(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 558 967**

(51) Int. CI.:

H04N 21/235 (2011.01)
H04N 21/434 (2011.01)
H04N 21/435 (2011.01)
H04N 21/443 (2011.01)
H04N 21/81 (2011.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **05.04.2001 E 01924699 (0)**

(97) Fecha y número de publicación de la concesión europea: **04.11.2015 EP 1281279**

(54) Título: **Motor de tratamiento de datos genérico**

(30) Prioridad:

06.04.2000 US 195579 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
09.02.2016

(73) Titular/es:

OPENTV, INC. (100.0%)
275 Sacramento Street
San Francisco, CA 94111, US

(72) Inventor/es:

DUREAU, VINCENT;
HENSGEN, DEBRA;
PIERRE, LUDOVIC y
MENAND, JEAN-RENE

(74) Agente/Representante:

TOMAS GIL, Tesifonte Enrique

DESCRIPCIÓN

Motor de tratamiento de datos genérico

Campo de la invención

[0001] La presente invención se refiere en general a un motor genérico para el tratamiento de datos, y más particularmente, a un sistema y método para un motor genérico para el tratamiento de peticiones de aplicación de datos formateados, tales como información relacionada con la televisión en un sistema de televisión interactiva.

Fondo

[0002] US 5,844,620 describe una guía interactiva de interfaz visual en pantalla que guía a un usuario a través de un menú de eventos individuales disponible vía una red informática.

[0003] US 5,847,751 describe una arquitectura de red para entrega de servicios digitales y de emisión interactivos por un sistema de distribución híbrido de fibra-coaxial.

[0004] US 5,734,589 describe un terminal de entretenimiento digital que incluye un módulo de interfaz de red que junta el terminal a un tipo específico de red de comunicación para recibir un canal de ancho de banda digital.

[0005] Un proveedor de servicio de radiotransmisión transmite corrientes audio-video a una televisión del espectador.
Sistemas de televisión interactiva son capaces de mostrar texto e imágenes gráficas además de programas audio-video típicos.
También pueden proporcionar varios servicios, tales como comercio vía la televisión, y otras aplicaciones interactivas a espectadores.
La señal de televisión interactiva puede incluir una parte interactiva consistente en código de aplicación, datos, y señalización de información, además de partes audio-video.
La abreviatura "SI" en esta aplicación se utiliza para referirse tanto a señalización de información como a cualquiera de la aplicación de datos que se envía según un formato rígido.
El SI puede incluir información tal como tiempos o canales en los que un programa de televisión particular será mostrado, el género de un programa particular, o información identificando qué flujo elemental llevará el audio para un programa particular en un lenguaje particular.
Esta información se puede combinar en una única señal o diferentes señales para su transmisión a un receptor conectado a la televisión del espectador o el proveedor puede incluir solo un subconjunto de la información, posiblemente con localizadores de recurso.
Tales localizadores de recurso pueden utilizarse para indicar fuentes alternativas de información interactiva y/o audio-video.
Por ejemplo, el localizador de recurso podría coger la forma de un red Localizador Universal de Recursos (URL) en la red informática mundial.

[0006] La señal televisiva es generalmente comprimida antes de su transmisión y transmitida a través de medios de emisión típicos tales como líneas de televisión por cable (CATV) o sistemas de transmisión por satélite directos.
Información referenciada por localizadores de recursos se pueden obtener por medios diferentes, por ejemplo, a través de un canal de regreso siempre encendido, tal como un módem DOCSIS.

[0007] Un decodificador de recepciones integrado (IRD) controla la funcionalidad interactiva de la televisión.
El IRD recibe la señal, separa la parte interactiva de la parte audio-video, y descomprime las partes respectivas de la señal.
Los IRD usan alguna de la información interactiva para ejecutar una aplicación mientras alguna de la información audio-video se transmite a la televisión.

[0008] Un motor SI se ejecuta dentro de un IRD, filtrando las corrientes de emisión, extrayendo información solicitada por aplicaciones, y entregando información a aplicaciones.
Tales motores SI son típicamente construidos para uso con una especificación de SI particular que se diseña para un cable, satélite, RF u otro sistema particular.
Esto es, el código para el motor SI se confecciona directamente al formato de SI usado por ese sistema.
En respuesta a una solicitud de datos de la aplicación, el motor SI establece máscaras para filtros, modifica máscaras, recibe información de los filtros, y devuelve la información a las aplicaciones.
La codificación de SI en el flujo de datos depende del formato usado en un sistema particular, y varía típicamente de un sistema a otro, al igual que varía lentamente a lo largo del tiempo en el mismo sistema.

[0009] Así, si un sistema diferente debe usar una especificación de SI diferente, un motor nuevo, posiblemente derivado de un motor existente, debe ser construido.

Especificaciones de SI son frecuentemente modificadas después de que un sistema se opere, con el propósito de, por ejemplo, proporcionar funcionalidad adicional.
En estos casos, el motor SI en tales sistemas debe ser dinámicamente mejorado.
Como el motor SI es típicamente incorporado en el sistema operativo o el software personalizado que se ejecuta en el IRD, la instalación y/o modificación son logísticamente complejas, frecuentemente costosas y ciertamente llevan mucho tiempo.

[0010] También es posible siempre transmitir información de formo (tales como identificativos de HTLM) junto con datos formateados, no obstante, esta solución no es viable cuando el ancho de banda está limitado, como es el caso con metadatos relacionados con la televisión, porque en tales casos, la transmisión de los propios datos de formato cada vez que los datos formateados son enviados requeriría un orden de magnitud de más ancho de banda.

[0011] Hay una necesidad, por lo tanto, de un motor SI mejorado capaz de tratar cualquier formato de SI, que se pueda mejorar fácilmente y sin requerir uso continuo de precioso ancho de banda en el sistema de radiodifusión.

Resumen de la invención

[0012] Según un primer aspecto de la invención se proporciona un receptor para el tratamiento de datos donde dicho receptor comprende un motor de procesamiento de datos genérico configurado para: recibir una definición de formato, donde dicha definición de formato comprende una descripción de una gramática que define una sintaxis de un lenguaje objetivo; configurar dicho motor para procesar datos que se formatean según la definición de formato, sensible a recibir la definición de formato; reciben datos adicionales que conforman el lenguaje objetivo; y procesan los datos adicionalmente recibidos conforme a la definición de formato.

[0013] Según un segundo aspecto de la invención se proporciona un sistema para tratar datos formateados, que comprende: un transmisor configurado para transmitir una definición de formato asociada a los datos, donde dicha definición de formato comprende una descripción de una gramática que define una sintaxis de un lenguaje objetivo; y el receptor según el primer aspecto.

[0014] Según un tercer aspecto de la invención se proporciona un método para actualizar un motor de tratamiento de datos genérico3 operable para procesar datos independientes de información de formateo, que comprende: recibir una definición de formato, donde dicha definición de formato comprende una descripción de una gramática que define una sintaxis de un lenguaje objetivo; configurar dicho motor para procesar datos que se formatean según la definición de formato, sensible a recibir la definición de formato; datos adicionales de recepción que se ajustan al lenguaje objetivo; y tratar los datos adicionalmente recibidos conforme a la definición de formato.

[0015] Según un cuarto aspecto de la invención se proporciona un producto de programa informático para tratar datos formateados, que comprende un medio utilizable informático con código legible por máquina concretado en él para: recibir una definición de formato, donde dicha definición de formato comprende una descripción de una gramática que define una sintaxis de un lenguaje objetivo; configurar un procesamiento de datos de motor sensible a recibir la definición de formato; recibir datos adicionales que se ajustan al lenguaje objetivo; y tratar los datos adicionalmente recibidos conforme a la definición de formato.

[0016] Un motor de tratamiento de datos genérico3 es operable para recibir una definición de formato y datos de tratamiento formateados según la definición, sin uso de información de formateo en los datos.

[0017] En una forma de realización, la definición de formato incluye una descripción de la sintaxis del formato, y una descripción de la semántica del formato.
La sintaxis y semántica puede ser descrita en el mismo lenguaje o en diferentes lenguajes, y el motor se configura para producir una representación interna de la sintaxis y semántica.

[0018] El motor se puede configurar para recibir preguntas y las usa con la representación interna para ajustar máscaras para los filtros.
Los filtros aplican las máscaras a los datos y devuelven datos filtrados al motor, que pueden pasar una parte de los datos filtrados a aplicaciones, almacenar una parte de los datos filtrados, establecer nuevas máscaras basado en una parte de los datos filtrados, o modificar las máscaras existentes basado en una parte de los datos filtrados.
Los filtros también se puede configurar para devolver datos filtrados directamente a aplicaciones, desviando el motor.

[0019] Métodos y productos de programa de ordenador conforme a lo mencionado anteriormente son también descritos.

[0020] Otras características, ventajas, y formas de realización de la invención será aparentes a los expertos en la técnica por la siguiente descripción, dibujos, y reivindicaciones.

Breve descripción de los dibujos

[0021]

FIG. 1 es un diagrama que ilustra la corriente práctica de sustitución de software del motor SI cuando una versión diferente se necesita para procesar una especificación de SI diferente;
FIG. 2 es un diagrama que ilustra un proceso previsto para cambios de motor SI para nuevos "mercados libres para la emisión horizontal";
FIG. 3 es un diagrama que ilustra el motor SI genérico y su reconfiguración para usar una especificación de SI nueva;
FIG. 4 es un diagrama que ilustra la distribución de programas de televisión y señalización de información de una emisora de radiodifusión a una estación de recepción;
FIG. 5 es un diagrama que ilustra un decodificador de señales digitales que incorpora un motor SI genérico en una forma de realización de la invención;
FIG. 6 es un diagrama que ilustra una forma de realización de los componentes funcionales de un motor SI genérico y su interacción;
FIG. 7 ilustra una forma de realización de un lenguaje de especificación de sintaxis de SI;
FIG. 8 ilustra una forma de realización de una especificación de la sintaxis para parte de un formato de SI;
FIG. 9 ilustra una forma de realización de una estructura de los datos que se puede utilizar para almacenar una representación interna de una descripción de formato de sintaxis de SI;
FIG 10a y 10b ilustran una forma de realización de parte de una especificación del sistema particular para la semántica de su SI;
FIG 11a y 11b ilustran una forma de realización de una gramática que define la sintaxis para los no terminales de parte de un lenguaje de semántica de SI particular;
FIG 12a, 12b, y 12c ilustran una forma de realización de una estructura de datos que se puede utilizar para almacenar una representación interna de una descripción de formato de semántica de SI;
FIG. 13 ilustra una forma de realización de una solicitud de aplicación para información de SI que utiliza un lenguaje de pregunta de bajo nivel;
FIG. 14 es un diagrama de bloques que muestra la relación entre diferentes estructuras de SI descritas en la solicitud de aplicación mostrada en la FIG. 13;
FIG. 15 ilustra una forma de realización donde semántica compleja de una es son especifica en una solicitud de aplicación;
FIG. 16 ilustra una forma de realización donde Prolog se utiliza para definir la semántica de parte de un SI del sistema;
Figuras 17a = 17f ilustran especificaciones en una forma de realización de semántica para un formato de SI complejo;
FIG. 18 ilustra una forma de realización de una solicitud de aplicación expresada utilizando el formato de SI cuya semántica se definen en figuras 17a - 17f; y FIG. 19 ilustra un flujo de proceso conforme a la invención.

[0022] Caracteres de referencia correspondientes indican partes correspondientes en todas las vistas de los dibujos.

Descripción detallada de la invención

[0023] La siguiente descripción se presenta para permitir un experto en la técnica a hacer y usar la invención.
Descripciones de formas de realización específicas y aplicaciones se proporcionan solo como ejemplo y varias modificaciones serán fácilmente aparente a los expertos en la técnica. Los principios generales descritos aquí se pueden aplicar a otras formas de realización y aplicaciones sin apartarse del ámbito de la invención.
Así, la presente invención no se debe limitar a las formas de realización mostradas, pero debe ser acordado el alcance más ancho de acuerdo con los principios y características descritos aquí.
Será entendido por un experto en la técnica que muchas formas de realización son posibles, tal como el uso de una configuración y pantalla para ejecutar las funciones y características descritas aquí.
Para fines de claridad, la invención se describe en su aplicación a un decodificador de señales digitales usado con una televisión, y detalles relacionadas con material técnico que se conocen en los campos técnicos relacionados con la invención no han sido incluidos.

Visión de conjunto

[0024] Como se describe aquí, la presente invención se refiere a un motor para tratar datos rígidamente formateados.
Por medio de ilustración no limitativa, la invención se describe en su aplicación para tratar metadatos relacionados con las difusiones de televisión (tales como SI) en un decodificador de receptor integrado (IRD), que puede, por ejemplo, ser implementado dentro de una televisión, incorporado con un aparato de vídeo personal (PVR), o estar en un descodificador separado.
El término "metadatos" como se utiliza en este caso debería entenderse para referirse a cualquier tipo de datos formateados, y los principios de la invención pueden recurrir a otras aplicaciones implicando el uso de datos formateados que necesariamente no se refieren a televisión, tales como datos de previsiones meteorológicas.

También, datos se pueden transmitir por otros medios además de la radiodifusión, tales como multidifusión y conexiones punto a punto.
Descritos aquí son un método y sistema para un motor SI genérico que trata solicitudes de aplicación para metadatos formateados relacionados con la televisión.

[0025] Motores SI se utilizan para sostener aplicaciones de televisión interactiva digital.
El motor SI y las aplicaciones interactivas digitales se ejecutan en un decodificador de recepciones integrado (IRD).
Como declarado por encima, el IRD se puede implementar en un decodificador de señales digitales, en una televisión, u otro dispositivo.
Aplicaciones de televisión interactiva digital frecuentemente requieren acceso a información al corriente que está siendo enviada por un emisor u operador del sistema, tales como tiempos o canales donde un programa de televisión particular será mostrado, el género de un programa particular, o información que identifica qué flujo elemental llevará el audio en un lenguaje particular.
SI también puede incluir cualquier dato cuyo fin es describir otros datos o contenido televisivo.

[0026] Típicamente, SI es enviado, introducido en el flujo de transmisión, según un formato rígido.
SI no es autodescriptor; es decir, no hay información introducida en el SI que describe su formato, tales como etiquetas.
En mercados donde el emisor u operador de red provee el IRD, cada emisor u operador de red en última instancia decide el formato particular usado para SI para su red, mientras que en mercados donde el consumidor pueden tener comprados cualquier número de IRD disponibles comercialmente, múltiples formatos de SI se pueden utilizar al mismo tiempo.
Normalmente, un formato se basa en una de las diferentes especificaciones de formato de SI que han sido desarrolladas por organizaciones estándar, tales como DVB (Radiodifusión de Vídeo Digital) o ATSC (Comité de Sistemas de Televisión Avanzada).
Estas especificaciones han sido escritas de modo que pueden fácilmente ser extendidas paran adaptar necesidades adicionales de otros, tales como emisoras individuales o subcomisiones de la misma organización estándar que están tratando problemas diferentes.
Esta flexibilidad es normalmente proporcionada reservando alguna secuencia de bits en varias posiciones para definir más tarde, es decir, si un comité ha visto una necesidad para solo 3 valores diferentes, puede permitir a 3 o 4 bits permitir hasta 5 o 13 usos adicionales, respectivamente, para un campo dado.
No obstante, una vez un valor se elige para significar un uso particular, el formato es rígid que hasta usos nuevos son agregados.
Cuando nuevos usos son agregados, el formato debe ser cambiado para reflejar la asignación de valores a los usos nuevos.

[0027] Frecuentemente, los decodificadores de receptor integrado tienen filtros que pueden utilizarse para rápidamente buscar a través de flujos de datos la presencia de modelos de datos particulares.
Motores SI típicamente establecen máscaras para estos filtros.
Una máscara se utiliza para describir un modelo de datos particulares.
Por ejemplo, la máscara binaria 111xx000 identifica el modelo de tres I, seguido de cualquier dos bits, seguidos de tres 0s.
Además de designar un modelo, una máscara puede designar posición; por ejemplo, estipulando que tal modelo binario debe ocurrir a principios de un paquete dimensionado fijo.

[0028] SI se envía en un formato rígido por lo que los filtros, frecuentemente implementados en el hardware, pueden separar eficazmente la información deseada por la aplicación o espectador que utiliza el IRD.
Otro IRD siendo usado por otros espectadores puede filtrar en busca se otras información, dependiendo de las preferencias de los espectadores y/o aplicaciones que son ejecutadas.
Típicamente, el espectador interactúa con el IRD por presión de teclas en el control remoto o teclado, que a su vez causa que la información sea entregada a software de nivel de aplicación ejecutándose en un procesador en el IRD.
Para cumplir una solicitud del espectador, el software de aplicación puede precisar acceder a algunos de datos de la parte de SI de un flujo que está siendo recibidos.
El software de aplicación luego solicitaría estos datos del motor SI que se ejecuta en el IRD.

[0029] Como el software de nivel de aplicación, el motor SI puede ser software, sin embargo típicamente a a nivel de OS o software personalizado antes que a nivel de aplicación, que se ejecuta en el IRD, aunque éste también se puede implementar en el hardware.
Su función es de asistir al software de nivel de aplicación en la obtención eficaz de información de SI del flujo transmitido.
Cuando un motor SI recibe una solicitud de un programa de aplicación, usará los filtros subyacentes (que se pueden implementar en el hardware) para obtener la información solicitada por la aplicación.

[0030] Por ejemplo, la aplicación puede solicitar los nombres de todas las películas que se emitirán entre 9 p.m. del día actual a 1 a.m. del día siguiente en un conjunto de 16 canales diferentes, numerados del 16 ale 31.

En un determinado formato de SI, este puede traducir a un conjunto de modelos de bits posible en los primeros 13 bytes de una estructura MPEG-2 junto con ciertas estructuras más atrás en el paquete.
Un IRD particular puede contener hardware de fin especial que es capaz de filtrar en los primeros 8 bytes de un paquete.
El motor SI podría luego crear un conjunto de máscaras para proporcionar a los filtros de modo que los filtros quisieran descartar cualquier paquete que no coincidiera con uno de los modelos de 8-byte viables de los elementos del conjunto deseado.
Esto reduce el número de paquetes enteros que el motor SI mismo necesitaría procesar.
Esto requiere que el motor SI entienda la estructura rígida donde SI se envía por una red dada, al igual que el significado detrás modelos de bit particular aparece en esa estructura.

[0031] SI es típicamente transmitido por emisoras televisivas junto con vídeo, audio, y otra información privada.
Las estructuras de datos donde el SI es contenido están en un estado de flujo, debido a mejora y cambios por organismos de estándares internacionales y emisoras, y en algunos casos, estructuras nuevas en su totalidad son definidas, dando como resultado cambios en la especificación de SI.
El motor SI debe ser capaz de manipular datos en la especificación de SI cambiada.
Como será descrito aquí, la presente invención proporciona un motor SI flexible capaz de manipular cualquier especificación de SI transmitida conforme a la invención, lo que le permite ser fácilmente mejorado a una especificación de SI revisada.

[0032] Aunque la aplicación del motor SI en un sistema de transmisión televisiva puede no requerir que sea capaz de tratamiento de más de uno de formato de SI a la vez, un motor SI genérico que se puede configurar para procesar cualquier formato de SI tiene diferentes ventajas.
Tal software puede ser más íntegramente evaluado que su homólogo no genérico, y más importante, es sustancialmente más fácil de mejorar a versiones nuevas del formato de SI.
Además, el tiempo de definición de un formato de SI nuevo al uso de este formato puede ser sustancialmente acortado.
Con motores SI típicos, cualquier cambio a la definición de SI necesita un cambian al software de sistema que se ejecuta en el decodificador de recepciones integrado (IRD).
Así, las emisoras no pueden usar una definición de SI nueva hasta que: software nuevo (i) ha sido diseñado, escrito, y evaluado; y (ii) todos decodificadores de receptor integrados han sido mejorados, incluyendo aquellos actualmente en uso por clientes.

[0033] FIG. 1 ilustra una situación que no usa el motor SI genérico para un mercado vertical (un mercado donde el IRD es provisto por el operador del sistema o emisor).
Aquí al menos el software para el motor SI debe ser reinstalado.
La dificultad de mejoramiento del software en un entorno de difusión es compuesto por la naturaleza del sistema.
El software por ser reinstalado debe ser continuamente difundido si no hay mecanismo disponible para permitir la descarga vía un canal de regreso.
Aunque haya un canal de regreso, una señal para indicar la disponibilidad de la nueva versión debe ser emitida reiteradamente porque no todo IRD será necesariamente encendido al mismo tiempo.
Además, el uso del nuevo formato posible sería retrasado hasta que un porcentaje sustancial del IRD hubiera sido mejorado.

[0034] En un método conocido como el "mercado horizontal," los fabricantes de decodificadores receptores integrados fabricarán y venderán decodificadores-operadores independientes.
Estos decodificadores será útiles para cualquier consumidor, independientemente del operador siendo usado para el consumidor.
No obstante, este método se complica por el deseo de las emisoras de difundir su propia señalización que no puede ser completamente estándar, y así el fabricante tendría que producir un motor SI diferente para cada especificación de SI con el cual éste desea ser compatible.
Como ilustrado en FIG. 2, cuando diferentes emisoras han modificado estándares para sistemas suyos, el motor SI debe ser capaz de alojar varios formatos de SI simultáneamente.
Adicionalmente, si una especificación de SI nuevo es más tarde introducida o una especificación existente es actualizada, el decodificador no tendrá un motor SI para procesar la nueva especificación de SI, y se deben actualizar con software del motor SI nuevo de la manera ilustrado por FIG. 1.
En otras palabras, este escenario requeriría que todos los formatos de SI fueran eventInfo definidos antes de construcción del motor SI, de lo contrario el motor SI debería ser actualizado, como se describe anteriormente.

[0035] Un motor SI genérico conforme a la invención habilita a un emisor u operador del sistema a configurar el motor SI de modo que pueda manejar la señalización del emisor, mediante la diseminación de una descripción del SI en un lenguaje entendido por el motor SI genérico.
Al recibir la descripción, el motor SI genérico se reconfigura a sí mismo para manejar la nueva señalización, como se muestra en FIG. 3.
Usando el motor SI reconfigurable inventivo, ya nos e requiere que el decodificador de recepciones integrado contenga software específico para manejar cada formato de señalización del operador.

Ningún código de fuente necesita ser escrito o modificado para utilizar un formato de SI nuevo, y el software para el motor SI no requiere modificación o sustitución.
Solo una descripción de la nueva sintaxis del formato de SI (estructura) y semántica (significado) debe ser provista al motor SI.
Esta información es típicamente mucho más pequeña que un motor SI nuevo mismo y es mucho más fácil de instalar que software nuevo.
Finalmente, la probabilidad de incompatibilidades de introducción con software instalado existente es eliminado, debido a que el software mismo no necesita ser cambiado.

[0036] En una forma de realización de la invención, el motor SI comprende una interfaz de aplicación, una interfaz de filtración, y una interfaz de especificación de formato.
La aplicación interfaz es responsable de recibir solicitudes de aplicaciones, y también se puede usar para devolver información a las aplicaciones.
La interfaz de filtración se utiliza para construir o modificar máscaras para filtros, que se pueden implementar en hardware o software.
Conforme los datos se reciben de un flujo de emisión (o vía otros medios tales como una conexión punto a punto) y procesado por los filtros, los datos extraídos por los filtros se pueden proporcionar al motor SI genérico o directamente a las aplicaciones.
Antes de proveer información obtenida vía los filtros a las aplicaciones, el motor SI genérico puede procesar esos datos, y, al hacer eso, puede ajustar máscaras adicionales o modificar máscaras existentes.
La interfaz de especificación de formato es capaz de recibir y tratar descripciones de nuevos formatos, que pueden más tarde ser usados por aplicaciones cuando éstas hacer solicitudes vía la interfaz de aplicación.
Los datos formateados y las especificaciones de formato se pueden introducir en un flujo de transmisión televisiva o ser transmitidos separadamente por otros medios tales como multidifusión o una conexión punto a punto.
La sintaxis y semántica de nuevos formatos se pueden transmitir separadamente del uno al otro o juntas.

[0037] Si una especificación de formato nuevo se usa, se puede transmitir al motor SI genérico, que será reconfigurado para usar la nueva especificación de formato.
La operación del motor SI genérico se describe aquí por referencia a su uso, en una forma de realización de la invención, como un componente de sistemas de televisión interactiva distribuidos.

Descripción detallada

[0038] En referencia a FIG. 4, un diagrama de una transmisión televisiva y sistema receptor se muestra y generalmente se indica a 10.
El sistema 10 incluye una emisora de radiodifusión 12 donde audio-video e información de control se ensamblan en forma de datos digitales y se aplican en señales digitales (que pueden también ser análogas) para su transmisión por satélite a una estación de recepción.
El emisor puede incluir metadatos rígidamente formateados relacionados con la televisión llamados SI.
El SI se introduce en el flujo de emisión.
El SI puede, por ejemplo, enumerar cada uno de los identificadores de flujo elemental y asociar con cada identificador una codificación que describe el tipo del flujo asociado (por ejemplo, si éste contiene vídeo o audio, qué perspectiva representa o qué lenguaje lenguaje se están llevando al flujo), información de programa televisivo tal como hora, fecha, y canal.
El SI se convierte por la emisora de radiodifusión a un formato adecuado para transmisión por medio de emisión.
Los datos se pueden formatear en paquetes, por ejemplo, que se pueden transmitir sobre una red de satélite digital 22, hilos de televisión por cable, líneas telefónicas, redes celulares, óptica de fibra, o cualquiera de los otros medios apropiados.
Los paquetes se pueden multiplexar con otros paquetes para transmisión.

[0039] La estación de recepción incluye un decodificador de recepciones integrado en forma de un decodificador de señales digitales 16, conectado a un dispositivo de almacenamiento 18 y una televisión 20 que se usa para presentar programas a un espectador, como se muestra en FIG. 5.
El decodificador de señales digitales 16 es operable para descomprimir los datos digitales.
Las señales de vídeo descomprimido se pueden convertir en señales analógicas tales como señales de formato para pantalla televisiva NTSC (comité de estándares de televisión nacional), o pueden estar en el formato digital para su uso por una pantalla de televisión digital.
Decodificador de señales digitales 16 comprende además un motor SI genérico 36, que comprende una interfaz de aplicación, una interfaz de filtración, y una interfaz de especificación de formato, como se describe en este caso.
Señales enviadas al decodificador de señales digitales 16 se filtran por la fase de transporte 28 bajo la dirección del motor SI genérico 36, y de aquellos que cumplen los requisitos de filtración, algunos se pueden usar por el procesador 30 inmediatamente, mientras otros se puede colocar en el almacenamiento local tal como RAM o dispositivo de almacenamiento 18.
Ejemplos de requisitos por los que necesitarían ser filtradas induyen un valor particular en la ubicación reservada para un identificador de flujo elemental o un identificador de red originante.

El decodificador de señales digitales 16 se puede usar para cubrir o combinar señales diferentes para formar la pantalla deseada en la televisión del espectador 20.

[0040] Las señales audio-video y señales de control de programa recibidas por el decodificador de señales digitales 16 corresponden a programas televisivos y selecciones de menú que el espectador puede acceder a través de una interfaz de usuario, al igual que aplicaciones que pueden ser ejecutadas, por ejemplo, interpretadas, por el procesador de control 30.
El espectador puede controlar el decodificador de señales digitales 16 a través de una unidad de control remoto infrarrojo, un panel de control en el decodificador de señales digitales, o un menú visualizado en la pantalla, por ejemplo.
Selecciones y entradas hechas por el espectador pueden a su vez causar que aplicaciones cambien sus requisitos de filtración, y envíen peticiones al motor SI 36 de cambiar las máscaras para los filtros y recibir información basada en los requisitos de filtración modificada.

[0041] El decodificador de señales digitales 16 puede ser capaz de vídeo de descodificación, audio, y datos.
En una forma de realización, puede ser un decodificador de señales digitales digital para uso con un receptor satélite o receptor de decodificador integrado satélite que sea capaz de decodificar vídeo MPEG, audio, y datos.
El decodificador de señales digitales 16 puede ser configurado, por ejemplo, para recibir canales digitales de video que soportan comunicaciones de banda ancha utilizando modulación de amplitud en cuadratura (QAM) y para controlar canales para de señalización y mensajería de dos direcciones.
Los canales QAM digital llevan corrientes de transporte MPEG (grupo experto de película cinematográfica) multiprograma comprimidas y codificadas.
Una fase de transporte 28 extrae el programa deseado del flujo de transporte y separa el audio, vídeo, y componentes de datos, que se dirigen a dispositivos que procesan las corrientes, tales como uno o varios decodificadores de audio, uno o varios decodificadores de vídeo, y opcionalmente a RAM (u otra forma de memoria) o un disco duro.
Debe entenderse que el decodificador de señales digitales 16 y dispositivo de almacenamiento 18 (así como cualquier dato y señal del proveedor de servicio de radiotransmisión), pueden ser análogos, digitales, o tanto análogo como digital.

[0042] Dispositivo de almacenamiento 18 es opcionalmente acoplado al decodificador de señales digitales 16.
El dispositivo de almacenamiento 18 se utiliza para proporcionar almacenamiento suficiente para grabar programas y datos que no encajan en la cantidad limitada de memoria principal (por ejemplo; RAM) típicamente disponible en decodificadores de señales digitales.
El dispositivo de almacenamiento 18 puede comprender cualquier dispositivo de almacenamiento adecuado, tal como una unidad de disco duro, una unidad de DVD grabable, cinta magnética, disco óptico, disco magnético-óptico, memoria flash, o memoria de estado sólido, por ejemplo.
El dispositivo de almacenamiento 18 puede ser interno al decodificador 16 o conectado externamente (por ejemplo, a través de una conexión IEEE 1394-1995), ya sea con una conexión permanente o una conexión extraíble.
Más de un dispositivo de almacenamiento 18 se puede unir al decodificador de señales digitales 16. El decodificador de señales digitales 16 y/o dispositivo de almacenamiento 18 también se puede incluir en un embalaje con la televisión 20.

[0043] El decodificador de señales digitales 16 incluye generalmente un procesador de control 30 compuesto de una unidad de control (por ejemplo; microprocesador), memoria principal (por ejemplo; RAM), y otros componentes que son necesarios para procesar la señal de televisión interactiva recibida.

[0044] Como se muestra en FIG. 5, el decodificador de señales digitales 16 incluye un extremo frontal 26 operable para recibir audio, vídeo, y otros datos de la emisora de radiodifusión 12.
La fuente de emisión se alimenta en el decodificador de señales digitales 16 en el extremo frontal 26, que comprende un convertidor análogo a digital (A/D) y sintonizador/demoduladores (no mostrados).
El extremo frontal 26 filtra una banda particular de frecuencias, la desmodula, y la convierte a un formato digital.
La salida digitalizada es luego enviada a una fase de transporte 28.
La fase de transporte 28 procesa más los datos, enviando de una parte de los datos a una fase audiovisual (AV) 34 para visualización y otra parte al procesador de control 30, y filtrando el resto de los datos.
La información de señal y control también se puede grabar como emisión junto con los datos audio-video o pueden ser manipulado antes por software en el decodificador de señales digitales 16.

[0045] Debe entenderse que el sistema 10 descrito aquí es solo un ejemplo de un sistema usado para transportar señales a la televisión 20.
El sistema de red emisión y decodificador de señales digitales 16 puede ser diferente al descrito aquí sin apartarse del ámbito de la invención.
Por ejemplo, distintos componentes representados en el decodificador de señales digitales 16 de FIG. 5 pueden ser combinados, como la colocación de motor SI 36 dentro de procesador 30 o parcialmente en la fase de transporte 28 y el procesador de control 30, o la integración del dispositivo de almacenamiento 18 dentro de decodificador de señales digitales 16.

El motor SI genérico

[0046] Construcción del motor SI genérico implica lo siguiente:

- definir/seleccionar un lenguaje para expresar la sintaxis del SI o metadatos formateados.
- definir/seleccionar un lenguaje para expresar la semántica de los metadatos de SI.

Este lenguaje puede ser el mismo lenguaje que el definido para la expresión de la sintaxis, una extensión del lenguaje definido para la expresión de la sintaxis, o un lenguaje diferente.

- definir/seleccionar un lenguaje para expresar preguntas de Si.

Este lenguaje puede ser el mismo lenguaje que el definido para la expresión de la sintaxis y semántica, una extensión, o un lenguaje diferente.

- construir un motor SI genérico que entiende descripciones SI escritas en el lenguaje(s) para expresar sintaxis y semántica, y puede usar aquellas descripciones para obtener información de SI en respuesta a una solicitud del programa de aplicación.

En una forma de realización de la invención, el motor SI genérico se configura para convertir versiones transmitidas de la definición de semántica de SI y sintaxis de SI en representaciones internas para ser almacenada por el motor SI.

El motor SI genérico es posteriormente configurado para usar la estructura de las representaciones internas de la definición(s) de SI para responder a preguntas para SI.

[0047] Un experto en la técnica notará que los pasos anteriores no necesitan ser realizados en el orden enumerado por encima.

[0048] Por consiguiente, en una forma de realización de la invención, un lenguaje para la expresión de la sintaxis y semántica de una definición de SI es definido, aunque otra forma de realización podría usar lenguajes separados para la sintaxis y semántica.

Este lenguaje para la sintaxis y semántica se utiliza para expresar el formato donde los datos de SI serán transmitidos, al igual que las relaciones entre datos en el mismo o estructuras transmitidas diferentes.

También definido es un método para inteligentemente tratar la especificación(s) de SI que son escritas en ese o esos lenguajes.

Además de uno o varios lenguajes para la especificando de la sintaxis y semántica del formato de SI, un lenguaje se requiere para su uso por aplicaciones en hacer solicitudes de datos de SI particulares.

Las solicitudes de las aplicaciones deben corresponder a términos identificados en las definiciones sintácticas y semánticas de modo que el motor SI genérico pueda producir máscaras y filtros y demás datos de tratamiento que deben ser retornados a la aplicación.

[0049] FIG. 6 ilustra la arquitectura de un motor SI genérico 36 en una forma de realización de la invención.

El motor SI genérico 36, que se muestra dentro de decodificador de señales digitales 16 pero podría ser implementado en otro tipo de IRD o colocado dentro de una televisión, comprende una interfaz de especificación de formato 60, una aplicación interfaz 70, y una interfaz de filtro 80.

[0050] En una forma de realización, la reconfiguración del motor SI genérico procede de la siguiente manera.

Cuando el motor SI genérico recibe una descripción de un formato de SI nuevo o una descripción de una mejora a un formato de SI existente, usará la descripción para crear un conjunto de estructuras de datos.

Estas estructuras de datos pueden utilizarse para configurar, o reconfigurar, el motor SI genérico y pueden ser usadas por el motor SI genérico para determinar cómo manejar solicitudes de aplicaciones para datos de SI, y cómo manejar datos recibidos de los filtros.

[0051] En una forma de realización, cuando el motor SI genérico recibe una solicitud de una aplicación de datos de SI particulares, el motor SI genérico usa las estructuras de datos mencionadas arriba y otras estructuras de datos almacenadas en el IRD para determinar cómo los filtros en el IRD pueden ser usados mejor para adquirir la información solicitada por la aplicación, o un superconjunto de esta información.

La solicitud de la aplicación se convierte por la interfaz de pregunta de SI 70 en una serie de solicitudes (una o más) a ser hecha al generador de máscara de SI y filtro generalizado 82.

En respuesta a cada uno de estas solicitudes, el generador de máscara de SI y filtro generalizado 82 crea una máscara o un conjunto de máscaras, y elige uno o varios conjuntos de filtros dentro del IRD para usar estas máscaras.

Puede haber diferentes tipos de filtros presentes, cada uno diseñado para filtrar eficazmente información que ha sido codificado en un formato de codificación de sistema particular tal como MPEG o DSS, por ejemplo.

La solicitud de la aplicación incluye, bien implícitamente (porque esta información de nivel inferior se puede definir por las especificaciones de SI) o explícitamente, el formato de codificación de sistema particular o formatos que se deben usar al igual que la codificación de transporte, tal como MPEG-2 o DSS, donde los datos son codificados.

Los filtros pueden ser bien hardware o software o una combinación de ambos.

Los filtros usan las máscaras para determinar qué datos devolver al generador de máscara de SI y filtro generalizado 82 para otro tratamiento, ignorando cualquier dato que no concuerde o entre dentro de una gama especifica por las máscaras.

[0052] En una forma de realización, al recibir los datos de los filtros, el generador de máscara de SI y filtro generalizado 82 usa las estructuras de datos que describen la sintaxis y semántica de SI, junto con las actuales preguntas pendientes, para determinar qué filtración adicional y tratamiento se puede necesitar antes de devolver los resultados a la aplicación solicitante.
Por ejemplo, los filtros pueden ser capaces de filtración sólo en un subconjunto determinado de los bits, dejando al motor SI genérico ejecutar la filtración restante.
Las capacidades de los filtros particulares podrían ser almacenadas en estructuras de datos asociadas a cada tipo filtro, por ejemplo, en uno o más objeto de características de filtro 84.
Además, información retornada de los filtros puede ser analizada por el generador de máscara de SI y filtro generalizado 82 para determinar que datos adicionales, requiriendo disposición adicional de máscaras y filtros, se necesita.
Por lo tanto, los datos devueltos no se pueden devolver inmediatamente o en general a la aplicación, pero en cambio se pueden utilizar para determinar máscaras adicionales para su uso por los filtros.
Finalmente, el generador de máscara de SI y filtro generalizado 82 recibiría todos los datos necesarios para satisfacer la solicitud de la aplicación y guardarlos/almacenarlos o devolverlos a la aplicación, posiblemente tras aplicar otro tratamiento a los datos.
La información se puede esconder en la RAM o en el almacenamiento local tal como dispositivo de almacenamiento 18.

[0053] En una forma de realización, los filtros se pueden configurar para localizar y aislar de manera autónoma información requerida por la aplicación, y devolver la información solicitada directamente a la aplicación en vez de pasarla a través del motor SI.
Adicionalmente, dependiendo del tipo de solicitud, el motor SI podría simplemente memorizar un tipo particular de datos presentados por la aplicación hasta que los datos sean más tarde específicamente solicitados por la aplicación.

[0054] La interfaz de especificación de formato 60 comprende un motor de inicialización de sintaxis de SI 62 y un motor de inicialización de semántica de SI 64.
El motor de inicialización de sintaxis de SI 62 incluye un analizador léxico, programa analizador sintáctico, y motor de inicialización configurado para procesar descripciones escritas en el lenguaje elegido o creado para especificar sintaxis de SI.
De forma similar, el motor de inicialización de semántica de SI 64 incluye un analizador léxico, programa analizador sintáctico, y motor de inicialización configurado para procesar descripciones escritas en el lenguaje elegido o creado para especificar semántica de SI.
Si el mismo lenguaje se utiliza para expresar la sintaxis y semántica del SI, luego el motor de inicialización de sintaxis de SI 62 y motor de inicialización de semántica de SI 64 pueden compartir algunos de los mismos componentes.
Independiente de si el lenguaje de sintaxis de SI es el mismo que el lenguaje de semántica de SI, las representaciones internas puede mantenerse como entidades diferentes o pueden ser incorporadas.

[0055] En una forma de realización, la aplicación interfaz 70, que es también referida como una interfaz de preguntas de SI, puede comprender un analizador léxico y un programa analizador sintáctico para el tratamiento de preguntas de aplicaciones.
Estas preguntas solicitan que datos de SI sea devueltos a las aplicaciones, o guardados.
Si el lenguaje para describir las preguntas de SI es el mismo que el lenguaje usado para describir sintaxis de SO y/o semántica de SI, puede compartir, por ejemplo, la implementación del analizador léxico y programa analizador sintáctico con el motor de inicialización de sintaxis de SI 62 y/o motor de inicialización de semántica de SI 64, respectivamente.
El mismo ejemplo puede ser utilizado, si escrito con sincronización apropiada.

[0056] Como se muestra en FIG. 6, la interfaz de filtro 80 comprende un generador de máscara de SI y filtro genérico 82, y un objeto de características de filtro 84.
El generador de máscara de SI y filtro genérico 82 se puede controlar por el programa analizador sintáctico en la interfaz de pregunta de SI 70.
El objeto de características de filtro 84 es una estructura u objeto que incluye una descripción de las capacidades de filtro de nivel inferior del IRD, que puede incluir, por ejemplo, (i) los tamaños de paquete asociados al filtro; (ii) el número de bytes en el paquete para el que la filtración de hardware está disponible; y/o (iii) si los filtros se pueden configurar para rechazar ciertos modelos de bit antes que para aceptar ciertos modelos de bit.

[0057] Se debe entender que los componentes descritos anteriormente se pueden implementar como módulos diferentes dentro de un único proceso, como un a entidad integrada, o como cualquier combinación de la misma.
También pueden ser más subdivididos en más componentes.
Si implementados como módulos múltiples, pueden ser instanciados como hilos separados dentro de un único programa de ejecución, o como programas separados que comunican el uno al otro o son colocados juntos en una rosca única de un programa de ejecución.

Adicionalmente, los tres lenguajes(para especificar sintaxis de SI, semántica, y preguntas) se pueden combinar en uno o dos lenguajes o expresar como más de tres lenguajes.

[0058] FIG. 6, ilustra uso de flechas, las interacciones de distintos componentes entre sí en el motor SI.
Un proveedor de servicio de radiotransmisión u operador del sistema transmite un flujo que comprende una descripción de la sintaxis de SI y semántica, datos de SI, datos de aplicación (incluyendo código), audio, vídeo, y varias otras informaciones.
Debe observarse que el flujo no puede necesariamente contener toda esta información al mismo tiempo.
En la recepción del flujo de bits transmitido por el IRD, paso 100, el motor de inicialización de sintaxis de SI 62 y el motor de inicialización de semántica de SI 64 convertirá sus descripciones de SI respectivas a una o varias representaciones internas que se pueden usar por varios otros componentes del motor SI, como indicado por paso 102 en la FIG. 6.

[0059] El flujo de bits transmitido puede contener código de aplicación, que es extraído del flujo de bits para ejecución por el IRD, paso 104.
Alternativamente, la aplicación puede ya existir en el IRD o puede haber sido recientemente recibida del flujo de bits transmitido.
Cuando la aplicación empieza la ejecución, puede emitir preguntas (también referidas como solicitudes) para datos de SI particulares, como indicado por paso 106, y las solicitudes se entregan a la interfaz de pregunta de SI 70.
Las solicitudes pueden ser sincrónicas (la aplicación detiene y espera una respuesta) o asincrónicas (la aplicación continúa ejecución, realización otras tareas, hasta que o bien para por alguna otra razón o recibe una respuesta).
La solicitud también puede ser discreta o continua.
Una solicitud discreta es una donde los primeros n casos de la información solicitada son requeridos por la aplicación, donde n es un número entero mayor o igual a 1.
Una solicitud continua es una donde la aplicación desea tener versiones nuevas de la información solicitada continuamente devuelta a esta hasta que cancela la solicitud.
Además, la pregunta de la aplicación se puede clasificar bien como una solicitud de que datos sean devueltos cuanto antes o como una solicitud de que el motor SI guarde datos de SI particulares, como recursos permiten (tal como en la RAM o dispositivo de almacenamiento 18).
Cualquier datos guardados pueden luego ser solicitados más tarde.

[0060] En una forma de realización de la invención, la interfaz de pregunta de SI 70 transmite la solicitud al generador de máscara de SI y filtro genérico 82, paso 108.
El generador de máscara de SI y filtro genérico 82 puede usar información almacenada en la representación interna de sintaxis de SI, representación interna de semántica de SI, y objetos de características de filtro (pasos 109 y 110) para construir una secuencia de una o varias preguntas.
Por ejemplo, la aplicación puede pedir toda información de guía de programa electrónico, que puede corresponder con tener bien un modelo de bit 01 o 10 empezando en el tercer byte de un paquete.
En respuesta, para un tipo de máquina, el generador de máscara de SI y filtro genérico 82 puede construir dos máscaras, una para el modelo de bit "01" y un para el modelo de bit "10", y asignar cada a un filtro de hardware diferente.
Las máscaras también se pueden usar para rebuscar identificativos, tales como identificativos XML, con valores específicos.
En un tipo diferente de máquina, un único filtro de hardware puede ser capaz de simultáneamente buscar paquetes que corresponden a cada máscara.

[0061] Esta secuencia de preguntas se pueden modificar conforme información es devuelta de los filtros en el flujo indicado por 116.
Alternativamente, las solicitudes se pueden construir por la interfaz de pregunta de SI 70, líneas adicionales de uso de comunicación (no mostradas) entre la interfaz de pregunta de SI 70 y la interfaz de especificación de formato 60 o el generador de máscara de SI y filtro genérico 82 se pueden combinar con interfaz de pregunta de SI 70.

[0062] El generador de máscara de SI y filtro genérico 82, posiblemente después de obtener información de las representaciones internas de las descripciones de SI y la descripción de filtro, compondrá máscaras apropiadas y las atribuirá a los filtros apropiados, como indicado a 112.
Los filtros, que pueden ser completamente o parcialmente implementados en el hardware o software, usar las máscaras para obtener los datos de SI solicitados del flujo de bits transmitido, paso 114.
Los datos de SI filtrados es luego devuelto al generador de máscara de SI y filtro genérico 82, mostrado por flujo 116.
Alternativamente, los datos de SI filtrados podrían ser devueltos directamente a aplicaciones a través de un mecanismo de manipulación de interrupción o por sondeo.
Después de la recepción de los datos de SI, el generador de máscara de SI y filtro genérico 82 puede filtrar además la información antes de devolverla a la interfaz de pregunta de SI 70, paso 118.
Como declarado por encima, la interfaz de pregunta de SI 70 y generador de máscara de SI y filtro genérico 82 se puede implementar como un único componente, en cuyo caso la interfaz de pregunta de SI 70 (comprendiendo el generador de máscara de SI y filtro genérico) desempeñaría la otra filtración.

[0063] Cuando la interfaz de pregunta de SI 70 recibe los datos de SI, examina la información y pueden coger cualquier combinación de las siguientes acciones:

- hacer otra solicitud al generador de máscara de SI y filtro genérica 82, basado en los valores devueltos por el generador de máscara de SI y filtro genéricos 82 hasta el momento, paso 108.
- hacer otra solicitud del generador de máscara de SI y filtro genérica 82, independiente de los valores devueltos hasta entonces.

Paso 108.

- pasar la información retornada, posiblemente combinada con información previamente retornada o un subconjunto de la misma, de nuevo a la aplicación que hace la pregunta, paso 120.
- guardar parte o toda la información devuelta, como permitan los recursos.

Guardar puede hacerse directamente por el generador de máscara de SI y filtro genérico 82, la interfaz de pregunta de SI 70, o por un módulo dedicado a asignar recursos para guardar y desempeñar el guardado.

- anular la solicitud al generador de máscara de SI y filtro genérica 82 de modo que los filtros se puedan reutilizar para otro fin.

[0064] Después de que la información se entregue a la aplicación en el paso 120, la aplicación puede anular la solicitud que produjo la información o dejó la solicitud abierta si la solicitud fue continua.
Si la solicitud fue discreta, y la interfaz de pregunta de SI 70 ha devuelto el número solicitado de versiones (que, en muchos casos será uno), o la aplicación pide cancelación, la interfaz de pregunta de SI 70 anulará la solicitud al generador de máscara de SI y filtro genérico 82, que a su vez liberará los filtros.

[0065] Un ejemplo de un lenguaje de especificación de sintaxis de SI utilizable conforme a la invención se describe por debajo.
La forma de realización descrita no es más que un ejemplo de muchos lenguajes posibles que pueden ser utilizados. Se presenta como una implementación posible, y debería no ser leída de ninguna manera limitando el ámbito de la invención.

[0066] La sintaxis de un simple lenguaje, que permitiría introducir casi al pie de la letra muchos de los documentos de definición de SI existentes, se puede expresar en la BNF extendida (notación de Backus-Naur) como se muestra en FIG. 7.
Como es típico, λ significa la cadena vacía y literales se encierran en comillas "".
FIG. 8 ilustra una descripción escrita en esta sintaxis.
Esta descripción define parcialmente una sección DVB (emisión de vídeo digital) de MPEG de SI con una subtabla de información de red.
Si un motor SI genérico 36 estuviera ya en su lugar en el extremo de receptor (por ejemplo en un IRD, televisión, u otro dispositivo), entonces sería transmitida alguna codificación, quizás según ASCII o unicódigo, de la definición textual de FIG. 8 (tal como por emisión, punto a punto, etc.) al receptor, quizás como una sección privada MPEG-2.

[0067] Referir nuevamente a FIG. 8, la definición de la subtabla de información de red asume que MPEG se usa como la codificación de sistema del flujo de bits.
En esta codificación, la existencia de la subtabla de información de red se señala por ajuste del PID llamado de campo definido por MPEG (identificador de paquete) al valor 8.
La definición también muestras que esta sección es reconocible por un valor table_id (identificador de tabla) de 64 o 65 de una sección.
El Valor de PID y el valor table_id se usan en este lenguaje de SI porque filtros deben ser capaces de localizar subtablas parciales que son identificable solo por valores de PID, y no son identificable por valores table_id (como si la subtabla es demasiado grande para encuadrar un único paquete).
Las longitudes de cada campo se dan en el número de bits.
Tanto "bslbf" como "uimsbf" se enumeran como tipos básicos en la tabla interna, con lo que el motor habría sido inicializado, que describe MPEG, uno de unos cuantos estándares comúnmente usados.

[0068] Este ejemplo muestra bucles dentro de bucles.
Debido a que el alcance de las longitudes de bucle está en el bucle o estructura actual, ninguna notación "." es necesaria para identificar la longitud de bucle.
Dos de los bucles, el primero y el tercero (que está anidado dentro del segundo), pueden contener descriptores, que pueden ser cualquiera del aquellos enumerados bajo "alterno" porque son conocidos como DescriptorNIT.
Ocurre que en la definición de DVB SI cualquier lista larga de descriptores es posible en numerosos posiciones en la mayor parte de las subtablas (el término que DVB usa para las estructuras en cuestión, aunque pueden que no estén necesariamente en formato de tabulado).
No obstante, el lenguaje que es usado aquí permite la posibilidad de tener restricciones en los tipos de subtablas donde descriptores pueden aparecer.
Será aparente para un experto en la materia que el ejemplo mostrado en la FIG. 8 es solo una descripción parcial.
Campos tales como network_name_descriptor (descriptor de nombre de red) y data_broadcast_id_descriptor (descriptor de identificador de emisión de datos) necesitarían ser más refinado, y las otras subtablas necesitarían ser definidas y la lista exacta de descriptores sería necesitada.

[0069] El lenguaje discutido aquí es solo un ejemplo muy simple de un lenguaje que podría usarse para expresar sintaxis de SI, y hay muchas extensiones posibles y modificaciones que causarían o permitirían a la gente escribir descripciones más largas o más cortas de especificaciones de SI particulares.
Por ejemplo, el lenguaje anterior se puede mejorar para incluir herencia de modo que una tabla podría ser definida para ser casi idéntica a otra tabla, con ciertos campos sobregrabados.
Adicionalmente, un realce del lenguaje podría permitir a una persona escribir todas las tablas posibles donde un descriptor nuevo podría aparecer, antes que tener que añadir ese descriptor nuevo a todos los grupos representantes de estructuras apropiadas de descriptores alternos para cada tabla dada.

[0070] Una implementación posible de un motor de inicialización de sintaxis de SI 62 sería un programa que lee una descripción dada en un lenguaje similar al mencionado anteriormente y crea estructuras de datos similares a aquellas ilustradas en la FIG. 9, usando técnicas bien conocidas en la técnica. Las estructuras de datos pueden ser dinámicamente asignadas y pobladas con información, tales como la de FIG. 8, y pueden sostener cualquier cosa expresable en el lenguaje de FIG. 7.
Estas estructuras de datos corresponden a la representación interna de la sintaxis de SI y más tarde serían accedidas por el generador de máscara y filtro genérica 82, junto con posiblemente información del objeto de características de filtro 84, para permitirle localizar campos solicitados en una cadena de bits al igual que para configurar máscaras para filtros de nivel inferior.

[0071] Debido a la gran cantidad de datos en un flujo de emisión de televisión digital típica, es actualmente poco práctico examinar, dentro del software de aplicación o el motor SI, todos los datos de SI contenidos en el flujo.
Así, esta divulgación proporciona la capacidad de aplicaciones de hacer dos diferentes tipos de solicitudes al software de nivel inferior.
El primer tipo de solicitud produce la devolución de datos a la aplicación.
La aplicación solicita que estructuras específicas, que puede ser simples o muy complejas, sean devueltas al software de aplicación si tienen valores particulares en campos específicos.
El software de capa mediana establecería entonces máscaras que los filtros de hardware de nivel bajo (y posiblemente el software) usarían para limitar el número de candidatos que debe ser además analizado y filtrado por la capa mediana, antes de devolver las estructuras solicitadas al software de aplicación.

[0072] El segundo tipo de solicitud es conocida como una solicitud de guardado.
Una solicitud de ocultamiento es casi idéntica al primer tipo de solicitud, excepto que las estructuras sacadas del flujo transmitido no son inmediatamente devueltas al software de aplicación.
En cambio estas estructuras serían guardadas, como permitan tiempo, espacio, y asignación de filtros de hardware, por la capa mediana (por ejemplo, en la RAM, en el dispositivo de almacenamiento 18, etc.).
Valores guardados podrían luego ser usados como una fuente adicional de información cuando una aplicación hace una solicitud del primer tipo.

[0073] Por lo tanto, es necesario un lenguaje en el que hacer estas solicitudes, aunque los conceptos de esta invención no dependen de cualquier elección de lenguaje particular.
Es solo necesario que el lenguaje permita al programador de aplicaciones especificar exactamente las estructuras que deben ser devueltas o guardadas.
Una gama de métodos podrían ser tomadas en el diseñar del lenguaje para que use el programador de aplicaciones.
Cualquier método específico se caracteriza por la cantidad de conocimiento que el programador de aplicaciones debe tener en lo que se refiere el significado asociado a la sintaxis de SI.

[0074] A un extremo del espectro, el programador de aplicaciones entiende todo acerca de la parte de la sintaxis de SI y semántica que el emisor u operador del sistema está usando y usa esa comprensión para solicitar específicamente estructuras y campos de aquellas estructuras.
En el otro lado del espectro, el programador no entiende nada acerca de la sintaxis o semántica subyacente particular de SI, debido a que el emisor u operador ha dado significado a términos de mayor nivel, que se expresan en el nivel inferior de sintaxis y semántica de SI.
Usando este segundo modelo, el programador de aplicaciones solicitaría información utilizando solo estos términos de mayor nivel.

[0075] Una forma de realización de la invención puede usar una implementación en algún lugar entre estos dos extremos.
Usando tal método intermedio, el emisor u operador del sistema definiría algunos términos de nivel alto que el programador de aplicaciones puede usar para solicitar tipos comunes de datos de SI.
Adicionalmente, bajo este método, el programador de aplicaciones tendría la flexibilidad para solicitar información menso común usando conocimiento del formato de SI subyacente.
Ejemplos de estos métodos se describen aquí.

[0076] Diferentes formas de realización de pregunta y lenguajes de especificación semántica de SI se describen más abajo, junto con descripciones de interfaces de pregunta de SI 70 y motores de inicialización de semántica de SI 64 utilizables con los lenguajes.
Estas formas de realización se presentan con motivo de que ilustran los conceptos de la invención, y se basan en la suposición de que la sintaxis de SI es especifica en el lenguaje de especificación de sintaxis de SI anteriormente descrito y en la FIG. 7.
Se debe entender que cualquier otro lenguaje lo suficientemente potente para describir la sintaxis de SI puede ser utilizado, y en tales lenguajes, un convenio que depende del lenguaje de especificación de sintaxis de SI puede ser adoptado.
Por ejemplo, network_information_section.loop_2.loop_1.transport_stream_id (sección de información de red.bucle 2.bucle 1.identificación de flujo de transporte) referiría a un valor transport_stream_id (identificación de flujo de transporte) dentro del primer bucle que está dentro del segundo bucle de un network_information_section.
Como otro ejemplo, network_information_section.loop_1.descriptor.service_list_descrip se referiría a una colección de todos los campos de un service_list_descriptor (descriptor de lista de servicio) que serían encontrados en el primer bucle de un network_information_section, mientras que

```
network_information_section.loop_1.descriptor
.service_list_descrip
  tor.service_id
```

se referiría solo al campo service_id (identificador de servicio) de este mismo descriptor.
Extensiones a los convenios arriba sugeridos se pueden utilizar para permitir la identificación de un elemento de SI particular.
Adicionalmente, si reglas de alcance son necesitadas, podrían ser implícitos o explícitos.
Un ejemplo donde reglas de alcance se usan es ahora mostrado.
Considerando el caso donde las siguientes dos referencias existen:

```
network_information_section.loop_1 =

named_loop.descriptor.service_list_descriptor

named_loop.descriptor.satellite_delivery_desc
riptor
```

[0077] Si aparecen en el mismo alcance, este ejemplo indica que ambos descriptores deben aparecer en el mismo bucle de un network_information_section.
En cambio, colocación de las siguientes dos referencias en un alcance diferente significarían lo mismo que lo siguiente:

```
network_information_section.loop_1.descriptor
.service_list_descrip
  tor

network_information_section.loop_1.descriptor
.satellite_delivery_d
  escriptor
```

[0078] Este ejemplo último se refiere a dos descriptores diferentes, que pueden ocurrir en la misma instanciación o una instanciación diferente del loop_1 (bucle 1).

[0079] En una forma de realización, un lenguaje de pregunta de alto nivel se puede acoplar con un lenguaje de semántica de SI apropiado.
Para este método, dos lenguajes adicionales se usan.
El primer lenguaje, que se llama el lenguaje de semántica de SI, típicamente sería usado por el emisor u operador o sus representativos y contractores para especificar significados para objetos comúnmente usados en su representación de SI.
El segundo lenguaje, que se llama el lenguaje de pregunta, hace uso de términos definidos por el emisor en el lenguaje de semántica de SI.
El segundo lenguaje permite al programador de aplicación preguntar información de las representaciones internas de SI en el motor SI genérico 36 sin requerir que el programador de aplicaciones conozcan cómo la información se almacena en las estructuras de SI.

[0080] Para ilustrar adicionalmente esta forma de realización, considerar un escenario de ejemplo que demuestra el uso de un lenguaje de semántica de SI que complementa un lenguaje de pregunta de alto nivel.
Palabras clave de los dos lenguajes se representan en negrita en este ejemplo.

Suponga que el espectador ha sacado un "menú de configurar la guía de TV" que está disponible como una aplicación en el decodificador de señales digitales (o TV u otro dispositivo usado para TV interactiva).
Esta aplicación puede ser bien descargada del flujo de emisión a la carta, descargada del Internet o conexión punto a punto, o ya estar guardada en el decodificador de señales digitales del espectador.
En este ejemplo, el espectador ha oído que habrá un festival de John Wayne en el canal 17 en algún momento en los próximos dos días y quiere determinar si habrá alguna películas de John Wayne, en particular cualquier producidas por la productora Metro-Goldwyn-Mayer, entre las 07 a.m. y las 01:30 p.m. del día actual.
Después del espectador elija artículos de menús apropiados y quizás introduzca información (utilizando un control remoto, teclado, u otro dispositivo de entrada), la aplicación formula la siguiente llamada al software subyacente:

```
 Success = O_si_query("acquire eventInfo where
channel (= 17) and
 startTime(>=, 25200) and endTime(<=, 48600)and
eventType(=, Movie)
 and itemPair(=, Factor, =, 'John Wayne') and
itemPair(=,
 ProductionCompany, =, 'MetroGoldwynMayr')",
&event);
```

[0081] En este ejemplo, la aplicación ha convertido la hora de inicio y finalización a número de segundos pasada la medianoche del día actual.
La pregunta es el componente incluido entre comillas dobles, "". El resto del extracto arriba representa una forma donde la pregunta se puede utilizar dentro de una interfaz del programador de aplicación (API).

[0082] Algunas suposiciones de simplificación han sido hechas en esta ilustración.
El SI que el emisor está usando para este ejemplo es muy similar al DVB SI, aunque una diferencia es que todos los tiempos se expresan como el número de segundos pasada la medianoche, hora local.
También, el mismo SI DVB no proporciona forma de asociar qué piensa el espectador que es un número de canal (que el espectador introduce con el control remoto, por ejemplo) para el triplete que SI DVB normalmente usa para identificar un servicio; es decir el identificador de servicio, el identificador de red original, y el identificador de flujo de transporte o los valores usados en una tabla de ATSC similar.
Por lo tanto, para esta ilustración, se supone que el emisor ha definido sus propias secciones, llamadas secciones de correspondencia de canal, cuyo fin es a asociar el concepto del espectador de un número de canal con valores para este triplete (o los Valores de ATSC).

[0083] El emisor u operador habrá ya escrito, usando su lenguaje identificado, y difundido a IRD, una traducción de los términos usados por el escritor de la aplicación para definir un conjunto de constantes, en este caso película, actor, y productora, y un tipo objeto, eventInfo, como se muestra en figuras 10a y 10b. La definición de eventInfo (información de evento) indica que un event_information_section (sección de información de evento) se obtiene y campos particulares del objeto solicitado son devueltos.
Incorporado en unas de las definiciones de campo es una clave de cálculo.Esta es una indicación de que la hora de finalización no se obtiene directamente de los campos en la tabla extraída del flujo, pero es calculada en base a ellos.
Los balances bajo la palabra *where* definen varios métodos que, cuando solicitados por una aplicación en este caso, resultan en una reducción de los candidatos para los valores a ser devueltos al interemisor.
Como se muestra, no todos métodos necesitan ser usados por una pregunta particular, y un método se puede usar múltiples veces, como con el método de instancia.
Un balance calculado, como explicado más adelante, puede consistir en operandos y los operadores +, -, *, /, div, mod, min, y max. Por lo tanto, los cálculos se pueden realizar utilizando una simple estructura de pila incorporada en el motor SI 36.

[0084] Una gramática que define la sintaxis para el lenguaje de semántica de SI ejemplo se describe.
Los identificadores de autenticación son definidos de la siguiente manera:

```
 DEFINE= "Define", OBTAIN = "obtain", EQUALS =
"=", GT = ">", LT = "<",
 NOTEQUALS = "I=",GTEQ = ">=", LTEQ =
"<=",SEMICOLON = ";",STRINGTYPE
 = "string", INTTYPE= "int", OBJECT = "Object",
LEFTCURLY = "{",
 RIGHTCURLY = "}", FETCH = "fetch", RETURN =
"return", COLON = ":",
 ASSIGNOP = ":=", LSQUARE= "[", RSQUARE = "]",
DOT = ".", PLUS = "+",
 MINUS = "-", TIMES = "*", DIV = "div", MOD =
"mod", MIN = "min", MAX =
 "max", RELOP = "relop", SET = "set", FILTER =
"filter", COMPUTE =
 "compute", LPAREN = "(", COMMA = ",". RPAREN
= ")", WHERE = "where",
 DBLEQUALS = "= ="
 INTEGER = digit (digit)*,
 STRING = "'" (any_char_except_')* "'", and
 VARIABLE = letter
(any_letter_or_digit_or_underscore)*.
```

[0085] En la definición anterior, digit representa cualquiera de los caracteres 0..9, any_letter_or_digit_or_underscore representa cualquier carácter que es una letra en el rango de a..z o A..Z o 0..9. El término ny_char_except_' representa cualquier carácter excepto la comilla simple.
El símbolo * en las definiciones anteriores significa "cualquier número (incluso 0) de los elementos entre paréntesis puede ser incluido".

[0086] Los no terminales se definen en una versión de BNF, como ilustrado en figuras 11a y 11b.
El programa no terminal es el objetivo inicial.
Como de costumbre, λ se refiere a la cadena vacía, y el símbolo "|" significa que el no terminal se puede sustituir por la expresión a la izquierda del "|" o la expresión a la derecha.

[0087] Un motor de inicialización de semántica de SI correspondiente 64 será ahora descrito.
Usando la gramática definida en figuras 11a y 11b, el emisor u operador puede describir estructuras nuevas de nivel más alto que incluyen campos elegidos de las estructuras de SI originales.
El propósito del motor de inicialización de semántica de SI 64 es analizar un conjunto de descripciones de las estructuras de nivel más alto y memorizar una representación interna de las descripciones en una estructura.
Esta estructura de representación interna se usa por el generador de máscara de SI y filtro genérica 82 (o posiblemente por la interfaz de pregunta 70, como arriba declarado) para determinar exactamente qué datos de SI obtener, basado en la pregunta de la aplicación.
Por lo tanto, cuanto se necesita es código que lee los datos en forma de la gramática y almacena en una forma donde esto puede más tarde ser recuperado.

[0088] Un ejemplo de una estructura que puede así ser generada por el motor de inicialización de semántica de SI 64 se muestra en figuras 12a, 12b, y 12c en una notación tipo C.
El dispositivo de visualización PtrToDefns es inicializado por el motor de inicialización de semántica de SI 64.
Este dispositivo de visualización contiene la dirección del primer elemento de una lista de definiciones.
Cada definición señala a la siguiente.
De forma similar, cada definición declara si es una definición de un número entero, una cadena, o un objeto nuevo.
Si es un número entero o cadena, entonces es una definición constante, así la definición almacena el valor real.
De lo contrario, almacena la estructura del nuevo objeto que está siendo definido.
Este nuevo objeto induye indicadores a otras estructuras que deben ser adquirida, indicadores a información acerca de métodos para invocar aquellos otros objetos, indicadores a estructuras que contienen nuevos nombres para valores que son devueltos, y una lista de indicadores a filtros que será conjunto para objetos que serán adquiridos más tarde.
Cada uno de estos elementos de objeto es lo suficiente complejo para sostener toda la información en la descripción semántica de SI que se envían en el flujo de transmisión.
Al mismo tiempo, son lo suficiente simples para ser recorridos para determinar objetos de Si que deben ser obtenidos y para determinar filtros en aquellos objetos de SI que deberían ser usados para obtener los valores reales para los objetos de nivel más alto que se definen en este lenguaje.

[0089] Como se ha establecido por encima, el lenguaje de pregunta, que es usado por aplicaciones, es muy simple en este caso.
Su gramática está muy cerca un subconjunto del lenguaje de semántica de SI anterior, donde cada invocación del programa de aplicación corresponde a algo similar a una expresión de obtain (obtener).
El lenguaje de pregunta difiere en que o **acquire** (adquirir) o **cache** (guardar) se pueden usar en lugar de **obtain**.
La clave **acquire** puede utilizarse para indicar que la aplicación quiere que los datos de SI solicitados sean devueltos cuando se encuentren.
Por otro lado, **cache** sería usada para indicar que el motor SI 36 debería guardar este tipo de datos de SI, si los recursos lo permiten, y que la aplicación más tarde ejecutaría un extracto **acquire** para obtenerlos.
La gramática para el lenguaje de pregunta correspondiente, por lo tanto, puede parecer similar a la descripción de aquí debajo.

```
Query : := Request ObjName OptConstraint
Request : := "acquire" | "cache"
ObjName : := VARIABLE
OptConstraint : := WHERE OptNot OptConstraints
OptConstraints : := λ| Constraint Connector
OptNot OptConstraints
Constraint : := MethodName LPAREN
ActualParamList RPAREN
ActualParamList : := ActualParam
OptMoreActualParams
OptMoreActualParams ::= λ| COMMA ActualParam
OptMoreActualParams
ActualParam : := VARIABLE | INTEGER |
Comparator
Comparator : := DBLEQUALS | GT | GTEQ | LTEQ
| LT NOTEQUALS
Connector : := "and" | "or"
OptNot : : = λ | "not"
```

[0090] Como antes, esta forma de realización ha sido presentada con motivo de ilustración, y debe ser reiterado que hay un número infinito de posibilidades para este tipo de lenguaje.
Por ejemplo, el lenguaje descrito aquí incluye los dos conectores lógicos AND y OR, al igual que el opcional lógico NOT, donde subconjuntos varios de estos conectores habrían bastados (por ejemplo OR se pueden expresar como una una combinación de funciones AND y NOT, porque o es equivalente a AND-NOT con todas entradas negadas).

[0091] La interfaz de pregunta de SI correspondiente se puede invocar a través de un API que contiene una cadena formateada según el Si lenguaje de pregunta de arriba.
El API también puede permitir solicitudes sincrónico o asincrónico del programador de aplicaciones.
Unas pausas de solicitud sincrónica el programa de aplicación hasta que el valor de SI se obtiene y devuelta.
Una solicitud asincrónica permite el programa a continuar inmediatamente.
En cualquier caso, el programa de aplicación puede usar el API para especificar donde memorizar los datos de SI devueltos, si los hay.
Si el programa de aplicación ha previamente solicitado que tipos determinados de datos de SI sean guardado, luego una solicitud más tarde de obtener datos pueden suponer control la ubicación guardada antes de traer cualquier dato de SI nuevo.
En todas estas situaciones, la interfaz de pregunta de SI analizaría la solicitud.
La interfaz de pregunta de SI 70 o el generador de máscara de SI y filtro genérico 82 usaría conocimiento de la estructura anteriormente descrito a localizar la descripción del objeto de nivel de alto que fue solicitada en la pregunta.
Luego usaría esta descripción, que puede indicar que un conjunto de estructuras intermedias o múltiples sean obtenidas de los datos de SI, para crear la estructura de nivel más alto solicitada por la aplicación.

[0092] En una forma de realización de la invención, el lenguaje de pregunta se puede implementar como un lenguaje de nivel bajo.
Usando un lenguaje de pregunta de bajo nivel, el programador de aplicaciones podría solicitar datos de SI específicos usando conocimiento de la estructura de la emisión SI.
FIG. 13 demuestra como tal lenguaje se puede utilizar para la construcción de una pregunta.
El espectador televisivo, que puede estar ocupado durante las siguiente pocas horas, puede desear grabar algunos programas de noticias interesantes entremedias.

Por lo tanto, el espectador puede querer ver una lista de tales programas que estarán ofrecidos en el "servicio básico" al que se suscribe, representado por un bouquet particular (grupo de canales).
Este servicio básico puede consistir en algunos canales que son portados vía satélite y otros canales que son portados vía cable.
Por lo tanto, el espectador necesitará configurar el IRD a cable o satélite, dependiendo de las muestras que son ofrecidas.
Usando una interfaz de usuario apropiada, el espectador puede indicar un interés en programas "de noticias", el período de tiempo de interés (RequestedStartTime (hora de inicio solicitada) y RequestedEndTime (hora de fin solicitada)), que se interesa solo por cadenas que son transmitidas vía cable (si han configurado para cable), y que las cadenas deben incluirse en el "servicio básico". Estas opciones pueden, por ejemplo, ser presentadas en un menú extensible u otro formato adecuado.
Las selecciones del espectador serían traducidas en una pregunta que es algo SQL, como se muestra en FIG. 13.

[0093] La pregunta ilustrada pide que todo el contenido de cada instanciación del primer bucle de un event_information_section (sección de información de evento) sea devuelta a la aplicación, si condiciones se encuentran en tanto: (i) algunos campos del event_information_section que están dentro del bucle; y (ii) algunos campos que están dentro de ese bucle.
Los campos que son pertinentes fueran del bucle incluyen el original_network_id (identificador de red original) y el transport_stream_id.
Estos dos campos sirven únicamente para identificar cualquier flujo de transporte de cualquier otro.
Por conocer estos valores, es posible usar información en otras tablas para determinar, por ejemplo, si un flujo de transporte particular se soporta sobre cable o vía satélite y a qué conjuntos que transportan flujo pertenece.
Información dentro del bucle referenciado es específico a un caso particular, permitiendo la determinación en cuanto a la hora de comienzo y duración del evento, el tipo del evento (por ejemplo, si es un drama, un evento deportivo, o un evento de noticias), el título, productor, y en algunos casos actores individuales que aparecen en el evento.

[0094] El primer segmento que expresa tales condiciones asegura que solo eventos que están en la categoría de un programa de noticias son devueltos.
El segundo segmento de condición es más complejo, como se ilustra por FIG. 14.
Los transportes stream_id (identificador de flujo) y original_network_id encontrados en el event_information_section deben ser idénticos al encontrado en una instanciación del segundo bucle de un bouquet_association_section (sección de asociación de bouquet).
No obstante, no bastará simplemente cualquier bouquet_association_section.
El bouquet_association_section donde este transport_stream_id y original_network_id se encuentran debe ser el mismo bouquet_association_section que contiene un transport_stream_id y original_network_id cuyos valores son idénticos a los descubierto en un network_information_section para el corriente flujo de transporte.
El bucle donde este segundo par de transport_stream_id yoriginal_network_id son encontrados puede ser el mismo bucle donde el primer par fue encontrado; es decir, en la figura x = y y v = z. En vez de generar esta pregunta de compuesto, la aplicación podría haber primero consultado el original_network_id y transport_stream_id en el network_information_section del flujo de transporte actual.
Utilizando esta información, podría luego haber consultado el bouquet_id que corresponde con el actual flujo de transporte, y para el conjunto de todos los pares de original_network_id y transport_stream_id en el bouquet con aquel bouquet_id.
La aplicación podría haber restringido el conjunto de estas a aquellos portados en el cable (como será discutido por debajo), y, finalmente, podría restringir los eventos según categoría y hora.

[0095] La tercera restricción define el requisito de que el flujo de transporte debe ser accesible vía cable.
Esto es, el transport_stream_id y original_network_id del flujo sobre el que el evento es portado debe también estar enumerado en un network_information_section que contiene un cable_delivery_system_descriptor.
Nótese que esta puede ser la misma network_information_section a la que se refiere en la descripción de la segunda condición.
También puede ser diferente, debido a que la misma estación (es decir, transport_stream_id y original_network_id) puede ser retransmitido por medios múltiples.

[0096] La cuarta y final condición en el ejemplo de FIG. 13 se muestra en FIG. 15, que expande la condición "// DVB_time_Between" (tiempo de DVB entre).
Como se puede observar por referencia a la figura, estas condiciones temporales se traducen en condiciones en los campos section_number (número de sección) de event_information_sections al igual que en los campos de start_time (hora de inicio) y duración.
Los eventos apropiados, colocados en los periodos de tiempo apropiados, podrían ser localizados y devueltos a la aplicación sin especificar las condiciones en el campo section_number.
No obstante, no especificar las condiciones en el campo section_number requerirían que un IRD típico ejecutara sustancialmente más filtración (paquetes de eliminación en los que la aplicación no está interesada) en el software que en hardware (ya que los filtros son normalmente implementados en el hardware), quizás causando que pierda (debido a rebosamiento de protección), o al menos retrase, paquetes que la aplicación necesita definitivamente.

[0097] Un experto en la técnica reconocerá de la divulgación anteriormente mencionada que el lenguaje en que se expresan preguntas para este tipo particular de SI deben incluir la capacidad para especificar lo siguiente:

- de qué estructuras debe ser extraída la información;
- operaciones aritméticas íntegras arbitrariamente complejas que usan operandos comúnmente encontrados en la mayoría de los lenguajes de programación;
- atribuciones;
- comparaciones arbitrariamente complejas hechas de los operadores de comparación típicos: <,>, ΞΞ (es igual a), ≥, y ≤;
- limitaciones lógicas arbitrariamente complejas hechas de los operadores lógicos típicos: y, o, y no.

El lenguaje de semántica de SI arriba descrito posee todas estas propiedades.

[0098] En una forma de realización de la invención, no es necesario crear un lenguaje o conjunto de lenguajes nuevo para usar para especificar la semántica de SI y las preguntas.
Por ejemplo, Prolog puede ser utilizado.
Se debe entender que si Prolog, u otro lenguaje lógico o interpretado de computación de uso general se usa, la sobrecarga en tiempo de ejecución puede ser significativa, para tanto el motor de inicialización de semántica de SI 64 como, si éste también usado como la forma de representación interna, el generador de máscara de SI y filtro generalizado 82.
El IRD debe tener suficiente de potencia de tratamiento para manejar la sobrecarga requerida.

[0099] Un ejemplo del uso de Prolog para la expresión de la semántica de una parte de una definición de SI se ilustra en FIG. 16.
La primera regla declara que X es el actual transport_stream_id si A es una program_association_table (tabla de asociación de programa) y A tiene un campo llamado transport_stream_id cuyo valor es X.
La segunda regla define cuando C es un miembro del bouquet B. C es un miembro del bouquet B si L es un bouquet_association_section cuyo campo llamado bouquet_id tiene el valor B y cuyo campo nombrado transport_stream_id tiene el valor C.
Las últimas dos reglas identifican los dos casos donde se puede determinar que el flujo cuyos transport_stream_id es X viene enviado en unos medios particulares (es decir, cable, satélite o terrestre).
La primera de las últimas dos reglas indica que X viene transmitido vía los medios específicos si X es el actual transport_stream_id (que hacen uso de la primera regla) y la network_information_section que corresponde con el actual flujo de transporte (significadas por un table_id de 32) tiene un descriptor en su primer bucle que es de tipo frequency_list_descriptor (descriptor de lista de frecuencia) cuyo campo coding_type (tipo codificante) tiene el valor *media*.

[0100] Figuras 17a-17f ilustran un ejemplo ligeramente más complejo.
FIG. 17a muestra una regla que puede utilizarse para determinar una lista de eventos que pueden iniciar tan pronto como la hora de comienzo solicitada y que termina antes de la hora de finalización solicitada (inclusive).
Para obtener una lista no vacía de tales eventos, al menos un servicio en el flujo de transporte solicitado debe emitir un horario.
Si al menos un tal horario es emitido, una gama de números de segmento debe ser obtenida por como l SI DVB especifica que tablas de información de evento son divididas, hasta 8 valores de número de segmento para cada intervalo de 3 horas en el día.
Después de que todos los eventos descritos en tablas de información de evento con los números de segmento apropiado se obtengan, debe ser verificado que los eventos reales yacen entre los tiempos solicitados.
Esto es hecho por la última regla mostrada en la FIG. 17a.

[0101] FIG. 17b muestra una regla para determinar si cualquier horario son emitidos para servicios en un flujo de transporte dado.
La Figura 17c muestra reglas que pueden utilizarse para obtener información de evento que corresponde con una gama de números de segmento.
Como este ejemplo se basa en DVB, y debido a la forma en que DVB estipula que números de segmento sean asignados a números, hay dos reglas diferentes.
El primero es para encontrar información en el primer segmento que corresponde con un tres bloque de hora, y el segundo es para encontrar el resto de la información para esos tres bloques de hora.
Dos reglas diferentes se usan porque algunos números de segmento pueden estar sin usar, y sería ineficiente tener un filtro o conjunto de filtros dedicado a localizar información que no aparecerá en el flujo de transporte.

[0102] FIG. 17d ilustra varias reglas que se necesitan para determinan la diferencia entre los valores de hora local corriente dadas en la gama solicitada y medianoche de la fecha actual en el huso horario UTC (tiempo universal coordinado)-0.
FIG. 17e presenta las reglas necesarias para determinar los números de segmento que corresponden a tiempos particulares.
Finalmente, FIG. 17f muestra como los eventos se controlan para determinan si de verdad se sitúan en el período de tiempo especifico.

[0103] Un motor de inicialización de semántica de SI configurado para usarse con las definiciones anteriores serían uno que simplemente guardara reglas similares a aquellos mostrados en figuras 16 y 17a a través de 17f.
Para el lenguaje de pregunta de SI, si Prolog o lenguaje similar fue usado para la expresión de la semántica de SI, el mismo lenguaje se puede utilizar para expresar las solicitudes.
En una forma de realización, FIG. 18 muestra una pregunta que pide los títulos de todos eventos de noticias que se deben mostrar en un canal de cable, que se une al mismo bouquet como el programa que el usuario está actualmente viendo entre el 13 de junio de 2000 a las 9:30 am y 13 de junio de 2000 a 1 pm, inclusivo.

[0104] La sintaxis de SI interno y representaciones de semántica se pueden usar por la interfaz de pregunta de SI 64 en una forma de realización de la invención.
El generador de máscara de SI y filtro genérico 82 también puede usar la sintaxis de SI interna y representaciones de semántica.
Los lenguajes y estructuras discutidos aquí se pueden utilizar para especificar la estructura de datos de SI y para memorizar qué especificación de estructura de SI, aunque hay muchas formas diferentes de especificar una estructura de SI y para memorizar la especificación.
No importa qué estructuras se utilizan para memorizar la especificación de SI, en esta aplicación la versión almacenada de la especificación ha sido referida como una representación interna de especificación de sintaxis de SI.

[0105] Los métodos de la presente invención se pueden resumir como se muestra en FIG. 19.
En el paso 190, la descripción de formato es transmitida, con la sintaxis y semántica del formato.
La descripción de formato se recibe en el paso 192.
Una representación interna o representaciones (tal como si lenguajes diferentes se usan para ambos) de la sintaxis y semántica será creada, paso 194.
Una solicitud de aplicación se recibe en paso 196, y luego utilizando la pregunta, representación(s) interna, e información de filtro (que se puede almacenar en un objeto de características de filtro), una máscara o conjunto de máscaras serán creadas, paso 198.
Las máscaras se aplican a filtros seleccionados en el paso 200, y los metadatos son filtrados utilizando las máscaras en el paso 202.
Diferentes pasos son posibles después de que la información haya sido recogida.
La información se puede utilizar para ajustar o modificar máscaras, o máscaras se pueden establecer o modificar independientemente de los metadatos filtrados, como se muestra en paso 204.
La información devuelta se puede pasar de nuevo a la aplicación que hace la pregunta, bien de por sí o en combinación con información previamente devuelta (y almacenada/guardada), paso 206.
Parte o toda la información devuelta puede ser almacenada, en el paso 208.
Las máscaras también pueden ser canceladas, paso 210.

[0106] Un motor reconfigurable para metadatos formateados de tratamiento se ha descrito.
El motor se puede implementar en el software, hardware, o una combinación de los mismos.
Si hay parte de la invención se implementa en software, este software se puede almacenar en alguna forma de medio legible por ordenador, tal como memoria o CD-ROM, o transmitido por una red, y ejecutado por un procesador.
Adicionalmente, donde métodos han sido descritos, secuencias varias de pasos pueden ser posibles, y puede ser posible de ejecutar tales pasos simultáneamente, sin apartarse del ámbito de la invención.

[0107] Aunque la presente invención ha sido descrita de acuerdo con las formas de realización mostradas, un técnico en la materia reconocerá fácilmente que podría haber variaciones hechas a las formas de realización sin apartarse del alcance de la presente invención.
Por ejemplo, el motor reconfigurable se puede utilizar para procesar cualquier dato rígidamente formateado, y no está limitado a SI o metadatos relacionados con la televisión.
Por consiguiente, se pretende que todo material contenido en la descripción mencionada anteriormente y mostrado en los dibujos anexos debe ser interpretado como ilustrativo y no en un sentido limitativo.

REIVINDICACIONES

1. Receptor para el tratamiento de datos donde dicho receptor comprende un motor de tratamiento de datos genérico (36) configurado para:
recibir una definición de formato, donde dicha definición de formato comprende una descripción de una gramática que define una sintaxis de un lenguaje objetivo;
configurar dicho motor (36) para tratar datos que se formatean según l definición de formato, sensible a recibir la definición de formato;
recibir datos adicionales que se ajusten al lenguaje objetivo; y
tratar los datos adicionalmente recibidos conforme a la definición de formato.

2. Receptor según lo expresado en la reivindicación 1, además configurado para recibir una emisión que incluye los datos.

3. Receptor según lo expresado en la reivindicación 2, donde el motor (36) es posteriormente configurado para recibir la definición de formato de la emisión.

4. Receptor según lo expresado en la reivindicación 1, además configurado para recibir una emisión con la definición de formato.

5. Receptor según lo expresado en la reivindicación 1, además configurado para recibir una multidifusión con los datos.

6. Receptor según lo expresado en la reivindicación 5, donde el motor (36) es posteriormente configurado para recibir la definición de formato de la multidifusión.

7. Receptor según lo expresado en la reivindicación 1, donde la definición incluye una descripción de una sintaxis del formato.

8. Receptor según lo expresado en la reivindicación 7, donde la definición incluye una descripción de semántica del formato.

9. Receptor según lo expresado en la reivindicación 8, donde la descripción de semántica asocia al menos un identificador con los datos.

10. Receptor según lo expresado en la reivindicación 8, donde la sintaxis y semántica son descritas en un primer lenguaje.

11. Receptor según lo expresado en la reivindicación 10, donde el motor (36) es posteriormente configurado para producir una representación interna de la sintaxis y semántica.

12. Receptor según lo expresado en la reivindicación 11, donde el motor (36) es posteriormente configurado para recibir una pregunta y usa la representación interna para crear al menos una máscara que define un modelo de datos particular.

13. Receptor según lo expresado en la reivindicación 12, donde la descripción de semántica asocia al menos un identificador con los datos, y la pregunta usa al menos un identificador.

14. Receptor según lo expresado en la reivindicación 12, donde el motor (36) comprende además al menos un filtro (82) operable para aplicar al menos una máscara para filtrar los datos.

15. Receptor según lo expresado en la reivindicación 14, donde el motor (36) comprende además un objeto de características de filtro incluyendo información acerca de al menos un filtro (82), y donde el motor (36) es posteriormente configurado para usar la información de filtro para seleccionar al menos un filtro (82) para aplicar la al menos una máscara.

16. Receptor según lo expresado en la reivindicación 14, donde el motor (36) es posteriormente configurado para reenviar al menos una parte de los datos filtrados a una aplicación.

17. Receptor según lo expresado en la reivindicación 14, donde el motor (36) es posteriormente configurado para producir una máscara adicional, basada en los datos filtrados.

18. Receptor según lo expresado en la reivindicación 14, donde el motor (36) es posteriormente configurado para modificar la al menos una máscara, basada en los datos filtrados.

19. Receptor según lo expresado en la reivindicación 12, donde el motor (36) es posteriormente configurado para recibir una segunda pregunta.

20. Receptor según lo expresado en la reivindicación 19, donde el motor (36) es posteriormente configurado para crear al menos una máscara adicional, basada en la segunda pregunta.

21. Receptor según lo expresado en la reivindicación 12, donde la pregunta es formulada utilizando el primer lenguaje.

22. Receptor según lo expresado en la reivindicación 12, donde la pregunta es formulada utilizando un segundo lenguaje.

23. Receptor según lo expresado en la reivindicación 12, que comprende además un mecanismo operable para ejecutar una aplicación que formula la pregunta.

24. Receptor según lo expresado en la reivindicación 23, donde la pregunta es discreta.

25. Receptor según lo expresado en la reivindicación 23, donde la pregunta es continua.

26. Receptor según lo expresado en la reivindicación 8, donde la sintaxis es descrita en un primer lenguaje y la semántica es descrita en un segundo lenguaje.

27. Receptor según lo expresado en la reivindicación 26, donde el motor (36) es posteriormente configurado para producir una representación interna de la sintaxis y una representación interna de la semántica.

28. Receptor según lo expresado en la reivindicación 27, donde el motor (36) es posteriormente configurado para recibir una pregunta y usa las representaciones internas para crear al menos una máscara que define un modelo de datos particulares.

29. Receptor según lo expresado en la reivindicación 28 donde la descripción de semántica asocia al menos un identificador con los datos, y la pregunta usa al menos un identificador.

30. Receptor según lo expresado en la reivindicación 28, donde el motor (36) comprende además al menos un filtro (82) operable para aplicar al menos una máscara para filtro los datos.

31. Receptor según lo expresado en la reivindicación 30, donde el motor (36) comprende además un objeto de características de filtro incluyendo información acerca de al menos un filtro (82), y donde el motor (36) es posteriormente configurado para usar la información de filtro para seleccionar al menos un filtro (82) para aplicar a la al menos una máscara.

32. Receptor según lo expresado en la reivindicación 30, donde el motor (36) es posteriormente configurado para reenviar al menos una parte de los datos filtrados a una aplicación.

33. Receptor según lo expresado en la reivindicación 30, donde el motor (36) es posteriormente configurado para producir una máscara adicional, basada en los datos filtrados.

34. Receptor según lo expresado en la reivindicación 30, donde el motor (36) es posteriormente configurado para modificar la al menos una máscara, basada en los datos filtrados.

35. Receptor según lo expresado en la reivindicación 28, donde el motor (36) es posteriormente configurado para recibir una segunda pregunta.

36. Receptor según lo expresado en la reivindicación 35, donde el motor (36) es posteriormente configurado para crear al menos una máscara adicional, basada en la segunda pregunta.

37. Receptor según lo expresado en la reivindicación 28, donde la pregunta es formulada utilizando al menos uno del primer lenguaje y el segundo lenguaje.

38. Receptor según lo expresado en la reivindicación 28, donde la pregunta es formulada utilizando un tercer lenguaje.

39. Receptor según lo expresado en la reivindicación 28, que comprende además un mecanismo operable para ejecutar una aplicación que formula la pregunta.

40. Receptor según lo expresado en la reivindicación 39, donde la pregunta es discreta.

41. Receptor según lo expresado en la reivindicación 39, donde la pregunta es continua.

42. Receptor según lo expresado en la reivindicación 1, donde los datos comprenden información relacionada con la televisión.

43. Receptor según lo expresado en la reivindicación 42, donde los datos comprenden información de servicio.

44. Sistema para tratar datos formateados, que comprende:
un transmisor configurado para transmitir una definición de formato asociada a los datos, donde dicha definición de formato comprende una descripción de una gramática que define una sintaxis de un lenguaje objetivo; y
el receptor según la reivindicación 1.

45. Sistema según lo expresado en la reivindicación 44, donde los datos comprenden información relacionada con la televisión.

46. Sistema según lo expresado en la reivindicación 44, donde los datos incluyen información de formateo.

47. Sistema según lo expresado en la reivindicación 44, donde los datos excluyen información de formateo.

48. Método para actualización de un motor de tratamiento de datos genérico operable para procesar datos independiente de información de formateo, que comprende:
recibir una definición de formato, donde dicha definición de formato comprende una descripción de una gramática que define una sintaxis de un lenguaje objetivo;
configurar dicho motor para tratar datos que se formatean según el formato definición, sensible a recibir la definición de formato;
recibir datos adicionales que se ajusten al lenguaje objetivo; y
tratar los datos adicionalmente recibidos conforme a la definición de formato.

49. Método según lo expresado en la reivindicación 48, donde los datos comprenden información relacionada con la televisión.

50. Método según lo expresado en la reivindicación 48, donde la definición de sintaxis y definición de semántica son transmitidas separadamente.

51. Método según lo expresado en la reivindicación 48, donde transmitir la definición de sintaxis incluye emitir la definición de sintaxis.

52. Método según lo expresado en la reivindicación 48, donde transmitir la definición de sintaxis incluye multidifundir la definición de sintaxis.

53. Producto de programa informático para tratar datos formateados, que comprende un medio utilizable informático con código legible por máquina concretado dentro para:
recibir una definición de formato, donde dicha definición de formato comprende una descripción de una gramática que define una sintaxis de un lenguaje objetivo;
configurar un procesamiento de datos de motor (36) sensible a recibir la definición de formato;
recibir datos adicionales que se ajusten al lenguaje objetivo; y
tratar los datos adicionalmente recibidos conforme a la definición de formato.

54. Producto de programa informático según lo expresado en la reivindicación 53, donde la definición incluye una definición de sintaxis del formato.

55. Producto de programa informático según lo expresado en la reivindicación 54, donde la definición incluye una definición de semántica del formato.

56. Producto de programa informático según lo expresado en la reivindicación 55, donde el código es posteriormente configurado para producir una representación interna de la sintaxis y semántica.

57. Producto de programa informático según lo expresado en la reivindicación 56, donde el código es posteriormente configurado para recibir una pregunta y usa la representación interna para crear al menos una máscara que define un modelo de datos particulares para la filtración de los datos.

58. Producto de programa informático según lo expresado en la reivindicación 57, donde el código es posteriormente configurado para proporcionar al menos una máscara a por lo menos un filtro (82).

59. Producto de programa informático según lo expresado en la reivindicación 57, donde el código es posteriormente configurado para memorizar datos filtrados devueltos por al menos un filtro (82).

60. Producto de programa informático según lo expresado en la reivindicación 58, donde el código es posteriormente configurado para ajustar una máscara según al menos una parte de datos filtrados devuelta por al menos un filtro (82).

61. Producto de programa informático según lo expresado en la reivindicación 58, donde el código es posteriormente configurado para modificar al menos una máscara según al menos una parte de datos filtrados devuelta por al menos un filtro (82).

62. Producto de programa informático según lo expresado en la reivindicación 53, donde los datos incluyen información relacionada con la televisión.

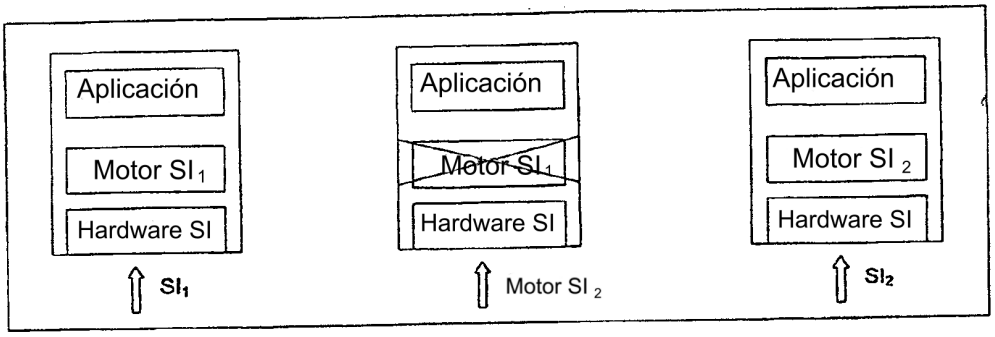

Aplicación
Motor SI 1
Hardware SI
SI1
Aplicación
Motor SI 1
Hardware SI
Motor SI 2
Aplicación
Motor SI 2
Hardware SI
SI2

FIG. 1

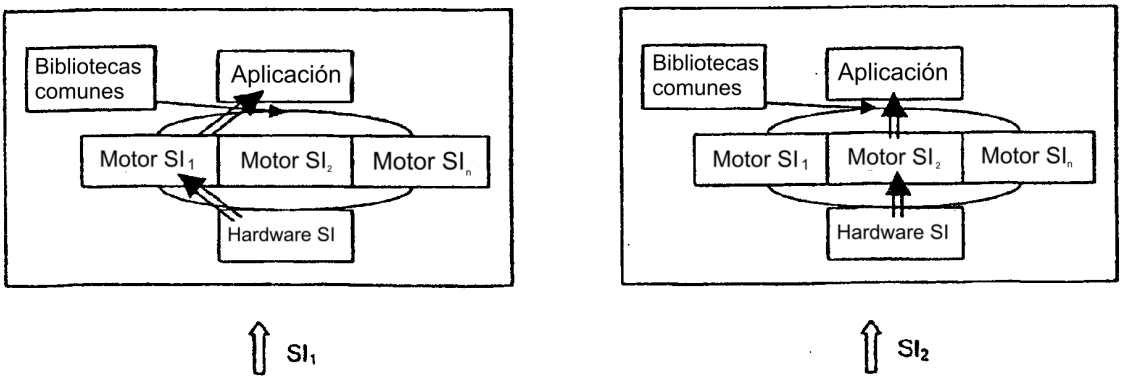

Bibliotecas comunes
Aplicación
Motor SI 1
Motor SI 2
Motor SI n
Hardware SI
SI1
Bibliotecas comunes
Aplicación
Motor SI 1
Motor SI 2
Motor SI n
Hardware SI
SI2

FIG. 2

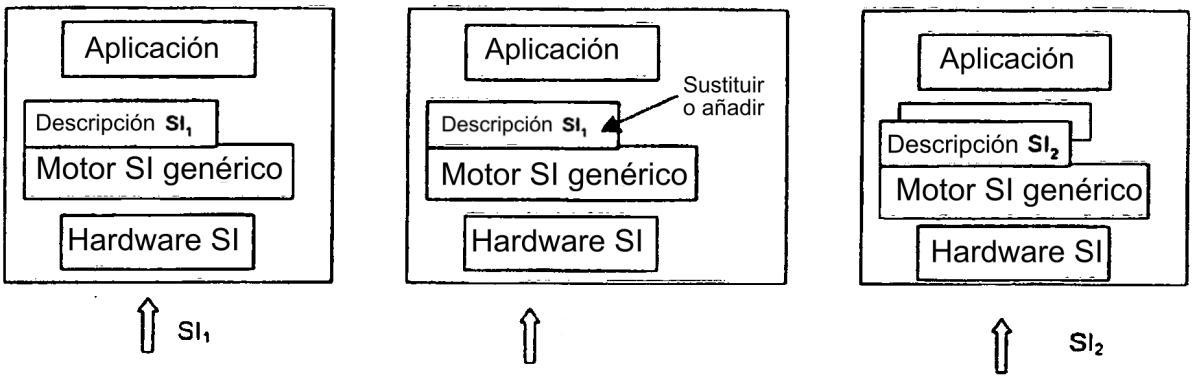

Aplicación
Descripción SI1
Motor SI genérico
Hardware SI
SI1
Aplicación
Descripción SI1
Sustituir
o añadir
Motor SI genérico
Hardware SI
Aplicación
Descripción SI2
Motor SI genérico
Hardware SI
SI2

FIG. 3

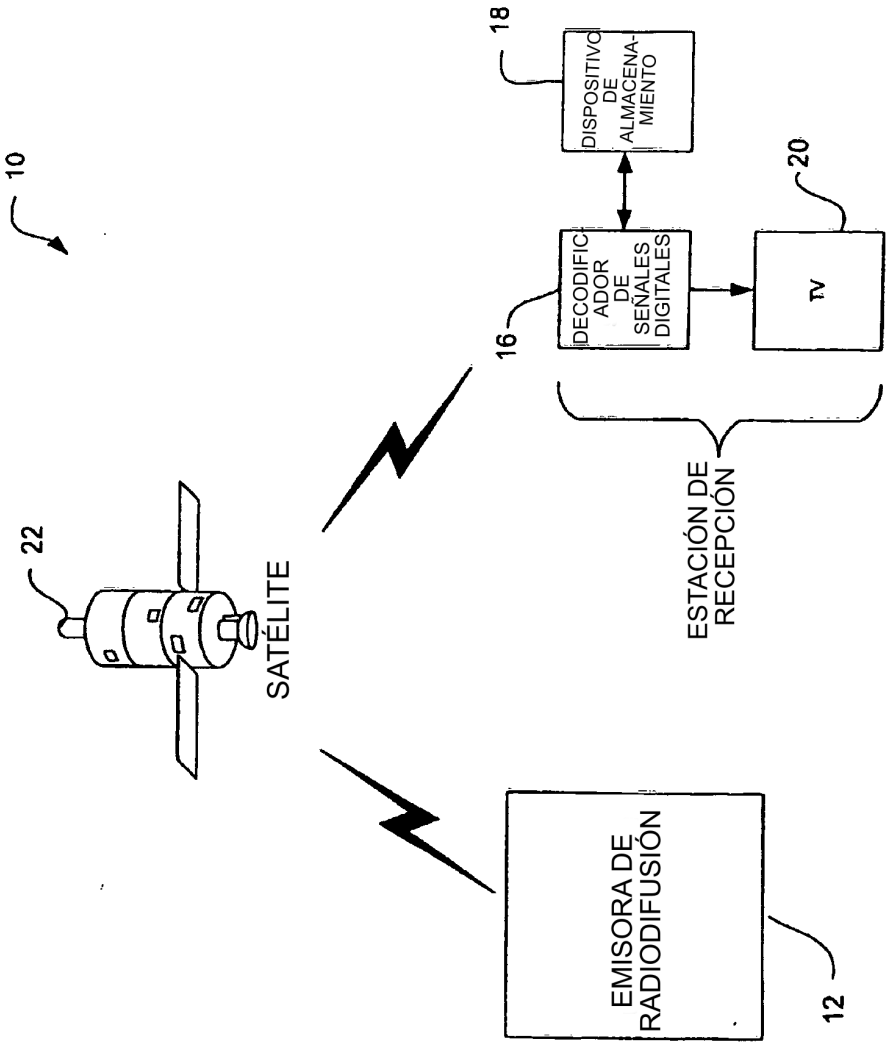

10
22
SATÉLITE
18
DISPOSITIVO DE ALMACENA-MIENTO
20
DECODIFIC ADOR DE SEÑALES DIGITALES
TV
16
ESTACIÓN DE RECEPCIÓN
EMISORA DE RADIODIFUSIÓN
12

FIG. 4

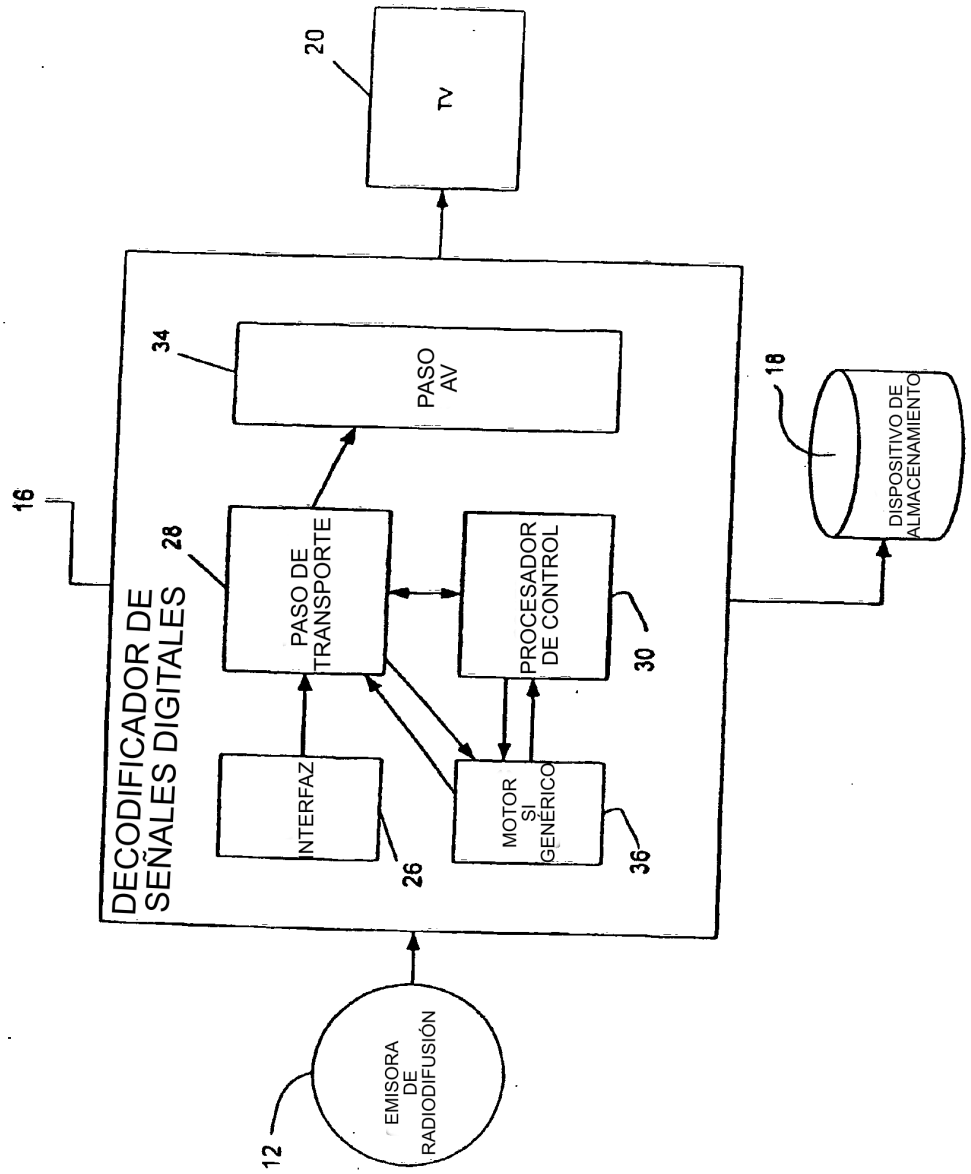

20
TV
34
PASO AV
18
DISPOSITIVO DE ALMACENAMIENTO
16
DECODIFICADOR DE SEÑALES DIGITALES
28
PASO DE TRANSPORTE
PROCESADOR DE CONTROL
30
INTERFAZ
26
MOTOR SI GENÉRICO
36
EMISORA DE RADIODIFUSIÓN
12

FIG. 5

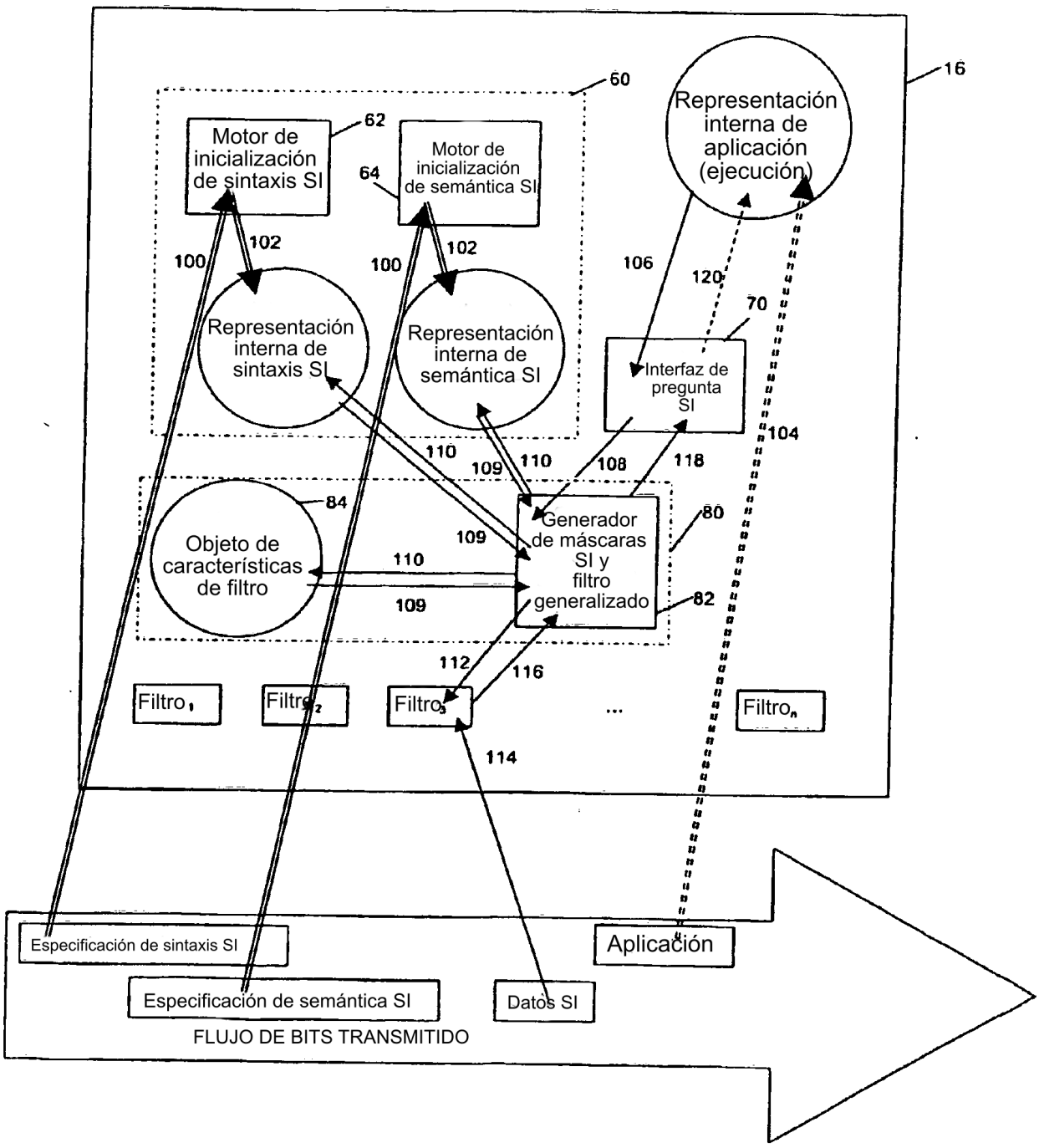

16
60
62
64
Motor de inicialización de sintaxis SI
Motor de inicialización de semántica SI
Representación interna de aplicación (ejecución)
100
102
Representación interna de sintaxis SI
Representación interna de semántica SI
106
120
70
Interfaz de pregunta SI
104
110
109
108
118
84
Objeto de características de filtro
Generador de máscaras SI y filtro generalizado
80
82
112
116
Filtro1
Filtro2
Filtro3
...
Filtron
114
Especificación de sintaxis SI
Aplicación
Especificación de semántica SI
Datos SI
FLUJO DE BITS TRANSMITIDO

FIG. 6

```
Description ::= StructureDefinitions
StructureDefinitions ::= StructureDefinition MoreStructureDefinitions
MoreStructureDefinitions ::= StructureDefinition | λ
StructureDefinition ::= StructureName "{" StructureBody "}"
StructureName ::= Variable
StructureBody ::= OptHeader FieldDefs
OptHeader ::= "type" "=" TypeName OptFilterValues";" | λ
TypeName ::= Variable
OptFilterValues ::= "," "filterValues" "=" FilterPairs | λ
FilterPairs ::= FilterPair OptMoreFilterPairs
OptMoreFilterPairs ::= FilterPair OptMoreFilterPairs | λ
FilterPair ::= "(" FilterField "," FilterValue ")"
FilterField ::= Variable
FilterValue:: PosInteger
FieldDefs ::= FieldDef OptMoreFieldDefs
OptMoreFieldDefs ::= FieldDef OptMoreFieldDefs | λ
FieldDef ::= SimpleFieldDef  | ComplexFieldDef
SimpleFieldDef ::= FieldName "," FieldSize "," SimpleFieldFormat OptValues ";"
FieldName ::= Variable | "reserved"
FieldSize ::= PosInteger
SimpleFieldFormat ::= PredefinedType
PredefinedType ::= Variable
OptValues ::= "," "=" Value OptValues | λ
Value ::= PosInteger
ComplexFieldDef ::= LoopDef | StructDef SEMICOLON | AlternateDef
LoopDef ::= "loop" "," LoopLength "," "{" FieldDefs "}"
LoopLength ::= Term OptMoreLoopLength
Term ::= PosInteger | StructVariable
OptMoreLoopLength ::= Operation Term OptMoreLoopLength | λ
Operation ::= "+" | "-"
StructVariable ::= Variable OptMoreVariables
OptMoreVariables ::= "." Variable OptMoreVariables | λ
StructDef ::= StructureName "(" ")"
AlternateDef ::= "alternate" "{" FieldDefs "}"
Variable ::= Letter RestrictedCharSet
RestrictedCharSet ::= Letter | Digit | "_" | λ
PosInteger ::= Digit MoreDigits
MoreDigits ::= Digit MoreDigits
Letter ::= "a" | "b" | … | "z" | "A" | "B" | … | "Z"
Digit ::= "0" | "1" | … | "9"
```

FIG. 7

```
network_information_section {type = MPEG, filterValues = (PID, 8);
       table_id, 8, uimsbf, = 64, = 65;
       section_syntax_indicator, 1, bslbf;
       reserved, 1, bslbf;
       reserved, 2, bslbf;
       section_length, 12, uimsbf;
       network_id, 16, uimsbf;
       reserved, 2, bslbf;
       version_number, 5, uimsbf;
       current_next_indicator, 1, bslbf;
       section_number, 8, uimsbf;
       last_section_number, 8, uimsbf;
       reserved, 4, bslbf;
       network_descriptors_length, 12, uimsbf;
       loop, network_descriptor_length, {NITDescriptor();}
       reserved, 4, bslbf;
       transport_stream_loop_length, 12, uimsbf;
       loop, transport_stream_loop_length, {transport_stream_id, 16, uimsbf;
              original_network_id, 16, uimsbf;
              reserved, 4, bslbf;
              transport_descriptors_length, 12, uimsbf;
              loop, transport_descriptors_length,{NITDescriptor();}}}
NITDescriptor{alternate {network_name_descriptor(); service_list_descriptor();…
              data_broadcast_id_descriptor();}}
```

FIG. 8

```
struct complexType {
        { this struct would be used to represent any structure or complex field}
        string Name;  {name of the structure or field; used to match value from semantic analyzer}
        complexTypeKind kind; {enumerated type: one of structure, loop, or alternate}

        string baseProtocolType;  { for structures that have types }
                { above field is a handle allowing determination of the name of the lower level filter
                        and the mask representing the bits that the lower level filter is capable of
                        filtering on.}
        couplets *baseProtocolFieldValues;
                { this field shall contain a ptr to a list of memory locations that will contain the filter
                        pairs}
        alternativeList_t *alternatives;
                { pointer to a list of all of the possible alternatives for fields of this field or structure }
}tableType_t;

struct alternativeList {
        string Name;
        fieldDescription_t *fieldDescriptions
        { this would be a pointer to a linked list of lists of descriptions of the alternate lists of fields;
        since it is possible to know how many elements need to be in this list, space for it can be
        dynamically allocated and efficient access using pointer arithmetic is possible. }
        alternativeList_t *next;
}alternativeList_t;

struct fieldDescription {
        { this particular representation assumes that exactly enough space to hold all of the fields is
        allocated allowing direct access to any field without walking through a linked list, simply by
        using pointer arithmetic -- hence, there is no "next" field at the end.}
        string fieldName;  { this is also used to match value from semantic analyzer.}
        fieldTypeKind_t fieldTypeKind;  { enumerated type: one of basic, structure, or loop.}
        fieldType_t *fieldType; { if basic, this is a pointer to one of the basic types listed in the
                system encoding type tables;  in example above, these would be uimsbf and bslbf.
                if structure or loop, this field is a pointer to a linked list of tableType_t structures, any
                of which could occupy this field.}
        indexList_t *lengthList { see below for definition of structure pointed to by this field; allows
the
                length of the field to be a fixed constant, a value provided in another field (of any
                structure or loop), or an arbitrary sum or difference of any of these. }
        indexList_t *offsetList {same format as lengthList, and computed/computable from the
                lengthLists, but storing this valueallows direct access to this fieldDescription}
        value_list_t *valueList; { this is a list of values that have been specified for this field}
}fieldDescription_t;

struct indexList{
        enum valueKind; { one of fixed_constant, reference_field or variable.
        fixed_constant means a fixed number of bits;  reference_field means that the length is given
        by another field in this (or another) structure, and variable means that it is not known.}
        valuePtr_t *valuePtr;{ this would be a pointer to an integer if type is fixed_constant, another
                field if type is reference_field; or pointer is null if type is variable}

        int multiplier; { either +1 or -1; allows arbitrary sums/differences for length }
        indexList *next;  { in case this value is a sum of other values}
}indexList_t;
```

FIG. 9

```
Define int Movie = 1;
Define string Actor = 'Actor';
Define string ProductionCompany = 'Production Company';
Define Object eventInfo = (
fetch(event_information_section,
filter: table_id > 90, table_id <= 113;
return:
name := loop_1 = outer_loop.loop1.extended_event_descriptor
= named_descriptor.loop_2.text_char[],
start_time := outer_loop.start_time,
end_time := compute(outer_loop.start_time + outer_loop.duration))
where
channel(relop relationalOp, int value) =
{obtain triplet := identifyingTrio where channel(relationalOp, value);
set(event_information_section,
       filter: original_network_id == triplet.original_network_id,
       transport_stream_id == triplet.transport_stream_id,
       service_id == triplet.service_id);}
/* See below for the definition of the identifyingTrio object */
eventType(relop relationalOp, int value) =
{set(event_information_section,
       filter: outer_loop.loop1.descriptor.content_descriptor.loop_1.
       content_nibble_1 relationalOp value); }
instantiate(relop relationalOp1, string str1, relop relationalOp2,
       string str2) =
{set(event_information_section,
       filter: named_descriptor.loop_1=chosen_loop.item_description_char
       relationalOp str1,
       chosen_loop1.item_char relationalOp2 str2); }
startTime(relop relationalOp, int t1) =
/* assumes that application program has already converted the requested t1 into seconds
past midnight local time. Also, assumes that the following global variables are known,
all in seconds */
{set (event_information_section, filter: section_number >=
compute( ( (t1-Current_local_time) +
( ( Current_local_time - UTCDifference) mod 24) div 3) * 8),
outer_loop.start_time relationalOp t1);}
endTime(relop relationalOp, int t2) =
{set(event_information_section,
filter:
section_number <= compute ( ( ( ( ( t2 - Current_local_time) +
       ( ( Current_local_time - UTCDifference) mod 24 ) )
       div 3) * 8) + 7),
       compute(outer_loop.start_time + outer_loop.duration)
       relationalOp t2);}
```

FIG. 10a

```
event_id(relop relationalOp, int value) =
{set(event_information_section,
       filter: outer_loop.event_id relationalOp value); })
Define Object identifyingTrio = (
fetch(channel_correspondence_section,
return:
original_network_id :=
       loop_1 = channel_loop.original_network_id,
transport_stream_id := channel_loop.transport_stream_id,
service_id := channel_loop.service_id)
where
channel(relop relopetionOp, int value) =
       {set(channel_correspondence_section,
       filter: channel_number relationalOp value);})
```

FIG. 10b

Program ::= Definitions
Definitions ::= Definition OptMoreDefinitions
OptMoreDefinitions::= λ | Definition OptMoreDefinitions
Definition ::= DEFINE RestOfDef
RestOfDef ::= ConstantDef | ObjectDef
ConstantDef ::= Type VARIABLE EQUALS Value SEMICOLON
Type ::= INTTYPE | STRINGTYPE
Value ::= INTEGER| STRING
ObjectDef ::= OBJECT ObjName EQUALS LEFTCURLY Fetches OptRestObjDef RIGHTCURLY
ObjName ::= VARIABLE
Fetches ::= Fetch OptMoreFetches
OptMoreFetches ::= λ | Fetch OptMoreFetches
Fetch ::= FETCH LPAREN StructureName COMMA OptFilterPart ReturnPart RPAREN
StructureName ::= VARIABLE
OptFilterPart::= λ | FilterPart
ReturnPart ::= RETURN COLON ReturnStmts
ReturnStmts ::= ReturnStmt OptMoreReturnStmts
OptMoreReturnStmts ::= λ | COMMA ReturnStmt OptMoreReturnStmts
ReturnStmt ::= FieldName ASSIGNOP FieldValue
FieldName ::= VARIABLE
FieldValue ::= PathValue | ComputeExpression
PathValue ::= PathName OptArrayInd
OptArrayInd ::= λ | LSQUARE RSQUARE
PathName ::= NarrowName OptTempName OptRestOfPath
OptTempName ::= λ | EQUALS TempName
NarrowName ::= VARIABLE
TempName ::= VARIABLE
OptRestOfPath ::= λ | DOT PathName
ComputeExpression ::= COMPUTE LPAREN Expr RPAREN
Expr ::= Term RestOfExpr
RestOfExpr ::= λ | BinOperator Expr
Term::= CompoundTerm | SimpleTerm
CompoundTerm ::= LPAREN Expr RPAREN
SimpleTerm ::= INTEGER | PathName
BinOperator ::= PLUS | MINUS | TIMES | DIV | MOD | MIN | MAX
OptRestObjDef ::= λ | WHERE Methods
Methods ::= Method OptMoreMethods
OptMoreMethods ::= λ | Method OptMoreMethods
Method ::= MethodName LPAREN ParamList RPAREN EQUALS MethodDefn
MethodName ::= VARIABLE
ParamList ::= λ | Parameter OptMoreParameters
OptMoreParameters ::= λ | COMMA Parameter OptMoreParameters
Parameter ::= SpecifiedArg | ConstantSpec | RelSpec
RelSpec ::= RELOP RelOpVariable
SpecifiedArg ::= VARIABLE
ConstantSpec ::= Type ConstantVar
ConstantVar ::= VARIABLE
RelOpVariable ::= VARIABLE
MethodDefn ::= LEFTCURLY OptAcquisitions FilterSets RIGHTCURLY
OptAcquisitions ::= λ | Acquisition OptAcquisitions
FilterSets ::= FilterSet OptMoreFilterSets
OptMoreFilterSets ::= λ | FilterSet OptMoreFilterSets
FilterSet ::= SET LPAREN StructureName COMMA FILTER COLON Filters RPAREN SEMICOLON
FilterPart ::= FILTER COLON Filters SEMICOLON
Filters ::= CompOperand OptArrayInd Comparator CompOperand OptMoreFilters

FIG. 11a

```
OptMoreFilters ::= λ | COMMA Filters
Comparator ::= RelOpVariable | DBLEQUALS | GT | GTEQ | LTEQ | LT | NOTEQUALS
CompOperand ::= PathName | ComputeExpression | INTEGER | STRING
Acquisition ::= OBTAIN ObjectInstantiation ASSIGNOP ObjName OptConstraint SEMICOLON
OptConstraint ::= WHERE OptConstraints
ObjectInstantiation ::= VARIABLE OptArrayInd
OptConstraints ::= λ | Constraint OptConstraints
Constraint ::= MethodName LPAREN ActualParamList RPAREN
ActualParamList ::= ActualParam OptMoreActualParams
OptMoreActualParams ::= λ | COMMA ActualParam OptMoreActualParams
ActualParam ::= VARIABLE | INTEGER
```

FIG. 11b

```
Selec.todos SoughtLoop = loop_1 de event_information_section donde
        //evento es de tipo "noticias"
        (content_nibble_1 from content_descriptor from SoughtLoop = = newstype)
                //donde el tipo de noticias es un constante predefinida a la que se ha asignado el valor 2
         y
        y está en el mismo bouquet que el actual_transport_stream actual
        (transport_stream_id = =
                (Select transport_stream_id de MidLoop = loop_2 de
                CurrentBouquet = bouquet_association_section donde
                        ((Select transport_stream_id
                                de OtherMidLoop = loop_2 //ok =MidLoop
                                de CurrentBouquet = =
                                        (Selec.transport_stream_id de
                                        Actual_NIT = network_information_section
                                         de table_id = = 64)) y
                        (Selec.original_network_id
                                de OtherMidLoop
                                de CurrentBouquet = =
                                (Select network_id from Actual_NIT)))
         y
        (original_network_id = =
                (Selec.original_network_id from MidLoop de CurrentBouquet))
        y se porta por cable (en lugar de por satélite o terrestre)
        y
        transport_stream_id = =
                (Select transport_stream_id from DeliveryLoop = loop_2 de
                        AnyNIT = network_information_section donde
                        existe cable_delivery_system_descriptor de DeliveryLoop de AnyNIT)
        y
        original_network_id ==
                (Select original_network_id de DeliveryLoop de AnyNIT)
        y su tiempo está en el rango solicitado
        y
        DVB_time_Between(RequestedStartTime, RequestedEndTime, SoughtLoop.start_time,
                SoughtLoop.duration)
```

FIG. 13

```
DefinitionList_t *PtrToDefns;
struct DefinitionList {Defn_t *DefPtr;DefinitionList_t *NextDefn;}DefinitionList_t;
struct Defn{enum DefnType; {One of "Integer", "String", or "Object"}
        string Name; {VARIABLE from ConstantDef or ObjName from
                ObjectDef}
        void *valuePtr; {points to an integer, a string or  ObjParts_t
                structure (defined below)}} Defn_t;
struct ObjectParts{FetchList_t *FList;MethodList_t *MethodList;}ObjParts_t;
struct MethodList{MethodStruct_t *MethodPtr;MethodList_t *NextMethod;} MethodList_t;
struct FetchList{FetchStruct_t *FStruct; FetchList_t *NextFetch;}FetchList_t;
struct FetchStruct{string StructureName;FilterList_t *FiltList; ReturnList_t *RtrnList;}FetchStruct_t;
struct FilterList{FilterStruct_t *FiltStruct;FilterList_t *NextFilter;}FilterList_t;
struct FilterStruct{string StructureName;FilterFieldList_t *FilterFields;}FilterStruct_t;
struct FilterFieldList{FilterFieldStruct_t *FilterField; FilterFieldList_t *NextFilterField;}FilterFieldList_t;
struct FilterFieldStruct{
        void *FirstOperand; {points to one of a pathList_t, ComputeStack_t, an integer, or a string }
        void *RelationalOp; {points to a string (if operator can
```

FIG. 12a

```
                vary) or an enum containing one of eq, gt, gteq, etc. )}
        enum RelationalType; { one of "string" or "actual"}
        void *SecondOperand; { same type as FirstOperand }}FilterFieldStruct_t;
struct ReturnList{ReturnStruct_t *RtrnStruct; ReturnList_t *NextRtrn;}ReturnList_t;
struct ReturnStruct{string NewFieldName;
        void *FieldValue; { ptr to one of a pathList_t or ComputeStack_t}
        enum type; { one of "path" or "computeStack" }}ReturnStruct_t;
struct pathList{string Name; string tempName; {used in scope to mean that a single
        (sub-)structure is multiply constrained} pathList_t *nextName;
        enum Type; { one of "array" or "scalar"}} pathList_t;
struct ComputeStack{void *ItemPtr; {points to item to push on list}
        {Item is of type integer, pathname, or operator}
        {operator is an enum type containing one of: plus, minus, times, div, min, max, or mod}
        enum type; {one of "integer", "pathname" or "operator" }
        ComputeStack_t *nextToPushOn;}ComputeStack_t;
struct MethodStruct{string MethodName; ParamList_t *ParamListPtr;
        MethodDefnStruct_t *MethodDefn;}MethodStruct_t;
struct MethodDefnStruct{AcqList_t *AcqList; FilterList_t *FilterList;}MethodDefnStruct_t;
struct AcqList{AcqStruct_t *AcqPtr; AcqList_t *NextAcq;}AcqList_t;
struct AcqStruct{string NewObjName; enum ObjType; { one of "scalar" or "array"}
        string OldObjName; ConstraintList_t *Constraints;} AcqStruct_t;
struct ConstraintList{
```

FIG. 12b

```
        ConstraintStruct_t *Constraint; ConstraintList_t *nextConstraint;}ConstraintList_t;
struct ConstraintStruct{string MethodName; ActualParamList_t *ActualParams;} ConstraintStruct_t;
struct ActualParamList{void *ActualParam; { points to a string or an int}
        enum type; {one of "string" or "integer"} ActualParamList_t *NextParam;}ActualParamList_t;
struct ParamList{void_t *Param; {ptr to type SpecifiedArg_t, ConstantSpec_t, or RelSpec_t}
        enum type; { one of SpecifiedArg, ConstantSpec, or RelSpec}ParamList_t *NextParam;
}ParamList_t;
struct SpecifiedArg{string Argument;}SpecifiedArg_t;
struct ConstantSpec{enum type; { one of "integer" or "string"}
        void *value; { pointer to an integer or a string}} ConstantSpec_t;
struct RelSpec{string RelOpVariable;}RelSpec_t;
```

FIG. 12c

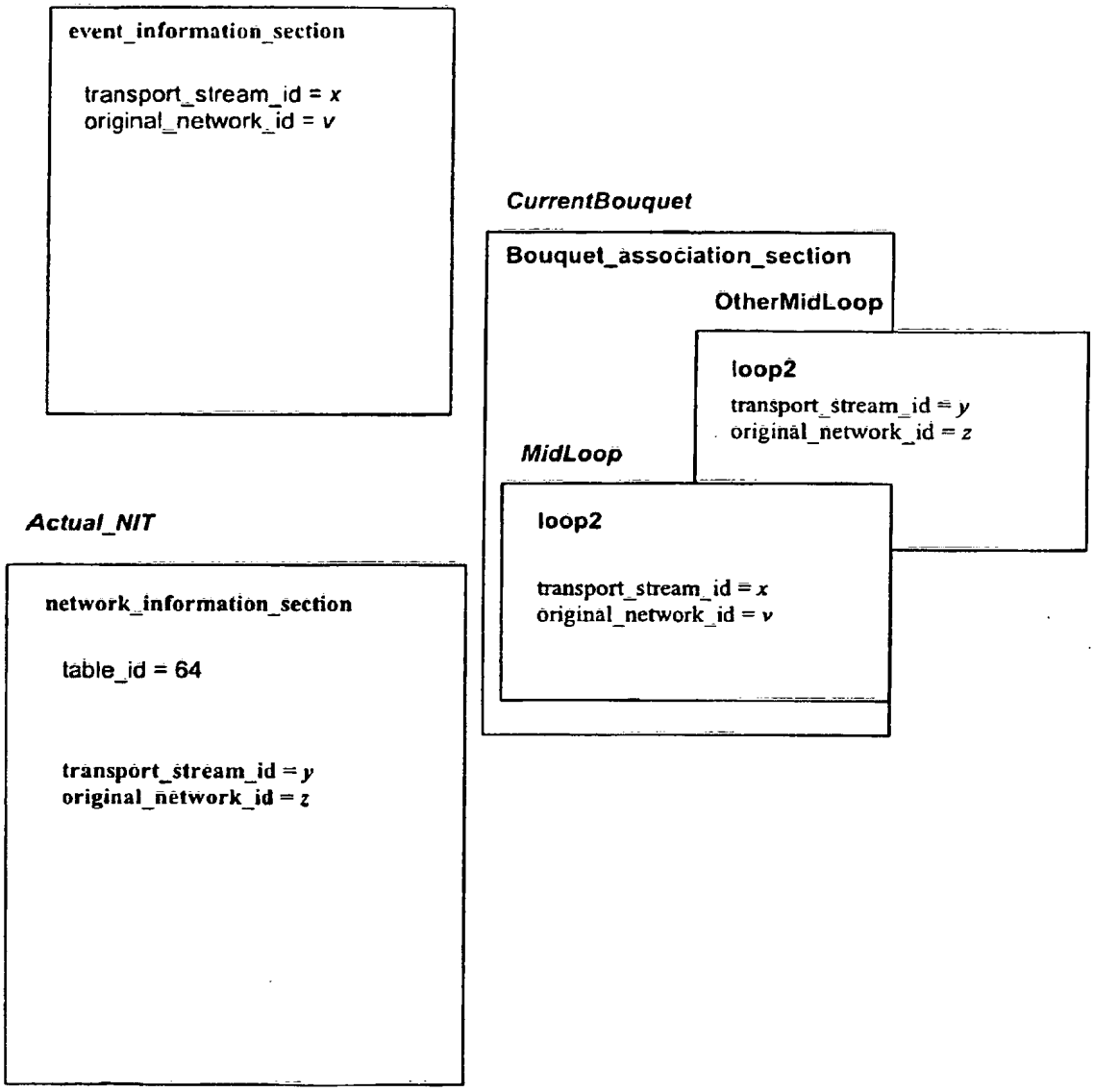

event_information_section
transport_stream_id = x
original_network_id = v
CurrentBouquet
Bouquet_association_section
OtherMidLoop
loop2
transport_stream_id = y
original_network_id = z
MidLoop
loop2
transport_stream_id = x
original_network_id = v
Actual_NIT
network_information_section
table_id = 64
transport_stream_id = y
original_network_id = z

FIG. 14

```
// expansión de DVB_time_Between de la figura previa (casi completo-ver último comentario más abajo para
//un par de detalles no incluidos deliberadamente)
start_time >=RequestedStartTime
y
start_time + duration <=RequestedEndTime
y
time_since_midnight=CurrentLocalTime-(TimeDiff)(mod24)
//CurrentLocalTime y la compensación de UTC-0 es generalmente guardado en un
//decodificador de señales digitales-si no lo están, se pueden obtener de tablas TOT o TDT.
Y
StartINUTC0=max(0, (RequestedStartTime)-CurrentLocalTime)-time_since_midnight)
Y
(StartINUTC0 dic 3)*8<=section_number<= ((EndInUTC0 div 3)*8)+7
y
this_section_number=section_number<=last_section_in_segment_number de event_information_section
donde
        first_section_in_segment=section_number de event_information_section donde
                first_section_in_segment <=this_section_number and
                first_section_in_segment div 8 ==0
//Debería ser código al principio convirtiendo todos los parámetros y constantes conocidas (incluyendo
//RequestedStartTime, RequestedEndTime, CurrentLocalTime y TimeDiff) así como los valores
//de tiempo extraídos de cada bucle event_information_section (incluyendo start_time y duration) al
//número de minutos desde medianoche GMT, 1 de enero de 1970, o alguna otra fecha de referencia. Son
//los valores de referencia que se espera utilizar para las comparaciones de arriba.
```

FIG. 15

```
ActualTransportStreamID(X) :- program_association_table(A), is_field(A, transport_stream_id, X).
//X es el identificador de flujo de transporte real si la sección de asociación de programa (A) tiene, como un
//campo transport_stream_id con un valor de X.
TransportStreamInBouquet(C, B) :-bouquet_association_section(L), is_field (L, bouquet_id, B),
        is_field(L, transport_stream_id, C).
//C es en Bouquet con bouquet_id B si una bouquet_association_section L tiene un campo bouquet_id
//con valor de B y un campo transport_stream_id con un valor de C.
TransmissionMedia(X, media) :- ActualTransportStreamID(X),
        //caso 1: X es el flujo de transporte real
        network_information_section(N), is_field(N, table_id, 32), is_field(N, loop1, L),
        is_field(L, descriptor, D), frequency_list_descriptor(D), is_field(D, coding_type, media).
//La media de transmisión del flujo de transporte X es "media" si X es el flujo
//de transporte real ID, N es una network_information_section del flujo actual, L es bucle 1 de N
//D es un frequency_list_descriptor en L y D tiene un campo coding_type para esa "media".
TransmissionMedia(X, media) :- network_information_table(N), is_field(N, loop2, L),
        is_field(L, transport_stream_id, X), is_field(L, descriptor, D), frequency_list_descriptor(D),
        is_field(D, coding_type, media).
//Otra posibilidad para X con una media de transmisión "media"
//Aquí N puede ser cualquier network_information_table donde el segundo bucle contiene el
//transport_stream_id en cuestión y tiene un campo coding_type de valor esa "media"
```

FIG. 16

```
GetAllEITEvents(X, [LocalStartTime,LocalEndTime], DefiniteEvents) :- SchedulesAreBroadcast(X),
        //Esta regla dice cómo obtener todos los EIT que contienen horarios para el flujo
        //de transporte cuyo id es X para eventos que ocurren en el rango de tiempo local.
        //paso 1: determinar si alguno de los servicios portado en el flujo de transporte cuyo id es x tiene su
        //horario de emisión. Si no lo tiene, no habrá EIT, informar de ello.
ObtainLocalTime(CurrentTime),
LocalEndTime > CurrentTime,
ComputeOffsetRange([LocalStartTime, LocalEndTime], CurrentTime, OffsetRange),
        //paso 2: determinar la hora local y lo lejos que está de (al menos parte de) este rango. Si el rango
        //completo ha pasado, informar de error. De lo contrario, tenemos un nuevo rango, al que llamaremos rango de compensación.
(OffsetStart, OffsetEnd).
GetUTC_0Time(CurrentUTC_0),
        //paso 3: determinar la hora actual en UTC-0, suponiendo que se transmite
        //como un reloj de 24 horas, por lo que es el  de horas desde medianoche "hoy".
TimeSince12Range(CurrentUTC_0, OffsetRange, RangeSince12),
        //paso 4: añadir las horas desde medianoche obtenidas en paso 3 tanto a OffsetStart como OffsetEnd para
        //determinar el bloque de tiempo desde medianoche UTC-0 para el que se desea el horario.
TranslateTimeToEITSegmentNumber(RangeSince12, SegmentStart, SegmentEnd),
        //paso 5: determinar el rango de segmento y solicitar el horario EIT en esos rangos de segmento.
GetEITEventsInRange(SegmentStart, SegmentEnd, X, PossibleEvents),
CheckTimes(LocalStartTime, LocalEndTime, PossibleEvents, DefiniteEvents).
```

FIG. 17a

```
SchedulesAreBroadcast(X) :- service_description_table(S), is_field(S, transport_stream_id, X),
        is_field(S, loop1, L), is_field(L, EIT_schedule_flag, Flag), Flag = 1.
```

FIG. 17b

```
GetEITEventsInRange(SegmentStart, SegmentEnd, X, Events) :- SegmentEnd > SegmentStart,
        event_information_section(S), is_field(S, transport_stream_id, X),
        is_field(S, section_number, SegmentStart),  is_field(S, segment_last_section_number, SLN),
        is_field(S, loop_1, L), Append( L, Events), GetRestOfEITSegment(SegmentStart+1, SLN, X, Events),
        NewStart = SegmentStart + 8, GetEITsInRange(NewStart, SegmentEnd, X, Events).
GetRestOfEITSegment(SegmentStart, SegmentEnd, X, Events) :- SegmentEnd >= SegmentStart,
        event_information_section(S), is_field(S, transport_stream_id, X),
        is_field(S, section_number, SegmentStart), is_field(S, segment_last_section_number, SLN),
        is_field(S, loop_1, L), Append( L, Events), GetRestOfEITSegment(SegmentStart+1, SLN, X, Events).
```

FIG. 17c

```
ObtainLocalTime(CurrentTime) :- time_offset_section(S), is_field(S, UTC_time, CurrentMJD_UTC_0),
        is_field(S, loop1, L), is_field(L, descriptor, D),
        local_time_offset_descriptor(D), is_field(D, local_time_offset_polarity, P),
        is_field(D, local_time_offset, Offset),
        ConvertToHoursSince_1_1_1900(CurrentMJD_UTC_0, CurrentTime, P, Offset).
        //ConvertToHoursSince_1_1_1900 no mostrado – se puede implementar usando fórmulas
        //aritméticas copiadas casi idénticamente de DVB EN300_468, Anexo C – nótese
        //que (-1) se aumenta al poder P y se multiplica por la compensación (offset).
GetUTC_0Time(CurrentUTC_0) :- time_date_section(S), is_field(S, UTC_time, CurrentMJD_UTC_0),
        ConvertToHoursSince_1_1_1900(CurrentMJD_UTC_0, CurrentUTC_0, 0, 0).
ComputeOffsetRange([LocalStartTime, LocalEndTime], CurrentTime, [OffsetStart, OffsetEnd]) :-
        LocalStartTime > CurrentTime, OffsetStart = LocalStartTime – CurrentTime, OffsetEnd =
        LocalEndTime – CurrentTime.
        //si el rango completo está después de la hora actual – de lo contrario, véase la siguiente regla
ComputeOffsetRange([LocalStartTime, LocalEndTime], CurrentTime, [OffsetStart, OffsetEnd]) :-
        OffsetStart = 0, OffsetEnd = LocalEndTime – CurrentTime.
TimeSince12Range(CurrentUTC_0, [OffsetStart, OffsetEnd], [StartTimeSince12, EndTimeSince12]):-
        StartTimeSince12 = CurrentUTC_0 + OffsetStart,
        EndTimeSince12 = CurrentUTC_0 + OffsetEnd.
```

FIG. 17d

```
TranslateTimeToEITSegmentNumber([StartTimeSince12, EndTimeSince12], SegmentStart, SegmentEnd):-
        SegmentEnd = ( (EndTimeSince12 div 3) * 8 ) +7,
        SegmentEnd < 256,
        SegmentStart = (StartTimeSince12 div 3) * 8).
        //usar la de arriba si todas las horas están en el rango – de lo contrario, usar la de abajo
TranslateTimeToEITSegmentNumber([StartTimeSince12, EndTimeSince12], SegmentStart, SegmentEnd):-
        SegmentEnd = ( (EndTimeSince12 div 3) * 8 ) +7,
        SegmentEnd > 255, SegmentEnd = 255,
        SegmentStart = (StartTimeSince12 div 3) * 8).
```

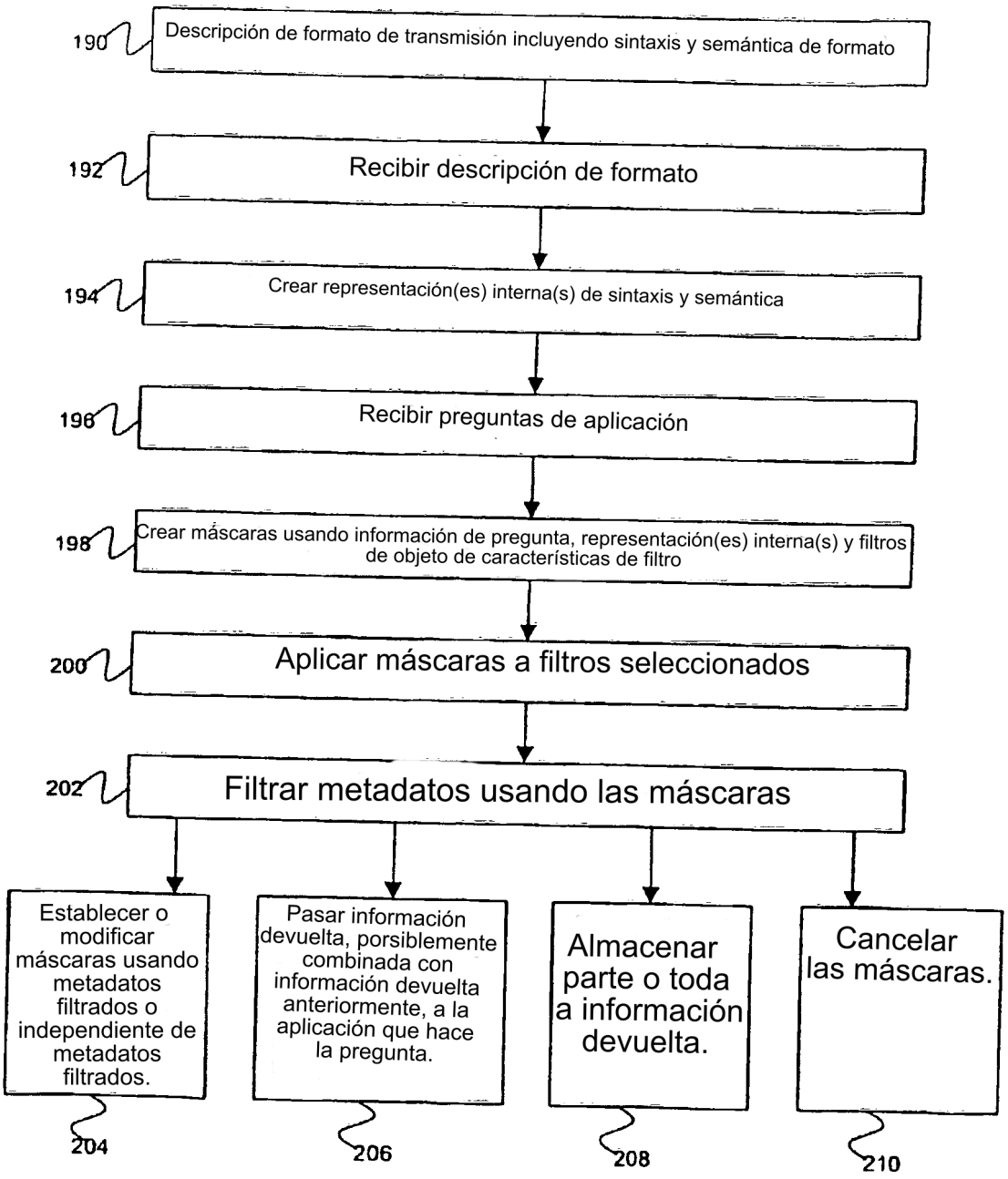
FIG. 17e

```
CheckTimes(LocalStartTime, LocalEndTime, [FirstEvent | MoreEvents], DefiniteEvents) :-
        //un evento corresponde a los contenidos del primer bucle de un EIT.
        is_field(FirstEvent, start_time, Start), Start >= LocalStartTime,
        is_field(FirstEvent, duration, duration), LocalEndTime >= LocalStartTime + duration,
        Append(FirstEvent, DefiniteEvents),
        CheckTimes(LocalStartTime, LocalEndTime, [MoreEvents], DefiniteEvents).
CheckTimes(LocalStartTime, LocalEndTime, [FirstEvent | MoreEvents], DefiniteEvents) :-
        //un evento corresponde a los contenidos del primer bucle de un EIT.
        is_field(FirstEvent, start_time, Start), Start < LocalStartTime,
        CheckTimes(LocalStartTime, LocalEndTime, [MoreEvents], DefiniteEvents).
CheckTimes(LocalStartTime, LocalEndTime, [FirstEvent | MoreEvents], DefiniteEvents) :-
        //un evento corresponde a los contenidos del primer bucle de un EIT.
        is_field(FirstEvent, start_time, Start), Start >= LocalStartTime,
        is_field(FirstEvent, duration, duration), LocalEndTime < LocalStartTime + duration,
        CheckTimes(LocalStartTime, LocalEndTime, [MoreEvents], DefiniteEvents).
```

FIG. 17f

```
?- event(X), eventTitle(X, Y), eventType (X, "news"), eventsTS(X, A), TransportStreamInBouquet(A, B),
ActualTransportStreamID(Z), TransportStreamInBouquet(Z, B),
GetAllEITEvents(A, [[June, 13, 2000, 0930],[June,13,2000,1300], DefiniteEvents],
is_member(X, DefiniteEvents), networkType(A, "cable").
```

FIG. 18


190
Descripción de formato de transmisión incluyendo sintaxis y semántica de formato
192
Recibir descripción de formato
194
Crear representación(es) interna(s) de sintaxis y semántica
196
Recibir preguntas de aplicación
198
Crear máscaras usando información de pregunta, representación(es) interna(s) y filtros de objeto de características de filtro
200
Aplicar máscaras a filtros seleccionados
202
Filtrar metadatos usando las máscaras
Establecer o modificar máscaras usando metadatos filtrados o independiente de metadatos filtrados.
204
Pasar información devuelta, porsiblemente combinada con información devuelta anteriormente, a la aplicación que hace la pregunta.
206
Almacenar parte o toda a información devuelta.
208
Cancelar las máscaras.
210