ACTEMAT
(ACCESIBILIDAD A TEXTOS MATEMÁTICOS)

- Traducción de lenguaje matemático a lenguaje natural para interfaces sonoras -

Resumen

ACTEMAT es un desarrollo software que permite la representación de contenidos de carácter matemático a través de interfaces sonoras, estando destinado en particular a facilitar la accesibilidad a dichos contenidos de los usuarios con deficiencias visuales.

La accesibilidad de los deficientes visuales a los documentos electrónicos adquiere una importancia creciente en la legislación, en el mercado y en la oferta de software. En la actualidad existen herramientas que permiten la representación sonora de documentos electrónicos, tradicionalmente representados mediante elementos gráficos de texto. Sin embargo, apenas hay herramientas que interpreten adecuadamente las estructuras específicas del lenguaje matemático. Por ello, ACTEMAT se encarga de traducir a el l enguaje matemático a lenguaje natural junto con instrucciones para los sintetizadores de voz, de modo que la lectura final sea correcta.

ACTEMAT es capaz de procesar directamente documentos escritos en el lenguaje MathML, el cual ofrece ventajas como la capacidad para expresar la semántica del contenido o la facilidad de su extensión. No obstante, usando el filtro adecuado, se puede partir de documentos en otros lenguajes que se traduzcan a MathML. Como interfaz con el sintetizador de voz, se utiliza otro l enguaje XML (JSML o SABLE), lo que permite utilizar hojas XSL para llevar a cabo la traducción. Estas hojas XSL no son estáticas, sino que se generan a partir de un contexto definido en un lenguaje XML propio, en el cual quedan definidas las reglas que emplear para la traducción. La existencia de los ficheros de definición de contexto es fundamental para permitir la extensibilidad y personalización de las reglas de traducción a distintas disciplinas científicas y distintos idiomas. La herramienta proporciona además un contexto básico con las construcciones matemáticas más comunes.

El usuario habitual se limitará a solicitar a ACTEMAT que traduzca documentos. Además, existirá el diseñador de ficheros de definición de contextos. El lenguaje de los contextos se encuentra dividido en dos niveles: uno para definir tipos genéricos de estructuras matemáticas y otro para particularizarlas con estructuras concretas. En la mayoría de las ocasiones, se usará sólo la parte más sencilla de alto nivel; no obstante, un diseñador avanzado de contextos puede emplear también toda la potencia que le ofrece la parte de bajo nivel. Los contextos definidos pueden ser reutilizados gracias a dos características: la importación en cascada (y sombreado de definiciones) y la utilización de plantillas instanciables.

ACTEMAT está diseñado pensando en la extensibilidad y la reutilización, y así se prevé su uso junto a otros módulos para traducir lenguajes distintos de MathML, integrarlo como plug-in de un visor html+MathML, filtro bajo Web Services en servidores de programas educativos, etc. ACTEMAT supera a otras herramientas con fines semejantes al conjugar su versatilidad, el tratamiento semántico de expresiones matemáticas, la naturalidad de la lectura resultante y la capacidad de emplear distintos grados de complejidad en la definición de las reglas.

Introducción

Tradicionalmente, la comunicación de información de forma perdurable ha estado basada en la escritura gráfica: aquellas personas con deficiencias visuales se encontraban incapacitadas para transmitir y recibir información en un soporte físico sin la ayuda de un intérprete. Sólo en los últimos ciento cincuenta años se han desarrollado métodos de transmisión de información sonoros o táctiles (Braille), pero no han sustituido a la escritura gráfica, sino que su uso se limita a aquellos documentos que se decide codificar expresamente así. Con la informática personal, ha sucedido algo semejante, aunque referido a las interfaces de usuario: tradicionalmente han predominado las interfaces gráficas. No obstante, en este caso no es necesario reeditar cada uno de los documentos para facilitar su accesibilidad a invidentes, sino que basta con desarrollar interfaces de usuario adecuadas. Actualmente, la síntesis de voz y los lectores de pantalla son tecnologías maduras que permiten –junto a otras– la interacción con herramientas software por parte de discapacitados visuales y el acceso a documentos electrónicos. Sin embargo, existen áreas aún poco cubiertas, como es el acceso a documentos con contenido matemático. Esto se debe principalmente a dos factores:

- Es un aspecto concreto de la accesibilidad que sólo afecta a parte de los deficientes visuales que pudieran ser usuarios de ordenadores (sólo es relevante para aquellos que vayan a trabajar con texto matemático).
- Es un lenguaje que no sigue las reglas del texto ordinario (se escribe en dos dimensiones, no es lineal, y su traducción sonora no tiene siempre el mismo orden que la traducción gráfica).

La legislación vigente a nivel nacional (Ley 34/2002 “LSSICE”, Ley General de Telecomunicaciones) y europeo (Resoluciones ResAP(2001)1 y ResAP(2001)3 del Consejo de Europa, Resolución 2003/C 39/03 del Consejo de la Unión Europea) incluye menciones específicas respecto a la accesibilidad a documentos electrónicos y a herramientas informáticas, si bien es cierto que en la mayoría de las veces aún se habla de “promoción” y no de “obligación”. Es previsible que en los próximos años se vaya haciendo más rigurosa, siguiendo el modelo de la “Rehabilitation Act” estadounidense y haciendo que las normas técnicas sobre accesibilidad sean de obligado cumplimiento. Se calcula que un 8.6 por mil de la población española entre 6 y 64 años padece una deficiencia visual. Este segmento de la población es el que resulta más perjudicado por las dificultades en la accesibilidad no gráfica a textos científicos, ya que son los potenciales receptores de educación y usuarios profesionales.

En este contexto, queda justificada la necesidad de una herramienta que simplifique el acceso a textos con contenido matemático para las personas con deficiencias visuales. La eliminación de barreras para la inclusión social de los discapacitados incluye a las barreras de información. Así, la posibilidad de

acceder en igualdad de condiciones a textos matemáticos abriría a los deficientes visuales nuevos campos de formación, investigación y empleo, al facilitar la consulta y creación de documentos, así como el uso de herramientas software matemáticas y científicas. En especial, se verían fuertemente reforzados los programas que hicieran un uso intensivo de la tele-educación y el tele-trabajo, donde se alcanzaría prácticamente la igualdad entre videntes y deficientes visuales.

Por ello, surge el proyecto ACTEMAT (Accesibilidad a Textos Matemáticos), concebido para suplir las necesidades de acceso de la comunidad de deficientes visuales a los contenidos y a los servicios finales que incluyan textos matemáticos. Fundamentalmente, se trata de procesar los contenidos matemáticos, para enviárselos después con la forma adecuada a las herramientas actuales (lectores de pantalla, motores de síntesis de voz a partir de texto) con el fin de suplir las dificultades que tienen estas herramientas para pronunciar adecuadamente textos matemáticos. Además, se dota a nuestra herramienta de gran flexibilidad de configuración, de modo que la comunidad científica pueda ir adaptándola a distintos contextos. Por otra parte, aunque el público objetivo principal es el de deficientes visuales, nuestra herramienta se podría utilizar en general en cualquier interfaz sonora (conducción, teléfono, etc.).

Descripción

ACTEMAT es una herramienta que permite el acceso a texto matemático-científico mediante medios no visuales; en particular, sonoros. Aunque existen convenios para la representación táctil del lenguaje matemático (Nemeth Code), no existe aún una API estándar para comunicarse con dispositivos táctiles. Además, la representación sonora permite el acceso a todo tipo de usuarios (videntes o no), independientemente de que conozcan los mencionados códigos táctiles.

La herramienta en sí no se encarga de la representación sonora, sino que traduce el texto matemático a un conjunto de frases en lenguaje natural junto con comandos comprensibles por un gran abanico de motores TTS (Text To Speech). Existen varios lenguajes propietarios de distintos fabricantes, pero para garantizar la máxima compatibilidad, se debe escoger alguno de los lenguajes propuestos por diversos consorcios o aceptados de facto (JSML de Sun Microsystems, SSML del W3C o SABLE de Sun y la Universidad de Edimburgo entre otros). En nuestro caso, optamos por JSML, dado que facilita el uso de la tecnología Java, escogida por facilitar la operación multiplataforma. De todos modos, la traducción entre cualesquiera de ellos resulta trivial, y se podría redefinir fácilmente la herramienta para que los soportara.

En cuanto a los textos que la herramienta es capaz de procesar, se trata de aquellos codificados según el lenguaje MathML del World Wide Web Consortium (W3C). Sin embargo, una gran parte de los textos científicos actualmente disponibles, se encuentran en formato TeX, lo que parece indicar que esa debería haber sido la elección. No obstante, hay una serie de condicionantes que hacen decantarse por MathML:

- MathML es capaz de soportar semántica de los elementos matemáticos a la vez que reglas de presentación, mientras que en TeX se tratan habitualmente de la misma manera elementos con distinta semántica pero igual representación gráfica.
- MathML se puede emplear como salida de herramientas comerciales (Maple, Mathematica), manteniendo toda la semántica que permiten dichas herramientas.
- MathML es más regular que TeX y, por tanto, más fácil de procesar. Sin embargo, admite tanta extensibilidad como TeX, gracias al elemento semantics y al atributo definitionURL.
- MathML es XML, lo que facilita la utilización de tecnologías existentes (parsers de XML, XSLT) para su procesado, así como su integración en documentos HTML.
- MathML está concebido, según se indica en su especificación, para incluir a las interfaces audibles, si bien no logra este objetivo de la forma

que sería deseable, por lo que sigue haciendo falta una gran cantidad de procesamiento para transformarlo a lenguaje hablado.

Para traducir desde MathML a JSML, dado que ambos lenguajes son XML, la forma más natural es emplear transformaciones XSL, pues sólo hace necesario especificar las reglas traducción, sin necesidad de implementar el traductor en sí. Sin embargo, no basta con las transformaciones XSL, dado que un usuario puede preferir llevar a cabo las traducciones de distintas maneras:

- Dependiendo del campo de conocimiento que se trate, una misma expresión se puede leer de diferentes formas. Por ejemplo, el símbolo "ø" puede significar "conjunto vacío" en teoría de conjuntos o "diámetro" en geometría.
- Dependiendo del lenguaje del usuario, será necesario emplear distintas transformaciones de los símbolos matemáticos a las palabras del lenguaje natural.
- Finalmente, distintos usuarios pueden tener distintas preferencias respecto a la concisión con la que traducir al lenguaje natural (compárese "la integral con límite inferior cero y límite superior uno" con "la integral entre cero y uno").

Por ello, el usuario no indica directamente las traducciones XSL, sino que carga una serie de ficheros de contexto que contienen las reglas para las disciplinas matemáticas particulares que quiere emplear, a partir de los cuales se genera la XSL. El contenido de este contexto no es XSL, sino un lenguaje definido para el proyecto ACTEMAT, que permite simplificar la escritura por el usuario de reglas de traducción. Existen múltiples elementos que comparten las mismas características de traducción al lenguaje natural. Por ejemplo, los operadores ">", "<", "=", "≤", "≥", etc. son todos ellos, operadores relacionales binarios y las expresiones en las que intervienen sólo se diferencian en el nombre del operador. Por ello, resulta muy útil escribir una única plantilla para las reglas de traducción de todos los operadores relacionales binarios, la cual se instanciará en las reglas concretas para cada uno de los operadores, particularizando el elemento de MathML al que corresponden y el valor de la traducción de su nombre al lenguaje natural. Así, será necesario escribir una única plantilla para los operadores relacionales binarios, que tendrá en cuenta el orden de los argumentos, la forma verbal de estos (si fuera necesario), etc., mientras que el creador de las reglas no tiene más que indicar los valores de una serie de argumentos que la plantilla recibirá (en este caso, recibe el elemento MathML que se desea poder representar y nombre del operador). Dado que la herramienta proporciona un contexto básico con los principales tipos de expresiones que pueden aparecer en un documento matemático, en la mayoría de las ocasiones, la personalización se limitará a definir conjuntos de argumentos para instanciar una plantilla. No obstante, también se permite personalizar las plantillas, redefiniéndolas o añadiendo otras nuevas.

Lenguaje de los ficheros de contexto

Como se ha indicado, el contexto no está escrito directamente en XSL, sino en un lenguaje XML diseñado para el proyecto. Dentro de este lenguaje, cabe distinguir varios tipos de elementos:

- Existen elementos meramente descriptivos, que se limitan indicar una serie de parámetros y de plantillas que instanciar.
- Existen elementos diseñados para contener estructuras de código que sustituyen a las estructuras XSL. Uno de estos elementos son las plantillas mencionadas.
- Dentro de los contenedores de código, existen elementos de control que permiten generar distintas XSL a partir del mismo contenedor, en función de los parámetros que se escojan. Esto permite reutilizar las plantillas más allá de sustituir el nombre de un operador. Por ejemplo, se puede instanciar la misma plantilla para operadores cuyos operandos se leen en orden directo o inverso, sin más que hacer porciones del código condicionales a un argumento que puede tomar los valores “forward” o “reverse”. Esto es especialmente útil en los casos en los que distintos tipos de expresiones matemáticas comparten las mismas reglas traducción salvo detalles (el orden de una iteración, la condición de una bifurcación, ...). Estos elementos de “código de bajo nivel” se comportan de manera semejante a directivas de preprocesado, generando distintas XSL en distintas situaciones.
- En los contenedores de código, hay otros elementos que tienen una traducción unívoca a elementos XSL. Se trata de elementos de “código de alto nivel” que se emplean con el fin de simplificar la sintaxis de XSL, a la vez que permite una mayor potencia del código de bajo nivel (al generar condicionalmente el código de alto nivel en lugar del código XSL, se flexibiliza la cantidad de estructuras que resultan legales).

De acuerdo con esta división, existe una serie de procesados en cadena de nuestro lenguaje para generar la XSL final.

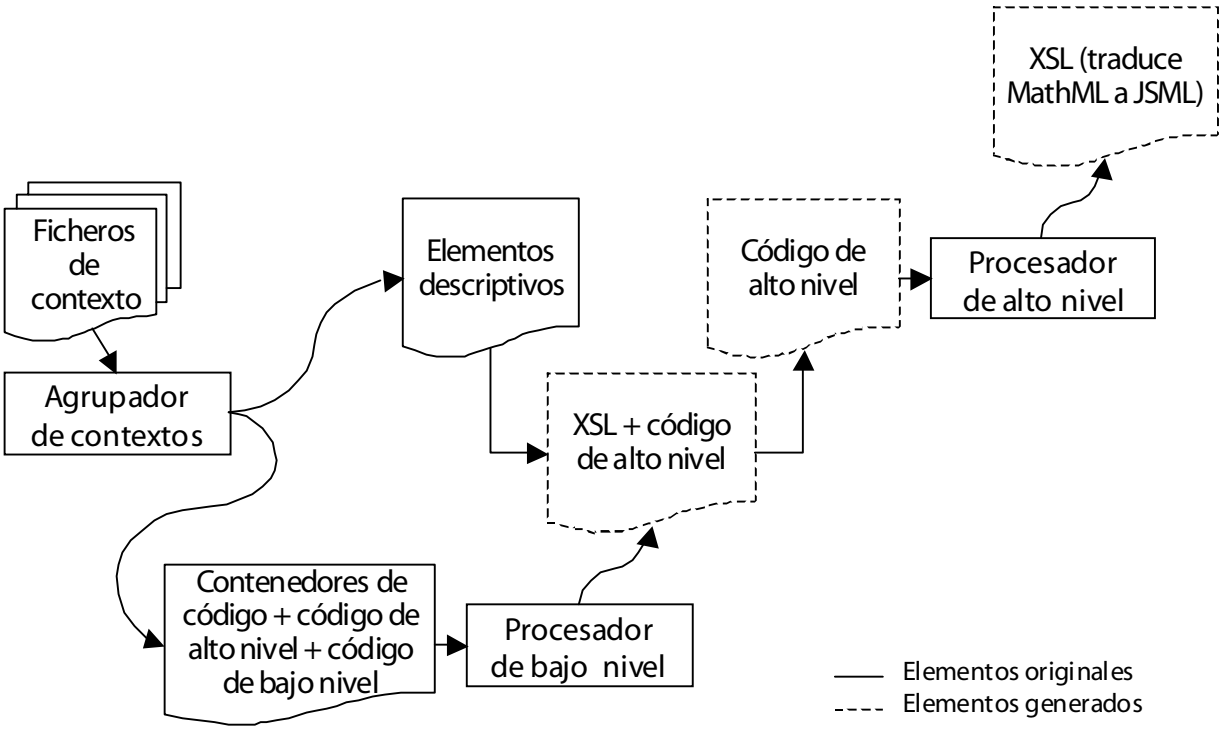

Figura 1. Diagrama de transformaciones en la generación del contexto

En primer lugar, el procesador de código de bajo nivel, traduce el código de bajo nivel a una XSL intermedia, manteniendo tal cual el código de alto nivel. A continuación, esta XSL toma los elementos descriptivos y genera con ellos código de alto nivel. Finalmente, el procesador de alto nivel genera la XSL que servirá para traducir de MathML a JSML. Se hace notar que, aunque este proceso es relativamente largo, no es necesario cada vez que se va a leer un documento, sino sólo cuando el usuario desea cargar nuevos contextos. Por otra parte, el usuario queda abstraído de la complejidad conceptual del proceso, ya que esto sólo le interesa al diseñador de contextos (y sólo en parte, pues el diseño de contextos se puede comprender sin necesidad de conocer el proceso que luego seguirán los contextos).

Adicionalmente, existe una serie de consideraciones que añadir al proceso:

- El contexto no viene definido de un único archivo, sino que es la suma de varios contextos (cada fichero de contexto puede importar a otros). Para ello, se usa una política de cascada, de modo que, si hay elementos, plantillas, etc. con el mismo nombre en el contexto importado y en el original, predomina el del original (importador). Así, es necesario primero reunir todos los ficheros en una sola estructura..
- El resultado final no es una única XSL, sino varias, dado que la transformación de MathML a JSML se tiene que llevar a cabo en varios pasos.

- MathML tiene particularidades que hacen que la representación pueda llegar a ser distinta para cada documento. Así, tenemos que elementos de MathML como “declare” obligan a añadir otro paso en la transformación final que no se conoce más que en el momento en el que se accede al documento.

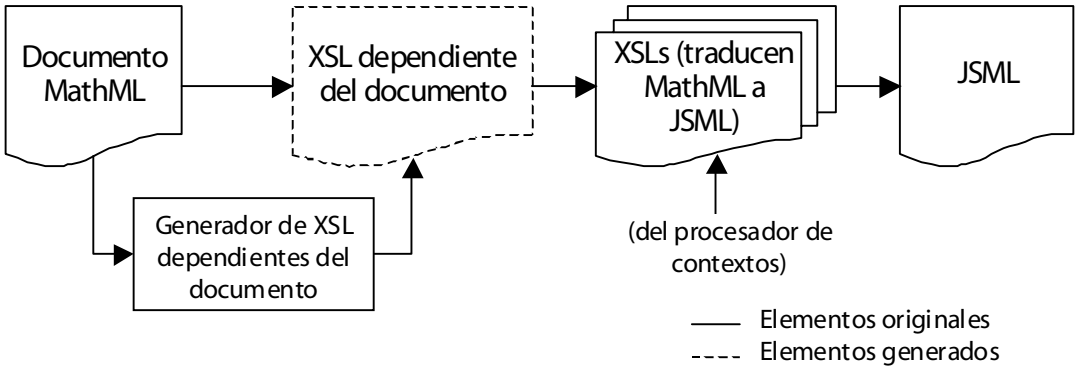

Figura 2. Diagrama de transformaciones del documento MathML a JSML

Estrategias de traducción

Como se ha indicado, la herramienta permite configurar para cada usuario la traducción de cualquier expresión, mediante la definición adecuada de los elementos del contexto. No obstante, la herramienta proporciona un contexto básico que cubre las principales expresiones matemáticas. Este contexto básico sirve para:

- Traducir documentos matemáticos con expresiones habituales, además de ofrecer una traducción por defecto para otro tipo de expresiones.
- Ampliar el contexto instanciando nuevos operadores definiendo sus reglas de traducción, siempre que sean de alguno de los tipos dados en el contexto (lo que sucederá en la mayoría de las ocasiones, ya que se cubren los principales tipos de expresiones y operadores).
- Tener un repertorio de plantillas ya creadas que sirvan de referencia a la hora de ampliar el contexto con nuevas plantillas para nuevos tipos de expresiones.

No se trata aquí de explicar las traducciones elegidas, pero sí se indicarán las líneas generales seguidas al escoger las reglas de traducción.

Los distintos tipos de **números** (enteros, reales de coma flotante, complejos, etc.) tienen sendas reglas de traducción. Pero la lógica principal de la traducción no se encuentra en esas reglas, sino en otras a las que se llama según el caso, encargadas de la lectura de las diversas **formas numerales** (cardinal, ordinal, fraccional, potencial, etc.). Estas reglas de formas numerales pueden ser llamadas por otras reglas que deseen leer un número de alguna forma determinada (por ejemplo, la regla para la operación de la potenciación).

Los **identificadores** y símbolos constantes se leen en general literalmente, salvo que se indique que tienen un nombre específico. Pueden admitir también un nombre corto y un nombre largo, en función de la posición dentro de una expresión. Además, dependiendo del **tipo** que tengan (vector, escalar, conjunto, elemento), y de las reglas de traducción para ese tipo en el contexto, los identificadores pueden leerse cualificados por la descripción del tipo (por ejemplo: “el vector x”).

Las **funciones** y operaciones vienen definidas en diversas reglas, para cada uno de los diversos tipos existentes. No obstante, hay un conjunto de reglas disponibles a las que se puede llamar para aplicar cuestiones comunes a todas las operaciones, como por ejemplo, la variación del tono y volumen al leer los operandos en función de su tamaño. Los criterios de traducción incluyen, entre otros, la multiplicidad de los operadores (n-ario, binario, unario) y la posición del operador (funcional, prefijo, infijo, postfijo). Existen también tipos propios para operadores del tipo integral, diferencial, cualificador, etc.

Los elementos de **presentación** tienen traducciones fijas si son elementos puramente de presentación (espacios, caracteres individuales); o bien siguen una interpretación heurística si se trata de las versiones de presentación de los elementos semánticos (operadores, identificadores, números) siguiendo los trabajos de T.V. Raman en su lector de LaTeX llamado ASTER.

Otros módulos

Hasta ahora se ha descrito el núcleo de la herramienta, que basta para tener la funcionalidad de traducir de MathML a JSML. No obstante, el objetivo global es el de poder acceder en general a un texto científico de forma sonora. Para ello, la herramienta irá añadiendo funcionalidad en el futuro, incorporando módulos externos ya existentes de libre distribución, algunos de los cuales son sustituibles por otras versiones que el usuario desee:

- El lenguaje de entrada no tiene por qué ser MathML. Existen herramientas capaces de traducir desde a MathML+HTML desde LaTeX, OpenMath, formatos nativos de herramientas comerciales, etc. A continuación se puede emplear un lector de HTML y nuestra herramienta como plug-in, o bien añadir alguna de las XSLs de libre distribución para traducir de HTML a JSML.
- La salida se envía a un sintetizador de voz que sea capaz de procesar JSML o SABLE (o SSML cuando su uso se generalice). Existen motores TTS de libre distribución (FreeTTS, Festival) pero es válido el sintetizador habitual que emplee el usuario.
- El usuario debe ser capaz de comunicar sus órdenes a la herramienta. Para ello, puede usar un reconocedor de órdenes de voz, o un lector de

menús y el teclado. Para permitirlo, la herramienta hace uso de las APIs de accesibilidad de Java.

Innovaciones y mejoras

Las ventajas que puede aportar nuestro traductor a sistemas similares se apoyan sobre todo en dos características fundamentales de su diseño:

- Uso de estándares
- Facilidad de configuración y adaptación al usuario.

El uso de lenguajes estándar para la entrada y salida del traductor (MathML y JSML o SABLE respectivamente), posibilita el poder utilizarlo junto con múltiples sistemas externos aumentando su funcionalidad. Por ejemplo, debido al uso de MathML, puede ser utilizado junto a aplicaciones comerciales de software matemático (como Maple o Mathematica) o editores matemáticos dentro de suites de edición (OpenOffice), acercando estos tipos de software a la comunidad de invidentes. A su vez, el uso de JSML (o SABLE) como salida posibilita el uso de multitud de sistemas sintetizadores de voz para la pronunciación. Teniendo en cuenta que la mayoría de invidentes que utilizan un PC ya suelen disponer de un TTS para diversos usos, el hecho de permitir numerosos sintetizadores para nuestro traductor permite que los usuarios utilicen su sintetizador habitual, facilitando así el acceso a nuestra herramienta, e incluso permitiendo la fácil localización del texto pronunciado cambiando las diferentes voces a diferentes idiomas.

Además de en los lenguajes de entrada y salida, también se han utilizado lenguajes estándar en la comunicación interna entre los diferentes módulos del traductor (XML y XSL), lo cual permite la utilización de los diferentes módulos por separado, haciendo muy fácil extender la funcionalidad general del sistema. Como ejemplo, se podría utilizar tan sólo el modulo de creación de contextos para la producción de hojas de estilo XSL (capaces de transformar MathML en JSML) y distribuirlas de tal manera que se pudiera procesar el MathML con sólo un procesador XSL estándar.

Otra posible ampliación del traductor debida a la utilización de estándares es su distribución como plug-in para un navegador web para invidentes. De esta manera se podría traducir de manera transparente al usuario todo tipo de documentos web que llevaran incluido algún tipo de texto matemático.

Llevando más allá la idea de generación de contextos de manera independiente a su posterior utilización, sería muy fácil integrar nuestro sistema con un repositorio de documentos en HTML + MathML colocado en un servidor con acceso a Internet mediante web services. Sistemas de este tipo son comunes en universidades virtuales (como ADL/SCORM, o incluso OCW del MIT). Mediante nuestro traductor podría crearse un filtro personalizado para usuarios invidentes, que sirviera directamente el contenido en JSML. La principal ventaja estaría en que para traducir todo el repositorio, no habría más

que crear una hoja de estilo y generar dinámicamente el JSML con cada petición, sin tener que traducir uno a uno los documentos.

En cuanto a la facilidad de adaptación al usuario, se ha dedicado un especial esfuerzo al sistema de creación de contextos resultando un sistema muy adaptable basado en 3 características principales:

- Potencia del lenguaje de creación de contextos: El lenguaje de creación de contextos tiene una potencia similar a un lenguaje de programación de alto nivel aunque con algunas limitaciones, dando un sinfín de posibilidades al creador del contexto. Por contra, esto hace que se requiera una cierta habilidad para crear un simple contexto. Para subsanar este inconveniente se añadieron las otras dos características al sistema de creación de contextos.
- Arquitectura por niveles: Debido a la existencia de dos niveles en los contextos (plantillas e instancias descriptivas) los creadores de contextos también se pueden dividir en dos niveles: aquellos que se dedicarán a crear plantillas de bajo nivel (conteniendo plantillas generales principalmente para operadores e identificadores) y aquellos que aprovechen estas plantillas para personalizar la pronunciación de algunos operadores o incluso añadir algunos nuevos basándose en plantillas existentes. La ventaja de este diseño es que sólo los creadores de contextos de bajo nivel deben conocer a fondo el lenguaje de programación de contextos, mientras que aquellos que sólo quieran personalizar levemente un contexto podrán aprovechar las plantillas existentes.
- Importación/inclusión de contextos en cascada: La principal utilidad de este sistema de importación/inclusión de contextos es que un usuario que sólo esté interesado en redefinir una pequeña parte de un contexto para su propio uso no debe ni crear ni rescribir uno ya existente. Tan sólo debe redefinir (en un archivo aparte) aquella parte que quiera modificar y después importar el módulo completo original. El sistema de importación asignará mayor prioridad al contexto del usuario y redefinirá las partes necesarias en el contexto original (el sistema es similar al sombreado en los sistemas orientados a objetos).

Sobre este sistema de creación de contextos se puede crear una jerarquía de archivos de contexto que permita la reutilización de contextos generales (sobre todo de bajo nivel) mientras que se permita fácilmente la "sobreescritura" de partes del contexto más proclives a la adaptación al usuario (como por ejemplo la pronunciación de las constantes en diferentes idiomas).

Se pueden comparar algunas de las ventajas que el proyecto ACTEMAT ofrece frente a otras soluciones existentes.

- ASTER fue uno de los primeros sistemas capaces de traducir documentos escritos en LaTeX a un lenguaje propio (AFL) que incluye

los términos del lenguaje natural y elementos de formato de audio. Sin embargo, sus capacidades de configuración son reducidas, requiriendo conocimientos de programación en LISP; en ningún caso permite definir nuevas familias de operadores. Al partir de LaTeX, no soporta interpretaciones semánticas de los contenidos. Se trata de una herramienta tradicional en el acceso a texto científico per o no aprovecha las actuales técnicas (tiene ya diez años).

- NIDE es un proyecto aún en fase de desarrollo para crear un conjunto de hojas de estilo XSL que permitan la traducción de MathML a un lenguaje de entrada de un sintetizador de voz. Aún se encuentra en fase de estudio y su objetivo se limita a crear las hojas XSL estáticas, en ningún caso existe una generación automática de XSL (no hay capacidad de configuración sin rescribir toda la hoja XSL, no hay reutilización de plantillas, se hace muy complejo el sombreado de reglas de traducción con nuevas reglas).
- MAVIS es un programa que incluye un proyecto de un lector de pantalla / navegador para matemáticas. El proyecto se encuentra parado desde hace años.
- MathTalk es una herramienta capaz de reconocer y leer texto matemático a través de una interfaz sonora. Tanto el reconocimiento como la lectura se efectúan basándose en la presentación y símbolo a símbolo. En ningún caso interpreta estructuras globales sino que lee símbolos individuales.
- Los lectores genéricos de pantalla (JAWS, MultiWeb, MicroTalk ASAW, etc.) no procesan por sí mismos el lenguaje matemático, dependiendo del sintetizador de voz sobre el que se apoyan. Estos normalmente tampoco leen adecuadamente el lenguaje matemático, si acaso se limitan a aplicar ciertas heurísticas para pronunciar expresiones aritméticas básicas.

Por último, cabe destacar que se trata de la primera solución desarrollada para la traducción al idioma español, sin que por ello se renuncie a la posibilidad de emplearlo en otros idiomas dada su gran adaptabilidad; así como la primera capaz de distinguir expresiones matemáticas por su semántica, en lugar de por sus características de presentación gráfica.

Conclusiones

ACTEMAT permite el acceso sonoro a textos científicos empleando un lenguaje muy próximo a cómo se leería naturalmente. Hasta ahora sólo se podía acceder de forma deficiente, o bien escuchando directamente la codificación en lenguajes de representación de contenido (TeX, SGMLs, XMLs) no concebidos para ser leídos directamente por un humano. La evolución del proyecto tratará de reducir al mínimo la intervención del usuario, siguiendo en estas tres líneas:

- Adición de un sintetizador de voz de libre distribución a la herramienta, de modo que, si el usuario lo desea, no sea necesario configurar la conectividad de ACTEMAT y el sintetizador.
- Integración en un visor de html como plug-in para traducción de contenidos científicos.
- Integración con APIs de accesibilidad a contenidos de pantalla, de modo que el intérprete de los elementos matemáticos en pantalla sea un módulo de conexión (plug-in) con ACTEMAT. Esto se irá desarrollando a medida vayan mejorando las APIs de accesibilidad de los sistemas operativos.

La gran versatilidad de la herramienta permitirá la ampliación de los contextos disponibles para el público a gran velocidad, añadiendo nuevos idiomas y nuevas expresiones; especialmente si se cuenta con la realimentación de los propios usuarios y los creadores de contextos.