicho o dichos subárboles.

18. Una señal según la reivindicación 17, en la que dicha señal comprende también al menos un campo que indica al menos un parámetro de dicha técnica de codificación por compresión asignada a dicho subárbol para al menos uno de dichos tipos de datos.

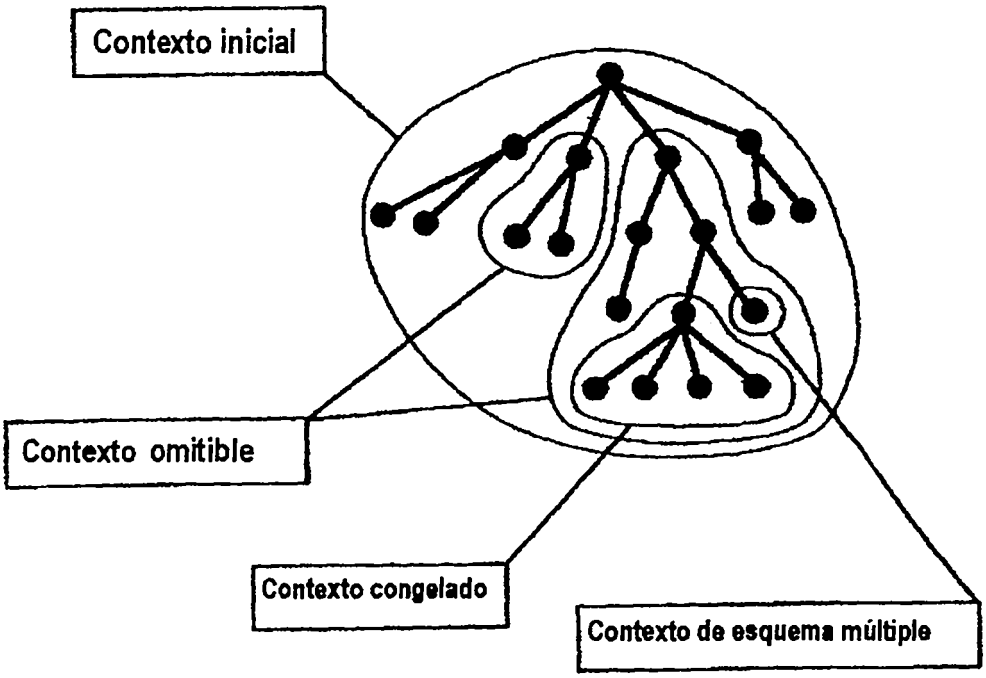
Contexto inicial
Contexto omitible
Contexto congelado
Contexto de esquema múltiple

Fig. 1


MC
C
Elemento

Fig. 2

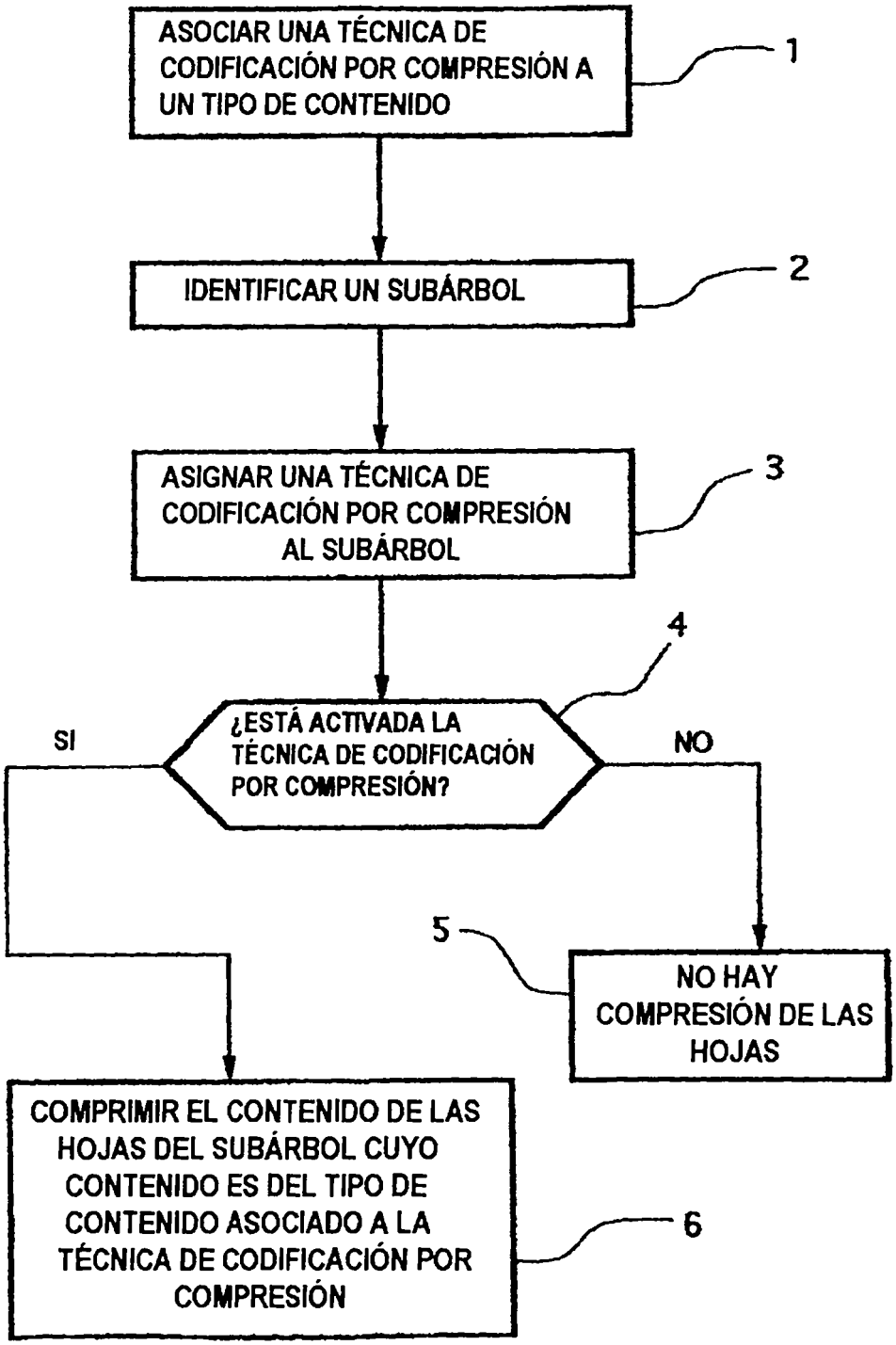
ASOCIAR UNA TÉCNICA DE CODIFICACIÓN POR COMPRESIÓN A UN TIPO DE CONTENIDO
1
IDENTIFICAR UN SUBÁRBOL
2
ASIGNAR UNA TÉCNICA DE CODIFICACIÓN POR COMPRESIÓN AL SUBÁRBOL
3
4
¿ESTÁ ACTIVADA LA TÉCNICA DE CODIFICACIÓN POR COMPRESIÓN?
SI
NO
5
NO HAY COMPRESIÓN DE LAS HOJAS
COMPRIMIR EL CONTENIDO DE LAS HOJAS DEL SUBÁRBOL CUYO CONTENIDO ES DEL TIPO DE CONTENIDO ASOCIADO A LA TÉCNICA DE CODIFICACIÓN POR COMPRESIÓN
6

Fig. 3